BoxeR: Box-Attention for 2D and 3D Transformers

Duy-Kien Nguyen[1] Jihong Ju[2] Olaf Booij[2] Martin R. Oswald[1] Cees G. M. Snoek[1]
Atlas Lab - [1]University of Amsterdam [2]TomTom
{d.k.nguyen, m.r.oswald, cgmsnoek}@uva.nl {jihong.ju, olaf.booij}@tomtom.com

Abstract

In this paper, we propose a simple attention mechanism, we call Box-Attention. It enables spatial interaction between grid features, as sampled from boxes of interest, and improves the learning capability of transformers for several vision tasks. Specifically, we present BoxeR, short for Box Transformer, which attends to a set of boxes by predicting their transformation from a reference window on an input feature map. The BoxeR computes attention weights on these boxes by considering its grid structure. Notably, BoxeR-2D naturally reasons about box information within its attention module, making it suitable for end-to-end instance detection and segmentation tasks. By learning invariance to rotation in the box-attention module, BoxeR-3D is capable of generating discriminative information from a bird's-eye view plane for 3D end-to-end object detection. Our experiments demonstrate that the proposed BoxeR-2D achieves state-of-the-art results on COCO detection and instance segmentation. Besides, BoxeR-3D improves over the end-to-end 3D object detection baseline and already obtains a compelling performance for the vehicle category of Waymo Open, without any class-specific optimization. Code is available at https://github.com/kienduynguyen/BoxeR.

1. Introduction

For object detection, instance segmentation, image classification and many other current computer vision challenges, it may seem a transformer with multi-head self-attention is all one needs [40]. After its success in natural language processing, learning long range feature dependencies has proven an effective tactic in computer vision too, *e.g.*, [1, 7]. Surprisingly, existing transformers for computer vision do not explicitly consider the inherent regularities of the vision modality. Importantly, the image features are vectorized in exactly the same way as language tokens, resulting in the loss of local connectivity among pixels. Once fed with sufficient data, a traditional transformer may be powerful enough to compensate for this loss of spatial structure, but in this paper we rather prefer to equip the transformer with spatial

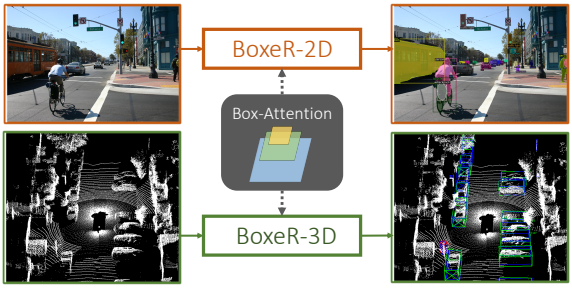

Figure 1. **BoxeR with box-attention** for object detection and instance segmentation. BoxeR-2D perceives an image and generates object bounding boxes and pixel masks. Extended from BoxeR-2D, BoxeR-3D predicts 3D bounding boxes from point cloud input.

image-awareness by design. Recent evidence [5, 39, 47] reveals that an inductive bias is of crucial importance in both natural language processing and computer vision, and the leading works on image recognition [24] and object detection [47] all utilize "spatial information". Furthermore, a strong and effective inductive bias enables us to converge faster and generalize better [39].

A solution is to enrich image features with positional encoding, which explicitly encodes the position information at the feature level. This is already common practice when applying multi-head attention to vision tasks. Both Carion *et al*. [1] and Zhu *et al*. [47] convert absolute 2D positions, while Ramachandran *et al*. [29] encode relative 2D positions into vectors and sum them up to image features in the attention computation. However, this approach only acts as a data augmentation to image features. It requires the network to infer the spatial information implicitly inside its weight, causing a slow convergence rate during training due to the lack of spatial-awareness in the network architecture. It is well known that an inductive bias in the network architecture delivers a strong ability to learn, which has been proven by well-known architectures such as the convolutional neural network [19] and the long short-term memory [13]. In particular, we postulate a better spatial inductive bias in the transformer's attention module leads to a better learned representation of image features.

Motivated by this observation, the first contribution of this paper is a *Box-Attention* mechanism for end-to-end vision

representation learning using transformers that we present in Section 3. Instead of using image features within a region of interest, it treats a set of learnable embeddings representing relative positions in the grid structure as the key vectors in the attention computation. In our second contribution, in Section 4, these computations are encapsulated into a composite network that we call BoxeR-2D, short for Box transformeR, which enables a better prediction in end-to-end object detection and instance segmentation tasks. In Section 5, the BoxeR-2D and box-attention are then extended into BoxeR-3D to tackle end-to-end 3D object detection without the requirements for 3D-IoU computation, anchors, and a heatmap of object centers. This extension to 3D object detection serves as our third contribution, see Fig. 1.

In Section 6, we show the effectiveness of our contributions by several experimental results on the COCO dataset [21], achieving leading results in end-to-end object detection. The proposed method introduces a simple solution for end-to-end instance segmentation that outperforms many well-established and highly-optimized architectures with fewer number of parameters on the challenging COCO instance segmentation dataset. By utilizing only data-independent prior information, our method presents a compelling solution for end-to-end 3D object detection on the Waymo Open dataset [35].

2. Related Work

We briefly review recent developments in computer vision with focus on attention mechanisms for backbones, object detection, instance segmentation and 3D object detection.

Attention for Vision Backbones. With the advancement of attention mechanisms, there are several approaches to create and use attention in convolutional networks, *e.g.*, [14, 29, 34, 42]. It was recently shown in the Vision Transformer (ViT) [7] that an attention-only network achieves comparable performance in image recognition, and outperforms convolutional neural networks in the setting of more data and longer training time. As the ViT becomes computationally more expensive with high resolution images, while only producing a single-scale feature map, several works [8, 24] have focused on speeding up the self-attention computation and generating multi-scale feature maps for object detection and segmentation. In this paper, we instead focus on the prediction module which takes features extracted from vision backbones as inputs and provides a prediction for several vision tasks.

Attention for Object Detection. Modern two-stage object detection methods [23] (*i.e.*, Faster R-CNN [31]) utilize a region proposal network (RPN) and a prediction module on top of a pretrained backbone to predict a set of predefined objects. The attention mechanism is then considered as an addition of the RPN and prediction modules to further improve performance in [3, 37]. Alternatively, one-stage object detection methods [30, 41] remove the need for RPN and predict objects directly from convolutional feature maps. While the detection performance improves considerably, these convolution-based architectures still rely on many hand-crafted components. Recently, Carion *et al*. introduced a transformer-based prediction model, called DETR [1], which gave the prediction in an *end-to-end* manner. Pointing out the slow convergence and high computational cost of self-attention on image features, Zhu *et al*. [47] introduced multi-head deformable attention, replacing the dot-product in the attention computation with two linear projections for sampling points and computing their attention weights. While improving in both the convergence rate and accuracy, the strategy of sampling positions around a reference point prevents it to efficiently capture object information like object size and location. As sampled points on the image feature maps are separated, the module is unaware of the local connectivity of the attended region. Our BoxeR closely follows the overall framework of end-to-end object detection by Carion *et al*. [1], but differs at its core by the use of the spatial prior and the multi-head box-attention mechanism. Our multi-head box-attention is inspired by the standard multi-head attention and convolution operation, which have both been shown to learn robust image representation. The box-attention considers a box region by only predicting its center and size, which is more efficient and allows us to extract structured information within the predicted region.

Attention for Instance Segmentation. A method for tackling instance segmentation is required to locate objects and segment the pixels belonging to the object at the same time. Inspired by modern object detectors, earlier studies [2, 27] predict segment proposals in a first stage; the segment proposals are then classified in a second stage. He *et al*. [11] proposed to train object detection and instance segmentation simultaneously in a multitask setting to boost the performance of both tasks. Different from modern segmentation models, which predict bounding boxes and masks from the same set of features (*i.e.*, ResNet features), DETR relies on transformer features for object detection and ResNet features augmented with attention maps from the transformer for segmentation. This causes a mismatch in information level since these two tasks are highly related. Dong *et al*. [6] suggested to learn unified queries for both object detection and instance segmentation by taking advantage of deformable attention. However, this approach still lags behind convolution-based architectures by a large margin. We introduce box-attention which naturally extends to both object detection and instance segmentation in a single BoxeR-2D architecture achieving state-of-the-art performance on both tasks.

Attention for 3D Object Detection. The main challenge in 3D object detection is to deal with rotated bounding boxes from bird's-eye view image features. Many meth-

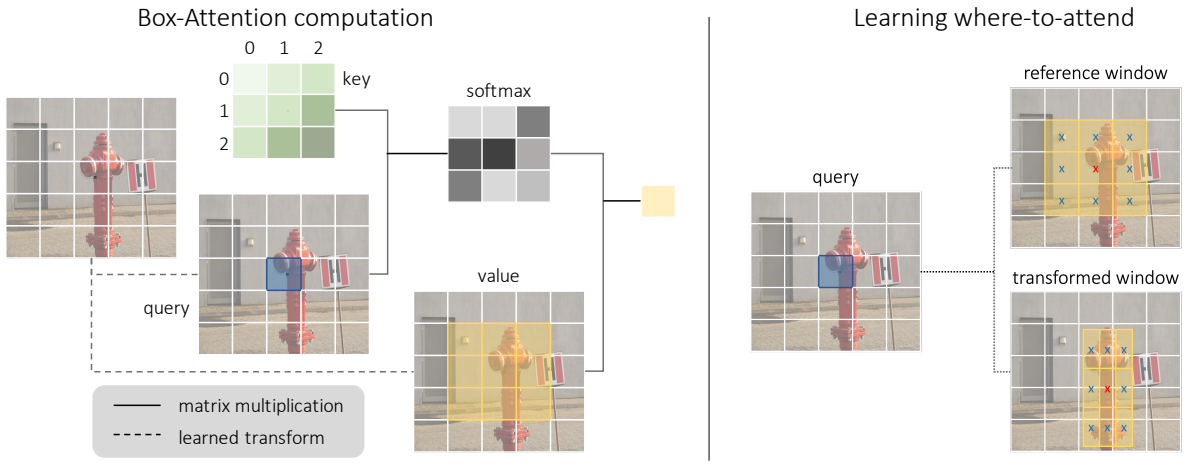

Figure 2. **Box-Attention.** Left: attention computation in Box-Attention with a reference window (denoted in yellow) without any transformation. Given a query vector, the Box-Attention computes an attention map over 3×3 grid features with the query position as its center. The attention weights are generated by a matrix multiplication between query and relative position encodings as key. Right: Box-Attention without and with transformations. The Box-Attention with transformations is able to focus on a dynamic region in the image.

ods [20,33,35] adapted Faster R-CNN by generating anchors of different angles as object proposals, followed by classification and regression. As anchor-based methods generate a high number of overlapping proposals, which requires non-maximum suppression to be tuned for each of the classes, approaches in [10,44] focused on predicting a heat-map of object centers in a scene. While the number of overlapping proposals is reduced, predicting a heat-map leads to the loss of prior information compared to anchors and still relies on non-maximum suppression to filter out object proposals. A transformer with self-attention was also adopted for 3D object detection in [25,32]. Unfortunately, they exhibit the same problems as traditional detectors since their methods require initial object predictions from previous methods. The recent work of Misra *et al*. [26] introduced 3DETR for indoor 3D object detection. This method utilizes self-attention in both the encoder and decoder with object queries generated by a Farthest Point Sampling algorithm on point clouds [28]. Instead, BoxeR presents a solution for end-to-end 3D object detection on outdoor scenes that simply uses bird's-eye view features to predict objects without non-maximum suppression, 3D rotated IoU, or a complicated initialization method.

3. Box-Attention

Box-attention is a multi-head attention mechanism designed to attend to boxes of interest in an image feature map. To do so, it samples a grid within each box and computes attention weights on sampled features from the grid structure, making the module easy to generalize to 2D or 3D object detection as well as instance segmentation. In each head of the attention computation, a box of interest is generated by predicting geometric transformations (*i.e*., translation, scaling, and rotation) from a pre-defined reference window. The box-attention design allows the network to attend to dynamic regions of image features with reasonable computational cost.

Multi-Head Self-Attention. We start by briefly summarizing the standard multi-head self-attention in the Transformer [40]. The multi-head self-attention of l attention heads generates output features of the queries (Q) by calculating weighted average vectors of the value features (V) corresponding to the key vectors (K):

$$\text{MultiHead}(Q,K,V) = \text{Concat}(h_1,\dots,h_l)\,W^O \quad , \quad (1)$$

where $h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. The self-attention module computes an attention map in each head using the dot-scale product of features between Q and K, in which the computation increases quadratically with the matrix size.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \quad , \quad (2)$$

where d_k is the dimension of the key feature in one head.

Multi-Head Box-Attention. Box-attention adopts the multi-head attention computation in Eq. (1) with the same feature aggregation of multiple heads and a learnable projection matrix W^O. In the attention computation stage, given a box of interest b_i of query vector $q \in \mathbb{R}^d$ in the i^{th} attention head, box-attention extracts a grid feature map v_i of size $m\times m$ from b_i using bilinear interpolation as illustrated in Fig. 2. The use of bilinear interpolation to compute the exact values of the grid features reduces the quantization error of the box-attention in box regression and pixel segmentation. This differs from deformable attention [47], which predicts unstructured points causing ambiguity in capturing object information. Instead, our attention mechanism inherits the spirit of RoIAlign [11] that precisely samples a grid structure within a region of interest (*i.e*., bounding box proposals)

to obtain accurate pixel-level information which has been found to be important for pixel-accurate masks.

During the i^{th} head attention computation, we treat the grid feature map $v_i \in \mathbb{R}^{m\times m\times d_h}$ as a set of value features corresponding to the query $q \in \mathbb{R}^d$. The $m\times m$ attention scores are then generated by computing the dot-product between q and $m\times m$ learnable key vectors K_i, where each vector represents a relative position in the grid structure, followed by a softmax function. Thus, we share the same set of keys across queries. By treating K_i as relative location embedding of the sampled grid, box-attention can efficiently capture spatial information regarding the region. In the implementation, the attention map generation can be performed efficiently via a simple linear projection ($\mathbb{R}^d \rightarrow \mathbb{R}^{m\times m}$) that is equivalent to the dot-product with learnable key vectors. The final $h_i \in \mathbb{R}^{d_h}$ is the weighted average of the $m\times m$ vectors in v_i with attention weights.

$$\begin{aligned} h_i &= \text{BoxAttention}(Q, K_i, V_i) \\ &= \sum_{m\times m} \text{softmax}\big(QK_i^\top\big) * V_i \ , \end{aligned} \tag{3}$$

where $Q \in \mathbb{R}^{N\times d}$, $K_i \in \mathbb{R}^{(m\times m)\times d}$, $V_i \in \mathbb{R}^{N\times(m\times m)\times d_h}$, and d_h is the dimension of features in one head.

It has been shown in [22] that multi-scale feature maps lead to large improvements in both object detection and instance segmentation. Our box-attention can be simply extended to work on multi-scale features. Given a set of boxes $\{b_i^1, \ldots, b_i^t\}$ of the query vector q in an attention head, each of which belongs to each of t multi-scale feature maps, we sample a grid of features from each box, resulting in $v_i \in \mathbb{R}^{(t\times m\times m)\times d_h}$. The $t\times m\times m$ attention scores are computed in the same way with $t\times m\times m$ learnable key vectors $K_i \in \mathbb{R}^{(t\times m\times m)\times d}$, where each vector represents a relative position in t grid structures, followed by a softmax normalization. The $h_{(1,\ldots,l)} \in \mathbb{R}^{d_h}$ feature now is the weighted average of $t\times m\times m$ vectors in $v_{(1,\ldots,l)}$ as in Eq. (3).

Multi-Head Instance-Attention. Instance-attention is a simple extension of box-attention without any extra parameters. Our goal is to generate an accurate mask from the box of interest for instance segmentation. In the i^{th} attention head, it generates two outputs, $h_i \in \mathbb{R}^{d_h}$ for object detection and $h_i^{\text{mask}} \in \mathbb{R}^{m\times m\times d_h}$ for instance segmentation. While weighted-averaging the $t\times m\times m$ features in v_i to create h_i, we collapse v_i in the first dimension (which contains the number of multi-scale features) for h_i^{mask}. To do this, we normalize the first dimension of the $t\times m\times m$ attention scores using the softmax function which are then applied to v_i. Note that we share all parameters of the attention module in generating $h_{(1,\ldots,l)}$ and $h_{(1,\ldots,l)}^{\text{mask}}$, including the learnable projection matrix W^O.

Where-to-attend. Where-to-attend is a key component of our box-attention, it refers to an operation for predicting a box of interest in the attention computation. Specifically, the module learns to transform a reference window of query q on a feature map into an attended region via simple geometric transformations, such as translation and scaling (see Fig. 2).

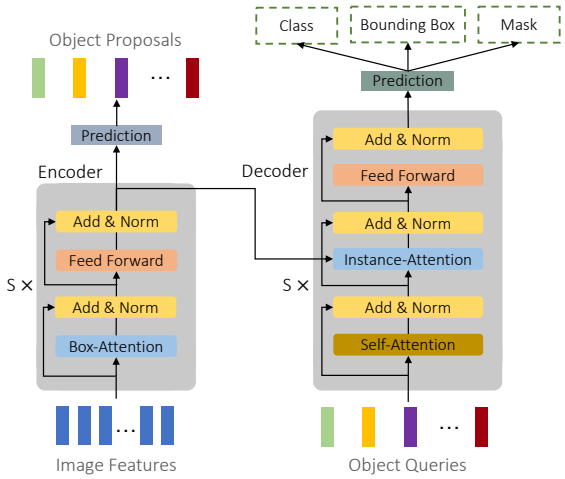

Figure 3. **BoxeR structure.** The BoxeR-2D takes encoder features corresponding to object proposals as its object queries. The object queries are then decoded into bounding boxes and pixel masks using instance-attention.

To be specific, we denote the reference window of query q by $b_q{=}[x, y, w_x, w_y] \in [0, 1]^4$ where x, y indicate its center position, w_x, w_y are width and height of the window in normalized coordinates. The translation function, $\mathcal{F}_t$, takes q and b_q as its inputs and performs translation, which outputs b'_q as follows:

$$\mathcal{F}_t(b_q, q) = b'_q = [x + \Delta_x, y + \Delta_y, w_x, w_y] \ , \tag{4}$$

where Δ_x and Δ_y are offsets relative to the center of the reference window. Similarly, the scaling function, $\mathcal{F}_s$, takes the same inputs and adjusts the size of b_q

$$\mathcal{F}_s(b_q, q) = b'_q = [x, y, w_x + \Delta_{w_x}, w_y + \Delta_{w_y}] \ , \tag{5}$$

where Δ_{w_x} and Δ_{w_y} are offsets for the reference window size. The offset parameters (*i.e.*, $\Delta_x, \Delta_y, \Delta_{w_x}, \Delta_{w_y}$) are predicted using a linear projection on q for efficiency. In the multi-head attention setting of l heads and t multi-scale feature maps, we use $l\times t$ transformation functions where each function predicts a box of interest b_i^j for i^{th} head and j^{th} feature map.

Where-to-attend is a combination of transformations and allows our box-attention to effectively attend to necessary regions with a small number of parameters and low computational overhead. It can also be seen as a pseudo prediction step since it provides the network spatial information to predict a region of interest within the attention module.

4. BoxeR-2D: A Box Transformer

To demonstrate the effectiveness of our approach, we present BoxeR, a Transformer-based network with box-

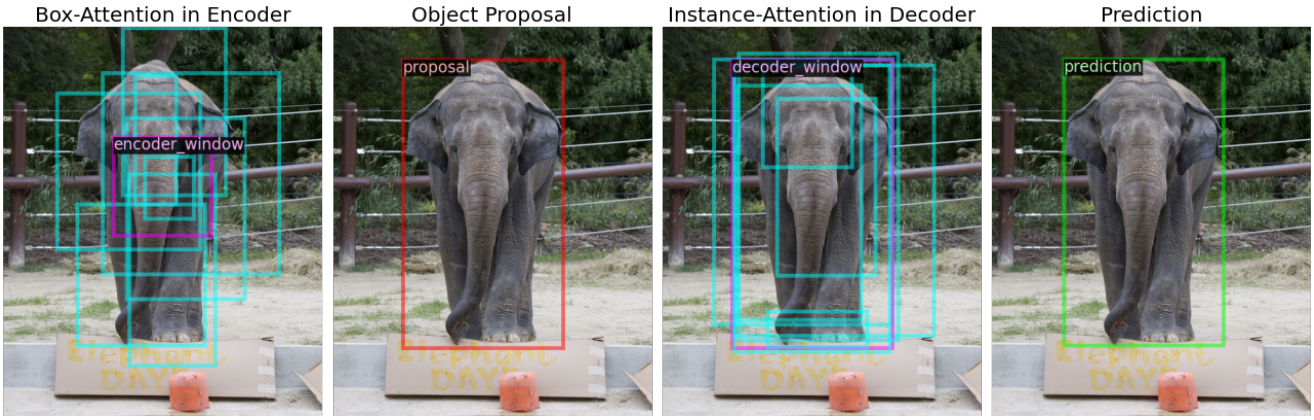

Figure 4. **BoxeR-2D behavior.** We show the behavior of BoxeR-2D by tracing back its prediction. Box-Attention in the encoder is able to capture regions of multiple aspect ratios from the reference window (denoted in purple), while Instance-Attention in the decoder plays a role to refine the object proposal. The BoxeR-2D predicts an object proposal which highly overlaps with the final prediction.

attention in its architecture; see Fig. 3. BoxeR consists of an encoder for encoding multi-scale feature maps extracted from a backbone and a decoder for predicting instance bounding boxes and pixel masks. Our approach follows the spirit of end-to-end object detection frameworks (*i.e.*, DETR [1]), that reduce the need for hand-crafted modules, like non-maximum suppression and anchor-groundtruth matching.

BoxeR Encoder. As in Transformer, each BoxeR encoder layer contains box-attention and feed forward sub-layers, each of which is followed by a LayerNorm [15] with residual connection. Following [47], the encoder takes multi-scale image feature maps $\{x^j\}_{j=1}^{t-1}(t{=}4)$ extracted from C_3 through C_5 of a ResNet backbone [12] (transformed by a $1{\times}1$ convolution to the hidden dimension) as its inputs. The t^{th} feature map x^t is obtained by applying a $3{\times}3$ convolution layer with stride 2 on the final C_5 feature. The BoxeR encoder will transform multi-scale inputs into multi-scale contextual representations $\{e^j\}_{j=1}^{t}$. Note that the multi-scale contextual representations $\{e^j\}_{j=1}^{t}$ are in the same resolution as the inputs $\{x^j\}_{j=1}^{t}$.

In the encoder, both Q and V are features from multi-scale feature maps. We assign a reference window to each query vector where the window is centered at the query spatial position. The sizes of the sliding windows are $\{32^2, 64^2, 128^2, 256^2\}$ pixels on multi-scale feature maps $\{x_1, x_2, x_3, x_4\}(t{=}4)$ (or 4^2 features on each of the multi-scale feature maps), as suggested in [22]. Because l parallel attention heads of box-attention are able to implicitly capture boxes of multiple aspect ratios at each feature level, we found that it is not necessary to have reference windows of multiple aspect ratios (see Fig. 4). Beside augmenting each query with a position embedding, we add a size embedding, which represents the size of the reference window corresponding to each query. The size embedding only differs between query vectors of different levels. Both embeddings are normalized and encoded with sinusoid encodings.

Since two-stage networks indicate a significant improvement in object detection [31, 47], we show that the BoxeR encoder is able to produce high-quality object proposals as inputs for the decoder. In the object proposal stage, features from the last encoder layer are fed into a prediction head to predict object proposals w.r.t. their reference windows. Instead of treating the sinusoid embedding of bounding boxes predicted in the object proposal stage as object queries [47], we simply take the encoder features (transformed by a linear projection) with the highest classification scores as input features for the decoder. This provides richer information to the BoxeR decoder as encoder features contain both spatial and contextual information. The predicted bounding boxes are treated as reference windows for its corresponding proposals in the decoder.

BoxeR Decoder. In each BoxeR decoder layer, the cross-attention sub-layer is our multi-head instance-attention, while the self-attention and feed forward sub-layers are left unchanged. The features of the object proposals from the encoder are the inputs of BoxeR decoder. The reference windows of the object proposals are refined in this stage in order to give accurate predictions.

To be specific, we denote the inputs to the $(s+1)^{\text{th}}$ decoder layer by $x_s \in \mathbb{R}^{N\times d}$. The $(s+1)^{\text{th}}$ decoder layer then outputs $x_{s+1} \in \mathbb{R}^{N\times d}$ and $x_{s+1}^{\text{mask}} \in \mathbb{R}^{N\times m\times m\times d}$. The feed forward sub-layer is the same for both outputs. The output features $x_S \in \mathbb{R}^{N\times d}$ are then decoded into box coordinates and class labels as in [47], while $x_S^{\text{mask}} \in \mathbb{R}^{N\times m\times m\times d}$ are used to generate instance masks. We follow the training strategy in Mask R-CNN [11] to predict instance masks with a per-pixel *sigmoid* and a *binary* loss.

Since the where-to-attend module in the attention module predicts regions of interest based on reference windows, we design the detection head to predict a bounding box as a relative offset w.r.t. its reference window size and position. The reference window serves as an initial guess of its object proposal feature in the prediction stage. The auxiliary de-

coding losses for other decoder layers are also effective in our case. All prediction heads in the BoxeR decoder share their parameters. We found that it is not necessary to add a mask cost into the Hungarian matcher [17], which results in a more efficient training. More details are provided in the supplementary document.

5. BoxeR-3D: End-to-end 3D Object Detection

We enable end-to-end 3D object detection by extending our box-attention and BoxeR to work with point cloud input.

Box-Attention for 3D Object Detection. Along with translation and scaling in the where-to-attend module, we add rotation transformation in the bird's-eye view plane to model the angle of objects. We denote the reference window of q by $b_q{=}[x, y, w_x, w_y, \theta] \in [0, 1]^5$ where x, y indicate its center position, w_x, w_y are width and height of the window, and θ is the rotation angle of b_q around its center in normalized coordinates. The final rotation function, $\mathcal{F}_r$, predicts an offset of the window rotation angle. It then applies a rotation matrix on the $m{\times}m$ grid coordinates sampled from b_q

$$\mathcal{F}_r(b_q, q) = b'_q = [x, y, w_x, w_y, \theta + \Delta_\theta] \ , \qquad (6)$$

where Δ_θ is an offset w.r.t. the reference window angle. Again, we use a linear projection on q to predict Δ_θ.

BoxeR for 3D Object Detection. To better capture objects of different angles, we assign reference windows of multiple angles to each query vector of BoxeR encoder features. At each sliding position, based on the 2D object detection setting, we use three reference windows of 4^2 features on each of the multi-scale feature maps with three angles $\{\frac{-2\pi}{3}, 0, \frac{2\pi}{3}\}$. Each attention head will be assigned a reference window of one angle. By doing so, features generated from our box-attention are strong for rotation prediction (see Fig. 5). In the object proposal stage, for each of the encoder features, we predict class scores and bounding boxes for the three proposals w.r.t. their reference windows of three angles. The 3D Hungarian matcher is used during training. More details are provided in the supplementary document.

We note that only *minimal* prior knowledge about specific object classes, such as the typical size of a vehicle is embedded in our system due to the uniform distribution of the reference window. This is different from previous methods [33, 35, 36, 44] which use different anchor sizes, heat-maps, or backbones for each class. Our network also removes the need for hand crafted modules such as rotated non-maximum suppression or 3D IoU computation.

6. Experiments

6.1. Datasets, Tasks and Implementation Details

MS-COCO 2017. For 2D object detection and instance segmentation, we use the MS-COCO 2017 dataset [21] consisting of 118,000 training images and 5,000 validation images. The instance is categorized based on its size: small, medium and large. We report the standard COCO metrics for bounding boxes and masks. We use the `train` split for training and report ablations on the `val` split. We also report results on the `test-dev` set.

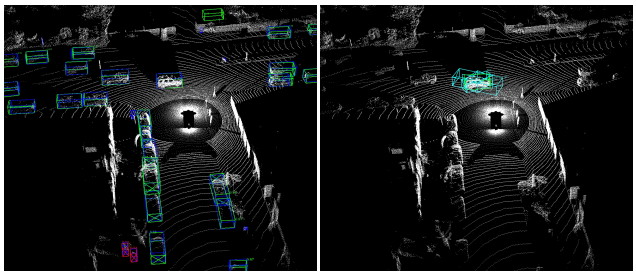

Figure 5. **BoxeR-3D prediction.** Left: BoxeR-3D prediction in an intersection (ground-truth boxes denoted in blue; vehicle and pedestrian predictions denoted in green and red). Right: Visualization of Box-Attention behavior corresponding to one object query. Multiple attention heads of the object query in Box-Attention capture boxes of different angles where the best attended region is well-aligned with BoxeR-3D prediction.

We use the Adam optimizer [16] with α=0.0002, and weight decay set to 0.0001. The learning rate of our backbone and transformation functions in the attention module is multiplied by a factor of 0.1. We find that dropout is not necessary for BoxeR and makes the training slower. Thus, we remove it from the BoxeR architecture. We train our network for 50 epochs with a batch size of 32, the learning rate is decayed at the 40$^{\text{th}}$ epoch by a factor of 0.1. Other hyperparameter settings follow Deformable DETR [47]. During the training procedure, the same data augmentation is used as in [1]. For a better comparison, we also report BoxeR-2D trained with a 3$\times$ schedule as in [43].

Waymo Open. For 3D object detection, we use the Waymo Open dataset [35], which contains 798 training sequences and 202 validation sequences. Each sequence consists of 200 frames where each frame captures the full 360 degrees around a vehicle. We report the official 3D detection evaluation metrics including the standard 3D bounding box mean average precision (mAP) and mAP weighted by heading accuracy (mAPH) in three categories: vehicle, pedestrian, and cyclist.

	FLOPs↓	AP↑	AP_S↑	AP_M↑	AP_L↑
Self-Attention [40]	187G	36.2	16.3	39.2	53.9
Deformable-Attention† [47]	173G	46.9	29.6	50.1	61.6
Dynamic-Attention [4]	-	47.2	28.6	49.3	59.1
Box-Attention (Ours)	167G	**48.7**	**31.6**	**52.3**	**63.2**
w/o ($\mathcal{F}_t$ and $\mathcal{F}_s$)	164G	46.4	29.6	49.8	59.7

† Based on author-provided github, which is higher than in their original paper.

Table 1. **Box-Attention vs. alternatives** in end-to-end object detection on the COCO `val` set using a R-50 backbone pretrained on ImageNet. Box-Attention performs best with the least FLOPs.

	$AP\uparrow$	$AP_S\uparrow$	$AP_M\uparrow$	$AP_L\uparrow$
Box-Attention	**48.7**	**31.6**	**52.3**	**63.2**
w/ *proposal refinement*	47.2	30.4	50.7	62.2

(a) **Object Proposals.**

	$AP\uparrow$	$AP_S\uparrow$	$AP_M\uparrow$	$AP_L\uparrow$	$AP^m\uparrow$	$AP^m_S\uparrow$	$AP^m_M\uparrow$	$AP^m_L\uparrow$
Box-Attention *only*	48.7	31.6	52.3	63.2	-	-	-	-
w/ Instance-Attention	**50.0**	**32.4**	**53.3**	**64.5**	**42.7**	**22.7**	**45.9**	**61.5**

(b) **Instance-Attention.**

Table 2. **BoxeR-2D ablation** on the COCO `val` set using a R-50 backbone pretrained on ImageNet. (a) Our reference windows improve the quality of the object proposals and removes the need for the refinement stage of [47]. (b) BoxeR-2D shows strong results when training on both 2D object detection and instance segmentation simultaneously.

We use the Adam optimizer with weight decay set to 0.0001. Following previous works [36], we use cosine learning rate decay with the initial learning rate set to 5e-4, 5000 warm-up steps starting at 5e-7, and 140K iterations in total. The learning rate of the transformation functions in the attention module is multiplied by a factor of 0.1. We train our network on BEV image features extracted from PointPillar [18] with a grid size of $(0.32\text{m}, 0.32\text{m})$. The detection range is $[-75.0\text{m}, 75.0\text{m}]$ for the x and y axis, and $[-4\text{m}, 8\text{m}]$ for the z axis. For ablation studies, we train our network on only 20% of the training data.

6.2. Ablation Study

Box-Attention vs. Alternatives. We first compare Box-Attention with Self-Attention [40], Deformable-Attention [47] and Dynamic-Attention [4] in end-to-end object detection. Results in Table 1 indicate an improvement for Box-Attention on all metrics, with the highest gain from *small* objects (AP_S) (up to 2 points). Furthermore, the Box-Attention requires a smaller number of FLOPs compared to other attention mechanisms. We also report Box-Attention *without* the where-to-attend module that adopts the reference window but not the transformation functions (translation and scaling). It can be seen in Table 1 that the where-to-attend module contributes more than 2 points in all categories. This shows the importance of translation and scaling functions in learning to attend to the relevant region.

BoxeR-2D Ablation. As BoxeR-2D utilizes multi-scale reference windows in its encoder for predicting object proposals, these proposals serve as reference windows in the decoder. In Table 2a, we evaluate the quality of our object proposals by adding object proposal refinement in the decoder layers. While such refinement proved beneficial in [47], we observe more than 1 point drop in AP. This suggests that when object proposals are generated by the BoxeR-2D encoder with reference windows, they are sufficient for the BoxeR-2D decoder to predict objects without the need for a refinement in each step (see Fig. 4). Our BoxeR-2D is flexible, as we can easily plug Instance-Attention into its decoder in order to predict both the object location and its overlay. Table 2b shows BoxeR-2D benefits from multi-task training (object detection and instance segmentation). Note that this is not the case for DETR [1]. In our setting, the multi-task training does not require more parameters except for a small mask prediction head. The training is also stable without any change in hyper-parameters.

BoxeR-3D Ablation. We ablate the effectiveness of our BoxeR-3D design on 3D object detection in Table 3. The table indicates the role of rotation transformation in the *where-to-attend* module, which contributes more than 1 point in all categories at the expense of a small amount of computation. Specifically, we found rotation transformation is most effective when added to box-attention in the decoder layers. Table 3 also shows the comparison between multi-angle *vs*. single-angle reference window in the BoxeR-3D encoder layers. Using a multi-angle reference window yields an improvement for the vehicle and cyclist category, while remaining stable for pedestrians. This suggests that each head in multi-head attention is able to effectively capture the information of different rotation angles.

	Vehicle		Pedestrian		Cyclist	
	$AP\uparrow$	$APH\uparrow$	$AP\uparrow$	$APH\uparrow$	$AP\uparrow$	$APH\uparrow$
$\mathcal{F}_r$ + multi-angle	**70.4**	**70.0**	**64.7**	53.5	**50.2**	**48.9**
w/o $\mathcal{F}_r$	69.4	68.7	63.3	52.8	47.4	46.1
w/o multi-angle	70.0	69.3	**64.7**	**53.7**	48.2	47.0

Table 3. **BoxeR-3D ablation** on the Waymo `val` set (LEVEL_1 difficulty). Adding $\mathcal{F}_r$ gives better performance for detecting 3D bounding boxes. Multi-angle reference windows further improve results by taking advantage of an explicit angle prior.

6.3. Comparison with Existing Methods

2D Object Detection. Table 4 lists the performance of previous methods and BoxeR-2D using ResNet-50 and ResNet-101 backbones. The first part contains convolution-based object detectors while the second part focuses on transformer-based methods. Across backbones BoxeR-2D achieves better results on all metrics. Notably, BoxeR-2D outperforms other methods in detecting small objects, with more than 2 AP_S points improvement compared to Deformable DETR. In addition, our network is able to converge quickly with the standard $3\times$ schedule setting [43]. It is further worth

Figure 6. **Qualitative results** for instance detection and segmentation in the COCO 2017 `test-dev` set generated by BoxeR-2D (More qualitative results are in the supplementary document).

Method	Backbone	Epochs	end-to-end	AP↑	AP_{50}↑	AP_{75}↑	AP_S↑	AP_M↑	AP_L↑
Faster RCNN-FPN [31]	R-101	36	✗	36.2	59.1	39.0	18.2	39.0	48.2
ATSS [46]	R-101	24	✗	43.6	62.1	47.4	26.1	47.0	53.6
Sparse RCNN [37]	X-101	36	✓	46.9	66.3	51.2	28.6	49.2	58.7
VFNet [45]	R-101	24	✗	46.7	64.9	50.8	28.4	50.2	57.6
Deformable DETR [47]	R-50	50	✓	46.9	66.4	50.8	27.7	49.7	59.9
Deformable DETR [47]	R-101	50	✓	48.7	68.1	52.9	29.1	51.5	62.0
Dynamic DETR [4]	R-50	50	✓	47.2	65.9	51.1	28.6	49.3	59.1
TSP-RCNN [38]	R-101	96	✓	46.6	66.2	51.3	28.4	49.0	58.5
BoxeR-2D	R-50	50	✓	50.0	67.9	54.7	30.9	52.8	62.6
BoxeR-2D (3× schedule)	R-50	36	✓	49.9	68.0	54.4	30.9	52.6	62.5
BoxeR-2D (3× schedule)	R-101	36	✓	**51.1**	**68.5**	**55.8**	**31.5**	**54.1**	**64.6**

Table 4. **Comparison of BoxeR-2D in object detection** on the COCO 2017 `test-dev` set with various backbone networks. BoxeR-2D outperforms other methods including transformer-based object detectors with a faster training schedule.

	Epoch	end-to-end	AP↑	AP_S↑	AP_M↑	AP_L↑	AP^m↑	AP^m_S↑	AP^m_M↑	AP^m_L↑
Mask R-CNN [11]	36	✗	43.1	25.1	46.0	54.3	38.8	21.8	41.4	50.5
QueryInst [9]	36	✗	48.1	-	-	-	42.8	24.6	45.0	55.5
SOLQ [6]	50	✓	48.7	28.6	51.7	63.1	40.9	22.5	43.8	54.6
BoxeR-2D (3× schedule)	36	✓	**51.1**	**31.5**	**54.1**	**64.6**	**43.8**	**25.0**	**46.5**	**57.9**

Table 5. **Comparison of BoxeR-2D in instance segmentation** on the COCO 2017 `test-dev` set using a R-101 backbone. BoxeR-2D shows better results in both detection and instance segmentation.

to point out that BoxeR-2D trained with the 3× schedule reaches competitive results.

2D Instance Segmentation. We compare BoxeR-2D with other instance segmentation methods. In Table 5, the 3× schedule is used in the training of our network. BoxeR-2D improves on all of the metrics for bounding boxes and instance masks against QueryInst [9]. Furthermore, our method outperforms SOLQ [6], a transformer-based method, by around 2 points on all categories. The visualization of the BoxeR-2D prediction can be seen in Fig. 6.

3D Object Detection. Table 6 shows the performance of BoxeR-3D and other 3D object detectors along with a naive implementation of Deformable DETR [47] as our baseline. It can be seen that BoxeR-3D consistently improves over the baseline on all metrics, specially for small objects like pedestrians. Our network reaches a competitive result compared to highly optimized methods in the vehicle category. However, there is still a gap between BoxeR-3D and previous methods in the pedestrian category. It should be noted that compared to others we only use *minimal* prior knowledge per category.

	end-to-end	Vehicle		Pedestrian	
		AP↑	APH↑	AP↑	APH↑
PointPillar [18]	✗	55.2	54.7	60.0	49.1
PV-RCNN [33]	✗	**65.4**	**64.8**	-	-
RSN S_1f [36]	✗	63.0	62.6	**65.4**	**60.7**
Deformable DETR [47]	✓	59.6	59.2	45.8	36.2
BoxeR-3D	✓	63.9	63.7	61.5	53.7

Table 6. **Comparison of BoxeR-3D in 3D object detection** on the Waymo Open `val` set (LEVEL_2 difficulty). Despite the lack of any class-specific optimization, BoxeR-3D is surprisingly effective and even competitive on the Vehicle category.

7. Conclusion and Limitations

In this paper, we presented a transformer-based detector for end-to-end object detection and instance segmentation named BoxeR. The core of the network is the box-attention, which is designed to attend to an image region by learning the transformations from an initial reference window. Because of its flexibility, BoxeR can perform both 2D and 3D end-to-end object detection along with instance segmentation without hand-crafted modules. Experiments on the COCO and Waymo Open datasets confirm the effectiveness of the proposed architecture.

Similar to other transformer-based architectures, we observed a larger memory footprint during the training of our networks compared to convolution-based architectures such as Faster R-CNN or Mask R-CNN. This results in the need of more advanced GPUs and higher energy consumption. Moreover, under the same FLOPs, our box-attention is slower than a convolution operation. The reasons may come from the unordered memory access of the grid sampling in our box-attention and the highly-optimized hardware and implementation for the traditional convolution. We expect to mitigate some of these problems with a more optimized implementation.

Acknowledgements

This work has been financially supported by TomTom, the University of Amsterdam and the allowance of Top consortia for Knowledge and Innovation (TKIs) from the Netherlands Ministry of Economic Affairs and Climate Policy.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 2, 5, 6, 7

[2] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016. 2

[3] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *CVPR*, 2021. 2

[4] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic DETR: End-to-end object detection with dynamic attention. In *ICCV*, 2021. 6, 7, 8

[5] Stéphane d'Ascoli, Hugo Touvron, Matthew L. Leavitt, Ari S. Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *ICML*, 2021. 1

[6] Bin Dong, Fangao Zeng, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. SOLQ: Segmenting objects by learning queries. In *NeurIPS*, 2021. 2, 8

[7] 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. In *ICLR*, 2021. 1, 2

[8] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 2

[9] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *ICCV*, 2021. 8

[10] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. In *CVPR Workshop*, 2020. 3

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 2, 3, 5, 8

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997. 1

[14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015. 2

[15] Geoffrey E. Hinton Jimmy Lei Ba, Jamie Ryan Kiros. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[16] Durk P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[17] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart*, 1955. 6

[18] 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. 7, 8

[19] Yann Lecun and Yoshua Bengio. *Convolutional Networks for Images, Speech and Time Series*, pages 255–258. MIT Press, 1995. 1

[20] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar R-CNN: an efficient and universal 3d object detector. In *CVPR*, 2021. 3

[21] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 2, 6

[22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 4, 5

[23] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul W. Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *IJCV*, 2020. 2

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 2

[25] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. In *ICCV*, 2021. 3

[26] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *ICCV*, 2021. 3

[27] Pedro H. O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *ECCV*, 2016. 2

[28] 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*, 2017. 3

[29] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019. 1, 2

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 2, 5, 8

[32] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. In *ICCV*, 2021. 3

[33] 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 *CVPR*, 2020. 3, 6, 8

[34] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *CVPR*, 2021. 2

[35] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev andScott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov.

Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2, 3, 6

[36] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. RSN: range sparse net for efficient, accurate lidar 3d object detection. In *CVPR*, 2021. 6, 7, 8

[37] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse r-cnn: End-to-end object detection with learnable proposals. In *CVPR*, 2021. 2, 8

[38] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris Kitani. Rethinking transformer-based set prediction for object detection. In *ICCV*, 2021. 8

[39] Yi Tay, Mostafa Dehghani, Jai Prakash Gupta, Dara Bahri, Vamsi Aribandi, Zhen Qin, and Donald Metzler. Are pre-trained convolutions better than pre-trained transformers? In *ACL*, 2021. 1

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 3, 6, 7

[41] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. In *CVPR*, 2021. 2

[42] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2

[43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6, 7

[44] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 3, 6

[45] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sünderhauf. Varifocalnet: An iou-aware dense object detector. In *CVPR*, 2021. 8

[46] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 8

[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 1, 2, 3, 5, 6, 7, 8