

ABSTRACT

Title of Dissertation: DEEP LEARNING FOR SCENE PERCEPTION
AND UNDERSTANDING

Jun Wang
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Larry S. Davis
Department of Computer Science and
Professor Joseph F. JaJa
Department of Electrical and
Computer Engineering

The ability to accurately perceive objects and capture motion information from the environment is crucial in many real-world applications, including autonomous driving, augmented reality and robotics. This dissertation focuses on some fundamental challenges, regarding scene perception, scene understanding, and learning-based autonomous system.

We first address the problem of developing a good representation of 3D sensor data for solving scene perception tasks. We start by focusing on learning how to explore the environment of a 3D perception system, including accurately perceiving objects and understanding the motion of dynamic objects. For example, it is critical for robotic agents to be able to develop a good understanding of objects in their environment. We investigate and tackle this problem through different computer vision tasks using a variety of input data. Compared with images, 3D point clouds provide reliable depth and precise geometric information; however, they are gener-

ally sparse with varying densities. To handle these challenges, we present a number of methods for efficient object detection and motion learning in the case of large-scale LiDAR point cloud data. In the first part, we consider the problem of 3D point cloud density, not well-explored characteristic for the task of 3D object detection. Our proposed InfoFocus method improves detection by adaptively refining features guided by the information of point cloud density in an end-to-end manner. Inspired by the success of transformer-based architectures in a variety of computer vision tasks, we consequently present another method M3DETR, which unifies multiple point cloud representations, feature scales, as well as model mutual relationships between point clouds simultaneously using transformers for 3D object detection. We also consider the problem of understanding dynamic 3D environments and identifying motion information of objects, which is critical for 3D perception. In the third part, we focus on a temporal sequence of 3D point clouds to extract point-wise motion information. Specifically, we propose a point-based spatiotemporal pyramid architecture, PointMotionNet which handles multiple frames and large-scale scenes, avoids discretization and explicitly learns from the temporal ordering.

We note that having a deeper and holistic understanding of environment is quite important to help safely navigate through complex traffic scenarios. Besides accurately classifying, locating objects and predicting their behaviors, it would be crucial for the autonomous system to understand traffic rules of the road, such as spotting traffic signals or temporary road signs. The long-term goal is to build a perception system that has the ability to reason about the environment and adaptively make plans under uncertainty in real time. To reason and make real-time

adjustments, the system needs to be able to develop a good understanding of the road signs information. Here we address this task of Text-VQA which aims at answering questions that require understanding the textual cues in an image. In the fourth part of the thesis, we develop a method to generate high-quality and rich question-answer (QA) pairs by explicitly utilizing the existing rich text available in the scene context of the input image. The proposed architecture, TAG exploits underexplored scene text information and enhances scene understanding of Text-VQA models by producing meaningful, and accurate QA samples using a multimodal transformer. This method has the potential to be applied to identify challenging traffic situations that the autonomous vehicles will encounter on roads, such as traffic signs (stop/speed limit), one-way street, or evolving streets including road closure or a construction zone.

DEEP LEARNING FOR SCENE PERCEPTION AND UNDERSTANDING

by

Jun Wang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:

Professor Joseph F. JaJa, Chair/Advisor

Professor Larry S. Davis, Co-Chair/Co-Advisor

Professor Min Wu

Professor Furong Huang

Professor Yang Tao

© Copyright by
Jun Wang
2023

Dedication

This thesis is dedicated to my parents, Xiaolan Sun and Jinsheng Wang, as well as my girlfriend Yuqian Hu, for their love and support.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisors, Professor Larry Davis and Professor Joseph JaJa, for their generous support throughout my Ph.D. study and research. Their patience, motivation, enthusiasm, and immense knowledge have been invaluable to me. I am particularly grateful for Professor Davis' insightful advice "one thing at a time," which has helped me stay focused and productive, and for Professor JaJa's wise counsel to "prepare early," which has enabled me to tackle complex problems with confidence. I have learned so much from both of them and deeply appreciate their help and inspiration. Their knowledgeable guidance has been a constant source of support throughout the entire research and writing process of this thesis. I feel incredibly fortunate to have had such outstanding mentors for my Ph.D. study, and I am truly grateful for their unwavering support and encouragement.

I am deeply grateful to Professor Min Wu, Professor Furong Huang, and Professor Yang Tao for their invaluable contributions as members of my dissertation committee and for their kind advice during the preparation of my dissertation. I am also grateful to my collaborators and mentors, Mingfei Gao, Siheng Chen, Ran Xu, Kishore Prahallad, Shiyi Lan, Tianrui Guan, Xiaolong Li, Yangheng Zhao, Bo He, and Zuxuan Wu. Working with them has been an absolute pleasure, and I appreciate their guidance, advice, and the opportunities they have provided me. I would

also like to express my gratitude to all my colleagues in the computer vision lab, who have made my graduate study a truly colorful experience. Xintong Han, Xiyang Dai, Venkat Santhanam, Luyu Yang, Peng Zhou, Hengduo Li, Xitong Yang, Zhe Wu, Ruichi Yu, Guangxiao Zhang, Yuheng Lu, Pallabi Ghosh, and Bharat Singh - thank you for your support, friendship, and camaraderie. Your diverse perspectives and expertise have enriched my learning and research journey in countless ways, and I am truly grateful for the time we've spent together.

I would like to express my heartfelt gratitude to my dear friends at the University of Maryland - Vivian Lu, Chen Bai, Lei Chen, Jinjing Han, Chihuang Liu, Yu Jin, Han Zhou, Xiaomin Lin, Yuntao Liu, Melanie Prange, Emily Irwin, and Maria Hoo. Life with them during the past years has been an absolute pleasure, and I am deeply grateful for their kind and generous help in my everyday life. Their unwavering support, encouragement, and companionship have made my graduate journey all the more meaningful and memorable. I would also like to thank some old friends - Yuqiu Luo, Xin Yang, Fang Cao, Honglin Zhang, Yi Zhang, Yi Wang, and Shouyun Deng - for their understanding and support during the past years. Their friendship has been a constant source of comfort and strength to me, and I am truly grateful for the enduring bonds we share.

I am deeply indebted to my parents for their unwavering love and unconditional support throughout all these years. They have always been my rock, standing firmly behind me as a strong backup, and helping me navigate some of the most challenging times during my Ph.D. journey. Looking back, I realize that they had already given me all the important things I needed a long time ago. Their life lessons have now

become my lifetime treasures, and I am forever grateful for their guidance and wisdom. I would also like to express my gratitude to my girlfriend, Yuqian Hu, for her endless patience and unwavering understanding. Her unwavering support and encouragement have been a source of comfort and inspiration to me, and I cannot thank her enough for being a constant source of joy and positivity in my life.

Lastly, I would like to express my gratitude to all the people who have supported me and helped me navigate the challenges of pursuing my Ph.D. I am deeply thankful for their contributions to my success, and I could not have done it without them.

Table of Contents

Dedication	ii
Acknowledgements	iii
List of Tables	ix
List of Figures	xi
Chapter 1: Introduction	1
Chapter 2: InfoFocus: 3D Object Detection for Autonomous Driving with Dynamic Information Modeling	4
2.1 Introduction	4
2.2 Related Work	7
2.3 Proposed Approach	9
2.3.1 Deep Feature Extractor	9
2.3.2 Region Proposal Network (RPN)	10
2.3.3 InfoFocus	11
2.3.4 Loss Function	14
2.3.5 Comparisons with Existing Approaches	16
2.4 Experiments	17
2.4.1 Dataset and Evaluation Metric	18
2.4.2 Implementation Details	18
2.4.3 Main Results	22
2.4.4 Ablation Studies	23
2.4.5 Real-time Inference Analysis	26
2.5 Conclusion	27
Chapter 3: M3DETR: Multi-representation, Multi-scale, Mutual-relation 3D Object Detection with Transformers	28
3.1 Introduction	28
3.2 Related Work	33
3.3 Our Approach	35

3.3.1	Multi-Representation Feature Embeddings	36
3.3.2	Multi-representation, Multi-scale, and Mutual-relation Trans- former	38
3.3.3	Detection Heads Network	42
3.3.4	Loss Functions	44
3.4	Experiments	45
3.4.1	Datasets and Evaluation Metric	45
3.4.2	Implementation Details	46
3.4.3	Results	48
3.4.4	Ablation Studies	50
3.4.5	Robustness of M3DETR	51
3.5	Conclusion	52
Chapter 4: PointMotionNet: Point-Wise Motion Learning for Large-Scale LiDAR Point Clouds Sequences		53
4.1	Introduction	53
4.2	Related Works	56
4.3	PointMotionNet	59
4.3.1	4D Point Spatiotemporal Convolution	59
4.3.2	4D Point Pyramid Network Architecture	62
4.4	Motion-Related Systems	66
4.4.1	Point-based Motion Prediction	66
4.4.2	Multisweep Semantic Segmentation	68
4.5	Experiments Results	71
4.5.1	Point-based Motion Prediction	71
4.5.2	Multisweep 3D Semantic Segmentation	75
4.6	Conclusion	76
Chapter 5: TAG: Boosting Text-VQA via Text-aware Visual Question-answer Generation		77
5.1	Introduction	77
5.2	Related Work	80
5.2.1	Text-related VQA	80
5.2.2	Data Augmentation for VQA	81
5.3	Our Approach	82
5.3.1	Multi-modality Feature Embeddings	84
5.3.2	Multi-modality Fusion	85
5.3.3	Text-aware Visual Question Prediction	85
5.4	Experiments	87
5.4.1	Datasets and Evaluation Metrics	87
5.4.2	Implementation Details	88
5.4.3	Main Results	90
5.4.4	Ablation Studies	92
5.5	Conclusion	94

Chapter 6: Conclusions	95
Bibliography	97

List of Tables

2.1	Object detection results (%) on nuScenes validation set	22
2.2	Object detection results (%) on nuScenes test set. Note that MonoDIS and CenterNet are camera based methods, and the rest are LiDAR based. Ours 2× indicates 2× training time with other settings being the same with Ours	23
2.3	Ablation studies on nuScenes validation set. "Vis. Att" and "Adp. Att." refer to Visibility Attention Module and Adaptive Attention Module, respectively	23
2.4	Inference time of 3D object detectors. Note that inference time for the baseline here is the network reproduced by ourselves	25
2.5	Comparison with rotated RoIAlign feature extraction results (%) on the nuScenes validation set	26
3.1	M3DETR outperforms in the Vehicle, Pedestrian and Cyclist classes for both LEVEL_1 and LEVEL_2 difficulty levels on Waymo Open Dataset test set. Note that PV-RCNN* is our reproduced results with single point cloud frame input.	43
3.2	M3DETR outperforms in the Vehicle class with different size range on Waymo Open Dataset test set. Note that PV-RCNN* is our reproduced results with single point cloud frame input. We underscore the second best method in each column for comparison.	43
3.3	M3DETR outperforms in the Vehicle class with IoU threshold of 0.7 on the full 202 Waymo Validation Set, especially on the far range (50m to Inf). We underscore the second best method in each column for comparison.	44
3.4	M3DETR outperforms in both Car and Cyclist classes for 3D detection benchmark on KITTI Test Set. We underscore the second best method in each column for comparison.	48
3.5	Ablation studies of transformers in Car class with IoU threshold of 0.7 on KITTI Validation dataset. "Rel. Trans." and "Rep. and Scal. Trans." refer to mutual-relation trans-former of 2 MHSA layer with 4 heads, and multi-representation and multi-scale transformer of 1 MHSA layers with 8 heads, respectively.	49
4.1	PointMotionNet outperforms in the motion prediction task on Argoverse Validation Set.	67

4.2	PointMotionNet outperforms in motion state estimation on SemanticKITTI. RangeNet++ and MotionNet are in the range view and BEV, respectively, while the rest are point-based.	67
4.3	PointMotionNet outperforms in multisweep semantic segmentation on SemanticKITTI test set. Note that S represents the static foreground category, and M denotes the moving foreground category. . .	69
4.4	Ablation Study of PointMotionNet on Argoverse Validation Set. We vary several hyper-parameters: number of proxy points k , initial sampling resolution d_1 , temporal operation, and input frames number n	70
4.5	Additional Ablation Study of PointMotionNet on Argoverse Validation Set. We further test different number of proxy points k	74
5.1	TAG's outperformance on the TextVQA dataset when trained on original and augmented dataset under two settings. Note that M4C [†] is the improved version from [1].	89
5.2	Our framework outperforms prior work on the ST-VQA dataset. Note that M4C [†] is the improved version from [1]. Specifically, our model with TextVQA outperforms the SOTA approach TAP ^{††} [1] that is pre-trained with extra large-scale data from external TextCaps [2] and OCR-CC [1] datasets.	90
5.3	Hyper-parameters of TAG.	91
5.4	Ablation study of TAG with TAP [1] under constrained setting on TextVQA validation set. "Ans.", "Obj." and "OCR." refer to embedding of answer words, detected objects and OCR tokens, respectively.	92
5.5	Ablation study of TAG with TAP [1] under constrained setting on TextVQA validation set. <i>Random</i> means a random OCR token is selected as the answer input to TAG, while <i>top three</i> means the top three largest OCR tokens are selected.	93

List of Figures

2.1	Left: we calculate the average point density across different parts of objects in BEV of nuScenes training set. $E1-E4$ indicate four edges sorted by their normalized density scores (sum to 100%) and <i>others</i> denotes areas inside objects. We set each edge width as 10% of the length along the object size and only objects over more than 100 points are counted. Middle and right: we visualize an example of the LiDAR point cloud in 2D image and its corresponding bird's eye view (BEV). Clearly, most of the point clouds locate on the contour of the object	6
2.2	The proposed 3D object detection framework. It consists of three parts: Deep Feature Extractor(DFE), Region Proposal Network, and InfoFocus. InfoFocus contains three modules: PoI Pooling, Visibility Attentive Module, and Adaptive Point-wise Attention Module	10
2.3	RoI Pooling <i>vs.</i> PoI Pooling. The grid represents the feature map, and the dots denotes sampling points of interest. RoI Pooling samples the whole box, while PoI Pooling focuses on the key-points from edge-of-interest	13
2.4	Left: illustration of the Visibility Attentive Module. We compute hard attention for each sampled point depending on whether it is visible to the sensor. We also show the visibility map on the bottom left. Points on the blue line are visible while points on the orange line are invisible. Right: the architecture of the Adaptive Point-wise Attention Module. The point-wise attention is generated using a fully connected (FC) layer followed by a Sigmoid function. The input of FC layer is the feature of each point	16
2.5	We visualize the detection results on nuScenes with 2D and 3D BEV images. On the top, we demonstrate the 2D images with the 3D bounding box annotated, while the BEV of LiDAR with ground truth (red) and detection (blue) box are shown on the bottom. Note that the line in the frame denotes the direction of the object	25
2.6	We visualize the BEV detection results for the same point cloud sample on nuScenes with the vanillar PointPillars (left) and InfoFocus (right)	27

3.1	M3 Transformers used in our approach: Left: Multi-representation transformers; Middle Part: Multi-scale transformers; Right: Mutual-relation transformers. We use these characteristics to present a novel 3D object detection architecture.	29
3.2	An overview of our M3DETR architecture: M3DETR is a transformer based framework for object detection in a coarse-to-fine manner. It consists of three parts. PointNets, VoxelNet, and 2D ConvNets modules enable individual multi-representation feature learning. M3 Transformers enable inter-intra multi-representation, multi-scale, and multi-location feature attention. With the Region Proposal Network (RPN), the initial box proposals are generated. R-CNN captures and refines region-wise feature representations from M3 transformer output to improve detection performance.	30
3.3	M3 Transformers consist of two parts: a multi-representation and multi-scale transformer, and a mutual-relation transformer. Multi-representation and multi-scale transformer takes the different feature embedding and generates enriched cross-representations and cross-scales embedding. On top of it, the mutual-relation transformer further models point-wise feature relationship to extract the refined features.	31
3.4	We highlight the 3D detection results of M3DETR on the Waymo Open Dataset. The 3D ground truth bounding boxes are in green, while the detection bounding box are shown in blue.	46
3.5	We visualize the 3D detection results for the same input point cloud between PV-RCNN (left) and M3DETR (right) on Waymo Open Dataset. We highlight the false negative boxes from PV-RCNN in red.	50
3.6	Performance comparison of different M3 Transformers variants in Car class on KITTI validation set. We show the mAP results with IoU threshold of 0.7 for both 11 and 40 recall positions. Note that "l" and "h" represent layer number and head dimension of M3 Transformers, respectively. "Top" denotes the number of proposals used for keypoint sampling from RPN stage.	51
4.1	PointMotionNet is a point-based spatiotemporal pyramid network to achieve motion learning from a sequence of 3D point clouds. It can handle multiple frames, large-scale open scenes, explicitly model the temporal ordering, and does not suffer from discretization error. . . .	54

4.2	4D point spatiotemporal convolution operation (Point-STC). To extract spatiotemporal features, we gather all points across time to the same spatial coordinate system. For each query point (with red circle), we find its neighbors (in the orange ball) in each frame. With the set of proxy points (in red triangle) of the query point, we extract spatial features in each individual frame, which follows from (4.1a). We next concatenate the spatial features across all timestamps and use a MLP to obtain the final spatiotemporal features, which follows from (4.1b).	58
4.3	PointMotionNet is a point-based spatiotemporal pyramid network. Point-STC layers enable spatiotemporal feature learning. Downsampling and upsampling layers enable multiscale feature learning. With the skip connections, the aggregation layers fuse features from both encoder and decoder.	63
4.4	Point-based motion prediction system. With PointMotionNet as a backbone, the system takes a sequence of 3D point clouds as the input and generates the predicted motion vector and the estimated motion state for each single 3D point in the current frame through the motion vector head and the motion state head. For visualization, we sample the moving points for the motion vector output.	64
4.5	Multisweep semantic segmentation system. It consists of four parts: the segmentation backbone, extracting semantic features from each individual 3D point cloud, the segmentation head, producing the estimated semantic labels, PointMotionNet, extracting motion features based on semantic features across time stamps, and the motion state head, producing the estimated motion states.	64
4.6	PointMotionNet (right) qualitatively outperforms FlowNet3D (middle) in point-based motion prediction on Argoverse. Moving points are associated with green motion vectors and the static points are in red; the remaining points are background, marked in grey. . . .	68
4.7	PointMotionNet (right) visually outperforms MeteorNet (middle) in motion state estimation on SemanticKITTI. For six foreground classes, moving points are in green and static ones are in red; for the remaining background classes, points are in grey.	70
5.1	Left: statistics of the numbers of annotated QA pairs (blue) and scene text words (orange) for each image in the TextVQA training set [3]. Clearly, the scene text words are not fully leveraged in the annotations. Right: an example of a training image with more than 5 scene text words, which is typical. Best viewed in color.	78

5.2	The proposed Text-VQA framework. It consists of two parts: a text-aware visual question-answer generation module (TAG), followed by a downstream Text-VQA model. TAG is based on a multi-modal transformer architecture, which takes a single image and text words (the answer) as input, and outputs a newly-generated question corresponding to the input answer. The generated QA pairs from TAG together with the originally labeled data are subsequently used to train Text-VQA models, leading to better Text-VQA performance. . .	82
5.3	The illustration of our proposed TAG. High-dimensional feature representations are first extracted for three modalities, including extended answer words, visual objects, and scene text. Then, a multi-modal transformer is used to model the interactions of different modalities. Finally, a decoding module is used to predict the question corresponding to the answer through iterative decoding with an auto-regressive mechanism. Left: Originally labeled QA pairs are used for training. Right: During inference, detected OCR words are used as a novel answer to generate a question. Best viewed in color.	83
5.4	We visualize the examples of the generated QA pairs (bottom in orange) by TAG module compared with the original annotated QA pairs (top in blue) on the TextVQA training set. "Q" and "A" refer to question and answer, respectively. Best viewed in color.	88

Chapter 1: Introduction

We live in a three dimensional world. In a variety of applications, such as autonomous driving, and robotic exploration, the agents need to understand the 3D environment around it. While a number of sensor data types can be used, the LiDAR point cloud provides accurate geometry information. Moreover, LiDAR sensors have become more available on commercial mobile devices. Compared with images generated by cameras, 3D point clouds provide more reliable depth information; however, they are generally sparse with varying densities. To handle these challenges, we present methods for efficient object detection using large-scale LiDAR point cloud.

We also live in a dynamic world. Many recognition tasks, such as estimating the acceleration of a moving object, have benefited from sequences of temporal data. In particular, we would like the agents to learn representations from dynamic 3D point cloud sequences. Learned representation can be used in downstream tasks, such as semantic segmentation, and motion learning scene flow estimation.

Chapter 2 based on our work in [4] addresses the problem of efficiently using 3D point cloud data with varying densities, not a well-explored characteristic of point cloud data for the task of 3D object detection. The goal of 3D object detection is to localize and recognize objects in the 3D scene. The existing methods model

the input space with features extracted from equally divided sub-regions without considering the fact that point cloud is generally non-uniformly distributed over the space. We propose a novel 3D object detection framework with dynamic information modeling. The proposed framework is designed to make predictions progressively in a coarse-to-fine manner. Coarse predictions are generated during the first stage via a voxel-based region proposal network. Then our method, called InfoFocus, improves the coarse detections by adaptively refining features guided by the information of point cloud density. This results in a superior performance of our method compared to previously published works.

In Chapter 3 based our work in [5], we extend the technique of Chapter 2 by presenting a novel architecture for 3D object detection, M3DETR, which combines different point cloud representations (raw, voxels, bird-eye view) with different feature scales based on multi-scale feature pyramids. M3DETR is the first approach that unifies multiple point cloud representations, feature scales, as well as models mutual relationships between point clouds simultaneously using transformers. Specifically, we propose two stacked transformer encoder layers called M3 Transformers, where the first transformer integrates features with different scales and representations, and the second transformer aggregates information from the keypoints.

Chapter 4 focuses on learning motion information from a sequence of large-scale 3D point clouds, based our work in [6]. We propose a point-based spatiotemporal pyramid architecture, called PointMotionNet. A core component of PointMotionNet is a novel technique for point-based spatiotemporal convolution, which finds

the point correspondences across time by leveraging a time-invariant spatial neighboring space and extracts spatiotemporal features. To validate PointMotionNet, we consider two motion-related tasks: point-based motion prediction and multi-sweep semantic segmentation. For each task, we design an end-to-end system where PointMotionNet is the core block that learns motion information.

In Chapter 5 based our work in [7], we propose TAG, the first generic text-aware question-answer generation approach for Text-related Visual Question Answering (VQA). Text-VQA aims at answering questions that require understanding the textual cues in an image. Despite the great progress of existing Text-VQA methods, their performance suffers from insufficient human-labeled question-answer (QA) pairs. However, we observe that, in general, the scene text is not fully exploited in the existing datasets – only a small portion of the text in each image participates in the annotated QA activities. This results in a huge waste of useful information. To address this deficiency, we develop a new method to generate high-quality and diverse QA pairs by explicitly utilizing the existing rich text available in the scene context of each image. Specifically, TAG is a text-aware visual question-answer generation architecture that learns to produce meaningful, and accurate QA samples using a multimodal transformer. The architecture exploits underexplored scene text information and enhances scene understanding of Text-VQA models by combining the generated QA pairs with the initial training data.

Finally, we conclude in Chapter 6 by outlining some future research directions.

Chapter 2: InfoFocus: 3D Object Detection for Autonomous Driving with Dynamic Information Modeling

2.1 Introduction

With growing interests in autonomous vehicles, 3D object detection has received considerable attention. Due to the superior capability of modeling 3D objects, point cloud is the most popular type of data source. Most existing 3D detectors are point-based [8, 9, 10, 11, 12] and voxel-based [13, 14, 15, 16, 17]. Point-based approaches generate features from raw point cloud data directly. Although achieving promising performance, these methods suffer from high computational complexity which discourages their application in real-time scenarios. Voxel-based approaches [13, 14, 15, 16, 17] firstly convert point cloud into voxels and then employ deep convolutional neural networks (DCNN) to conduct object detection. Taking advantage of the advanced DCNN architecture, voxel-based approaches achieve the state-of-the-art performance with low computational cost. Our work follows the setting of voxel-based methods for their advanced balance of efficiency and effectiveness.

Although much progress has been made in improving the performance of voxel-

based detectors, an important characteristic of point cloud is not well explored: input data points are usually not uniformly distributed over the space. The density of point cloud can be affected by different factors, e.g., the distance of objects from LiDAR sensor and object self-occlusion. As illustrated in Fig. 2.1, the density of point cloud over objects highly depends on the relative locations of different parts. It is also intuitive that the amount of information is highly related to the point density. However, existing voxel-based detectors extract features from uniformly divided sub-regions, regardless of the actual distribution of the points. We believe that this will lead to loss of useful information and ultimately result in sub-optimal detection performance.

To fully exploit the non-uniform distribution of point cloud, we propose a novel 3D object detection framework, to adaptively model the rich feature of 3D objects according to the information density of points. Illustrated in Fig. 2.2, our framework contains two stages. Coarse detection results are obtained in the first stage via voxel-based region proposal network. In the second stage, we introduce InfoFocus, to model and extract the informative features from regions of interest (RoI) (formed by the coarse predictions) according to the distribution of point cloud, and the predictions are improved with the help of the refined features.

The InfoFocus is the core structure of our framework which contains three sequentially connected modules including the Point-of-interest (PoI) Pooling, the Visibility Attentive Module, and the Adaptive Point-wise Attention.

PoI Pooling. Unlike 2D objects which contain densely distributed information over the whole RoI, more of the points of 3D objects locate on the their



Figure 2.1: Left: we calculate the average point density across different parts of objects in BEV of nuScenes training set. $E1-E4$ indicate four edges sorted by their normalized density scores (sum to 100%) and *others* denotes areas inside objects. We set each edge width as 10% of the length along the object size and only objects over more than 100 points are counted. Middle and right: we visualize an example of the LiDAR point cloud in 2D image and its corresponding bird’s eye view (BEV). Clearly, most of the point clouds locate on the contour of the object

surfaces. Therefore, we hypothesize that most informative feature concentrates on the edge of RoI. Motivated by this intuition, we propose PoI Pooling which densely samples features on the edge and sparsely samples feature in the middle of RoI to accommodate the non-uniform information distribution of point cloud.

Visibility Attentive Module. Heavy self-occlusion is presented because of the nature of LiDAR data that is no point cloud exists on the backside of object relatively to the sensor. To mitigate this issue, our proposed Visibility Attentive Module applies hard attention to emphasize the visible parts of objects and eliminate the features from invisible points.

Adaptive Point-wise Attention. PoIs may contain different amount of information, although they are all visible. We introduce Adaptive Point-wise Attention to re-weight the features to improve the modeling of 3D objects.

We conduct extensive experiments on the largest public 3D object detection

benchmark, i.e, nuScenes [18]. Experimental results show that our approach significantly outperforms the baselines, achieving 39.5% mAP with 31 FPS. Results of comprehensive ablation studies demonstrate the effectiveness of our InfoFocus and that each sub-module makes considerable contributions to our framework.

2.2 Related Work

Point-based Detectors. Inspired by the powerful feature learning capability of PointNet [19, 20] and the advanced modeling structure of 2D object detectors [21, 22, 23], Frustum PointNets [8] extrude the 2D object proposals into frustums to generate the 3D bounding boxes from raw point cloud. Lan et al. [10] add a decomposition-aggregation module modeling local geometry to extract the global feature descriptor of point cloud. Limited by initial 2D box proposals, those methods yield low performance when objects are occluded. In contrast, PointRCNN [11] generates 3D proposals directly from point cloud instead of 2D images. The recent STD [12] attempts to refine the detection boxes in a coarse-to-fine manner. However, all those methods are computationally expensive due to the large amount of data points to be processed.

Multi-view 3D Detectors. MV3D [24] is proposed to fuse multi-view feature maps for the generation of 3D box proposals. Following [24], Ku et al. [25] explore high resolution feature maps to compensate the information loss for small objects. These methods address the feature alignment between multi-modality in a coarse level and are typically slow. Liang et al. [26] design a continuous fusion layer to

deal with the continuous state of LiDAR and the discrete state of images. Later, [27, 28] leverage different strategies to jointly fuse related tasks to improve feature representation.

Voxel-based Detectors. Recently, there is a trend of using regular 3D voxel grids to represent point cloud such that the input data can be easily processed by the 3D/2D convolution networks. Among those, VoxelNet [14] is the pioneering work of performing voxelization on the raw 3D point cloud. To improve its efficiency, Second [15] adopts Sparse Convolution and speeds up detection process without compromising the detection accuracy. PointPillars [13] dynamically converts the 3D point cloud into a 2D pseudo image, making it more suitable for the application of the existing 2D object detection techniques. In [16], Ye et al. design a new voxel generator to preserve the information loss along the vertical direction. Building upon voxel-based detectors, our model captures richer information of objects by refining their feature representations at a second stage guided by the point cloud density and ultimately improves the detection results.

There are several recent studies [29, 30] focusing on fusing the voxel-based features with PointNet-based features in order to extract more fine-grained 3D features. InfoFocus is complementary to these techniques and can be further applied on top of them. WYSIWYG [17] is the most related method to our approach since we both drive the model to encode visibility information. However, instead of using a separate branch to generate the hidden invisibility representation, our method directly aggregates the valuable point-wise features together from existing backbone network to refine the proposals in an end-to-end manner.

2.3 Proposed Approach

The proposed framework is illustrated in Fig. 2.2, which consists of a deep feature extractor followed by a two-stage architecture. The deep feature extractor containing a Pillar Feature Network and a DCNN, converts the input point cloud to representative feature maps. Specifically, the Pillar Feature Network divides the whole space into equal pillars and generates the so-called pseudo images [13]. The pseudo images are then processed by the DCNN to obtain the feature maps which are shared by the two stages, i.e., Region Proposal Network (RPN) and InfoFocus. The RPN generates the initial coarse bounding box proposals that are refined by InfoFocus, with dynamic information modeling. Note that our Deep Feature Extractor and RPN follow the setting of [13].

2.3.1 Deep Feature Extractor

Deep Feature Extractor is composed of two parts: 1) voxelization using Pillar Feature Network that converts the orderless point cloud into a sparse pseudo image via a simplified PointNet-like architecture and 2) feature extraction using DCNN to learn informative feature maps.

Pillar Feature Network. The Pillar Feature Network operates on the raw point cloud, and learns point-wise features for each pillar. After voxelizing raw point cloud into evenly spaced pillars, we randomly sample N points from each non-empty pillar and then obtain a dense tensor with the size of $D \times P \times N$, where D indicates the information dimension of each point, P denotes the number of non-empty pillars,

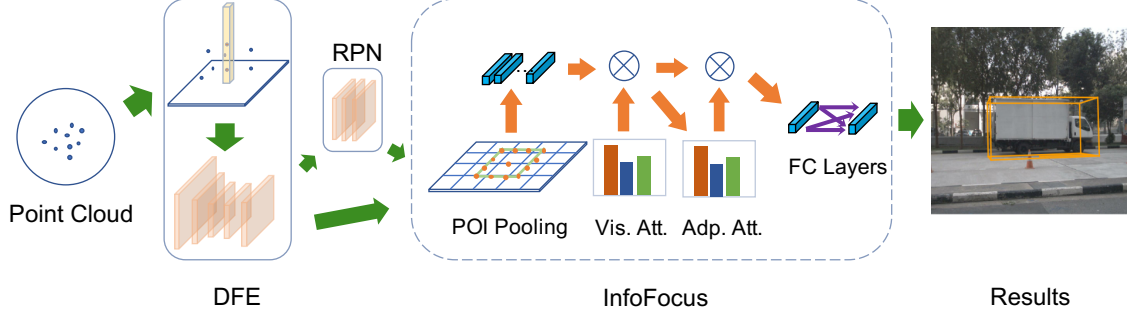


Figure 2.2: The proposed 3D object detection framework. It consists of three parts: Deep Feature Extractor(DFE), Region Proposal Network, and InfoFocus. InfoFocus contains three modules: PoI Pooling, Visibility Attentive Module, and Adaptive Point-wise Attention Module

and N denotes the number of points in each pillar. The Pillar Feature Network utilizes a PointNet-like block to learn a multi-dimensional feature vector for each pillar. The pillar-wise features are encoded into a 2D pseudo image with the shape of $W \times L \times C$, where W and L indicate the width and length of the pseudo image, and C is the channel of the feature map.

Deep Convolution Neural Network (DCNN). DCNN learns feature maps from the generated pseudo 2D image. The DCNN uses conv-deconv layers to extract features of different levels, and concatenates them to get the final features from different strides.

2.3.2 Region Proposal Network (RPN)

The RPN takes the feature maps provided by DCNN as inputs and produces high-quality 3D object proposals. Similar to the proposal generation in 2D object detection, anchor boxes are predefined at each position and proposals are generated by learning the offsets between anchors and the ground truths. To handle different

scales of objects, a dual-head strategy is adopted. Specifically, the small-scale head takes features from the first conv-deconv phase of the DCNN, while the large-scale head takes the features from its concatenation phase.

2.3.3 InfoFocus

The InfoFocus serves as the second stage of our framework, which takes the candidate proposals from RPN and extracts features of objects in a hierarchical manner from the feature maps produced by the DCNN. Specifically, given each 3D object proposal, InfoFocus dynamically focuses on the informative parts of the feature maps by gradually emphasizing the representative PoIs in the following three steps: 1) the edge points are selected out from the whole proposal region by PoI Pooling; 2) Visibility Attention module emphasizes on the informative points according to their relative visibility to the LiDAR sensor and 3) in the Adaptive Point-wise Attention module, the features of the visible points are further weighted adaptively. The re-weighted features of the visible points are then fused to form the final representation of the proposal, on top of which two fully-connected layers are utilized to predict the refined box.

PoI Pooling. When representing a 3D proposal, the most intuitive way is adopting the commonly used strategy in the two-stage 2D object detectors, i.e., RoI Pooling (see Fig. 2.3 left). However, unlike the 2D images that have densely distributed information over the region proposals, the 3D point cloud mostly resides on the object surface which results in non-uniform information over the regions (most information

locates on the edges of proposals).

The proposed PoI pooling is illustrated in Fig. 2.3 (right). Instead of equally sampling points over a region of the feature maps, we focus on sampling the points on the informative parts including four corners, the center point and key-points on the edges. Note that we consider the center position as an additional useful signal since it is likely to capture the semantic-level information.

We first project the 3D proposal to the birds' view coordinate system. Let p_0, p_1, p_2 and p_3 represent the positions of top-left, top-right, bottom-right, and bottom-left corners of a proposal on the pseudo image, respectively and p_c denotes the center point. Along each edge, n more key-points are uniformly sampled. For example, for the top edge between p_0 and p_1 , the position of a sampled key-point $kp_j = (p_0 \frac{j}{n+1} + p_1 \frac{n+1-j}{n+1})$, where j is an integer and $1 \leq j \leq n$. To this end, $(5 + 4 * n)$ PoIs are obtained. A high-dimensional feature is extracted for each PoI according to its relative position on the feature map and then we obtain a feature set $F_{poi} = \{f_1^{poi}, f_2^{poi}, \dots, f_{N_{poi}}^{poi}\}$, where $N_{poi} = (5 + 4 * n)$ representing the number of selected PoIs within the considered region.

Visibility Attentive Module. Severe self-occlusion typically occurs in point cloud, but is ignored by most of the existing methods. The Visibility Attentive Module (see Fig. 2.4 left) is proposed to mitigate this issue by focusing on the information provided by the visible parts of objects. We argue that visible regions contain more useful information than the occluded ones. Formally, we propose to re-weight features of PoIs according to their corresponding visibility by exploiting the geometric relationship between the proposals and the LiDAR sensor in bird's

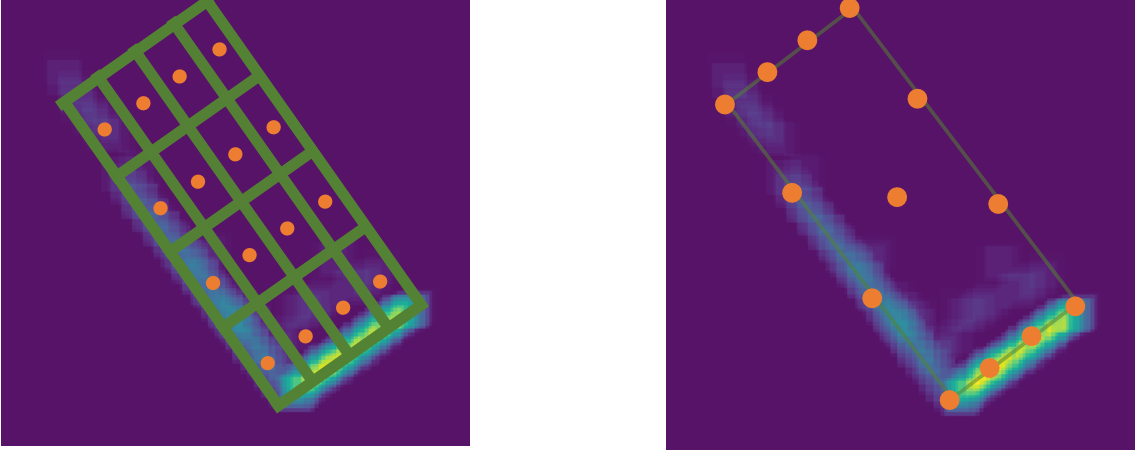


Figure 2.3: RoI Pooling *vs.* PoI Pooling. The grid represents the feature map, and the dots denotes sampling points of interest. RoI Pooling samples the whole box, while PoI Pooling focuses on the key-points from edge-of-interest

eye view. As shown in Eq. 2.1, F_{vis} denotes the updated feature set, where v_i^{poi} indicates the visibility score of the i th PoI. Different weighting strategies can be used and we use a hard attention strategy in this work for its simplicity, that is assigning $v_i^{poi} = 1$ if the i th PoI is visible and $v_i^{poi} = 0$ otherwise. In other words, we only take PoIs on the visible edges to represent the proposal.

$$F_{vis} = \{f_1^{poi} * v_1^{poi}, f_2^{poi} * v_2^{poi}, \dots, f_{N_{poi}}^{poi} * v_{N_{poi}}^{poi}\} \quad (2.1)$$

For the consideration of model efficiency, a simple yet effective method is used to estimate the visibility of points in the bird’s eye view. To figure out the sides of proposals facing to the sensor, we first compute the distance of each corner to the LiDAR sensor and determine the one that is closest to the sensor. Then, we consider the two edges passing this closest corner as the visible edges and the other two as the occluded ones.

Adaptive Point-wise Attention Module. PoI Pooling and Visibility Attentive

Module are motivated by the nature of the non-uniform density of point cloud. However, two points may offer different amount of information even though they are all visible by the sensor. Adaptive Point-wise Attention Module provides the flexibility for the visible PoIs to contribute unequally to the prediction. Suppose $F_{vis} = \{f_1^{vis}, f_2^{vis}, \dots, f_{N_{vis}}^{vis}\}$ indicates the feature set of visible PoIs. Adaptive Point-wise Attention Module learns an attention weight, w_i , for each f_i^{vis} adaptively for the next-step feature aggregation. Specifically, a shared fully connected (FC) layer with sigmoid as the activation function is used to learn the attention weights, formally expressed as $v_i^{vis} = \text{Sigmoid}(\mathbf{W}f_i^{vis} + b)$. We use $F_{att} = \{f_1^{att}, f_2^{att}, \dots, f_{N_{vis}}^{att}\}$ to represent the re-weighted feature set of visible PoIs updated using F_{vis} and the attention weights, where $f_i^{att} = f_i^{vis} * v_i^{vis}$.

The final representation of each proposal aggregates the features of its visible PoIs. Let e_0, e_1, e_2 and e_3 denote the top, right, down, left edges of a proposal, respectively. We first compute f_i^e by applying max pooling to all the visible points on e_i . Then, the final representation is obtained by $f_0^e || f_1^e || f_2^e || f_3^e || f_c^p$, where f_c^p indicates the feature of the center point and $||$ indicates concatenation.

2.3.4 Loss Function

Given the output PoI feature representation from the aforementioned three modules topped by fully-connected layers, the head network consists of three branches predicting the box class, localization and direction. The ground truth and anchor boxes are parameterized as $(x, y, z, w, l, h, \theta)$, where (x, y, z) is the center of box,

(w, l, h) is the dimension of box, and θ is the heading along the z-axis in the LiDAR coordinate system. The box regression target is computed as the residuals between the ground truth and the anchors as:

$$\begin{aligned}\Delta x &= \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{h^a}, \\ \Delta w &= \log\left(\frac{w^{gt}}{w^a}\right), \Delta l = \log\left(\frac{l^{gt}}{l^a}\right), \Delta h = \log\left(\frac{h^{gt}}{h^a}\right), \\ \Delta \theta &= \theta_{gt} - \theta_a\end{aligned}\tag{2.2}$$

where x^{gt} and x^a refer to ground truth and anchor box respectively, and $d^a = \sqrt{(w^a)^2 + (l^a)^2}$. To deal with severe class imbalance problem in the dataset, we adopt the focal loss [31] for the classification loss. Smooth L1 loss [21] is used for the regression loss. In addition, to compensate for direction prediction missing in the regression, we adopt a softmax classification loss on orientation prediction.

Similar with that of the vanilla PointPillars network [13], we formally define a multi-task loss for both stages as threefold,

$$L_{stage_i} = \frac{1}{N_{pos}}(\beta_{cls}L_{cls_i} + \beta_{reg}L_{reg_i} + \beta_{dir}L_{dir_i}),\tag{2.3}$$

where i could be either RPN or InfoFocus stage, N_{pos} refers to the number of positive anchors and $\beta_{cls}, \beta_{reg}, \beta_{dir}$ are chosen to balance the weights among classification loss, regression loss and direction loss.

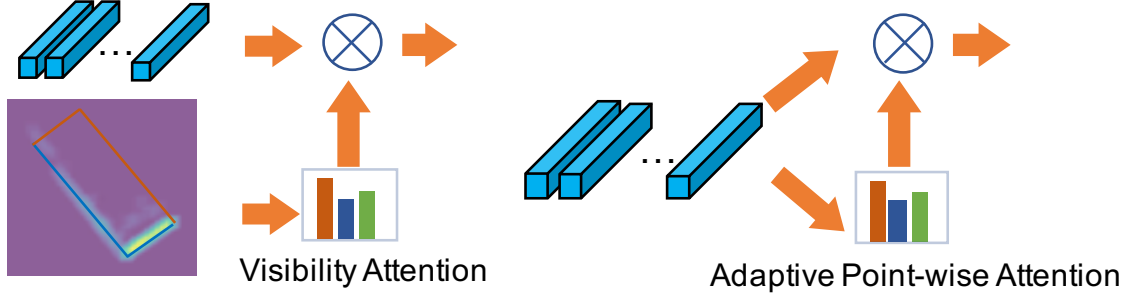


Figure 2.4: Left: illustration of the Visibility Attentive Module. We compute hard attention for each sampled point depending on whether it is visible to the sensor. We also show the visibility map on the bottom left. Points on the blue line are visible while points on the orange line are invisible. Right: the architecture of the Adaptive Point-wise Attention Module. The point-wise attention is generated using a fully connected (FC) layer followed by a Sigmoid function. The input of FC layer is the feature of each point

2.3.5 Comparisons with Existing Approaches

Point-based Approaches. Our framework uses PointNet to extract features from equally divided sub-grids and employs a DCNN to generate 2D feature maps while point-based techniques [8, 9, 10, 11] only use PointNet as its backbone. Both our approach and point-based approaches apply two-stage architecture to infer objects. Meanwhile, we both sample features considering the distribution of point cloud. However, compared to PointNet, InfoFocus is more computationally efficient without performance degradation.

Fusion-based Approaches. Fusion-based detectors [29, 30] make use of both RGB images and point cloud data for 3D object detection. InfoFocus is much faster than fusion-based approaches, since they contain two backbones to process multi-view sources and are heavily engineered. On the other hand, InfoFocus also achieves competitive results compared to fusion-based approaches.

Traditional Voxel-based Approaches. Our method shares the similar backbone as the existing voxel-based architectures [13, 14, 15, 16]. However, previous voxel-based detectors pay less attention to the distribution of LiDAR data that most 3D point cloud locates on the surface of the objects. Our proposed PoI Pooling, Visibility Attentive Module, and Adaptive Point-wise Attention model the non-uniform point cloud using dynamic information focus. First, the PoI Pooling decreases the sampling from the inside of objects where few points locate. Next, the Visibility Attentive Module eliminates the noise from the back of objects where points are occluded. Last, we apply the Adaptive Point-wise Attention to learn the focus on each sampled points. Jointly, these modules contribute significantly to the superior performance of InfoFocus.

2.4 Experiments

Our method is mainly evaluated on the nuScenes dataset [18] which is considered as the most challenging 3D object detection benchmark. We first present our implementation details. We compare with the existing approaches both quantitatively and qualitatively. Then, extensive ablation studies are conducted to demonstrate the effectiveness of each designed module. Last, we analyze the inference time and the desired speed accuracy trade-off provided by our method.

2.4.1 Dataset and Evaluation Metric

NuScenes [18] is one of the largest datasets for autonomous driving. There are 1000 scenes of 20s duration each, including 23 object classes annotated with 28,130 training, and 6,019 validation samples. We use the LiDAR point cloud as the only input to our method and all the experiments follow the standard protocol on the training and validation sets. Officially, nuScenes evaluates the detection accuracy across different classes, based on the average precision metric (AP) which is computed based on 2D center distance between ground truth and the detection box on the ground plane. In details, the AP score is determined as the normalized area under the precision recall curve above 10%. The final mean AP (mAP) is the average among the set of ten classes over matching thresholds of $\mathbb{D} = \{0.5, 1, 2, 4\}$ meters.

2.4.2 Implementation Details

We integrate InfoFocus into a state-of-the-art real-time 3D object detector [13] to improve the detection performance without largely compromising speed. Closely following the codebase¹ recommended by the authors of PointPillars [13], we use PyTorch to implement our InfoFocus modules and integrate it into vanilla PointPillars network.

Pillar Feature Network. Closely following the codebase, we focus on the input LiDAR point cloud range with $[-50, -50]$, $[-50, 50]$ and $[-5, 3]$ meters in x, y, z axis

¹<https://github.com/traveller59/second.pytorch>.

respectively. The voxel size is $[0.25, 0.25, 8]$ and thus the pillars' size is 400×400 after the voxelization. The generated pseudo 2D image [13] is $400 \times 400 \times 64$ from Pillar Feature Network with 64-d output channels. Specifically, the width and length of the pseudo image, W and L , are set to be 400 and 400, while the channel size C is 64.

DCNN. The structure of DCNN includes three blocks of fully convolutional layers, where each of block consists of down-sampling convolutional layers to produce top-down features and de-convolutional layers to upsample and concatenate the feature maps from different strides [13]. Each convolution/deconvolutional uses the kernel with size 3×3 and is followed by a BatchNorm layer and a ReLU layer. In details, the input/output size of the first convolutional block is $400 \times 400 \times 64$ and $200 \times 200 \times 64$, respectively. After the first deconvolutional layer, the dimension of first block feature map becomes $100 \times 100 \times 128$. Similarly, the input/output size of the second convolutional block is $200 \times 200 \times 64$ and $100 \times 100 \times 128$. After the second deconvolutional layer, the dimension of second block feature map is $100 \times 100 \times 128$. For the last deconvolutional layer, the input/output size is $100 \times 100 \times 128$ and $50 \times 50 \times 256$. After the last deconvolutional layer, the dimension of third block feature map becomes $100 \times 100 \times 128$. The final concatenated feature map from the output of three deconvolutional layers has a size of $100 \times 100 \times 384$.

RPN. Unlike the original PointPillars [13] that adopts a Single Shot Detector (SSD) [32] as detection head, we utilize an improved implementation with a dual-head for the RPN. Specifically, an 1×1 convolutional layer is used in each of three branches following the suggested codebase. The dataset is empirically divided into

two groups based on the object size, e.g., {car, bus, construction_vehicle, trailer, truck} and {barrier, bicycle, motorcycle, pedestrian, traffic_cone}. Specifically, the small-scale head takes the feature map from the first convolutional block with size of $200 \times 200 \times 64$, while the large-scale head takes the concatenated feature map with size of $100 \times 100 \times 384$, both from the deep feature extractor. The output of RPN is the candidate proposals with classification, bounding box and direction predictions.

For each class of objects, the RPN anchor size is set by calculating the average of all objects from the corresponding class in training set of nuScenes. In addition, the matching thresholds are based on the custom configuration following the suggested codebase. 1,000 proposals are obtained from RPN, on which NMS with a threshold of 0.5 is applied to remove the overlapping proposals for both training and inference. The final top-ranked 300 proposals are kept for the InfoFocus stage to simultaneously predict the category, location and direction of objects during both the training and inference.

InfoFocus. The second stage is our proposed InfoFocus. The three novel modules process object-centric feature sequentially based on the initial bounding box proposals from RPN. The number of sampled key-points for each edge, n , is set to be 2. Thus, the total number of PoIs, N_{poi} , is 13, including a center, 4 corners and 2 key-points on each edge. Similar to RoIAlign [33], bi-linear interpolation is used to compute the deep feature from four neighboring regular locations of each point.

As mentioned before, we apply a max-pool layer to summarize the features of points along each edge, resulting in 5 features for each proposal, including features

from top, right, down, left edges and the center. When concatenating these features, we always treat the edge that is closest to the sensor as the top edge. A fully connected layer with a single node is used to generate point-wise attention weight for each point.

The feature of each proposal is transformed by two consecutive FC layers with 512 nodes each and passed to three sibling linear layers, a box-regression branch, a box-classification branch and a box-direction branch. For the regression target assignment, anchors having Intersection over Union (IoU) bigger than 0.6 with the ground truth are considered positive, and smaller than 0.55 are assigned negative labels.

Training Parameters. Experiments are conducted on a single NVIDIA 1080Ti GPU. The weight decay is set to be 0.01. We adopt the Adam optimizer [34], and use a one-cycle scheduler proposed in [35]. We train our model with a total of 20 epochs as a default choice, taking about 40 hours from scratch. For the first 8 epochs, the learning rate progressively increases from 3×10^{-4} to 3×10^{-3} with decreasing momentum from 0.95 to 0.85, while in the remaining 12 epochs learning rate decreases from 3×10^{-3} to 3×10^{-6} with increasing momentum from 0.85 to 0.95. The focal loss [31] with $\alpha = 0.25$ and $\gamma = 2.0$ is adopted for the classification loss. The balancing weights for the classification, box regression, and direction loss $\beta_{cls}, \beta_{reg}, \beta_{dir}$ of both stages are 1, 2 and 0.2, respectively.

Method	car	peds.	barri.	traff.	truck	bus	trail.	const.	motor.	bicyc.	mAP
PointPillars [13]	70.5	59.9	33.2	29.6	25.0	34.3	16.7	4.5	20.0	1.6	29.5
WYSIWYG [17]	80.0	66.9	34.5	27.9	35.8	54.1	28.5	7.5	18.5	0	35.4
Ours	77.6	61.7	43.4	33.4	35.4	50.5	25.6	8.3	25.2	2.5	36.4

Table 2.1: Object detection results (%) on nuScenes validation set

2.4.3 Main Results

First, we compare our framework with the state-of-the-art methods on the nuScenes validation set, including the vanilla PointPillars [13] as our baseline, and recently published WYSIWYG [17]. As can be seen from Table. 2.4.3, the baseline has an mAP of 29.5% with a single stage, while InfoFocus improves it by a massive 6.9%. This demonstrates the effectiveness of InfoFocus. We also visualize the detection results of our framework on 2D and 3D BEV images in Fig. 2.5. As shown in Fig. 2.6, compared to the vanilla PointPillars qualitatively, InfoFocus helps remove the false positives significantly and obtains better results.

In addition, we submit the detection results of test set on the nuScenes test server. The results show that our method achieves the state-of-the-art performance with inference speed of 31 FPS, improving the baseline performance by 7% mAP. Note that all methods listed in Table. 2.2 are LiDAR-based except that MonoDIS [36] and CenterNet[37] are camera-based methods. Without bells and whistles, our approach works better than WYSIWYG [17]. Considering that our model contains more parameters than the vanilla PointPillars, we empirically increase the number of the training epoch by 2 times. With all the others settings the same, our method is improved by 2% mAP on the nuScenes test set as shown in Table. 2.2 (Ours 2×).

Method	car	peds.	barri.	traff.	truck	bus	trail.	const.	motor.	bicyc.	mAP
MonoDIS [36]	47.8	37.0	51.1	48.7	22.0	18.8	17.6	7.4	29.0	24.5	30.4
PointPillars [13]	68.4	59.7	38.9	30.8	23.0	28.2	23.4	4.1	27.4	1.1	30.5
SARPNET [16]	59.9	69.4	38.3	44.6	18.7	19.4	18.0	11.6	29.8	14.2	32.4
CenterNet [37]	53.6	37.5	53.3	58.3	27.0	24.8	25.1	8.6	29.1	20.7	33.8
WYSIWYG [17]	79.1	65.0	34.7	28.8	30.4	46.6	40.1	7.1	18.2	0.1	35.0
Ours	77.2	61.5	45.3	40.4	31.5	44.1	35.9	9.8	25.1	4.0	37.5
Ours 2×	77.9	63.4	47.8	46.5	31.4	44.8	37.3	10.7	29.0	6.1	39.5

Table 2.2: Object detection results (%) on nuScenes test set. Note that MonoDIS and CenterNet are camera based methods, and the rest are LiDAR based. Ours 2× indicates 2× training time with other settings being the same with Ours

PoIPool	Vis. Att.	Adp. Att.	mAP
			29.5
✓			32.5
✓	✓		34.8
✓		✓	34.8
✓	✓	✓	36.4

Table 2.3: Ablation studies on nuScenes validation set. "Vis. Att" and "Adp. Att." refer to Visibility Attention Module and Adaptive Attention Module, respectively

In total, our method outperforms WYSIWYG[17] by 4.5% mAP on the nuScenes test set. In the rest of chapter, the default setting of training epochs is adopted. To the best of our knowledge, our framework is superior than all the published real-time methods with respect to mAP.

2.4.4 Ablation Studies

To understand the contribution of our major component to the success of InfoFocus, Table. 2.4.4 summarizes the performance of our framework when a certain module is disabled, including PoI Pooling, Visibility Attention Module and Adaptive Attention Module.

PoI Pooling. To investigate the effect of PoI Pooling, we simply add the PoI Pooling on top of the vanilla PointPillar. This baseline introduces 3.0% mAP improvement. However, when we vary the number of pooling key-points on each edge, we see that our framework with four key-points ($n = 4$) on each edge degrades slightly by 0.8% mAP than that of two key-points ($n = 2$). A possible reason is that the higher number of samples along each edge might bring more noise which harms the detection performance.

Visibility Attention. We further add the Visibility Attention module to filter out invisible edges before PoI pooling. Table. 2.4.4 shows that when using the features from two visible edges, the mAP result is improved by 2.3% mAP compared to *baseline+PoIPool*. Generally, the visible parts of objects correspond to their sides closer to the LiDAR sensor, thus they may capture richer information. By applying visibility attention, our method focuses more on the representative information which results in better performance.

Adaptive Point-wise Attention. Without the Adaptive Point-wise Attention module, the framework naturally allows the same weight for each PoI feature. As we can see in Table. 2.4.4, when adding this module, the result of *baseline+PoIPool* improves by 2.3% mAP and that of *baseline+PoIPool+Vis.Att.* improves by 1.6%. These results suggest that the Adaptive Point-wise Attention module helps emphasize on useful points which leads to a better performance.

Rotated RoIAlign Comparison. One widely considered way to extract the region-wise features in the two-stage architecture is RoIAlign [33]. So, it is intuitive to compare with this strategy under the setting of 3D object detection. We

Method	Input Format	mAP	Inference Time (ms)
Baseline [13]	LiDAR	30.5	26.9
MonoDIS [36]	RGB	30.4	29.0
SARPNET [16]	LiDAR	32.4	70.0
Ours	LiDAR	37.5	32.9

Table 2.4: Inference time of 3D object detectors. Note that inference time for the baseline here is the network reproduced by ourselves

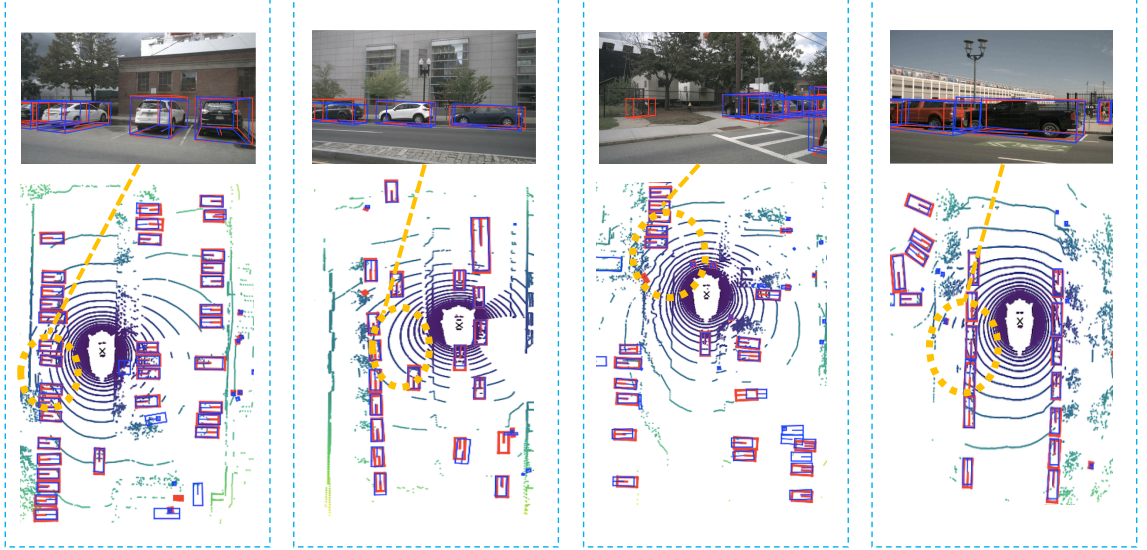


Figure 2.5: We visualize the detection results on nuScenes with 2D and 3D BEV images. On the top, we demonstrate the 2D images with the 3D bounding box annotated, while the BEV of LiDAR with ground truth (red) and detection (blue) box are shown on the bottom. Note that the line in the frame denotes the direction of the object

implement rotated RoIAlign (RRoI) operation [38] to compensate for the rotated bounding box, since in our case they are often not axis-aligned. We conduct experiments exploring two different pooling sizes, 4×4 (pooled length and pooled width), and 8×4 with 4 sampled points in each bin. One of the reasons that we use 8×4 is that most of the objects like car and bus’s length is larger than their width. With all other implementations the same as InfoFocus, Table. 2.4.4 presents detection results utilizing the rotated-RoI with different pooling sizes. Compared with the

Method	car	peds.	barri.	traff.	truck	bus	trail.	const.	motor.	bicyc.	mAP
RRoI 4x4	76.9	60.1	37.6	29.5	32.4	50.6	22.4	5.0	20.8	3.8	33.9
RRoI 8x4	77.0	59.5	36.7	29.2	33.2	51.5	25.4	4.5	24.0	1.8	34.3
Ours	77.6	61.7	43.4	33.4	35.4	50.5	25.6	8.3	25.2	2.5	36.4

Table 2.5: Comparison with rotated RoIAlign feature extraction results (%) on the nuScenes validation set

vanilla PointPillars [13], adding the RoIAlign layer with size of 4×4 increases the mAP performance by 4.4%. However, InfoFocus still outperforms RoIAlign by 2.5% with the better information modeling scheme.

2.4.5 Real-time Inference Analysis

As indicated in Table. 2.4.4, our framework takes about 32.9 ms to perform detection on an example of point cloud in the nuScenes, compared with 26.9 ms of the vanilla PointPillars when both are evaluated on a single NVIDIA 1080Ti GPU. In details, the pillar feature extraction time is 12.6 ms, the DCNN costs 1.1 ms, RPN takes 7.3 ms to generate proposals, and the InfoFocus stage takes 11.9 ms. Specifically, the proposal generation for the InfoFocus stage including NMS is 5.1 ms, the PoI feature extraction time is 3.1 ms, and the second stage including three branches takes 0.7 ms. We also note that WYSIWYG [17] provides the overhead of computing visibility over a 32-beam LiDAR point to be 24.4 ± 3.5 ms on average and InfoFocus is faster than WYSIWYG [17] since we share the similar backbone network. The framework with RRoIAlign has an inference time of 32.2 ms. Further, compared with other point-based methods [11, 12], InfoFocus is considerably faster and conceptually simpler.

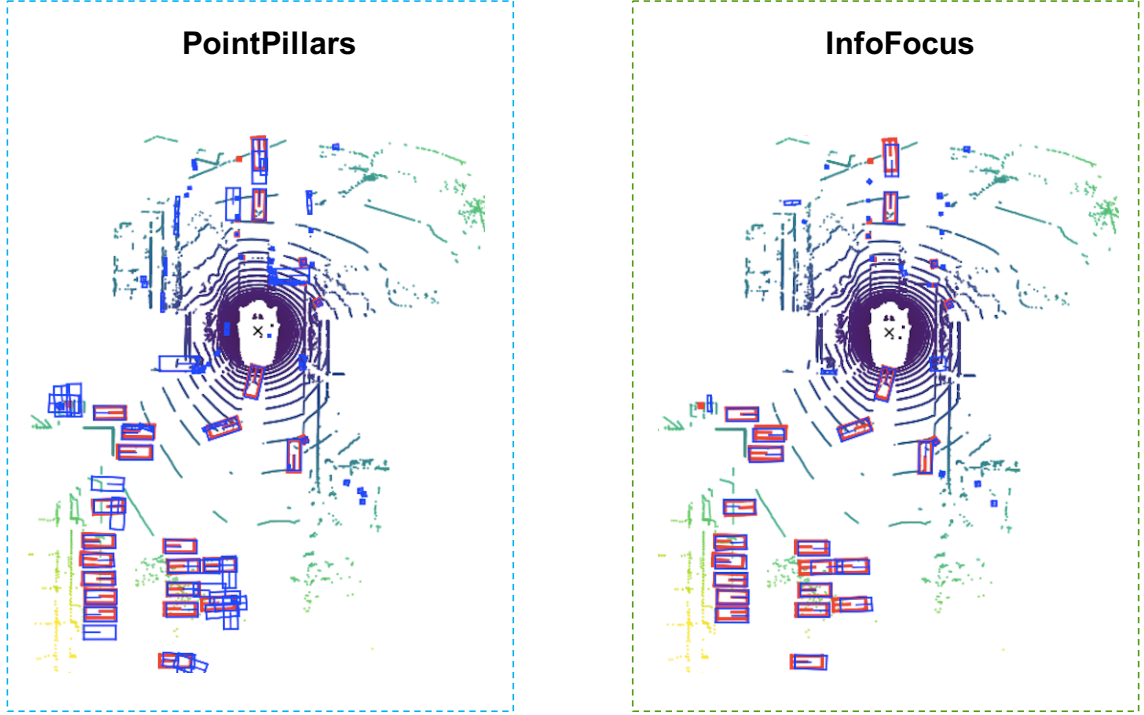


Figure 2.6: We visualize the BEV detection results for the same point cloud sample on nuScenes with the vanillar PointPillars (left) and InfoFocus (right)

2.5 Conclusion

Non-uniform distribution of point cloud causes varying amount of information at different locations. Therefore, we argue that this imbalance distribution of information may result in degradation on previous 3D voxel-based detectors when modeling 3D objects. To address this issue, we propose a 3D object detection framework with InfoFocus to dynamically conduct information modeling. InfoFocus contain three effective modules including PoI Pooling, the Visibility Attentive Module, and the Adaptive Point-wise Attention. Demonstrated by the comprehensive experiments, our framework achieves the state-of-art performance among all the real-time detectors on the challenging nuScenes dataset.

Chapter 3: M3DETR: Multi-representation, Multi-scale, Mutual-relation 3D Object Detection with Transformers

3.1 Introduction

3D object detection is a fundamental problem in computer vision and many applications, including autonomous driving [39, 40], augmented reality [41] and robotics [42]. Moreover, different methods have been proposed for various sensors, including monocular cameras, depth cameras, LiDAR, and radars [8, 12, 19, 20, 43, 44, 45, 46]. 2D object detection deals with detecting objects from RGB images and videos, while 3D object detection utilizes point cloud-based representations obtained from LiDARs or other sensors. Moreover, it is known that point cloud data obtained from LiDAR sensors tends to be more accurate than RGB images and videos [47]. Consequently, point clouds are being widely used for scene understanding in autonomous driving and AR.

The previous state-of-the-art methods for 3D object detection base on different networks [45, 46]. However, there are two key limitations:

Ineffective point cloud representations: The three major techniques used to process point clouds are based on voxels [14, 15], raw point clouds [8, 11, 12, 19, 20,

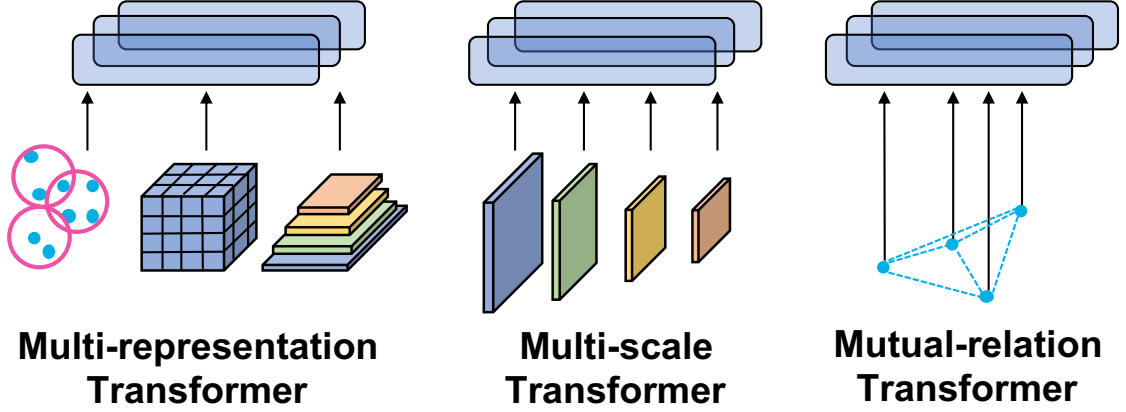


Figure 3.1: M3 Transformers used in our approach: Left: Multi-representation transformers; Middle Part: Multi-scale transformers; Right: Mutual-relation transformers. We use these characteristics to present a novel 3D object detection architecture.

30, 48], and bird’s-eye-view [24, 25, 49]. Each representation has a unique advantage and it has been shown that combining these representations can result in terms of detection accuracy [8, 29, 45]. However, fusing these representations is non-trivial. First, the architectures corresponding to the VoxelNets, the PointNets, and the 2D convolutional neural networks are different. Moreover, raw point clouds need to be converted to voxels and pixels before techniques based on VoxelNets and 2D convolutional neural networks can be applied. The differences between the inputs of these three neural models can result in semantic gaps. Previous works [29, 45] tend to use feature concatenation and attention modules to fuse multi-representation features. However, the correlation between features of different representations has not been addressed.

Insufficient modeling of multi-scale features: Fusing multi-scale feature maps is a well-known technique used for improving the detection performance of 2D object detection. In terms of 3D object detection, current approaches tend to use multi-

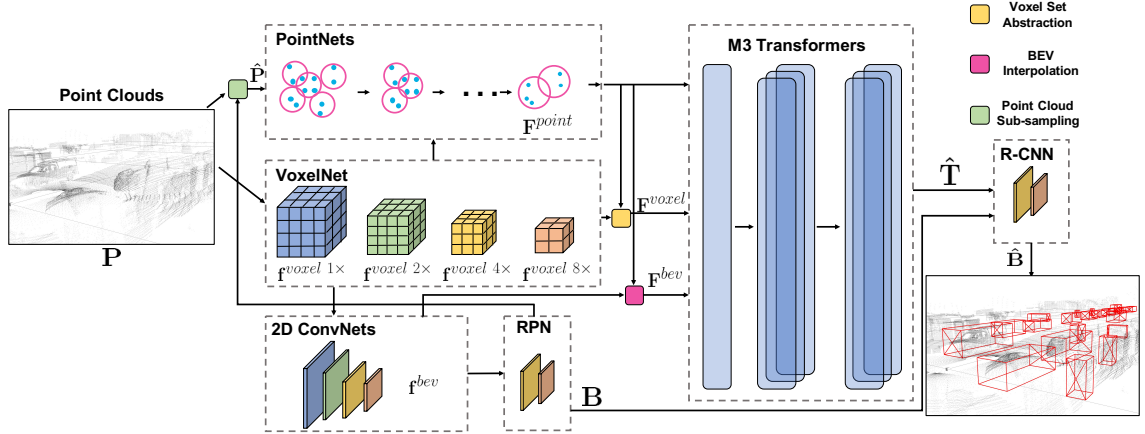


Figure 3.2: An overview of our M3DETR architecture: M3DETR is a transformer based framework for object detection in a coarse-to-fine manner. It consists of three parts. PointNets, VoxelNet, and 2D ConvNets modules enable individual multi-representation feature learning. M3 Transformers enable inter-intra multi-representation, multi-scale, and multi-location feature attention. With the Region Proposal Network (RPN), the initial box proposals are generated. R-CNN captures and refines region-wise feature representations from M3 transformer output to improve detection performance.

scale feature pyramids [13, 19, 20, 43, 44, 50]. However, fusing these multiple feature pyramids is non-trivial because the higher resolution and the larger receptive fields are conflicting [20, 51]. Existing methods [20, 43, 45] fuse the multi-scale features using bi-linear down-sampling/up-sampling and concatenation. Although these approaches [20, 43, 45] can improve the accuracy by a large margin, there are many challenges with respect to the underlying fusion method in terms of the correlation between feature maps of different scales.

A key issue in terms of designing a good approach for object detection is exploiting the correlation between different representations and the large size of the receptive fields. Our approach is motivated by use of transformers [52], a form of neural network based on attention that has been used in natural language processing [53, 54, 55, 56]. Specifically, transformers use multi-head attention to narrow the

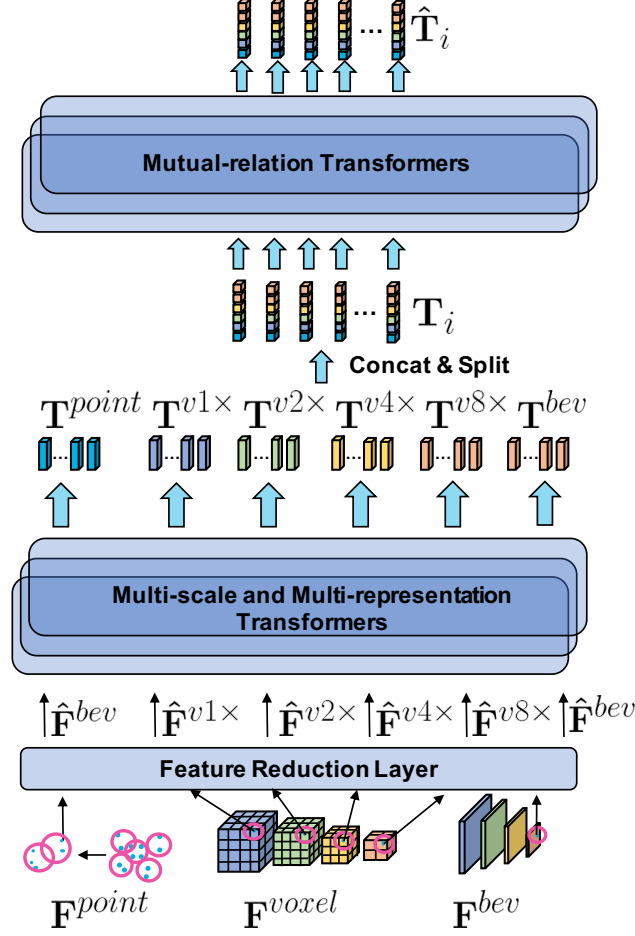


Figure 3.3: M3 Transformers consist of two parts: a multi-representation and multi-scale transformer, and a mutual-relation transformer. Multi-representation and multi-scale transformer takes the different feature embedding and generates enriched cross-representations and cross-scales embedding. On top of it, the mutual-relation transformer further models point-wise feature relationship to extract the refined features.

semantic gap between different representations by adapting to informative features and eliminating noise.

Another key aspect of 3D object detection is to model the mutual relationships between different points in the point cloud data [19, 20, 50, 57]. Modeling mutual relationships can enhance the ability to recognize the fine-grained patterns and can generalize to complex scenes. Prior works in 3D object detection have modeled these relationships using multi-layer perceptrons [8, 19, 20, 45], farthest point sampling

layers [11, 20, 45], max pooling layers [11, 20, 45, 50], and graph convolutional networks [58]. However, a key challenge is to model these mutual relationships along with fusing different representations and multi-scale features.

We present M3DETR, a novel two-stage architecture for 3D object detection task. Given raw 3D point cloud data, our approach can localize static and dynamic obstacles with state-of-the-art accuracy. As illustrated in Figure 3.1, the key components of our approach are M3 transformers, which are used to combine different feature representations. Conceptually, each of the M3 transformers are used for aggregating point cloud representations, multi-scale representations and mutual relationships among a subset of points in the point cloud data.

Summary of contributions:

- M3DETR is the first unified architecture for 3D object detection with transformers that accounts for multi-representation, multi-scale, mutual-relation models of point clouds in an end-to-end manner.
- M3DETR is robust and insensitive with respect to the hyper-parameters of transformer architectures. We test multiple variants with different transformer blocks designs. We demonstrate improved performance of M3DETR regardless of hyper-parameters.
- Our unified architecture achieves state-of-the-art performance on KITTI 3D Object Detection Dataset [39] and Waymo Open Dataset [40]. We outperform the previous state-of-the-art approaches by 2.86% mAP for car class on the

Waymo validation set and 1.48% mAP for all classes on the Waymo test set.

3.2 Related Work

Multi-representation modeling. Existing techniques for modeling 3D point cloud data include bird-eye-view (BEV), volumetric, and point-wise representations. Generally, BEV-based approaches [24, 25, 27, 59] first project 3D point clouds into 2D BEV space and then adopt the standard 2D object detectors to generate 3D object proposals from projected 2D feature maps. To deal with the irregular format of input point clouds, voxel-based architectures [13, 14, 15] use equally spaced 3D voxels to encode the point clouds such that the volumetric representation can be consumed by the region proposal network (RPN) [23]. Inspired by the PointNet/PointNet++ approach [19, 20], which is invariant under transformation, [8, 11] extend this method to the task of 3D object detection and directly process the raw point clouds to infer 3D bounding boxes. However, these methods are typically limited due to either information loss or high computation cost. Recently, many approaches [12, 30, 45, 46, 60] have combined the advantages of speed (of voxel-based representation) and efficiency (of point-based representation) by fusing point-voxel features for 3D object prediction.

Multi-scale modeling. Modeling multi-scale features is an important procedure in deep learning-based computer vision [4, 20, 51, 61, 62, 63, 64, 65, 66] because it is able to enlarge the receptive field and increase resolution. In 3D representation, modeling multi-scale features is also popular and important. PointNet++ [20]

proposes the set abstraction module to model local features of a cluster of point clouds. To model multi-scale patterns of point clouds, they use 3 different sampling ranges and radii with 3 parallel PointNets and thus fuse the multi-scale. In 3D object detection, [4, 43, 66] adopt different detection heads with multi-scale feature maps to handle both large and small object classes.

Mutual-relation modeling. 2D Convolutional Neural Networks [67] are commonly used to process mutual relations in 2D images. As point clouds are scattered and lacking structure, passing information from one point to its neighbors is not trivial. PointNets [20] proposes the set abstraction module to model the local context by using the subsampling and grouping layers. After this well-known work, many convolution-like operators [50, 57, 58] on point clouds have been proposed to model the local context and the mutual relation between points. Recently, transformers [68, 69, 70, 71] have been introduced in PointNets to model mutual relation. However, those previous works mainly focus on the local and global contexts of point clouds by applying mutual-relation transformers on points. Instead, our approach not only models the mutual relation between points, but it also models multi-scale and multi-representation features of point clouds.

Transformers in computer vision. Inspired by their success in machine translation [52], transformer-based architectures have recently become effective and popular in a variety of computer vision tasks. Particularly, the design of self-attention and cross-attention mechanisms in transformers has been successful in modeling dependencies and learning richer information. [72] leverages the direct application of transformers on the image recognition task without using convolu-

tion. [73, 74] apply transformers to eliminate the need for many handcrafted components in conventional object detection and achieve impressive detection results. [75, 76] explore transformers on image and video synthesis tasks. [68] investigates the self-attention networks on 3D point cloud segmentation task.

Recently, there have been several joint representation fusion research attempts. PointPainting [77] proposes a novel method that accepts both images and point clouds as inputs to do 3D object detection by appending 2D semantic segmentation labels to LiDAR points. In the visual question answering task, [53] jointly fuses and reasons over three different modality representations. [78] combines multimodal information to solve robust emotion recognition problems. Building on a multi-representation and multi-scale transformer, our proposed model addresses the voxel-wise, point-wise, and BEV-wise feature representation gap and enables effective cross-representation interactions with different levels of semantic features. Coupled with a point-wise mutual-relation transformer, our framework learns to capture deeper local-global structures and richer geometric relationships among point clouds.

3.3 Our Approach

M3DETR takes point cloud data as input and generates 3D boxes for different object categories with state-of-the-art accuracy as shown in Figure 3.2. Our goal is to perform multi-representation, multi-scale, and mutual-relation fusion with transformers over a joint embedding space. Our method consists of three main steps:

- Generate feature embeddings for different point cloud representations using

VoxelNet, PointNet, and 2D ConvNet.

- Fuse these embeddings using *M3 transformer* that leverages multi-representation and multi-scale feature embedding and models mutual relationships between points.
- Perform 3D detection using detection heads network, including RPN and R-CNN stages.

3.3.1 Multi-Representation Feature Embeddings

Our network processes the raw input point clouds $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$ and encodes them into three different embedding spaces, namely, voxel-, point-, and BEV-based feature representations. We discuss the embedding process for each representation in detail.

Voxels: Voxel-wise feature extraction is divided into two steps: (1) the voxelization layer, which is used by VoxelNet [14], takes the raw input point clouds $\mathbf{P}$ and converts them into equally spaced 3D voxels $\mathbf{v} \in \mathbb{R}^{L \times W \times H}$; (2) voxel-wise features $\mathbf{f}^{voxel}$ at different scales are extracted with 3D sparse convolutions, which is visualized in Figure 3.2. We denote the size of each discretized voxel as $L \times W \times H \times C$, where L, W, H indicate the length, width, and height of the voxel grid and C represents the channel of the voxel features. We adopt the average of the point-wise features from all the points to represent the whole non-empty voxel feature. After voxelization, the input feature is propagated through a series of $3 \times 3 \times 3$ sparse cubes, including four consecutive blocks of 3D sparse convolution with downsampled

sizes of $1\times, 2\times, 4\times, 8\times$, using convolution operations of stride 2. Specifically, each sparse convolutional block includes a 3D convolution layer followed by a LayerNorm layer and a ReLU layer. Unlike [14, 15], all four different scales of obtained 3D voxel CNN features, $\mathbf{f}^{voxel} = \{\mathbf{f}^{voxel\ 1\times}, \mathbf{f}^{voxel\ 2\times}, \mathbf{f}^{voxel\ 4\times}, \mathbf{f}^{voxel\ 8\times}\}$ are passed to the transformer stage as shown in Figure 3.2.

Bird’s-eye-view: 2D ConvNets takes the $8\times$ downsampled voxel-based feature map $\mathbf{f}^{voxel\ 8\times}$ from the 3D Voxel CNN branch and generates a corresponding BEV-representation embedding. To directly apply 2D convolution, we start by combining the z-dimension and the channel dimension of the input voxel features into a single dimension. The structure of 2D ConvNets includes two encoder-decoder blocks, where each block consists of 2D down-sampling convolutional layers to produce top-down features, as well as de-convolutional layers to upsample to the input feature size. Specifically, both encoder and decoder paths are composed of a number of 2D convolutional layers with the kernel size of 3×3 followed by a BatchNorm layer and a ReLU layer. The output BEV-based feature from 2D ConvNets, denoted as $\mathbf{f}^{bev} \in \mathbb{R}^{L_{bev} \times W_{bev} \times C_{bev}}$, was further converted into keypoint feature $\mathbf{F}^{bev} \in \mathbb{R}^{n \times c_{bev}}$ through bi-linear interpolation.

Points: Typically, there are more than 10K raw points inside an entire point cloud scene. In order to cover the entire point set effectively without large memory consumption, we apply Furthest-Point-Sampling (FPS) algorithm to sample n keypoints, denoted as $\hat{\mathbf{P}} \subset \mathbf{P}$. Adopted from PointNet++ [20] and PV-RCNN [45], Set Abstraction and Voxel Set Abstraction (VSA) module take raw point coordinates $\mathbf{P}$ and the 3D voxel-based features $\mathbf{f}^{voxel}$, respectively, to generate keypoint features

for $\hat{\mathbf{P}}$. In particular, the corresponding keypoint features from raw points $\mathbf{P}$ is $\mathbf{F}^{point} \in \mathbb{R}^{n \times c_{point}}$, and the corresponding keypoint features from voxels $\mathbf{f}^{voxel}$ are $\{\mathbf{F}^{voxel\ 1\times}, \mathbf{F}^{voxel\ 2\times}, \mathbf{F}^{voxel\ 4\times}, \mathbf{F}^{voxel\ 8\times}\}$, where $\mathbf{F}^i \in \mathbb{R}^{n \times c_i}$.

3.3.2 Multi-representation, Multi-scale, and Mutual-relation Transformer

Once the three feature embedding sequences, $\mathbf{F}^{voxel}$, $\mathbf{F}^{point}$ and $\mathbf{F}^{bev}$ are generated, they are able to dynamically and intelligently attend to each other, generating final cross-representations, cross-scales, and cross-points descriptive feature representations as shown in Figure 3.2. In the remainder of this section, we will briefly review transformers basics followed by discussing the multi-representation, multi-scale, and mutual-relation transformer layers.

Transformer basics. A transformer is a stacked encoder-decoder architecture relying on a self-attention mechanism to compute representations of its input and output [52]. Consider two input matrices $\mathbf{X}_l \in \mathbb{R}^{l \times d_{in}}$ and $\mathbf{X}_s \in \mathbb{R}^{s \times d_{in}}$, where l and s are the lengths of the input sequence of dimension d_{in} . The attention layer output is defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{V}, \mathbf{K}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{in}}}\right)\mathbf{V} \in \mathbb{R}^{l \times d_{out}},$$

where $\mathbf{Q} = \mathbf{X}_l \mathbf{W}_q$, $\mathbf{K} = \mathbf{X}_l \mathbf{W}_k$ and $\mathbf{V} = \mathbf{X}_s \mathbf{W}_v$.

The matrices $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ represent query, key and value respectively, obtained through projection matrices $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d_{in} \times d_h}$ and $\mathbf{W}_v \in \mathbb{R}^{d_{in} \times d_{out}}$, where d_h is the

hidden dimension, and d_{out} is the output dimension. When two input matrices $\mathbf{X}_l$ and $\mathbf{X}_s$ represent the same feature maps, the attention module is usually referred to as self-attention. Each transformer layer consists of one Multi-head Self-attention (MHSA) module and a few linear layers, as well as normalization and activation layers.

Building on the attention mechanism, Multi-head Self-attention (MHSA) with N heads and the input matrix $\mathbf{X}$ is defined as follows:

$$\text{MHSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concatenate}[\text{head}_1, \dots, \text{head}_N] \mathbf{W}^O,$$

where $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^q, \mathbf{K}\mathbf{W}_i^k, \mathbf{V}\mathbf{W}_i^v)$ and $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are linear projections of $\mathbf{X}$. The projection matrices $\mathbf{W}_i^q \in \mathbb{R}^{d_h \times d_q}$, $\mathbf{W}_i^k \in \mathbb{R}^{d_h \times d_k}$, $\mathbf{W}_i^v \in \mathbb{R}^{d_h \times d_v}$, $\mathbf{W}^O \in \mathbb{R}^{d_v \times d_{out}}$ are learnable parameters in the network that correspond to each of the attention heads, where d_k , d_v and d_q are the hidden dimensions of each attention head for $\mathbf{K}$, $\mathbf{V}$, and $\mathbf{Q}$, respectively.

Now, we present the proposed transformers used to capture the inter- and intra- interactions among input features. Specifically, we propose two stacked transformer encoder layers named M3 Transformers, as shown in Figure 3.3: (1) the multi-representation and multi-scale transformer, and (2) the mutual-relation transformer.

Multi-representation and multi-scale transformer layer. First, we focus on the intra-point representation fusion. For each individual point, the input sequence to this transformer layer is its several different corresponding point cloud features,

including various scales and distinctive representations. The input sequence to the transformer is $\mathbf{F} = [\mathbf{F}^{voxel\ 1\times}, \mathbf{F}^{voxel\ 2\times}, \mathbf{F}^{voxel\ 4\times}, \mathbf{F}^{voxel\ 8\times}, \mathbf{F}^{point}, \mathbf{F}^{bev}]$, where each $\mathbf{F}^i \in \mathbb{R}^{n \times c_i}$. After explicitly modeling all element-wise interactions among those features for each point separately, the output from the block is the updated feature vectors after aggregating the information from all the input sequence $\mathbf{F}$.

The multi-representation and multi-scale transformer layer takes 6 different inputs: $\mathbf{F}^{point}, \mathbf{F}^{voxel1\times}, \mathbf{F}^{voxel2\times}, \mathbf{F}^{voxel4\times}, \mathbf{F}^{voxel8\times}, \mathbf{F}^{bev}$. As different inputs may have different feature dimensions, we use the single-layer perceptron to apply feature reduction on the input features to align the feature dimensions of each feature embeddings. The outputs of the feature reduction layer are $\hat{\mathbf{F}} = [\hat{\mathbf{F}}^{voxel\ 1\times}, \hat{\mathbf{F}}^{voxel\ 2\times}, \hat{\mathbf{F}}^{voxel\ 4\times}, \hat{\mathbf{F}}^{voxel\ 8\times}, \hat{\mathbf{F}}^{point}, \hat{\mathbf{F}}^{bev}]$, where the output dimension of each feature is equivalent to $\hat{c}$.

After the feature reduction layer, the multi-representation and multi-scale transformer layer takes $\hat{\mathbf{F}}$ as inputs and generates self-attention features $\mathbf{T}^{voxel\ 1\times}, \mathbf{T}^{voxel\ 2\times}, \mathbf{T}^{voxel\ 4\times}, \mathbf{T}^{voxel\ 8\times}, \mathbf{T}^{point}, \mathbf{T}^{bev}$, which corresponds to $\hat{\mathbf{F}}^{voxel\ 1\times}, \hat{\mathbf{F}}^{voxel\ 2\times}, \hat{\mathbf{F}}^{voxel\ 4\times}, \hat{\mathbf{F}}^{voxel\ 8\times}, \hat{\mathbf{F}}^{point}, \hat{\mathbf{F}}^{bev}$. We visualize the multi-representation and multi-scale transformer in Figure 3.3.

Mutual-relation transformer layer. Inspired by [19, 20, 71], inter-points feature fusion within a spatial neighboring space is leveraged in the mutual-relation layer of the transformer. Our goal is to attend to and aggregate the neighboring information in an attention manner for each point with an enriched feature.

From the first transformer block output, we obtain aggregated features $\mathbf{T}$ of different scales and representations after the first transformer block. We concatenate

channels along the point dimension and rearrange those points into a sequence as the input of the mutual-relation transformer. Let the learned concatenated feature $\mathbf{T} = \text{concat} \{ \mathbf{T}^{voxel\ 1\times}, \mathbf{T}^{voxel\ 2\times}, \mathbf{F}^{voxel\ 4\times}, \mathbf{T}^{voxel\ 8\times}, \mathbf{T}^{point}, \mathbf{T}^{bev} \} \in \mathbb{R}^{n \times c_T}$, where c_T is the sum of all channel size in $\mathbf{T}$. Note that, the first dimension of $\mathbf{T}$ is the number of keypoints n . To model the mutual-relation between keypoints, we need to split $\mathbf{T}$ into n point-wise feature $\mathbf{T} = \{\mathbf{T}_i\}$, where $\mathbf{T}_i \in \mathbb{R}^{c_T}$. The concatenation and split module is showed in Figure 3.3.

The mutual-relation transformer takes point-wise features of n keypoints as inputs and uses the multi-head self-attention head to model the mutual relationship between keypoints. The outputs of the mutual-relation transformer are $\hat{\mathbf{T}}$.

Comparison with previous point-based transformers Prior work has explored the transformer application on the task of point cloud processing. First, [68, 69, 70] leverage the inherent permutation invariance of transformers to capture local context within the point cloud on the shape classification and segmentation tasks, while our M3DETR mainly investigates the strong attention ability of transformers between input embeddings for the 3D object detection task. Pointformer [71] is the approach most related to our method because we both address the 3D object detection task by capturing the dependencies among points’ features. However, Pointformer adopts a single PointNet branch to extract points feature, while M3DETR considers all three different representations and also applies the transformer to learn aggregated representation-based features.

3.3.3 Detection Heads Network

After we obtain the enriched embedding $\hat{\mathbf{T}}$ from the M3 transformer, the detection network is composed of two stages that predict 3D bounding box class, localization, and orientation in a coarse-to-fine manner, including RPN and R-CNN. Please refer to PV-RCNN [45] for more details.

RPN: Region Proposal Networks take the deep semantic features $\mathbf{f}^{bev}$ produced by the 2D ConvNets as inputs and generate high-quality 3D object proposals $\mathbf{B}$, as shown in Figure 3.2. A 3D object box $\mathbf{B}_i$ is parameterized as $(x, y, z, l, h, w, \theta)$, where (x, y, z) is the center of the box, (l, h, w) is the dimension of the box, and θ is the orientation in bird’s-eye-view. Similar to the conventional RPN in 2D object detection [23] each position on the deep feature map is placed by predefined bounding box anchors denoted as $(x^a, y^a, z^a, l^a, h^a, w^a, \theta^a)$. Then initial proposals are generated by predicting the relative offset of an object’s 3D ground truth bounding box $(x^{gt}, y^{gt}, z^{gt}, l^{gt}, h^{gt}, w^{gt}, \theta^{gt})$.

R-CNN: R-CNN serves as the second stage and it takes the initial region proposals $\mathbf{B}$ from RPN as input to conduct further proposal refinement. For each input proposal box, the RoI-grid pooling module [45] is adopted to extract the corresponding proposal-specific grid points’ features from the transformer-based embeddings $\hat{\mathbf{T}}$. Compared with previous works, M3DETR leverages the richer embedding information from the learned transformers for the fine-grained proposal refinement. As the main component to extract refined features, the RoI-grid pooling module uniformly samples $N \times N \times N$ grid points per 3D proposal. For each grid point, the

output feature is generated by applying a PointNet-block [19] on a small number of surrounding keypoints, M , within its spatial surrounding region with a radius of r . Specifically, keypoints are the subset of input points that are sampled using the Furthest-point-sampling algorithm to cover the entire point set.

Method	Vehicle				Pedestrian				Cyclist				All			
	L1 mAP	L1 mAPH	L2 mAP	L2 mAPH	L1 mAP	L1 mAPH	L2 mAP	L2 mAPH	L1 mAP	L1 mAPH	L2 mAP	L2 mAPH	L1 mAP	L1 mAPH	L2 mAP	L2 mAPH
PV-RCNN* [45]	76.89	76.30	67.99	67.46	65.43	55.85	59.56	50.74	66.38	64.68	63.42	61.81	69.57	65.61	63.65	60.00
M3DeTR	77.66	77.09	70.54	69.98	68.20	58.50	60.64	52.03	67.28	65.69	65.31	63.75	71.05	67.09	65.50	61.92
Improvement	+0.77	+0.79	+2.55	+2.52	+2.77	+2.65	+1.08	+1.29	+0.90	+1.01	+1.89	+1.94	+1.48	+1.48	+1.85	+1.92

Table 3.1: M3DeTR outperforms in the Vehicle, Pedestrian and Cyclist classes for both LEVEL_1 and LEVEL_2 difficulty levels on Waymo Open Dataset test set. Note that PV-RCNN* is our reproduced results with single point cloud frame input.

Method	3D mAP LEVEL_1				3D mAPH LEVEL_1				3D mAP LEVEL_2				3D mAPH LEVEL_2			
	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
RCD [79]	69.59	87.2	67.8	46.1	-	-	-	-	-	-	-	-	-	-	-	-
StarNet [80]	61.50	82.20	56.60	32.20	61.00	81.70	56.00	31.80	54.9	81.3	49.5	23.0	54.50	80.80	49.00	22.70
PointPillars [13]	68.62	87.20	65.50	40.92	68.08	86.71	64.87	40.19	65.21	87.93	63.80	38.20	64.29	87.30	62.36	36.87
Det3D [81]	73.29	90.31	70.54	49.10	72.27	89.65	68.96	47.45	65.21	87.93	63.80	38.20	64.29	87.30	62.36	36.87
RangeDet [82]	75.83	88.41	73.83	55.31	75.38	87.95	73.38	<u>54.84</u>	67.12	87.53	67.99	<u>44.40</u>	66.73	87.08	67.58	<u>44.01</u>
PV-RCNN* [45]	<u>76.89</u>	<u>92.27</u>	<u>75.51</u>	<u>55.35</u>	<u>76.30</u>	<u>91.82</u>	<u>74.79</u>	54.27	<u>67.99</u>	<u>89.18</u>	<u>69.39</u>	42.80	<u>67.46</u>	<u>88.75</u>	<u>68.70</u>	41.95
M3DeTR	77.66	92.54	76.27	57.12	77.09	92.09	75.61	56.02	70.54	89.43	70.19	45.57	69.98	89.01	69.54	44.62
Improvement	+0.77	+0.27	+0.76	+1.77	+0.79	+0.27	+0.82	+1.18	+2.55	+0.25	+0.80	+1.17	+2.52	+0.26	+0.84	+0.61

Table 3.2: M3DeTR outperforms in the Vehicle class with different size range on Waymo Open Dataset test set. Note that PV-RCNN* is our reproduced results with single point cloud frame input. We underscore the second best method in each column for comparison.

Finally, the refined representations of each proposal are first passed to two fully connected layers and then generate the box 3D Intersection-over-Union (IoU) guided confidence scoring of class prediction and location refinement of regression targets, $\hat{\mathbf{B}}$. Compared with the traditional classification-guided box scoring, 3D IoU guided confidence scoring considers the IoU between the proposal box and its corresponding ground truth box. Empirically, [45, 83] show that it achieves better results compared with the traditional classification confidence based techniques.

Method	3D mAP LEVEL_1				3D mAPH LEVEL_1				3D mAP LEVEL_2				3D mAPH LEVEL_2			
	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
LaserNet [84]	52.11	70.90	52.90	29.60	50.05	68.70	51.40	28.60	-	-	-	-	-	-	-	-
PointPillars [13]	56.62	81.00	51.80	27.90	-	-	-	-	-	-	-	-	-	-	-	-
RCD [79]	69.59	87.20	67.80	46.10	69.16	86.80	67.40	<u>45.50</u>	-	-	-	-	-	-	-	-
RangeDet [82]	<u>72.85</u>	87.96	69.03	<u>48.88</u>	-	-	-	-	-	-	-	-	-	-	-	-
PV-RCNN [45]	70.30	<u>91.90</u>	<u>69.20</u>	42.20	<u>69.69</u>	<u>91.34</u>	<u>68.53</u>	41.31	<u>65.36</u>	<u>91.58</u>	<u>65.13</u>	<u>36.46</u>	<u>64.79</u>	<u>91.00</u>	<u>64.49</u>	<u>35.70</u>
M3DETR	75.71	92.69	73.65	52.96	75.08	92.22	72.94	51.80	66.58	91.92	65.73	40.44	66.02	91.45	65.10	39.52
Improvement	+2.86	+0.79	+4.45	+3.98	+5.39	+0.88	+4.41	+6.3	+1.22	+0.34	+0.6	+3.94	+1.23	+0.45	+0.61	+3.82

Table 3.3: M3DETR outperforms in the Vehicle class with IoU threshold of 0.7 on the full 202 Waymo Validation Set, especially on the far range (50m to Inf). We underscore the second best method in each column for comparison.

3.3.4 Loss Functions

In this section, we define the loss function. The bounding box regression target for both RPN and R-CNN stages is calculated as the relative offsets between the anchors and the ground truth as: $\Delta x = \frac{x^{gt} - x^a}{d^a}$, $\Delta y = \frac{y^{gt} - y^a}{d^a}$, $\Delta z = \frac{z^{gt} - z^a}{h^a}$, $\Delta h = \log(\frac{h^{gt}}{h^a})$, $\Delta w = \log(\frac{w^{gt}}{w^a})$, $\Delta \theta = \theta^{gt} - \theta^a$, where $d^a = \sqrt{(w^a)^2 + (l^a)^2}$. Similar to [45, 83], the focal loss is applied [31] for the classification loss, $\mathcal{L}_{cls}$. Smooth L1 loss [85] is adopted for the box localization regression target’s losses, $\mathcal{L}_{reg}$ and $\mathcal{L}_{ref}$. In addition, 3D IoU loss [45, 83] is used for $\mathcal{L}_{iou}$.

Similar to PV-RCNN [45], we formally define a multi-task loss for both the RPN and R-CNN stages,

$$\begin{aligned}
\mathcal{L}_{cls} &= -\alpha_a (1 - p^a)^\gamma \log p^a, \\
\mathcal{L}_{reg} &= \sum_{b \in (x, y, z, w, l, h, \theta)} \mathcal{L}_s(\Delta b), \\
\mathcal{L} &= \mathcal{L}_{cls} + \beta_{reg} \mathcal{L}_{reg} + \beta_{iou} \mathcal{L}_{iou} + \beta_{ref} \mathcal{L}_{ref},
\end{aligned} \tag{3.1}$$

where p^a is the model’s estimated class probability for an anchor box, and β_{reg} , β_{iou} and β_{ref} are chosen to balance the weights between classification loss, IoU

loss and regression loss for RPN stage and R-CNN stage. We adopt the default $\alpha = 0.25$, and $\gamma = 2.0$ from the parameters of focal loss [31].

3.4 Experiments

In this section, we evaluate M3DETR both qualitatively and quantitatively on the Waymo Open Dataset [40] and the KITTI Dataset [39] in the task of LiDAR-based 3D object detection. Our main results include achieving a state-of-the-art accuracy on these datasets and robustness to hyper-parameter tuning.

3.4.1 Datasets and Evaluation Metric

Waymo Open Dataset: The Waymo Open Dataset [40] is a large-scale autonomous driving dataset containing 1000 scenes of 20s duration each, with 798 scenes for training and 202 scenes for validation. Each scene is sampled at a frequency of 10Hz. Overall, the dataset includes 12M labeled objects and thus we only use one fifth of the training scenes for the following experiment. We consider LiDAR data as the input to our approach. The evaluation protocol on the Waymo dataset consists of the mean average precision (mAP) and mean average precision weighted by heading (mAPH). For each object category, the detection outcomes are evaluated based on two difficulty levels: LEVEL_1 denotes the annotated bounding box with more than 5 points and LEVEL_2 represents the annotated bounding box with more than 1 point.

KITTI dataset The KITTI 3D object detection benchmark [39] is another

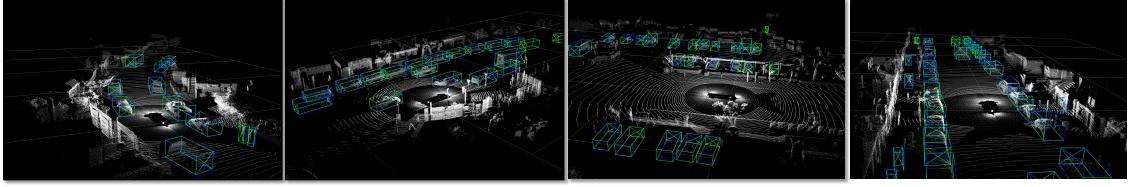


Figure 3.4: We highlight the 3D detection results of M3DETR on the Waymo Open Dataset. The 3D ground truth bounding boxes are in green, while the detection bounding box are shown in blue.

popular dataset for autonomous driving. It contains 7,481 training and 7,518 testing LiDAR scans. We follow the standard split on the training (3,712 samples) and validation sets (3,769 samples). For each object category, the detection outcomes are evaluated based on three difficulty levels based on the object size, occlusion state, and truncation level.

3.4.2 Implementation Details

Closely following the codebase¹, we use PyTorch to implement our M3 transformer modules and integrate them into the PV-RCNN network [45].

Backbone: The 3D voxel CNN branch consists of 4 blocks of 3D sparse convolutions with output feature channel dimensions of 16, 32, 64, 64. Those 4 different voxel representations of different scales, as well as point features from point cloud input, are used to refine keypoint features by PointNet [20] through set abstraction and voxel set abstraction [45]. The number of sampled keypoints n is 2,048 for both Waymo and KITTI. In order to sample the keypoints effectively and accurately, we uses FPS on points within the range of top 1000 or 1500 initial proposals with a radius r of 2.4.

¹<https://github.com/open-mmlab/OpenPCDet>.

For the Waymo Open Dataset, the voxel size is $[0.1, 0.1, 0.15]$, and we focus on the input LiDAR point cloud range with $[-75.2, 75.2]$, $[-75.2, 75.2]$, and $[-2, 4]$ meters in x, y, and z axis, respectively. The 2D ConvNets output size is $188 \times 188 \times 512$.

For the KITTI dataset, the voxel size is $[0.05, 0.05, 0.1]$, and we focus on the input LiDAR point cloud range with $[0, 70.4]$, $[-40, 40]$, and $[-3, 1]$ meters in x, y, and z axis, respectively. The 2D ConvNets output size is $200 \times 176 \times 512$.

M3 Transformer We project the embeddings obtained from the backbone network with different scales and representations to 256 channels, as the input of the multi-representation and multi-scale transformer requires, and project the output features back to their original dimensions before passing into the mutual-relation transformer. Due to the GPU memory constraint, we experiment with two types of MHSA module designs: 2 encoder layers with 4 attention heads and 1 encoder layer with 8 attention heads.

Detection Heads The RPN anchor size for each object category is set by computing the average of the corresponding objects from the annotated training set. Uniformly, the RoI-grid pooling module samples $6 \times 6 \times 6$ grid points within each initial 3D proposal to form a refined features. Specifically, the number of surrounding points used to extract the grid point’s feature, M, is 16.

During the training phase, 512 proposals are generated from RPN to R-CNN, where non-maximum suppression (NMS) with a threshold of 0.8 is applied to remove the overlapping proposals. In the validation phase, 100 proposals are fed into R-CNN, where the NMS threshold is 0.7.

Training Parameters Models are trained from scratch on 4 NVIDIA P6000 GPUs.

Method	Car			Cyclist		
	Easy	Mod	Hard	Easy	Mod	Hard
F-PointNet [8]	72.27	56.12	49.01	82.19	69.79	60.59
VoxelNet [14]	77.47	65.11	57.73	61.22	48.36	44.37
SECOND [15]	83.34	72.55	65.82	75.83	60.82	53.67
PointPillars [13]	82.58	74.31	68.99	77.10	58.65	51.92
PointRCNN [11]	86.96	75.64	70.70	74.96	58.82	52.53
STD [12]	87.95	79.71	75.09	78.69	61.59	55.30
HotSpotNet [86]	87.60	78.31	73.34	<u>82.59</u>	<u>65.95</u>	<u>59.00</u>
PVRCNN [45]	<u>90.25</u>	<u>81.43</u>	<u>76.82</u>	78.60	63.71	57.65
M3DETR	90.28	81.73	76.96	83.83	66.74	59.03
Improvement	+0.03	+0.3	+0.14	+1.24	+0.79	+0.03

Table 3.4: M3DETR outperforms in both Car and Cyclist classes for 3D detection benchmark on KITTI Test Set. We underscore the second best method in each column for comparison.

We use the Adam optimizer with a fixed weight decay of 0.01 and use a one-cycle scheduler proposed in [35]. For the Waymo Open Dataset, we train our models for 45 epochs with a batch size of 8 scenes and a learning rate 0.01, which takes around 50 hours. For the KITTI dataset, we train our models for 80 epochs with a batch size of 8 scenes per and a learning rate 0.01, which takes around 15 hours.

3.4.3 Results

Waymo Open Dataset: We first present our object detection results for the vehicle, pedestrian, and cyclist classes on the test set of Waymo Open Dataset in Table 3.1 compared with PV-RCNN [45]. We evaluate our method at both LEVEL_1 and LEVEL_2 difficulty levels. Note that we reproduce the baseline PV-RCNN with a single frame input since they recently adopt two-frames input on the test set. As we can see, PV-RCNN achieves 69.57% and 63.65% on average in

Rel. Trans.	Rep. and Scal. Trans.	Recall_11			Recall_40		
		Easy	Mod	Hard	Easy	Mod	Hard
x	x	88.66	79.07	78.49	91.17	82.61	82.06
✓	x	88.82	83.23	78.64	91.37	84.40	82.34
x	✓	88.93	83.63	78.59	91.72	84.68	82.39
✓	✓	89.28	84.16	79.05	92.29	85.41	82.85

Table 3.5: Ablation studies of transformers in Car class with IoU threshold of 0.7 on KITTI Validation dataset. "Rel. Trans." and "Rep. and Scal. Trans." refer to mutual-relation trans-former of 2 MHSA layer with 4 heads, and multi-representation and multi-scale transformer of 1 MHSA layers with 8 heads, respectively.

LEVEL_1 mAP and LEVEL_2 mAP, respectively, while M3DETR improves them by 1.48% and 1.85%, respectively. Without bells and whistles, our approach works better than PV-RCNN [45]. Furthermore, we compare our framework on the vehicle class for different distances with state-of-the-art methods, including StarNet [80], PointPillars [13], RCD [79], Det3D [81], RangeDet [82] and PV-RCNN [45]. In Table 3.2, M3DETR outperforms PV-RCNN significantly in both LEVEL_1 and LEVEL_2 difficulty levels across all distances, demonstrating the effectiveness of the newly proposed framework. Moreover, we visualize the detection results of M3DETR in Figure 3.4.

Compared with the PV-RCNN shown in Figure 3.5, M3DETR successfully captures the inter- and intra- interactions among input features and effectively helps the model generate high-quality box proposals. To the best of our knowledge, M3DETR achieves the state-of-the-art in the Vehicle class in both LEVEL_1 and LEVEL_2 difficulty levels among all the published papers with a single frame LiDAR input.

We also evaluate the overall object detection performance with an IoU of 0.7

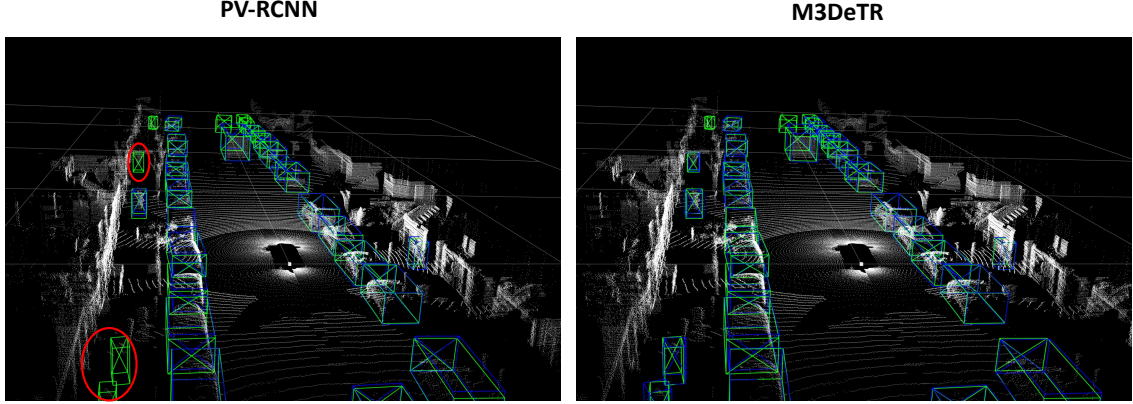


Figure 3.5: We visualize the 3D detection results for the same input point cloud between PV-RCNN (left) and M3DeTR (right) on Waymo Open Dataset. We highlight the false negative boxes from PV-RCNN in red.

for the Vehicle class on the full Waymo Open Dataset validation set as in Table 3.3, further proving that our architecture is more efficient for jointly modeling the input features.

KITTI dataset: We compare our approach with the state-of-the-art methods on the KITTI test set [45, 86]. We compute the mAP on three difficult types of both car and cyclist classes in 3D detection metric. Table 3.4 shows that M3DeTR achieves state-of-the-art performance and outperforms the previous work by a large margin especially on the cyclist class. In particular, HotSpotNet [86] achieves 82.59% in the “easy” categories of 3D detection metric, while M3DeTR improves these results by significant 1.24%.

3.4.4 Ablation Studies

To demonstrate the individual benefits of the multi-representation, multi-scale, and mutual-relation layers of the M3 transformer, we perform ablation experiments and tabulate the results in Table 3.5. All experiments are conducted on the valida-

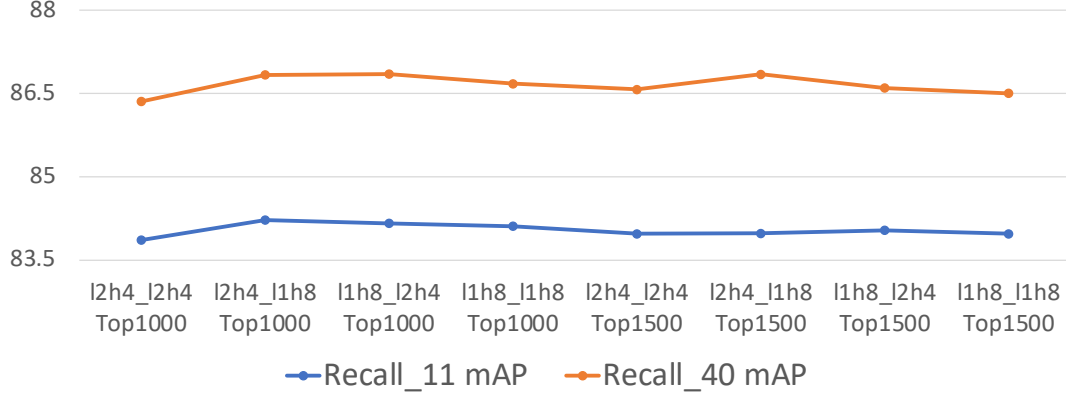


Figure 3.6: Performance comparison of different M3 Transformers variants in Car class on KITTI validation set. We show the mAP results with IoU threshold of 0.7 for both 11 and 40 recall positions. Note that "l" and "h" represent layer number and head dimension of M3 Transformers, respectively. "Top" denotes the number of proposals used for keypoint sampling from RPN stage.

tion set of KITTI dataset.

With the single multi-representation and multi-scale transformer layer, we can achieve 4.07% and 1.71% on the moderate difficulty in car class with 11 and 40 recall positions, respectively compared with the PV-RCNN baseline. On the other side, with the single mutual-relation transformer layer, the performance gain are 4.47% and 1.99% compared with the PV-RCNN baseline. Without hyper-parameter tuning, M3DETR benefits from unifying multiple point cloud representations, feature scales, and model mutual-relations simultaneously which results in the best performance.

3.4.5 Robustness of M3DETR

To demonstrate the robustness of M3DETR to hyper-parameter tuning, we perform a series of tests by varying the sampling size, number of detection heads, and number of transformer encoder layers. We present the results of these tests

in Figure 3.6, where we observe that M3DETR performs consistently well for the “car” category with IoU threshold of 0.7 for both 11 and 40 recall positions on the KITTI validation set.

3.5 Conclusion

In this chapter, we present M3DETR, a novel transformer-based framework for object detection with LiDAR point clouds. M3DETR is designed to simultaneously model multi-representation, multi-scale, mutual-relation features through the proposed M3 Transformers. Overall, the first transformer integrates features with different scales and representations, and the second transformer aggregates information from all keypoints. Experimental results show that M3DETR outperforms previous work by a large margin on the Waymo Open Dataset and the KITTI dataset. Without bells and whistles, M3DETR is demonstrated to be invariant to the hyper-parameters of transformer.

Chapter 4: PointMotionNet: Point-Wise Motion Learning for Large-Scale LiDAR Point Clouds Sequences

4.1 Introduction

Capturing and interpreting motion information from the environment is critical for autonomous systems that must interact with the three-dimensional (3D) world. Although many researchers have emphasized visual sensing for this purpose (e.g., optical flow for 2D images [87]), the growing popularity of 3D sensing techniques has led to a need for improved motion analysis of 3D point clouds.

It is nontrivial to achieve effective motion learning on a sequence of 3D point clouds. A particular issue is that fine-grained point correspondences are often not available between consecutive sweeps due to the spatial and temporal resolution of the sensors. Recent systems, such as FlowNet3D [88], HPLFlowNet [89] and PointPWC-Net[90], have addressed this problem by generalizing optical flow to the 3D domain. However, these techniques rely on learning point-to-point correspondences between consecutive 3D point clouds, and this assumption often does not extend to multiple sweeps. An alternative is MotionNet [91], which handles multiple sweeps by representing 3D point clouds in bird’s-eye-view (BEV) maps. However,

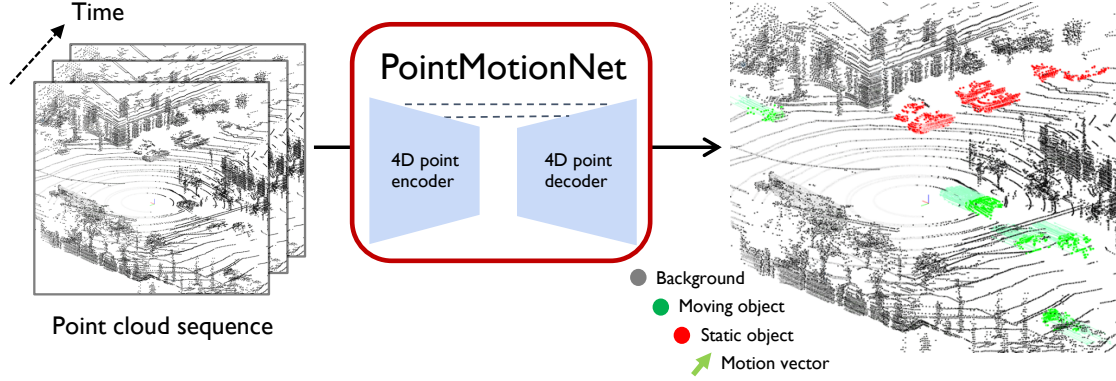


Figure 4.1: PointMotionNet is a point-based spatiotemporal pyramid network to achieve motion learning from a sequence of 3D point clouds. It can handle multiple frames, large-scale open scenes, explicitly model the temporal ordering, and does not suffer from discretization error.

this approach results in inevitable discretization errors. MeteorNet [92] is a recent method that performs motion learning based on PointNet, which utilizes raw points and does not require voxelization. To handle temporal information, it gathers 3D points from multiple sweeps and encodes the time stamp in an additional feature channel. However, MeteorNet does not design specific operations to explicitly handle temporal information. Occupancy Flow [93] and CaSPR [94] explicitly consider the temporal ordering by leveraging tools like continuous normalizing flows and neural ordinary differential equations. However, these methods work well for 3D point sets that are densely sampled, which is often not the case for sparse, outdoor scenes.

To address these issues, we propose a novel method to achieve motion learning that can handle multiple frames, does not suffer from discretization (voxelization) error, explicitly models temporal ordering, and handles large-scale open scenes. Because this method is based on raw points and focuses explicitly on motion learning, we call it *PointMotionNet*; see Figure 4.1. It has a pyramid structure that enables

spatiotemporal feature learning at multiple spatial scales. A core operation in PointMotionNet is a novel point spatiotemporal convolution (*Point-STC*) operation. Instead of looking for point-to-point correspondences, the proposed Point-STC finds the correspondences across time by leveraging the time-invariant neighboring space. It handles 4D points (3D coordinates + time index) in two steps. Step 1 considers spatial information: at each timestamp, the system finds the spatial neighbors for each 4D point and designs trainable multi-kernel functions to aggregate neighboring information to extract spatial features. Step 2 considers temporal information: It concatenates the spatial features sequentially based on their time stamps and then uses a multilayer perceptron to aggregate spatiotemporal features and output final motion features. Overall, PointMotionNet is a point-based architecture that avoids the voxelization issue, explicitly models the temporal ordering, and accommodates multiple sweeps and large-scale scenes.

To validate PointMotionNet, we apply it as a core module to handle two motion-related tasks: point-based motion prediction and multisweep semantic segmentation. For the task of point-based motion prediction, we aim to predict the 3D motion vector for each point in the current frame, which is the displacement from the current 3D position to a future 3D position. We validate the system’s performance empirically on the Argoverse [95], and find that PointMotionNet significantly outperforms others. For the task of multisweep semantic segmentation, the system aims to estimate the semantic and motion labels for each point in the current frame, extending semantic segmentation from a single sweep to multiple sweeps. We validate the system’s performance on the SemanticKITTI [96], and find that PointMo-

tionNet achieves top performance on the leaderboard.

In summary, the main contributions include:

- we design a novel point-based spatiotemporal convolution operation (Point-STC) that extracts spatiotemporal features from raw points;
- we implement two systems based on PointMotionNet to handle two motion-related tasks, point-based motion prediction, and multisweep semantic segmentation; and
- we have conducted extensive experiments on point-based motion prediction and multisweep semantic segmentation. Experimental results show that the proposed approach consistently outperforms previous state-of-the-art methods.

4.2 Related Works

Deep learning on 3D point clouds.. Current approaches can be divided into three categories, based on the input representations: projection-based, voxel-based, and point-based. The projection-based methods, including [97, 98, 99, 100, 101, 102], leverage well-established 2D convolutional networks by projecting 3D point clouds into 2D images. SqueezeSeg [100, 101] projects a 3D LiDAR point cloud onto a spherical surface. RangeNet++ [102] further proposes a nearest-neighbor search for point labels. However, projection-based approaches might suffer from discretization errors that result from the projections. The voxel-based methods, such as [4, 60, 103, 104, 105, 106, 107, 108, 109], focus on mapping points into regular 3D voxels and then applying 2D/3D convolution operations to extract features. PolarNet [108] and Cylinder3D [110] discretize the points based on a polar coor-

dinate system, which makes input points more evenly distributed compared with Cartesian coordinates. However, those methods are characterized by large memory and computational demands.

To directly process raw points, [20, 111] introduce PointNet/PointNet++ for point-cloud classification and segmentation. Many researchers [112, 113, 114] have extended those techniques to 3D object detection. Instead of using the multi-layer perceptron (MLP) from PointNet/PointNet++, KPConv [57] implements an explicit spatial kernel point convolution to directly operate on the input points without any intermediate representation. RandLA-Net [115] demonstrates a new random point sampling method to encode spatial information. In this work, instead of working with a single point cloud, we aim to learn motion information from a temporal sequence of 3D point clouds based on a novel point-based method. Although concurrent work [116, 117, 118] deal with point cloud sequences, [116, 117, 119] is generic for recognition and segmentation, and [118] aims at panoptic segmentation.

Motion learning on 3D point cloud sequences. Modeling and learning motion information for a temporal sequence of 3D point clouds is crucial to scene understanding for dynamic environments. Recently, FlowNet3D [88], MeteorNet [92] and HPLFlowNet [120], and PointPWC-Net [90] leveraged various strategies to generate 3D scene flow between two consecutive frames to represent previous motion displacements. Recent works like [121] can do motion estimation for multiple frames, however, it still follows a sequential scene flow learning strategy between adjacent frames. Besides scene flow estimation, there are several studies of motion prediction jointly trained with

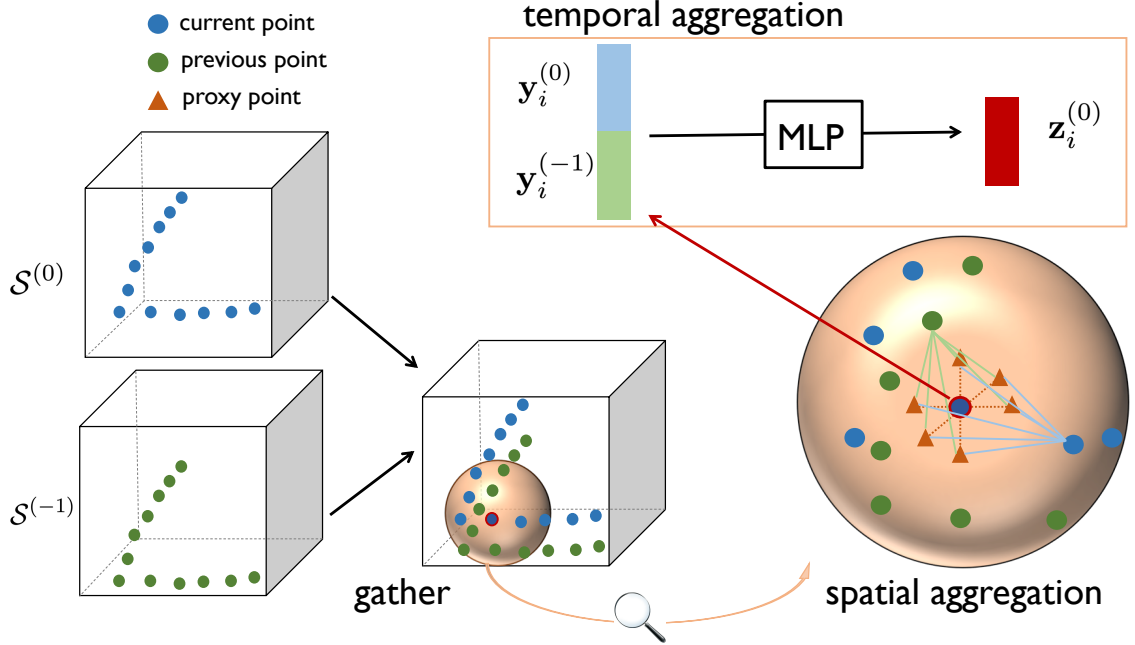


Figure 4.2: **4D point spatiotemporal convolution operation (Point-STC).** To extract spatiotemporal features, we gather all points across time to the same spatial coordinate system. For each query point (with red circle), we find its neighbors (in the orange ball) in each frame. With the set of proxy points (in red triangle) of the query point, we extract spatial features in each individual frame, which follows from (4.1a). We next concatenate the spatial features across all timestamps and use a MLP to obtain the final spatiotemporal features, which follows from (4.1b).

object detection using deep neural networks [122, 123, 124], which typically require bounding-box annotations. MotionNet [91] predicts future motion behaviors from a sequence of point clouds without bounding box annotations. However, MotionNet works with bird’s-eye-view representations, with limited precision by the voxelization. In this work, we aim to capture motion information for each point without any discretization.

4.3 PointMotionNet

Our goal is to propose a generic neural network module to extract motion information from a temporal sequence of 3D point clouds. The motion information could include the motion state (static/moving), the motion displacement and many others. Let $\mathcal{P} = \{\mathcal{S}^{(t)}\}_{t=0}^{-T}$ be a temporal sequence of 3D point clouds, where the set $\mathcal{S}^{(t)} = \{p_i^{(t)}\}_{i=1}^{N_t}$ is the point cloud collected at the t th frame with $p_i^{(t)}$ the i th point in the t th frame. Here $t = 0$ denotes the current frame and $t < 0$ denotes previous frames. Let $\mathbf{p}_i^{(t)} \in \mathbb{R}^3$ be the 3D coordinate of the point $p_i^{(t)}$. Since $\mathbf{p}_i^{(t)}$ is also indexed by time, together with 3D coordinates, we consider it a *4D point* and $\mathcal{P}$ a *4D point cloud*. We aim to design a neural network $f(\cdot)$ to produce motion information for each point in the current frame; that is, $\mathbf{m}_i^{(0)} = f(p_i^{(0)}, \mathcal{P})$, where $\mathbf{m}_i^{(0)}$ could be a scalar to indicate the moving probability, or a 3D vector to indicate the motion displacement.

4.3.1 4D Point Spatiotemporal Convolution

To extract motion features for 4D points, we consider a spatiotemporal convolution operation. Let $\mathbf{x}_i^{(t)}$ be the input features of the point $p_i^{(t)}$. For $p_i^{(t)}$, the 4D

point spatiotemporal convolution (Point-STC) operates as,

$$\mathbf{y}_i^{(t+\tau)} = \sum_{\mathbf{p}_j^{(t+\tau)} \in \mathcal{S}^{(t+\tau)} \cup \mathcal{N}(\mathbf{p}_i^{(t)})} \Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}} \left(\mathbf{x}_j^{(t+\tau)} \right), \quad (4.1a)$$

$$\mathbf{z}_i^{(t)} = \text{MLP} \left(\text{concat} \left(\{\mathbf{y}_i^{(t+\tau)}\}_{\tau=0}^{-T} \right) \right), \quad (4.1b)$$

where $\text{MLP}(\cdot)$ denotes the multilayer perceptron network shared by all the points and $\text{concat}(\cdot)$ denotes the concatenation operation that combines several short vectors to output a long vector, a kernel function $\Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}} \left(\mathbf{x}_j^{(t+\tau)} \right)$ that evaluates the effect from point $p_j^{(t+\tau)}$ to point $p_i^{(t)}$ with τ being time difference and $\mathcal{N}(\mathbf{p}) = \{\mathbf{q} \in \mathbb{R}^3 \mid \|\mathbf{q} - \mathbf{p}\|_2 \leq r\}$ defines a neighboring set of points for point $\mathbf{p}$ with $r \in \mathbb{R}$ a pre-defined radius, reflecting a fixed 3D-ball space. Step (4.1a) is a spatial convolution that aggregates the neighboring information at each single frame and $\mathbf{y}_i^{(t+\tau)}$ denotes the intermediate feature extracted at frame $t+\tau$; and step (4.1b) aggregates temporal features sequentially based on time stamps and $\mathbf{z}_i^{(t)}$ denotes the output feature for $p_i^{(t)}$. Since the sequence length T is usually small in practice, MLP works effectively. Combining (4.1a) and (4.1b), we achieve information aggregation over both the spatial and temporal dimensions; see in Figure 4.2. Note that instead of looking for point-to-point correspondences, Point-STC finds correspondences across time by leveraging the time-invariant neighboring space. Because for each point, the neighboring space is fixed given a radius, we can evaluate the spatial features within this space across time to extract motion features.

To define the kernel function $\Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}}(\mathbf{x}_j^{(t+\tau)})$, instead of directly considering the pairwise effect from $p_j^{(t+\tau)}$ to $p_i^{(t)}$, inspired by [57], we first introduce a few proxy points to represent each 4D point and then consider the effect from $p_j^{(t+\tau)}$ to each of the proxy points. The benefit is to leverage multiple kernels to make the training more stable and effective. The set of proxy points for a 4D point $\mathbf{p}_i^{(t)}$ is $\mathcal{K}(\mathbf{p}_i^{(t)}) = \{\mathbf{p}_i^{(t)} - \mathbf{b}_k \in \mathbb{R}^3\}_{k=1}^K$, where the offsets $\mathbf{b}_k \in \mathbb{R}^3$ are pre-defined so that each proxy point belongs to the vertices or center of a regular polyhedron in 3D space. With this, we consider a multi-kernel function as

$$\Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}}(\mathbf{x}_j^{(t+\tau)}) = \sum_{\delta_k \in \mathcal{K}(\mathbf{p}_i^{(t)})} w_k h(\delta_k, \mathbf{p}_j^{(t+\tau)}) \mathbf{x}_j^{(t+\tau)}, \quad (4.2)$$

where w_k are trainable parameters shared by all the points and the kernel associated with each proxy point is

$$h(\delta_k, \mathbf{p}_j^{(t+\tau)}) = \max \left(0, 1 - \frac{\|\delta_k - \mathbf{p}_j^{(t+\tau)}\|_2}{\sigma} \right) \in \mathbb{R},$$

where $\sigma = r/2$ is a hyperparameter related to the neighboring radius. In (4.2), $h(\delta_k, \mathbf{p}_j^{(t+\tau)})$ reflects the spatial proximity prior between $\mathbf{p}_j^{(t+\tau)}$ and the k th proxy point of $\mathbf{p}_i^{(t)}$ and w_k can further adaptively change the kernel weight. Note that the set of proxy points is similar to rigid kernel points in [57]. The difference is that KP-Conv considers a pair of points at the same timestamp, while (4.2) evaluates a pair of points across time. This is also related to grid points in [125], where they use a fixed global grid; while we recenter the proxy points to each query point, where

the relative displacements are shift-invariant.

The proposed Point-STC (4.1) directly handles raw 4D points without any discretization. Since the operation only takes the neighboring points, we do not need to handle the entire point cloud at the same time; instead, we can split a large-scale point cloud into multiple patches, which are small-scale point clouds, and handle each patch locally. This can significantly ease the computation and make it easier to handle large-scale 3D point clouds.

Compared to recent point-based methods, the proposed Point-STC can effectively operate on a sequence of large-scale 3D point clouds. Specifically, FlowNet3D [88] and HPLFlowNet [120] generate scene flow by learning the point-to-point correspondence between two frames. However, they could only operate on two frames; while the proposed Point-STC can operate on multiple frames by concatenating spatial features from all frames. Although MeteorNet [92] can take multiple frames, it gathers the 3D points from all frames together and encodes the time stamp as a separate feature channel, which is not dedicated to extracting the temporal information; while the proposed Point-STC preserves the temporal ordering by sequentially concatenating spatial features from each frame and uses MLP to fully extract temporal information.

4.3.2 4D Point Pyramid Network Architecture

Here we propose a network architecture to systematically handle a sequence of 3D point clouds. This architecture has a pyramid structure; see Figure 4.3. The left

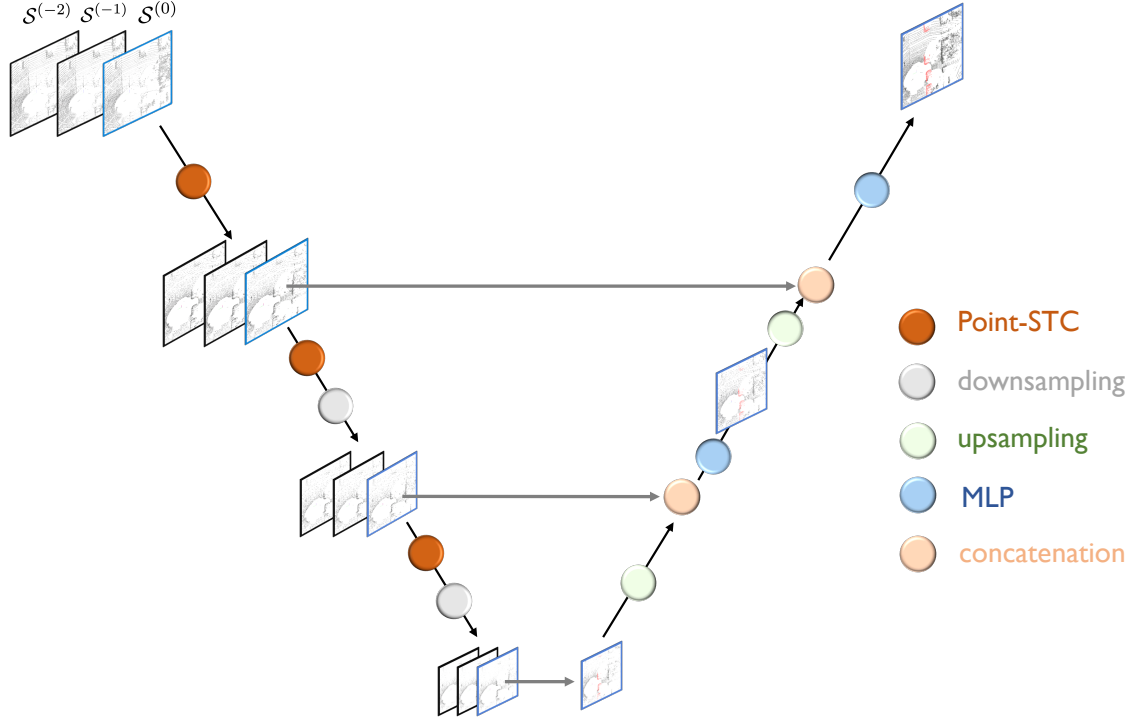


Figure 4.3: **PointMotionNet is a point-based spatiotemporal pyramid network.** Point-STC layers enable spatiotemporal feature learning. Downsampling and upsampling layers enable multiscale feature learning. With the skip connections, the aggregation layers fuse features from both encoder and decoder.

part acts like an encoder that extracts spatiotemporal features at multiple scales; and the right part acts like a decoder that only focuses on aggregating features at the current frame from both the previous layer and the encoder.

Point-STC layer As the main component to extract spatiotemporal features from points, a Point-STC layer implements the 4D point spatiotemporal convolution (4.1). The first layer takes raw points from multiple frames as the input, where the point features are their 3D coordinates. For the deeper layers, the point features are hidden features from the previous layer. For each point, the spatiotemporal neighborhood is implemented by collecting a certain number of the 3D points within a pre-defined spatial distance to this query point at each timestamp. The neighborhood radius is

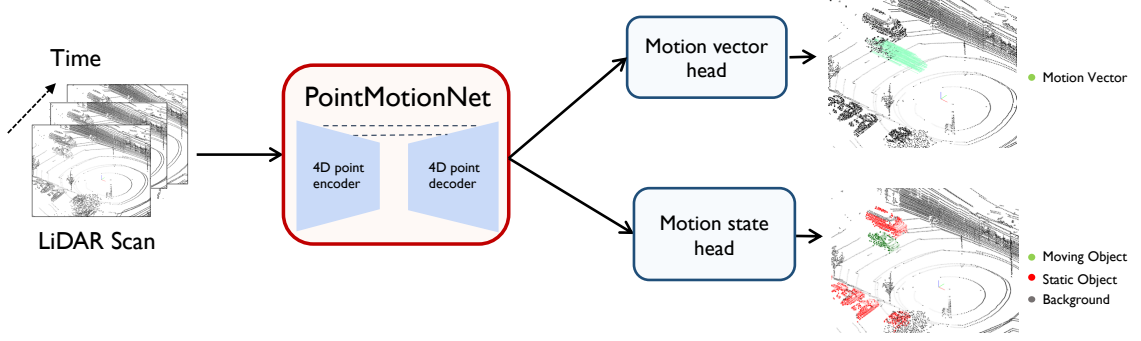


Figure 4.4: **Point-based motion prediction system.** With PointMotionNet as a backbone, the system takes a sequence of 3D point clouds as the input and generates the predicted motion vector and the estimated motion state for each single 3D point in the current frame through the motion vector head and the motion state head. For visualization, we sample the moving points for the motion vector output.

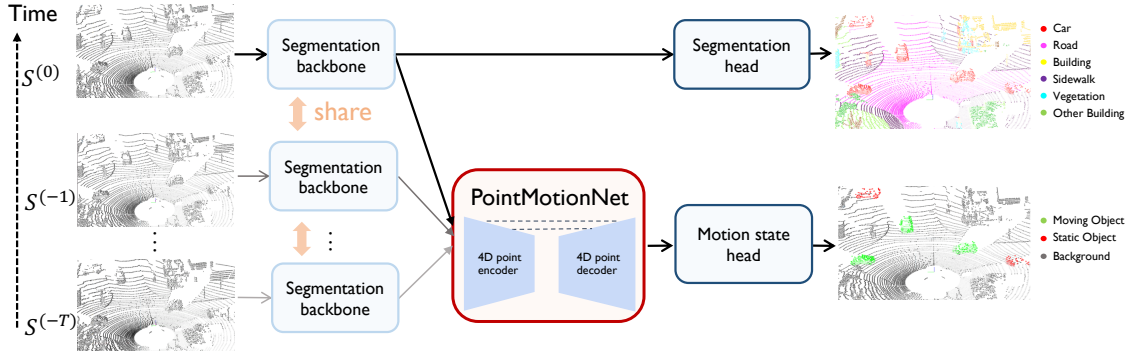


Figure 4.5: **Multisweep semantic segmentation system.** It consists of four parts: the segmentation backbone, extracting semantic features from each individual 3D point cloud, the segmentation head, producing the estimated semantic labels, PointMotionNet, extracting motion features based on semantic features across time stamps, and the motion state head, producing the estimated motion states.

defined as $r_{\ell+1} = 2r_{\ell}$, where ℓ is the network layer index. In a deeper layer, we need a larger reception field for a Point-STC to extract more global features.

Downsampling layer. Since we increase the reception field in a deep layer, the corresponding Point-STC needs to handle many more points. In a large-scale scene, a point cloud is usually irregularly sampled, where nearby points are relatively denser and far-away points are relatively sparser. Uniformly downsampling would make far-away points too sparse. To resolve this issue, we adopt downsampling

based on grids following [57]. We first partition a 3D scene into a series of non-overlapping 3D grids. For a non-empty grid cell with multiple 3D points, we only take their barycenter as their proxy. At each timestamp, we collect the sampled points from all the grids to be a *downsampled 3D point cloud*. In this manner, we would downsample more points nearby and fewer points far away, balancing the spatial distribution of point density. The grid size is defined as $d_{\ell+1} = 2d_{\ell}$, where ℓ is the network layer index. Note that the downsampled 3D points are still irregularly scattered in the 3D space, which is clearly different from the regular voxelization.

Upsampling layer. Each upsampling layer aims to populate point features from a downsampled point cloud to an original point cloud (the previous layer before the downsampling layer). For each 3D point in the original point cloud, we find its closest point in the downsampled point cloud and take its associated point features.

Aggregation layer. Each aggregation layer aims to aggregate point features from both the encoder and the decoder at the same scale. In our pyramid architecture, we use a skip connection to introduce the point features at the current timestamp from the encoder side and concatenate them with the corresponding point features from the upsampled point cloud. We then use a multilayer perceptron to aggregate the concatenated features.

Compared with previous works, the proposed pyramid architecture explicitly preserves the temporal ordering. MeteorNet [92] merges the entire input sequence of 3D point clouds directly to a denser 3D point cloud without ordering them among the temporal axis; while the proposed architecture preserves the complete temporal information and our design decouples the spatio-temporal convolution along spatial

and temporal dimensions. In addition, MeteorNet [92] uses FPS sampling as in PointNet++ [20], while we adopt grid-based sampling, which balances the spatial distribution of point density. [105] relies on high-dimensional sparse convolutions to extract spatiotemporal information from discretized 4D data; while PointMotionNet directly operates on raw points instead of voxels. The proposed Point-STC leverages a fixed set of proxy points, and it covers a general polyhedron compared to regular voxels.

4.4 Motion-Related Systems

In this section, we apply PointMotionNet as a core module to two motion-related systems.

4.4.1 Point-based Motion Prediction

Our aim is to predict a motion vector for each point in the current frame given a sequence of 3D point clouds. Each motion vector of a point is the displacement from the 3D coordinate at the current timestamp to the 3D coordinate at some future timestamp. Here we consider point-level motion, instead of an instance level or a cell level.

Architecture. Figure 4.4 illustrates a point-based motion prediction system. The proposed PointMotionNet takes a sequence of 3D point clouds as the input and produces the intermediate motion features to two output heads: the motion vector head, which predicts the motion vector for each 3D point in the current frame, and the motion state head, which estimates the binary motion state for

Method	View	mIoU _{total} ↑ (%)	mIoU _{static} ↑ (%)	mIoU _{moving} ↑ (%)	Static ↓	Speed ≤ 5m/s ↓	Speed > 5m/s ↓
MotionNet [91]	BEV	94.8	97.8	91.7	0.007	0.755	0.503
FlowNet3D [88]	Point	75.2	90.5	59.9	0.003	1.535	1.834
MeteorNet [92]	Point	93.8	97.4	90.2	0.005	1.014	0.749
PointMotionNet	Point	98.5	99.4	97.6	0.001	0.416	0.445

Table 4.1: **PointMotionNet outperforms in the motion prediction task on Argoverse Validation Set.**

each 3D point in the current frame. As the backbone network, the PointMotionNet has 4 downsampling layers in total. We also set the neighborhood radius in the first Point-STC layer to be $r_\ell = 0.15m$, the grid size in the first downsampling layer be $d_\ell = 0.06m$, the number of the proxy points be 15. The motion vector head is implemented by three fully-connected layers and the motion state head is implemented by one fully-connected layer followed by a softmax layer.

Loss function. We apply the smooth L1 loss L_{vector} to supervise motion vector prediction and use the cross-entropy loss L_{state} for the motion state estimation. We jointly train motion state estimation and motion vector prediction with the overall loss $L = L_{\text{vector}} + 0.1L_{\text{state}}$. In the testing phase, the estimated binary motion states are used to refine the predicted motion vectors: when the estimated motion state is static, the corresponding motion vector is set to zero.

Models	mIoU _{total} ↑ (%)	mIoU _{static} ↑ (%)	mIoU _{moving} ↑ (%)
RangeNet++ [102]	72.0	97.6	46.5
MotionNet [91]	81.4	98.6	64.2
FlowNet3D [88]	59.0	97.1	20.8
MeteorNet [92]	72.0	97.9	46.1
4D-PointNet2 [20]	68.3	97.1	39.5
PointMotionNet	81.8	98.7	64.9

Table 4.2: **PointMotionNet outperforms in motion state estimation on SemanticKITTI.** RangeNet++ and MotionNet are in the range view and BEV, respectively, while the rest are point-based.

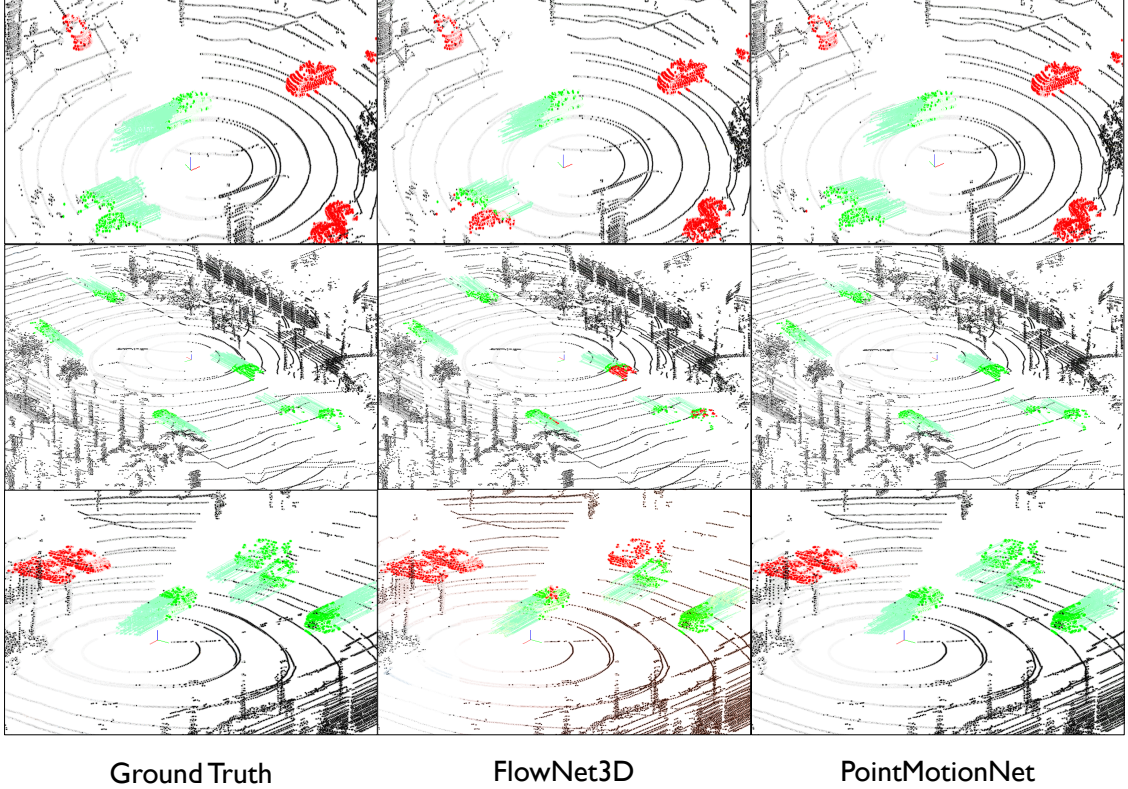


Figure 4.6: **PointMotionNet (right) qualitatively outperforms FlowNet3D (middle) in point-based motion prediction on Argoverse.** Moving points are associated with green motion vectors and the static points are in red; the remaining points are background, marked in grey.

4.4.2 Multisweep Semantic Segmentation

The task of multisweep semantic segmentation aims to estimate the semantic label and motion state for each 3D point in the current sweep given a sequence of point clouds.

Architecture. A straightforward design for multisweep semantic segmentation is to take multiple sweeps as the input and directly estimate all the semantic and motion labels. However, we observe that for foreground categories, such as car, person, and bicyclist, semantic information and motion information are mostly independent. Jointly training the semantic and motion information would potentially

Approach	25-class mIoU	12-class mIoU	car (S)	car(M)	bicyc. (S)	bicyc.(M)	person (S)	person(M)	motor. (S)	motor.(M)	other-veh. (S)	other-veh.(M)	truck (S)	truck(M)
TangentConv [98]	34.1	20.7	84.9	40.3	0.0	30.1	1.6	1.9	0.0	42.2	18.5	6.4	21.1	1.1
DarkNet53 [96]	41.6	24.2	84.1	61.5	0.0	28.9	7.5	0.2	0.0	37.8	20.7	15.2	20.0	14.1
SpSeqnet [107]	43.1	25.5	88.5	53.2	0.0	2.3	6.3	36.2	0.0	0.1	22.7	26.2	29.2	41.2
LatticeNet [99]	45.2	33.7	91.1	54.8	0.0	44.6	6.8	49.9	0.0	64.3	23.1	0.6	65.4	3.5
KP-Conv [57]	51.2	40.5	93.7	69.4	0.0	67.4	21.6	67.5	0.0	47.2	38.6	0.5	42.5	0.5
Cylinder3D [110]	51.5	31.4	93.8	68.1	0.0	60.0	12.9	63.1	0.1	0.4	37.6	0.1	41.2	0.0
PointMotionNet	54.8	39.2	93.8	71.4	6.2	62.2	13.7	61.8	23.4	15.1	38.1	29.9	44.2	10.5

Table 4.3: PointMotionNet outperforms in multisweep semantic segmentation on SemanticKITTI test set. Note that S represents the static foreground category, and M denotes the moving foreground category.

harm the performances on both tasks. Moreover, previous sweeps might not significantly benefit the point-wise semantic labeling of the current sweep because the points across sweeps do not have a clear point-to-point correspondence. Therefore, our design rationale is to split multisweep semantic segmentation into two separated tasks: semantic segmentation based on a single sweep and motion state estimation based on multiple sweeps.

Figure 4.5 illustrates a multisweep semantic segmentation system. The overall architecture consists of four parts: the segmentation backbone, the segmentation head, PointMotionNet and the motion state head. The segmentation backbone consumes each individual sweep and learns per-point semantic features. All sweeps share the same backbone network. The segmentation head takes semantic features of all the points in the current sweep to generate the per-point semantic labels. It is implemented by a fully-connected layer followed by a softmax layer. The third component, PointMotionNet, learns motion features based on semantic features from

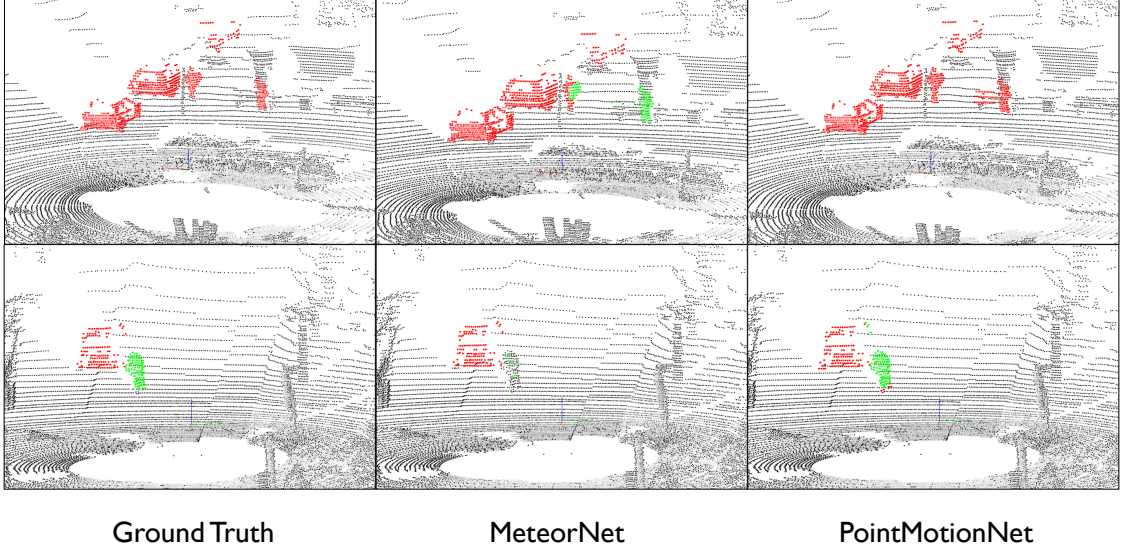


Figure 4.7: PointMotionNet (right) visually outperforms MeteorNet (middle) in motion state estimation on SemanticKITTI. For six foreground classes, moving points are in green and static ones are in red; for the remaining background classes, points are in grey.

the segmentation backbones across all sweeps. The motion state head takes the motion features to estimate the motion state for each point in the current sweep. It is implemented by a fully-connected layer followed by a softmax layer.

proxy points(k)	d_1	temporal op.	# frames (n)	mIoU _{total}	mIoU _{static}	mIoU _{moving}	Static	Speed $\leq 5\text{m/s}$	Speed $> 5\text{m/s}$
15	0.06	MLP	3	0.985	0.994	0.976	0.001	0.416	0.445
6	0.06	MLP	3	0.981	0.992	0.970	0.002	0.461	0.543
15	<i>0.24</i>	MLP	3	0.983	0.993	0.974	0.002	0.398	0.461
15	0.06	<i>Conv.</i>	3	0.976	0.990	0.962	0.002	0.558	0.782
15	0.06	MLP	<i>4</i>	0.982	0.996	0.969	0.001	0.341	0.422

Table 4.4: Ablation Study of PointMotionNet on Argoverse Validation Set. We vary several hyper-parameters: number of proxy points k , initial sampling resolution d_1 , temporal operation, and input frames number n .

Loss function. We adopt the cross-entropy losses for both multi-category semantic segmentation and the binary motion state estimation. Based on the groundtruth of the semantic category, we apply a foreground mask to the motion state estimation, so that only the losses of foreground points are counted. The

overall loss function combines the semantic segmentation loss and the motion state loss. In the training phase, we could also use a pre-trained segmentation backbone to focus on motion training and accelerate the training time. In the inference phase, for estimated background categories, the system would set the corresponding estimated motion states to be static.

4.5 Experiments Results

In this section, we validate PointMotionNet both quantitatively and qualitatively on Argoverse and SemantiKITTI.

4.5.1 Point-based Motion Prediction

Dataset. Argoverse [95] is an autonomous driving dataset for 3D tracking and forecasting. We follow the official split to generate 65 logs for training and 24 logs for validation. Since there is no direct labeling for motion information, we generate the ground truth based on 3D bounding boxes.

Baselines. We compare PointMotionNet with several state-of-the-art methods, including MotionNet [91], which is a BEV-based method, and MeteorNet [92], which is a point-based method. We adjust the output head of MeteorNet for future motion vector prediction and motion state estimation. We also consider a 3D scene flow method, FlowNet3D [88], which only takes 2 frames. We train it to estimate the 3D scene flow between one past frame and the current frame, and then lift the predictions to motion displacements in future frames using a linear motion model.

Experimental setup. The inputs are 3 sequential frames of point clouds $\mathcal{S}^{(t-0.2s)}, \mathcal{S}^{(t-0.1s)}, \mathcal{S}^{(t)}$, which are synchronized to the same coordinate system. The final motion prediction contains a binary motion state, i.e. static or moving, together with the displacements from current time stamp to the next 5 future time stamps $\{t + \tau\}_{\tau=0.1s}^{0.5s}$.

Since Argoverse dataset [95] doesn't provide motion vectors information, we generate motion vectors annotations by leveraging the labels from Argoverse 3D Tracking v1.1. We end with 6561 samples for training, and 2507 samples for validation. The preprocessing could be divided into two steps:

For the Ego-motion compensation, each original frame is taken from its own local coordinate while the vehicle is moving forward. After selecting a short video clip containing several sequential scans, we set a frame as current frame, and synchronize all other frames to current frame, together with the original 3D bounding boxes. More specifically, given the vehicle poses registered in global map, M_t of current frame S_{local}^t , $M_{t+\tau}$ of another frame $S_{local}^{t+\tau}$, we could compute the synchronized point cloud coordinates $S_{synced}^{t+\tau} = M_{t+\tau}^{-1} M_t S_{local}^{t+\tau}$, here M is 4×4 homogeneous transformation matrix, while S is padded with 1 to create homogeneous coordinates. The same operation is applied to the center and 6 corners of each 3D bounding box in frame $t + \tau$ to get synchronized pose in current frame;

On the other hand, for the motion vector assignment, Using the unique tracker ids, we could associate poses between current frame and synchronized future frames for each instance belonging to current frame, and compute the relative 6D pose

transformation between them. Then for each bounding box in current frame, we will first rotate it to align with xyz axis, and the same transformation is applied to the whole point cloud. So we could check the membership with the aid of aligned axis to find all points that are inside the 3D bounding box. Finally we apply the relative transformation to all points belonging to current bounding box, the subtraction between transformed coordinates and original coordinates will result in the 3D motion vector annotations. We apply the same process to all 5 future frames.

Results. We adopt the mean intersection-of-union (mIoU) [96] to evaluate motion state estimation for static and moving points. For the motion vector prediction, we evaluate on two speed intervals, which shows performances on both slow and fast motions. Since MotionNet will only predict 2D motion vector at each cell, we reproject its predictions back to each point, and quantitatively evaluate all methods on xy plane only. Table 4.1 shows that as a point-based method, PointMotionNet outperforms its competitors. Figure 4.6 shows that PointMotionNet visually outperforms FlowNet3D. Due to the limitation of fusing only two consecutive frames, FlowNet3D tends to easily fail to predict the motion state of objects either too closer or the too far away from the LiDAR sensor.

Ablation Studies. Table 4.4 considers several variants of PointMotionNet. We see that i) Less proxy points results in slightly worse results; ii) increasing the grid resolution up to 0.24m in downsampling makes minor effects; iii) MLP works significantly better than convolution in the temporal aggregation (4.1b); and iv) increasing the number of sweeps improves the performance.

proxy points(k)	# frames	mIoU _{total} ↑	mIoU _{static} ↑	mIoU _{moving} ↑	Static ↓	Speed ≤ 5m/s ↓	Speed > 5m/s ↓
15	3	0.985	0.994	0.976	0.001	0.416	0.445
15	2	0.980	0.992	0.969	0.001	0.453	0.568
15	4	0.982	0.996	0.969	0.001	0.341	0.422
12	3	0.984	0.994	0.975	0.002	0.395	0.489
9	3	0.982	0.992	0.971	0.002	0.406	0.559
6	3	0.981	0.992	0.970	0.002	0.461	0.543
3	3	0.964	0.985	0.942	0.003	0.588	0.847

Table 4.5: Additional Ablation Study of **PointMotionNet** on Argoverse Validation Set. We further test different number of proxy points k .

To better explore the performance bound of our method, and understand how different parameters would affect the model performance, we designed several more experiments over the choice of input frame number, number of proxy points. The results are shown in Table 4.5.

Number of frames. Using more frames as input would bring in richer information of the 3D scene regarding both geometric features and motion dynamics, but also more complexity to the learning process. We tested our model’s performance on motion estimation by inputting 2-4 sequential frames, while other parameters are fixed. As demonstrated by Table 4.5, our model could still work well when taking 4 frames as input, and it actually achieves the best performance under the fact that more computation is involved. To achieve a better balance between efficiency and accuracy, we consider 3 frames as our final setting for our motion estimation task.

Number of proxy points. Proxy points interact with all spatiotemporal neighbors at multiple scales in our pyramid network, and their number will affect the model performance. By decreasing the proxy points number from 15 to 3, we found that the accuracy drops consistently on future motion vector regression, but we don’t necessarily always need that much proxy points. Using 9 or 12 proxy points could

still work very well for those slow-moving points.

4.5.2 Multisweep 3D Semantic Segmentation

Dataset. SemanticKITTI [96] is another one of the largest datasets for LiDAR-based semantic segmentation. There are 22 labeled sequences consisting of over 43,000 input laser scans. The experiments follow the standard protocol on the training, validation, and test sequences. Note that we manually remove the first 130 corrupted validation scans with inconsistent annotation when we evaluate PointMotionNet and previous architectures. The multiple-scan benchmark evaluates mIoUs on the 25 classes, including six moving and non-moving foreground classes, car, bicyclist, person, motorcyclist, other-vehicle and truck.

Baselines. We compare PointMotionNet with several state-of-the-art methods, including KP-Conv [57], and recent cylinder coordinates-based Cylinder3D [110].

Experimental setup. We use the SGD optimizer with momentum 0.98 and weight decay of 10^{-3} and an initial learning rate of 0.01. We train the PointMotionNet and the heads for 500 epochs. Following MotionNet [91], we use a consistency loss to enforce that the points belonging to the same object have the same motion state.

Results. First, we validate PointMotionNet in the point motion state estimation task on SemanticKITTI validation set. We compute the overall $\text{mIoU}_{\text{total}}$, $\text{mIoU}_{\text{static}}$ for static points and $\text{mIoU}_{\text{moving}}$ for moving points. Table. 4.2 shows that PointMotionNet outperforms its competitors. Second, we visualize the motion

state estimation results for the foreground classes on SemanticKITTI validation set. Figure 4.7 shows that with the same semantic segmentation backbone, PointMotionNet outperforms MeteorNet. Third, we evaluate the overall multisweep semantic segmentation performance on the SemanticKITTI test server. Table 4.3 shows that the proposed PointMotionNet could improve the overall performance on 25 classes semantic segmentation using a simple single-frame semantic segmentation backbone.

4.6 Conclusion

The proposed PointMotionNet is a novel point-based method to achieve motion learning. It handles multiple frames and large-scale scenes, avoids discretization, and explicitly learns from the temporal ordering. We further integrate PointMotionNet into two systems to handle point-based motion prediction and multisweep semantic segmentation. Experimental results show that PointMotionNet outperforms baselines on Argoverse and SemanticKITTI.

Chapter 5: TAG: Boosting Text-VQA via Text-aware Visual Question-answer Generation

5.1 Introduction

Visual question answering (VQA) task [126] aims at inferring the answer to a question based on a holistic understanding of an image. It facilitates many AI applications such as robot interactions [127], document analysis [128] and assistance for visually impaired people [129]. Text-VQA specifically addresses question answering requests where reasoning text in an image is essential to answer a question. It is a more challenging task in a sense that it requires not only understanding the question and the visual context, but also the embedded text in an image [3]. To achieve this goal, Text-VQA methods [1, 53, 130] aim at studying the interactions among question words, visual objects, and scene text in an image. Recent approaches have focused on either improving transformer-based architectures [52] in a multi-modal manner [53, 130, 131, 132, 133, 134, 135], or adopting pre-training using additional large-scale data [1] to further boost their model performance. All of these methods heavily rely on annotations of question-answer (QA) pairs for model training. Intuitively, the more annotated pairs are leveraged, the better



Figure 5.1: **Left:** statistics of the numbers of annotated QA pairs (blue) and scene text words (orange) for each image in the TextVQA training set [3]. Clearly, the scene text words are not fully leveraged in the annotations. **Right:** an example of a training image with more than 5 scene text words, which is typical. Best viewed in color.

performance a model can achieve. Thanks to the development of text-related VQA datasets [128, 136, 137, 138], Text-VQA has achieved rapid progress.

However, the amount of Text-VQA annotations available is still limited due to the sparse labeling of QA pairs in recent datasets. Consider for example the TextVQA dataset [3] whose statistics are illustrated in Figure 5.1. It shows that only one or at most two QA pairs are annotated in the training images. Meanwhile, we also compute the number of text words presented in each image¹ and observe that most of the images contain at least 5 text words. This observation indicates that scene text is not fully utilized in the annotations, and hence not fully leveraged by recent methods. A natural question would be – can we fully take advantage of text words in images without incurring extra annotation costs?

As illustrated in Figure 5.2, we propose to tackle the problem by learning

¹We compute the average of different OCR tokens acquired by the Microsoft-OCR system [1] for the entire training set.

to generate large-scale and diverse text-related QA pairs from existing Text-VQA datasets, using the generated QA pairs to expand the training set and ultimately improving Text-VQA models. Towards this end, we introduce TAG, a text-aware QA generation model, that generates novel text-related QA pairs at scale. It takes text words (the answer) as one of the inputs and aims at generating a question corresponding to this answer by leveraging the rich visual and scene textual cues. TAG is trained using the originally annotated QA pairs and adapts to generate new QA pairs containing scene text words in images that are not utilized in original annotations. No extra human annotation is required in our framework, so the size and diversity of the training data could be easily and largely increased. Since our generation process is disentangled with the training of Text-VQA models, our generated QA pairs can be used by most of the recent methods.

In summary, we introduce a simple yet efficient text-aware generation approach, which automatically and efficiently generates new QA pairs to improve the performance of the current Text-VQA methods. The main contributions of our work are three-fold:

- We identify and analyze possible deficiencies of current Text-VQA datasets—sparse annotations of QA pairs - and propose to better utilize unused scene text information within each image to improve the model performance.
- To the best of our knowledge, TAG is the first method that explores scene text-related QA pairs generation for improving Text-VQA tasks without additional labeled data.

- We consistently demonstrate the effectiveness of our method with two recent Text-VQA models on two Text-VQA datasets. The experimental results suggest that the existing Text-VQA algorithms can benefit from training with the high-quality and diverse QA pairs generated by our method.

5.2 Related Work

5.2.1 Text-related VQA

To study and evaluate the Text-VQA task, several scene text-based datasets are introduced, including VizWiz [136], OCR-VQA [128], TextVQA [3], and ST-VQA [138]. With the help of these datasets, numerous approaches have been proposed in recent years which increasingly improve Text-VQA performance [1, 3, 53, 130, 131, 132, 133, 134, 135, 139, 140, 141, 142, 143, 144]. LoRRA [3] is an early work that extends the original VQA models [139, 140] with an extra OCR attention branch to select the answer from either a fixed word vocabulary or OCR tokens. Recent studies [72, 73, 109, 145, 146, 147, 148, 149, 150, 151, 152, 153] show the benefits of transformer for different vision, language and speech tasks. M4C [53] develops a transformer-based architecture to fuse different input modalities and iteratively predicts answers through a multi-step answer decoder. Inspired by M4C, more transformer-based models have been proposed with varied structure modifications. Among them, CRN [132] constructs a graph network to model the interactions between text and visual objects. LaAP-Net [133] predicts a bounding box to explain the generated answer. SSBaseline [134] proposes to split the OCR token features

into separate visual and linguistic attention branches. SMA [131] reasons over structural text-object graphs and produces answers in a generative way. LOGOS [135] introduces a question-visual grounding pre-training task to connect question text and image regions. SA-M4C [130] builds a spatial graph to explicitly model relative spatial relations between visual objects and OCR tokens. TAP [1] presents three text-aware pre-training tasks to align representations among scene text, text words, and visual objects.

However, most of the existing works focus on designing sophisticated architectures that leverage the annotated text in an image and overlook the rich text information that is underused by the annotated QA activities. We fully explore the embedded scene text in images and explicitly generate novel QA pairs that can be used to boost the performance of downstream Text-VQA models.

5.2.2 Data Augmentation for VQA

Data augmentation has been demonstrated to be an effective approach to improve the performance of the VQA task [154, 155, 156, 157, 158, 159, 160]. Kafle et al. [154] propose to generate new questions using the existing semantic segmentation annotations and templates. Shah et al. [155] introduce a cycle-consistent scheme generating question rephrasings to make VQA models more robust to linguistic variations. Ray et al. [156] propose a consistency-improving data augmentation module to make VQA models answer consistently. Agarwal et al. [157] explore data augmentation to improve the VQA model’s robustness to semantic visual variations.

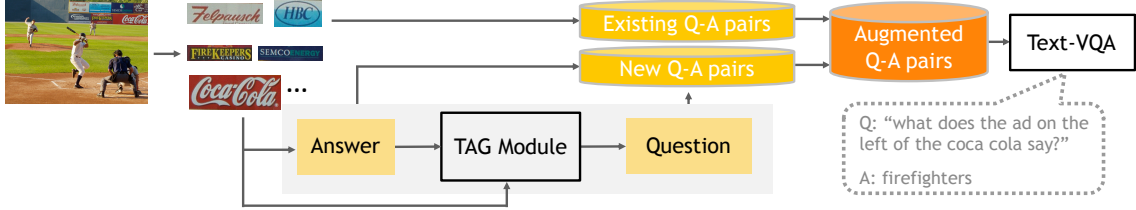


Figure 5.2: **The proposed Text-VQA framework.** It consists of two parts: a text-aware visual question-answer generation module (TAG), followed by a downstream Text-VQA model. TAG is based on a multi-modal transformer architecture, which takes a single image and text words (the answer) as input, and outputs a newly-generated question corresponding to the input answer. The generated QA pairs from TAG together with the originally labeled data are subsequently used to train Text-VQA models, leading to better Text-VQA performance.

Tang et al. [158] use data augmentation to inject proper inductive biases into the VQA model. Wang et al. [159] introduce a generative model for cross-modal data augmentation on VQA. Kant et al. [160] adopt the contrastive loss to make the VQA model robust to linguistic variations in generated questions. However, these approaches are designed for the traditional VQA systems that do not emphasize the importance of scene text in their QA tasks. Our method is tailored for the problem of Text-VQA. It takes advantage of the underexploited scene text in images and enlarges the training samples by generating novel text-related QA pairs without the extra labeling cost.

5.3 Our Approach

The proposed framework is illustrated in Figure 5.2, which consists of a transformer based text-aware visual QA generation module named TAG, followed by a downstream Text-VQA model. Our core module, TAG, carries out text-aware data augmentation tailored for the Text-VQA task and generates novel QA pairs

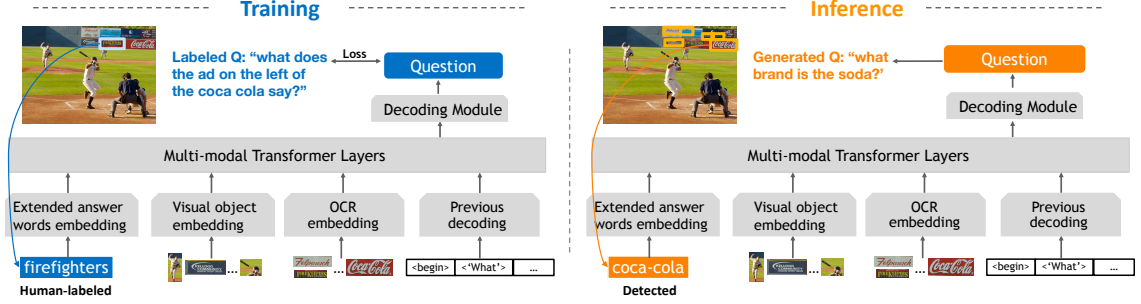


Figure 5.3: **The illustration of our proposed TAG.** High-dimensional feature representations are first extracted for three modalities, including extended answer words, visual objects, and scene text. Then, a multi-modal transformer is used to model the interactions of different modalities. Finally, a decoding module is used to predict the question corresponding to the answer through iterative decoding with an auto-regressive mechanism. **Left:** Originally labeled QA pairs are used for training. **Right:** During inference, detected OCR words are used as a novel answer to generate a question. Best viewed in color.

by leveraging underused scene text in an image. After the TAG module generates a large amount of new QA pairs, we directly augment the training data by combining the generated set and the originally labeled set. The augmented set is used by the downstream Text-VQA models to boost the model performance.

The workflow of our method is as follows. Given an image, an OCR system and an object detector are used to detect scene text and visual objects, respectively. As illustrated in Figure 5.3, our TAG takes the scene text words of interest (the answer words), the visual objects and all the detected OCR tokens in the image as inputs and generates a question explicitly corresponding to the answer. Specifically, the answer words, visual objects, and all the OCR tokens are first represented by high-dimensional features (Section 5.3.1). Then, the multi-modality information is fully aggregated through a transformer architecture with the attention mechanism (Section 5.3.2). Finally, the enriched features are used to predict a question to the answer through iterative decoding in an auto-regressive manner (Section 5.3.3).

5.3.1 Multi-modality Feature Embeddings

We describe the feature embedding strategy of our work. The answer words, detected visual objects, and all the detected OCR tokens are embedded as high-dimensional features and then projected into a common d-dimensional embedding space.

Embedding of extended answer words. We follow [1] to use an extended representation to embed answer words. Given an answer input w^{ans} , we extend the words with labels of objects w^{obj} (detected from the object detector) and scene text OCR words w^{ocr} (generated from the OCR system) as a set of K text words. A trainable BERT-style model [145] is adopted to extract the embedding of those text words, $\mathbf{F}^{ans} = \{\mathbf{f}_1^{ans}, \mathbf{f}_2^{ans}, \dots, \mathbf{f}_K^{ans}\}$, where $k = \{1, 2, \dots, K\}$, and $\mathbf{f}_k^{ans}$ is the d-dimensional feature vectors for k_{th} text word. The embeddings of the set of words are used jointly as the feature of the answer.

Embedding of detected objects. Following M4C [53], we run a pre-trained 2D object detector, Faster R-CNN [23] to localize M visual objects for each image. Two visual object features, including appearance and location features are extracted and then combined together to encode each detected object, $\mathbf{F}^{obj} = \{\mathbf{f}_1^{obj}, \mathbf{f}_2^{obj}, \dots, \mathbf{f}_m^{obj}\}$, where $m = \{1, 2, \dots, M\}$ and $\mathbf{f}_m^{obj}$ is the projected d-dimensional feature vectors for m_{th} object. Specifically, the feature vector output of the object detector (from the fc7 layer) is used to encode the appearance feature and the relative bounding box coordinates are employed as the location feature.

Embedding of OCR tokens. For the N OCR tokens extracted by an OCR system,

we construct the embedding for each token containing both its visual and text feature. The visual feature extraction follows the strategy of the above visual object embedding. Additionally, FastText [161] and PHOC features [162] are extracted for each OCR token to represent its textual cues. A rich OCR representation is thus obtained, $\mathbf{F}^{ocr} = \{\mathbf{f}_1^{ocr}, \mathbf{f}_2^{ocr}, \dots, \mathbf{f}_n^{ocr}\}$, where $n = \{1, 2, \dots, N\}$ and $\mathbf{f}_n^{ocr}$ is the projected d-dimensional feature vectors for n_{th} OCR token.

5.3.2 Multi-modality Fusion

Once the feature embedding representation from individual modality, $\mathbf{F}^{ans}$, $\mathbf{F}^{obj}$ and $\mathbf{F}^{ocr}$ are generated, they are able to dynamically attend to each other from a stack of L transformer layers [52] as shown in Figure 5.3. The input sequence to the multi-modal transformer is $\mathbf{F} = \{\mathbf{F}^{ans}, \mathbf{F}^{obj}, \mathbf{F}^{ocr}\}$. The multi-modal transformer leverages feature embeddings from different modalities and accordingly models interaction among them through the multi-head attention mechanism. From the output of the multi-modal transformer, we extract a sequence of d-dimensional feature vectors for each modality, which is an enriched feature from a joint semantic embedding space.

5.3.3 Text-aware Visual Question Prediction

With the enriched embedding from the multi-modal transformer, the multi-step decoding module predicts a question to the input answer and iteratively generates the question word by word. At each iterative decoding step, we feed in an

embedding of previously predicted words, and then the next output word could be either selected from the fixed frequent word vocabulary or from the extracted OCR tokens. Similar to [1, 53], two special tokens $\langle begin \rangle$ and $\langle end \rangle$ are appended to the word vocabulary, where $\langle begin \rangle$ is used as the input to the first decoding step and $\langle end \rangle$ indicates the end of the decoding process. Alternatively, the decoding process ends when the maximum number of steps T is reached. Our decoding module is based on a dynamic pointer network [163], which allows both copying words via pointing, and generating words from a fixed vocabulary obtained from the training set.²

During training, our TAG is supervised with the binary cross-entropy loss applied using the originally annotated QA pairs and adapts to generate novel QA pairs during generation. During the QA pairs generation process, we pass an input answer, each of which is selected from the extracted OCR tokens, into the TAG module and generate the corresponding question accordingly. In this way, the generated QA pairs cover a diverse set of scene text which was not directly exploited in the original annotation set. For answer selection, we perform a simple yet efficient strategy that is feeding the OCR token with the largest bounding box as the answer candidate to the proposed TAG. The intuition behind this design is that the scene text with the largest bounding box region is likely to encode semantically meaningful information for scene text-based understanding and reasoning. Also, scene text with a larger font size has a higher chance to be detected correctly without

²Our decoding module is implemented following the implementation of the decoding module in TAP [1]. We keep their default hyper-parameters except otherwise noted.

recognition error in general. As we illustrate in our experiments, our simple design facilitates a better understanding of the visual content and provides promising Text-VQA performance. Note that, more high-quality QA pairs could be continuously augmented with a more sophisticated answer-candidate selection strategy. We leave this direction as future work.

5.4 Experiments

We evaluate TAG both qualitatively and quantitatively on the TextVQA [3] and the ST-VQA [138] datasets. We first present a brief overview of the datasets and implementation details. Then, we empirically validate the effectiveness of our proposed method by comparing it with the existing Text-VQA approaches. Our framework clearly outperforms previous work by a significant margin on both datasets.

5.4.1 Datasets and Evaluation Metrics

TextVQA dataset [3] is a widely used benchmark for the Text-VQA task. It consists of 28,408 images sourced from the Open Images dataset [164], with human-annotated questions that require reasoning over text in the images. We follow the standard split on the training, validation and test sets [1, 53]. For each question, the answer prediction is evaluated based on the soft-voting accuracy of 10 human-annotated answers [1, 53, 165].

ST-VQA dataset [138] is another popular dataset for the Text-VQA task. It

contains 23,038 images from multiple sources including ICDAR 2013 [166], ICDAR 2015 [167], ImageNet [168], VizWiz [136], IIIT STR [169], Visual Genome [170], and COCO-Text [171]. The standard evaluation protocol on the ST-VQA dataset consists of accuracy and Average Normalized Levenshtein Similarity (ANLS) [138].

5.4.2 Implementation Details

We use PyTorch to implement our TAG³ that is used to augment the initially labeled data. The augmented dataset is used to improve two recent Text-VQA models, M4C [53] and TAP [1]. M4C[†] is a variant version [1] of M4C, where the detected object labels and scene text tokens are also included in the text encoder, which further improves the performance.



Figure 5.4: We visualize the examples of the generated QA pairs (bottom in orange) by TAG module compared with the original annotated QA pairs (top in blue) on the TextVQA training set. "Q" and "A" refer to question and answer, respectively. Best viewed in color.

TAG. We project the multi-modality feature embedding to be $d = 768$ channels. We extract the embedding of extended answer words using the same trainable structure as $BERT_{BASE}$ [145]. Specifically, we initialize the weights of the model from the first three layers of $BERT_{BASE}$ and eliminate the separate text transformer. In

³Our implementation is built upon the codebase: <https://github.com/microsoft/TAP>.

Method	OCR system	Extra Data	Val Acc.	Test Acc.
CRN [132]	Rosetta-en	×	40.39	40.96
LaAP-Net [133]	Rosetta-en	×	40.68	40.54
SMA [131]	SBD-Trans OCR	×	43.74	44.29
SSBaseline [134]	SBD-Trans OCR	×	43.95	44.72
LOGOS [135]	Microsoft-OCR	×	50.79	50.65
M4C [†] [53]	Microsoft-OCR	×	44.50	44.75
M4C [†] + TAG	Microsoft-OCR	×	45.68	45.96
TAP [1]	Microsoft-OCR	×	49.91	49.71
TAP + TAG	Microsoft-OCR	×	52.54	52.57
LaAP-Net [133]	Rosetta-en	ST-VQA	41.02	41.41
SA-M4C [130]	Google-OCR	ST-VQA	45.40	44.60
SMA [131]	SBD-Trans OCR	ST-VQA	44.58	45.51
SSBaseline [134]	SBD-Trans OCR	ST-VQA	45.53	45.66
LOGOS [135]	Microsoft-OCR	ST-VQA	51.53	51.08
M4C [†] [53]	Microsoft-OCR	ST-VQA	45.22	-
M4C [†] + TAG	Microsoft-OCR	ST-VQA	46.33	46.38
TAP [1]	Microsoft-OCR	ST-VQA	50.57	50.71
TAP + TAG	Microsoft-OCR	ST-VQA	53.63	53.69

Table 5.1: **TAG’s outperformance on the TextVQA dataset when trained on original and augmented dataset under two settings.** Note that M4C[†] is the improved version from [1].

terms of the object embedding, a Faster R-CNN object detector [23] pre-trained on the Visual Genome dataset [170] is adopted to extract $M = 100$ top-scoring objects on each image and represents each object with its appearance and location features. The Microsoft-OCR system [1] is used to extract OCR tokens per image with each token represented with its appearance, location, FastText [161] and PHOC features [162]. The multi-modality fusion module is a four-layer transformer with 12 attention heads, which has the same hyper-parameters as $BERT_{BASE}$. We use $T = 30$ decoding steps to predict the output question word by word in an auto-regressive manner.

Training parameters. Experiments are conducted on 4 Nvidia P6000 GPUs. We

Method	Extra Data	Val Acc.	Val ANLS	Test ANLS
CRN [132]	×	-	-	0.483
LaAP-Net [133]	×	39.74	0.497	0.485
SMA [131]	×	-	-	0.486
SA-M4C [130]	×	42.23	0.512	0.504
SSBaseline [134]	×	-	-	0.509
LOGOS [135]	×	44.10	0.535	0.522
M4C [†] [53]	×	42.28	0.517	0.517
M4C [†] + TAG	×	44.52	0.540	0.529
TAP [1]	×	45.29	0.551	0.543
TAP + TAG	×	50.18	0.595	0.586
SSBaseline [134]	TextVQA	-	-	0.550
LOGOS [135]	TextVQA	48.63	0.581	0.579
M4C [†] [53]	TextVQA	46.60	0.560	0.552
M4C [†] + TAG	TextVQA	48.69	0.579	0.571
TAP ^{††} [1]	TextVQA, TextCaps, OCR-CC	50.83	0.598	0.597
TAP + TAG	TextVQA	53.53	0.620	0.602

Table 5.2: **Our framework outperforms prior work on the ST-VQA dataset.** Note that M4C[†] is the improved version from [1]. Specifically, our model with TextVQA outperforms the SOTA approach TAP^{††} [1] that is pre-trained with extra large-scale data from external TextCaps [2] and OCR-CC [1] datasets.

train TAG for 24K iterations with a batch size of 128. We adopt the Adam optimizer [34] with a learning rate of 1e-4 and a staircase learning rate schedule, where we multiply the learning rate by 0.1 at 14K and at 19K iterations. We keep the original parameter settings of downstream Text-VQA models except that we increase their maximum iteration in proportion to the increased size of the augmented data to accommodate the enlarged number of training samples. Table. 5.3 overviews the hyper-parameter settings of TAG.

5.4.3 Main Results

TextVQA dataset. To perform a fair comparison with prior work, we conduct experiments in both the constrained setting (top part of Table 5.1) and the uncon-

Hyper-parameters	Value
Max length of answer words w^{ans}	20
Max length M of visual objects w^{obj}	100
Max length N of scene texts w^{ocr}	100
Max length K of extended answer words	220
max length T of decoding step	30
optimizer	Adam
batch size	128
max iterations	24K
base learning rate	1e-4
learning rate steps	14K, 19K
learning rate decay	0.1

Table 5.3: Hyper-parameters of TAG.

strained setting (bottom part of Table 5.1) on the TextVQA dataset [3].⁴ The number of our augmented training QA examples for Text-VQA is 69.2K compared with 34.6K for the original one. In the constrained setting (top), our TAG improves the corresponding M4C and TAP baselines by 1.18% and 2.63% on the validation set, respectively. We note that although LOGOS [135] in Table 5.1 uses an extra grounding dataset with 1.1 million images for pre-training and yet our method performs better. In the unconstrained setting (bottom), TAG further boosts M4C and TAP baselines by 1.11% and 3.06% on the validation set, respectively. On the TextVQA test set, TAG also obtains significant performance gains over existing methods. This validates the effectiveness of TAG.

We also visualize the generated QA pairs of our TAG in Figure 5.4. It shows that our TAG generates meaningful QA pairs that are novel compared to the originally annotated ones.

ST-VQA dataset. We also compare our approach with the state-of-the-art (SOTA)

⁴The constrained setting means training without extra data and the unconstrained one indicates otherwise.

methods under both the constrained setting and the unconstrained setting on the ST-VQA dataset [138]. We compute the accuracy and ANLS score as the evaluation metrics. The number of the newly built training QA examples for the ST-VQA task after augmentation is 46.8K compared with 23.4K for the original one. Table 5.2 suggests that TAG achieves SOTA performance and significantly outperforms the baselines. In particular, TAP [1] achieves 50.83%, and 0.598 in terms of the accuracy and ANLS score on the validation set with additional TextVQA and 1.4 million large-scale pre-training data, while TAG improves these results by a significant 2.70% and 0.022 with only additional TextVQA data. In addition, we submit the prediction results of test set on the ST-VQA test server. The results show that TAG with TAP achieves the SOTA performance with ANLS score of 0.602 on the test set. Without bells and whistles, our approach greatly outperforms the baselines, M4C [53] and TAP [1].

5.4.4 Ablation Studies

We conduct extensive ablation studies to demonstrate the effectiveness of TAG using TAP [1] under the constrained setting on the TextVQA validation set.

Ans.	Obj.	OCR.	Val Acc.
✓			48.76
✓		✓	48.95
✓	✓		49.13
✓	✓	✓	52.54

Table 5.4: Ablation study of TAG with TAP [1] under constrained setting on TextVQA validation set. "Ans.", "Obj." and "OCR." refer to embedding of answer words, detected objects and OCR tokens, respectively.

Answer Selection	Val Acc.
<i>random</i>	49.26
<i>largest</i>	52.54
<i>top three</i>	52.73
<i>top five</i>	52.19

Table 5.5: Ablation study of TAG with TAP [1] under constrained setting on TextVQA validation set. *Random* means a random OCR token is selected as the answer input to TAG, while *top three* means the top three largest OCR tokens are selected.

Contribution of each modality in TAG. To understand the contribution of different input modalities to the success of TAG, Table. 5.4 summarizes the performance of our framework when a certain modality is removed. It suggests that when both the visual objects and OCR tokens modalities are removed, the performance of our TAG decreases by 3.78%. On the other side, when removing the visual objects modality and OCR tokens modality separately, the performance drops by 3.59% and 3.41%, respectively.

Impact of the answer selection strategy. To better explore the performance of our TAG, and understand how different answer selection strategies would affect the model performance, we design several experiments over the choice of input answer selection strategy. Our method adopts the *largest* OCR word as the answer candidate for TAG. We compare this strategy with other possibilities in Table. 5.5. The table shows that, if we use a *random* OCR token as the input answer, the performance drops by 3.28%. On the other hand, if we increase the number of answer candidates by including the top-3 largest OCR tokens to augment the labeled data by 3 $\times$, the performance boosts additional 0.19% as compared to the *largest* strategy while it introduces 3 $\times$ training time. To achieve a better balance between training

efficiency and accuracy, we consider the OCR token with the *largest* bounding box as our final setting for the input answer to TAG. As we have mentioned previously, more high-quality QA pairs could be continuously augmented with a more sophisticated answer-candidate selection strategy. We leave this direction for future work.

5.5 Conclusion

We propose a novel architecture TAG, a text-aware visual question-answer (QA) generation method to deal with the sparse annotation of existing Text-VQA datasets. Our approach leverages the rich yet underexplored visual and scene text information and directly enlarges the existing training set by generating high-quality and rich QA pairs without extra labeling cost. Without bells and whistles, experimental results show that our generated QA pairs boost the performance of recent Text-VQA models by a large margin on both TextVQA and ST-VQA datasets.

Chapter 6: Conclusions

In this dissertation, we mainly focus on learning how to interact with the environment for the 3D perception system, including accurately perceiving objects, capturing the motion of dynamic objects and understanding the scenes. It is critical for the robotic agents to perceive the real world. We investigate and tackle the problem from different computer vision tasks through a variety of forms of input data.

In Chapter 2 and Chapter 3, we proposed different frameworks for effective and efficient 3D object detection from a single frame of point clouds. Specifically, we introduced InfoFocus, an end-to-end framework which dynamically models point cloud distribution to capture varying amount of information at different locations. InfoFocus improves the accuracy of 3D object detection with little overhead by forcing the network to attend to the most informative part of the point cloud. On the other hand, we developed M3DETR, a transformer-based architecture that accounts for multi-representation, multi-scale, mutual-relation models of point clouds in an end-to-end manner.

In Chapter 4, we study on the scene understanding by leveraging a sequences of 3D raw point clouds, especially on the motion of dynamic objects. It is crucial

for the robotic agents to understand dynamic 3D environment. We propose a novel technique for representation learning of 3D point cloud sequences. We also introduce PointMotionNet, a pyramid architecture building on the proposed point-based spatiotemporal convolution, which finds the point correspondences across time by leveraging a time-invariant spatial neighboring space and extracts spatiotemporal features.

In Chapter 5, we investigate an additional modality information for the scene understanding. Besides accurately classifying, locating objects, and predicting their behaviors, the scene is always text-rich scenarios, which provides useful contextual information and can further help the perception. For example, to safely navigate through complex traffic scenarios, an autonomous system needs to understand traffic rules of the road, such as spotting traffic signals or temporary road signs. We introduce TAG, which exploits underexplored scene text information and enhances scene understanding of Text-VQA models by producing meaningful, and accurate question-answer (QA) samples using a multimodal transformer. TAG has the potential to be applied to identify challenging traffic situations that the autonomous vehicles will encounter on roads.

Bibliography

- [1] Zhengyuan Yang, Yijuan Lu, Jianfeng Wang, Xi Yin, Dinei Florencio, Lijuan Wang, Cha Zhang, Lei Zhang, and Jiebo Luo. Tap: Text-aware pre-training for text-vqa and text-caption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8751–8761, 2021.
- [2] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension. In *European conference on computer vision*, pages 742–758. Springer, 2020.
- [3] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019.
- [4] Jun Wang, Shiyi Lan, Mingfei Gao, and Larry S Davis. Infofocus: 3d object detection for autonomous driving with dynamic information modeling. In *European Conference on Computer Vision*, pages 405–420. Springer, 2020.
- [5] Tianrui Guan, Jun Wang, Shiyi Lan, Rohan Chandra, Zuxuan Wu, Larry Davis, and Dinesh Manocha. M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 772–782, January 2022.
- [6] Jun Wang, Xiaolong Li, Alan Sullivan, Lynn Abbott, and Siheng Chen. Pointmotionnet: Point-wise motion learning for large-scale lidar point clouds sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4419–4428, June 2022.
- [7] Jun Wang, Mingfei Gao, Yuqian Hu, Ramprasaath R Selvaraju, Chetan Ramaiah, Ran Xu, Joseph F JaJa, and Larry S Davis. Tag: Boosting text-vqa via text-aware visual question-answer generation. *arXiv preprint arXiv:2208.01813*, 2022.

- [8] 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.
- [9] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *arXiv preprint arXiv:1903.01864*, 2019.
- [10] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S. Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [11] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.
- [12] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1951–1960, 2019.
- [13] 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 Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.
- [14] 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.
- [15] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.
- [16] Yangyang Ye, Houjin Chen, Chi Zhang, Xiaoli Hao, and Zhaoxiang Zhang. Sarpnet: Shape attention regional proposal network for lidar-based 3d object detection. *Neurocomputing*, 379:53–63, 2020.
- [17] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. *arXiv preprint arXiv:1912.04986*, 2019.
- [18] 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. *arXiv preprint arXiv:1903.11027*, 2019.

- [19] 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.
- [20] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.
- [21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [22] Ross Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [24] 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.
- [25] 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.
- [26] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.
- [27] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.
- [28] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Point-painting: Sequential fusion for 3d object detection. *arXiv preprint arXiv:1911.10150*, 2019.
- [29] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9775–9784, 2019.

- [30] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, pages 963–973, 2019.
- [31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.
- [32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [33] 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.
- [34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [35] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.
- [36] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kotschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1991–1999, 2019.
- [37] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.
- [38] Jing Huang, Viswanath Sivakumar, Mher Mnatsakanyan, and Guan Pang. Improving rotated text detection with rotation region proposal networks. *arXiv preprint arXiv:1811.07031*, 2018.
- [39] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.
- [40] 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.

- [41] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 117–120. IEEE, 2008.
- [42] Yong-Joo Oh and Yoshio Watanabe. Development of small robot for home floor cleaning. In *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, volume 5, pages 3222–3223. IEEE, 2002.
- [43] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1631–1640, 2020.
- [44] 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.
- [45] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.
- [46] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020.
- [47] 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.
- [48] Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3d object detection from point clouds. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4144–4152, 2020.
- [49] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.
- [50] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 998–1008, 2019.
- [51] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

- [52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [53] Ronghang Hu, Amanpreet Singh, Trevor Darrell, and Marcus Rohrbach. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9992–10002, 2020.
- [54] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.
- [55] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vi-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.
- [56] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019.
- [57] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.
- [58] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.
- [59] 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.
- [60] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, pages 685–702. Springer, 2020.
- [61] Yuanqiang Cai, Dawei Du, Libo Zhang, Longyin Wen, Weiqiang Wang, Yanjun Wu, and Siwei Lyu. Guided attention network for object detection and counting on drones. *arXiv preprint arXiv:1909.11307*, 2019.
- [62] Xinhai Liu, Zhizhong Han, Xin Wen, Yu-Shen Liu, and Matthias Zwicker. L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 989–997, 2019.

- [63] Miao Zhang, Yu Zhang, Yongri Piao, Beiqi Hu, and Huchuan Lu. Feature reintegration over differential treatment: A top-down and adaptive fusion network for rgb-d salient object detection. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4107–4115, 2020.
- [64] Guibiao Liao, Wei Gao, Qiuping Jiang, Ronggang Wang, and Ge Li. Mmnet: Multi-stage and multi-scale fusion network for rgb-d salient object detection. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2436–2444, 2020.
- [65] Lei Zhu, Zijun Deng, Xiaowei Hu, Chi-Wing Fu, Xuemiao Xu, Jing Qin, and Pheng-Ann Heng. Bidirectional feature pyramid network with recurrent attention residual modules for shadow detection. In *ECCV*, 2018.
- [66] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11001–11009, 2020.
- [67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [68] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020.
- [69] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *arXiv preprint arXiv:2011.00931*, 2020.
- [70] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020.
- [71] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. *arXiv preprint arXiv:2012.11409*, 2020.
- [72] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [73] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.
- [74] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

- [75] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020.
- [76] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *European Conference on Computer Vision*, pages 528–543. Springer, 2020.
- [77] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Point-painting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4604–4612, 2020.
- [78] Trisha Mittal, Uttaran Bhattacharya, Rohan Chandra, Aniket Bera, and Dinesh Manocha. M3er: Multiplicative multimodal emotion recognition using facial, textual, and speech cues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1359–1367, 2020.
- [79] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. *arXiv preprint arXiv:2005.09927*, 2020.
- [80] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, et al. Starnet: Targeted computation for object detection in point clouds. *arXiv preprint arXiv:1908.11069*, 2019.
- [81] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.
- [82] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. *arXiv preprint arXiv:2103.10039*, 2021.
- [83] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [84] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12677–12686, 2019.
- [85] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In

- Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [86] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *European Conference on Computer Vision*, pages 68–84. Springer, 2020.
 - [87] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981.
 - [88] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019.
 - [89] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3254–3263. Computer Vision Foundation / IEEE, 2019.
 - [90] Wenxuan Wu, Zhiyuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. *arXiv preprint arXiv:1911.12408*, 2019.
 - [91] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird’s eye view maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11385–11395, 2020.
 - [92] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9246–9255, 2019.
 - [93] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5379–5389, 2019.
 - [94] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J Guibas. Caspr: Learning canonical spatiotemporal point cloud representations. *Advances in Neural Information Processing Systems*, 33, 2020.
 - [95] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

- [96] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.
- [97] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017.
- [98] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.
- [99] Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. Latticenet: fast spatio-temporal point cloud segmentation using permutohedral lattices. *Autonomous Robots*, pages 1–16, 2021.
- [100] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.
- [101] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382. IEEE, 2019.
- [102] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019.
- [103] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.
- [104] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018.
- [105] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

- [106] Yangheng Zhao, Jun Wang, Xiaolong Li, Yue Hu, Ce Zhang, Yanfeng Wang, and Siheng Chen. Number-adaptive prototype learning for 3d point cloud semantic segmentation. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 695–703. Springer, 2023.
- [107] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4574–4583, 2020.
- [108] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020.
- [109] Tianrui Guan, Jun Wang, Shiyi Lan, Rohan Chandra, Zuxuan Wu, Larry Davis, and Dinesh Manocha. M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 772–782, 2022.
- [110] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Wei Li, Yuexin Ma, Hongsheng Li, Ruigang Yang, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar-based perception. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [111] 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 (CVPR)*, July 2017.
- [112] 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 (CVPR)*, June 2018.
- [113] 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 (CVPR)*, June 2019.
- [114] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

- [115] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *arXiv preprint arXiv:1911.11236*, 2019.
- [116] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. In *International conference on learning representations*, 2020.
- [117] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14204–14213, 2021.
- [118] Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixé. 4d panoptic lidar segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5527–5537, 2021.
- [119] Guangming Wang, Hanwen Liu, Muyao Chen, Yehui Yang, Zhe Liu, and Hesheng Wang. Anchor-based spatio-temporal attention 3-d convolutional networks for dynamic 3-d point cloud sequences. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2021.
- [120] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3254–3263, 2019.
- [121] Pan He, Patrick Emami, Sanjay Ranka, and Anand Rangarajan. Learning scene dynamics from point cloud sequences. *International Journal of Computer Vision*, pages 1–27, 2022.
- [122] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.
- [123] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018.
- [124] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving. *arXiv preprint arXiv:2003.00601*, 2020.
- [125] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

- [126] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.
- [127] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.
- [128] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *ICDAR*, 2019.
- [129] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samuel White, et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 333–342, 2010.
- [130] Yash Kant, Dhruv Batra, Peter Anderson, Alexander Schwing, Devi Parikh, Jiasen Lu, and Harsh Agrawal. Spatially aware multimodal transformers for textvqa. In *European Conference on Computer Vision*, pages 715–732. Springer, 2020.
- [131] Chenyu Gao, Qi Zhu, Peng Wang, Hui Li, Yuliang Liu, Anton Van den Hengel, and Qi Wu. Structured multimodal attentions for textvqa. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [132] Fen Liu, Guanghui Xu, Qi Wu, Qing Du, Wei Jia, and Minghui Tan. Cascade reasoning network for text-based visual question answering. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4060–4069, 2020.
- [133] Wei Han, Hantao Huang, and Tao Han. Finding the evidence: Localization-aware answer prediction for text visual question answering. *arXiv preprint arXiv:2010.02582*, 2020.
- [134] Qi Zhu, Chenyu Gao, Peng Wang, and Qi Wu. Simple is not easy: A simple strong baseline for textvqa and textcaps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3608–3615, 2021.
- [135] Xiaopeng Lu, Zhen Fan, Yansen Wang, Jean Oh, and Carolyn P Rosé. Localize, group, and select: Boosting text-vqa by scene text modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2631–2639, 2021.

- [136] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3617, 2018.
- [137] Qingqing Wang, Liqiang Xiao, Yue Lu, Yaohui Jin, and Hao He. Towards reasoning ability in scene text visual question answering. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2281–2289, 2021.
- [138] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluís Gomez, Marçal Rusinol, Ernest Valveny, CV Jawahar, and Dimosthenis Karatzas. Scene text visual question answering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4291–4301, 2019.
- [139] Yu Jiang, Vivek Natarajan, Xinlei Chen, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia v0. 1: the winning entry to the vqa challenge 2018. *arXiv preprint arXiv:1807.09956*, 2018.
- [140] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.
- [141] Difei Gao, Ke Li, Ruiping Wang, Shiguang Shan, and Xilin Chen. Multi-modal graph neural network for joint reasoning on vision and scene text. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12746–12756, 2020.
- [142] Xuanyu Zhang and Qing Yang. Position-augmented transformers with entity-aligned mesh for textvqa. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2519–2528, 2021.
- [143] Gangyan Zeng, Yuan Zhang, Yu Zhou, and Xiaomeng Yang. Beyond ocr+ vqa: involving ocr into the flow for robust and accurate textvqa. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 376–385, 2021.
- [144] Ali Furkan Biten, Ron Litman, Yusheng Xie, Srikar Appalaraju, and R. Manmatha. Latr: Layout-aware transformer for scene-text vqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16548–16558, June 2022.
- [145] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [146] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation

- learning. In *European conference on computer vision*, pages 104–120. Springer, 2020.
- [147] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.
 - [148] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.
 - [149] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13041–13049, 2020.
 - [150] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.
 - [151] Jun Wang. ESSumm: Extractive Speech Summarization from Untranscribed Meeting. In *Proc. Interspeech 2022*, pages 3243–3247, 2022.
 - [152] Junke Wang, Dongdong Chen, Zuxuan Wu, Chong Luo, Luowei Zhou, Yucheng Zhao, Yujia Xie, Ce Liu, Yu-Gang Jiang, and Lu Yuan. Omnivl: One foundation model for image-language and video-language tasks. *arXiv preprint arXiv:2209.07526*, 2022.
 - [153] Bo He, Jun Wang, Jielin Qiu, Trung Bui, Abhinav Shrivastava, and Zhaowen Wang. Align and attend: Multimodal summarization with dual contrastive losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
 - [154] Kushal Kafle, Mohammed A Yousefhussien, and Christopher Kanan. Data augmentation for visual question answering. In *INLG*, pages 198–202, 2017.
 - [155] Meet Shah, Xinlei Chen, Marcus Rohrbach, and Devi Parikh. Cycle-consistency for robust visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6649–6658, 2019.
 - [156] Arijit Ray, Karan Sikka, Ajay Divakaran, Stefan Lee, and Giedrius Burachas. Sunny and dark outside?! improving answer consistency in vqa through entailed question generation. *arXiv preprint arXiv:1909.04696*, 2019.

- [157] Vedika Agarwal, Rakshith Shetty, and Mario Fritz. Towards causal vqa: Revealing and reducing spurious correlations by invariant and covariant semantic editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9690–9698, 2020.
- [158] Ruixue Tang, Chao Ma, Wei Emma Zhang, Qi Wu, and Xiaokang Yang. Semantic equivalent adversarial data augmentation for visual question answering. In *European Conference on Computer Vision*, pages 437–453. Springer, 2020.
- [159] Zixu Wang, Yishu Miao, and Lucia Specia. Cross-modal generative augmentation for visual question answering. *arXiv preprint arXiv:2105.04780*, 2021.
- [160] Yash Kant, Abhinav Moudgil, Dhruv Batra, Devi Parikh, and Harsh Agrawal. Contrast and classify: Training robust vqa models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1604–1613, 2021.
- [161] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.
- [162] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2552–2566, 2014.
- [163] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.
- [164] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.
- [165] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.
- [166] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluís Gómez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernández Mota, Jon Almazan Almazan, and Lluís Pere De Las Heras. Icdar 2013 robust reading competition. In *2013 12th international conference on document analysis and recognition*, pages 1484–1493. IEEE, 2013.
- [167] Dimosthenis Karatzas, Lluís Gómez-Bigorda, Angelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition

- on robust reading. In *2015 13th international conference on document analysis and recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.
- [168] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
 - [169] Anand Mishra, Karteek Alahari, and CV Jawahar. Image retrieval using textual cues. In *Proceedings of the IEEE international conference on computer vision*, pages 3040–3047, 2013.
 - [170] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.
 - [171] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.