DETR：2020

End-to-End Object Detection with Transformers

code:https://github.com/facebookresearch/detr

目标检测术语：https://hotelll.github.io/2021/03/27/%E7%9B%AE%E6%A0%87%E6%A3%80%E6%B5%8B%E5%85%A5%E9%97%A8%E2%80%94%E2%80%94%E6%9C%AF%E8%AF%AD%E7%AF%87/

论文背景

目标检测的目标是为每个感兴趣的对象预测一组边界框和类别标签，现代的检测头通过在一组建议区域，锚框，或者窗口中心间接完成这个任务

论文创新

传统目标检测的性能取受后处理影响比较大，我们提出了一种直接集合预测的方法绕过代理任务。

二部匹配损失和非自回归编码的transformer

论文方法

自注意力机制较好模拟了序列元素之间的成对相互作用

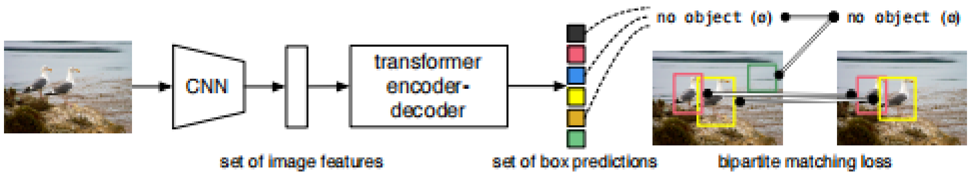


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "no object" (∅) class prediction.

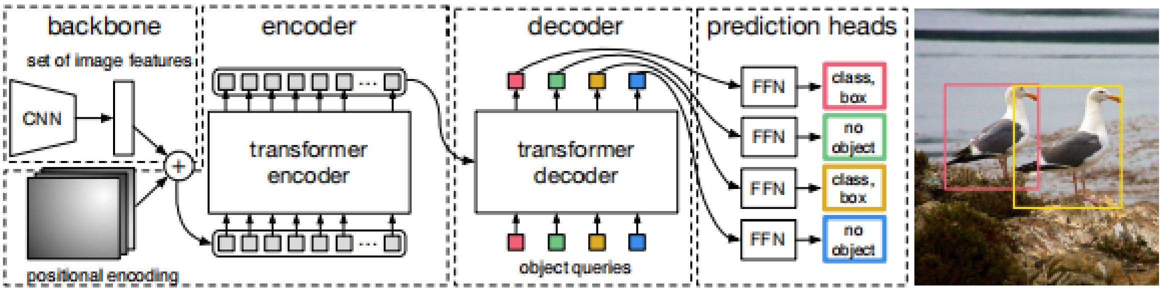
文章方法建立在集合预测的二部匹配损失、transformer的编解码结构、并行解码、目标检测方法基础之上

集合预测：

在所有情况下，对预测结果的某一排列来说，损失函数应是不变的）。通常的解决方案是在匈牙利算法的基础上设计一个损失，以找到真实值和预测值之间的二部匹配。这就强制了置换不变性，并保证每个目标元素都有惟一的匹配。我们采用二部匹配损失。

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

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



CNN提取特征，编码器-解码器 FFN进行最终的检测预测

实际效果

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**

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

解码器的注意区域是局部的，在编码器通过全局注意分离出实例后，解码器只需要注意到末端就可以提取出类和对象的边界。

FFN对于获得良好的结果很重要

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

个人理解