Exploring Box-attention

Team

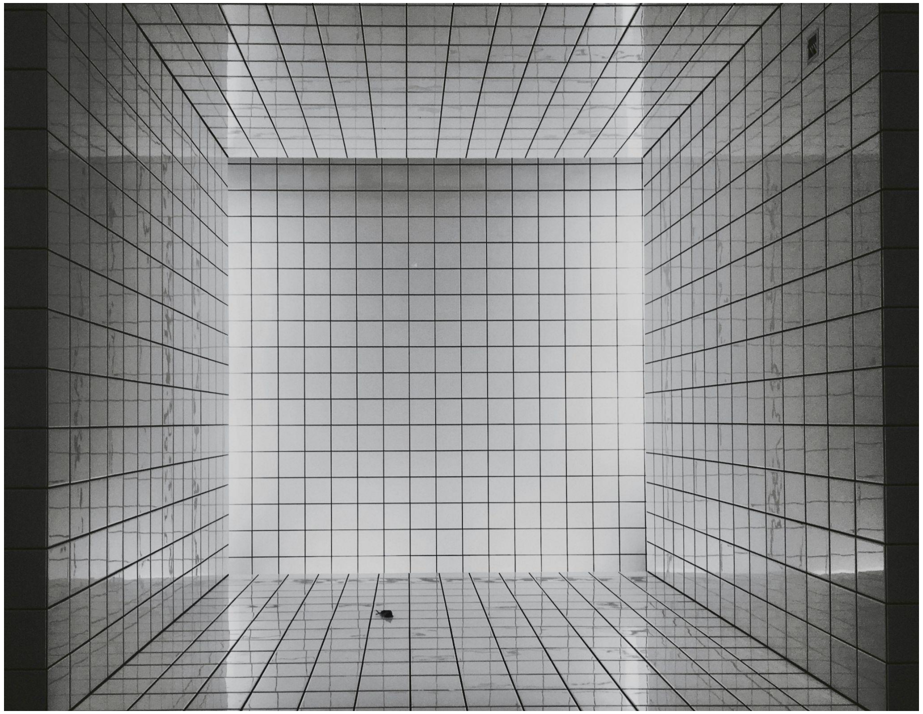
Member1 Tianyi Bao
Member2 Chenhao Qi

Georgetown University Computer Science Department

Motivation

- Existing transformers for computer vision do not explicitly consider the inherent regularities of vision modality e.g. positional encoding - infer the spatial information implicitly inside weights

- Recent evidence reveals - Inductive bias

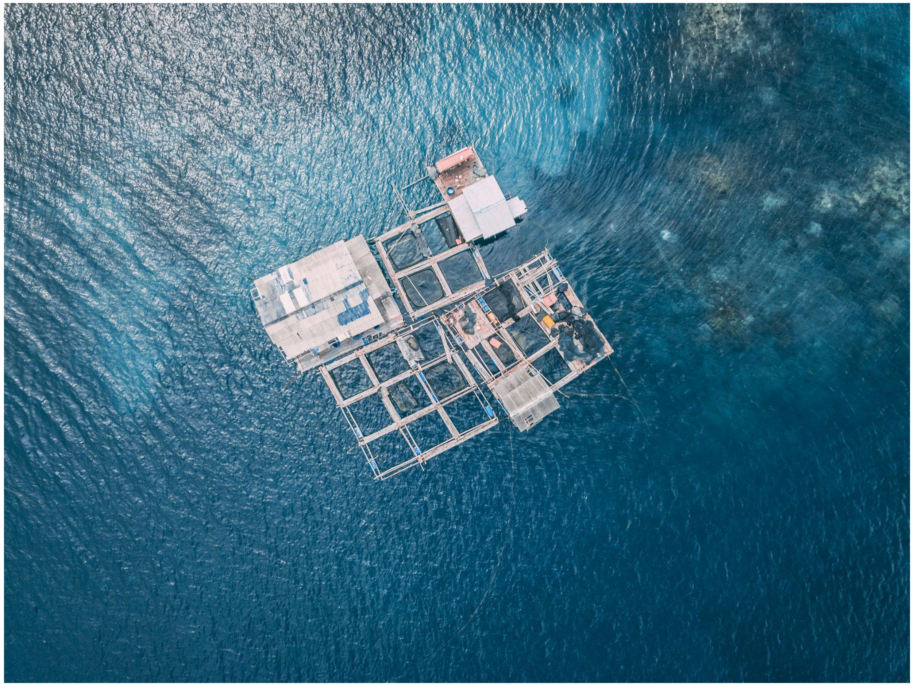


Goal

- Exploring image representation learning
- Research on how to improve performance on image tasks, such as object detection and instance segmentation
- Exploring novel visual Transformer architectures

Main Reference Work(s)

Nguyen, D., Ju, J., Booji, O., Oswald, M.R., & Snoek, C.G. (2021). BoxeR: Box-Attention for 2D and 3D Transformers. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4763-4772.



Method

- learnable embeddings representing relative positions in the grid structure as the key vectors in the attention computations

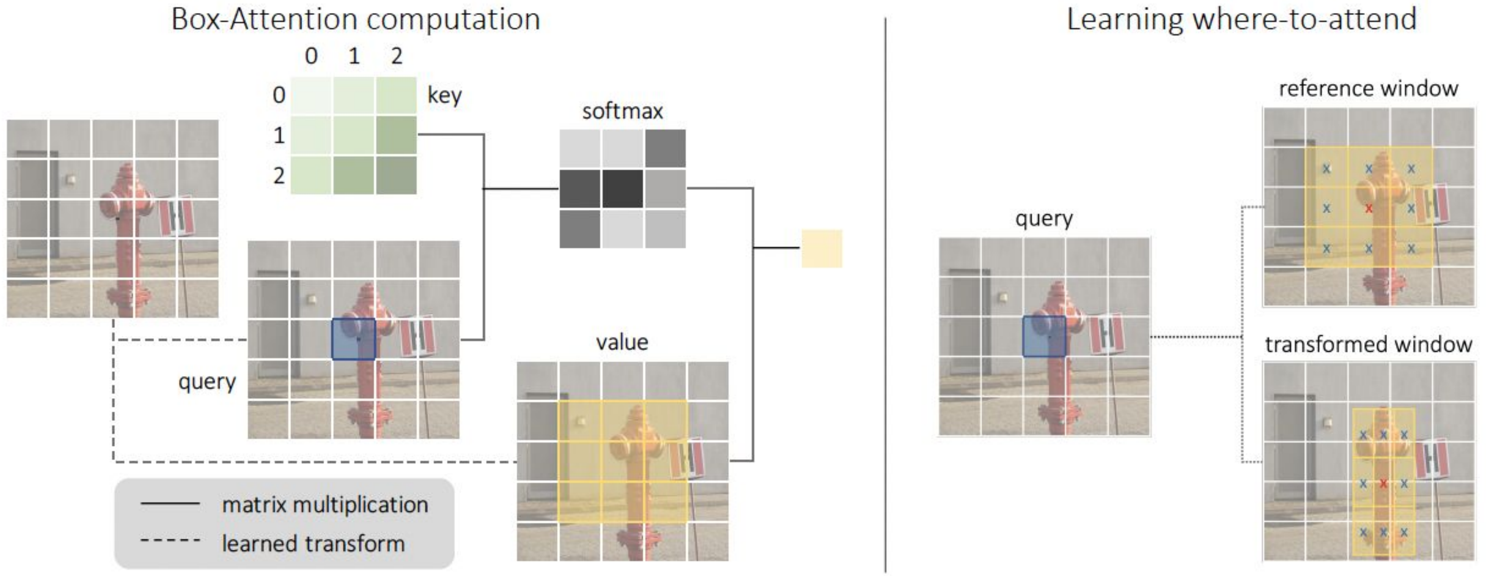
- the module learns to transform a reference window into an attended region by predicting geometric transformations (i.e., translation, scaling, and rotation)

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

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

Method

$$h_i = \mathrm{BoxAttention}(Q, K_i, V_i) = \sum_{m \times m} \mathrm{softmax}(QK_i^\top) * V_i$$



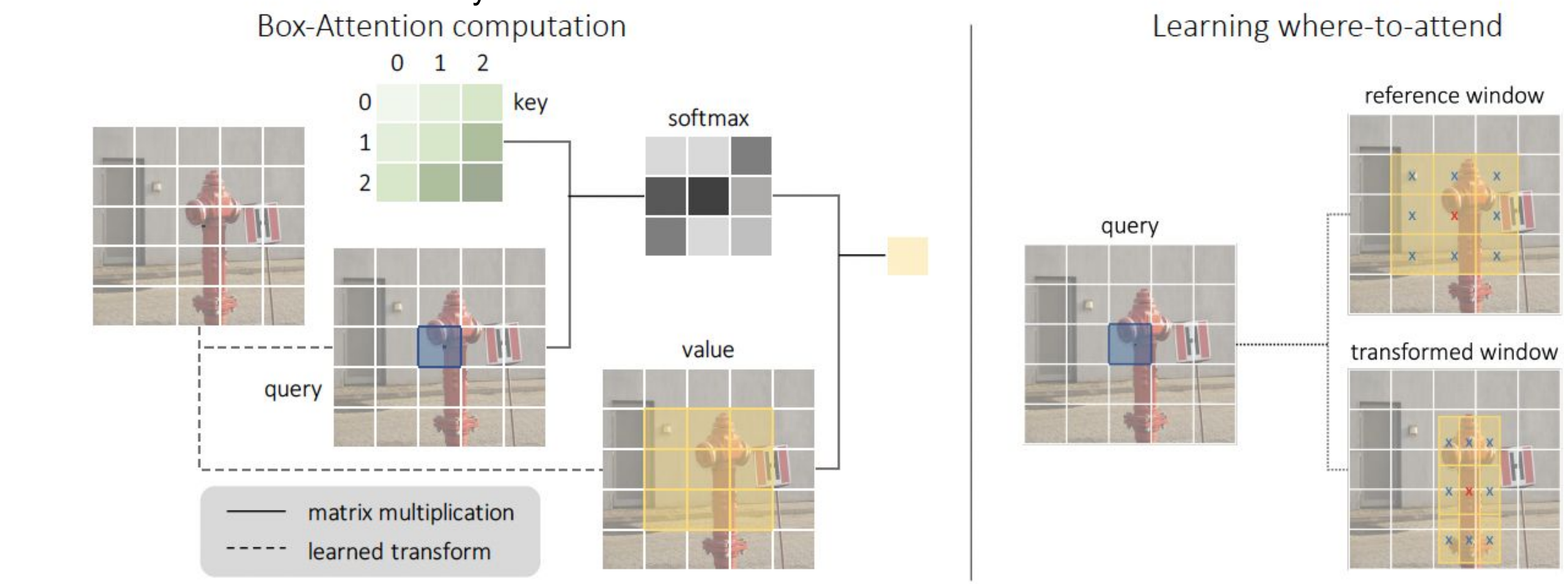
Method

$$h_i = \mathrm{BoxAttention}(Q, K_i, V_i) = \sum_{m \times m} \mathrm{softmax}(QK_i^\top) * V_i$$

given bi, query vector q in the ith attention head extracts a grid feature map vi of size (m,m) from bi using bilinear interpolation

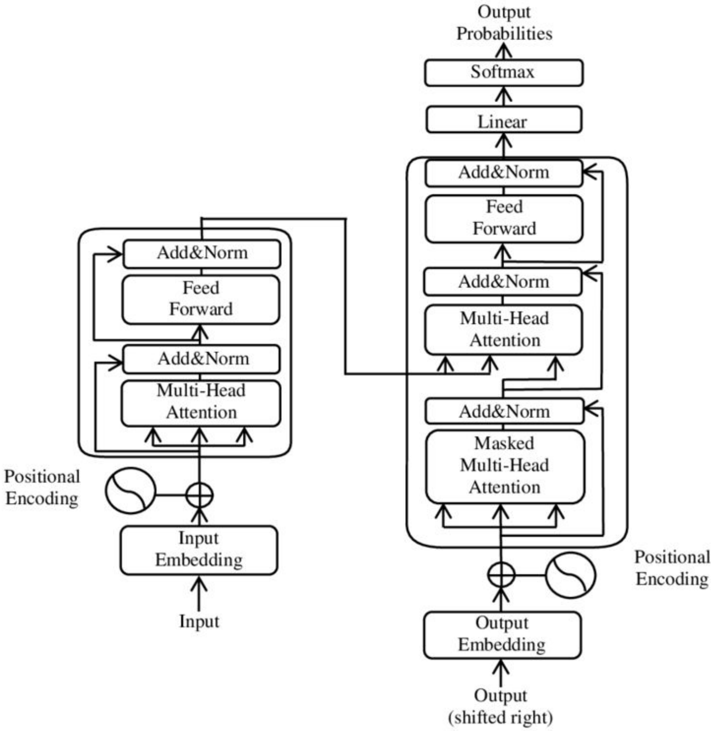
reference window bq = [x,y,wx,wy]
F(bq, q) = b'q = [x, y, wx+Δx, wy+Δy]

learned key vectors

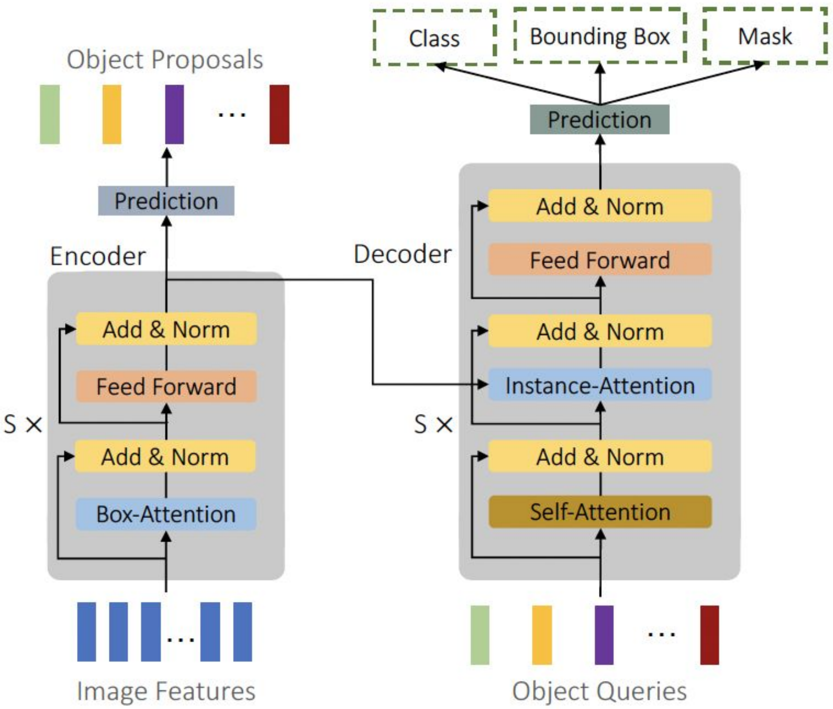


Architecture

Self-attention



Box-attention



Experimental Setup

- Implemented in PyTorch
- Evaluated on the COCO 2017 dataset for object detection
- Uses a ResNet-101 backbone pretrained on ImageNet
- Transformer has 6 encoder and 6 decoder layers
- 8 attention heads, 256 hidden dim, 1024 dim feedforward
- 300 learned object queries
- Loss coefficients: 5x bbox, 2x GIoU, 2x classification

Training Details

- BoxeR is trained using the AdamW optimizer
- Learning rate: 2e-4 for the overall model, 2e-5 for the backbone
- Weight decay: 1e-4
- Max gradient norm: 0.1
- Trained for 270,000 iterations with a batch size of 16
- Learning rate is reduced by 0.1 at iterations 210,000 and 250,000
- Augmentations: horizontal flips, resizing, cropping, normalization
- Auxiliary losses are used during training

Experimental Results

- BoxeR-R101 achieves strong object detection performance on COCO:
  - 50.7 box AP
  - 33.4 AP small, 53.8 AP medium, 65.7 AP large
- Competitive with state-of-the-art detection transformers
- 43.3 mask AP when trained for instance segmentation
- Visualizations show BoxeR attends to relevant regions for each object
- Scales well from 50 to 101 layer backbones

What we got first …

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
```

Final Result:

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.415
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.612
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.449
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.253
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.452
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.541
```

Thank you!