大模型面经——大模型中用到的归一化方法总结

原创 喜欢卷卷的瓦力 瓦力算法学研所 2024年03月26日 14:42 广东

◈ 技术总结专栏 ◈

作者：vivida

本文按归一化位置和方法作为分类条件，对大模型中归一化方法做总结梳理。

大模型中的归一化主要是为了解决LLM训练不稳定的问题。这里属于NLP领域，由于样本序列长度不一致，具体归一化跟CV领域存在区别，CV领域归一化方法总结可以看视觉面经之归一化篇。

长话短说，LLM中归一化方法可以按照归一化方法来分，主要分为LayerNorm，BatchNorm，RMSNorm以及DeepNorm。

还可以按照归一化位置来分类，包括 postNorm 和 preNorm，但具体来说用postNorm比较多，具体的原因可以看之前这一篇 为什么大模型结构设计中往往使用postNorm而不用preNorm？。

一、归一化方法

1. BatchNorm

BatchNorm主要对数据的一定维度在batch数据中进行归一，一般来说应用于图像。

这种方法很难适用于序列数据，对于序列数据而言，在batch维度做归一意义不大，而且一个batch内的序列长度不同。

2. LayerNorm

LayerNorm是针对序列数据提出的一种归一化方法，主要在layer维度进行归一化，即对整个序列进行归一化。layerNorm**会计算一个layer的所有activation的均值和方差，利用均值和方差进行归一化。**

有时候面试官会让写公式，所以还是需要对公式比较熟悉，具体公式如下：

$$\mu = \sum_{i=1}^{d} x_i$$

$$\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2}$$

归一化后的激活值如下：

$$\mathbf{y}=\frac{\mathbf{x}-\mu}{\sqrt{\sigma+\epsilon}}\gamma+\beta$$

其中 γ 和 β 是可训练的模型参数。 γ 是缩放参数，新分布的方差 γ^2 ； β 是平移系数，新分布的均值为 β 。 ϵ 为一个小数，添加到方差上，避免分母为0。

3. RMSNorm

RMSNorm的提出是为了提升layerNorm的训练速度提出的。RMSNorm也是一种layerNorm，只是归一化的方法不同。相比layerNorm中利用均值和方差进行归一化，**RMSNorm 利用均方根进行归一化**。

具体公式修改如下：

$$RMS(x)=\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}$$

归一化后的激活值如下：

$$y=\frac{x}{RMS(x)}\gamma$$

对于layerNorm和RMSNorm，layerNorm包含缩放和平移两部分，RMSNorm去除了平移部分，只保留了缩放部分。这样做的依据是，有研究认为layerNorm取得成功的关键是缩放部分的缩放不变性，而不是平移部分的平移不变性。

RMSNorm相比一般的layerNorm，减少了计算均值和平移系数的部分，训练速度更快，效果基本相当，甚至有所提升。

4. DeepNorm

DeepNorm是由微软提出的一种Normalization方法。主要对Transformer结构中的残差链接做修正。DeepNorm可以缓解模型参数爆炸式更新的问题，把模型参数更新限制在一个常数域范围内，使得模型训练过程可以更稳定。模型规模可以达到1000层。
DeepNorm兼具PreLN的训练稳定和PostLN的效果性能。

具体的实现，可以参照下图，DeepNorm对layerNorm之前的残差链接进行了up-scale，在初始化阶段down-scale了模型参数。GLM-130B 模型中就采用了DeepNorm。

```python
def deepnorm(x):
    return LayerNorm(x * α + f(x))

def deepnorm_init(w):
    if w is ['ffn', 'v_proj', 'out_proj']:
        nn.init.xavier_normal_(w, gain=β)
    elif w is ['q_proj', 'k_proj']:
        nn.init.xavier_normal_(w, gain=1)
```

Architectures	Encoder		Decoder	
	α	β	α	β
Encoder-only (e.g., BERT)	$(2N)^{\frac{1}{4}}$	$(8N)^{-\frac{1}{4}}$	-	-
Decoder-only (e.g., GPT)	-	-	$(2M)^{\frac{1}{4}}$	$(8M)^{-\frac{1}{4}}$
Encoder-decoder (e.g., NMT, T5)	$0.81(N^4M)^{\frac{1}{16}}$	$0.87(N^4M)^{-\frac{1}{16}}$	$(3M)^{\frac{1}{4}}$	$(12M)^{-\frac{1}{4}}$

Figure 2: (a) Pseudocode for DEEPNORM. We take Xavier initialization (Glorot and Bengio, 2010) as an example, and it can be replaced with other standard initialization. Notice that α is a constant. (b) Parameters of DEEPNORM for different architectures (N-layer encoder, M-layer decoder).

二、归一化位置

1. PostLN

在transformer的原始结构中，采用了PostLN结构，即在残差链接之后layerNorm（如下图所示）。在LLM中训练过程中发现，PostLN的输出层附近的梯度过大会造成训练的不稳定性，需要结合warm up做一些学习率上的调整优化。在LLM还是会结合一些preNorm，如在GLM-130B中采用PostLN与PreLN结合的方式。

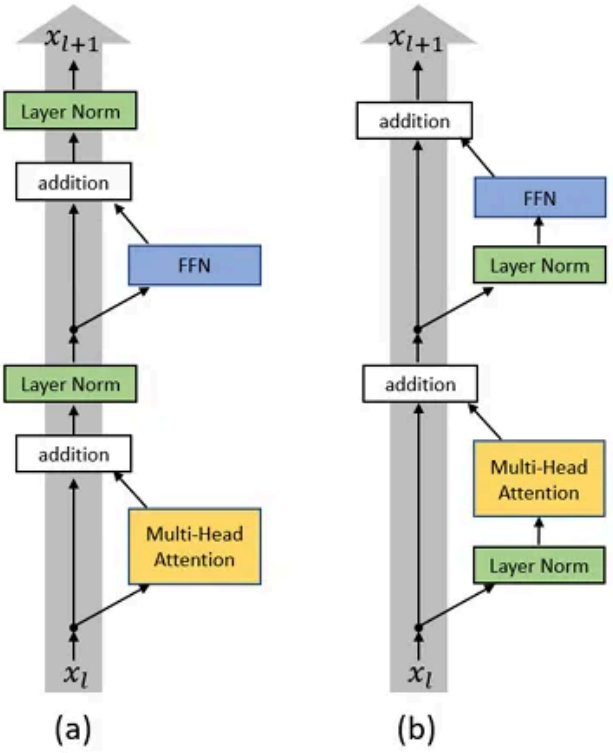

Figure 1. (a) Post-LN Transformer layer; (b) Pre-LN Transformer layer.

2. PreLN

PreLN将layerNorm放置在残差链接的过程中，如上图所示。PreLN在每层的梯度范数近似相等，有利于提升训练稳定性。相比PostLN，使用PreLN的深层transforme的训练更稳定，但是性能有一定损

伤。为了提升训练稳定性，很多大模型都采用了PreLN。

参考文献

[1] A Survey of Large Language Models(https://arxiv.org/abs/2303.18223)

[2] DeepNet: Scaling Transformers to 1,000 Layers(https://arxiv.org/abs/2203.00555)

[3] On Layer Normalization in the Transformer Architecture(https://arxiv.org/abs/2002.04745)

面试干货 70 学术理论解析 53

面试干货 · 目录

上一篇
大模型面经

下一篇
AIGC算法工程师面经—python基础篇