# 3D Object Detection with Pointformer

Xuran Pan[1*] Zhuofan Xia[1*] Shiji Song[1] Li Erran Li[2†] Gao Huang[1‡]
[1]Department of Automation, Tsinghua University, Beijing, China
Beijing National Research Center for Information Science and Technology (BNRist),
[2]Alexa AI, Amazon / Columbia University
{pxr18, xzf20}@mails.tsinghua.edu.cn, erranlli@gmail.com,
{shijis, gaohuang}@tsinghua.edu.cn

## Abstract

*Feature learning for 3D object detection from point clouds is very challenging due to the irregularity of 3D point cloud data. In this paper, we propose Pointformer, a Transformer backbone designed for 3D point clouds to learn features effectively. Specifically, a Local Transformer module is employed to model interactions among points in a local region, which learns context-dependent region features at an object level. A Global Transformer is designed to learn context-aware representations at the scene level. To further capture the dependencies among multi-scale representations, we propose Local-Global Transformer to integrate local features with global features from higher resolution. In addition, we introduce an efficient coordinate refinement module to shift down-sampled points closer to object centroids, which improves object proposal generation. We use Pointformer as the backbone for state-of-the-art object detection models and demonstrate significant improvements over original models on both indoor and outdoor datasets.*

## 1. Introduction

3D object detection in point clouds is essential for many real-world applications such as autonomous driving [10] and augmented reality [18]. Compared to images, 3D point clouds can provide detailed geometry and capture 3D structure of the scene. On the other hand, point clouds are irregular, which can not be processed by powerful deep learning models, such as convolutional neural networks directly. This poses a big challenge for effective feature learning.

The common feature processing methods in 3D detection can be roughly categorized into three types, based on the form of point cloud representations. *Voxel-based* approaches [28, 12, 42] gridify the irregular point clouds into regular voxels and are followed by sparse 3D convolutions to learn high dimensional features. Though effective, voxel-based approaches face the dilemma between efficiency and accuracy. Specifically, using smaller voxels gains more precision, but suffers from higher computational cost. Conversely, using larger voxels misses potential local details in the crowded voxels.

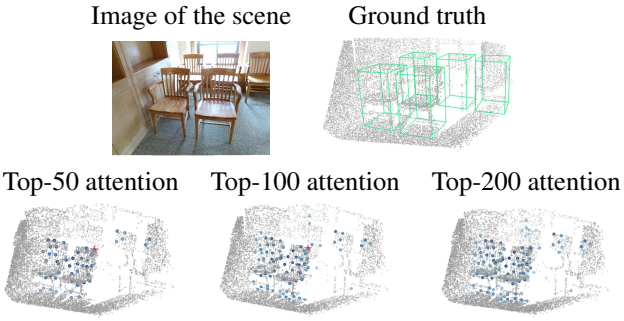


Figure 1. **Attention maps directly from *Pointformer* block, darker blue indicates stronger attention.** For the key point (star), *Pointformer* first focuses on the local region of the same object (the back of the chair), then spreads the attention to other regions (the legs), finally attends to points from other objects globally (other chairs), leveraging both local and global dependencies.

Alternatively, *point-based* approaches [25], inspired by the success of PointNet [21] and its variants, consume raw points directly to learn 3D representations, which mitigates the drawback of converting point clouds to some regular structures. Leveraging learning techniques for point sets, point-based approaches avoid voxelization-induced information loss and take advantage of the sparsity in point clouds by only computing on valid data points. Nevertheless, due to the irregularity of point cloud data, point-based learning operations have to be permutation-invariant and adaptive to the input size. To achieve this, it learns simple symmetric functions (e.g. using point-wise feedforward networks with pooling functions) which highly restricts its representation power.

*Equal contribution.
†Work done prior to Amazon.
‡Corresponding author.

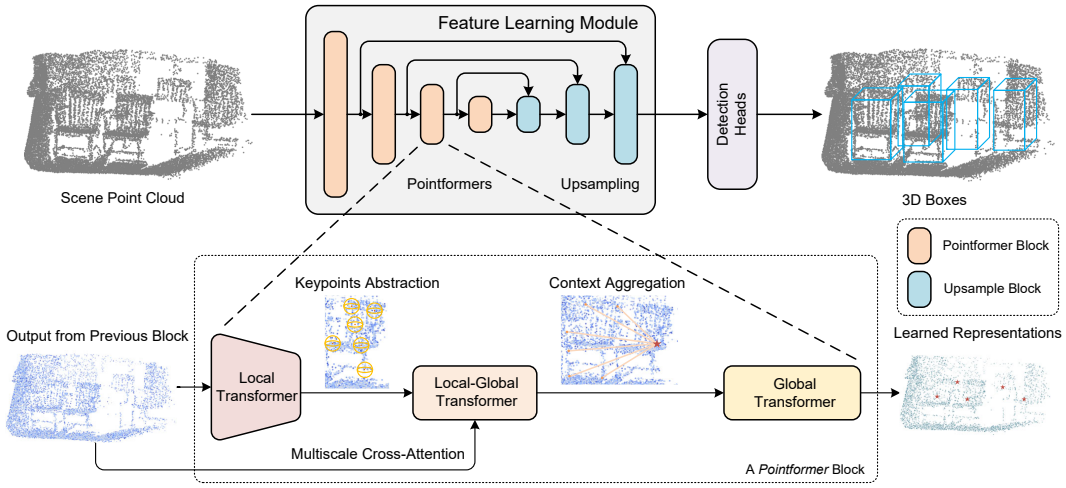


Figure 2. The **Pointformer** backbone for 3D object detection in point clouds. A basic feature learning block consists of three parts: a Local Transformer to model interactions in the local region; a Local-Global Transformer to integrate local features with global information; a Global Transformer to capture context-aware representations at the scene level.

*Hybrid approaches* [41, 15, 39, 24] attempt to combine both voxel-based and point-based representations. [41, 15] leverages PointNet features at the voxel level and a column of voxels (pillar) level respectively. [39, 24] deeply integrate voxel features and PointNet features at the scene level. However, the fundamental difference between the two representations could pose a limit on the effectiveness of these approaches for 3D point-cloud feature learning.

To address the above limitations, we resort to the Transformer [30] models, which have achieved great success in the field of natural language processing. Transformer models [8] are very effective at learning context-dependent representations and capturing long range dependencies in the input sequence. Transformer and the associate self-attention mechanism not only meet the demand of permutation invariance, but also are proved to be highly expressive. Specifically, [6] proves that self-attention is at least as expressive as convolution. Currently, self-attention has been successfully applied to classification [23] and 2D object detection [2] in computer vision. However, the straightforward application of Transformer to 3D point clouds is prohibitively expensive because computation cost grows quadratically with the input size.

To this end, we propose *Pointformer*, a backbone for 3D point clouds to learn features more effectively by leveraging the superiority of the Transformer models on set-structured data. As shown in Figure 2, Pointformer is a U-Net structure with multi-scale Pointformer blocks. A Pointformer block consists of Transformer-based modules that are both expressive and friendly to the 3D object detection task. First, a Local Transformer (LT) module is employed to model interactions among points in the local region, which learns context-dependent region features at an object level. Second, a coordinate refinement module is proposed to adjust centroids sampled from Furthest Point Sampling (FPS) which improves the quality of generated object proposals. Third, we propose Local-Global Transformer (LGT) to integrate local features with global features from higher resolution. Finally, Global Transformer (GT) module is designed to learn context-aware representations at the scene level. As illustrated in Figure 1, Pointformer can capture both local and global dependencies, thus boosting the performance of feature learning for scenes with multiple cluttered objects.

Extensive experiments have been conducted on several detection benchmarks to verify the effectiveness of our approach. We use the proposed Pointformer as the backbone for three object detection models, CBGS [42], VoteNet [19], and PointRCNN [25], and conduct experiments on three indoor and outdoor datasets, SUN-RGBD [27], KITTI [10], and nuScenes [1] respectively. We observe significant improvements over the original models on all experiment settings, which demonstrates the effectiveness of our method.

In summary, we make the following contributions:

- We propose a pure transformer model, *Pointformer*, which serves as a highly effective feature learning backbone for 3D point clouds. Pointformer is permutation invariant, local and global context-aware.
- We show that Pointformer can be easily applied as the drop-in replacement backbone for state-of-the-art 3D object detectors for the point cloud.
- We perform extensive experiments using Pointformer as the backbone for three state-of-the-art 3D object detectors, and show significant performance gains on several benchmarks including both indoor and outdoor datasets. This demonstrates that the versatility of Pointformer as 3D object detectors are typically designed and optimized for either indoor or outdoor only.

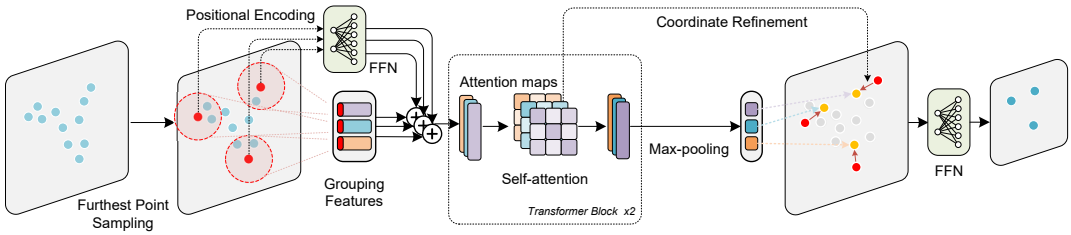


Figure 3. Illustration of the Local Transformer. Input points are first down-sampled by FPS and generate local regions by ball query. Transformer block takes point features and coordinates as input and generate aggregated features for the local region. To further adjust the centroid points, attention maps from the last Transformer layer are adopted for coordinate refinement. As a result, points are pushed closer to the object centers instead of surfaces.

## 2. Related Work

**Feature learning for 3D point clouds.** Prior work includes feature learning on voxelized grids, direct feature learning on point clouds and the hybrid of the two. 3D sparse convolution [11] is very effective on voxel grids. For direct feature learning, PointNet [21] and PointNet++ [22] learn point-wise features and region features using feed-forward networks and simple symmetric functions (e.g. max) respectively. PCCN [33] generalizes convolution to non-grid structured data by exploiting parameterized kernel functions that span the full continuous vector space. EdgeConv [34] exchanges local neighborhood information and acts on graphs dynamically computed in each layer of the network. Hybrid methods combine both types of features at the local level [41, 15] or at the network level [39, 24].

**Transformers in computer vision.** Image GPT [3] is the first to adopt the Transformers in 2D image classification task for unsupervised pretraining. Further, ViT [9] extends this scheme to large scale supervised learning on images. For high level vision tasks, DETR [2] and Deformable DETR [43] leverage the advantages of Transformers in 2D object detection. Set Transformer [16] uses attention mechanisms to model interactions among elements in the input set. In the field of 3D vision, PAT [38] designs novel group shuffle attentions to capture long range dependencies in point clouds. To the best of our knowledge, we are the first to propose a pure Transformer model for 3D points clouds feature learning with carefully designed Transformer blocks and a positional encoding module to capture geometric and rich context information.

**3D object detection in point clouds.** Detectors are designed either with point clouds as the only input [42, 41, 15, 39, 24, 25, 19, 26, 35, 40] or fusing multiple sensor modalities such as LiDAR and camera [20, 17, 31]. Their backbones are designed with the aforementioned feature learning approaches. We focus on point cloud only object detection. In this category, VoxelNet [41] divides the point cloud into voxels, followed by 3D convolutions to extract features. VoteNet [19] devises a novel 3D proposal mechanism using deep Hough voting, before H3DNet [40] makes further investigations on geometric primitives. In addition, MLCVNet [35] focuses more on contextual information aggregation based on VoteNet, and PointGNN [26] exploits graph learning methods in point cloud detection. We show that our novel Transformer based model, Pointformer, can be used as a drop-in replacement for voxel-based detector, CBGS [42] and point-based detectors, VoteNet [19] and PointRCNN [25].

## 3. Pointformer

Feature learning for 3D point clouds needs to confront its irregular and unordered nature as well as its varying size. Prior work utilizes simple symmetric functions, e.g., point-wise feedforward networks with pooling functions [21, 22], or resorts to the techniques in graph neural networks by aggregating information from the local neighborhood [34]. However, the former is not effective in incorporating local context-dependent features beyond the capability of the simple symmetric functions; the latter focuses on the message passing between the center point and its neighbors while neglecting the feature correlations among the neighbor points. Additionally, global representations are also informative but rarely used in 3D object detection tasks.

In this paper, we design Transformer-based modules for point set operations which not only increase the expressiveness of extracting local features, but incorporate global information into point representations as well. As shown in Figure 2, a Pointformer block mainly consists of three parts: Local Transformer (LT), Local-Global Transformer (LGT) and Global Transformer (GT). For each block, LT first receives the output from its previous block (high resolution) and extracts features for a new set with fewer elements (low resolution). Then, LGT uses the multi-scale cross-attention mechanism to integrate features from both resolutions. Lastly, GT is adopted to capture context-aware representations. As for the up-sampling block, we follow PointNet++ and adopt the feature propagation module for its simplicity.

### 3.1. Background

We first revisit the general formulation of the Transformer model. Let $F = \{f_i\}$ and $X = \{x_i\}$ denote a set

of input features and their positions, where $f_i$ and $x_i$ represent the feature and position of token $i$, respectively. Then, a Transformer block comprises of a multi-head self-attention module and feedforward network:

$$q_i^{(m)} = f_i W_q^{(m)}, k_i^{(m)} = f_i W_k^{(m)}, v_i^{(m)} = f_i W_v^{(m)}, \quad (1)$$

$$y_i^{(m)} = \sum_j \sigma(q_i^{(m)} k_j^{(m)} / \sqrt{d} + \mathrm{PE}(x_i, x_j)) v_j^{(m)}, \quad (2)$$

$$y_i = f_i + \mathrm{Concat}(y_i^{(0)}, y_i^{(1)}, \ldots, y_i^{(M-1)}), \quad (3)$$

$$o_i = y_i + \mathrm{FFN}(y_i), \quad (4)$$

where $W_q, W_k, W_v$ are projections for query, key and value. $m$ is the index of $M$ attention heads and $d$ is the feature dimension. $\mathrm{PE}(\cdot)$ is the positional encoding function for input positions, and $\mathrm{FFN}(\cdot)$ represents a position-wise feedforward network. $\sigma(\cdot)$ is a normalization function and *SoftMax* is mostly adopted.

In the following sections, for simplicity, we use

$$O = \mathrm{Transblock}(F, \mathrm{PE}(X)), \quad (5)$$

to represent the basic Transformer block (Eq.(1)$\sim$ Eq.(4)). Readers can refer to [30] for further details.

### 3.2. Local Transformer

In order to build a hierarchical representation for a point cloud scene, we follow the high level methodology to build feature learning blocks on different resolutions [22]. Given an input point cloud $\mathcal{P} = \{x_1, x_2, \ldots, x_N\}$, we first use furthest point sampling (FPS) to choose a subset of points $\{x_{c_1}, x_{c_2}, \ldots, x_{c_{N'}}\}$ as a set of centroids. For each centroid, ball query is applied to generate $K$ points in the local region within a given radius. Then we group these features around the centroids, and feed them as a point sequence to a Transformer layer, as shown in Figure 3. Let $\{x_i, f_i\}_t$ denote the local region for $t_{\mathrm{th}}$ centroid, where $x_i \in \mathbb{R}^3$ and $f_i \in \mathbb{R}^C$ represent the coordinates and features of the $i$-th points in the group, respectively. Subsequently, a shared $L$-layer Transformer block is applied to all local regions which receives the input of $\{x_i, f_i\}_t$ as follows:

$$f_i^{(0)} = \mathrm{FFN}(f_i), \ \forall i \in \mathcal{N}(x_{c_t}), \quad (6)$$

$$F^{(l+1)} = \mathrm{Transblock}(F^{(l)}, \mathrm{PE}(X)), l = 0, .., L-1, \quad (7)$$

where $F = \{f_i | i \in \mathcal{N}(x_{c_t})\}$ and $X = \{x_i | i \in \mathcal{N}(x_{c_t})\}$ denote the set of features and coordinates in the local region with centroid $x_{c_t}$.

Compared to the existing local feature extraction modules in [36, 37, 29], the proposed Local Transformer has several advantages. First, the dense self-attention operation in the Transformer block greatly enhances its expressiveness. Several graph learning based approaches can be approximated as special cases of the LT module with learned parameter space carefully designed. For instance, a generalized graph feature learning function can be formulated as:

$$e_{ij} = \mathrm{FFN}(\mathrm{FFN}(x_i \oplus x_j) + \mathrm{FFN}(f_i \oplus f_j)), \quad (8)$$

$$f_i' = \mathcal{A}(\sigma(e_{ij}) \times \mathrm{FFN}(f_j), \forall j \in \mathcal{N}(x_i)), \quad (9)$$

where most of the models utilize summation as the aggregation function $\mathcal{A}$ and the operation $\oplus$ is chosen from {Concatenation, Plus, Inner-product}. Therefore, the edge function $e_{ij}$ is at most a quadratic function of $\{x_i, x_j, f_i, f_j\}$. For a one-layer Transformer block, the learning module can be formulated with the inner-product self-attention mechanism as follows:

$$e_{ij} = \frac{f_i W_q W_k^T f_j^T}{\sqrt{d}} + \mathrm{PE}(x_i, x_j), \quad (10)$$

$$f_i' = \mathcal{A}(\sigma(e_{ij} \times \mathrm{FFN}(f_j), \forall j \in \mathcal{N}(x_i)), \quad (11)$$

where $d$ is the feature dimension of $f_i$ and $f_j$. We can observe that the edge function is also a quadratic function of $\{x_i, x_j, f_i, f_j\}$. With sufficient number of layers in FFNs, the graph-based feature learning module has the same expressive power as a one-layer Transformer encoder. When it comes to *Pointformer*, as we stack more Transformer layers in the block, the expressiveness of our module is further increased and can extract better representations.

Moreover, feature correlations among the neighbor points are also considered, which are commonly omitted in other models. Under some circumstances, neighbor points can be even more informative than the centroid point. Therefore, by leveraging message passing among all points, features in the local region are equally considered, which makes the local feature extraction module more effective.

### 3.3. Coordinate Refinement

Furthest point sampling (FPS) is widely used in many point cloud frameworks, as it can generate a relatively uniform sampled points while keeping the original shape, which ensures that a large fraction of the points can be covered with limited centroids. However, there are two main issues in FPS: (1) It is notoriously sensitive to the outlier points, leading to highly instability especially when dealing with real-world point cloud data. (2) Sampled points from FPS must be a subset of original point clouds, which makes it challenging to infer the original geometric information in the cases that objects are partially occluded or not enough points of an object are captured. Considering that points are mostly captured on the surface of objects, the second issue may become more critical as the proposals are generated from sampled points, resulting in a natural gap between the proposal and ground truth.

To overcome the aforementioned drawbacks, we propose a point coordinate refinement module with the help of the self-attention maps. As shown in Figure 3, we first take out

the self-attention map of the last layer of the Transformer block for each attention head. Then, we compute the average of the attention maps and utilize the particular row for the centroid point as a weight vector:

$$W = \frac{1}{M}\sum_{m=1}^{M} A_{0,:}^{(m)}, \tag{12}$$

where $M$ represents the number of attention heads and $A^{(m)}$ is the attention map for the $m_{\text{th}}$ head. Lastly, the refined centroid coordinates are computed as weighted average of all points in the local region:

$$x'_{c_t} = \sum_{k=1}^{K} w_k x_k, \tag{13}$$

where $w_k$ is the $k_{\text{th}}$ entry of $W$. With the proposed coordinate refinement module, centroid points are adaptively moving closer to object centers. Moreover, by utilizing the self-attention map, our module introduces little computational cost and no additional learning parameters, making the refinement process more efficient.

### 3.4. Global Transformer

Global information representing scene contexts and feature correlations between different objects is also valuable in the detection tasks. Prior work using PointNet++ [22] or sparse 3D convolution to extract high level features for 3D point clouds enlarges the receptive field as the depth of their networks increases. However, this has limitations on modeling long-range interactions.

As a remedy, we leverage the power of Transformer modules on modeling non-local relations and propose a Global Transformer to achieve message passing through the whole point cloud. Specifically, all points are gathered to a single group $\mathcal{P}$ and serves as input to a Transformer module. The formulation for GT is summarized as follows:

$$f_i^{(0)} = \text{FFN}(f_i),\ \forall i \in \mathcal{P}, \tag{14}$$

$$F^{(l+1)} = \text{Transblock}(F^{(l)}, \text{PE}(X)), l = 0, .., L-1. \tag{15}$$

By leveraging the Transformer on the scene level, we can capture the context-aware representations and promote message passing among different objects. Moreover, global representations can be particularly helpful for detecting objects with very few points.

### 3.5. Local-Global Transformer

Local-Global Transformer is also a key module to combine the local and global features extracted by the LT and GT modules. As shown in Figure 2, the LGT adopts a multi-scale cross-attention module and generates relations between low resolution centroids and high resolution points. Formally, we apply cross attention similar to the encoder-decoder attention used in Transformer. The output of LT serves as query and the output of GT from the higher resolution is used as key and value. With the $L$-layer Transformer block, the module is formulated as:

$$f^{(0)} = \text{FFN}(f_i), \forall i \in \mathcal{P}^l, \tag{16}$$

$$f'_j = \text{FFN}(f_j), \forall j \in \mathcal{P}^h, \tag{17}$$

$$F^{(l+1)} = \text{Transblock}(F^{(l)}, F'_j, \text{PE}(X)), l = 0, .., L-1, \tag{18}$$

where $\mathcal{P}^l$ (keypoints, the output of LT in Figure 2) and $\mathcal{P}^h$ (the input of a Pointformer block in Figure 2) represent subsamples of point cloud $\mathcal{P}$ from low and high resolution respectively. Through the Local-Global Transformer module, we utilize whole centroid points to integrate global information via an attention mechanism, which makes the feature learning of both more effective.

### 3.6. Positional Encoding

Positional encoding is an integral part of Transformer models as it is the only mechanism that encodes position information for each token in the input sequence. When adapting Transformers for 3D point cloud data, positional encoding plays a more critical role as the coordinates of point clouds are valuable features indicating the local structures. Compared to the techniques used in natural language processing, we propose a simple and yet efficient approach. For all Transformer modules, coordinates of each input point are firstly mapped to the feature dimension. Then, we subtract the coordinates of the query and key points and use relative positions for encoding. The encoding function is formalized as:

$$\text{PE}(x_i, x_j) = \text{FFN}(x_i - x_j). \tag{19}$$

### 3.7. Computational Cost Reduction

Since Pointformer is a pure attention model based on Transformer blocks, it suffers from extremely heavy computational overhead. Applying a conventional Transformer to a point cloud with $n$ points consumes $O(n^2)$ time and memory, leading to much more training cost.

Some recent advances in efficient Transformers have mitigated this issue [14, 13, 32, 4, 35], among which Linformer [32] reduces the complexity to $O(n)$ by low-rank factorization of the original attention. Under the hypothesis that the self attention mechanism is low rank, i.e. the rank of the $n \times n$ attention matrix

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right), \tag{20}$$

is much smaller than $n$, Linformer projects the $n$-dimension keys and values to the ones with lower dimension $k \ll n$, and $k$ is closer to the rank of $A$. Therefore, the $i$-th head in the projected multi-head self-attention is

$$\text{head}_i = \text{softmax}\left(\frac{Q(E_i K)^\top}{\sqrt{d_k}}\right) F_i V, \tag{21}$$

| Method | Modality | Car(IoU=0.7) | | | Pedestrian (IoU=0.5) | | | Cyclist (IOU=0.5) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Easy | Moderate | Hard | Easy | Moderate | Hard | Easy | Moderate | Hard |
| PointRCNN [25] | LiDAR | 85.94 | 75.76 | 68.32 | 49.43 | 41.78 | 38.63 | 73.93 | 59.60 | 53.59 |
| + Pointformer | LiDAR | **87.13** | **77.06** | **69.25** | **50.67** | **42.43** | **39.60** | **75.01** | **59.80** | **53.99** |

Table 1. Performance comparison of PointRCNN with and without Pointformer on KITTI test split by submitting to official test server. The evaluation metric is Average Precision(AP) with IoU threshold 0.7 for car and 0.5 for pedestrian/cyclist.

| Method | Modality | Car | Ped | Bus | Barrier | TC | Truck | Trailer | Moto | Cons. Veh. | Bicycle | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CBGS [42] | LiDAR | 81.1 | 80.1 | 54.9 | 65.7 | 70.9 | **48.5** | 42.9 | 51.5 | **10.5** | 22.3 | 52.8 |
| + Pointformer | LiDAR | **82.3** | **81.8** | **55.6** | **66.0** | **72.2** | 48.1 | **43.4** | **55.0** | 8.6 | **22.7** | **53.6** |

Table 2. Performance comparison of PointRCNN with and without Pointformer on the nuScenes benchmark.

| Method | Car(IoU=0.7) | | |
|---|---|---|---|
| | Easy | Moderate | Hard |
| PointRCNN | 88.88 | 78.63 | 77.38 |
| + Pointformer | **90.05** | **79.65** | **78.89** |

Table 3. Performance comparison of PointRCNN with and without Pointformer on the car class of KITTI val split set.

| RoIs | Recall(IoU=0.5) | | Recall(IoU=0.7) | |
|---|---|---|---|---|
| | PointRCNN | +Pointformer | PointRCNN | +Pointformer |
| 10 | 86.66 | **87.51** | 29.87 | **35.46** |
| 50 | 96.01 | **96.52** | 40.28 | **42.45** |
| 100 | 96.79 | **96.91** | 74.81 | **75.82** |
| 200 | **98.03** | 97.99 | 76.29 | **76.51** |

Table 4. Recall of proposal generation network with different number of RoIs and 3D IoU thresholds for the car class on the val split at moderate difficulty.

where $E_i, F_i \in \mathbb{R}^{k\times n}$ are the projection matrices, which reduces the complexity from $O(n^2)$ to $O(kn)$.

Compared with the Taylor expansion approximation technique used in MLCVNet [35], Linformer is easier to implement in out method. We thus adopt it to replace the Transformer layers in the vanilla Pointformer. Practically, we map the number of points $n$ to $k = \frac{n}{r}$, where $r$ is a factor controlling the number of projected dimensions. We apply this mapping in Local Transformer, Global Transformer and Local-Global Transformer blocks. By setting an appropriate factor $r$ for each block, there would be a significant boost in both time and space consumption with little performance decay.

# 4. Experimental Results

In this section, we use *Pointformer* as the backbone for state-of-the-art object detection models and conduct experiments on several indoor and outdoor benchmarks. In Sec. 4.1, we introduce the implementation details of the experiments. In Sec. 4.2 and Sec. 4.3, we show the comparison results on indoor and outdoor datasets respectively. In Sec. 4.4, we conduct extensive ablation studies to analyze our proposed Pointformer model. Finally, we show qualitative results in Sec. 4.5. More analysis and visualizations are provided in the appendix.

## 4.1. Experimental Setup

**Datasets.** We adopt SUN RGB-D [27] and ScanNet V2 [7] for indoor 3D detection benchmark. SUN RGB-D has 5K training images annotated with oriented 3D bounding boxes for 37 object categories and ScanNet V2 has 1513 labeled scenes with 40 semantic classes. We follow the same setting in VoteNet [19] and report performance on the 10 classes on SUN RGB-D and 18 classes on ScanNet V2. For outdoor datasets, we choose KITTI [10] and nuScenes [1] for evaluation. KITTI contains 7,481 training samples and 7,518 test samples for autonomous driving. NuScenes contains 1k different scenes with 40K key frames, which has 23 categories and 8 attributes. We follow the evaluation protocol proposed along with the datasets.

**Experimental setups.** We use the Pointformer as the backbone for three 3D detection models, including VoteNet [19], PointRCNN [25] and CBGS [42]. VoteNet is a point-based approach for indoor datasets, while PointRCNN and CBGS are adopted for outdoor datasets. PointRCNN is a classic approach for autonomous driving detection and CBGS is the champion of nuScenes 3D detection Challenge held in CVPR 2019. For a fair comparison, we adopt the same detection head, number of points for each resolution, hyperparameters and training configurations as baseline models.

## 4.2. Outdoor Datasets

**KITTI.** We first evaluate our method comparing with PointRCNN on KITTI's 3D detection benchmark. PointRCNN uses PointNet++ as its backbone with four set abstraction layers. Similarly, we adopt the same architecture, while switching the set abstraction layer in PointNet++ with the proposed Transformer block. The comparison results on the KITTI test server are shown in Table 1.

For the car category, we also report the performance of 3D detection results on the val split as shown in Table 3. As we can observe, by adopting Pointformer, our model achieves consistent improvements comparing to the original PointRCNN. Especially in the hard difficulty, our method shows the most promising result with 1.5% AP improvement. We believe the better performance on hard objects is

| Method | bathtub | bed | bookshelf | chair | desk | dresser | nightstand | sofa | table | toilet | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| VoteNet [19] | 74.4 | 83.0 | 28.8 | 75.3 | 22.0 | 29.8 | 62.2 | 64.0 | 47.3 | 90.1 | 57.7 |
| VoteNet* | 75.5 | **85.6** | **32.0** | **77.4** | 24.8 | 27.9 | 58.6 | **67.4** | 51.1 | 90.5 | 59.1 |
| + Pointformer | **80.1** | 84.3 | **32.0** | 76.2 | **27.0** | **37.4** | **64.0** | 64.9 | **51.5** | **92.2** | **61.1** |

Table 5. Perfomance comparison of VoteNet with and without Pointformer on **SUN RGB-D** validation dataset. The evaluation metric is Average Precision with **0.25 IoU threshold**.* denotes the model implemented in MMDetection3D [5].

| Method | cab | bed | chair | sofa | table | door | wind | bkshf | pic | cntr | desk | curt | fridg | showr | toil | sink | bath | ofurn | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VoteNet [19] | 36.3 | 87.9 | 88.7 | **89.6** | 58.8 | 47.3 | 38.1 | 44.6 | 7.8 | 56.1 | 71.7 | 47.2 | 45.4 | 57.1 | 94.9 | 54.7 | 92.1 | 37.2 | 58.6 |
| VoteNet* | **47.7** | **88.7** | 89.5 | 89.3 | 62.1 | 54.1 | 40.8 | 54.3 | 12.0 | **63.9** | **69.4** | 52.0 | **52.5** | **73.3** | 95.9 | 52.0 | **95.1** | 42.4 | 62.9 |
| + Pointformer | 46.7 | 88.4 | **90.5** | 88.7 | **65.7** | **55.0** | **47.7** | **55.8** | **18.0** | 63.8 | 69.1 | **55.4** | 48.5 | 66.2 | **98.9** | **61.5** | 86.7 | **47.4** | **64.1** |

Table 6. Performance comparison of VoteNet with and without Pointformer on **ScanNetV2** validation dataset. The evaluation metric is Average Precision with **0.25 IoU threshold**.* denotes the model implemented in MMDetection3D [5].

attributed to the higher expressiveness of local Transformer module. For hard objects which are often small or occluded, GT captures context-dependent region features, which contributes to the bounding box regression and classification.

Additionally, we evaluate the performance of proposal generation network by calculating the recall of 3D bounding box with various number of proposals and 3D IoU threshold. As shown in Table 4, our backbone module significantly enhances the performance of proposal generation network under almost all the settings. Analyzing the figures vertically, we observe that our backbone shows better performance when the number of RoIs are relatively small. As stated in Sec.3, the GT and LGT help to capture context-aware representations and models the relations among different objects (proposals). This provides additional references for locating and reasoning the bounding boxes. Therefore, despite the lack of RoIs, we can still improve the performance of the proposal generation module and achieve higher recall.

**NuScenes.** We also validate the effectiveness of Pointformer on the nuScenes dataset, which greatly extends KITTI in dataset size, number of object categories and number of annotated objects. Furthermore, nuScenes suffers from severe class imbalance issues, making the detection task more difficult and challenging. In this part, we adopt CBGS, the champion of nuScenes 3D detection Challenge held in CVPR 2019, as the baseline model and show the comparison results when replacing the backbone with Pointformer. We summarize the results in Table 2. As we can observe, by utilizing Pointformer as the backbone, our model achieves 0.8 higher mAP than baseline. For 8 of 10 classes, our model shows better performance, which demonstrates the effectiveness of Pointformer on larger and more challenging datasets.

### 4.3. Indoor Datasets

We evaluate our Pointformer accompanied by VoteNet [19] on SUN RGB-D and ScanNet V2. We follow the same hyperparameters on the backbone structure as VoteNet. Followed by the Pointformer blocks, two feature propagation(FP) modules proposed in PointNet++ [22] serve as upsamplers to increase the resolution for the subsequent detection heads.

| | LT | GT | LGT | CoRe | Car (IoU=0.7) Easy | Moderate | Hard |
|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | 88.88 | 78.63 | 77.38 |
| 2 | ✓ | - | - | - | 89.46 | 78.91 | 77.65 |
| 3 | ✓ | - | - | ✓ | 89.76 | 79.24 | 78.43 |
| 4 | ✓ | ✓ | - | - | 89.68 | 79.22 | 78.52 |
| 5 | ✓ | ✓ | ✓ | - | 89.82 | 79.34 | 78.62 |
| 6 | ✓ | ✓ | ✓ | ✓ | **90.05** | **79.65** | **78.89** |

Table 7. Effects of each component on the val split of KITTI. CoRe represents the coordinates refinement module.

| | Positional Encoding | Car (IoU=0.7) Easy | Moderate | Hard |
|---|---|---|---|---|
| 1 | - | 85.42 | 75.67 | 72.34 |
| 2 | ✓ | **90.05** | **79.65** | **78.89** |

Table 8. Effects of positional encoding on the val split of KITTI.

**SUN RGB-D.** We report the average precision(AP) over 10 common classes in SUN RGB-D, as shown in Table 5. Compared with the PointNet++ [22] in VoteNet [19], our Pointformer provides a significant boost with 2% mAP over the implementation in MMDetection3D [5]. On some categories with large and complex objects like dresser or bathtub, Pointformer shows its splendid capability on extracting non-local information by a sharp increase over 5% AP, which we attribute to the GT module in Pointformer.

**ScanNet V2.** We report the average precision(AP) over 18 classes in ScanNet V2, as shown in Table 6. Compared with VoteNet, Pointformer outperforms its original version by 1.2% mAP with MMDetection3D.

### 4.4. Ablation Study

In this section, we conduct extensive ablation experiments to analyze the effectiveness of different components of Pointformer. All experiments are trained on the train split

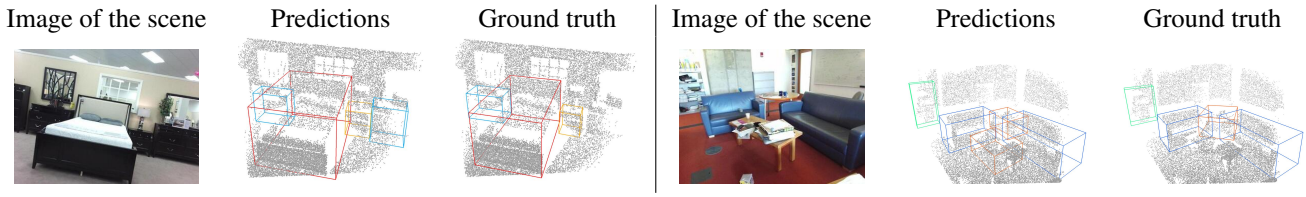


Figure 4. **Qualitative results of 3D object detection on SUN RGB-D.** From left to right: Originial scene image, our model's prediction, and annotated ground truth boxes.

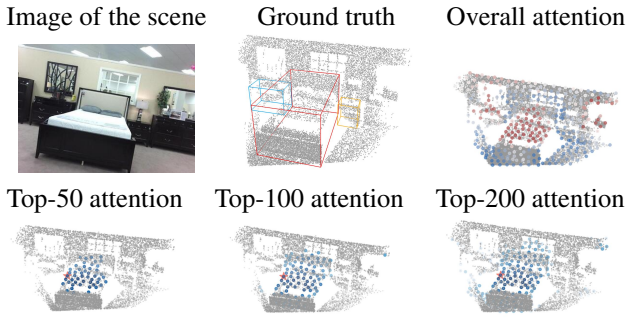


Figure 5. **Visualization results of the attention maps.** In top-k attention, darker color indicates larger attention weight, in overall attention red indicates large value.

with PointRCNN detection head and evaluated on the val split with the car class.

**Effects of each component.** We validate the effectiveness of each Transformer component and the coordinate refinement module, and summarized the results in Table 7. The first row corresponds to the PointRCNN baseline and the last row is the full Pointformer model. By comparing the first row and second row, we can observe that easy objects benefit more from the local Transformer with 0.6 AP improvement. By comparing the second row and fourth row, we can see that global Transformer is more suitable for hard objects with 0.9 AP improvement. This observation is consistent with our analysis in Sec. 4.2. As for Local-Global Transformer and coordinate refinement, the improvement is similar under three difficulty settings.

**Positional Encoding.** Playing a critical role in Transformer, position encoding can have huge impact on the learned representation. As we have shown in Table 8, we compare the performance of Pointformer without positional encoding and with two approaches to position encoding (adding or concatenating positional encoding with the attention map). We can observe that Pointformer without positional encoding suffers from a huge performance drop, as the coordinates of points can capture the local geometric information.

### 4.5. Qualitative Results and Discussion

**Qualitative results on SUN RGB-D.** Figure 4 shows representative examples of detection results on SUN RGB-D with VoteNet + Pointformer. As we can observe, our model achieves robust results despite the challenges of clutter and scanning artifacts. Additionally, our model can even recognize the missing objects in the ground truth. For instance, the dresser in the left scene is only partially observed by the sensor. However, our model can still generate precise proposals for the object with proper bounding box sizes. Similar results are shown in the right scene, where the table in the front suffers from clutter because of the books on it.

**Inspecting Pointformer with attention maps.** To validate how modules in Pointformer affect learned point features, we visualize the attention maps from the GT module of the second last Pointformer block. We show the attention of the particular points in Figure 5. The second row shows the 50, 100, 200 points with highest attention values towards the points marked with star. We can observe that Pointformer first focuses on the local region of the same object, then spread the attention to other regions, and finally attends points from other objects globally. The overall attention map shows the average attention weights of all the points in the scene, indicating that our model mostly focuses on points on the objects. These visualization results show that Pointformer can capture local and global dependencies, and enhance message passing on both object and scene levels.

## 5. Conclusion

This paper introduces *Pointformer*, a highly effective feature learning backbone for 3D point clouds that is permutation invariant to points in the input and learns local and global context-aware representations. We apply Pointformer as the drop-in replacement backbone for state-of-the-art 3D object detectors and show significant performance improvements on several benchmarks including both indoor and outdoor datasets.

Comparing to classification and segmentation tasks including part-segmentation and semantic segmentation in prior work, 3D object detection typically involves more points (4× - 16×) in a scene, which makes it harder for Transformer-based models. For future work, we would like to explore extensions to these two tasks and other 3D tasks such as shape completion, normal estimation, etc.

## Acknowledgments

This work is supported in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grants 2018AAA0100701, the National Natural Science Foundation of China under Grants 61906106 and 62022048, the Institute for Guo Qiang of Tsinghua University and Beijing Academy of Artificial Intelligence.

## References


[1] H. Caesar, Varun Bankiti, A. Lang, Sourabh Vora, Venice Erin Liong, Q. Xu, A. Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *CVPR*, pages 11618–11628, 2020.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End object detection with transformers. In *ECCV*, May 2020.

[3] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. *ICML*, 2020.

[4] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

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

[6] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[11] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, June 2018.

[12] Ji Hou, Angela Dai, and Matthias Niessner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *CVPR*, June 2019.

[13] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Franccois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *ArXiv*, abs/2006.16236, 2020.

[14] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *ICLR*, 2020.

[15] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[16] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[17] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, June 2019.

[18] Youngmin Park, Vincent Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 117–120, 2008.

[19] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *CVPR*, pages 9277–9286, 2019.

[20] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018.

[21] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017.

[23] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, I. Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[24] Shaoshuai Shi, Chaoxu Guo, L. Jiang, Zhe Wang, Jianping Shi, X. Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. *CVPR*, pages 10526–10535, 2020.

[25] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, June 2019.

[26] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020.

[27] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576, 2015.

[28] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *CVPR*, June 2016.

[29] H. Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *ICCV*, pages 6410–6419, 2019.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[31] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *CVPR*, June 2020.

[32] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[33] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, June 2018.

[34] 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.

[35] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. Mlcvnet: Multi-level context votenet for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10447–10456, 2020.

[36] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and U. Neumann. Grid-gcn for fast and scalable point cloud learning. *CVPR*, pages 5660–5669, 2020.

[37] Xu Yan, C. Zheng, Zhuguo Li, S. Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. *CVPR*, pages 5588–5597, 2020.

[38] 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 *CVPR*, June 2019.

[39] M. Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. *CVPR*, pages 1628–1637, 2020.

[40] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *European Conference on Computer Vision*, pages 311–329. Springer, 2020.

[41] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, June 2018.

[42] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection. *arXiv e-prints*, page arXiv:1908.09492, Aug 2019.

[43] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2020.