End-to-End Object Detection with Transformers（DETR）Notes（ECCV 2020 best）

Abstrat

在过去的工作中，目标检测模型基本都不是端到端网络，要么要在处理特征前用区域建议，要么就是在输出anchor时使用最大值抑制。本文的主要贡献就是利用集合预测把整个目标检测网络变成了端到端模型，把架构变得very very simple。首先DETR提出了一种新的目标函数最后每个物体只会生成一个框，不会生成那么多冗余；其次DETR使用了Transformer架构，注意力机制可以使用并行出框。

Introduction

先说下基础框架

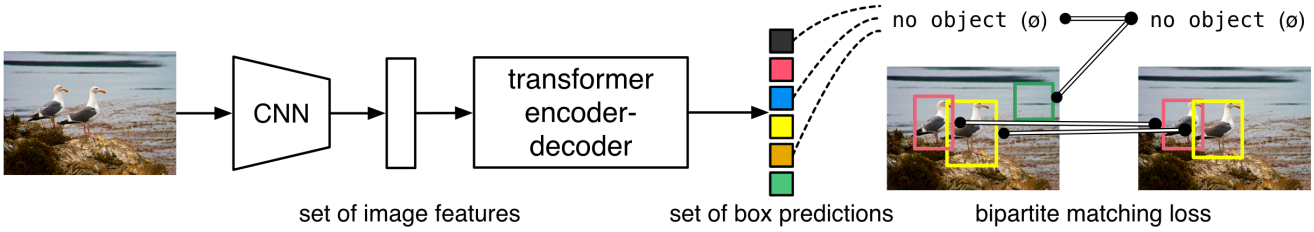

Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" ($\varnothing$) class prediction.

首先输入一张图片，用CNN抽取特征，再flat一下，进入transformer模块，这里这自注意力操作可以让像素进行交互，学习全局信息，再进行object query大概就知道了图片有什么物体，后面再固定生成100个预测框，最后利用二分图匹配进行和ground truth的匹配度，没有达到匹配条件的就会被分为no object（背景类）。

Method

首先是集合预测。

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

DETR的输出是固定的N个框（这里取100），计算每个框和ground truth的匹配度转换成了二分匹配问题。二分匹配最著名的是匈牙利算法，scipy里也有很简单的linear-sum-assignment算法，详情可见https://zhuanlan.zhihu.com/p/96229700。

和nms不同的是，DETR的二分匹配，最后只会有一个框和一个特定的ground truth匹配，而不是像nms那样多个框能匹配得上。

看看目标损失函数。

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

左边这一块是分类损失，作者表示为了保证值域和右边相近，他们把log去掉，发现效果不错；右边这一块是出框损失，而作者没有采用$L1$损失，因为L1损失特性是框越大损失越大，而Transformer因为关注全局信息，因此对大物体更友好，即使相关性和小框差不多，天然的$L1$损失就更大，所以作者使用了一种叫$generalized\ IoU\ loss$和$L1$相结合的损失函数。

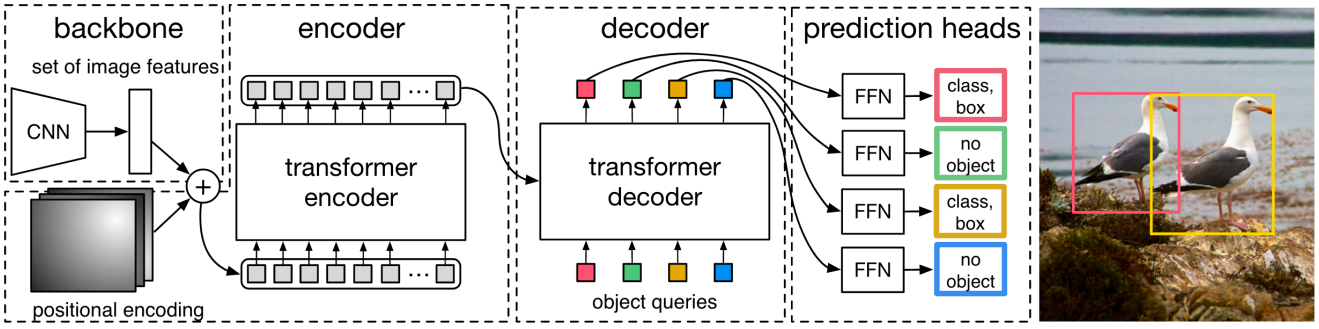

Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.