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

Tianrui Guan[1*] Jun Wang[1*] Shiyi Lan[1†] Rohan Chandra[1] Zuxuan Wu [2]
Larry Davis [1] Dinesh Manocha[1]
[1]University of Maryland, College Park [2]Fudan University
{rayguan, sylan, rohan}@cs.umd.edu, {junwang,lsd}@umiacs.umd.edu,
zxwu@fudan.edu.cn, dmanocha@umd.edu

## Abstract

*We present 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. We perform extensive ablation experiments that highlight the benefits of fusing representation and scale, and modeling the relationships. Our method achieves state-of-the-art performance on the KITTI 3D object detection dataset and Waymo Open Dataset. Results show that* M3DETR *improves the baseline significantly by 1.48% mAP for all classes on Waymo Open Dataset. In particular, our approach ranks* $1^{st}$ *on the well-known KITTI 3D Detection Benchmark for both car and cyclist classes, and ranks* $1^{st}$ *on Waymo Open Dataset with single frame point cloud input. Our code is available at: https://github.com/rayguan97/M3DETR.*

## 1. Introduction

3D object detection is a fundamental problem in computer vision and many applications, including autonomous driving [11, 44], augmented reality [33] and robotics [31]. Moreover, different methods have been proposed for various sensors, including monocular cameras, depth cameras, LiDAR, and radars [35, 36, 34, 56, 57, 55, 39, 58]. 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 [2]. Consequently, point clouds are being widely used for scene understanding in autonomous driving and AR.

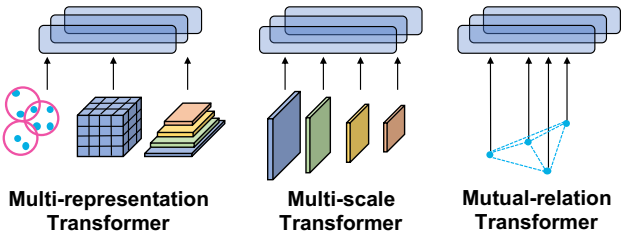


Figure 1. M3 Transformers used in our approach: Left: Multi-representation transformers; Middle: Multi-scale transformers; Right: Mutual-relation transformers. We use these characteristics to present a novel 3D object detection architecture.

The previous state-of-the-art methods for 3D object detection base on different networks [58, 39]. However, there are two key limitations:

**Ineffective point cloud representations:** The three major techniques used to process point clouds are based on voxels [62, 52], raw point clouds [35, 36, 34, 40, 56, 26, 37], and bird's-eye-view [6, 17, 53]. Each representation has a unique advantage and it has been shown that combining these representations can result in terms of detection accuracy [34, 39, 7]. 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 [7, 39] 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

*Equal contribution.
†Corresponding author: <sylan@cs.umd.edu>.

for improving the detection performance of 2D object detection. In terms of 3D object detection, current approaches tend to use multi-scale feature pyramids [35, 36, 55, 18, 19, 57]. However, fusing these multiple feature pyramids is non-trivial because the higher resolution and the larger receptive fields are conflicting [23, 36]. Existing methods [36, 57, 39] fuse the multi-scale features using bi-linear down-sampling/up-sampling and concatenation. Although these approaches [36, 57, 39] 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 [48], a form of neural network based on attention that has been used in natural language processing [16, 45, 43, 27]. Specifically, transformers use multi-head attention to narrow the 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 [35, 36, 18, 47]. 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 [35, 36, 34, 39], farthest point sampling layers [36, 40, 39], max pooling layers [18, 36, 40, 39], and graph convolutional networks [51]. 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 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 [11] and Waymo Open Dataset [44]. 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.

## 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 [6, 17, 21, 20] 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 [62, 52, 19] use equally spaced 3D voxels to encode the point clouds such that the volumetric representation can be consumed by the region proposal network (RPN) [38]. Inspired by the PointNet/PointNet++ approach [35, 36], which is invariant under transformation, [34, 40] 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 [56, 26, 46, 39, 58] 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 [23, 3, 25, 60, 22, 64, 36, 15, 50] 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++ [36] 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, [15, 50, 57] adopt different detection heads with multi-scale feature maps to handle both large and small object classes.

**Mutual-relation modeling.** 2D Convolutional Neural Networks [14] 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 [36] 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 [47, 18, 51]

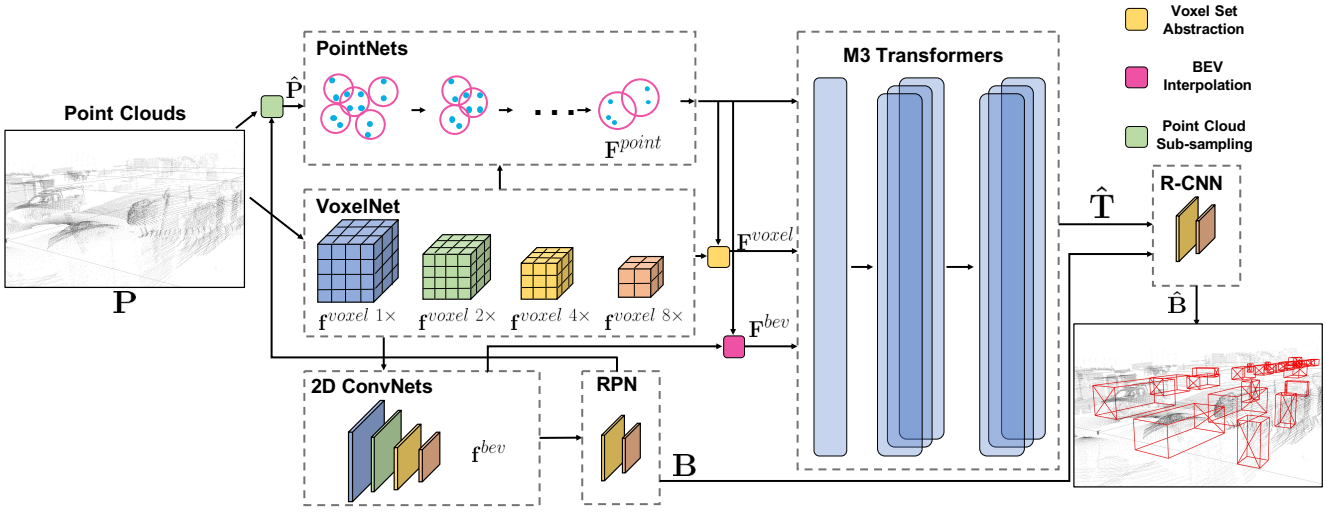


Figure 2. **An overview of our M3DETR architecture: M3DETR is a transformer based framework for object detection in a coarse-to-fine manner.** It consists of three parts. PointNets, VoxelNet, and 2D ConvNets modules enable individual multi-representation feature learning. M3 Transformers enable inter-intra multi-representation, multi-scale, 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.

on point clouds have been proposed to model the local context and the mutual relation between points. Recently, transformers [61, 9, 13, 32] 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 [48], 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. [8] leverages the direct application of transformers on the image recognition task without using convolution. [4, 65] apply transformers to eliminate the need for many handcrafted components in conventional object detection and achieve impressive detection results. [54, 59] explore transformers on image and video synthesis tasks. [61] investigates the self-attention networks on 3D point cloud segmentation task.

Recently, there have been several joint representation fusion research attempts. PointPainting [49] 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, [16] jointly fuses and reasons over three different modality representations. [29] 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. 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 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.

### 3.1. Multi-Representation Feature Embeddings

Our network processes the raw input point clouds $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_n\}$ and encodes them into three different em-

bedding 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 [62], takes the raw input point clouds $\boldsymbol{P}$ and converts them into equally spaced 3D voxels $\boldsymbol{v} \in \mathbb{R}^{L \times W \times H}$; (2) voxel-wise features $\boldsymbol{f}^{voxel}$ at different scales are extracted with 3D sparse convolutions, which is visualized in Figure 2. Unlike [62, 52], all four different scales of obtained 3D voxel CNN features, $\boldsymbol{f}^{voxel} = \{\boldsymbol{f}^{voxel\ 1\times}, \boldsymbol{f}^{voxel\ 2\times}, \boldsymbol{f}^{voxel\ 4\times}, \boldsymbol{f}^{voxel\ 8\times}\}$ are passed to the transformer stage as shown in Figure 2. More details will be introduced in the supplementary materials.

**Bird's-eye-view:** 2D ConvNets takes the $8\times$ downsampled voxel-based feature map $\boldsymbol{f}^{voxel\ 8\times}$ 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 deconvolutional 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 $3 \times 3$ followed by a BatchNorm layer and a ReLU layer. The output BEV-based feature from 2D ConvNets, denoted as $\boldsymbol{f}^{bev} \in \mathbb{R}^{L_{bev} \times W_{bev} \times C_{bev}}$, was further converted into keypoint feature $\boldsymbol{F}^{bev} \in \mathbb{R}^{n \times c_{bev}}$ 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 $\hat{\boldsymbol{P}} \subset \boldsymbol{P}$. Adopted from PointNet++ [36] and PV-RCNN [39], Set Abstraction and Voxel Set Abstraction (VSA) module take raw point coordinates $\boldsymbol{P}$ and the 3D voxel-based features $\boldsymbol{f}^{voxel}$, respectively, to generate keypoint features for $\hat{\boldsymbol{P}}$. In particular, the corresponding keypoint features from raw points $\boldsymbol{P}$ is $\boldsymbol{F}^{point} \in \mathbb{R}^{n \times c_{point}}$, and the corresponding keypoint features from voxels $\boldsymbol{f}^{voxel}$ are $\{\boldsymbol{F}^{voxel\ 1\times}, \boldsymbol{F}^{voxel\ 2\times}, \boldsymbol{F}^{voxel4\times}, \boldsymbol{F}^{voxel\ 8\times}\}$, where $\boldsymbol{F}^{i} \in \mathbb{R}^{n \times c_i}$.

### 3.2. Multi-representation, Multi-scale, and Mutual-relation Transformer

Once the three feature embedding sequences, $\boldsymbol{F}^{voxel}$, $\boldsymbol{F}^{point}$ and $\boldsymbol{F}^{bev}$ 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 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 [48]. Consider two input matrices $\boldsymbol{X}_l \in \mathbb{R}^{l \times d_{in}}$ and $\boldsymbol{X}_s \in \mathbb{R}^{s \times d_{in}}$, where $l$ and $s$ are the lengths of the input sequence of dimension $d_{in}$. The attention layer output is defined as:

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

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

The matrices $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ represent query, key and value respectively, obtained through projection matrices $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{d_{in} \times d_h}$ and $\boldsymbol{W}_v \in \mathbb{R}^{d_{in} \times d_{out}}$, where $d_h$ is the hidden dimension, and $d_{out}$ is the output dimension. When two input matrices $\boldsymbol{X}_l$ and $\boldsymbol{X}_s$ 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. More details will be introduced in the supplementary materials.

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: (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 $\boldsymbol{F} = [\boldsymbol{F}^{voxel\ 1\times}, \boldsymbol{F}^{voxel\ 2\times}, \boldsymbol{F}^{voxel\ 4\times}, \boldsymbol{F}^{voxel\ 8\times}, \boldsymbol{F}^{point}, \boldsymbol{F}^{bev}]$, where each $\boldsymbol{F}^{i} \in \mathbb{R}^{n \times c_i}$. 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 $\boldsymbol{F}$.

The multi-representation and multi-scale transformer layer takes 6 different inputs: $\boldsymbol{F}^{point}$, $\boldsymbol{F}^{voxel1\times}$, $\boldsymbol{F}^{voxel2\times}$, $\boldsymbol{F}^{voxel4\times}$, $\boldsymbol{F}^{voxel8\times}$, $\boldsymbol{F}^{bev}$. 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 $\hat{\boldsymbol{F}} = [\hat{\boldsymbol{F}}^{voxel\ 1\times}, \hat{\boldsymbol{F}}^{voxel\ 2\times}, \hat{\boldsymbol{F}}^{voxel\ 4\times}, \hat{\boldsymbol{F}}^{voxel\ 8\times}, \hat{\boldsymbol{F}}^{point}, \hat{\boldsymbol{F}}^{bev}]$, 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 $\hat{\boldsymbol{F}}$ as inputs and generates self-attention features $\boldsymbol{T}^{voxel\ 1\times}$,

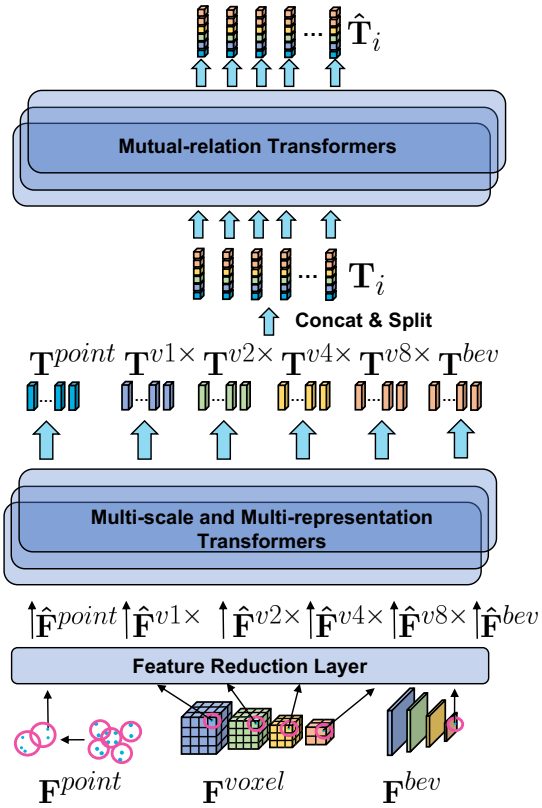


Figure 3. **M3 Transformers** consist of two parts: a multi-representation and multi-scale transformer, and a mutual-relation transformer. Multi-representation and multi-scale transformer takes the different feature embedding and generates enriched cross-representations and cross-scales embedding. On top of it, the mutual-relation transformer further models point-wise feature relationship to extract the refined features.

$\boldsymbol{T}^{voxel\ 2\times}$,$\boldsymbol{T}^{voxel\ 4\times}$,$\boldsymbol{T}^{voxel\ 8\times}$, $\boldsymbol{T}^{point}$, $\boldsymbol{T}^{bev}$, which corresponds to $\hat{\boldsymbol{F}}^{voxel\ 1\times}$, $\hat{\boldsymbol{F}}^{voxel\ 2\times}$,$\hat{\boldsymbol{F}}^{voxel\ 4\times}$,$\hat{\boldsymbol{F}}^{voxel\ 8\times}$, $\hat{\boldsymbol{F}}^{point}$, $\hat{\boldsymbol{F}}^{bev}$. We visualize the multi-representation and multi-scale transformer in Figure 3.

**Mutual-relation transformer layer.** Inspired by [35, 36, 32], 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 $\boldsymbol{T}$ 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 $\boldsymbol{T} = \text{concat}\ \{\boldsymbol{T}^{voxel\ 1\times}, \boldsymbol{T}^{voxel\ 2\times}, \boldsymbol{F}^{voxel\ 4\times}, \boldsymbol{T}^{voxel\ 8\times}, \boldsymbol{T}^{point}, \boldsymbol{T}^{bev}\} \in \mathbb{R}^{n\times c_T}$, where $c_T$ is the sum of all channel size in $\boldsymbol{T}$. Note that, the first dimension of $\boldsymbol{T}$ is the number of keypoints $n$. To model the mutual-relation between keypoints, we need to split $\boldsymbol{T}$ into $n$ point-wise feature $\boldsymbol{T} = \{\boldsymbol{T}_i\}$, where $\boldsymbol{T}_i \in \mathbb{R}^{c_T}$. The concatenation and split module is showed in Figure 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 $\hat{\boldsymbol{T}}$.

**Comparison with previous point-based transformers** Prior work has explored the transformer application on the task of point cloud processing. First, [13, 9, 61] 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 [32] 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. Detection Heads Network

After we obtain the enriched embedding $\hat{\boldsymbol{T}}$ 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 [39] for more details.

**RPN:** Region Proposal Networks take the deep semantic features $\boldsymbol{f}^{bev}$ produced by the 2D ConvNets as inputs and generate high-quality 3D object proposals $\boldsymbol{B}$, as shown in Figure 2. A 3D object box $\boldsymbol{B}_i$ 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 $\theta$ is the orientation in bird's-eye-view. Similar to the conventional RPN in 2D object detection [38] 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 $\boldsymbol{B}$ from RPN as input to conduct further proposal refinement. For each input proposal box, the RoI-grid pooling module [39] is adopted to extract the corresponding proposal-specific grid points' features from the transformer-based embeddings $\hat{\boldsymbol{T}}$. 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 [35] on a small number of surrounding key-

points, $M$, 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.

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, $\hat{\boldsymbol{B}}$. 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, [41, 39] show that it achieves better results compared with the traditional classification confidence based techniques.

### 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: $\triangle x = \frac{x^{gt}-x^a}{d^a}, \triangle y = \frac{y^{gt}-y^a}{d^a}, \triangle z = \frac{z^{gt}-z^a}{h^a}$ $\triangle h = \log(\frac{h^{gt}}{h^a}), \triangle w = \log(\frac{w^{gt}}{w^a}), \triangle \theta = \theta^{gt} - \theta^a$, where $d^a = \sqrt{(w^a)^2 + (l^a)^2}$. Similar to [39, 41], the focal loss is applied [24] for the classification loss, $\mathcal{L}_{cls}$. Smooth L1 loss [12] is adopted for the box localization regression target's losses, $\mathcal{L}_{reg}$ and $\mathcal{L}_{ref}$. In addition, 3D IoU loss [39, 41] is used for $\mathcal{L}_{iou}$.

Similar to PV-RCNN [39], we formally define a multi-task loss for both the RPN and R-CNN stages,

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

where $p^a$ is the model's estimated class probability for an anchor box, and $\beta_{reg}$, $\beta_{iou}$ and $\beta_{ref}$ 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 [24].

## 4. Experiments

In this section, we evaluate M3DETR both qualitatively and quantitatively on the Waymo Open Dataset [44] and the KITTI Dataset [11] 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.

### 4.1. Datasets and Evaluation Metric

**Waymo Open Dataset:** The Waymo Open Dataset [44] 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 [11] is another 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.

### 4.2. Implementation Details

Closely following the codebase[1], we use PyTorch to implement our M3 transformer modules and integrate them into the PV-RCNN network [39]. More details about backbone and detection heads network will be introduced in the supplementary materials.

**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.

**Training Parameters** Models are trained from scratch on $4$ NVIDIA P6000 GPUs. We use the Adam optimizer with a fixed weight decay of $0.01$ and use a one-cycle scheduler proposed in [42]. 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.

### 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 1 compared with PV-RCNN [39]. We evaluate our method at both

[1] https://github.com/open-mmlab/OpenPCDet.

| 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* [39] | 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 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 [1] | 69.59 | 87.2 | 67.8 | 46.1 | - | - | - | - | - | - | - | - | - | - | - | - |
| StarNet [30] | 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 [19] | 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 [63] | 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 [10] | 75.83 | 88.41 | 73.83 | 55.31 | 75.38 | 87.95 | 73.38 | 54.84 | 67.12 | 87.53 | 67.99 | 44.40 | 66.73 | 87.08 | 67.58 | 44.01 |
| PV-RCNN* [39] | 76.89 | 92.27 | 75.51 | 55.35 | 76.30 | 91.82 | 74.79 | 54.27 | 67.99 | 89.18 | 69.39 | 42.80 | 67.46 | 88.75 | 68.70 | 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 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.

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 test set. As we can see, PV-RCNN achieves 69.57% and 63.65% on average in 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 [39]. Furthermore, we compare our framework on the vehicle class for different distances with state-of-the-art methods, including StarNet [30], PointPillars [19], RCD [1], Det3D [63], RangeDet [10] and PV-RCNN [39]. In Table 2, M3DETR outperforms PV-RCNN significantly in both LEVEL_1 and LEVEL_2 difficulty levels across all distances, demonstrating the effectiveness of newly proposed framework. Moreover, we visualize the detection results of M3DETR in Figure 4.

Compared with the PV-RCNN shown in Figure 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 for Vehicle class on the full Waymo Open Dataset validation set as in Table 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 [39, 5]. We compute the mAP on three difficult types of both car and cyclist classes in 3D detection metric. Table 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 [5] achieves $82.59\%$ in the "easy" categories of 3D detection metric, while M3DETR improves these results by significant $1.24\%$.

### 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 5. All experiments are conducted on the validation 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.

### 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 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.

| 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 [28] | 52.11 | 70.90 | 52.90 | 29.60 | 50.05 | 68.70 | 51.40 | 28.60 | - | - | - | - | - | - | - | - |
| PointPillars [19] | 56.62 | 81.00 | 51.80 | 27.90 | - | - | - | - | - | - | - | - | - | - | - | - |
| RCD [1] | 69.59 | 87.20 | 67.80 | 46.10 | 69.16 | 86.80 | 67.40 | 45.50 | - | - | - | - | - | - | - | - |
| RangeDet [10] | 72.85 | 87.96 | 69.03 | 48.88 | - | - | - | - | - | - | - | - | - | - | - | - |
| PV-RCNN [39] | 70.30 | 91.90 | 69.20 | 42.20 | 69.69 | 91.34 | 68.53 | 41.31 | 65.36 | 91.58 | 65.13 | 36.46 | 64.79 | 91.00 | 64.49 | 35.70 |
| 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. 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.

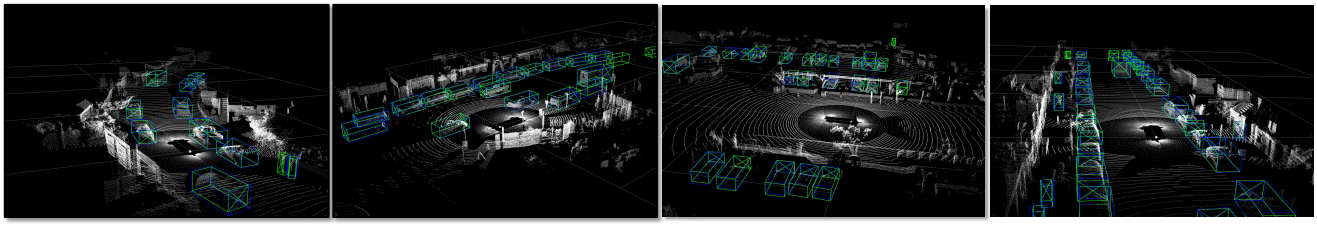

Figure 4. We highlight the 3D detection results of **M3DETR** on the Waymo Open Dataset. The 3D ground truth bounding boxes are in green, while the detection bounding box are shown in blue.

| Method | Car | | | Cyclist | | |
|---|---|---|---|---|---|---|
| | Easy | Mod | Hard | Easy | Mod | Hard |
| F-PointNet [34] | 72.27 | 56.12 | 49.01 | 82.19 | 69.79 | 60.59 |
| VoxelNet [62] | 77.47 | 65.11 | 57.73 | 61.22 | 48.36 | 44.37 |
| SECOND [52] | 83.34 | 72.55 | 65.82 | 75.83 | 60.82 | 53.67 |
| PointPillars [19] | 82.58 | 74.31 | 68.99 | 77.10 | 58.65 | 51.92 |
| PointRCNN [40] | 86.96 | 75.64 | 70.70 | 74.96 | 58.82 | 52.53 |
| STD [56] | 87.95 | 79.71 | 75.09 | 78.69 | 61.59 | 55.30 |
| HotSpotNet [5] | 87.60 | 78.31 | 73.34 | 82.59 | 65.95 | 59.00 |
| PVRCNN [39] | 90.25 | 81.43 | 76.82 | 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 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.

| 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 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.

## 5. Conclusions

In this paper, 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

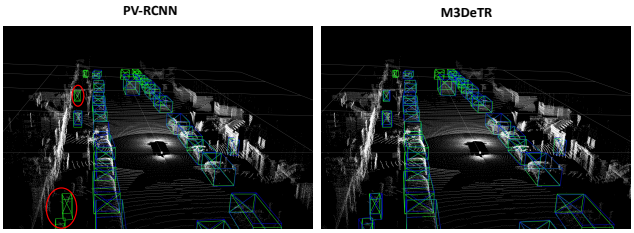


Figure 5. We visualize the 3D detection results for the same input point cloud between PV-RCNN (left) and **M3DETR** (right) on Waymo Open Dataset. We highlight the false negative boxes from PV-RCNN in red.

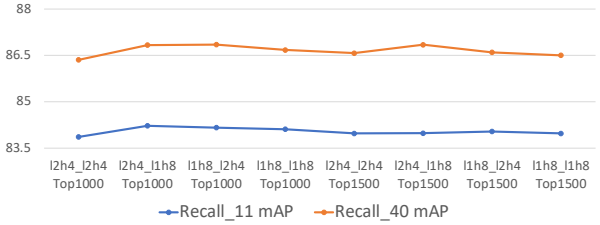


Figure 6. Performance comparison of different **M3 Transformers** variants in Car class on KITTI validation set. We show the mAP results with IoU threshold of 0.7 for both 11 and 40 recall positions. Note that "l" and "h" represent layer number and head dimension of M3 Transformers, respectively. "Top" denotes the number of proposals used for keypoint sampling from RPN stage.

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.

**Acknowledgement.** This work was supported in part by ARO Grants W911NF1910069, W911NF2110026 and U.S. Army Grant No. W911NF2120076.

## References


[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9775–9784, 2019.

[8] 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.

[9] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *arXiv preprint arXiv:2011.00931*, 2020.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[20] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *arXiv preprint arXiv:1907.03739*, 2019.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. *arXiv preprint arXiv:2012.11409*, 2020.

[33] 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.

[34] 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.

[35] 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.

[36] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[37] 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.

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[39] 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.

[40] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[41] 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.

[42] 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.

[43] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[44] 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.

[45] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[46] 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.

[47] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[49] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4604–4612, 2020.

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

[55] 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.

[56] 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*, pages 1951–1960, 2019.

[57] 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.

[58] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020.

[59] 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.

[60] 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.

[61] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

# A. More Our Approach Details

We further discuss our approach in the following.

## A.1. Voxel Representation in Multi-Representation Feature Embeddings

For the voxel-wise feature extraction from raw point clouds input, there are two steps, voxelization using voxelization layer and feature extraction using 3D sparse convolutions. 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 $3\times3\times3$ 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.

## A.2. Multi-head Self-attention Basics

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

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

where $\text{head}_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^q, \boldsymbol{K}\boldsymbol{W}_i^k, \boldsymbol{V}\boldsymbol{W}_i^v)$ and $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are linear projections of X. The projection matrices $\boldsymbol{W}_i^q \in \mathbb{R}^{d_h \times d_q}$, $\boldsymbol{W}_i^k \in \mathbb{R}^{d_h \times d_k}$, $\boldsymbol{W}_i^v \in \mathbb{R}^{d_h \times d_v}$, $\boldsymbol{W}^O \in \mathbb{R}^{d_v \times d_{out}}$ 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 $\boldsymbol{K}$, $\boldsymbol{V}$, and $\boldsymbol{Q}$, respectively.

# B. More Implementation Details

As mentioned in the Section 4.2, we give more details on our implementation for reproduction of our result. Our code will also be released later, including trained models that can match the performance that was included in the paper.

## B.1. 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 [36] through set abstraction and voxel set abstraction [39]. 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.

**Waymo:** 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 $188 \times 188 \times 512$.

**KITTI:** 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 $200 \times 176 \times 512$.

## B.2. 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. The RoI-grid pooling module samples $6 \times 6 \times 6$ grid points within each initial 3D proposal to form a refined features. 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.