# Title:

End-to-End Object Detection with Transformers

# Abstract:

本文就是把目标检测看成一个集合预测的问题（目标检测的任务就是给定一个图片，然后去预测一堆锚框（anchor），每个锚框不仅要知道它的坐标，还要知道它框内物体的类别，这些框其实就是一些集合，对于不同的图片来说，锚框的集合是不一样的。我们的任务就是给当一个图片，把这个锚框的集合预测出来。之前很少人做到端到端）。本文的贡献是把目标检测做成了端到端的框架，把之前依赖先验知识的部分删除了，比如非极大值抑制（non-maximum suppression）或生成锚框（anchor），不再需要NAS，整个网络变得非常简单。

DETR具体提出了什么呢？

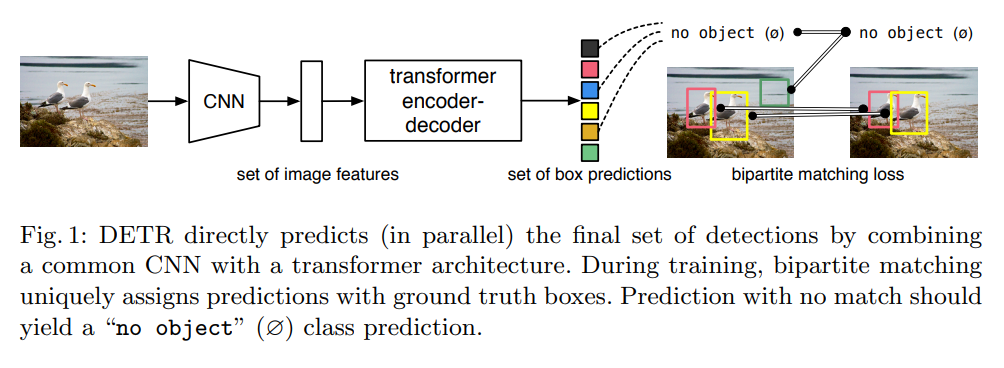
一是新的目标函数：通过二分图匹配的方式强制模型输出一组独一无二的预测（每个物体理想状态下只会输出一个框）。二是使用了Transformer的encoder-decoder架构：通过给定学习的物体查询和全局的图像信息，DERT会不停地做注意力操作，从而让模型直接输出最后的预测框（一起出框）。

这个新的模型想法非常简单，而且实现上也不需要一个特殊的深度学习库。而且性能也不错，

# Intro：

现在的好用的目标检测器都是用间接的方式来处理这个集合预测问题，比如proposal的方式：R-CNN系列。或者用anchor的方式：YOLO，Focal loss。或者是non Anchor base的方法：如物体的中心点center net或者FCOS的方法。这些方法要么是回归要么是分类的预测解决问题，没有直接去完成这个任务。之前的方法它们的性能极大地受限制于NMS这步后处理操作，非常难以优化和调参 。本文提出了一个直接的方式去解决这种集合预测问题，绕过了这些人工设计的部分。这种端到端的思想已经在别的任务里大范围地使用了，使这些任务变得非常简单。

DETR流程：

第一步：先用卷积神经网络抽取特征。第二步：拿到特征后把它拉直，送入Transformer encoder-decoder，这里编解码器地作用就是进一步地去学习全局的信息，为之后的出预测框做铺垫，这种全局的特征非常有利于去移除冗余的框。第三步：用Transformer decoder生成框的输出，这其中有一个object query（在论文内，作者限定其值为100，意味着无论有多少个物体，最终只会出100个框），它限定了要出多少个框，通过query的decoder不停做自注意力操作，从而得到生成框的集合。第四部分：DETR把这个问题看成集合预测的问题，最后能用二分图匹配的方法计算loss。如上图Ground Truth只有两个框，在训练时，通过计算这两个框和这两个框的匹配损失（matching loss），从而决定出在这100个框中哪两个框是独一无二的对应到这两个Ground Truth上的框，一点决定好匹配关系，然后就会和普通的目标检测一样算一个分类的loss和一个banding box的loss。剩下没有匹配到的框则会标注为没有物体（背景类）。

推理时，前三步是一致的，最后一步是设置一个阈值，将置信度高的保留（>0.7），所谓前景物体，剩下的被当成背景物体。

DETR在COCO上和Faster R-CNN表现差不多，但大物体检测较好，小物体不行，作者相信后人的工作会改进这一点。

DETR训练太慢，想要达到很好的效果，作者训练了500个epoch（一般在COCO上训练几十个就行）。

DETR是一个框架，可以拓展到很多别的任务上，如在这篇论文，DETR被验证在全景分割上很有用。

# Related Work：

## Object Detection

现在的大部分目标检测器都是通过已有的初始猜测然后去做预测的。比如对Two-stage的方法就是中间的proposal，single-stage的方法就是anchor，或者物体的中心点。以上工作的性能和初始的猜测非常相关，怎么做后处理也是对性能影响至关重要的。

**Set-base loss：**

以前的工作如可学习NMS或者关系型网络都可以用类似于自注意力的方法去处理物体之间的联系，从而导致可以做出独一无二的预测，以至于不用后处理的步骤了，但是这些方法有很多人共设置的场景和干预帮助学习。DETR不希望使用过多的先验知识，它希望过程尽可能简单。

**Recurrent Detectors：**

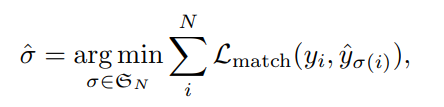
之前也有使用encoder-decoder架构做目标检测和实例分割的，但大多是RNN，时效性和性能比较差。而DETR使用Transformer的encoder-decoder，达到了平行输出所有预测框，时效性大大增强了。

# The DETR model

## Object detection set prediction loss

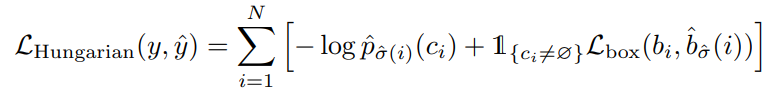
目标函数：

DETR这个模型输出是固定的集合，如何把输出的图和Ground Truth的图算loss呢？作者把它转化成了一个二分图匹配的问题。



使用Scipy.linear\_sum\_assigment

给定一个cost矩阵和100个框，可以用匈牙利算法算出最小loss的匹配，cost矩阵里放入每个框的loss，loss值由以下函数定义：



使用这个loss做梯度回传和更新参数，第一项是分类的loss，第二类是出框的loss

其实这种找最优匹配和原来用先验知识（proposal、anchor）这种匹配差不多。只不过这一约束更强，是一对一的方式，而不是一对多的。

## DETR architecture

前向过程例子：

3\*800\*1066输入仅卷积网络，得到2048\*800/32\*1066/32=2048\*25\*34的特征（C、H、W），然后进行1\*1的卷积降低通道数，输出为256\*25\*34，然后进入Transformer前，加入位置信息（维度为256\*25\*34）和拉直向量（将hw放一起），变成850\*256（hw、C）。Transformer encoder输入和输出维度不变，为850\*256.接下来进入Transformer decoder，这里有个可学习的object queries，维度为100\*256（256和encoder输出对应，100为最后得到100个输出），在Transformer decoder里做的是一个cross-attention，最后得到了100\*256的特征。然后把特征给予全连接层，全连接层做两个预测，分别为物体类别的预测（COCO为96类）和出框（中心点（x，y）和宽度高度）的预测，再进行最优匹配和计算loss，梯度反向回传更新参数。

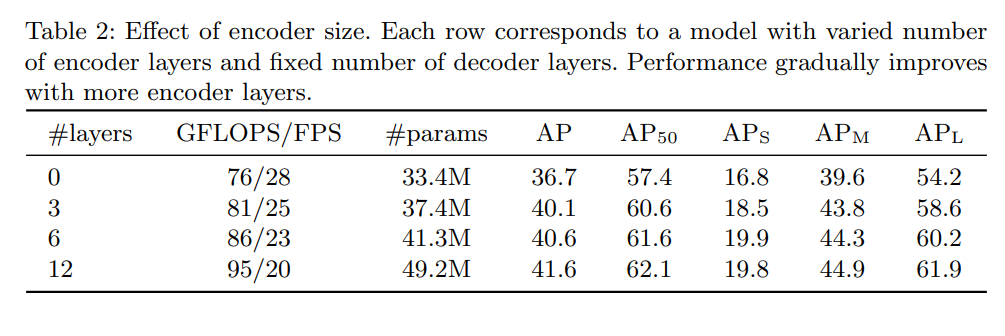
## Experiments

对比Faster R-CNN，DETR的性能如何

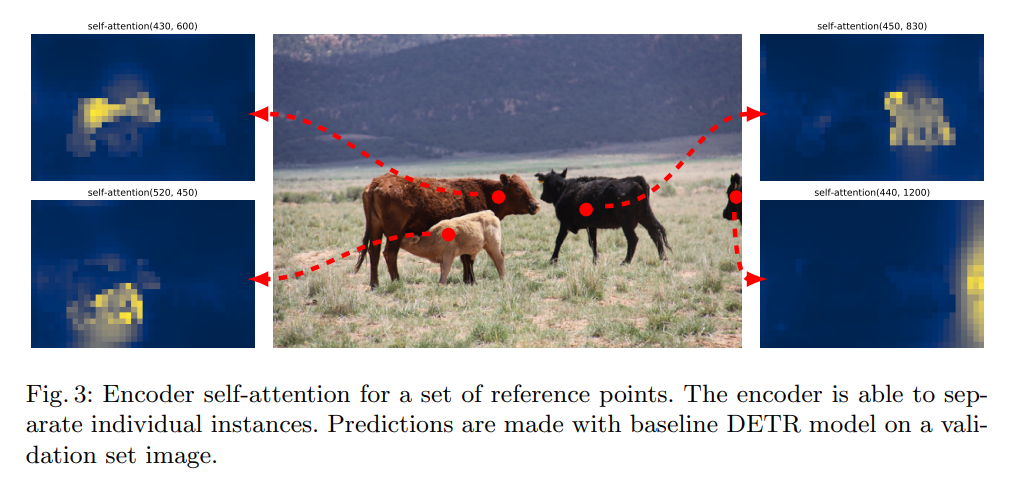
这里更想说明的是DETR比Faster-RCNN的大物体检测方面好很多

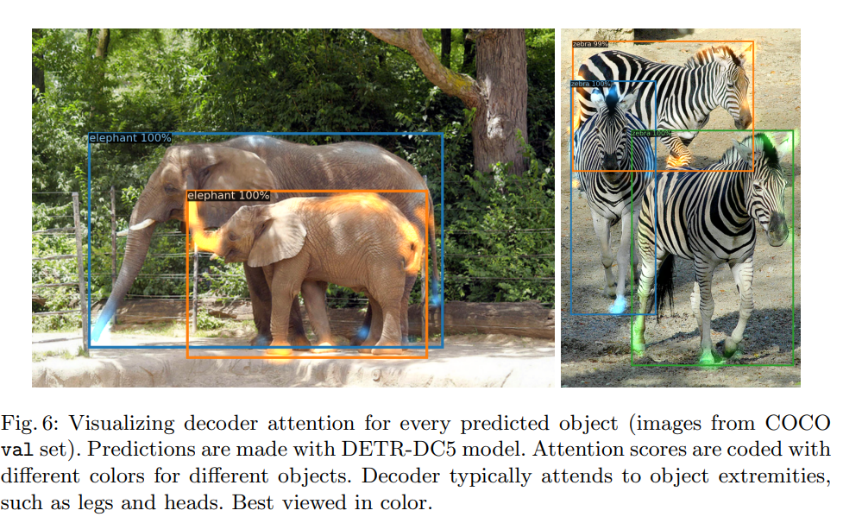
## Ablations

消融实验：



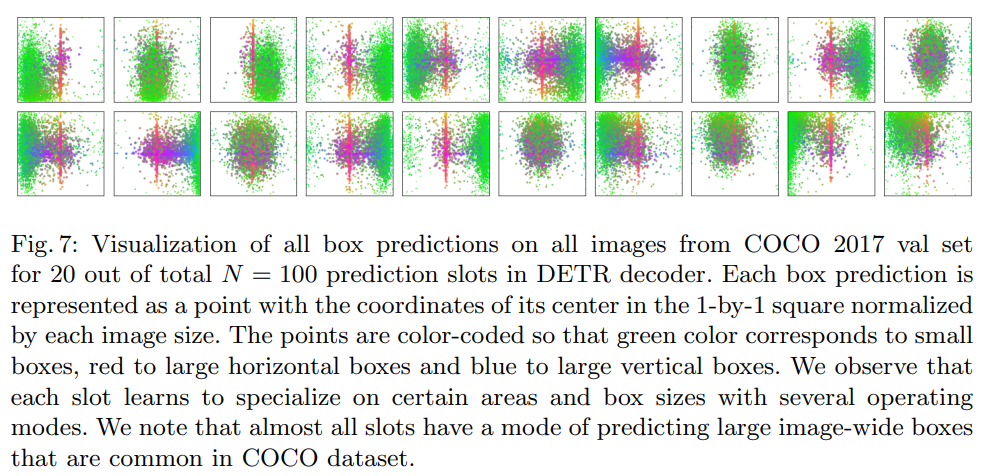
随着层数的增加，性能一直在增加，没有饱和的现象

这个图表示，在图片里点一些点，看看这些点与图片上其他点进行注意力操作，会发现已经做的非常好了。



这张图展示了Transformer decoder的威力，它把每个大象用不同颜色的框标注了出来。即使在遮挡很严重的情况下，还是能把轮廓显示出来。

encoder在学一个全局的特征，它尽可能让物体和物体之间分得开，最外围的极值点交给decoder去做

可视化COCO 2017验证集中所有图像上的所有框预测。对于DETR解码器中的总N = 100个预测插槽中的20个，每个框预测都表示为一个点，其中心坐标在每个图像大小归一化的1x1正方形中。这些点按颜色编码，绿色对应小盒子，红色对应大水平盒子，蓝色对应大垂直盒子。我们观察到每个插槽都学习了特定区域和尺寸的特定模式。我们注意到几乎所有插槽都有一个预测大型图像宽盒子的模式，这在COCO数据集中很常见。

这张图表示了可学习的object queries都学了什么，绿色的点表示这个向量每次密集地问某个区域有没有物体，红色的点表示这个向量在问中间有没有大的物体，100个queries相当于100个问问题的人，每当图片来的时候，他们用各自的方式问问题，如果找到了合适的答案，它就把答案返回。

# Conclusion：

作者提出了DETR这个框架，主要是应用了Transformer和二分图匹配，最后导致这个框架是非常简单的端到端可学习的网络。在COCO上和Faster RCNN打了个平手，在全景分割上取得了更好的结果，因为它的简单应用性，他还有很大潜力应用到其他任务上。

缺点是训练时间太长，不好优化，性能也不是很好，尤其在小物体上的性能。

它是用Object query代替了anchor，用二分图匹配代替了NMS，把之前不可学的东西变成了可以学习的东西，是一个里程碑式的工作