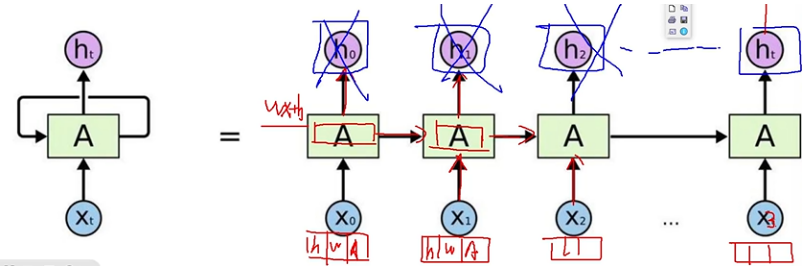
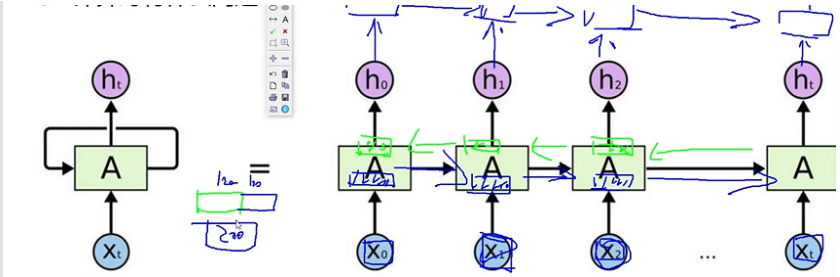
Transformer

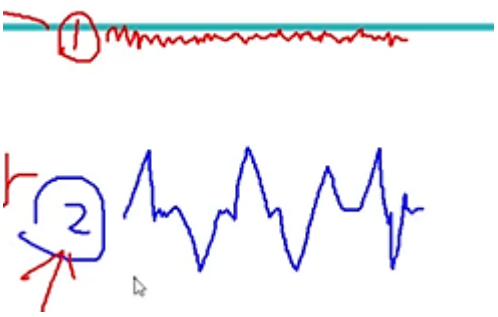
- 传统的RNN网络
  - 循环神经网络RNN 是一类用于处理序列数据的神经网络。具有循环结构，能够在处理序列的每个时间步时，将当前时间步的输入和前一个时间步的隐藏状态结合起来，从而对序列中的历史信息进行记忆和利用。
    - 从数学模型角度看，设输入序列为$x=(x_1,x_2,\cdots,x_T)$，在每个时间步t，隐藏状态h_t的更新公式为$h_t=f(Ux_t+Wh_{t-1}+b)$，其中U和W是权重矩阵，b是偏置向量，f是激活函数（如 tanh 函数）。h_t包含了序列从开始到时间步t的信息，最后可以通过输出层将隐藏状态转换为输出$y_t=g(Vh_t+c)$，其中V是输出层的权重矩阵，c是偏置向量，g是输出层的激活函数。
  - 实际输出时，看最后一个，考虑到前面的。问题：前面的东西不一定都能记住；好的坏的都会考虑进去；串行的计算（时间长）



    - 双向的（看过去也展望未来）

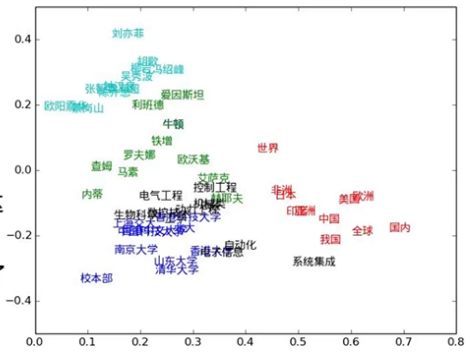


  - Attention is all you need（2017）（Transformer前身）
- AI本质-提特征学特征；transfrmer是“明显”特征



- 传统的word2vec

传统的word2vec

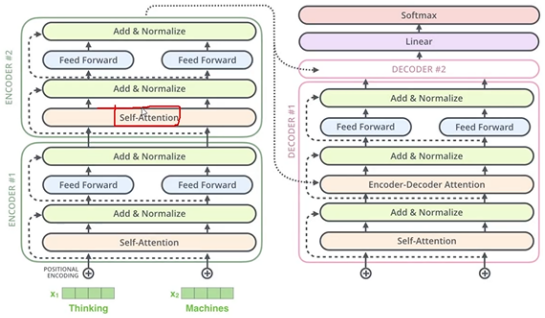


- 表示向量时有什么问题?
- 如果‘干哈那’是一个词
- 不同语境中相同的词如何表达
- 预训练好的向量就永久不变了

- 词向量，每个词都是一个向量
- 训练模型更新的是权重
- transformer既更新权重，又联系上下文

- transformer整体架构

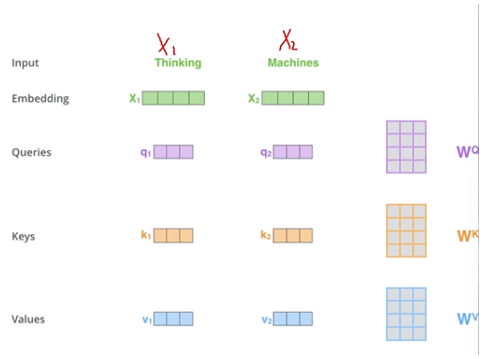
✓ 整体架构



- 输入如何编码?
- 输出结果是什么?
- Attention的目的?
- 怎样组合在一起?

- 注意力机制
  - 对于输入的数据，你的关注点（注意力），让计算机关注到这些有价值的信息
  - X1与X2的关系，内积为0代表关系极差，越大代表关系越好

✓ self-attention如何计算?



- 输入经过编码后得到向量
- 想得到当前词语上下文的关系，可以当作是加权
- 构建三个矩阵分别来查询当前词跟其他词的关系，以及特征向量的表达。

    - 使用内积(torch.matmul)算权重
      - 计算时，需要构建三个辅助向量，queries、keys、values

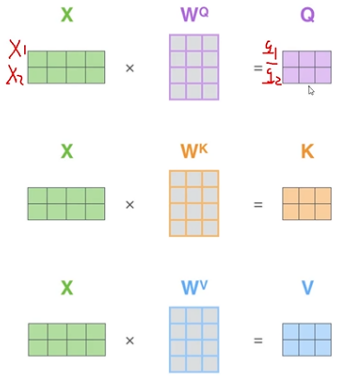
self-attention如何计算?

三个需要训练的矩阵

Q: query，要去查询的

K: key，等着被查的

V: value，实际的特征信息



- q“抛出一个问题”，k“答应一下”，v“表示自身特征”
- 由x1出发问跟x1的关系，q1·k1；x1->x2 q1·k2；x2->x1 q2·k1

- query、key、value通过矩阵乘法xw=q得到，通过额外全连接，设置权重参数得到

三个需要训练的矩阵

Q: query，要去查询的

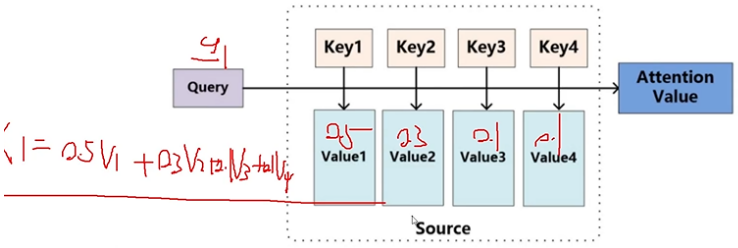
K: key，等着被查的

V: value，实际的特征信息

- value代表自身特征，也要去学习，通过学习更好认识自己

每个词的Attention计算

每个词的Q会跟整个序列中每一个K计算得分，然后基于得分再分配特征

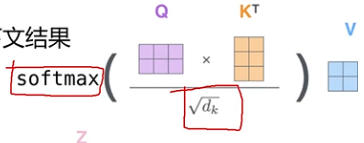


- x向量维度越大，内积越大，特征越重要。根号dk看成向量维度。**softmax用于归一化。**

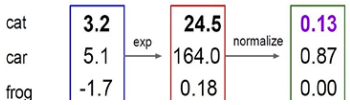
最终的得分值经过softmax就是最终上下文结果

Scaled Dot-Product Attention

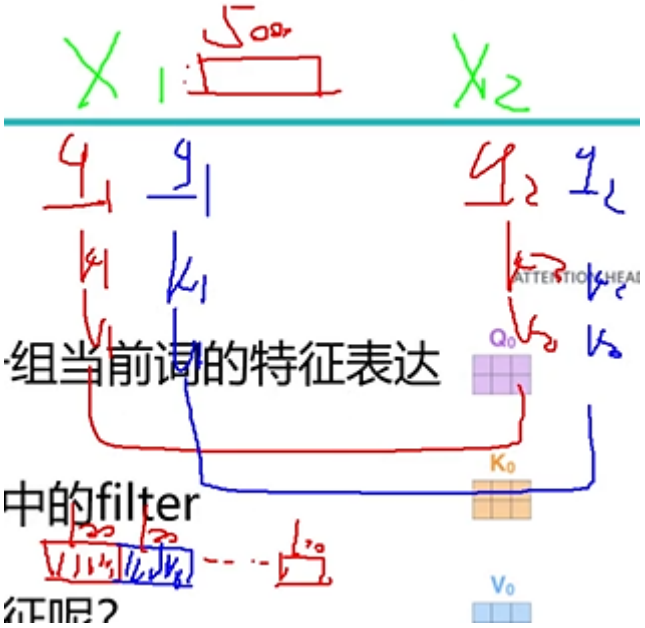
不能让分值随着向量维度的增大而增加



softmax回忆：



- 多头注意力机制
  - 头越多，学的东西越多，学到更多的特征
  - eg. x1要得到500维向量，得到五组qk，每组100维，得到500维

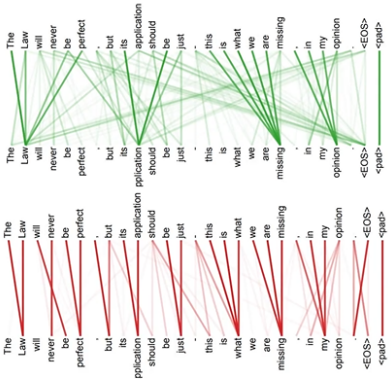


  - 把特征做的更丰富多样

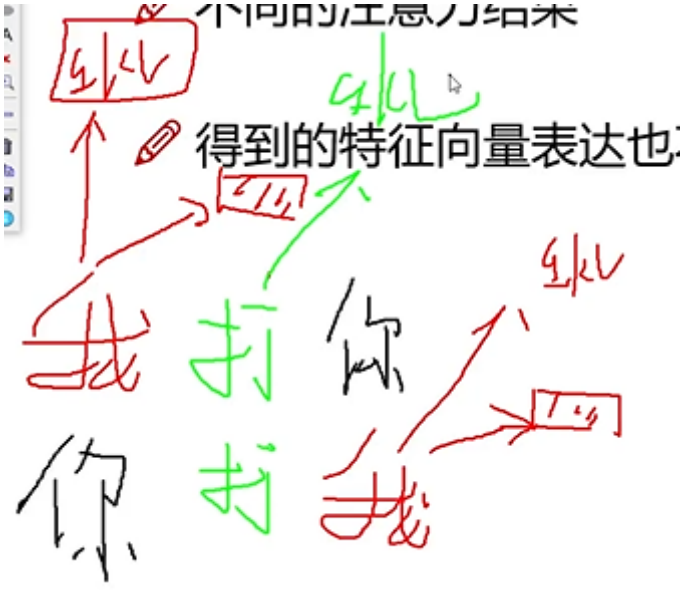
multi-headed结果

不同的注意力结果

得到的特征向量表达也不相同



  - “我”更新后的特征不会变--qkv只跟数据本身有关系，你在哪不影响，这样会导致计算机认为我打你=你打我



- 位置信息表达（transformer需要额外作位置编码，其本身对位置不敏感）

  ✓ 位置信息表达

  ✎ 在self-attention中每个词都会考虑整个序列的加权，所以其出现位置并不会对结果产生什么影响，相当于放哪都无所谓，但是这跟实际就有些不符合了，我们希望模型能对位置有额外的认识。

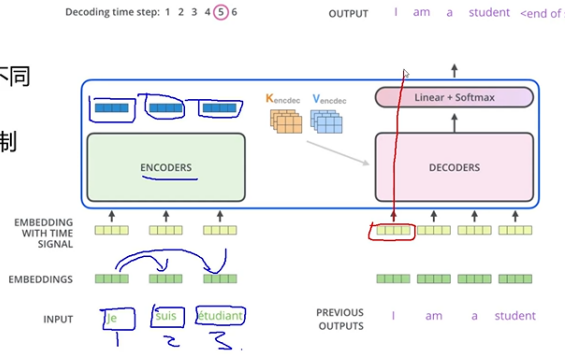
- self-attention在输入和输出都是词特征，只不过加上了语境
  - 一层够吗？能不能连多层-堆叠多层

- Decoder （解码层）

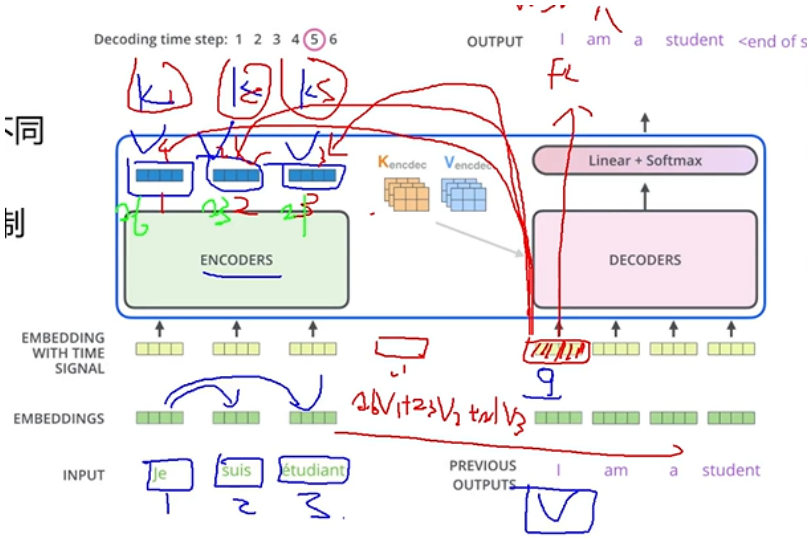
✓ Decoder

✎ Attention计算不同

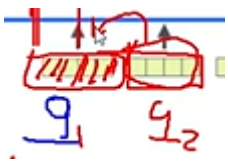
✎ 加入了MASK机制



- q-d;k-e;v-e



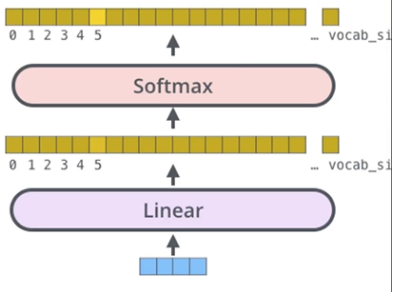
  - 考虑q2时需要与q1去看关系；



    - 那考虑q1时需要与q2看关系吗？不需要，看不到，因为q2相对q1是未来——**MASK机制**

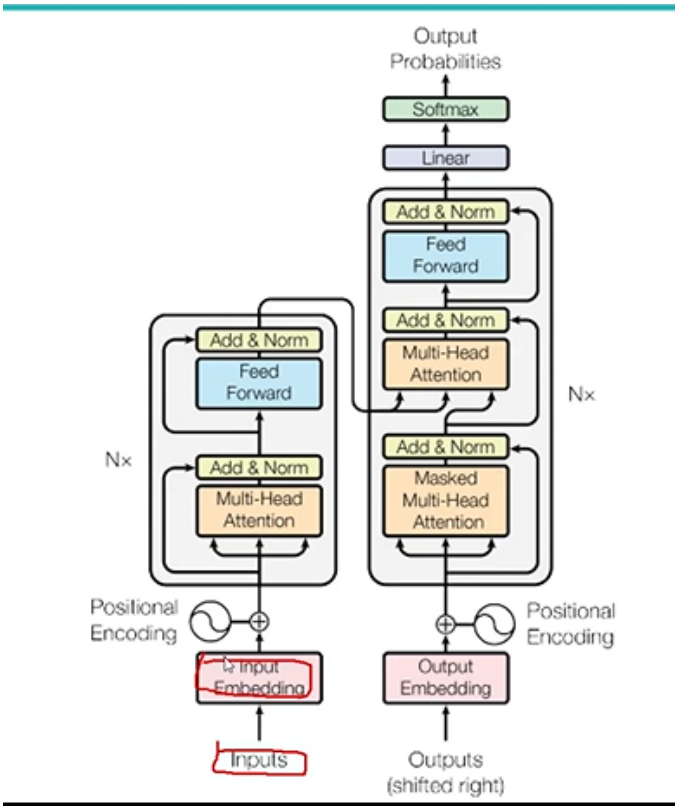
- 最终输出结果

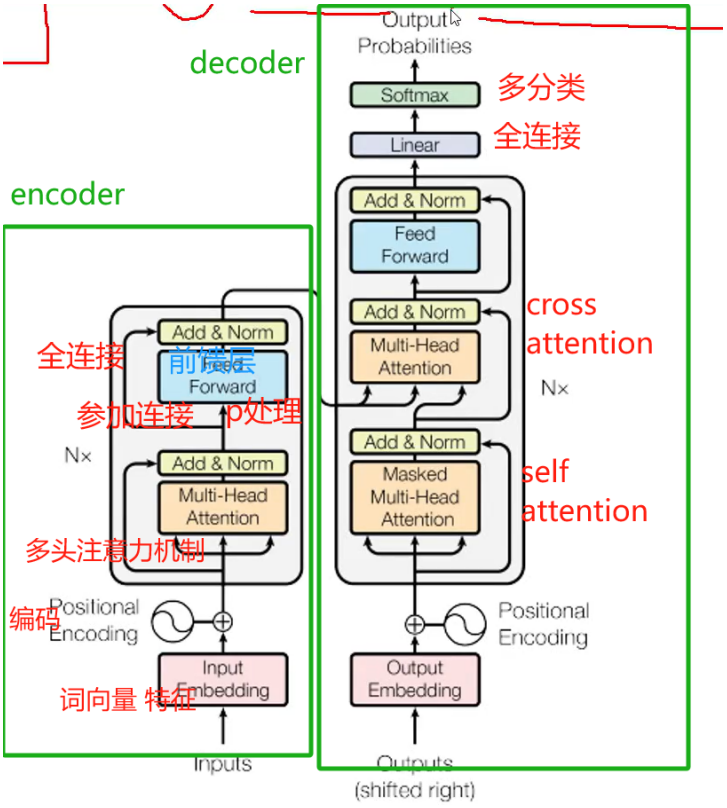
- 最终输出结果
- 得出最终预测结果
- 损失函数 cross-entropy即可



- 文本任务-分类；视觉任务-回归

- 整体梳理





- self与cross
  - Self - attention（自注意力机制） 是 Transformer 架构中的一个关键组件。它的主要目的是计算序列中每个元素与其他元素之间的关联程度，从而为每个元素生成一个考虑了整个序列信息的表示。在自然语言处理中，当处理一个句子时，self - attention 可以让模型学习到句子中每个单词与其他单词之间的语义关系。例如在机器翻译任务中，对于输入的源语言句子，self - attention 能够捕捉单词之间的语法结构和语义依赖。

    具体地，计算注意力权重α_{ij}的公式为$\alpha_{ij}=\frac{\exp(q_i\cdot k_j)}{\sum_{k=1}^{n}\exp(q_i\cdot k_k)}$，其中$q_i\cdot k_j$表示查询向量$q_i$和键向量$k_j$的点积。这个权重表示了元素$x_j$对于元素$x_i$的重要性。然后通过对所有的值向量$v_j$进行加权求和得到输出向量$y_i=\sum_{j=1}^{n}\alpha_{ij}v_j$。
  - Cross - attention （交叉注意力机制）涉及两个不同的序列。它的目的是让一个序列中的元素关注另一个序列中的元素。（解码过程中涉及到）

    计算注意力权重$\beta_{ij}=\frac{\exp(q_i\cdot k_j)}{\sum_{k=1}^{n}\exp(q_i\cdot k_k)}$，然后输出向量$z_i=\sum_{j=1}^{n}\beta_{ij}v_j$。这样，序列$\mathbf{X}$中的每个元素就可以根据与序列$\mathbf{Y}$中元素的关联程度来更新自己的表示。
    - 多模态-综合利用不同种类特征。在图像 - 文本跨模态任务中，比如图像字幕生成。图像特征序列和文本特征序列分别作为两个不同的序列。Cross - attention 可以让文本序列中的单词关注图像中的相关区域，从而生成更准确的图像字幕。
- 整体再梳理

- 输入处理
  - **输入嵌入（Input Embedding）**：将输入的词或子词转换为固定维度的向量表示。
  - **位置编码（Positional Encoding）**：由于 Transformer 不使用循环或卷积结构，无法直接捕捉序列的顺序信息，因此通过位置编码将位置信息注入到输入中。位置编码通常使用正弦和余弦函数生成。
- 编码器（Encoder）
  - 编码器由 N个相同的层堆叠而成（通常 N=6），每一层包含两个主要子层：
    - **多头自注意力机制（Multi-Head Self-Attention）**：
      - 计算输入序列中每个位置与其他位置的相关性。
      - 通过多个注意力头（Attention Heads）捕捉不同的语义信息。
    - **前馈神经网络（Feed-Forward Neural Network, FFN）**：
      - 对每个位置的表示进行非线性变换。
      - 通常由两个全连接层和激活函数（如 ReLU）组成。
  - 每个子层后都接有：
    - **残差连接（Residual Connection）**：将输入直接加到输出上，缓解梯度消失问题。
    - **层归一化（Layer Normalization）**：对输出进行归一化，加速训练。
- 解码器
  - 解码器也由 N个相同的层堆叠而成，每一层包含三个主要子层：
    - **掩码多头自注意力机制（Masked Multi-Head Self-Attention）**：
      - 与编码器的自注意力机制类似，但加入了掩码（Mask），防止解码器在预测时看到未来的信息。
    - **编码器-解码器注意力机制（Encoder-Decoder Attention）**：
      - 解码器通过该机制关注编码器的输出，捕捉输入序列和输出序列之间的关系。
    - **前馈神经网络（Feed-Forward Neural Network, FFN）**：
      - 与编码器中的 FFN 结构相同。
- 输出处理
  - **线性层（Linear Layer）**：将解码器的输出映射到词汇表大小的向量。
  - **Softmax 层**：将线性层的输出转换为概率分布，用于预测下一个词。
    - Softmax是一种激活函数，它可以将一个数值向量归一化为一个概率分布向量，且各个概率之和为1。Softmax可以用来作为神经网络的最后一层，用于多分类问题的输出。Softmax层常常和交叉熵损失函数一起结合使用。

- BERT
  - 不需要标注，无监督

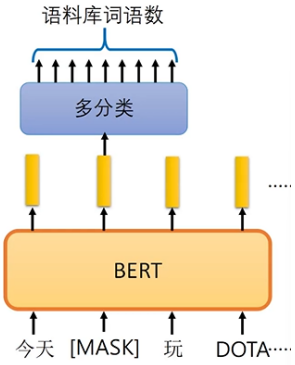
如何训练BERT

方法1：句子中有15%的词汇被随机mask掉

交给模型去预测被mask的家伙到底是什么

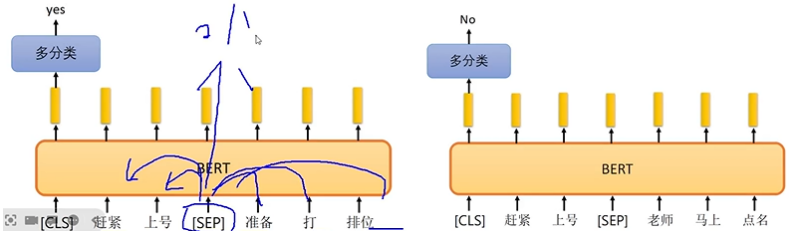
词语的可能性太多了，中文一般是字

如果BERT训练的向量好，那分类自然OK



方法2：预测两个句子是否应该连在一起

[seq]：两个句子之前的连接符，[cls]：表示要做分类的向量



- 锻炼学习能力——趋于多模态

- 一个典型的 Transformer 架构主要由**编码器（Encoder）**和**解码器（Decoder）**两部分组成。编码器用于对输入序列进行特征提取和编码，解码器则用于根据编码器的输出生成目标序列。
  - 编码器包含多个相同的编码器层，每个编码器层主要由**多头自注意力机制（Multi - Head Self - Attention）和前馈神经网络（Feed - Forward Neural Network）**构成。多头自注意力机制能够并行地从不同的表示子空间中计算输入序列中每个元素与其他元素之间的关联程度，从而捕捉序列中的语义关系。前馈神经网络则用于进一步对自注意力机制的输出进行非线性变换。
  - 解码器也包含多个解码器层，除了具有多头自注意力机制和前馈神经网络外，还包含一个**交叉注意力机制（Cross - Attention）**。交叉注意力机制用于让解码器中的元素关注编码器输出的信息，从而更好地生成目标序列。
- Transformer进军视觉

Transformer

为啥这么火呢?

NLP领域大哥大级别，一统天下好多年了

CV界新秀，开场即巅峰；满级大号直接上场

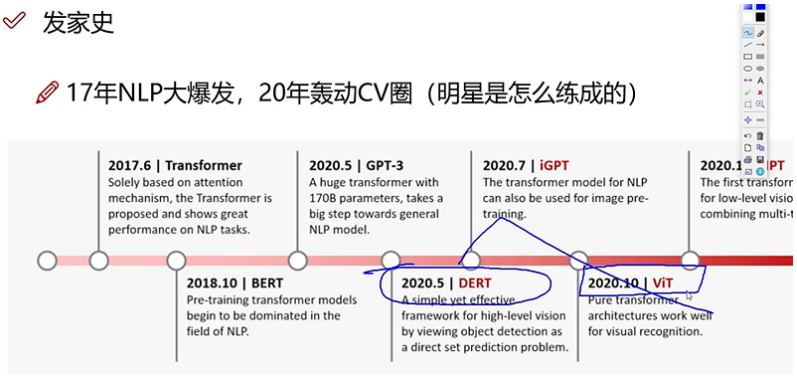
新一代backbone，可直接套用在各项下游任务中

分类，分割，检测各项任务均刷榜；

- 历史

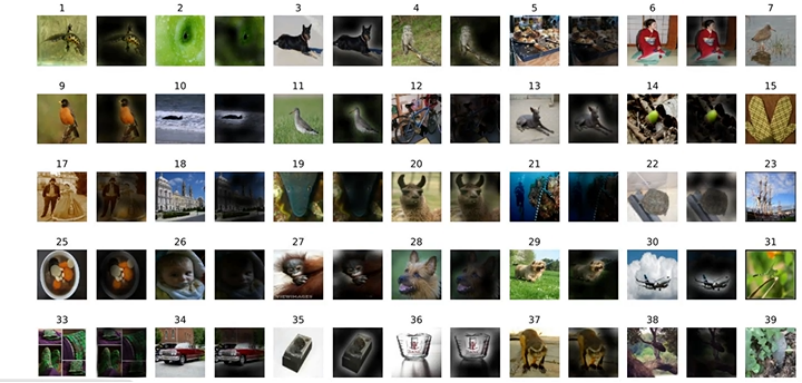
发家史

17年NLP大爆发，20年轰动CV圈（明星是怎么练成的）



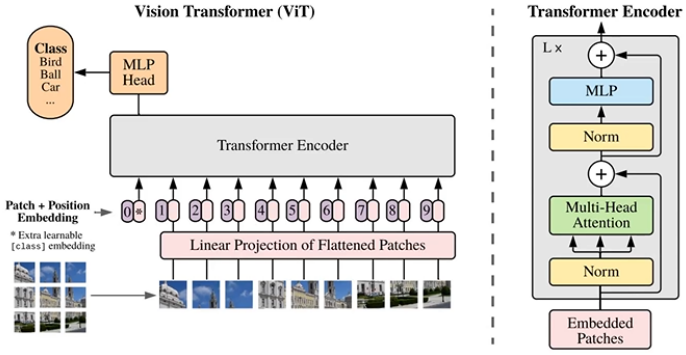
- 视觉中的Attention
  - 什么重要什么不重要

    视觉中的Attention



- 整体结构

  整体架构分析



  - 文本是转成向量，图像呢。先划分区域，对区域提特征（**卷积**），转化为向量
- 归一化Batch Norm和Ayear Norm
  - **Batch Normalization (Batch Norm)** 和 **Layer Normalization (Layer Norm)** 是两种常用的神经网络归一化方法，它们在神经网络训练中帮助提高收敛速度和稳定性，但两者的机制和适用场景有所不同。
  - Batch Normalization (Batch Norm)
    - 对一个小批量样本（batch）的激活值进行归一化。通过对每个 batch 内的神经元输出计算均值和标准差，然后进行归一化，使激活值具有零均值和单位方差。
    - 公式

- **公式**：

$$\hat{x}_i = \frac{x_i - \mu_{\text{batch}}}{\sigma_{\text{batch}} + \epsilon}, \quad y_i = \gamma \hat{x}_i + \beta$$

其中：

- x_i：输入激活值
- $\mu_{\text{batch}}, \sigma_{\text{batch}}$：batch 内的均值和标准差
- γ, β：可学习参数，用于缩放和平移归一化后的值
- ϵ：一个小常数，防止分母为 0

- 主要用于卷积神经网络（CNN）和其他以 batch 为中心的训练任务。在大 batch size 训练中效果显著。

- Layer Normalization (Layer Norm)
  - 对每个样本的激活值进行归一化，而不是依赖 batch 维度。对输入的特征维度（每一层的神经元）进行均值和标准差归一化，使每个样本都独立归一化。（**自己家处理自家的事情**）
  - 公式
    - **公式**：

$$\hat{x}_i = \frac{x_i - \mu_{\text{layer}}}{\sigma_{\text{layer}} + \epsilon}, \quad y_i = \gamma \hat{x}_i + \beta$$

其中：

- $\mu_{\text{layer}}, \sigma_{\text{layer}}$：基于输入特征维度计算的均值和标准差
- γ, β：可学习的缩放和平移参数

  - 主要用于循环神经网络（RNN）、Transformer 等 NLP 任务。对小 batch size 或单个样本任务更加稳定。