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

Nguyen, D.-K.; Yu, J.; Booij, O.; Oswald, M.R.; Snoek, C.G.M.

DOI
10.48550/arXiv.2111.13087
10.1109/CVPR52688.2022.00473

Publication date
2022

Document Version
Author accepted manuscript

Published in
2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition

Citation for published version (APA):
BoxeR: Box-Attention for 2D and 3D Transformers

Duy-Kien Nguyen1  Jihong Ju2  Olaf Booij2  Martin R. Oswald1  Cees G. M. Snoek1
Atlas Lab - 1University of Amsterdam  2TomTom
{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. 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 transformerR, 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 ViT Transformer [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 architecture for instance segmentation, 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-
A transformer with self-attention was also adopted for 3D object detection. 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 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. 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 ($K$):

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

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.

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

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).

![Figure 2. Box-Attention.](image-url)

- **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 $3 \times 3$ 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.

- **Box-Attention computation:**
  - **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 $3 \times 3$ 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:** learning where-to-attend

- **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 comparing 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.

$$h_i = \text{BoxAttention}(Q, K_i, V_i) = \sum_{m \times m} \text{softmax}(Q K_i^\top) \ast V_i,$$  

(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 $\{b_1, \ldots, b_t\}$ 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 $h_{(1,\ldots,t)} \in \mathbb{R}^{d_h}$ feature now is the weighted average of $t \times m \times m$ vectors in $v_{(1,\ldots,t)}$ 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 $h_i^\text{mask} \in \mathbb{R}^{m \times m \times d}$ for instance segmentation. When 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 $h_i^\text{mask}$. 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 $h_{(1,\ldots,t)}$ and $h_{(1,\ldots,t)}^\text{mask}$ 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). To be specific, we denote the reference window of query $q$ by $b_q = [x, y, w, w_y] \in [0, 1]^4$ where $x, y$ indicate its center position, $w, w_y$ 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:

$$\mathcal{F}_t(b_q, q) = b_q' = [x + \Delta_x, y + \Delta_y, w, w_y],$$  

(4)

where $\Delta_x$ and $\Delta_y$ 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 $b_q$

$$\mathcal{F}_s(b_q, q) = b_q' = [x, y, w, w_y + \Delta_w, w_y + \Delta_{w_y}],$$  

(5)

where $\Delta_w$ and $\Delta_{w_y}$ are offsets for the reference window size. The offset parameters (i.e., $\Delta_x, \Delta_y, \Delta_w, \Delta_{w_y}$) 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 $b_i'$ 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-

![BoxeR structure](image_url)
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\}^t_{j=1} (t=4) extracted from \(C_5\) 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\}^t_{j=1}. Note that the multi-scale contextual representations \{e^j\}^t_{j=1} are in the same resolution as the inputs \{x^j\}^t_{j=1}.

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 \(\{x_1, x_2, x_3, x_4\}\) (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 \((s + 1)^{th}\) decoder layer by \(x_s \in \mathbb{R}^{N \times d}\). The \((s + 1)^{th}\) 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 \( \theta \) is the rotation angle of \( b_q \) around its center in normalized coordinates. The final rotation function, \( F_r \), predicts an offset of the window rotation angle. It then applies a rotation matrix on the \( m \times m \) grid coordinates sampled from \( b_q \)

\[
F_r(b_q, q) = b_q' = [x, y, w_x, w_y, \theta + \Delta \theta],
\]

where \( \Delta \theta \) is a rotation offset w.r.t. the reference window angle. Again, we use a linear projection on \( q \) to predict \( \Delta \theta \).

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 \( \{\frac{-3\pi}{2}, 0, \frac{3\pi}{2}\} \). 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.

We use the Adam optimizer [16] with \( \alpha = 0.0002 \) 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 40th 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>FLOPs</th>
<th>AP</th>
<th>AP\text{p}</th>
<th>AP\text{m}</th>
<th>AP\text{n}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention [40]</td>
<td>187G</td>
<td>36.2</td>
<td>16.3</td>
<td>39.2</td>
<td>53.9</td>
</tr>
<tr>
<td>Deformable-Attention\textsuperscript{†} [47]</td>
<td>173G</td>
<td>46.9</td>
<td>29.6</td>
<td>50.1</td>
<td>61.6</td>
</tr>
<tr>
<td>Dynamic-Attention [4]</td>
<td>-</td>
<td>47.2</td>
<td>28.6</td>
<td>49.3</td>
<td>59.1</td>
</tr>
<tr>
<td>BoxeR (Ours)</td>
<td>167G</td>
<td>48.7</td>
<td>31.6</td>
<td>52.3</td>
<td>63.2</td>
</tr>
<tr>
<td>w/o ( F_r ) and ( F_s )</td>
<td>164G</td>
<td>46.4</td>
<td>29.6</td>
<td>49.8</td>
<td>59.7</td>
</tr>
</tbody>
</table>

\textsuperscript{†} 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.
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 (0.32m, 0.32m). The detection range is [-75.0m, 75.0m] for the x and y axis, and [-4m, 8m] 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.

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× schedule setting [43]. It is further worth

<table>
<thead>
<tr>
<th></th>
<th>Vehicle</th>
<th>Pedestrian</th>
<th>Cyclist</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_r$ multi-angle</td>
<td>70.4</td>
<td>70.0</td>
<td>64.7</td>
</tr>
<tr>
<td>w/o $F_r$</td>
<td>69.4</td>
<td>68.7</td>
<td>63.3</td>
</tr>
<tr>
<td>w/o multi-angle</td>
<td>70.0</td>
<td>69.3</td>
<td>64.7</td>
</tr>
</tbody>
</table>

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

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

**2D Instance Segmentation.** We compare BoxeR-2D with other instance segmentation methods. In Table 5, the 3x 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.

### 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 over, under the same FLOPs, our box-attention is slower than a convolution operation. The reasons may come from

### 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


[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


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


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


[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


[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