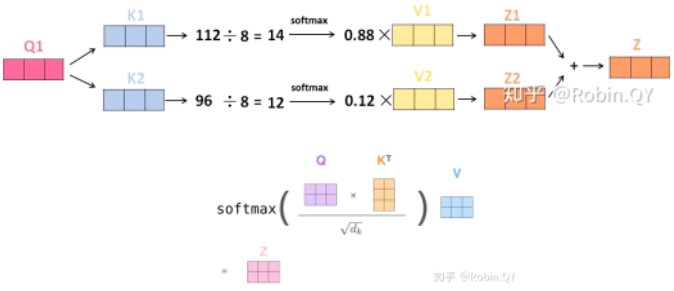
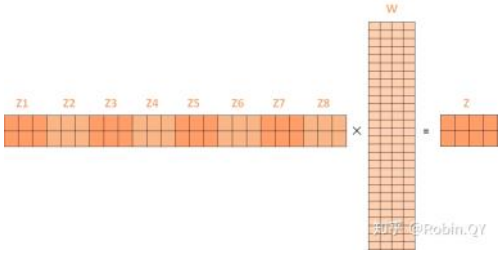
Transformer

2022年5月8日 10:07

1. Transformer
   - 完全基于注意力的模型
   - 组成：
     - 编码器（6组）
       - 组成：多头注意力+前馈神经网络
     - 解码器（6组）
       - 组成：掩蔽多头注意力+多头注意力+前馈神经网络（MLP）
   - 自注意力
     - 计算每个单词与其他单词的相关性
     - 除以根号dk的原因
       - 默认Q、K、V的维度是64
       - 原论文中作者发现当QK的乘积值太大时，经过Softmax映射后的值非0即1，产生的梯度太小，不利于网络训练
       - 除以根号dk，保证QK值在一个较小的范围内



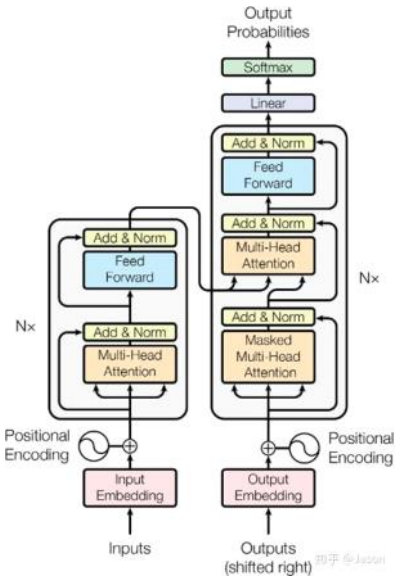
   - 多头注意力
     - 作用：
       - 扩展了模型专注于不同位置的能力
       - 为注意力层提供了多个“表示子空间”（8组）

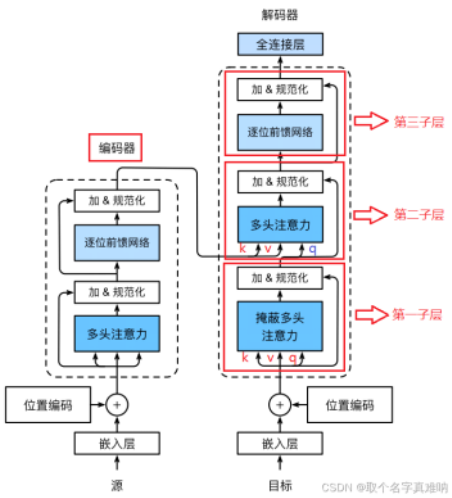


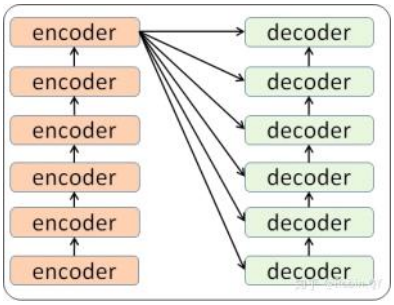
   - 掩蔽多头注意力
     - 解码器中，自注意力层只能作用于输出序列的较早位置，也就是需要在进行自注意力的softmax操作之前将未来位置的值屏蔽掉（设置为-inf）
   - 残差结构：缓解梯度消失
   - 位置编码
     - 作用：表示输入序列中单词的顺序
     - 函数：Sin+Cos
   - 最后的Linear+Softmax
     - Linear：全连接层，得到一个与单词数相同的向量（有100个词就得到100维的向量）
     - Softmax：将向量值转换为概率，概率最大的单词即为最终输出

	I	am	machine	he	is	learning
position1	0.01	0.02	0.93	0.01	0.01	0.01
position2	0.01	0.01	0.05	0.02	0.01	知乎 @Robin.QY

   - 结构图







2. ViT
   - 基于Transformer的图像分类模型（无解码器）
   - 与CNN相比，Transformer缺少归纳偏置，即先验知识
     - 目标周围的邻域信息对目标分类的影响
     - 卷积操作的平移不变性
   - 算法流程：

     (1) patch embedding：例如输入图片大小为224x224，将图片分为固定大小的patch，patch大小为16x16，则每张图像会生成224x224/16x16=196个patch，即输入序列长度为**196**，每个patch维度16x16x3=**768**，线性投射层的维度为768xN (N=768)，因此输入通过线性投射层之后的维度依然为196x768，即一共有196个token，每个token的维度是768。这里还需要加上一个特殊字符cls，因此最终的维度是**197x768**。到目前为止，已经通过patch embedding将一个视觉问题转化为了一个seq2seq问题

     (2) positional encoding（standard learnable 1D position embeddings）：ViT同样需要加入位置编码，位置编码可以理解为一张表，表一共有N行，N的大小和输入序列长度相同，每一行代表一个向量，向量的维度和输入序列embedding的维度相同（768）。注意位置编码的操作是sum，而不是concat。加入位置编码信息之后，维度依然是**197x768**

     (3) LN/multi-head attention/LN：LN输出维度依然是197x768。多头自注意力时，先将输入映射到q，k，v，如果只有一个头，qkv的维度都是197x768，如果有12个头（768/12=64），则qkv的维度是197x64，一共有12组qkv，最后再将12组qkv的输出拼接起来，输出维度是197x768，然后在过一层LN，维度依然是**197x768**

     (4) MLP：将维度放大再缩小回去，197x768放大为197x3072，再缩小变为**197x768**

     一个block之后维度依然和输入相同，都是197x768，因此可以堆叠多个block。最后会将特殊字符cls对应的输出 $\mathbf{z}_L^0$ 作为encoder的最终输出，代表最终的image presentation（另一种做法是不加cls字符，对所有的tokens的输出做一个平均），如下图公式(4)，后面接一个MLP进行图片分类

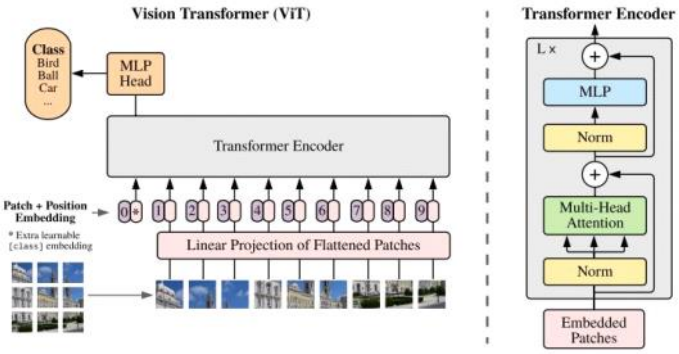
$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \quad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

   - 位置编码
     - 1-D编码：如果有9个patch，则位置编码为1-9
     - 2-D编码：编码为11,12,13,21,22,23,31,32,33. 同时考虑X轴和Y轴信息
   - 结构图



3. DETR
   - DETR是一种基于“集合预测”的目标检测框架（目标检测的新范式）
   - DETR结构
     - CNN网络：提取图像特征，输出为（C*H*W）
       - reshape为（W*H)*C　（行：共有W*H个像素，列：每个像素的特征维度为C）
     - Tansformer的编码器-解码器结构
       - 编码器输入输出维度一致
       - 解码器
         - 输入的object queries的个数是固定的，就是要预测的目标的个数N
           - object queries：可学习的位置编码，可以聚焦于图像的不同区域
           - Object Queries的含义
             - Object queries是一种可学习的位置编码
             - 用于计算Q矩阵：相当于提出问题（这个位置有什么目标）（Encoder提供了K,V）
             - Object queries是需要训练的，训练的目的相当于是学习如何提出更具体，更准确的问题
         - 并行解码N个object queries, 输出与输入是一一对应的，也是输出N个经过注意力映射后的token
     - 用于检测的前馈神经网络
       - 将N个token输入前馈神经网络进行预测，得到N个框的位置和类别分数
       - FFN负责预测边界框的位置参数，以及类别分数（1*1卷积）
   - 位置编码
     - 分别计算X和Y维度的编码，concat后与CNN的输出相加
   - 损失函数
     - 边界框匹配损失
       - DETR是集合预测，预测N个框的位置和类别，比原本的目标要多
       - 将GT扩展成N个框，此时预测集合和GT集合大小相同
       - 采用匈牙利算法进行二分图匹配，实现预测集合和GT集合的一一对应（原则：匹配损失最小）
     - 边界框损失
       - IOU损失和L1 损失的加权和
       - $$\mathcal{L}_{\text{Hungarian}}(y,\hat{y}) = \sum_{i=1}^{N}\left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + 1_{\{c_i \neq \varnothing\}}\mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\hat{\sigma}}(i)\right)\right]$$
         $$\mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\hat{\sigma}}(i)\right) = \lambda_{\text{iou}}\mathcal{L}_{\text{iou}}\left(b_i, \hat{b}_{\sigma(i)}\right) + \lambda_{\text{L1}}\left\|b_i - \hat{b}_{\sigma(i)}\right\|_1$$
   - 结构图

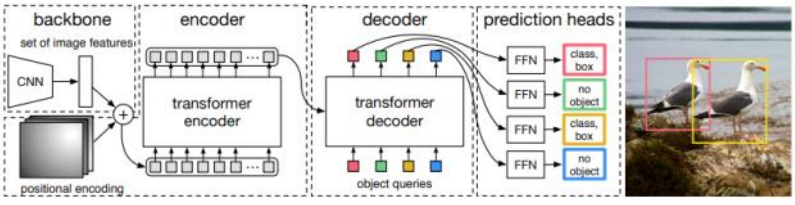
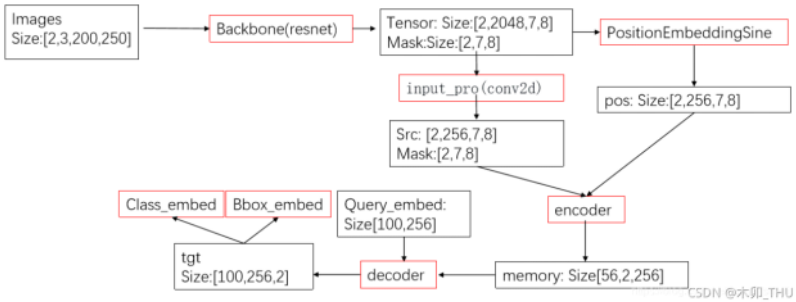
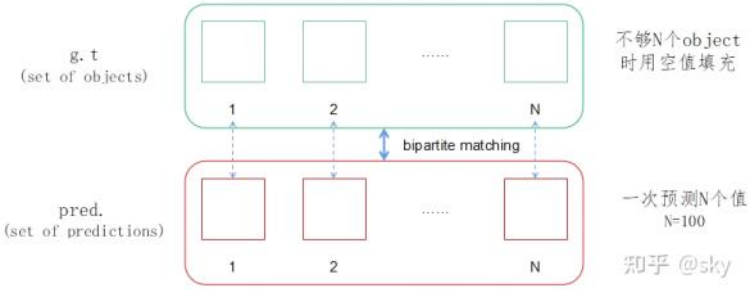


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, 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.





4. Swin Transformer
   - 架构

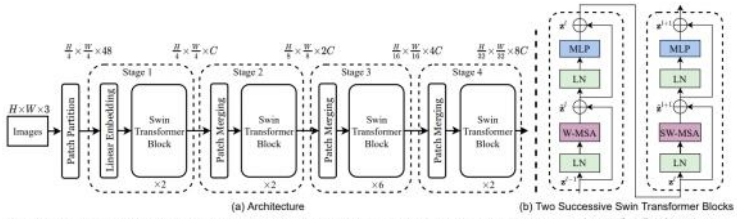
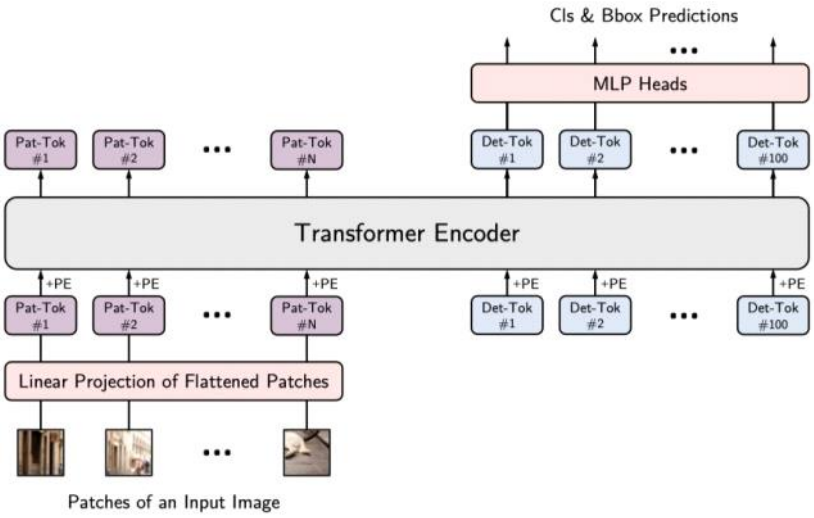


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

   - 结构解析
     - 总共分为4个阶段，每个阶段都会缩小特征图的分辨率，扩大感受野
     - Patch Embedding：将图片切成小patch，再展平为向量
     - Patch Merging：在每个stage中做下采样，缩小分辨率（在行和列方向上间隔2选取元素）
     - Swin Transformer Block
       - Window Attention：注意力计算限制在每个窗口内
       - Shift Window Attention：实现多个windows之间的交互
5. YOLOS
   - 将ViT结构迁移到目标检测任务
   - ViT是对全局上下文关系和长程依赖关系的建模，而不是局部区域关系的建模
   - YOLOS
     - YOLOS丢弃了ViT的【CLS】Token，而是添加了100个可学习的【DET】Token用于目标检测
     - YOLOS将ViT的图像分类损失替换为了匹配损失，按照DETR的预测方式执行目标检测
   - YOLOS相当于ViT和DETR的结合
   - 【DET】Token
     - 随机初始化的目标代理
     - 作用：
       - 标签分配：【DET】生成的预测框与GT集合中的目标进行一一匹配
       - 不需要将ViT的输出重新解释为2D结构再进行标签分配
   - 结构图



6. ViDT
   - 对ViT和DETR的高效集成模型
   - 动机：
     - 传统的目标检测方法的检测性能依赖于精心设计的组件：Anchor和NMS
     - DETR
       - 消除了这些精心设计的组件，基于Transformer的集合预测模型
         - ResNet提取图像特征
         - Transformer的Encoder-Decoder进行图像编码解码，Object Queries对应的Tokens通过FFN进行一一对应的最终的检测
       - 缺陷：
         - 收敛慢：传统37个epoch可以达到的效果，DETR需要500个epoch
         - 小目标检测效果差：用的是最后一层特征图生成的嵌入向量，小目标信息较少
       - 改进的Deformable DETR
         - 可形变注意力+多尺度特征图
           - 可行变注意力
             - 每一个Q不需要与所有的K进行计算
             - 以当前Q为参考点，通过线性映射计算偏移量，选择当前Q附近的M个点为采样点，计算注意力
             - $$\mathrm{DeformAttn}(z_q, p_q, x) = \sum_{m=1}^{M} W_m [\sum_{k=1}^{K} A_{mqk} \cdot W'_m x(p_q + \Delta p_{mqk})]$$
           - 多尺度特征图
             - 选取ResNet多个Stage的输出特征图生成嵌入向量，每一层的嵌入向量都添加了对应层的【层】标记
             - 训练过程中随机初始化【层】标记，Encoder-Decoder协同训练
     - ViT
       - 完全基于Transformer的图像分类模型
         - 额外添加了一个【CLS】Token负责类别预测
         - ViT也证明不需要额外的归纳偏置（先验知识：图像结构信息，卷积的平移不变性），模型依旧可以学习到有效的特征信息
       - 缺陷：复杂度较高，与图像大小呈二次增长
     - DETR（ViT）
       - 利用ViT代替ResNet提取图像特征，利用Encoder-Decoder作为Neck组件进行编码解码，Object Queries对应的Tokens通过FFN进行一一对应的最终的检测
       - 缺陷：
         - ViT的缺陷
         - Neck组件中Encoder-Decoder的注意力计算增加检测器的计算开销
         - 难扩展
     - YOLOS
       - 基于ViT的目标检测器
         - 在Encoder的输入中添加了【DET】Tokens，实现了无Neck组件的目标检测
           - 因为【DET】组件在ViT的Encoder中已经进行了全局注意力的交互，可直接用于检测，因此不再需要Decoder，从而实现无Neck组件的检测
         - YOLOS实现了高效的计算，同时也证明2D目标检测可以通过序列-to-序列的方式实现
         - 缺陷：

        - ◇ 继承了ViT的缺陷：高复杂度是由于全局注意力
        - ◇ 无Neck组件：无法使用多尺度特征提升检测性能
- ○ ViDT结构
    - ▪ RAM（重新配置的注意力模块）
        - □ 结构图

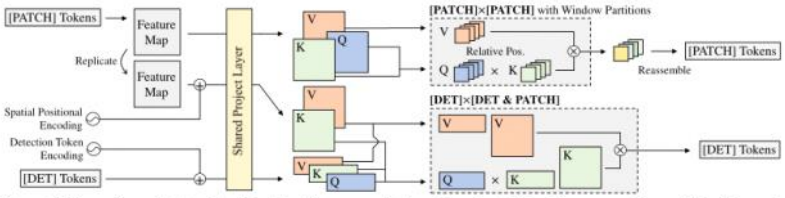


Figure 3. Reconfigured Attention Module (Q: query, K: key, V: value). The skip connection and feedforward networks following the attention operation is omitted just for ease of exposition.

        - □ RAM将【Patch】和【DET】的全局注意力分解为三部分：
            - ◆ 【Patch】*【Patch】
                - ◇ 作用：聚合全局关键内容
                - ◇ 操作：
                    - ▸ 窗口注意力：局部自注意力计算
                    - ▸ 移动窗口注意力：提供窗口之间的交互，捕获全局信息
            - ◆ 【DET】*【DET】
                - ◇ 作用：每个【DET】通过计算全局注意力来捕获他们之间的关系，有助于定位不同的目标
                - ◇ 操作：自注意力（self-attention）
            - ◆ 【Patch】*【DET]
                - ◇ 作用：每个【DET】都表示不同的对象，因此需要为每一个【DET】生成不同的embedding，【Patch】中的关键内容被聚合到每个【DET】上
                - ◇ 操作：交叉注意力
            - ◆ 利用RAM替换Swin Transformer中的所有注意力模块
            - ◆ 位置编码：
                - ◇ 【Patch】：Swin Transformer的相对位置编码
                - ◇ 【DET】：可学习的位置编码（【DET】没有特定的顺序）
            - ◆ 【Patch】*【DET】注意力只在Swin Transformer的最后一个Stage使用，避免增加额外的计算开销
    - ▪ Encoder-Free Neck组件
        - □ 无Encoder的原因：Swin Transformer已经提取了适用于目标检测的细粒度特征
        - □ Decoder：采用Deformable DETR的Decoder可以充分利用多尺度特征
            - ◆ 两个输入：每个Stage生成的【Patch】+最后一个Stage生成的【DET】
            - ◆ 使用多尺度形变注意力：用于生成新的【DET】，聚合从多尺度特征图中采样的一小部分关键内容
                - ◇ 【DET】*【DET】的自注意力
                - ◇ 【Patch】*【DET】的交叉注意力

$$\mathrm{MSDeformAttn}([\mathtt{DET}], \{\boldsymbol{x}^l\}_{l=1}^{L}) = \sum_{m=1}^{M} \boldsymbol{W}_m \Big[\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlk} \cdot \boldsymbol{W}'_m \boldsymbol{x}^l \big(\phi_l(\boldsymbol{p}) + \Delta \boldsymbol{p}_{mlk}\big) \Big], \qquad (1)$$

                - ◇ where m indices the attention head and K is the total number of sampled keys for content aggregation. In addition, $\phi_l(\boldsymbol{p})$ is the reference point of the $[\mathtt{DET}]$ token re-scaled for the l-th level feature map, while $\Delta \boldsymbol{p}_{mlk}$ is the sampling offset for deformable attention; and A_{mlk} is the attention weights of the K sampled contents. $\boldsymbol{W}_m$ and $\boldsymbol{W}'_m$ are the projection matrices for multi-head attention.
        - □ 辅助技术
            - ◆ 辅助解码损失
            - ◆ 迭代的边界细化
    - ▪ 基于标签匹配的知识蒸馏
        - □ 目的：降低计算成本
        - □ 操作：
            - ◆ 教师模型：预训练的大型ViDT
            - ◆ 学生模型：需要学习的小型ViDT
            - ◆ 将学生模型的【Patch】和【DET】与教师模型的【Patch】和【DET】匹配，将教师模型的知识转移到学生模型

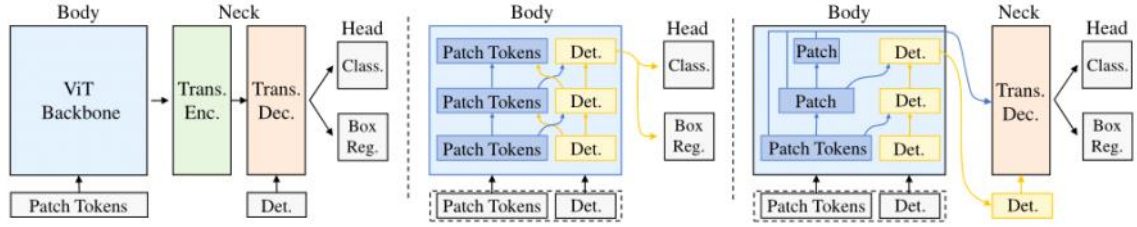
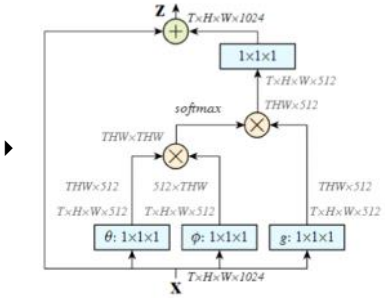
- $$\ell_{dis}(\mathcal{P}_s, \mathcal{D}_s, \mathcal{P}_t, \mathcal{D}_t) = \lambda_{dis}\Big(\frac{1}{|\mathcal{P}_s|}\sum_{i=1}^{|\mathcal{P}_s|}\Big\|\mathcal{P}_s[i] - \mathcal{P}_t[i]\Big\|_2 + \frac{1}{|\mathcal{D}_s|}\sum_{i=1}^{|\mathcal{D}_s|}\Big\|\mathcal{D}_s[i] - \mathcal{D}_t[i]\Big\|_2\Big), \qquad (2)$$
  - 结构图
  - 

Figure 2. Pipelines of fully transformer-based object detectors. DETR (ViT) means Detection Transformer that uses ViT as its body. The proposed ViDT synergizes DETR (ViT) and YOLOS and achieves best AP and latency trade-off among fully transformer-based object detectors.

- Transformer和CNN的区别
  - Transeformer仍属于机器学习，没有卷积，池化等操作
  - 输入形式
    - Transformer：一维序列
    - CNN：二维图像
  - 建模方式
    - Transformer：利用数据之间的全局相似性，更好的建立长程依赖
    - CNN：关注数据的局部关联性，更多的关注数据的空间结构信息
  - 先验知识
    - Transformer：位置编码信息
    - CNN：空间结构信息，平移不变性
- Transformer中的Q、K、V的含义
  - Q：Question查询向量
  - K：表示被查询的关键信息
  - V：表示被查询向量
  - 通俗解释：
  - 类似于搜索过程
  - 在网上搜索问题Q，每篇文章V都有对应的标题K，搜索引擎将问题Q与每篇文章V的标题K进行匹配，查看相关度QK，再对这些不同相关度的文章V进行加权求和得到一个新的Q'，而Q'融合了相关性强的文章V的更多信息，而融合了相关性弱的文章V的较少信息
- Transformer和Non-Local的区别
  - 相同点
    - 两者都基于注意力机制，有利于建立长距离依赖关系
  - 不同点
    - Transformer给出了一种建模方案，而Non-local只是自注意力机制的一种组件化模块（算子），即插即用
    - Non-local依旧作用于二维图像上，Transformer作用于一维序列上，两者作用在不同的输入模态上
  - Non-Local结构图
  - 

- Transformer中为什么用LN不用BN
  - Transformer用于处理文本数据，一个Batch_size对应着句子长度，每个样本相当于一个单词，通常来说，单词长度是不固定的
  - BN
    - BN是对多个样本的同一属性进行标准化处理
    - 如果使用BN层可以保证靠前的单词字符可以做相同的均值方差操作，而靠后的多余数据不能进行此操作，作使得句子与句子之间的关

联性消失

- LN
  - LN是对同一样本的所有属性进行标准化处理
  - LN操作依旧保留了句子的原始长度，使得句子中前后词之间的关联性更强

- 匈牙利算法
  - 作用：一种求解任务分配问题（指派问题）的组合优化算法
  - 指派问题模型
    -