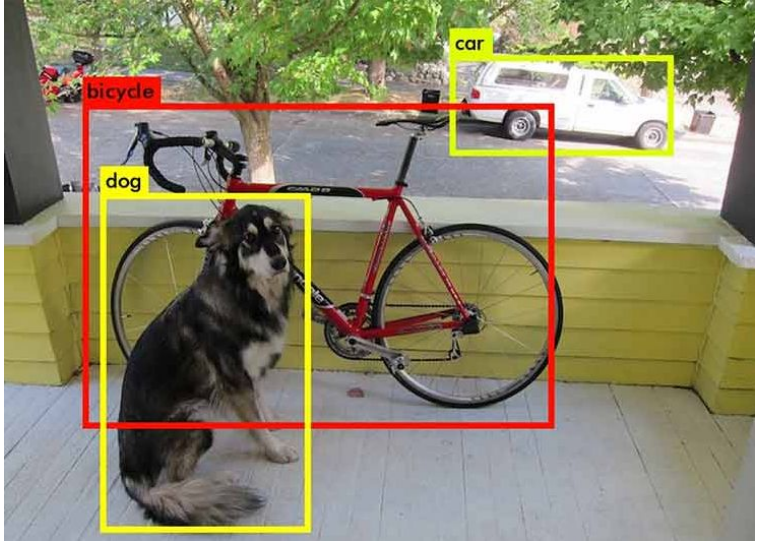
DE:TR: End-to-End Object Detection with Transformers

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier,
Alexander Kirillov, and Sergey Zagoruyko
Facebook AI

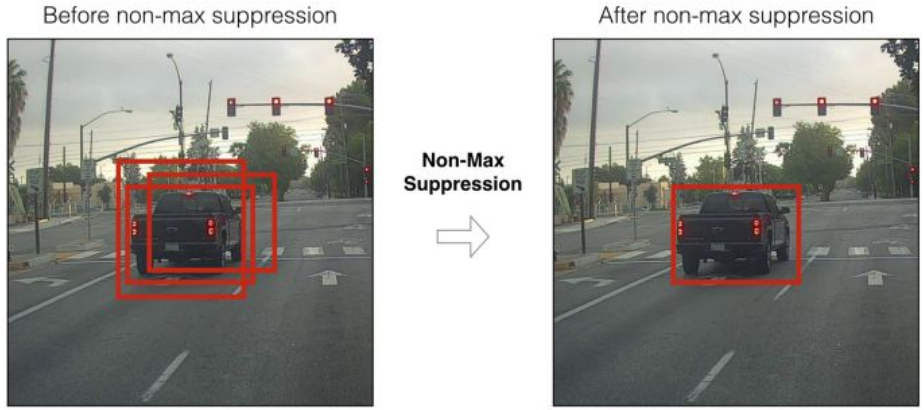
Motivation

Task: Predict a set of bounding boxes and category labels



Previous work:

- Often requires surrogate regressions and classification on a large set of proposals, anchors, window centers, etc.
- Influenced by post-processing steps
- End-to-end philosophy has not been applied to object detection successfully yet



DEtection TRansformer (DETR)

- Treat object detection as direct set prediction problem
- Predict in a single pass a set of objects and models their relation

Components: Backbone, encoder-decoder, and feed forward network

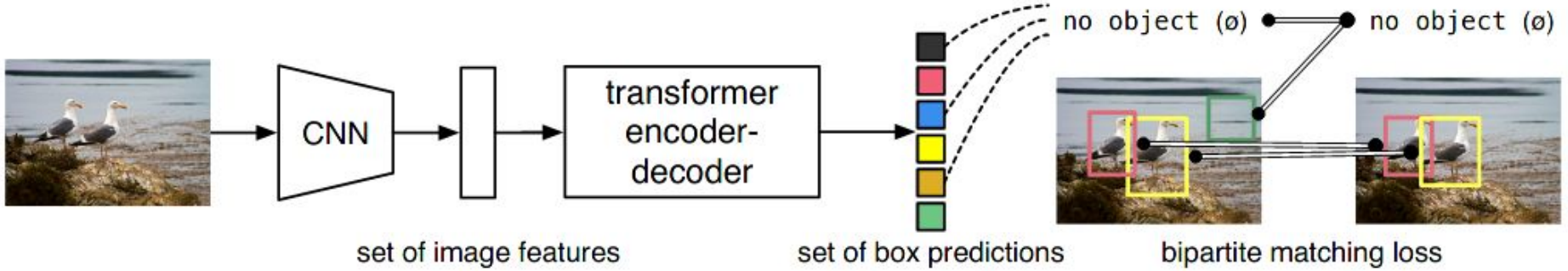


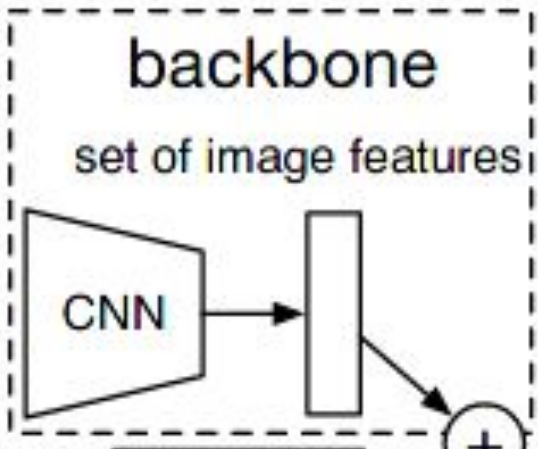
Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" ($\varnothing$) class prediction.

Architecture - Backbone and Preprocessing

Given an image of shape ($3xH_0xW_0$), CNN backbone generates a lower-resolution map of shape (CxHxW)

Typically, C=2048, H,W=$H_0/32$, $W_0/32$

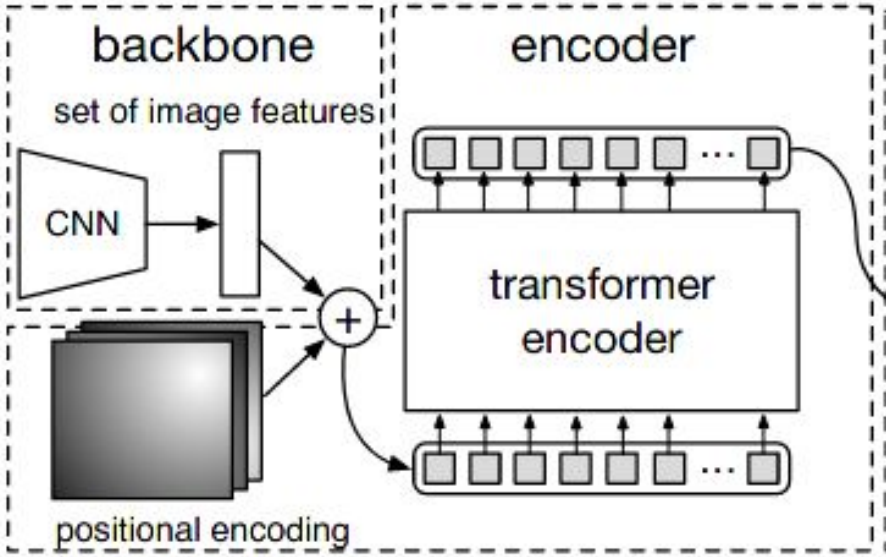
1x1 Convolution reduces channel dimension to shape (dxHxW)



Architecture - Encoder

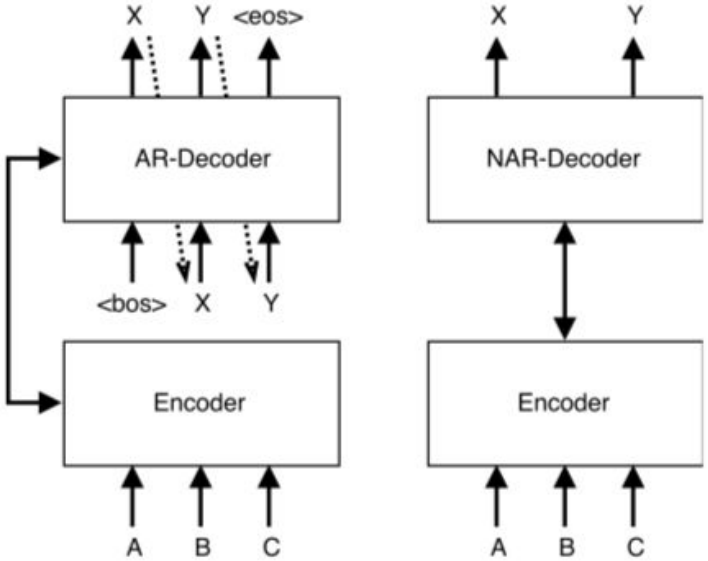
Encoder:

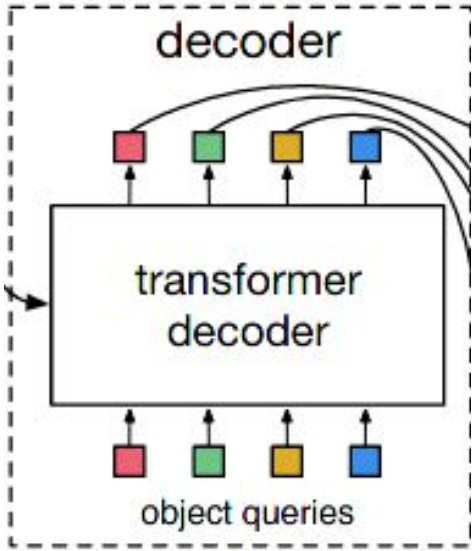
- Collapse the spatial dimensions into one dimension of of shape d x HW
- Add positional encoding before feeding to standard transformer encoder



Architecture - Decoder

- Non-autoregressive
- Input to the decoder are N learned positional encodings called object queries



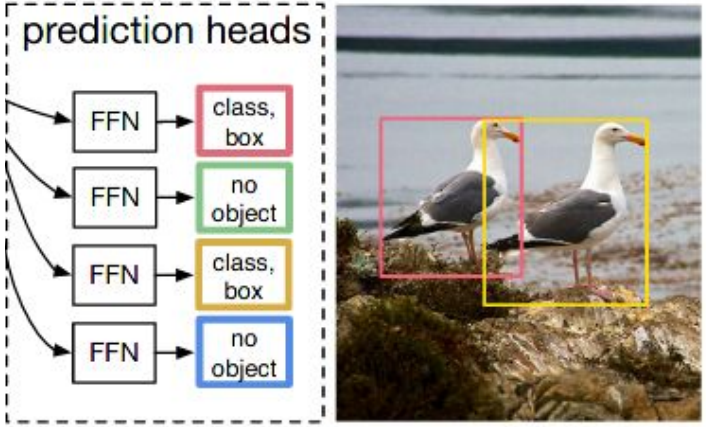


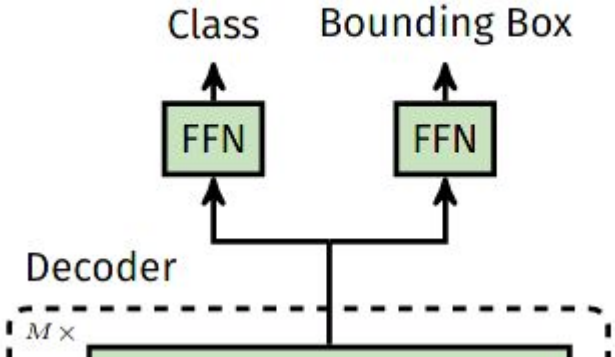
Architecture - Feed-forward networks

3-layer FFN with ReLu activation

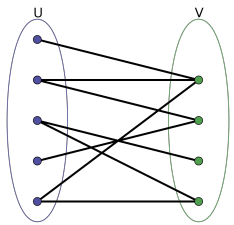
FFN predicts center coordinates, height and width of the box. Another linear layer predicts class label.

N is usually much larger than actual number of objects and thus include null class label (similar to “background”)





Loss - object detection set prediction loss



Find a bipartite matching by finding for a permutation of N with lowest cost

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}),$$

L_{match} is the pair-wise matching cost of label c_i and the boxes $b_i \in [0,1]^4$):

$$-\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}).$$

L_{box} is the bounding box loss - linear combination of L1 loss and Generalized IoU Loss

$$\lambda_{\text{iou}} \mathcal{L}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}} ||b_i - \hat{b}_{\sigma(i)}||_1$$

Loss - object detection set prediction loss

Bipartite matching finds the optimal assignment $\hat{\sigma}$

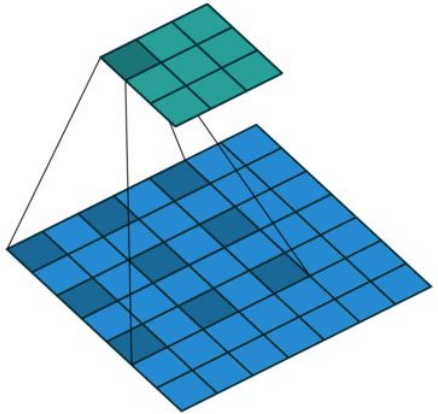
Hungarian Loss

- Linear combination of negative log-like and box loss
- Down-weight the log-probability of null label by a factor 10

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

Auxiliary Loss: Apply the loss after each decoder layer with shared FFNs

Experiments



Dataset: COCO 2017, 118k training images and 5k validation

Backbones:

- ResNet-50 and ResNet-101
- DC5: Add dilation and remove a stride front the first convolution to double resolution
- Scale augmentation to scale input images shortest side [480, 800] and longest side [800, 1333].
- Random crop augmentation

Faster R-CNN Baselines

Stronger faster R-CNN (typically trained with SGD and minimal data augmentation)

- Add generalized IoU to the box loss
- Same random crop augmentation
- Longer training (3x, 9x schedule)

Metric

Average Precision (AP): area under precision-recall curve evaluated at α IoU threshold

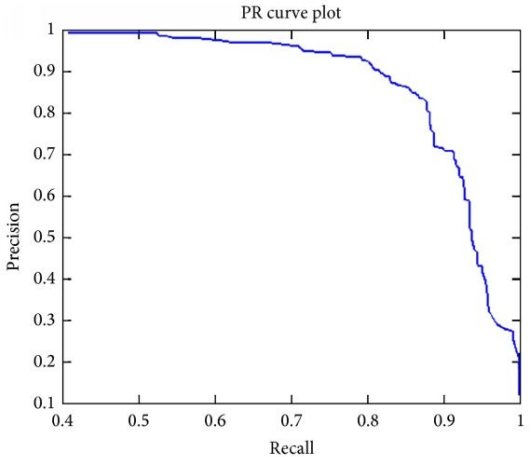
AP: average of AP ranging from IOU=50% to 95% at a 5% step size

AP_{α}: IOU=α

AP_S: AP for small objects: area< 32^2

AP_M: AP for medium objects: 32^2 < area < 96^2

AP_L: AP for large objects: area > 96^2



Result

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

Adding dilation improves smaller objects

Adding R101 improves overall performance

Result

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

DETR is comparable to Faster R-CNN with similar size by improving AP_L by +7.8, but lag behind in AP_S by -5.5 (according to the paper).

Result

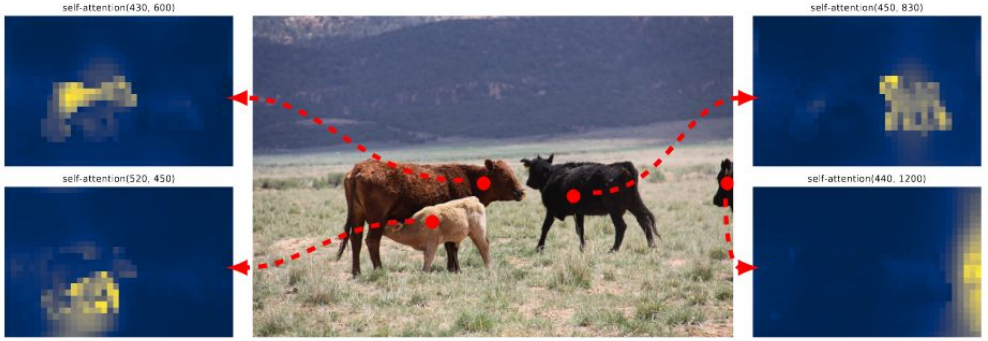
Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

With dilation and thus similar FLOP counts, AP is better but still behind in AP_S.

Ablations - Encoder

Number of encoder layers

Overall AP drops by 3.9, 6.0 AP on larg
objects without encoder layers



Visualization shows that it already
separates instances

Table 2: Effect of encoder size. Each row corresponds to a model with varied number of encoder layers and fixed number of decoder layers. Performance gradually improves with more encoder layers.

#layers	GFLOPS/FPS	#params	AP	AP_{50}	AP_S	AP_M	AP_L
0	76/28	33.4M	36.7	57.4	16.8	39.6	54.2
3	81/25	37.4M	40.1	60.6	18.5	43.8	58.6
6	86/23	41.3M	40.6	61.6	19.9	44.3	60.2
12	95/20	49.2M	41.6	62.1	19.8	44.9	61.9

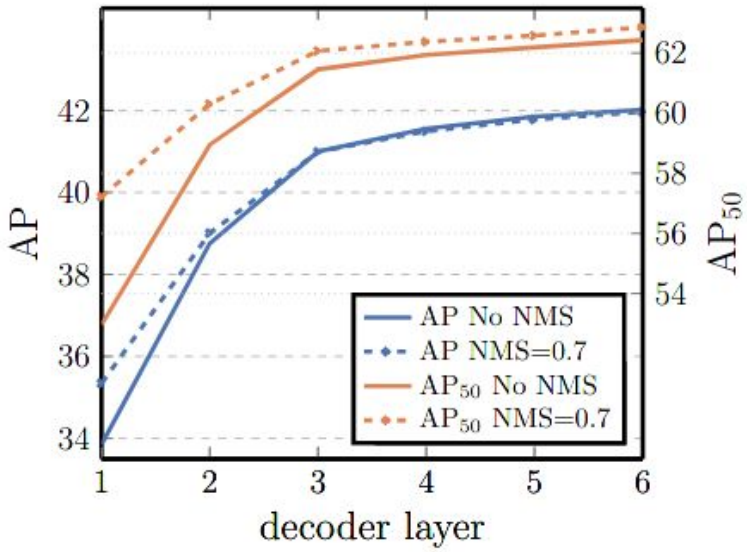
Ablations - Decoder

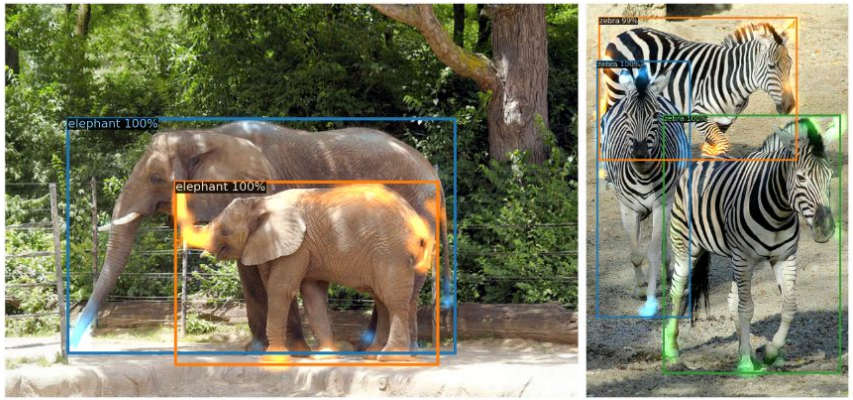
Auxiliary losses after each decoding layer and thus can predict output using every decoder layer

Every layer improves AP and AP_{50}

Non-max-suppression improves for early layers, but improvement diminishes

Decoder attention is local





Ablations - Positional Encodings

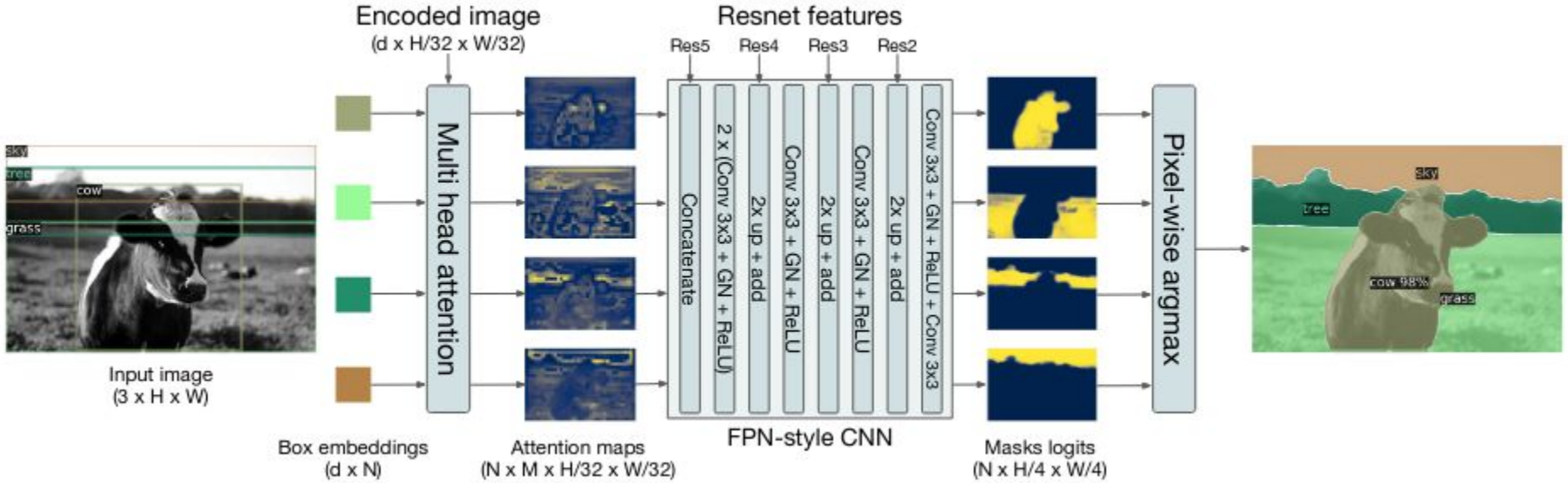
Surprising that removing spatial encoding in encoder only leads to AP drop of 1.3

spatial pos. enc. encoder	spatial pos. enc. decoder	output pos. enc. decoder	AP	Δ	AP_{50}	Δ
none	none	learned at input	32.8	-7.8	55.2	-6.5
sine at input	sine at input	learned at input	39.2	-1.4	60.0	-1.6
learned at attn.	learned at attn.	learned at attn.	39.6	-1.0	60.7	-0.9
none	sine at attn.	learned at attn.	39.3	-1.3	60.3	-1.4
sine at attn.	sine at attn.	learned at attn.	**40.6**	-	**61.6**	-

Extension - Panoptic Segmentation

Add a binary mask head on top of the decoder outputs

Computes multihead attention, the attention heatmap and then FPN



Result

- Outperform strong PanopticFPN baseline
- Dominant on “stuff” classes but lack behind in “things” classes

Table 5: Comparison with the state-of-the-art methods UPSNet [51] and Panoptic FPN [18] on the COCO `val` dataset We retrained PanopticFPN with the same data-augmentation as DETR, on a 18x schedule for fair comparison. UPSNet uses the `1x` schedule, UPSNet-M is the version with multiscale test-time augmentations.

Model	Backbone	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}	AP
PanopticFPN++	R50	42.4	79.3	51.6	49.2	82.4	58.8	32.3	74.8	40.6	37.7
UPSnet	R50	42.5	78.0	52.5	48.6	79.4	59.6	33.4	75.9	41.7	34.3
UPSnet-M	R50	43.0	79.1	52.8	48.9	79.7	59.7	34.1	78.2	42.3	34.3
PanopticFPN++	R101	44.1	79.5	53.3	**51.0**	**83.2**	60.6	33.6	74.0	42.1	**39.7**
DETR	R50	43.4	79.3	53.8	48.2	79.8	59.5	36.3	78.5	45.3	31.1
DETR-DC5	R50	44.6	79.8	55.0	49.4	80.5	60.6	**37.3**	**78.7**	**46.5**	31.9
DETR-R101	R101	**45.1**	**79.9**	**55.5**	50.5	80.9	**61.7**	37.0	78.5	46.0	33.0

Discussions

Why does the model perform worse on smaller objects and better at larger objects?

Discussions

Why does the model perform worse on smaller objects and better at larger objects?

Cannot use multi-scale features (Feature Pyramid Network) commonly used to improve detection of small objects