End-to-End Object Detection with Transformers

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

October 9, 2020

Overview

Introduction

DETR Network

Optimization

Experiments on object detection

Experiments on panoptic segmentation

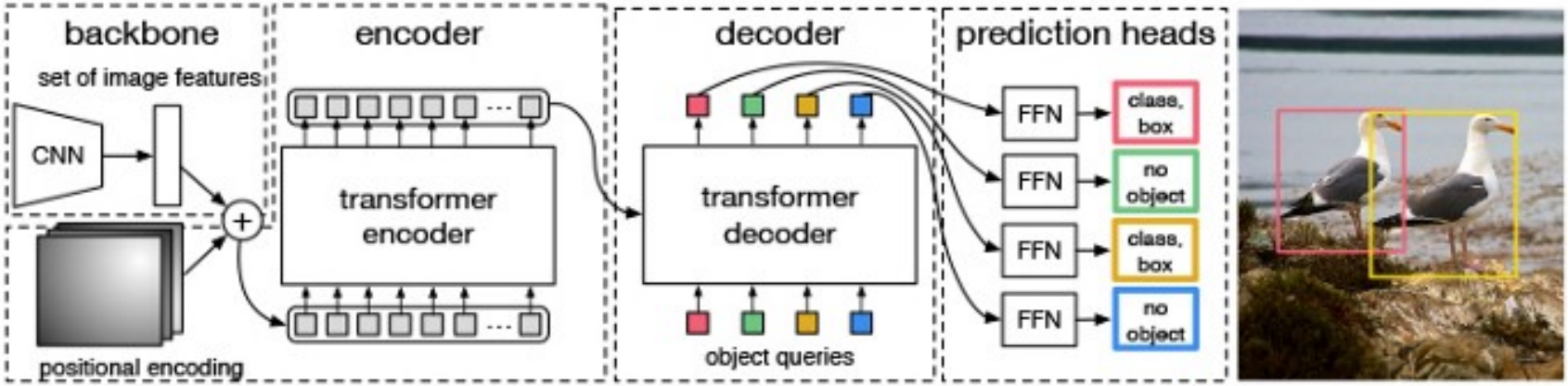
Introduction

Transformers are a deep learning architecture that has gained popularity in recent years. They rely on a simple yet powerful mechanism called attention, which enables AI models to selectively focus on certain parts of their input and thus reason more effectively. Transformers have been widely applied on problems with sequential data, in particular in natural language processing (NLP) tasks such as language modeling and machine translation, and have also been extended to tasks as diverse as speech recognition, symbolic mathematics, and reinforcement learning. But, perhaps surprisingly, computer vision has not yet been swept up by the Transformer revolution.

This paper presents a new method that views object detection as a direct set prediction problem. The approach streamlines the detection pipeline, effectively removing the need for many hand-designed components like a non-maximum suppression procedure or anchor generation that explicitly encode our prior knowledge about the task. The main ingredients of the new framework, called DE*tection* TR*ansformer* (DETR), are a set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture.

DETR Network

Unlike traditional computer vision techniques, DETR casts the object detection task as an image-to-set problem. Given an image, the model must predict an unordered set (or list) of all the objects present, each represented by its class, along with a tight bounding box surrounding each one.



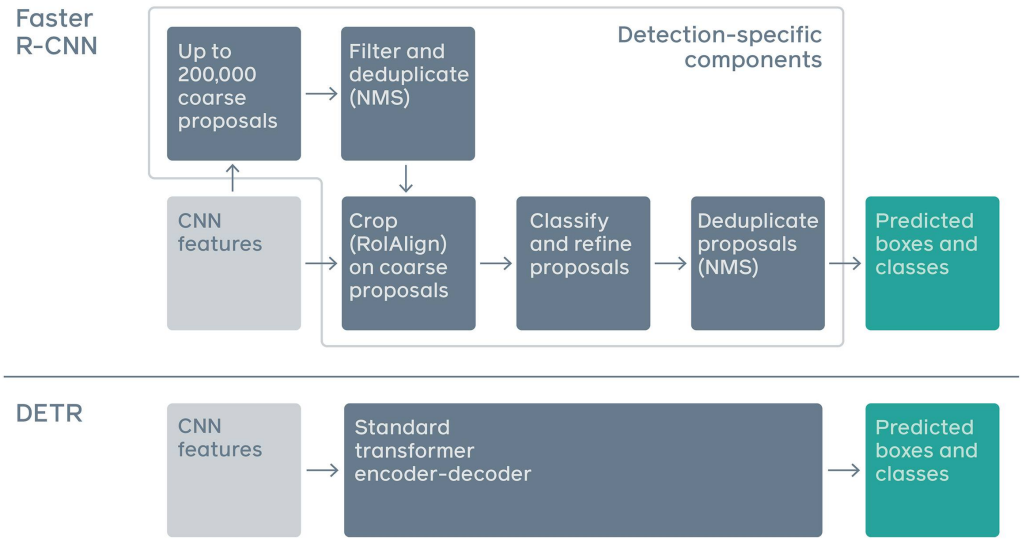
DETR Network

Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. The new model is conceptually simple and does not require a specialized library, unlike many other modern detectors.

DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster RCNN baseline on the challenging COCO object detection dataset. Moreover, DETR can be easily generalized to produce panoptic segmentation in a unified manner. It is shown that DETR significantly outperforms competitive baselines.

Comparison with existing methods

Traditional computer vision models typically use a complex, partly handcrafted pipeline that relies on custom layers in order to localize objects in an image and then extract features. DETR replaces this with a simpler neural network that offers a true end-to-end deep learning solution to the problem.



1

[1] Figures are copied from https://ai.facebook.com/blog/end-to-end-object-detection-with-transformers/

CNN feature extraction

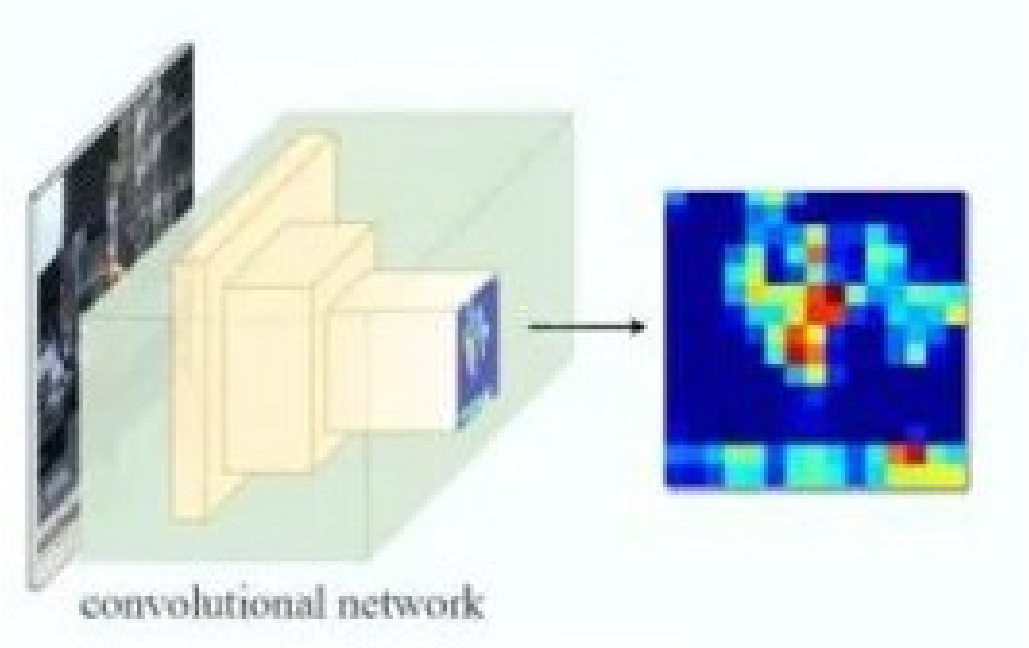


Figure 1: We first use a Convolutional Neural Network (CNN) to automatically extract the important characteristics of the image ,we extract the characteristics that describe the local pattern of the image through the convolution kernel ,CNN basically gives us this thing which is sort of a higher level representation.

Flatten feature maps

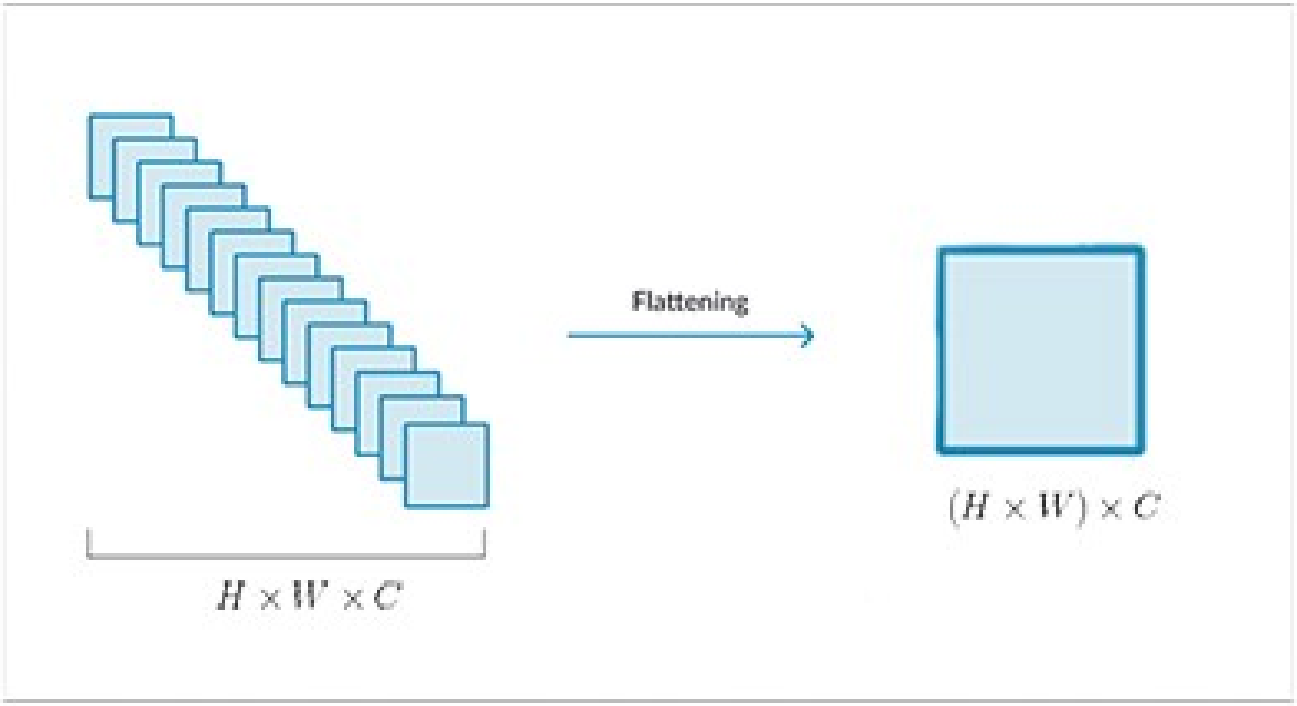


Figure 2: This set of feature maps is flattened because the transformer is naturally a sequence processing model, so it only takes a sequence of vectors as input. We are going to unroll and flatten that into one sequence of C dimensional feature vectors that you then put into Transformer encoder .

Transformer encoder

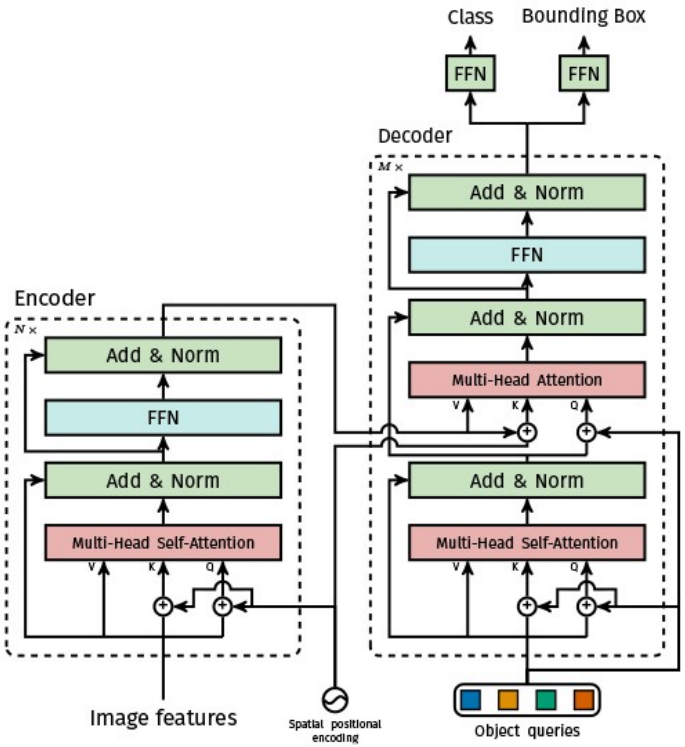


Figure 3: Encoder have attention layers that can attend from each position to each position in a one-shot manner, so as it transforms this representation up the transformer layers at each step it can basically aggregate information from every where in the sequence to any where else and therefore it is very powerful if you have a sequence and you need sort of global connections across the sequence.

Transformer encoder output

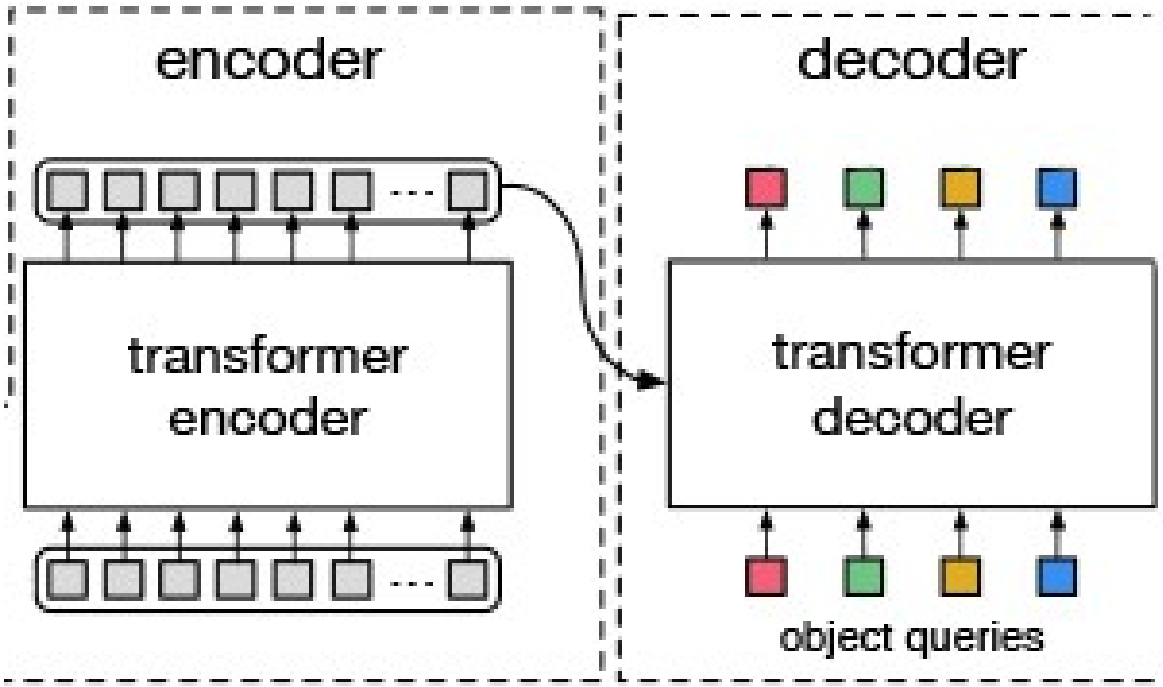


Figure 4: The encoder will now transform this sequence into an equally length and sized sequence of features that goes as input into Transformer decoder.

Transformer decoder

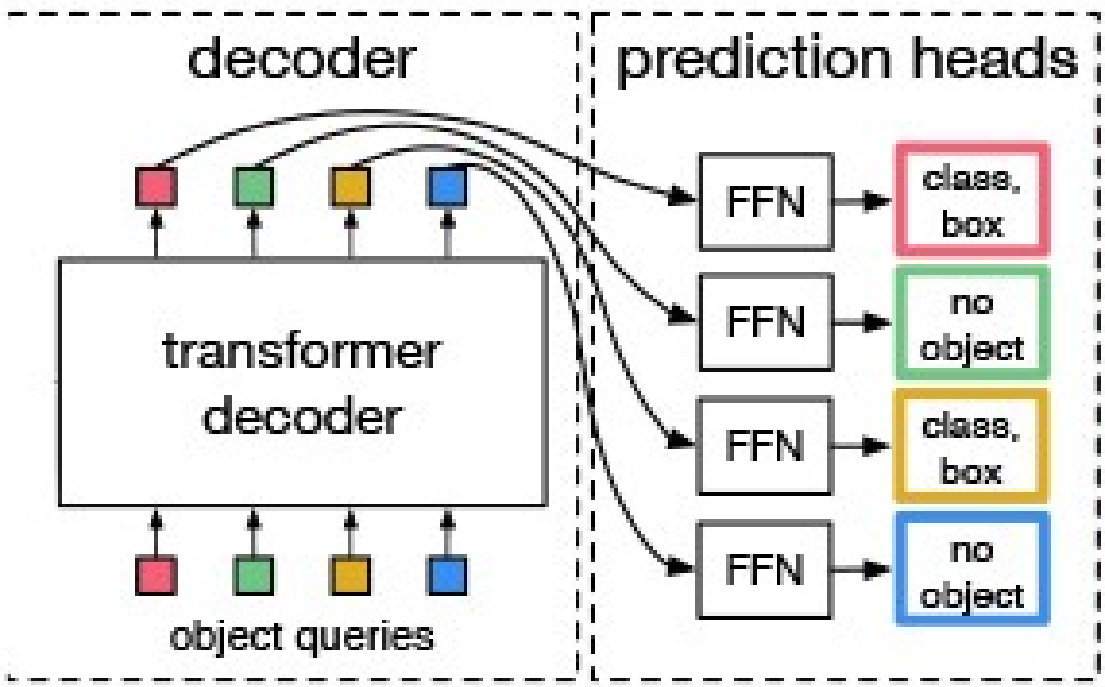
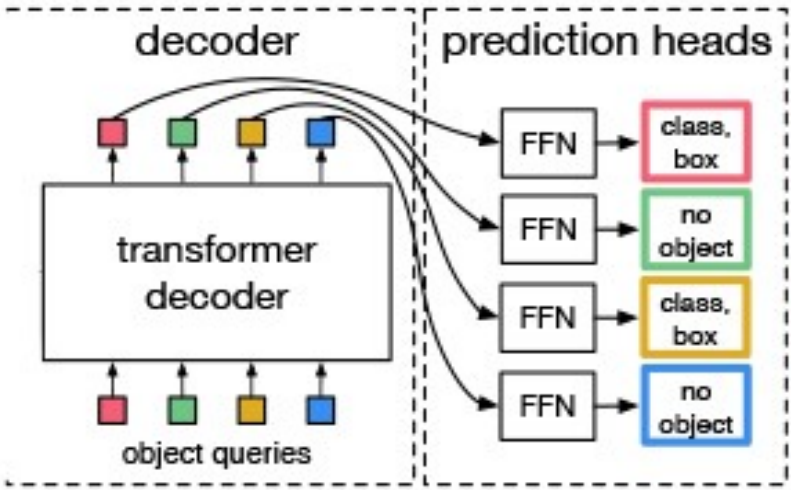


Figure 5: The decoder take a sequence as input called Object Queries, you start with a sequence of N objects and the decoder will output a sequence of N objects that will end up directly through a classifier that outputs the class label and bounding box either defining an object or saying that there is not an object.

Transformer decoder input — object queries

These Object Queries is like you have different people that can ask the input image different questions. In order to get a difference in bounding box predictions, one trains N different Object Queries to ask different questions of the input image. And these questions are exactly what an attention mechanism to a part of the image features. You basically learn the Queries in a way such that across the data set they all to get cover every possible image pretty well. Each one is interested in different objects and different regions in the image, and each one of these Queries is going to output their best guess based on what they are interested in.

Optimization objective



Aligning the prediction set and labels set is a bipartite graph matching problem. It uses the Hungarian algorithm as the solution method, and defines the minimum matching strategy as follows

$$\hat{\sigma} = \arg\min_{\sigma \in \mathcal{E}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}), \tag{1}$$

$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}). \tag{2}$$

Box loss

The most commonly-used $L1$ loss will have different scales for small and large boxes even if their relative errors are similar. To mitigate this issue we use a linear combination of the $L1$ loss and the generalized IoU loss [1] $L_{\text{iou}}(,)$ that is scale-invariant.

$$\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{\text{iou}}\mathcal{L}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}}||b_i - \hat{b}_{\sigma(i)}||_1, \lambda_{\text{iou}}, \lambda_{\text{L1}} \in \mathbb{R}. \quad (3)$$

These two losses are normalized by the number of objects inside the batch. The final optimization loss is

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-log\, \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right]. \quad (4)$$

Experiment setup

The paper shows that DETR achieves competitive results compared to Faster R-CNN in quantitative evaluation on COCO. Then, it provides a detailed ablation study of the architecture and loss, with insights and qualitative results. Finally, to show that DETR is a versatile and extensible model, it presents results on panoptic segmentation, training only a small extension on a fixed DETR model.

COCO 2017 detection and panoptic segmentation datasets. [2], [3] The dataset contains 118k training images and 5k validation images. Each image is annotated with bounding boxes and panoptic segmentation. There are 7 instances per image on average, up to 63 instances in a single image in training set, ranging from small to large on the same images.

Metrics If not specified, we report AP as bbox AP, the integral metric over multiple thresholds. For comparison with Faster R-CNN we report validation AP at the last training epoch, for ablations we report median over validation results from the last 10 epochs.

Experiment setup

Training details We report results with two different backbones: a *ResNet-50* and a *ResNet-101*. The corresponding models are called respectively **DETR** and **DETR-R101**.

Following [4], we also increase the feature resolution by adding a *dilation* to the last stage of the backbone and removing a stride from the first convolution of this stage. The corresponding models are called respectively **DETR-DC5** and **DETR-DC5-R101** (dilated C5 stage). This modification increases the resolution by a factor of two, thus improving performance for small objects, at the cost of a 16x higher cost in the self-attentions of the encoder, leading to an overall 2x increase in computational cost.

Comparison with Fast RCNN

Table 1: Comparison with Faster R-CNN with a ResNet-50 and ResNet-101 backbones on the COCO validation set. The top section shows results for Faster R-CNN models in Detectron2 [50], the middle section shows results for Faster R-CNN models with GIoU [38], random crops train-time augmentation, and the long `9x` training schedule. DETR models achieve comparable results to heavily tuned Faster R-CNN baselines, having lower AP_S but greatly improved AP_L. We use torchscript Faster R-CNN and DETR models to measure FLOPS and FPS. Results without R101 in the name correspond to ResNet-50.

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**

Ablation study on number of encoder layers

For the study, ResNet-50-based DETR model with 6 encoder, 6 decoder layers and width 256 is used.

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

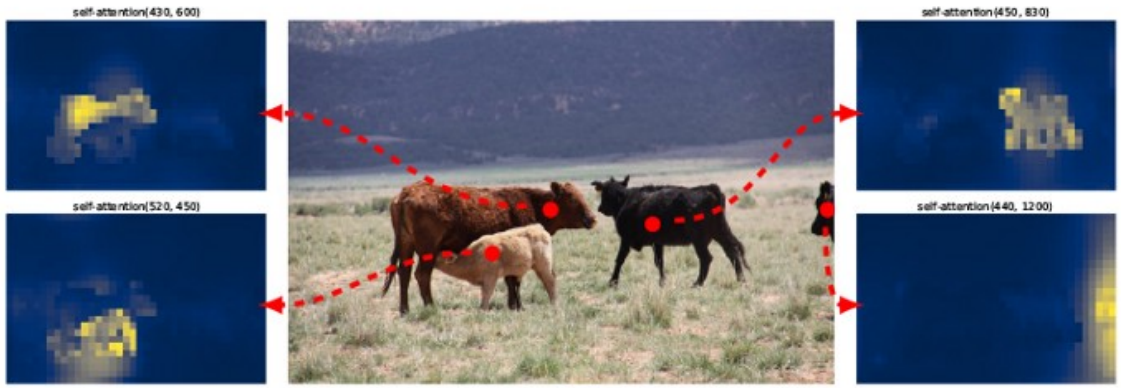


Figure 6: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

Ablation study on number of decoder layers

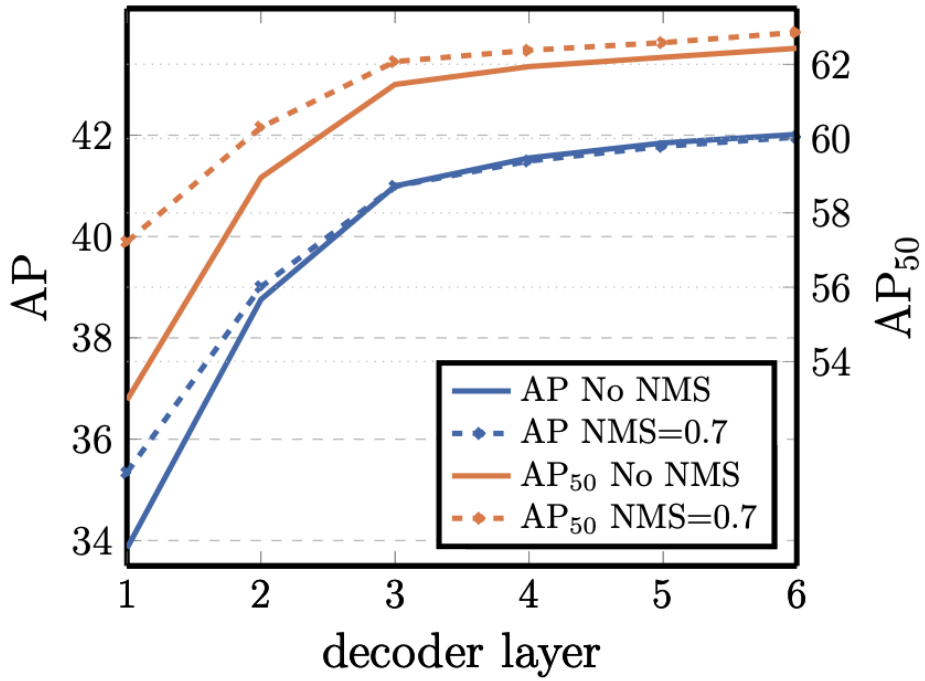


Figure 7: *AP* and AP_{50} performance after each decoder layer. A single long schedule baseline model is evaluated. Both AP and AP50 improve after every layer, totalling into a very significant +8.2/9.5 AP improvement between the first and the last layer. DETR does not need NMS by design, which is validated by this figure. NMS lowers AP in the final layers, removing TP predictions, but improves AP in the first decoder layers, removing double predictions, as there is no communication in the first layer, and slightly improves AP_{50}.

Importance of FFN

FFN inside tranformers can be seen as 1×1 convolutional layers, making encoder similar to attention augmented convolutional networks [3]. We attempt to remove it completely leaving only attention in the transformer layers. By reducing the number of network parameters from 41.3M to 28.7M, leaving only 10.8M in the transformer, performance drops by 2.3 AP, we thus conclude that FFN are important for achieving good results.

Importance of positional encodings

There are two kinds of positional encodings in our model: spatial positional encodings and output positional encodings (object queries).

Table 3: Results for different positional encodings compared to the baseline (last row), which has fixed sine pos. encodings passed at every attention layer in both the encoder and the decoder. Learned embeddings are shared between all layers. Not using spatial positional encodings leads to a significant drop in AP. Interestingly, passing them in decoder only leads to a minor AP drop. All these models use learned output positional encodings.

spatial pos. enc.		output pos. enc.				
encoder	decoder	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**	-

Loss ablations

There are three components to the loss: classification loss, $L1$ bounding box distance loss, and GIoU loss. The classification loss is essential for training and cannot be turned off, so we turn off $L1$ bounding box distance loss and/or GIoU loss.

Table 4: Effect of loss components on AP. We train two models turning off ℓ_1 loss, and GIoU loss, and observe that ℓ_1 gives poor results on its own, but when combined with GIoU improves AP_M and AP_L. Our baseline (last row) combines both losses.

class	ℓ_1	GIoU	AP	Δ	AP_{50}	Δ	AP_S	AP_M	AP_L
✓	✓		35.8	-4.8	57.3	-4.4	13.7	39.8	57.9
✓		✓	39.9	-0.7	**61.6**	0	**19.9**	43.2	57.9
✓	✓	✓	**40.6**	-	**61.6**	-	**19.9**	**44.3**	**60.2**

Decoder output slot analysis

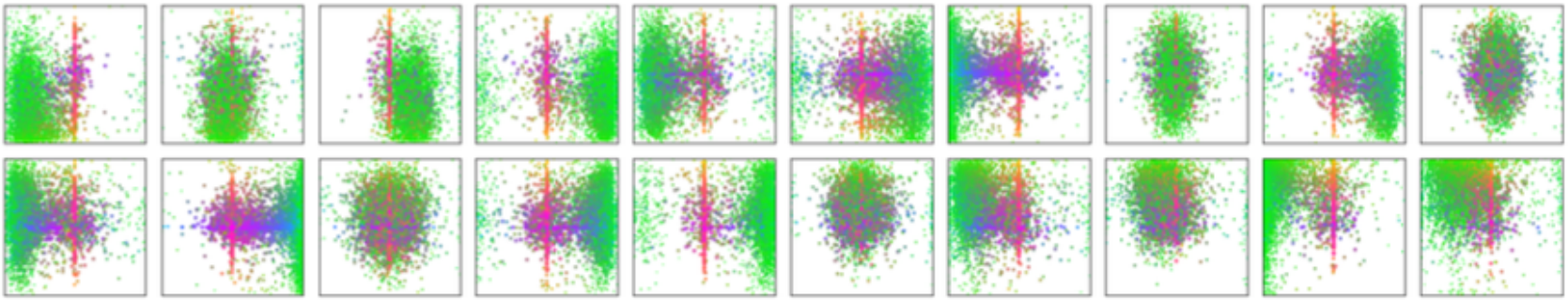


Fig. 7: Visualization of all box predictions on all images from COCO 2017 val set for 20 out of total $N = 100$ prediction slots in DETR decoder. Each box prediction is represented as a point with the coordinates of its center in the 1-by-1 square normalized by each image size. The points are color-coded so that green color corresponds to small boxes, red to large horizontal boxes and blue to large vertical boxes. We observe that each slot learns to specialize on certain areas and box sizes with several operating modes. We note that almost all slots have a mode of predicting large image-wide boxes that are common in COCO dataset.

Generalization to unseen numbers of instances

Some classes in COCO are not well represented with many instances of the same class in the same image. For example, there is no image with more than 13 giraffes in the training set. The author create a synthetic image to verify the generalization ability of DETR.

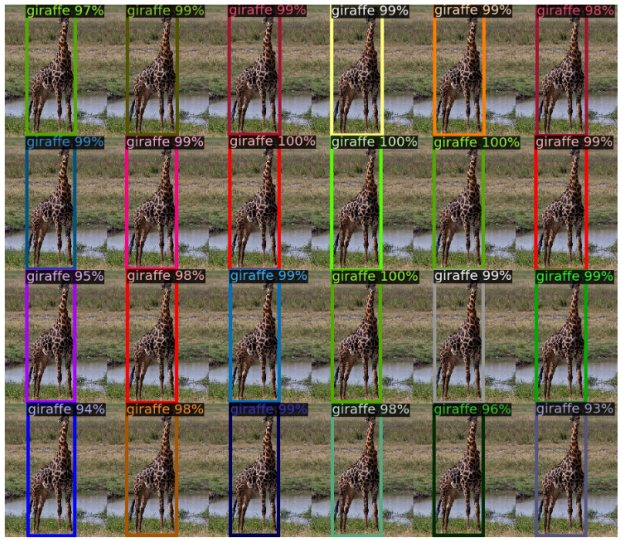


Figure 8: Out of distribution generation for rare classes. Even through no image in the training set has more than 13 giraffes, DETR has no difficulty generalizing to 24 and more instances of ghe same class.

DETR for panoptic segmentation

Panoptic segmentation [3] has recently attracted a lot of attention from the computer vision community. Similarly to the extension of Faster R-CNN [37] to Mask R-CNN [14], DETR is shown to be naturally extended by adding a mask head on top of the decoder outputs. Experiments are performed on the panoptic annotations of the COCO dataset that has 53 stuff categories in addition to 80 things categories.

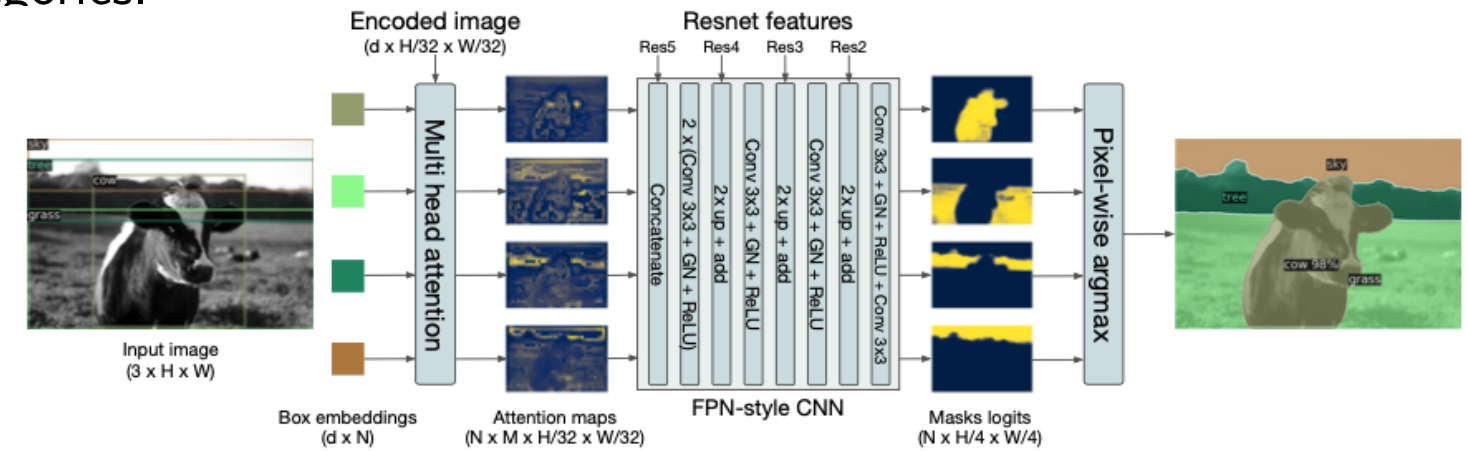


Fig. 8: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.

DETR for panoptic segmentation

A binary mask is generated in parallel for each object detected. They use the attention maps of the instances, then they scale it and then they simply classify each pixel, finally they merge everything into a single prediction.

Fig. 9: Qualitative results for panoptic segmentation generated by DETR-R101. DETR produces aligned mask predictions in a unified manner for things and stuff.

DETR for panoptic segmentation

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

The end.

Thanks!

Bibliography

[1] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 658–666.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[3] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 9404–9413.

[4] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2359–2367.