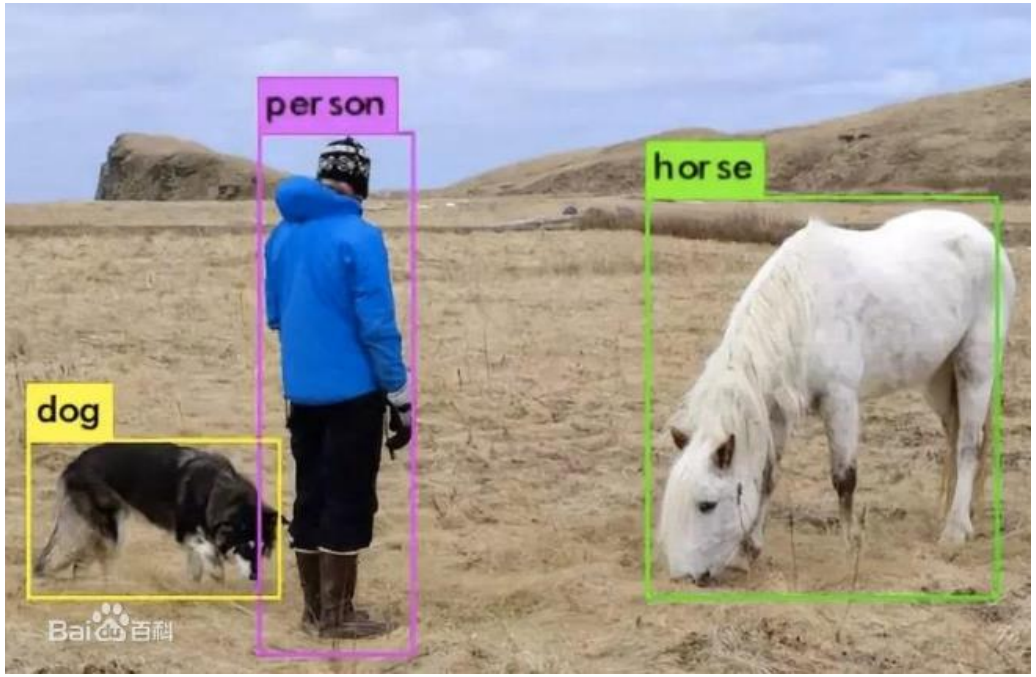
Detection Transformer

Background

目标检测的任务是要去检测一张图片里一系列的Bounding Box的坐标以及对应的Label。现代大多数检测器通过定义一些proposal、anchor或者windows，把问题构建成为一个分类和回归问题来间接地完成这个任务。



DETR

Detection Transformer = Object Detection + Transformer

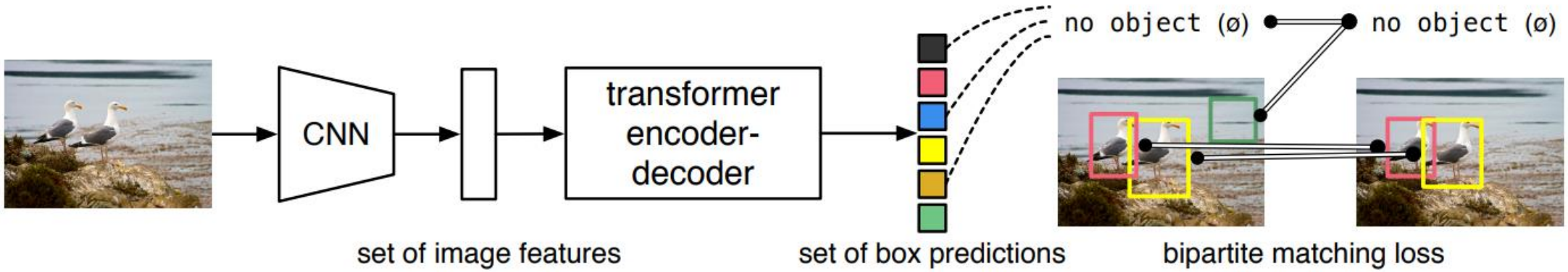


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" ($\varnothing$) class prediction.

Conclusion

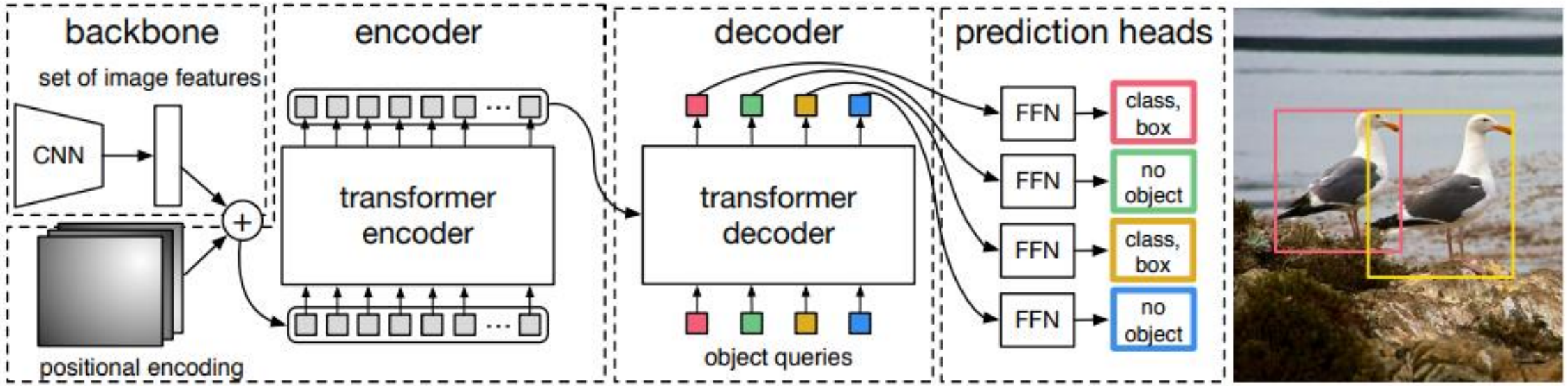
DETR所做的工作，是将Transformers运用到了Object Detection任务，将CNN和Transformer相结合，取代了现在的模型需要手工设计的工作，并且取得了不错的结果。

在Object Detection任务上DETR准确率和运行时间上和Faster R-CNN相当；将模型扩展到全景分割的任务上，DETR表现甚至还超过了其他的baseline。

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 Model

1.用Transformer的Encoder-Decoder架构一次性生成 N 个box prediction。其中 N 是一个事先设定的、远远大于image中object个数的一个整数。



DETR Model

1. Backbone: Take features
2. Encoder: Image embedding
3. Decoder: Set prediction
4. FFN: Class and Bounding box prediction

Backbone

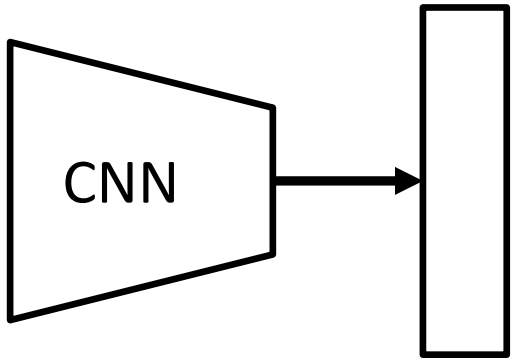
CNN从原始图片中提取图像特征。即，原始图片 $x_{image} \in R^{3\times H_0\times W_0}$，通过CNN卷积得到特征图谱 $f \in R^{C\times H\times W}$。其中 $C = 2048$，$H = \frac{H_0}{32}$，$W = \frac{W_0}{32}$。

Backbone模块的作用就是从原始图像中抽取图像特征因为Transformer结构无法直接读入一张图片，所以需要Backbone模块提取feature map。

在DETR代码中作者使用了ResNet50和ResNet101分别进行图像特征抽取。

```
class DETR(nn.Module):

    def __init__(self, num_classes, hidden_dim, nheads,
                 num_encoder_layers, num_decoder_layers):
        super().__init__()
        # We take only convolutional layers from ResNet-50 model
        self.backbone = nn.Sequential(*list(resnet50(pretrained=True).children())[:-2])
        self.conv = nn.Conv2d(2048, hidden_dim, 1)
        self.transformer = nn.Transformer(hidden_dim, nheads,
                                          num_encoder_layers, num_decoder_layers)
        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
        self.linear_bbox = nn.Linear(hidden_dim, 4)
        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))
        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
```



set of images features

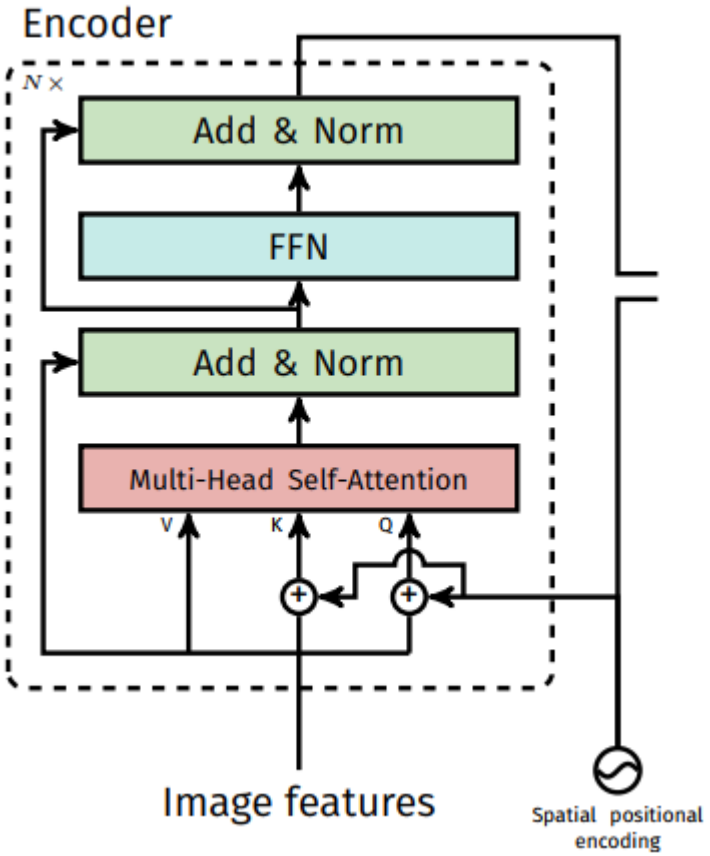
Encoder

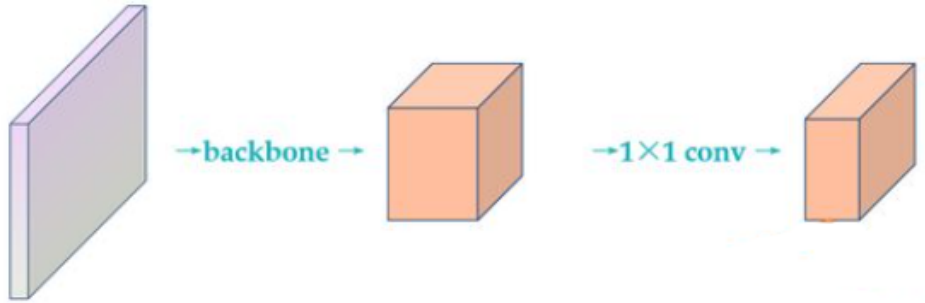
首先，使用1x1 卷积将特征图谱 f 的通道维度从 C 减少到更小的维度 d (typically d=256)。 创建一个新的特征图$z_0 \in R^{d\times H\times W}$。

由于Transformer需要序列化的输入，所以我们将z_0的维度变为$(d \times HW)$大小的特征图。

每一层Encoder都有一个标准架构，一个多头自注意力模块和一个前向神经网络构成。

与标准Transformer中的Encoder不同的是我们不会只在序列输入进第一层的Encoder时加入位置编码，而是在每各注意力层的输入中加入固定的位置编码。



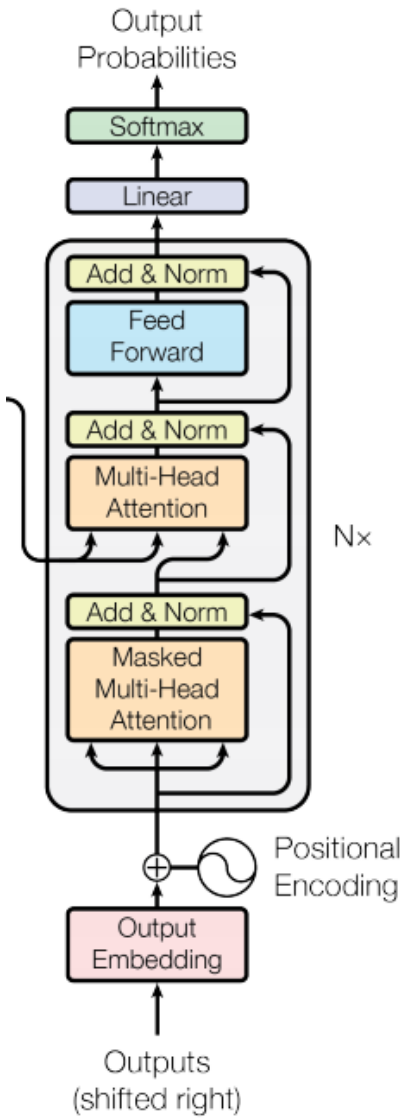


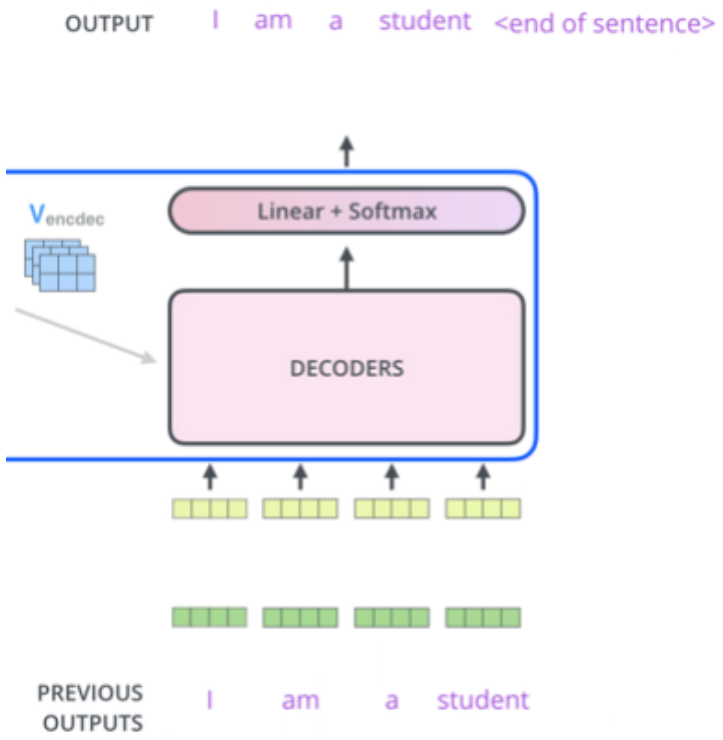
$(3 \times H_0 \times W_0)$ $(C \times H \times W)$ $(d \times H \times W)$

Typical Decoder

原版Transformer的输出是一个output probability代表我们每次经过decoder只能softmax一个输出。这种Auto-Regressive的产生方式并行度差消耗时间长。

过程表达为：predicts the output sequence one element at a time。

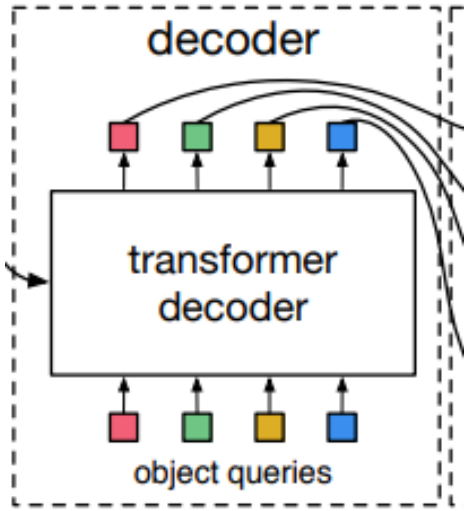


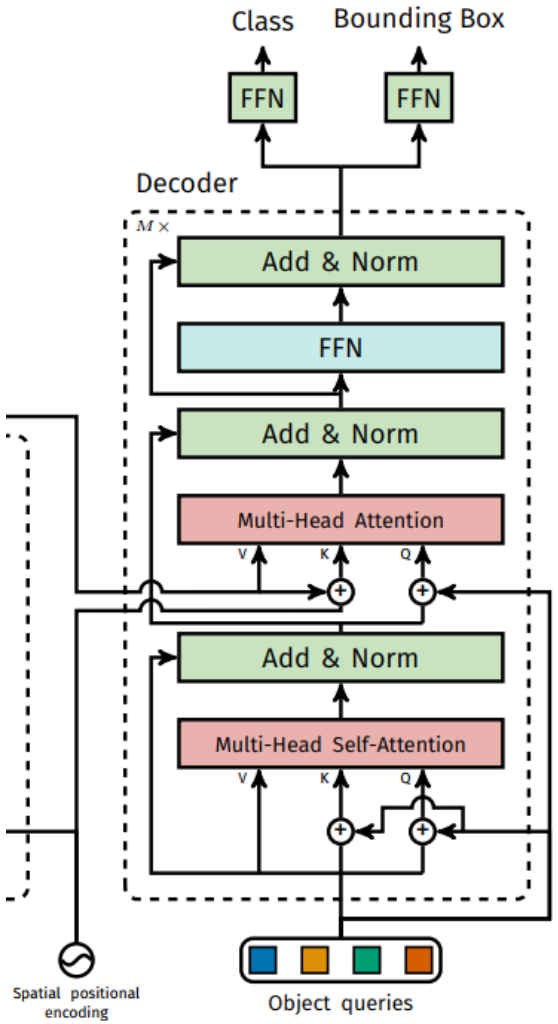


Decoder in DETR

DETR通过一次性处理输入直接输出所有的预测(set prediction)过程表达为：decodes the N objects in parallel at each decoder layer

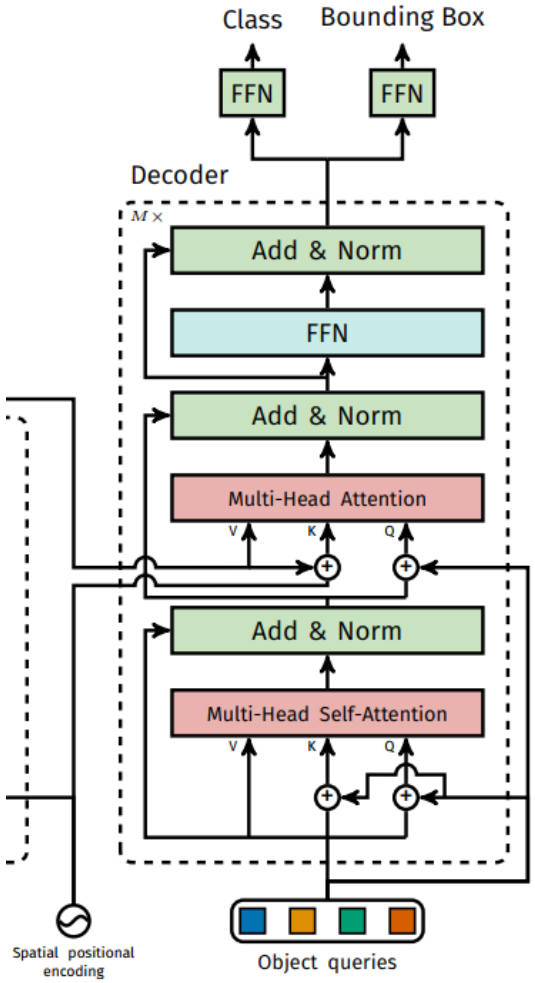
其中我们设定N类目标(N的值远大于Image中各物体的种类数)，通过Decoder学习这N类目标的参数(object queries)。





Object queries

Object queries是一个$(N \times 256)$的可学习的参数矩阵，Object queries矩阵内部通过学习建模了N个物体之间的全局关系。例如教室里面的桌子旁边(A类)一般是放椅子(B类)，而不会是放一台烤箱(C类)，那么在解码的时候就可以利用该全局注意力更好的进行解码预测输出。其中因为N远大于Image中各类物体的种类数所以Object queries矩阵中不同的类可以指代同一类别，即可以是N_6、N_{50}、 N_{77}可以共同指代一种类别。



Positional encoding

DETR中含有两个位置编码：Spatial positional encoding 、object queries.

Encoder中每个layer的位置编码	Query	Key
第一个multi-head self attention	Positional encoding	Positional encoding
Encoder中每个layer的位置编码	Query	Key
第一个multi-head self attention	Object queries	Object queries
第二个multi-head self attention	Object queries	Positional encoding

我们将一张图片各点看作如下矩阵：

$$\begin{bmatrix} H_1,W_1 & & H_1,W_n \\ H_2,W_1 & \cdots & H_2,W_n \\ H_3,W_1 & & H_3,W_n \\ \vdots & \ddots & \vdots \\ H_m,W_1 & \cdots & H_n,W_n \end{bmatrix}$$

对各点使用sincos进行位置编码：

a. $PE_{(pos_x,2i)} = \sin(pos_x/10000^{2i/128})$

b. $PE_{(pos_x,2i+1)} = \cos(pos_x/10000^{2i/128})$

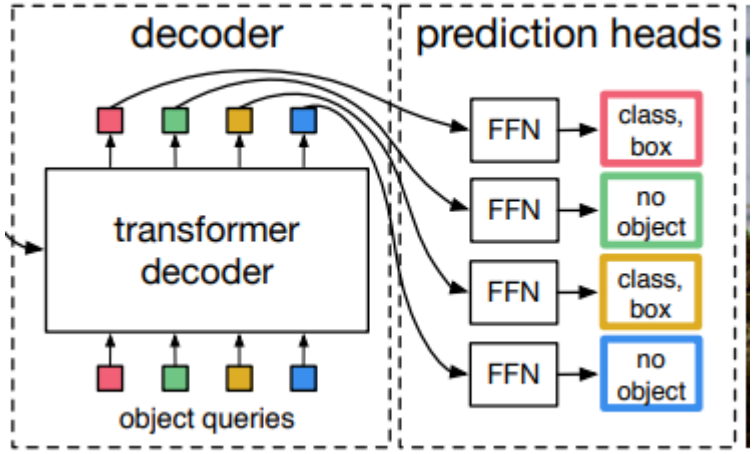
c. $PE_{(pos_y,2i)} = \sin(pos_y/10000^{2i/128})$

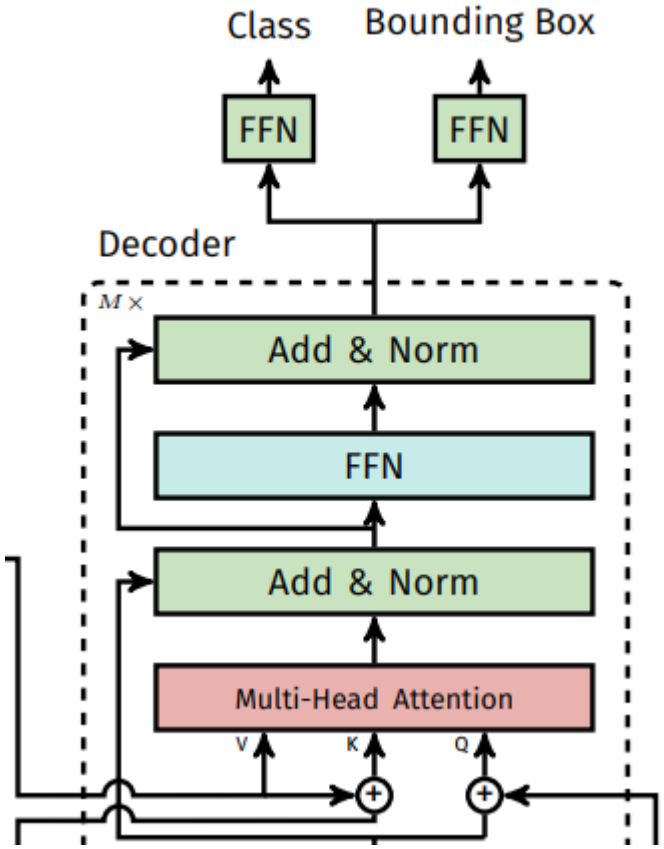
d. $PE_{(pos_y,2i+1)} = \cos(pos_y/10000^{2i/128})$

FFN

DETR中含有三个FNN层：

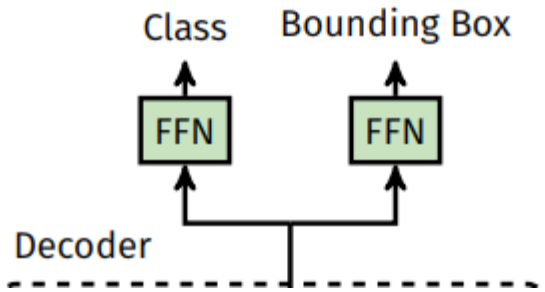
- 对Multi – head self – attention计算结果进行拟合
- 对物体类别进行预测
- 对各类别的锚框





Loss

通过两个FNN层我们分别得到图片里的分类分支class和回归分支bounding box，其中DETR输出的张量维度分别为(N,class+1)和(N,4)。(class+1所表示的是图片中已知的图片内物体的类别的个数和没有标注类别的类。4表示预测类的中心点、高和宽(c_x, c_y, h, w))。



DETR的输出是一个无序的集合，如何确定每一个框所对应的类？

Hungarian Algorithm

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

其中σ表示真值索引到预测值的映射，
再通过L_{match}函数求解真值与预测值的
损失，最小化损失就是我们判别的依据。

$$L_{match} = -1_{\{C_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(C_i) + 1_{\{C_i \neq \emptyset\}} L_{box}\left(b_i, \hat{b}_{\sigma(i)}\right)$$

利用上方式子求解最小损失

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

$L_{Hungarian}$是所有L_{match}的最小排列的组合，即对于图片中的每个真值i，$\hat{\sigma}(i)$就是需要找的对应的预测值的索引

Box loss Similarly to [41,36], we use a soft version of Intersection over Union in our loss, together with a ℓ_1 loss on $\hat{b}$:

$$\mathcal{L}_{\text{box}}(b_{\sigma(i)}, \hat{b}_i) = \lambda_{\text{iou}} \mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) + \lambda_{\text{L1}} ||b_{\sigma(i)} - \hat{b}_i||_1 , \quad (9)$$

where $\lambda_{\text{iou}}, \lambda_{\text{L1}} \in \mathbb{R}$ are hyperparameters and $\mathcal{L}_{\text{iou}}(\cdot)$ is the generalized IoU [38]:

$$\mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) = 1 - \left(\frac{|b_{\sigma(i)} \cap \hat{b}_i|}{|b_{\sigma(i)} \cup \hat{b}_i|} - \frac{|B(b_{\sigma(i)}, \hat{b}_i) \setminus b_{\sigma(i)} \cup \hat{b}_i|}{|B(b_{\sigma(i)}, \hat{b}_i)|} \right). \quad (10)$$

Characteristic

- End to End任务模式（原始图片输入无需处理直接做出预测无需人工设计）
- 设计了bipartite matching loss，基于预测的box和ground truth boxes的二分图匹配计算loss的大小，从而使得预测的box的位置和类别更接近于ground truth。
- 用Transformer的encoder-decoder架构一次性生成 N 个box prediction。其中 N 是一个事先设定的、比远远大于image中object个数的一个整数。