规定论文题目2：End-to-End ObjectDetection with Transformers

论文动机

目标检测的目标是预测一组边界框（bounding boxes）及其对应的标签。目前的大多数检测器并不是直接预测目标的集合，而是通过回归和分类处理大量的提议框（proposals）、锚框（anchors）或窗口中心。这样的方法会受到一系列问题的影响，例如如何通过后处理来消除大量重叠的预测、如何设计锚框、以及如何将目标框与锚框关联起来。

在这篇论文之前就有一些工作提出了比较好的思想。例如，2015年一篇关于人体检测的工作，其思路与DETR非常相似。此外，2018年ECCV Workshop上发表的一篇文章《Recurrent Neural Networks for Semantic Instance Segmentation》也展示了类似的创新。

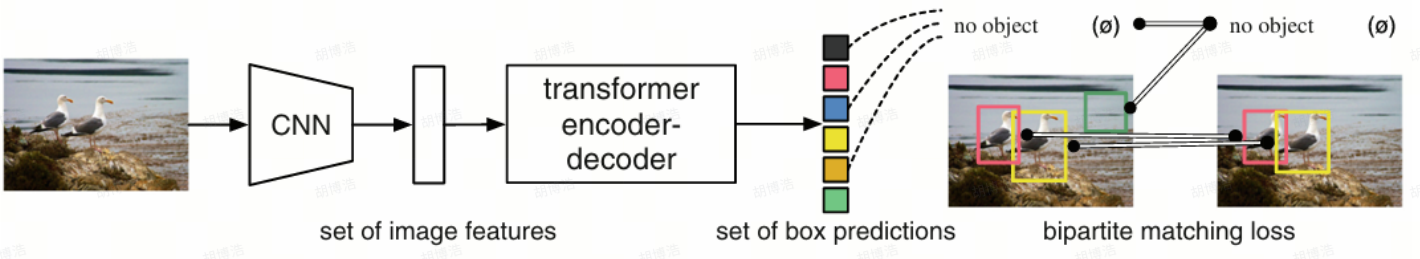
这篇文章提出了一种使用RNN（循环神经网络）和匈牙利算法进行实例分割的方法，采用了编码器-解码器架构，顺序生成图像中每个对象的二进制掩码和分类标签。具体过程如下：

1. **编码器-解码器架构**：该架构通过连接所有ConvLSTM层的侧输出，并应用每通道最大池化操作，以获得隐藏表示。
2. **预测分类标签和停止概率**：这个隐藏表示作为两个全连接层的输入，分别预测分类标签和停止概率。
3. **生成分割掩码**：模型逐个生成分割掩码，通过匈牙利算法将预测结果和ground-truth进行一一匹配，用于计算损失。

这种方法与DETR有着相似的核心思想，即通过端到端的方法简化目标检测和实例分割的流程，同时利用匈牙利算法进行结果匹配，提升了模型的精度和效率。

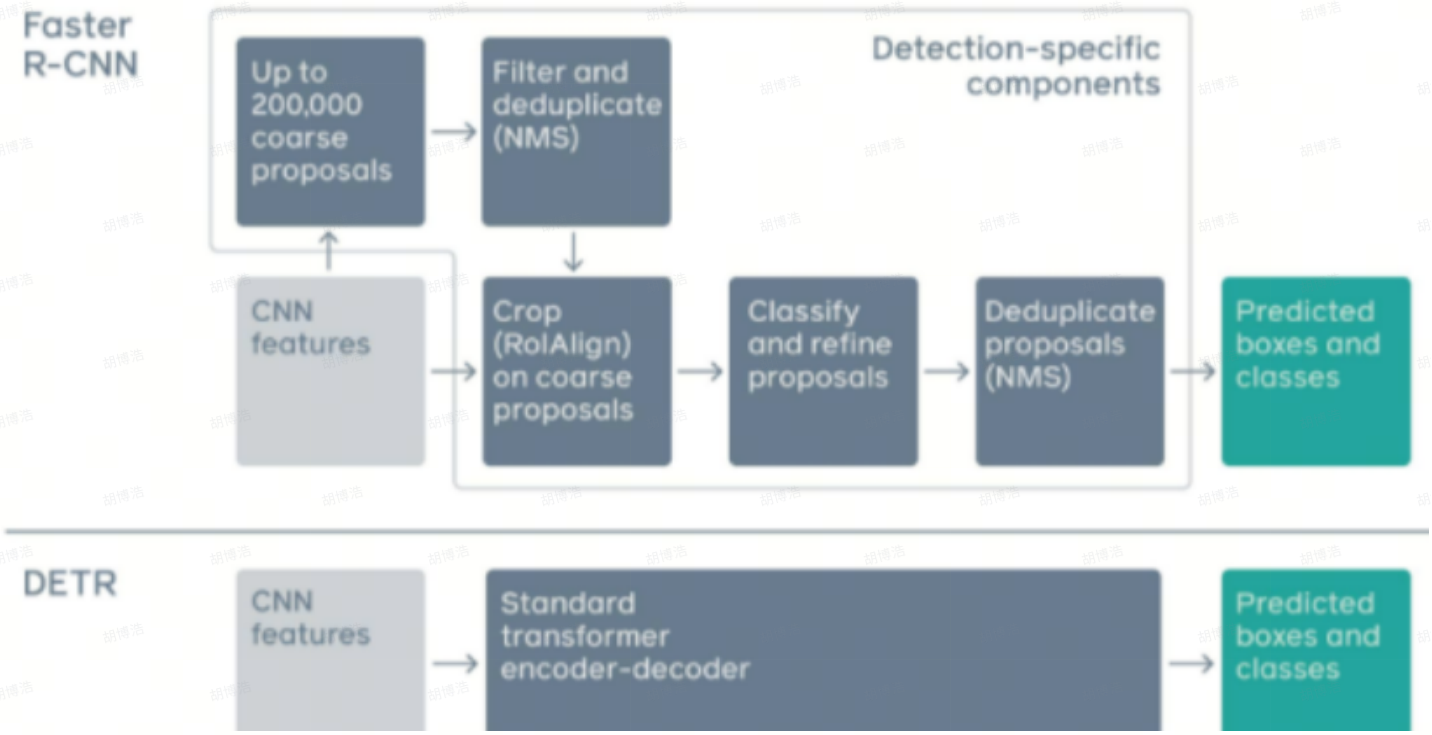
Transformer 比 RNN 更好的一个重要原因是它实现了并行化，不需要依赖之前的结果。因此，将目标检测任务转变为一个集合预测（Set Prediction）任务，即一次性预测一个集合，而不是像RNN那样逐个进行预测，这样的方式会更高效。为此，DETR（Detection Transformer）应运而生。

文章所做的工作，就是将transformers运用到了object detection领域，取代了现在的模型需要手工设计的工作（例如非极大值抑制和anchor generation），并且取得了不错的结果。DETR在COCO数据集上的表现与Faster R-CNN相当，在大目标检测上效果更好。此外，DETR还具有很好的可迁移性，能够容易地应用于其他任务，例如全景分割。

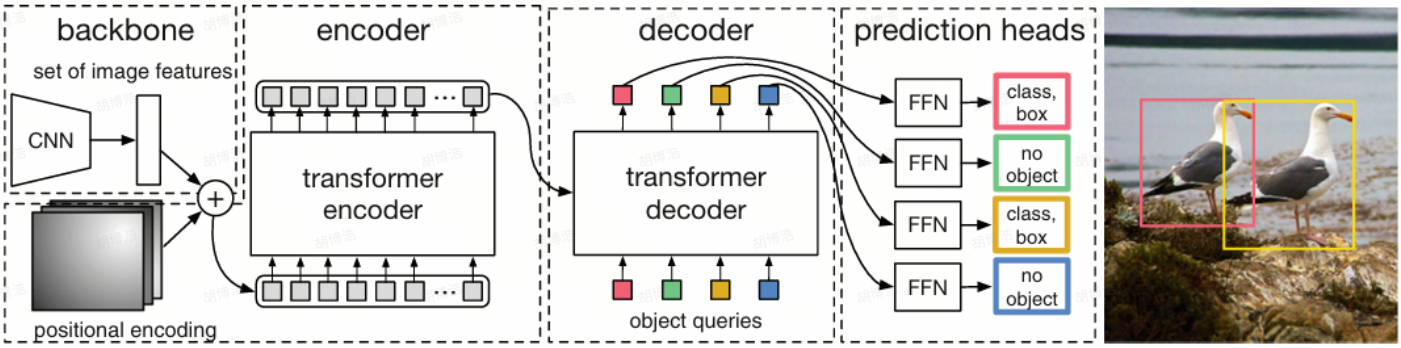


方法

DETR的总体思路是将目标检测视为集合预测（set prediction）问题，并使用Transformer来预测边界框的集合。DETR利用标准的Transformer架构来执行传统上特定于目标检测的操作，从而简化了检测的pipeline。



DETR整体结构可以分为四个部分：backbone，encoder，decoder和FFN。如下图所示：



为深入理解该算法整体结构，将其拆解为以下几个步骤：

1. **特征提取**：首先，使用一个卷积神经网络（通常是ResNet）作为Backbone来提取图像的特征。这个过程生成了一个特征图，该特征图保留了输入图像中的空间信息和上下文信息。
2. **特征铺平和位置编码**：将提取到的特征图铺平成一个序列，并添加位置编码（Position Embedding）。位置编码用于引入位置信息，弥补Transformer架构中缺乏空间信息的不足。
3. **编码器**：将带有位置编码的特征序列输入到一系列Transformer编码器（Encoder）中。编码器由多个自注意力层和前馈神经网络层组成，能够捕捉全局的上下文信息，并生成一组高维特征表示。
4. **候选特征**：编码器的输出是一组高维特征，这些特征作为检测候选对象（candidates）的特征表示。
5. **解码器**：将候选特征并行输入到Transformer解码器（Decoder）中。解码器也是由多个自注意力层和前馈神经网络层组成，结合编码器的输出和固定数量的查询向量（learnable queries），生成最终的检测框和分类结果。
6. **生成检测框**：解码器输出的每个查询向量对应一个检测框和一个类别标签。解码器的输出经过线性层，分别预测检测框的坐标和类别标签。
7. **匹配和损失计算**：使用匈牙利算法将解码器的输出与ground-truth进行一一匹配，计算匹配后的损失，从而优化模型。

通过以上方法的描述和与经典目标检测方法的对比，可以总结出本文的创新之处在于：

1. **提出了一种新的目标检测思路**：

DETR引入了一种真正的端到端（end-to-end）检测方法，简化了检测流程，减少了对先验知识的依赖。传统目标检测方法需要设计锚框（anchor boxes）和进行非极大值抑制（NMS）等步骤，而DETR完全摒弃了这些复杂的预处理和后处理操作。

2. **性能和效率**：

在COCO数据集上的实验结果表明，DETR的准确率和运行效率与高度优化的Faster R-CNN基本持平。在检测大目标时，DETR的效果比Faster R-CNN更好，展示了其在特定任务上的优势。

3. **易于复现和移植**：

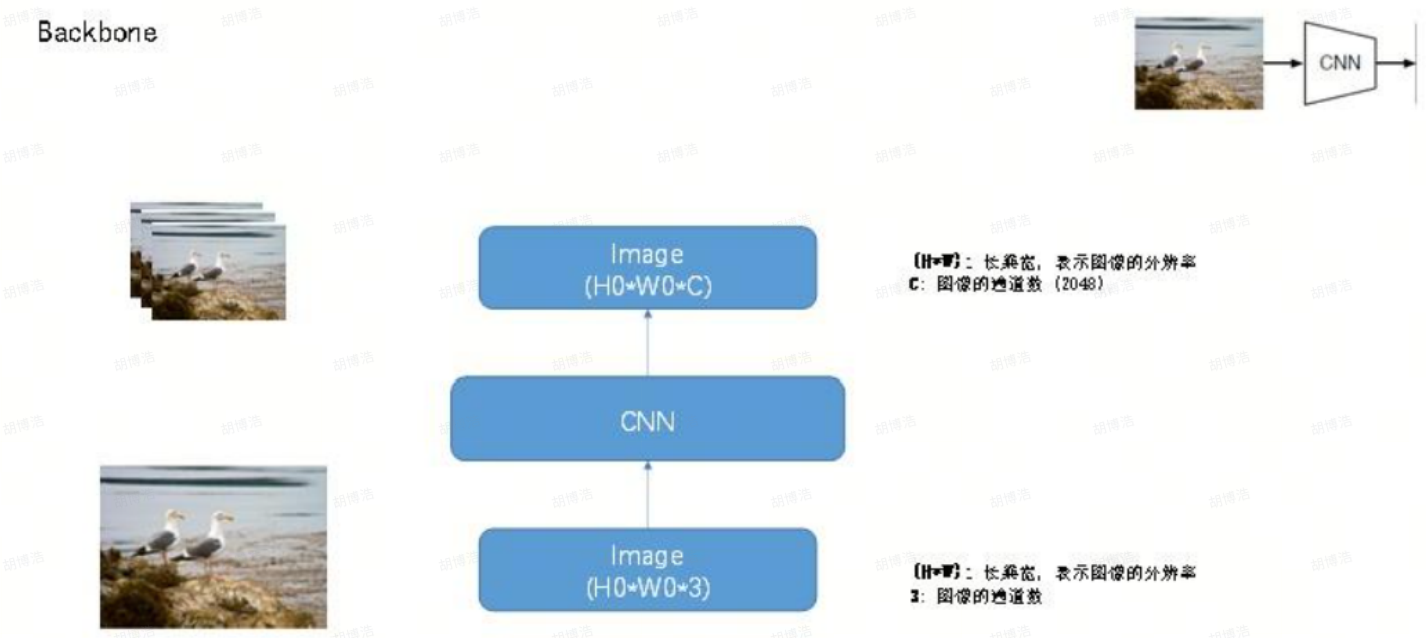
与大多数现有的检测方法不同，DETR不需要任何自定义层，这意味着它的实现完全依赖于标准的深度学习模块。因此，DETR可以在任何主流深度学习框架中轻松复现和移植。这一特性大大降低了使用和推广的难度，使得研究人员和工程师能够更方便地应用和改进这一方法。

了解算法的执行步骤后，我们来深入研究其网络搭建的过程：

1. CNN Backbone

原始DETR使用了在ImageNet上预训练的ResNet作为Backbone，处理 3×H0×W0 维的图像，得到 C×H×W 维的feature map（一般 C=2048,H=H0/32,W=W0/32 ）。然后将backbone输出的feature

map和position encoding相加，输入Transformer Encoder中处理。

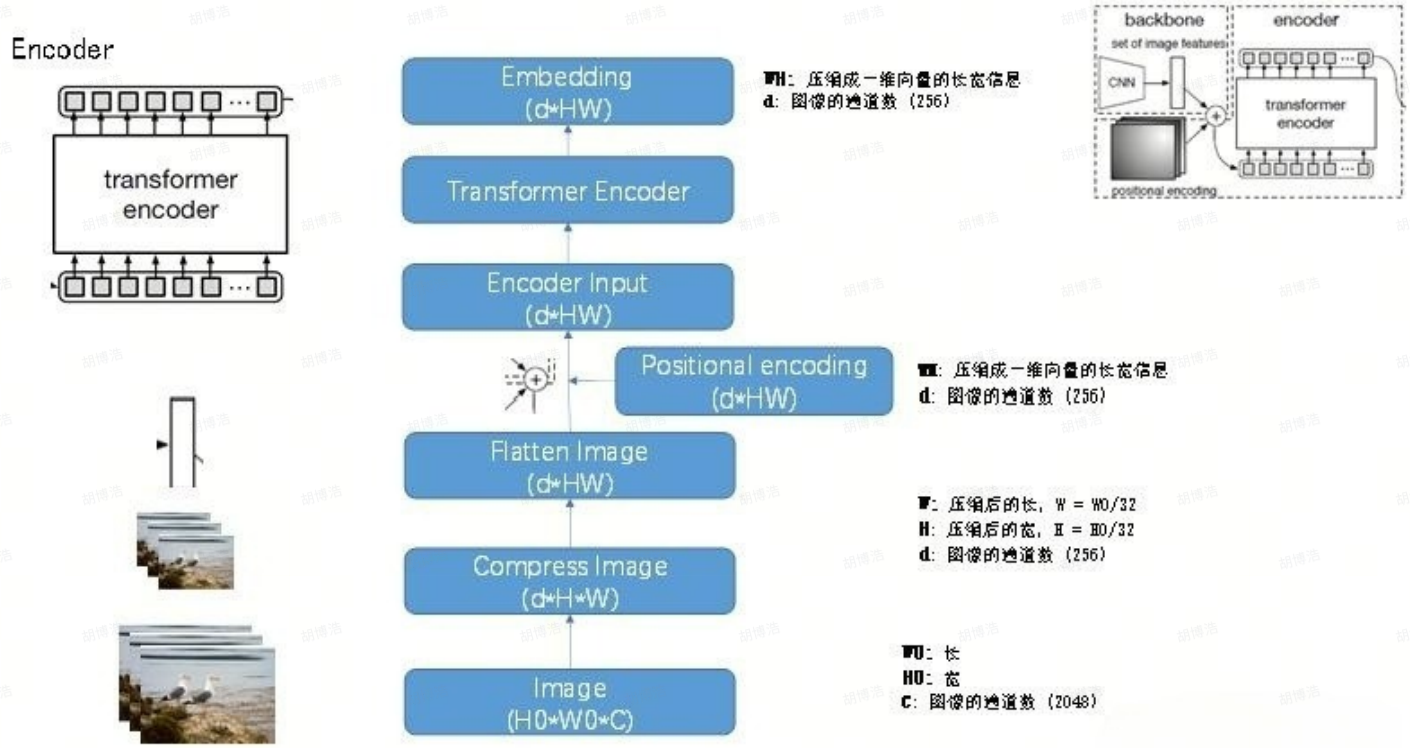


2. Transformer Encoder

经过Backbone处理后，输出的特征图被reshape为C × HW的形式，其中C=2048表示每个token的维度。由于这个维度较大，所以在输入Transformer Encoder之前，先通过一个1×1的卷积进行降维处理。将channels数量从 C 压缩到 d ，即得到 d×H×W 维的新feature map.

在此阶段，Transformer的自注意力机制在特征图上进行全局分析。由于最后的特征图对于大物体更友好，因此在上面进行Self-Attention有助于网络更好地提取不同位置上不同大小物体之间的关系。例如，有桌子的地方可能有杯子，有草坪的地方可能有树，有一只鸟的地方可能还有另一只鸟。因此，DETR在大目标检测上效果比Faster R-CNN好。

位置编码在每一个Multi-Head Self-Attention前都会加入，这一点非常重要。为了反映图像在x和y维度上的信息，DETR在代码中分别计算了两个维度的Positional Encoding，然后将它们拼接在一起。整个Transformer Encoder的结构与之前的标准Transformer Encoder相同。

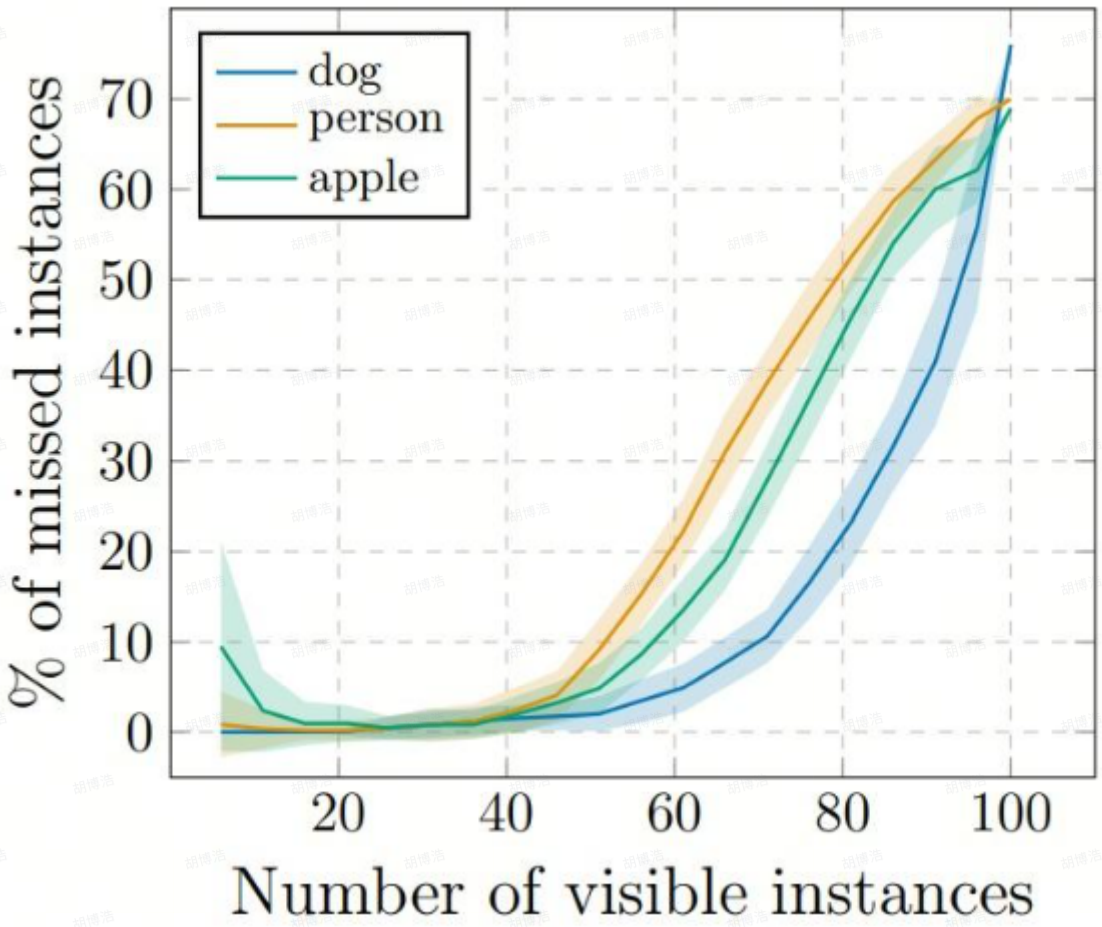


3. Transformer Decoder

在Transformer Decoder中有几个关键点需要强调：

首先就是如何考虑同时进行一个集合预测？不是像以往的transformer用法那样一个一个输出序列,在进行集合预测时，需要考虑同时预测多个类别。因为本质上每个对象检测都是独立的。为了解决这个问题，DETR选择了固定数量的查询token（Object Query），作者设定为N=100个token。这意味着网络同时可以检测最多100个物体。输入 100 个 decoder query slots (Object Query)，并行解码N个object，对应的 Transformer decoder 也就会输出 100 个经过注意力和映射之后的 token，然后将它们同时喂给一个 FFN 就能得到 100 个框的位置和类别分数（因为是多分类，所以类别个数是 K + 1，1 指的是背景类别）。

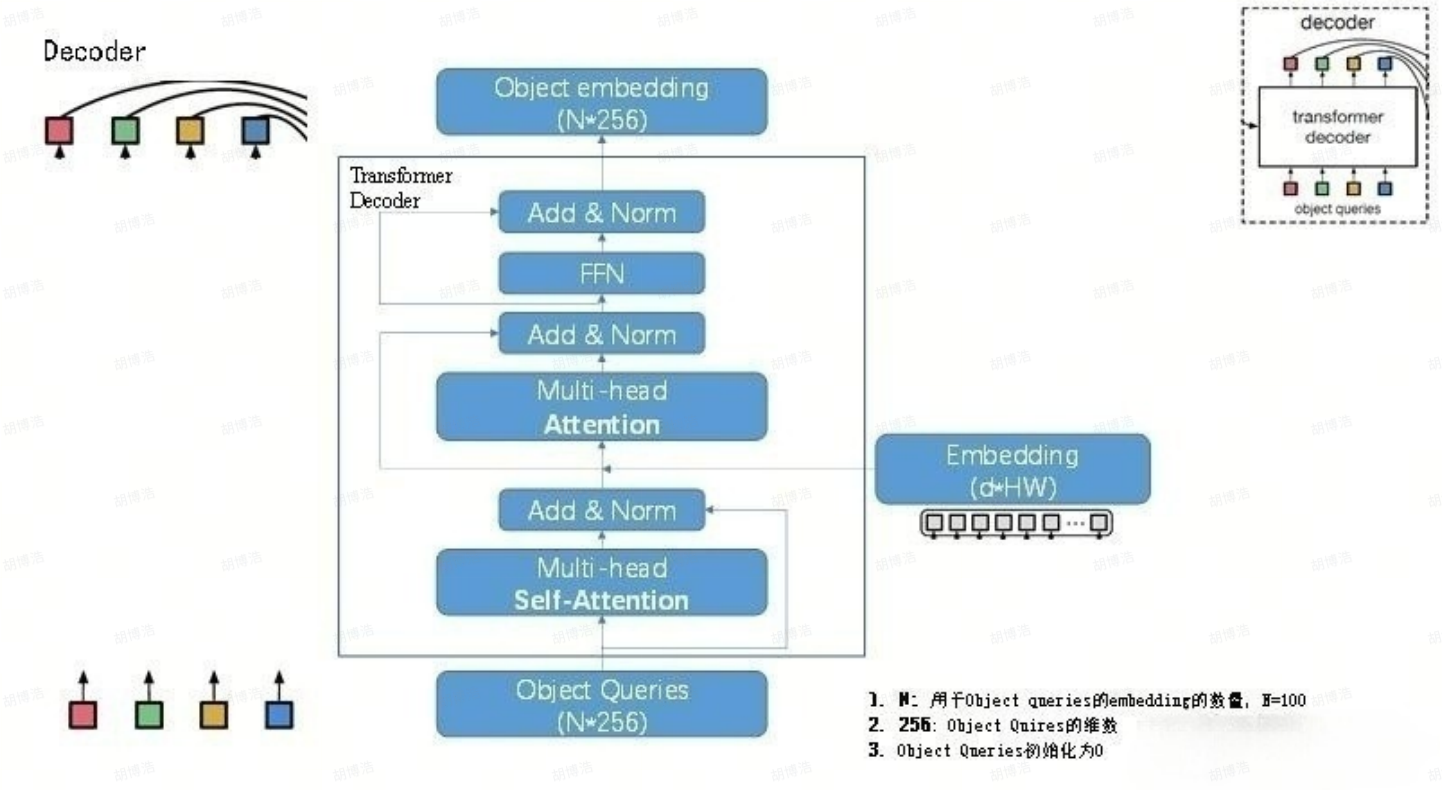
其次，固定预测个数简化了问题，同时也有利于显存对齐。然而，选择的N=100个token是否会过于冗杂呢？作者的实验表明，当图像中的目标个数在50左右时，网络就已经饱和了，之后可能出现目标丢失的情况。因此，当图像中目标个数在100个左右时，网络实际上只能检测到30-40个目标，这比图像中只有50个实例被检测到的情况还要少。这种反常现象可能是由于训练集中没有那么多多目标图片造成的，导致检测结果与训练分布相差甚远。



为了提高平均准确率（AP），作者提到了一些预测为背景的情况。在推理时，可以使用第二高分的类别来覆盖这些预测，使用相应的置信度。具体的选择策略可能取决于背景概率的阈值等细节。

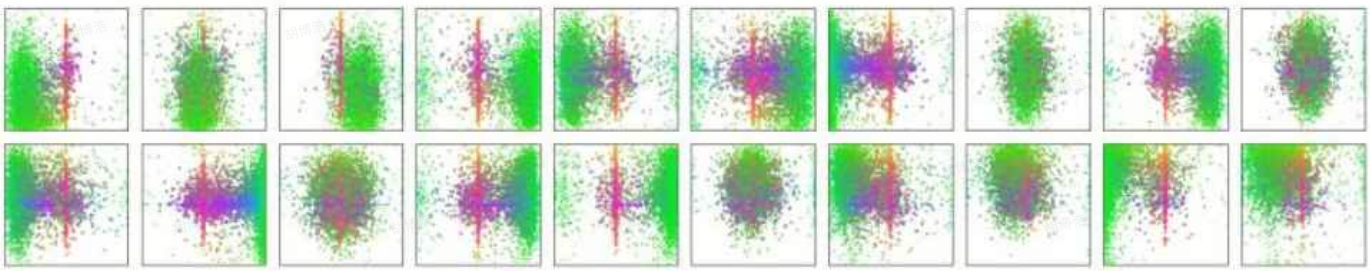
DETR的Decoder也加入了位置编码。在图像分类任务中，只有一个类别token，因此不需要位置编码。但在目标检测中，希望不同的Object Query对应图像中不同的位置。因此，Object Query自然就是位置编码，即查询这里的物体，预测出来的就是对应的位置和类别，只不过它是可以学习的位置编码。

在DETR中，位置编码的添加相当于将位置信息注入到Transformer Decoder中的每一层。具体来说，每层的key都添加了位置编码，而Decoder每一层的query都加了位置编码（Object Query）。



除此之外，还有一点需要注意的是，Decoder每一层的输出结果经过参数共享的最后的Feed-Forward Network (FFN)进行预测和计算损失，从而实现了深度监督。

作者给出了可视化结果，展示了在COCO2017验证集中预测得到的目标的中心点位置分布。可视化结果显示了100个Object Query中的20个Object Query对应的目标中心点位置分布。绿色表示小物体，红色表示水平的大物体，蓝色表示竖直的大物体。从可视化结果中可以看出，不同的Object Query确实实现了对不同位置出现的小物体的查询，比如左下、右边、右上等。但是对于大物体而言，大家检测出来的定位是相近的。

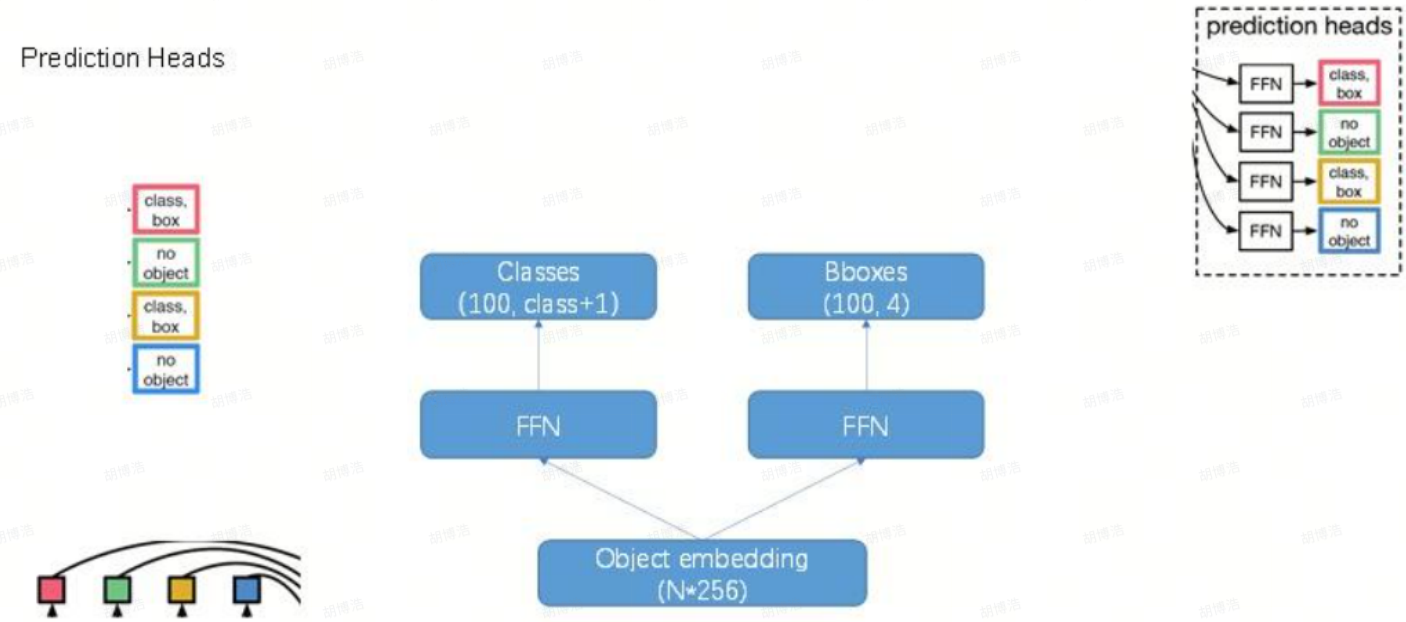


4. FFN

最终的Feed-Forward Network（FFN）由具有ReLU激活函数和隐藏层的3层线性层计算，或者说就是1×1卷积。FFN负责预测边界框的标准化中心坐标、高度和宽度，并通过softmax函数激活获得预测的类别标签。

需要注意的是，图中画的多个FFN模块实际上是同一个模块，所有类别共享参数，这意味着每个对象都是独立检测的。FFN的输入是Decoder对Object Query解码后的输出。边界框预测由ReLU激活的3层感知机完成，而类别预测则是一个线性层加上softmax预测层。

从更详细的结构图（下图）来看，实际上存在两种FFN：一种用于预测边界框的中心位置、高度和宽度，另一种用于预测类别标签。



在神经网络模型中，损失函数是非常关键的部分。DETR预测了一组固定大小的N = 100个边界框，这比图像中感兴趣的对象的实际数量要大得多。那么，如何计算损失呢？又如何知道预测的框对应哪个ground-truth的框呢？

为了解决这个问题，首先将ground-truth也扩展成N = 100个检测框，使用了一个额外的特殊类标签 $\varnothing$ 来表示在未检测到任何对象时的情况，或者认为是背景类别。这样一来，预测和真实都是两个100个元素的集合了。接着，采用匈牙利算法进行二分图匹配，即对预测集合和真实集合的元素进行一一对应，以最小化匹配损失。

1. 匈牙利算法

匹配时采用的损失函数如下所示。其目标是在保证类别正确的同时，最小化边框误差。

$$L_{match} = -\mathbf{1}_{\{c_i \neq \phi\}} \hat{p}_{\sigma(i)}(c_i) + \mathbf{1}_{\{c_i \neq \phi\}} L_{box}(b_i, \hat{b}_{\sigma(i)})$$

简单概括就是先到先让，abc和def在两个集合里，a先配对到d；如果b也配对到d，则a让出d，a再去配对其能配对的第2个(假设为e)；然后c如果配对到e，则让a再找a能配对的第三个(假设只剩f)，如果c配对到d，则b先让出d，假设b第二次配对到f，则这时a再让出f。

通过上面的配对损失，预测集和gt集能得到固定的一组最优配对。从而，我们就得到了 ground truth 和预测目标框之间的一一对应关系。然后就可以计算损失函数了。

2. 损失函数

DETR的损失和匹配损失类似，作者称为匈牙利损失(Hungarian Loss)。

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

$$L_{box}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{iou} L_{iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{L1} \|b_i - \hat{b}_{\sigma(i)}\|$$

不同之处在于，匈牙利损失将类别判定部分改为对数形式，并将背景类别 ϑ 纳入考虑。

在之前进行match时候的损失不是对数，而是二值函数，意味着把和ground truth类别一致的，且bbox最接近的预测结果对应上就完事了，不关心其他那些背景类 ϑ。

但是在算训练模型的Hungarian loss时，我们不希望模型预测出乱七八糟的结果，因此，对背景类 ϑ 的惩罚会加重，对 ϑ 类的权重除以10以平衡样本量。

边框损失Lbox如下，λiou和λL1是超参数，用于调整IoU损失和预测框的L1损失，Liou用的是GIOU损失

$$GIoU = IoU - \frac{|A_c - U|}{|A_c|}$$

通过匈牙利算法得到ground truth和预测目标框之间的一一对应关系后，我们可以计算损失函数，指导模型学习正确的目标检测任务。

实验结果

实验结果显示，DETR在COCO数据集上取得了比Faster R-CNN和RetinaNet更好的性能。研究人员还对其结构和损失进行了详细的消融研究，并提供了定性结果。最后，为了证明DETR是一个通用模型，作者进行了全景分割的实验，并取得了令人满意的结果，而这只是对DETR模型的一个小扩展。

首先，介绍实验的设计：

1. **数据集与评估标准**：
   - 使用COCO 2017检测和全景分割数据集，包含118k训练图像和5k验证图像。
   - 评估标准包括bbox AP（在多个阈值下的积分度量）、AP50（IoU阈值为0.5时的AP）、AP75（IoU阈值为0.75时的AP）、APS（小对象的AP）、APM（中等对象的AP）和APL（大对象的AP）。
2. **技术细节**：

- 使用AdamW优化器，设置transformer的初始学习率为10^-4，设置backbone的初始学习率为10^-5，权重衰减设置为10^-4。
- 采用Xavier方法初始化初始权重，backbone使用ImageNet预训练的ResNet模型。
- 进行了数据增强，如尺度缩放和随机裁剪。
- 训练了不同backbone的DETR模型，包括ResNet-50和ResNet-101，并对性能进行了比较。

对实验结果进行分析：

1. 模型性能对比试验
   - DETR的性能优于当前在MSCOCO数据集下的SOTA模型UPSnet。

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	45.1	79.9	55.5	50.5	80.9	61.7	37.0	78.5	46.0	33.0
DETR-DC5	R101	**45.6**	**80.0**	**56.1**	50.9	80.9	**62.2**	**37.5**	78.6	**46.8**	33.1

   - DETR模型获得了与经过大量调整的Faster R-CNN baseline相当的结果。

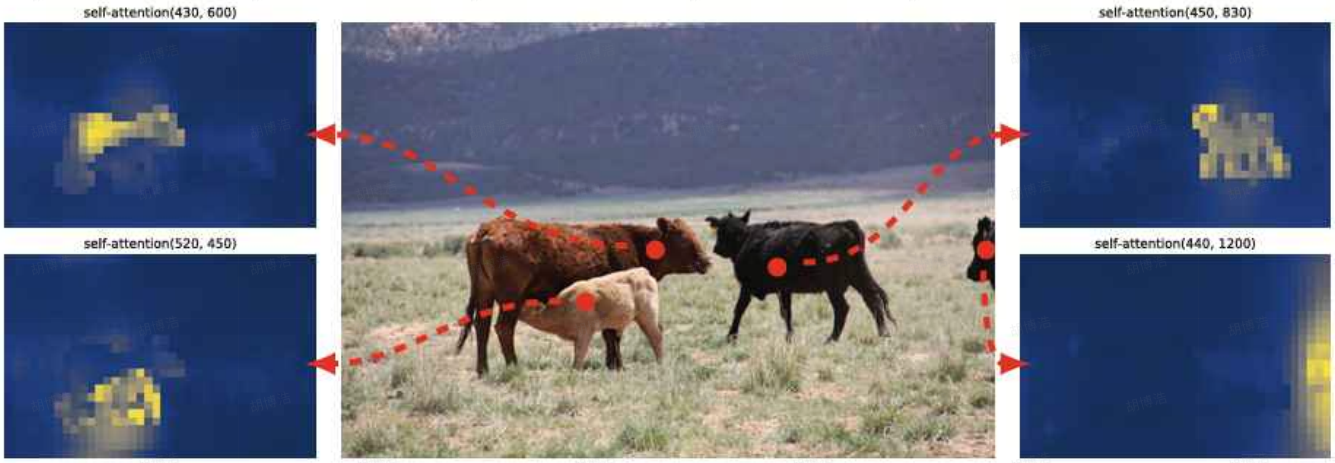
Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet	205/18	38M	38.7	58.0	41.5	23.3	42.3	50.3
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
RetinaNet+	205/18	38M	41.1	60.4	43.7	25.6	44.8	53.6
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**

2. Encoder层数对比试验

- 实验结果表明，Encoder层数越多越有利。

#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

3. Encoder Layer的attention可视化
   - 最后一个Encoder Layer的attention可视化显示，Encoder已经成功地分离了instances，简化了Decoder的对象提取和定位。



与Faster R-CNN和RetinaNet相比，DETR模型在COCO验证集上达到了与Faster R-CNN相近的性能，尤其在大对象检测方面表现出色。然而，在小对象检测方面，DETR的性能稍逊于Faster R-CNN。

综上所述，DETR作为一种基于Transformer的新型目标检测器，在处理全局信息和大型对象检测方面具有优势，并展示了在全景分割任务上的潜力。虽然在小对象检测方面还有改进空间，但其性能为未来工作提供了方向。