An End-to-End Transformer Model for 3D Object Detection

Ishan Misra Rohit Girdhar Armand Joulin
Facebook AI Research
https://facebookresearch.github.io/3detr

Abstract

We propose 3DETR, an end-to-end Transformer based object detection model for 3D point clouds. Compared to existing detection methods that employ a number of 3D-specific inductive biases, 3DETR requires minimal modifications to the vanilla Transformer block. Specifically, we find that a standard Transformer with non-parametric queries and Fourier positional embeddings is competitive with specialized architectures that employ libraries of 3D-specific operators with hand-tuned hyperparameters. Nevertheless, 3DETR is conceptually simple and easy to implement, enabling further improvements by incorporating 3D domain knowledge. Through extensive experiments, we show 3DETR outperforms the well-established and highly optimized VoteNet baselines on the challenging ScanNetV2 dataset by 9.5%. Furthermore, we show 3DETR is applicable to 3D tasks beyond detection, and can serve as a building block for future research.

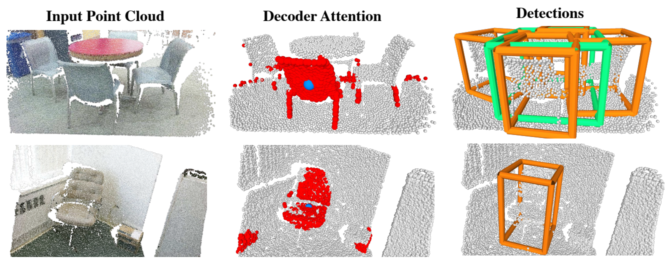

Figure 1: 3DETR. We train an end-to-end Transformer model for 3D object detection on point clouds. Our model has a Transformer encoder for feature encoding and a Transformer decoder for predicting boxes. For an unseen input, we compute the self-attention from the reference point (blue dot) to all points in the scene and display the points with the highest attention values in red. The decoder attention groups points within an instance which presumably makes it easier to predict bounding boxes.

1. Introduction

3D object detection aims to identify and localize objects in 3D scenes. Such scenes, often represented using *point clouds*, contain an unordered, sparse and irregular set of points captured using a depth scanner. This set-like nature makes point clouds significantly different from the traditional grid-like vision data like images and videos. While there are other 3D representations such as multiple-views [60], voxels [1] or meshes [8], they require additional post-processing to be constructed, and often loose information due to quantization. Hence, point clouds have emerged as a popular 3D representation, and spurred the development of specialized 3D architectures.

Many recent 3D detection models directly work on the 3D points to produce the bounding boxes. Of particular interest, VoteNet [42] casts 3D detection as a set-to-set problem, *i.e.*, transforming an unordered set of inputs (point cloud), into an unordered set of outputs (bounding boxes). VoteNet uses an encoder-decoder architecture: the encoder is a PointNet++ network [44] which converts the unordered point set into a unordered set of point features. The point features are then input to a decoder that produces the 3D bounding boxes. While effective, such architectures have required years of careful development by hand-encoding inductive biases, radii, and designing special 3D operators and loss functions.

In parallel to 3D, set-to-set encoder-decoder models have emerged as a competitive way to model 2D object detection. In particular, the recent Transformer [68] based model, called DETR [4], casts 2D object detection as a set-to-set problem. The self-attention operation in Transformers is designed to be permutation-invariant and capture long range contexts, making them a natural candidate for processing unordered 3D point cloud data. Inspired by this observation, we ask the following question: can we leverage Transformers to learn a 3D object detector without relying on hand-designed inductive biases?

To that end, we develop 3D DEtection TRansformer (3DETR) a simple to implement 3D detection method that uses fewer hand-coded design decisions and also casts detection as a set-to-set problem. We explore the similarities between VoteNet and DETR, as well as between the core mechanisms of PointNet++ and the self-attention of Transformers to build our end-to-end Transformer-based detection model. Our model follows the general encoder-decoder structure that is common to both DETR and VoteNet. For the encoder, we replace the PointNet++ by a standard Transformer applied directly on the point clouds. For the decoder, we consider the parallel decoding strategy from DETR with Transformer layers making two important changes to adapt it to 3D detection, namely non-parametric query embeddings and Fourier positional embeddings [64].

3DETR removes many of the hard coded design decisions in VoteNet and PointNet++ while being simple to implement and understand. Unlike DETR, 3DETR does not employ a ConvNet backbone, and solely relies on Transformers trained from scratch. Our transformer-based detection pipeline is flexible, and as in VoteNet, any component can be replaced by other existing modules. Finally, we show that 3D specific inductive biases can be easily incorporated in 3DETR to further improve its performance. On two standard indoor 3D detection benchmarks, ScanNetV2 and SUN RGB-D we achieve 65.0% AP and 59.0% AP respectively, outperforming an improved VoteNet baseline by 9.5% AP_{50} on ScanNetV2.

2. Related Work

We propose a 3D object detection model composed of Transformer blocks. We build upon prior work in 3D architectures, detection, and Transformers.

Grid-based 3D Architectures. Convolution networks can be applied to irregular 3D data after converting it into regular grids. Projection methods [3, 19, 25, 26, 59, 60, 65] project 3D data into 2D planes and convert it into 2D grids. 3D data can also be converted into a volumetric 3D grid by voxelization [1, 12, 15, 28, 35, 49, 56, 66]. We use 3D point clouds directly since they are suitable for *set* based architectures such as the transformer.

Point cloud Architectures. 3D sensors often acquire data in the form of unordered point clouds. When using unordered point clouds as input, it is desirable to obtain permutation invariant features. Point-wise MLP based architectures [17, 83] such as PointNet [44] and PointNet++ [45] use permutation equivariant set aggregation (downsampling) and pointwise MLPs to learn effective representations. We use a single downsampling operation from [45] to keep the number of input points tractable in our model.

Graph-based models [27, 73] can operate on unordered 3D data. Graphs are constructed from 3D data in a variety of ways – DGCNN [77] and PointWeb [90] use local neighborhoods of points, SPG [24] uses attribute and context similarity and Jiang *et al*. [18] use point-edge interactions.

Finally, continuous point convolution based architectures can also operate on point clouds. The continuous weights can be defined using polynomial functions as in SpiderCNN [80] or linear functions as in Flex-Convolutions [13]. Convolutions can also be applied by soft-assignment matrices [69] or specific ordering [28]. PointConv [78] and KPConv [67] dynamically generate convolutional weights based on the input point coordinates, while InterpCNN [34] uses these coordinates to interpolate weights. We build upon the Transformer [68] which is applicable for sets but not tailored for 3D.

3D Object Detection is a well studied research area where methods predict three dimensional bounding boxes from 3D input data [23, 41, 43, 52, 54, 55, 70, 72, 93]. Many methods avoid expensive 3D operations by using 2D projection. MV3D [6], VoxelNet [92] use a combination of 3D and 2D convolutions. Yan *et al*. [81] simplify the 3D operation while [82] uses a 2D projection, and [76] uses 'pillars' of voxels. We focus on methods that directly use 3D point clouds [40, 51, 75, 85]. PointRCNN [51] and PVR-CNN [50] are 2-stage detection pipelines similar to the popular R-CNN framework [47] for 2D images. While these methods are related to our work, for simplicity we build a single stage detection model as done in [11, 14, 42, 84]. VoteNet [42] uses Hough Voting on sparse point cloud inputs and detects boxes by feature sampling, grouping and voting operations designed for 3D data. VoteNet is a building block for many follow up works. 3D-MPA [11] combines voting with a graph ConvNet for refining object proposals and uses specially designed 3D geometric features for aggregating detections. HGNet [5] improves Hough Voting and uses a hierarchical graph network with feature pyramids. H3DNet [89] improves VoteNet by predicting 3D primitives and uses a geometric loss function. We propose a simple detection method that can serve as a building block for such innovations in 3D detection.

Transformers in Vision. The Transformer architecture by Vaswani *et al*. [68] has been immensely successful across domains like NLP [9, 46], speech recognition [33, 62], image recognition [4, 10, 16, 38, 74], and for cross-domain applications [32, 61, 63]. Transformers are well suited for operating on 3D points since they are naturally permutation invariant. Attention based methods have been used for building 3D point representations for retrieval [87], outdoor 3D detection [29, 36, 86], object classification [83]. Concurrent work [37, 91] also uses the Transformer architecture for 3D. While these methods use 3D specific information to modify the Transformer, we push the limits of the standard Transformer. Our work is inspired by the recent DETR model [4] for object detection in images by Carion *et al*. [4]. Different from Carion *et al*., our model is an end-to-end transformer (no convolutional backbone) that can be trained from scratch and has important design differences such as non-parametric queries to enable 3D detection.

3. Approach

We briefly review prior work in 3D detection and their conceptual similarities to 3DETR. Next, we describe

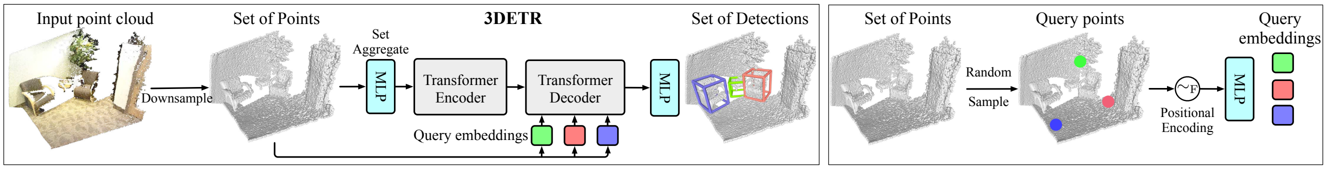

Figure 2: Approach. *(Left)* 3DETR is an end-to-end trainable Transformer that takes a set of 3D points (point cloud) as input and outputs a set of 3D bounding boxes. The Transformer encoder produces a set of per-point features using multiple layers of self-attention. The point features and a set of 'query' embeddings are input to the Transformer decoder that produces a set of boxes. We match the predicted boxes to the ground truth and optimize a set loss. Our model does not use color information (used for visualization only). *(Right)* We randomly sample a set of 'query' points that are embedded and then converted into bounding box predictions by the decoder.

3DETR, simplifications in bounding box parametrization and the simpler set-to-set objective function.

3.1. Preliminaries

The recent VoteNet [42] framework forms the basis for many detection models in 3D, and like our method, is a set-to-set prediction framework. VoteNet uses a specialized 3D encoder and decoder architecture for detection. It combines these models with a Hough Voting loss designed for sparse point clouds. The encoder is a PointNet++ [45] model that uses a combination of multiple downsampling (set-aggregation) and upsampling (feature-propagation) operations that are specifically designed for 3D point clouds. The VoteNet "decoder" predicts bounding boxes in three steps - 1) each point 'votes' for the center coordinate of a box; 2) votes are aggregated within a fixed radius to obtain 'centers'; 3) bounding boxes are predicted around 'centers'. *BoxNet* [42] is a non-voting alternative to VoteNet that randomly samples 'seed' points from the input and treats them as 'centers'. However, BoxNet achieves much worse performance than VoteNet as the voting captures additional context in sparse point clouds and yields better 'center' points. As noted by the authors [42], the multiple hand-encoded radii used in the encoder, decoder, and the loss function are important for detection performance and have been carefully tuned [44, 45].

The Transformer [68] is a generic architecture that can work on set inputs and capture large contexts by computing self-attention between all pairs of input points. Both these properties make it a good candidate model for 3D point clouds. Next, we present our 3DETR model which uses a Transformer for both the encoder and decoder with minimal modifications and has minimal hand-coded information for 3D. 3DETR uses a simpler training and inference procedure. We also highlight similarities and differences to the DETR model for 2D detection.

3.2. 3DETR: Encoder-decoder Transformer

3DETR takes as input a 3D point cloud and predicts the positions of objects in the form of 3D bounding boxes. A point cloud is a unordered set of N points where each point is associated with its 3-dimensional XYZ coordinates. The number of points is very large and we use the set-aggregation downsampling operation from [45] to downsample the points and project them to N' dimensional features. The resulting subset of N' features is passed through an encoder to also obtain a set of N' features. A decoder takes these features as input and predicts multiple bounding boxes using a parallel decoding scheme inspired by [4]. Both encoder and decoder use standard Transformer blocks with 'pre-norm' [21] and we refer the reader to Vaswani *et al*. [68] for details. Fig 2 illustrates our model.

Encoder. The downsample and set-aggregation steps provide a set of N' features of $d = 256$ dimensions using an MLP with two hidden layers of $64, 128$ dimensions. The set of N' features is then passed to a Transformer to also produce a set of N' features of $d = 256$ dimensions. The Transformer applies multiple layers of self-attention and non-linear projections. We do not use downsampling operations in the Transformer, and use the standard self-attention formulation [68]. Thus, the Transformer encoder has no specific modifications for 3D data. We omit positional embeddings of the coordinates from the encoder since the input already contains information about the XYZ coordinates.

Decoder. Following Carion *et al*. [4], we frame detection as a set prediction problem, *i.e*., we simultaneously predict a set of boxes with no particular ordering. This is achieved with a parallel decoder composed of Transformer blocks. This decoder takes as input the N' point features and a set of B query embeddings $\{\mathbf{q}^e_1, \ldots, \mathbf{q}^e_B\}$ to produce a set of B features that are then used to predict 3D-bounding boxes. In our framework, the query embeddings $\mathbf{q}^e$ represent locations in 3D space around which our final 3D bounding boxes are predicted. We use positional embeddings in the decoder as it does not have direct access to the coordinates (operates on encoder features and query embeddings).

Non-parametric query embeddings. Inspired by seed points used in VoteNet and BoxNet [42], we use non-parametric embeddings computed from 'seed' XYZ locations. We sample a set of B 'query' points $\{\mathbf{q}_i\}_{i=1}^B$ randomly from the N' input points (see Fig 2). We use Farthest Point Sampling [45] for the random samples as it ensures a good coverage of the original set of points. We associate each query point $\mathbf{q}_i$ with a query embedding $\mathbf{q}^e_i$, by converting the coordinates of $\mathbf{q}_i$ into Fourier positional embeddings [64] followed by projection with a MLP.

3DETR-m: Inductive biases into 3DETR. As a proof of concept that our model is flexible, we modify our encoder to include inductive biases in 3D data, while keeping the decoder and loss fixed. We leverage a weak inductive bias inspired by PointNet++, *i.e*., local feature aggregation matters more than global aggregation. Such an inductive bias can be easily implemented in Transformers by applying a mask to the self-attention [68]. The resulting model, 3DETR-*m* has a *m*asked self-attention encoder with the same decoder and loss function as 3DETR. 3DETR-m uses a three layer encoder which has an additional downsampling operation (from $N'=2048$ to $N''=1024$ points) after the first layer. Every encoder layer applies a binary mask of $N'' \times N''$ to the self-attention operation. Row i in the mask indicates which of the N'' points lie within the ℓ_2 radius of point i. We use the radius values of $[0.16, 0.64, 1.44]$. Compared to PointNet++, 3DETR-m does not rely on multiple layers of 3D feature aggregation and 3D upsampling.

3.3. Bounding box parametrization and prediction

The encoder-decoder architecture produces a set of B features, that are fed into prediction MLPs to predict bounding boxes. A 3D bounding box has the attributes **(a)** its location, **(b)** size, **(c)** orientation, and **(d)** the class of the object contained in it. We describe the parametrization of these attributes and their associated prediction problems.

The prediction MLPs produce a box around every query coordinate $\mathbf{q}$. **(a) Location:** We use the XYZ coordinates of box's center $\mathbf{c}$. We predict this in terms of an offset $\mathbf{\Delta q}$ that is added to the query coordinates, *i.e*., $\mathbf{c} = \mathbf{q} + \mathbf{\Delta q}$.
(b) Size: Every box is a 3D rectangle and we define its size around the center coordinate $\mathbf{c}$ using XYZ dimensions $\mathbf{d}$.
(c) Orientation: In some settings [53], we must predict the orientation of the box, *i.e*., the angle it forms compared to a given referential. We follow [42] and quantize the angles into 12 bins from $[0, 2\pi)$ and note the quantization residual. Angular prediction involves predicting the the quantized 'class' of the angle and the residual to obtain the continuous angle a. **(d) Semantic Class:** We use a one-hot vector $\mathbf{s}$ to encode the object class contained in the bounding box. We include a 'background' or 'not an object' class as some of the predicted boxes may not contain an object.

Putting together the attributes of a box, we have two quantities: the predicted boxes $\hat{\mathbf{b}}$ and the ground truth boxes $\mathbf{b}$. Each predicted box $\hat{\mathbf{b}} = [\hat{\mathbf{c}}, \hat{\mathbf{d}}, \hat{\mathbf{a}}, \hat{\mathbf{s}}]$ consists of (1) geometric terms $\hat{\mathbf{c}}, \hat{\mathbf{d}} \in [0,1]^3$ that define the box center and dimensions respectively, $\hat{\mathbf{a}} = [\hat{\mathbf{a}}_c, \hat{\mathbf{a}}_r]$ that defines the quantized class and residual for the angle; (2) semantic term $\hat{\mathbf{s}} = [0,1]^{K+1}$ that contains the probability distribution over the K semantic object classes and the 'background' class. The ground truth boxes $\mathbf{b}$ also have the same terms.

3.4. Set Matching and Loss Function

To train the model, we first match the set of B predicted 3D bounding boxes $\{\hat{\mathbf{b}}\}$ to the ground truth bounding boxes $\{\mathbf{b}\}$. While VoteNet uses hand-defined radii to do such set matching, we follow [4] to perform a bipartite graph matching which is simpler, generic (see § 4.2.1) and robust to Non-Maximal Suppression. We compute a loss for each predicted box using its matched ground truth box.
Bipartite Matching. We define a matching cost for a pair of boxes, predicted box $\hat{\mathbf{b}}$ and ground truth box $\mathbf{b}$, using a geometric and a semantic term.

$$C_{\text{match}}(\hat{\mathbf{b}}, \mathbf{b}) = \underbrace{-\lambda_1 \text{GIoU}(\hat{\mathbf{b}}, \mathbf{b}) + \lambda_2 \|\hat{\mathbf{c}} - \mathbf{c}\|_1}_{\text{geometric}} \\ - \underbrace{\lambda_3 \hat{\mathbf{s}}[s_{\text{gt}}] + \lambda_4 (1 - \hat{\mathbf{s}}[s_{\text{bg}}])}_{\text{semantic}} \tag{1}$$

These terms are similar to the loss functions used for training the model and λs are scalars used for a weighted combination. The geometric cost measures the box overlap using GIoU [48] and the distance between the centers of the boxes. Box overlap automatically accounts for the box dimensions, angular rotation and is scale invariant. The semantic cost measures the likelihood of the ground truth class s_{gt} under the predicted distribution $\hat{\mathbf{s}}$ and the likelihood of the box features belonging to a foreground class, *i.e*., of not belonging to the background class s_{bg}.

We compute the optimal bipartite matching between all the predicted boxes $\{\hat{\mathbf{b}}\}$ and ground truth boxes $\{\mathbf{b}\}$ using the Hungarian algorithm [22] as in prior work [4, 58]. As we predict a larger number of boxes than the ground truth, the predicted boxes that do not get matched are considered matched to the 'background' class. This encourages the model to not over-predict, a property that helps our model be robust to Non-Maximal Suppression (see § 5).
Loss function. We use ℓ_1 regression losses for the center and box dimensions, normalizing them both in the range $[0, 1]$ for scale invariance. We use Huber regression loss for the angular residuals and cross-entropy losses for the angular classification and semantic classification.

$$\mathcal{L}_{\text{3DETR}} = \lambda_c \|\hat{\mathbf{c}} - \mathbf{c}\|_1 + \lambda_d \|\hat{\mathbf{d}} - \mathbf{d}\|_1 + \lambda_{ar} \|\hat{\mathbf{a}}_r - \mathbf{a}_r\|_{\text{huber}} \\ - \lambda_{ac} \mathbf{a}_c^\intercal \log \hat{\mathbf{a}}_c - \lambda_s \mathbf{s}_c^\intercal \log \hat{\mathbf{s}}_c \tag{2}$$

Our final loss function is a weighted combination of the above five terms and we provide the full details in the appendix. For predicted boxes matched to the 'background' class, we only compute the semantic classification loss with the background class ground truth label. For datasets with axis-aligned 3D bounding boxes, we also use a loss directly on the GIoU as in [4, 48]. We do not use the GIoU loss for oriented 3D bounding boxes as it is computationally involved.

Intermediate decoder layers. At training time, we use the same bounding box prediction MLPs to predict bounding boxes at every layer in the decoder. We compute the set loss for each layer independently and sum all the losses to train the model. At test time, we only use the bounding boxes predicted from the last decoder layer.

3.5. Implementation Details

We implement 3DETR using PyTorch [39] and use the standard `nn.MultiHeadAttention` module to implement the Transformer. We use a single set aggregation operation [45] to subsample $N'=2048$ points and obtain 256 dimensional point features. The 3DETR encoder has 3 layers where each layer uses multiheaded attention with four heads and a two layer MLP with a 'bottleneck' of 128 hidden dimensions. The 3DETR decoder has 8 layers and closely follows the encoder, except that the MLP hidden dimensions are 256. We use Fourier positional encodings [64] of the XYZ coordinates in the decoder. The bounding box prediction MLPs are two layer MLPs with a hidden dimension of 256. Full architecture details in the appendix Appendix A.1.

All the MLPs and self-attention modules in the model use a dropout [57] of 0.1 except in the decoder where we use a higher dropout of 0.3. 3DETR is optimized using the AdamW optimizer [31] with the learning rate decayed by a cosine learning rate schedule [30] to 10^{-6}, a weight decay of 0.1, and gradient clipping at an ℓ_2 norm of 0.1. We train the model on a single V100 GPU with a batchsize of 8 for 1080 epochs. We use the RandomCuboid augmentation from [88] which reduces overfitting.

4. Experiments

Dataset and metrics. We evaluate models on two standard 3D indoor detection benchmarks - ScanNetV2 [7] and SUN RGB-D-v1 [53]. SUN RGB-D has 5K single-view RGB-D training samples with oriented bounding box annotations for 37 object categories. ScanNetV2 has 1.2K training samples (reconstructed meshes converted to point clouds) with axis-aligned bounding box labels for 18 object categories. For both datasets, we follow the experimental protocol from [42]: we report the detection performance on the val set using mean Average Precision (mAP) at two different IoU thresholds of 0.25 and 0.5, denoted as AP_{25} and AP_{50}. Along with the metric, their protocol evaluates on the 10 most frequent categories for SUN RGB-D.

4.1. 3DETR on 3D Detection

In this set of experiments, we validate 3DETR for 3D detection. We compare it to the BoxNet and VoteNet models since they are conceptually similar to 3DETR and are the foundations of many recent detection models. For fair comparison, we use our own implementation of these models with the same optimization improvements used in 3DETR–leading to a boost of +2-4% AP over the original paper (details in supplemental). We also compare against a state-of-the-art method H3DNet [89] and provide a more detailed comparison against other recent methods in the appendix. 3DETR models use 256 and 128 queries for ScanNetV2 and SUN RGB-D datasets.

Method	**ScanNetV2**		**SUN RGB-D**	
	AP_{25}	AP_{50}	AP_{25}	AP_{50}
BoxNet† [42]	49.0	21.1	52.4	25.1
3DETR	62.7	37.5	56.8	30.1
VoteNet† [42]	60.4	37.5	58.3	33.4
3DETR-m	65.0	47.0	59.0	32.7
H3DNet [89]	67.2	48.1	60.1	39.0

Table 1: Evaluating 3DETR on 3D detection. We compare 3DETR with BoxNet and VoteNet methods and denote by † our improved implementation of these baselines. 3DETR achieves comparable or better performance to these improved baselines despite having fewer hand-coded 3D or detection specific decisions. We report state-of-the-art performance from [89] that improves VoteNet by using 3D primitives. Detailed state-of-the-art comparison in Appendix B.

Observations. We summarize results in Table 1. The comparison between BoxNet and 3DETR is particularly relevant since both methods predict boxes around location queries while VoteNet uses 3D Hough Voting to obtain queries. Our method significantly outperforms BoxNet on both the datasets with a gain of $+13\%$ AP_{25} on ScanNetV2 and $+3.9\%$ AP_{25} on SUN RGB-D. Even when compared with VoteNet, our model achieves competitive performance, with $+2.3\%$ AP_{25} on ScanNetV2 and -1.5% AP_{25} on SUN RGB-D. 3DETR-m, which uses the masked Transformer encoder, achieves comparable performance to VoteNet on SUN RGB-D and a gain of $+4.6\%$ AP_{25} and $+9.5\%$ AP_{50} on ScanNetV2.

Compared to a state-of-the-art method, H3DNet [89], that builds upon VoteNet, 3DETR-m is within a couple of AP_{25} points on both datasets (more detailed comparison in Appendix B). These experiments validate that a encoder-decoder detection model based on the standard Transformer is competitive with similar models tailored for 3D data. Just as the VoteNet model was improved by the innovations of H3DNet [89], HGNet [5], 3D-MPA [11], similar innovations could be integrated to our model in the future.

Qualitative Results. In Fig 3, we visualize a few detections and ground truth boxes from SUN RGB-D. 3DETR detects boxes despite the partial (single-view) depth scans and also predicts amodal bounding boxes or missing annotations on SUN RGB-D.

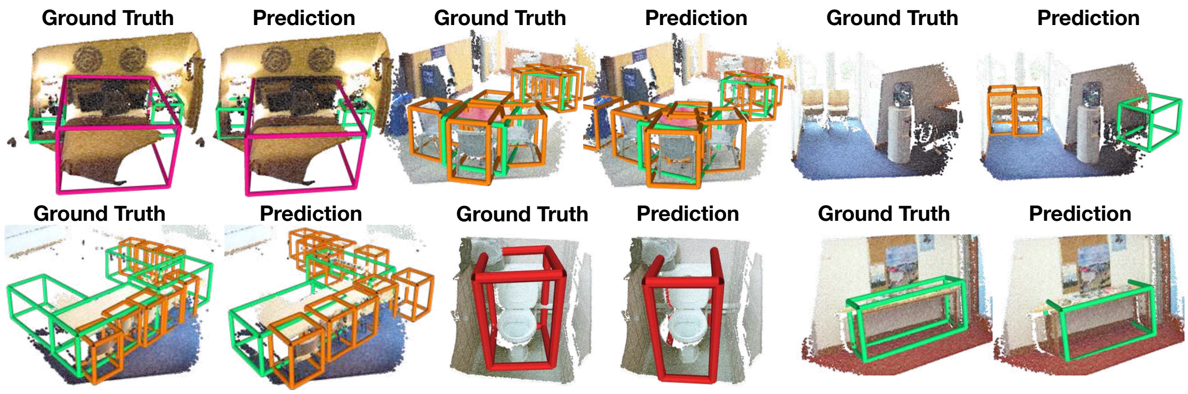

Figure 3: Qualitative Results using 3DETR. Detection results for scenes from the val set of the SUN RGB-D dataset. 3DETR does not use color information (used only for visualization) and predicts boxes from point clouds. 3DETR can detect objects even with single-view depth scans and predicts amodal boxes *e.g.*, the full extent of the bed (top left) including objects missing in the ground truth (top right).

Method	Encoder	Decoder	Loss	ScanNetV2		SUN RGB-D	
				AP_{25}	AP_{50}	AP_{25}	AP_{50}
3DETR	Tx.	Tx.	Set	62.7	37.5	56.8	30.1
	PN++	Tx.	Set	61.4	34.7	56.8	26.9

PN++: PointNet++ [45], Tx.: Transformer, Set loss § 3.4

Table 2: 3DETR with different encoders. We vary the encoder used in 3DETR and observe that the performance is unchanged or slightly worse when moving to a PointNet++ encoder. This suggests that the decoder design and the loss function in 3DETR are compatible with prior 3D specific encoders.

#	Method	Encoder	Decoder	Loss	ScanNetV2		SUN RGB-D	
					AP_{25}	AP_{50}	AP_{25}	AP_{50}
Comparing different decoders								
1	3DETR	Tx.	Tx.	Set	62.7	37.5	56.8	30.1
2		Tx.	Box	Box	31.0	10.2	36.4	14.4
3		Tx.	Vote	Vote	46.1	23.4	47.5	24.9
Comparing different losses								
4		Tx.	Tx.	Box	49.6	20.5	49.5	21.1
5		Tx.	Tx.	Vote	54.0	31.9	53.4	28.3

Tx.: Transformer, Vote/Box loss [42], Set loss § 3.4

Table 3: 3DETR with different decoders and losses. We vary the decoder and losses used with our transformer encoder. As the Box and Vote decoders are only compatible with their losses, we vary the loss function while using them. The Vote loss is compatible with our Transformer encoder-decoder, however a simpler set loss performs the best.

4.2. Analyzing 3DETR

We conduct a series of experiments to understand 3DETR. In § 4.2.1, we explore the similarities between 3DETR, VoteNet and BoxNet. Next, in § 4.2.2, we compare the design decisions in 3DETR that enable 3D detection to the original components in DETR.

4.2.1 Modules of VoteNet and BoxNet vs. 3DETR

The encoder-decoder paradigm is flexible and we can test if the different modules in VoteNet, BoxNet and 3DETR are interchangeable. We focus on the encoders, decoders and losses and report the detection performance in Tables 2 and 3. For simplicity, we denote the decoders and the losses used in BoxNet and VoteNet as Box and Vote respectively. We use PointNet++ to refer to the modified PointNet++ architecture used in VoteNet [42].

Replacing the encoder. We train 3DETR with a PointNet++ encoder (Table 2) and observe that the detection performance is unchanged or slightly worse compared to 3DETR with a transformer encoder. This shows that the design decisions in 3DETR are broadly compatible with prior work, and can be used for designing better encoder models.

Replacing the decoder. In Table 3, we observe that replacing our Transformer-based decoders by Box or Vote decoders leads to poor detection performance on both benchmarks. Additionally, the Box and Vote decoders work only with their respective losses and our preliminary experiments using set loss on these decoders led to worse results. Thus, the drop of performance could be attributed to changing the decoder used with our transformer encoder. We inspect this next by replacing the loss in 3DETR while using the transformer encoder and decoder.

Replacing the loss. We train 3DETR, *i.e.*, both Transformer encoder and decoder with the Box and Vote losses. We observe (Table 3 rows 4 and 5) that this leads to similar degradation in performance, suggesting that the losses are not applicable to our model. This is not surprising since the design decisions, *e.g.*, voting radius, aggregation radius *etc.* in the Vote loss was specifically designed for radius parameters in the PointNet++ encoder [45]. This set of observations exposes that the decoder and loss function used in VoteNet depend greatly on the nature of the encoder (additional results in Appendix B.4). In contrast, our set loss has no design decisions specific to our encoder-decoder.

Method	input	mAcc	OA
PointNet++ [45]	point	–	91.9
SpecGCN [71]	point	–	92.1
DGCNN [77]	point	90.2	92.2
PointWeb [90]	point	89.4	92.3
SpiderCNN [80]	point	–	92.4
PointConv [78]	point	–	92.5
KPConv [67]	point	–	92.9
InterpCNN [34]	point	–	93.0
3DETR encoder (Ours)	point	89.1	92.1
3DETR-m encoder (Ours)	point	89.9	91.9

Table 4: Shape classification. We report shape classification results by training our Transformer encoder model. Our model performs competitively with architectures designed for 3D suggesting that our design decisions can extend beyond detection and be useful for other tasks.

Visualizing self-attention. We visualize the self-attention in the decoder in Fig 1. The decoder focuses on whole instances and groups points within instances. This presumably makes it easier to predict bounding boxes for each instance. We provide visualizations for the encoder self-attention in the supplemental.

Encoder applied to Shape classification. To verify that our encoder design is not specific to the detection task we test the encoder on shape classification of of models including 3D Warehouse [79].

We use the three layer encoder from 3DETR with vanilla self-attention (no decoder) or the three layer encoder from 3DETR-m. To obtain global features for the point cloud, we use the '`CLS` token' formulation from Transformer, *i.e.*, append a constant point to the input and use this point's output encoder features as global features (see supplemental for details). The global features from the encoder are input to a 2-layer MLP to perform shape classification. Table 4 shows that both the 3DETR and 3DETR-m encoders are competitive with state-of-the-art encoders tailored for 3D. These results suggest that our encoder design is not specific to detection and can be used for other 3D tasks.

4.2.2 Design decisions in 3DETR

Our model is inspired by the DETR [4] architecture but has major differences - (1) it is an end-to-end transformer without a ConvNet, (2) it is trained from scratch (3) uses non-parametric queries and (4) Fourier positional embeddings. In Table 5, we show the impact of the last two differences by evaluating various versions of our model on ScanNetV2. The version with minimal modifications is a DETR model applied to 3D with our training and loss function.

First, this version does not perform well on the ScanNetV2 benchmark, achieving 15% AP_{25}. However, when replacing the parametric queries by non-parametric queries,

#	Method	Positional Embedding		Query Type	ScanNetV2	
		Encoder	Decoder		AP_{25}	AP_{50}
1	3DETR	-	Fourier	np + Fourier	62.7	37.5
2		Fourier	Fourier	np + Fourier	61.8	37.0
3		Sine	Sine	np + Sine	55.8	30.9
4		-	-	np + Sine	31.3	10.8
5	DETR [4]†	Sine	Sine	parametric [4]	15.4	5.3

np: non-parametric query (§ 3.2)

Table 5: Decoder Query Type and Positional Embedding. We how using non-parametric queries and Fourier positional embeddings [64] affect detection performance. DETR's parametric queries do not work well for 3D detection (rows 3, 5). The standard choice [4, 68] of sinusoidal positional embeddings is worse than Fourier embeddings (rows 2, 3). † - DETR is designed for 2D image detection and we adapt it for 3D detection.

Method	NMS	No NMS
VoteNet [42]	60.4	10.7
3DETR (ours)	62.7	59.5

Table 6: Effect of NMS. We report the detection performance (AP_{25}) for 3DETR and VoteNet on ScanNetV2. 3DETR works without NMS at test time because the set matching loss discourages excess predicted boxes.

we observe a significant improvement of +40% in AP_{25} (Table 5 rows 3 and 5). In fact, only using the non-parametric queries (row 4) without positional embeddings doubles the performance. This shows the importance of using non-parametric queries with 3D point clouds. A reason is that point clouds are irregular and sparse, making the learning of parametric queries harder than on a 2D image grids. Non-parametric queries are directly sampled from the point clouds and hence are less impacted by these irregularities. Unlike the fixed number of parametric queries in DETR, non-parametric queries easily enable the use different number of queries at train and test time (see § 5.1).

Finally, replacing the sinusoidal positional embedding by the low-frequency Fourier encodings of [64] provides an additional improvement of +5% in AP_{25} (Table 5 rows 2 and 3). As a side note, using positional encodings benefits the decoder more than the encoder because the decoder does not have direct access to coordinates.

5. Ablations

We conduct a series of ablation experiments to understand the components of 3DETR with settings from § 4.

Effect of NMS. 3DETR uses the set loss of DETR (§ 3.4) that forces a 1-to-1 mapping between the ground truth box and the predicted box. This loss penalizes models that predict too many boxes, since excess predictions are not matched to ground truth. In contrast, the loss used in VoteNet [42] does not discourage multiple predictions of the same object and thus relies on Non-Maximal Suppres-

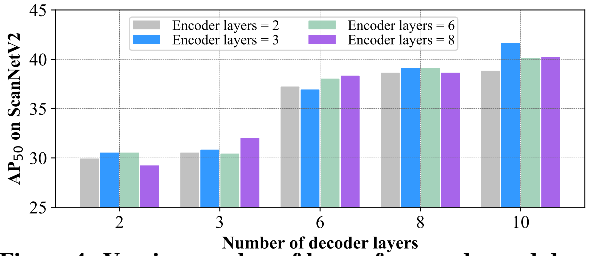

Figure 4: Varying number of layers for encoder and decoder. We train different models with varying number of encoder and decoder layers and analyze the impact on detection performance on ScanNetV2. Increasing the number of layers in either the encoder or decoder has a positive effect, but a higher number of decoder layers matters more than the encoder layers.

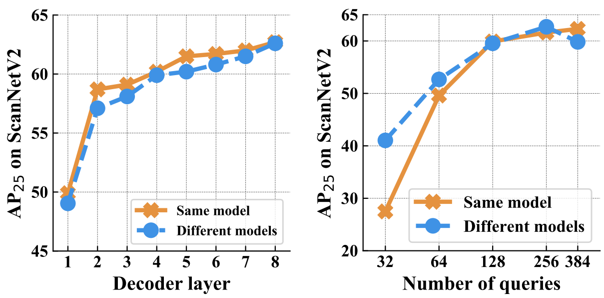

Figure 5: Adapting compute at test time. We change the number of decoder layers or the number of queries used at test time for a 3DETR model ('same model'). We compare this to different models trained with reduced depth of the decoder (left) or with different number of queries (right). 3DETR can adapt to different test time conditions and performs favorably compared to different models trained for the test time conditions.

sion to remove them as a post-processing step. We compare 3DETR and VoteNet with and without NMS in Table 6 with the detection AP metric, which penalizes duplicate detections. Without NMS, 3DETR drops in performance by only 3% AP while VoteNet drops by 50%, showing our set loss works without NMS.

Effect of encoder/decoder layers. We assess the importance of the number of layers in the encoder and decoder in Fig 4. While a higher number of layers improves detection performance in general, adding the layers in the decoder instead of the encoder has a greater impact on performance. For instance, for a model with three encoder and three decoder layers, adding five decoder layers improves performance by +7% AP_{50} while adding five encoder layers improves by +2%AP_{50}. This preference toward the decoder arises because in our parallel decoder, each layer further refines the prediction quality of the bounding boxes.

5.1. Adapting computation to inference constraints

An advantage of our model is that we can adapt its computation during inference by using less layers in the decoder or queries to predict boxes without retraining.

Adapting decoder depth. The parallel decoder of 3DETR is trained to predict boxes at each layer with the same bounding box prediction MLPs. Thus far, in all our results we used the predictions only from the last decoder layer. We now test the performance of the intermediate layers for a decoder with six layers in Fig 5 (left). We compare this to training different models with a varying number of decoder layers. We make two observations - (1) similar to Fig 4, detection performance increases with the number of decoder layers; and (2) more importantly, the same model with reduced depth at test time performs as well or better than models trained from scratch with reduced depth. This second property is shared with the DETR, but not with VoteNet. It allows adapting the number of layers in the decoder to a computation budget during inference without retraining.

Adapting number of queries. As we increase the number of queries, 3DETR predicts more bounding boxes, resulting in better performance at a cost of longer running time. However, our non-parametric queries in 3DETR allow us to adapt the number of box predictions to trade performance for running time. Note that this is also possible with VoteNet, but not with DETR. In Fig 5 (right), we compare changing the number of queries at test time to different models trained with varying number of queries. The same 3DETR model can adapt to a varying number of queries at test time and performs comparably to different models. Performance increases until the number of queries is enough to cover the point cloud well. We found this adaptation to number of queries at test time works best with a 3DETR model trained with 128 queries (see Appendix B for other models). This adaptive computation is promising and research into efficient self-attention should benefit our model. We provide inference time comparisons to VoteNet in Appendix A.1 for different versions of the 3DETR model.

6. Conclusion

We presented 3DETR, an end-to-end Transformer model for 3D detection on point clouds. 3DETR requires few 3D specific design decisions or hyperparameters. We show that using non-parametric queries and Fourier encodings is critical for good 3D detection performance. Our proposed design decisions enable powerful Transformers for 3D detection, and also benefit other 3D tasks like shape classification. Additionally, our set loss function generalizes to prior 3D architectures. In general, 3DETR is a flexible framework and can easily incorporate prior components used in 3D detection and can be leveraged to build more advanced 3D detectors. Finally, it also combines the flexibility of both VoteNet and DETR, allowing for a variable number of predictions at test time (like VoteNet) with a variable number of decoder layers (like DETR).

Acknowledgments: We thank Zaiwei Zhang for helpful discussions and Laurens van der Maaten for feedback on the paper.

References

[1] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR*, 2:7, 2017.

[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] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z Chen, and Jian Wu. A hierarchical graph network for 3d object detection on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 392–401, 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] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] Boris Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[9] 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.

[10] 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.

[11] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9031–9040, 2020.

[12] Ben Graham. Sparse 3d convolutional neural networks. *arXiv preprint arXiv:1505.02890*, 2015.

[13] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. Flex-convolution. In *Asian Conference on Computer Vision*, pages 105–122. Springer, 2018.

[14] JunYoung Gwak, Christopher B Choy, and Silvio Savarese. Generative sparse detection networks for 3d single-shot object detection. In *European conference on computer vision*, 2020.

[15] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

[16] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[17] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[18] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10433–10441, 2019.

[19] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[21] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

[22] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[23] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9256–9266, 2019.

[24] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[25] 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.

[26] 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.

[27] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276, 2019.

[28] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[29] Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai. Tanet: Robust 3d object detection from point clouds with triple attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11677–11684, 2020.

[30] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[32] 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.

[33] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Rwth asr systems for librispeech: Hybrid vs attention–w/o data augmentation. *arXiv preprint arXiv:1905.03072*, 2019.

[34] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1578–1587, 2019.

[35] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[36] Anshul Paigwar, Ozgur Erkent, Christian Wolf, and Christian Laugier. Attentional pointnet for 3d-object detection in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

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

[38] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[40] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2019.

[41] Trung T Pham, Markus Eich, Ian Reid, and Gordon Wyeth. Geometrically consistent plane extraction for dense indoor 3d maps segmentation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4199–4204. IEEE, 2016.

[42] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[43] 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.

[44] 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.

[45] Charles R 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.

[46] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems*, 2015.

[48] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union. June 2019.

[49] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[50] 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.

[51] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 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.

[52] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

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

[54] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, 2014.

[55] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[56] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene com-

pletion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017.

[57] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[58] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016.

[59] 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.

[60] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 945–953, 2015.

[61] 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.

[62] Gabriel Synnaeve, Qiantong Xu, Jacob Kahn, Tatiana Likhomanenko, Edouard Grave, Vineel Pratap, Anuroop Sriram, Vitaliy Liptchinsky, and Ronan Collobert. End-to-end asr: from supervised to semi-supervised learning with modern architectures. *arXiv preprint arXiv:1911.08460*, 2019.

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

[64] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[65] 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.

[66] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE, 2017.

[67] 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.

[68] 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.

[69] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2598–2606, 2018.

[70] 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.

[71] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–66, 2018.

[72] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607. Rome, Italy, 2015.

[73] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019.

[74] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[75] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4096–4105, 2019.

[76] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. *arXiv preprint arXiv:2007.10323*, 2020.

[77] 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.

[78] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[79] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[80] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

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

[82] 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.

[83] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019.

[84] 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.

[85] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019.

[86] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11495–11504, 2020.

[87] Wenxiao Zhang and Chunxia Xiao. Pcan: 3d attention map learning using contextual information for point cloud based retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12436–12445, 2019.

[88] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[89] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[90] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019.

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

[92] 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.

[93] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.