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

This supplementary material provides more information about our methodology, more implementation details for better reproducibility as well as additional qualitative results.

Contents

A. Additional Results

Here, we provide a comparison on the effects of dropout during the training of BoxeR. Specifically, we evaluate BoxeR-2D with and without dropout in Table 1 along with the speed (fps), # params, and FLOPs computation of our architecture.

B. Prediction Head in BoxeR

In order to take advantage of spatial information from the box-attention module, we design the prediction head to regress relative offsets w.r.t. the corresponding reference window. The reference window is used as the initial guess of the bounding box in both object proposal and detection stage. We denote σ and σ^{-1} as the $\mathrm{sigmoid}$ and the $\mathrm{inverse\ sigmoid}$ function, respectively. This design allows the prediction head of BoxeR to utilize the information of features in box-attention which learn to predict offsets from a reference window. Note that, we use $0{=}\sigma^{-1}(0.5)$ for the height prediction of the 3D bounding boxes since we have no information in the reference window.

BoxeR-2D. Given a pre-defined reference window $b_q{=}[x, y, w_x, w_y] \in [0, 1]^4$, the predicted bounding box $\hat{b}_q{=}\Big[\sigma\big(\Delta_x{+}\sigma^{-1}(x)\big), \sigma\big(\Delta_y{+}\sigma^{-1}(y)\big), \sigma\big(\Delta_{w_x}{+}\sigma^{-1}(w_x)\big), \sigma\big(\Delta_{w_y}{+}\sigma^{-1}(w_y)\big)\Big]$ where Δ_x, Δ_y, Δ_{w_x}, and Δ_{w_y} are offsets of the bounding box center, width, and height predicted by the prediction head.

In the object proposal stage, the prediction head contains a 3-layer perceptron with a ReLU activation function and a hidden dimension d for the offsets prediction and a linear projection for the bounding box binary classification (*i.e.*, foreground and background). The encoder features of the top scoring proposals are picked as object queries in the decoder, while its bounding boxes serve as its reference windows (the object queries and encoder features do not have gradient flow).

In the prediction stage, the same architecture is used to predict the offsets from the reference windows and to classify object categories. The mask prediction head contains a deconv layer with $2{\times}2$ kernel size and stride 2 followed by two $1{\times}1$ conv layers with ReLU activation function and hidden dimension d. Similar to [2], the last conv layer outputs a mask prediction of $28{\times}28{\times}\ \mathrm{num\_class}$ for each bounding box.

BoxeR-3D. Given a reference window $b_q{=}[x, y, w_x, w_y, \theta] \in [0, 1]^5$, the predicted bounding box $\hat{b}_q{=}\Big[\sigma\big(\Delta_x{+}\sigma^{-1}(x)\big), \sigma\big(\Delta_y{+}\sigma^{-1}(y)\big), \sigma\big(\Delta_z\big), \sigma\big(\Delta_{w_x}{+}\sigma^{-1}(w_x)\big), \sigma\big(\Delta_{w_y}{+}\sigma^{-1}(w_y)\big), \sigma\big(\Delta_{w_z}\big), \sigma\big(\Delta_\theta{+}\sigma^{-1}(\theta)\big)\Big]$ where Δ_x, Δ_y, Δ_z, Δ_{w_x}, Δ_{w_y}, Δ_{w_z}, and Δ_θ are the offsets of the bounding box center, length, width, height, and angle predicted by the prediction head. Note that, we normalize the center and size of the 3D bounding box by the detection range. The angle of the 3D bounding box is converted to the range $[0, 2\pi]$ and normalized by 2π.

The prediction head in both the object proposal and prediction stage shares the same architecture: a 3-layer perceptron with a ReLU activation function and a hidden dimension d for offsets prediction of the 3D bounding boxes, and a linear projection for the classification task (binary classification for object proposal and multi-class classification for prediction). Similarly, the prediction head is applied to each of the encoder features in the object proposal stage. Since BoxeR-3D uses reference windows of three angles per query, the prediction head outputs three object proposals each of which corresponds to a reference window of one angle. The top

	AP↑	AP_S↑	AP_M↑	AP_L↑	AP^m↑	AP^m_S↑	AP^m_M↑	AP^m_L↑	# params	FLOPs	fps
Box-Attention *only*	48.7	31.6	52.3	63.2	-	-	-	-	39.7M	167G	17.3
w/ dropout	47.8	30.9	50.6	62.1	-	-	-	-	39.7M	167G	17.3
Instance-Attention	50.0	32.4	53.3	64.5	42.7	22.7	45.9	61.5	40.1M	240G	12.5

Table 1. **More BoxeR-2D ablation** on the COCO 2017 `val` set using a R-50 backbone pretrained on ImageNet. By removing dropout during training, our BoxeR-2D shows better performance in all metrics without any extra computation. Note that we report the inference speed of our network including the post-processing time.

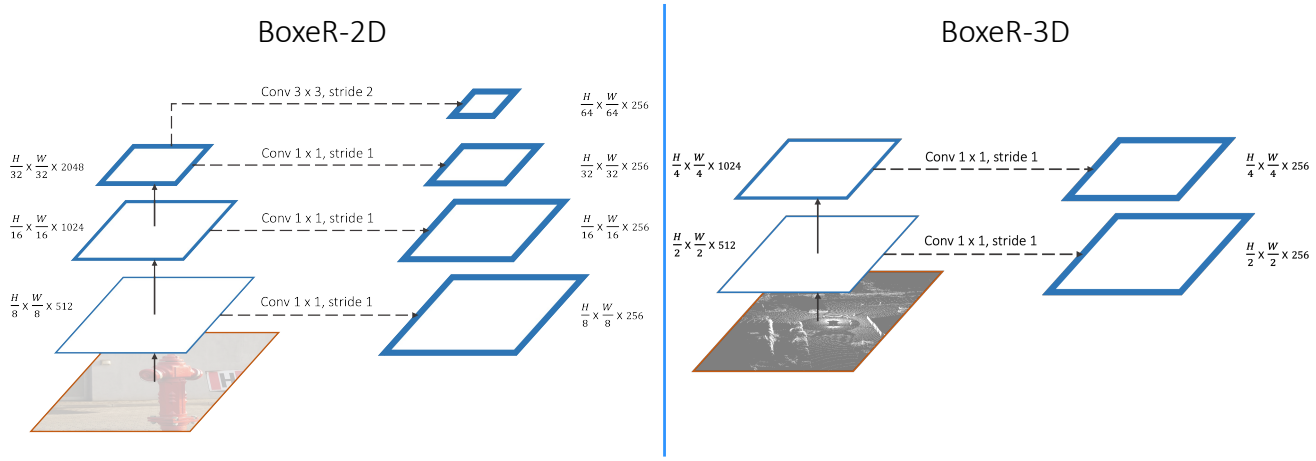

Figure 1. **Multi-scale feature maps** for BoxeR-2D (left) and BoxeR-3D (right).

scoring proposals are picked in the same way as in BoxeR-2D.

C. Losses in BoxeR Training

We review the losses that are used during our training in Table 2. BoxeR-2D is trained with a classification and a 2D box loss for the object proposal stage. In the prediction stage, classification, 2D box, and mask loss are used. The Hungarian matcher [4] in BoxeR-2D does not use the mask cost but only the classification and box cost as in [8].

For 3D object detection, we train BoxeR-3D with a classification and 3D box loss for both the object proposal and prediction stage. Due to the limitation of our computational resources, we use the same loss weights as in BoxeR-2D except for the weight of the angle loss. Following [1], the 3D Hungarian matcher is designed to be consistent with our setting of loss weights.

D. Multi-scale feature maps for BoxeR

The multi-scale feature maps are constructed for BoxeR-2D and BoxeR-3D as in Fig. 1. In end-to-end object detection and instance segmentation, we use ResNet [3] and ResNeXt [7] as backbones for BoxeR-2D. As discussed in Section 4, we follow [8] to construct our multi-scale feature maps.

In end-to-end 3D object detection, we first extract bird's-eye view image features from PointPillar [5] 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. This results in a feature map $x \in \mathbb{R}^{H \times W \times C}$ where $H{=}W{=}468$ and $C{=}128$. In order to construct multi-scale feature maps, we use a convolutional backbone as in Fig. 2. Note that each convolution layer is followed by a batch normalization layer and a ReLU activation layer.

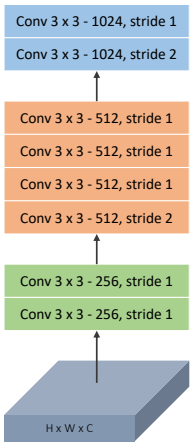

Figure 2. **Convolutional backbone** for BoxeR-3D.

E. More Implementation Details

Initialization of multi-scale box-attention. Unlike deformable attention [8], our box-attention module does not require a complicated initialization. Weight and bias parameters of the linear projection for generating attention weights

	Box Loss		Angle Loss	Classification Loss	Mask Loss	
	L1 loss	GIoU loss	L1 loss	Focal loss	BCE loss	DICE/F-1 loss
BoxeR-2D	5	2	n/a	2	5	5
BoxeR-3D	5	2	4	2	n/a	n/a

Table 2. **Losses and weights** used in training of BoxeR-2D and BoxeR-3D (n/a: not available). We use the same loss weights between BoxeR-2D and BoxeR-3D.

Figure 3. **Qualitative results** for object detection and instance segmentation generated by BoxeR-2D in the COCO 2017 `test-dev` set.

are initialized to zero. For predicting the offsets of the reference windows, weight and bias parameters are initialized to zeros and randomly, respectively. By initializing uniform attention weights, box-attention gathers all information within the attended region to make its decision. Other parameters are randomly initialized.

Initialization of prediction head. Following the initialization of the box-attention module, we initialize weight and bias parameters of the last layer in the 3-layer perceptron for offsets prediction to zero. Other parameters are randomly initialized. By doing so, the prediction head treats reference windows as its initial guess which is consistent with the box-attention module.

Figure 4. **Failure cases of BoxeR-2D.** BoxeR-2D fails to predict very small objects in low light conditions. The last image shows that BoxeR-2D is able to predict the object bounding box and mask but fails to classify the category.

Spatial Encoding. Each query $q \in \mathbb{R}^d$ in image features is assigned a reference window $b_q{=}[x, y, w_x, w_y]$ where x, y are coordinates of the window center corresponding to the query position and w_x, w_y are width and height of the reference window. As in [1], we use a fixed absolute encoding to represent $\{x, y\}$ which adopts a sinusoid encoding to the 2D case. Specifically, x and y are independently encoded in $\frac{d}{2}$ features using sine and cosine functions with different frequencies. The d channel position encoding is their concatenation. Similarly, we get a d channel size encoding of $\{w_x, w_y\}$. The final spatial encoding of query q is the summation of both position and size encodings.

Hyperparameter settings in BoxeR. Within its box of interest, the box-attention module samples a grid of $2{\times}2$ ($m{=}2$) while the instance-attention module samples a grid of $14{\times}14$ ($m{=}14$). We set the hidden dim of BoxeR, $d{=}256$; the hidden dim of feed-forward sub-layers, $d_{\text{feed-forward}}{=}1024$; the number of attention heads, $l{=}8$. This applies to both BoxeR-2D and BoxeR-3D. BoxeR-2D contains 6 encoder and decoder layers ($S{=}6$) while BoxeR-3D consists of 2 encoder and decoder layers ($S{=}2$).

During inference of BoxeR-2D, given BoxeR-2D decoder of S decoder layers, we only output $x_S^{\text{mask}} \in \mathbb{R}^{N\times m\times m\times d}$ in the last decoder layer to speed up its prediction time.

Details of the transformation functions. Our translation and scaling functions predict offsets $\Delta_x, \Delta_y, \Delta_{w_x}, \Delta_{w_y}$ w.r.t. the reference window $b_q{=}[x, y, w_x, w_y] \in [0, 1]^4$ of query q using a linear projection on q as below

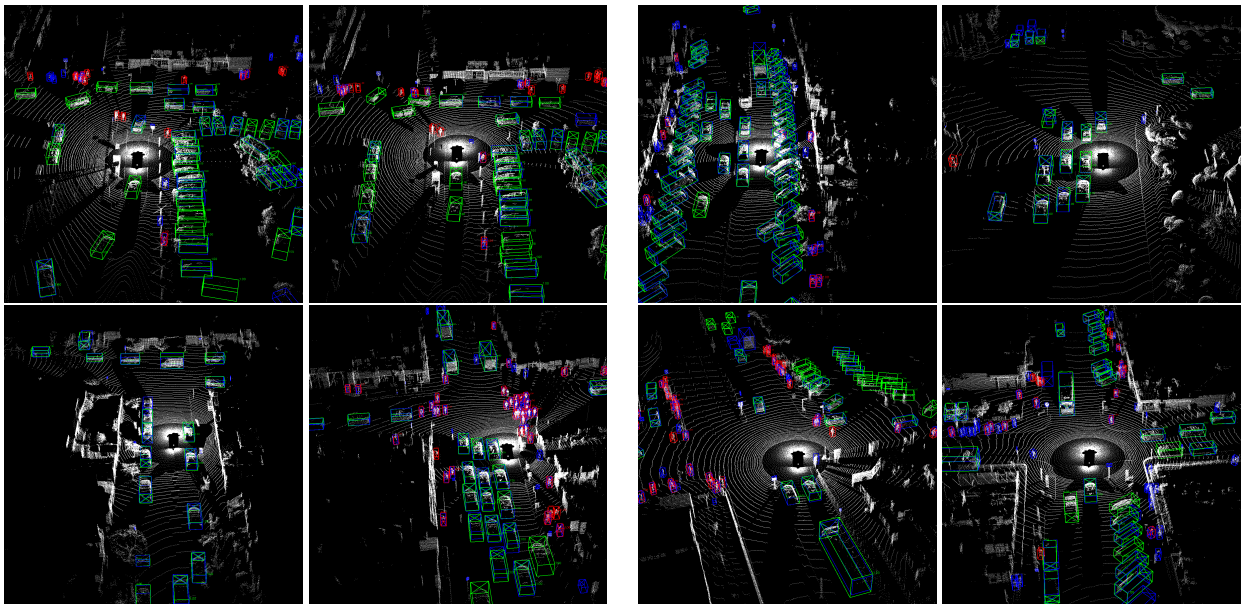

Figure 5. **Qualitative results** for 3D object detection generated by BoxeR-3D on the Waymo `val` set.

$$\Delta_x = (qW_x^\top + b_x) * \frac{w_x}{\tau} \quad , \tag{1}$$

$$\Delta_y = (qW_y^\top + b_y) * \frac{w_y}{\tau} \quad , \tag{2}$$

$$\Delta_{w_x} = \max(qW_{w_x}^\top + b_{w_x}, 0) * \frac{w_x}{\tau} \quad , \tag{3}$$

$$\Delta_{w_y} = \max(qW_{w_y}^\top + b_{w_y}, 0) * \frac{w_y}{\tau} \quad , \tag{4}$$

where W and b are weights and biases of the linear projections; τ=8 is the temperature hyperparameter. The multiplication of the window size $\{w_x, w_y\}$ helps the prediction of the linear projection to be scale-invariant. For angle prediction in 3D object detection, the offset is predicted as

$$\Delta_\theta = (qW_\theta^\top + b_\theta) * \frac{1}{\tau_\theta} \quad . \tag{5}$$

Implementation details of BoxeR-3D. During training of BoxeR-3D, following [6] we use random flipping along the x and y axis, global scaling with a random factor sampled from $[0.95, 1.05]$, global rotation around the z axis with a random angle sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$. The ground truth sampling augmentation is also conducted to randomly "paste" ground-truth objects from other frames to the current one. Similar to the BoxeR-2D, we take the top-300 scoring encoder features as object queries along with the corresponding predicted bounding boxes as their reference window for the BoxeR-3D decoder. In the final prediction, we pick 125 predictions with the highest scores.

F. More Qualitative Results

2D object detection and instance segmentation. We show extra qualitative results for object detection and instance segmentation of the BoxeR-2D with a R-101 backbone in Fig. 3 and Fig. 4.

3D object detection. We show extra qualitative results for 3D object detection of the BoxeR-3D in Fig. 5. The ground-truth boxes are denoted in blue while the predicted pedestrian and vehicle are in red and green. It can be seen that BoxeR-3D gives a high accuracy for vehicle prediction. Note that, BoxeR-3D detects some vehicles that are missed by annotators. However, BoxeR-3D still struggles with detecting all pedestrians.

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. 2, 3

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[4] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart*, 1955. 2

[5] 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. 2

[6] 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. 4

[7] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2

[8] 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. 2