Conformer论文阅读

Abstract

“Transformer models are good at capturing content-based global interactions, while CNNs exploit local features effectively.”

（1）Transformer 模型擅长捕捉文本中基于内容的全局交互。

为什么?

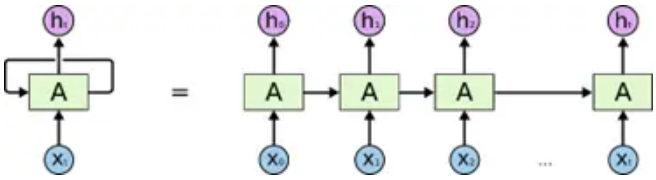
> https://blog.csdn.net/jarodyv/article/details/130867562
> 文章浏览阅读7.8k次，点赞62次，收藏139次。本文几乎涵盖了关于 Transformer 和注意力机制的所有必要...
> https://blog.csdn.net/jarodyv/article/details/130867562

要想理解Transformer，得先学习一下RNN，因为Transformer是对RNN的改良与优化。

传统前馈神经网络不考虑数据之间的关联性，网络的输出只和当前时刻网络的输入相关。（可以这样理解：经典的深度学习入门案例：玫瑰花识别or手势识别or肿瘤识别，都是只针对特定的输入，返回一个预测的结果）。即没有时序性。

RNN是一个具有记忆能力，支持序列数据的神经网络。其本质是：拥有记忆的能力，并且会根据这些记忆的内容来进行推断。因此，它的输出就依赖于当前的输入和记忆。

RNN包括三个部分：**输入层、隐藏层和输出层**。相对于前馈神经网络，RNN可以接收上一个时间点的隐藏状态(将说却没有说的数据，可以理解为在说话的某一瞬间，想表达什么，又不能完整的表达，只能遗留到下一时刻才能被完整的说出)。



RNN 的神经网络单元不但与输入和输出存在联系，而且自身也存在一个循环 / 回路 / 环路 / 回环 (loop)

RNN的问题：梯度消失（梯度爆炸）

循环神经网络的每个训练样本是一个时间序列，同一个训练样本前后时刻的输入值之间有关联，每个样本的序列长度可能不相同。训练时先对这个序列中的每个时刻的输入值进行正向传播，再通过反向传播计算出参数的梯度值并更新参数。

循环神经网络在进行反向传播时也面临梯度消失或者梯度爆炸问题，这种问题表现在时间轴上。如果输入序列的长度很长，人们很难进行有效的参数更新。通常来说梯度爆炸更容易处理一些。梯度爆炸时我们可以设置一个梯度阈值，当梯度超过这个阈值的时候可以直接截取。

Transformer 模型的精髓：**多头自注意力机制**

多头自注意力机制能够**直接建模任意距离**的词元之间的交互关系。
传统神经网络：**循环神经网络**迭代地利用前一个时刻的状态更新当前时刻的状态，因此在处理较长序列的时候，常常会出现**梯度爆炸或者梯度消失**的问题。而在**卷积神经网络**中，只有位于同一个卷积核的窗口中的词元可以直接进行交互，通过**堆叠层数（层数增加）**来实现远距离词元间信息的交换。

多头自注意力机制通常由多个自注意力模块组成。在每个自注意力模块中，对于输入的词元序列，将其映射为相应的查询（Query, $\boldsymbol{Q}$）、键（Key, $\boldsymbol{K}$）和值（Value, $\boldsymbol{V}$）三个矩阵。然后，对于每个查询，将和所有没有被掩盖的键之间计算点积。这些点积值进一步除以 $\sqrt{D}$ 进行缩放（D 是键对应的向量维度），被传入到 softmax 函数中用于权重的计算。进一步，这些权重将作用于与键相关联的值，通过加权和的形式计算得到最终的输出。在数学上，上述过程可以表示为：

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q}, \tag{5.2}$$

$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K}, \tag{5.3}$$

$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V}, \tag{5.4}$$

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{D}})\boldsymbol{V}. \tag{5.5}$$

与单头注意力相比，多头注意力机制的主要区别在于它使用了 H 组结构相同但映射参数不同的自注意力模块。输入序列首先通过不同的权重矩阵被映射为一组查询、键和值。每组查询、键和值的映射构成一个“头”，并独立地计算自注意力的输出。最后，不同头的输出被拼接在一起，并通过一个权重矩阵 $\boldsymbol{W}^{O} \in \mathbb{R}^{H \times H}$ 进行映射，产生最终的输出。如下面的公式所示：

$$\text{MHA} = \text{Concat}(\text{head}_1, \ldots, \text{head}_N)\boldsymbol{W}^{O}, \tag{5.6}$$

$$\text{head}_n = \text{Attention}(\boldsymbol{X}\boldsymbol{W}_n^{Q}, \boldsymbol{X}\boldsymbol{W}_n^{K}, \boldsymbol{X}\boldsymbol{W}_n^{V}). \tag{5.7}$$

（2）卷积神经网络 (CNN) 擅长高效地提取和利用局部特征（通过卷积？）

这篇工作综合了Transformer和CNN的优点。
“In this work, we achieve the best of both worlds by studying how to combine convolution neural networks and transformers to model both local and global dependencies of an audio sequence in a parameter-efficient way”

Introduction

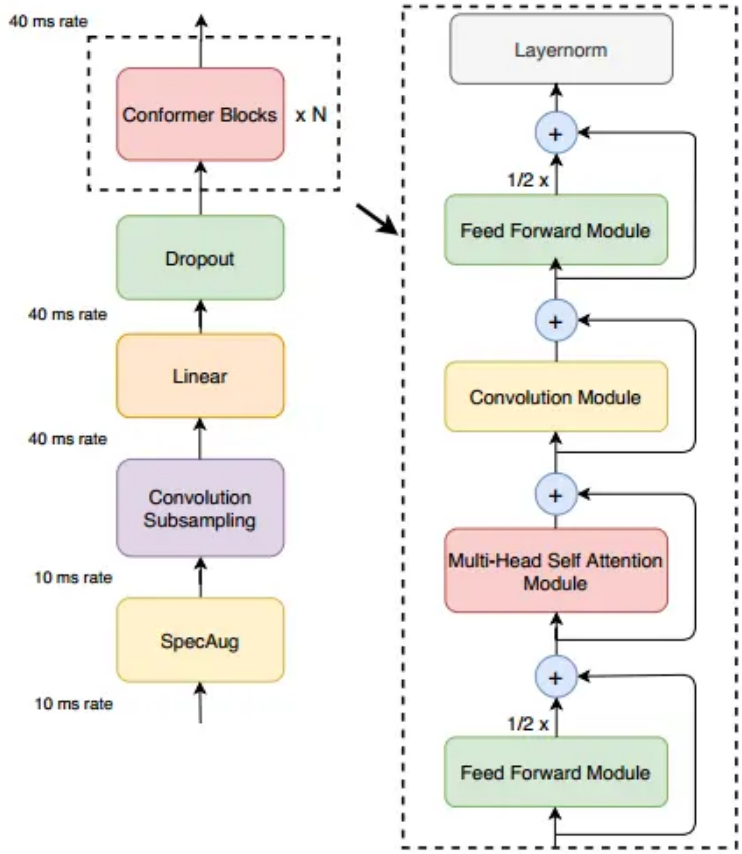
虽然transformer擅长对远程全局上下文进行建模，但它们在提取细粒度的局部特征模式方面能力较差。

“While Transformers are good at modeling long-range global context, they are less capable to extract finegrained local feature patterns.”

CNN在捕获动态全局上下文方面仍然有限
“However, it is still limited in capturing dynamic global context”

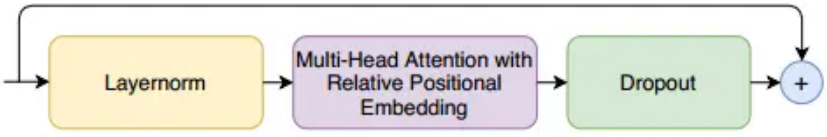
工作贡献：提出了一种新颖的自注意和卷积的组合，将实现两全其美——**自注意学习全局交互**，而**卷积**有效地捕获基于**相对偏移量的局部相关性**。

Conformer Encoder



Conformer模块由两个前馈模块、多头自注意力模块、卷积模块组成。

多头自注意力模块(Multi-Headed Self-Attention Module)



1 多头自注意力模块

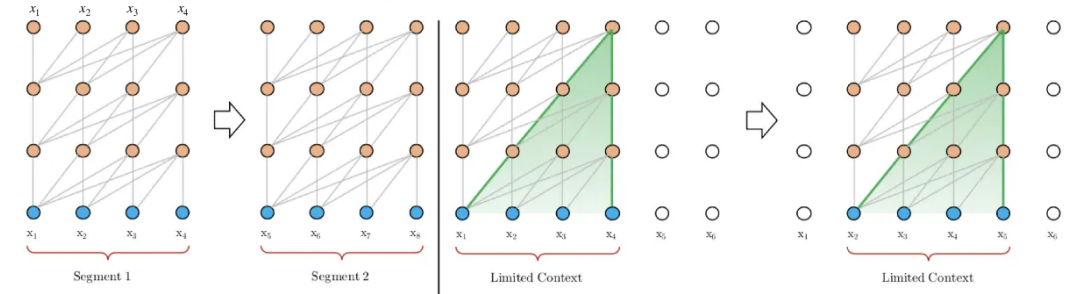
略

2 相对正弦位置编码 (Transformer-XL)

有缺陷的vanilla Transformer

Al-Rfou等人基于Transformer提出了一种训练语言模型的方法vanilla Transformer，来根据之前的字符预测片段中的下一个字符。

一个字符。例如，它使用$x_1, x_2, ..., x_{n-1}$预测字符x_n，而在x_n之后的序列则被mask掉。论文中使用64层模型，并仅限于处理 512个字符这种相对较短的输入，因此它将输入分成段，并分别从每个段中进行学习，如下图所示。 在测试阶段如需处理较长的输入，该模型会在每一步中将输入向右移动一个字符，以此实现对单个字符的预测。



(a) Training phase. (b) Evaluation phase.

这个方法的缺点

a. 上下文长度受限：字符之间的最大依赖距离受输入长度的限制，模型看不到出现在几个句子之前的单词。

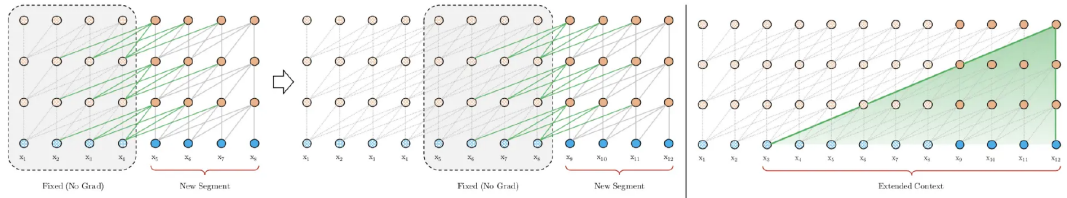
b. 上下文碎片：对于长度超过512个字符的文本，都是从头开始单独训练的。段与段之间没有上下文依赖性，会让训练效率低下，也会影响模型的性能。

c. 推理速度慢：在测试阶段，每次预测下一个单词，都需要重新构建一遍上下文，并从头开始计算，这样的计算速度非常慢。

Transformer-XL是对以上方法的改善。

Transformer-XL架构在vanilla Transformer的基础上引入了两点创新：循环机制（Recurrence Mechanism）和相对位置编码（Relative Positional Encoding）。

循环机制：与vanilla Transformer的基本思路一样，Transformer-XL仍然是使用分段的方式进行建模，但其与vanilla Transformer的本质不同是在于引入了段与段之间的循环机制，使得当前段在建模的时候能够利用之前段的信息来实现长期依赖性。



(a) Training phase. (b) Evaluation phase.

在训练阶段，处理后面的段时，每个隐藏层都会接收两个输入：

1. 该段的前面隐藏层的输出，与vanilla Transformer相同（上图的灰色线）。
2. 前面段的隐藏层的输出（上图的绿色线），可以使模型创建长期依赖关系。

相对位置编码：

Vanilla Transformer使用的是绝对位置编码，其计算方式如下，pos表示的是token的下标，d_{model}表示的是hidden size，i表示的是具体的某个维度。

$$\begin{cases} PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \\ PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \end{cases}$$

可见，不同的片段的同一个位置其位置编码都是一样的，模型没办法正确区分不同片段的位置信息。**这种方法在处理长序列时有局限性，因为位置编码的模式可能会在长序列中重复，导致模型难以区分距离较远的元素之间的关系。**

在Transformer-XL中，相对位置编码通过维护一个额外的相对位置嵌入矩阵来实现。这个矩阵的每一行对应一个特定的相对位置，而不是绝对位置。例如，如果我们要计算两个单词之间的注意力权重，我们会考虑它们之间的相对距离，而不是它们各自的绝对位置。

假设我们有两个位置*i*和*j*，那么它们的相对位置可以表示为*r=i−j*。然后，我们可以从相对位置嵌入矩阵中查找位置*r*对应的嵌入向量，该向量会用于计算注意力分数。
具体地，在注意力机制中，原始的注意力分数计算方式为：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

在引入相对位置编码后，注意力分数的计算会考虑相对位置信息，通常通过在查询*Q*和键*K*的点积中加入相对位置嵌入*Er*来实现：

$$\text{Attention}(Q, K, V, E_r) = \text{softmax}\left(\frac{(Q + E_r)K^T}{\sqrt{d_k}}\right)V$$

其中*Er*是相对位置嵌入向量，它取决于查询和键之间的相对位置。

Transformer-XL的**优势**

- 长距离依赖捕获：相对位置编码允许模型更好地理解长序列中的长距离依赖关系，这对于诸如文本生成、语音识别等任务至关重要。
- 泛化能力：由于相对位置编码不依赖于序列的绝对位置，因此模型在处理未知长度的序列时具有更好的泛化能力。

3 带有dropout的prenorm残差单元

在传统的深度神经网络中，随着网络层数的增加，模型可能会遇到梯度消失或梯度爆炸的问题，这使得网络难以训练。残差单元通过引入残差连接来缓解这些问题。残差连接将网络中某一层的输入直接添加到几层之后的输出上，形成一种跳跃连接的方式。这样做的目的是让网络学习残差函数，而不是直接学习从输入到输出的映射。

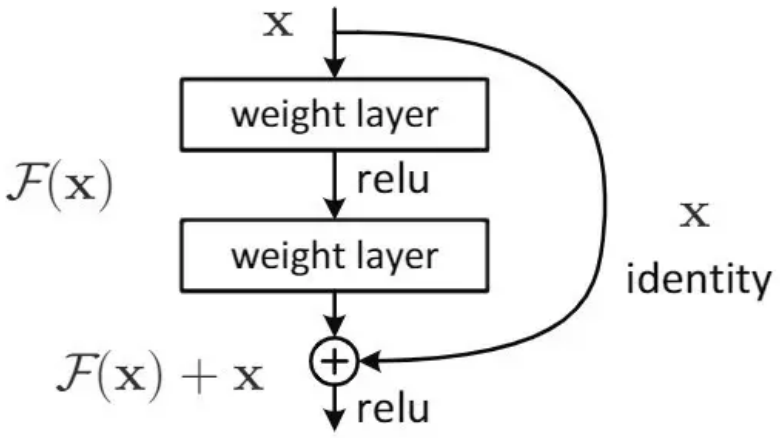


Figure 2. Residual learning: a building block.

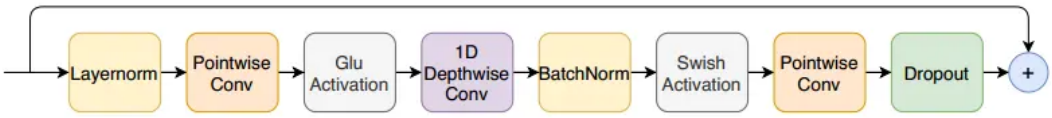
Prenorm是在残差连接之前，先进行层归一化处理。层归一化的目标是对每个样本的特征进行归一化，使其均值接近0，方差接近1，用以控制梯度的规模，使得梯度在反向传播过程中更加稳定。

4 Dropout技术

Dropout是一种正则化技术，通过在训练过程中随机丢弃一部分神经元来防止模型过拟合。

卷积模块

卷积模块包括一个门控机制(一个逐点卷积和一个门控线性单元(GLU))，一个单一的1-D深度卷积层。Batchnorm在卷积之后部署，以帮助训练深度模型。

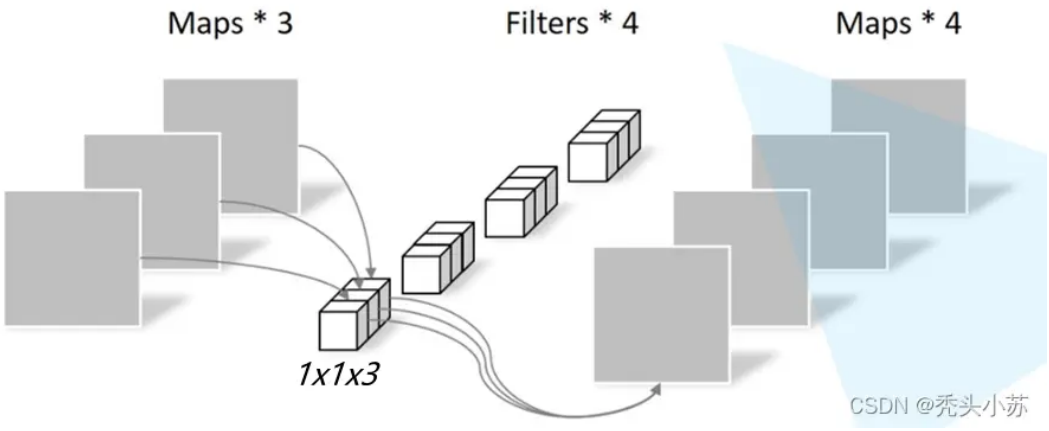


1 逐点卷积

逐点卷积（Pointwise Convolution），也被称为 1×1 卷积，是一种特殊的卷积操作，其中卷积核的尺寸为 1×1。这意味着卷积核只包含单个像素值，因此不会在空间维度上进行任何操作，而是只对输入特征图的通道进行线性组合。

当使用逐点卷积来增加通道数时，有以下几个好处：

1. 增加非线性映射能力：
    a. 通过增加通道数，模型可以学习更多的特征映射，从而增强网络的表达能力。这有助于捕捉更复杂的特征组合和模式，提高模型的准确性。
2. 跨通道信息融合：
    a. 逐点卷积允许网络在不同的输入通道之间进行信息的重新分配和组合。这有助于模型学习到不同特征之间的相关性，从而更好地理解输入数据。



3个特征图【或者可以说是一个有三个通道的特征图】，卷积核大小为1x1x3,卷积核个数为4，这样卷积后就可以得到4个特征图【得到特征图的个数取决于卷积核的个数】。

2 门控线性单元(GLU)

GLU的核心思想是引入门控机制来控制信息的流动。它由两个部分组成：门控部分和线性部分。对于一个输入x，GLU通过一个门控函数将输入分为两个部分，然后将其中一个部分与另一个部分进行逐元素相乘。这个过程可以用公式表示如下：

$$GLU(x) = x \otimes \sigma(Wx + b)$$

其中，$\otimes$表示逐元素相乘，σ表示sigmoid函数，W和b是GLU的参数。

GLU的优势在于它可以学习到输入数据的不同维度之间的交互关系，从而提高模型的表达能力。它可以帮助模型更好地捕捉词语之间的依赖关系和重要性。总之，GLU是一种有效的激活函数，通过引入门控机制来控制信息流动，提高神经网络模型的表达能力。

3 Batchnorm

批标准化的主要目的是解决所谓的内部协变量偏移问题（Internal Covariate Shift），即在训练过程中，由于前面层权重的变化导致后续层的输入分布发生变化，这会使得后续层需要不断调整以适应新的输入分布。通过批标准化，每一层的输入都会被标准化，从而使得训练更加稳定和快速。

前馈模块

为了学习复杂的函数关系和特征，Transformer 模型引入了一个前馈网络层（Feed Forward Netwok, FFN），对于每个位置的隐藏状态进行非线性变换和特征提取。具体来说，给定输入 $\boldsymbol{x}$，Transformer 中的前馈神经网络由两个线性变换和一个非线性激活函数组成:

$$\mathrm{FFN}(\boldsymbol{X}) = \sigma(\boldsymbol{X}\boldsymbol{W}^U + \boldsymbol{b}_1)\boldsymbol{W}^D + \boldsymbol{b}_2, \tag{5.8}$$

其中 $\boldsymbol{W}^U \in \mathbb{R}^{H\times H'}$ 和 $\boldsymbol{W}^D \in \mathbb{R}^{H'\times H}$ 分别是第一层和第二层的线性变换权重矩阵，$\boldsymbol{b}_1 \in \mathbb{R}^{H'}$ 和 $\boldsymbol{b}_2 \in \mathbb{R}^H$ 是偏置项，σ 是激活函数（在原始的 Transformer 中，采用 ReLU 作为激活函数）。前馈网络层通过激活函数引入了非线性映射变换，提升了模型的表达能力，从而更好地捕获复杂的交互关系。

conformer在这一模块还应用了Swish激活dropout。

Swish 激活函数的定义

Swish 激活函数定义如下:
$\mathrm{Swish}(x) = x \cdot \sigma(x)$
其中 $\sigma(x)$ 是 Sigmoid 函数，定义为:
$\sigma(x) = \frac{1}{1+e^{-x}}$

Swish 激活函数的特点

1. **平滑且非单调**：Swish 函数是处处可导的，这意味着它没有突变点，有利于优化过程中的梯度传播。此外，它不是单调递增的，这意味着它可以产生负输出值，这有助于网络学习更复杂的非线性映射。
2. **无上界（Avoiding Gradient Saturation）**：与 ReLU 类似，Swish 函数的输出没有上界，这意味着它不会像 Sigmoid 或 Tanh 函数那样在较大的输入值处产生梯度饱和问题。
3. **有下界（Stronger Regularization Effect）**：Swish 函数的输出有下界，这与 ReLU 函数相似，但是由于乘以 Sigmoid 函数，Swish 函数的输出通常更接近于零，这有助于产生更强的正则化效果。
4. **参数化 Swish**：Swish 函数还可以被参数化，通过引入一个额外的参数 β 来控制 Sigmoid 函数的斜率:
   $\mathrm{Swish}_\beta(x) = x \cdot \sigma(\beta x)$