Transformer一作又出新作！HaloNet：用Self-Attention的方式进行卷积

原创 CV开发者都爱看的 极市平台 2021-07-10 22:00:00 手机阅读

收录于话题
#Transformer

作者丨小马

编辑丨极市平台

极市导读

本文是谷歌团队Transformer的一作Ashish Vaswani 又一篇以一作身份发表的论文，也是今年CVPR的Oral文章。作者提出了HaloNet，并在多个任务上做了实验，证明了其有效性。HaloNet在ImageNet上达到了84.9%的top-1准确率，在目标检测和实力分割任务中也取得了不错的效果。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

写在前面

这篇文章是谷歌团队Transformer的一作**Ashish Vaswani** 又一篇以一作身份发表的论文，也是今年CVPR的Oral文章。作者考虑到了CNN是一种参数量和感受野相关（parameter-dependent scaling）交互与输入内容无关的操作（content-independent interactions），而Self-Attention（SA）是一种参数量与感受野无关、交互与输入内容相关的操作，因此就希望将CNN的操作换成SA的操作，以此来解决CNN的缺点。

这篇文章虽然也还是通过局部注意力来提升性能，但是不同于VOLO[2]、CoATNet[3]，只是将局部和全局结构进行串联换成并联，这篇文章就跟操作流程就CNN比较相似，在每一个窗口中都采用了SA的方式进行计算，以此来解决了上面提到的CNN的两个缺点（parameter-dependent scaling、content-independent interactions）。

基于此，作者提出了HaloNet，并在多个任务上做了实验，证明了其有效性。HaloNet在ImageNet上达到了84.9%的top-1准确率，在目标检测和实力分割任务中也取得了不错的效果。

1. 论文和代码地址

Scaling Local Self-Attention for Parameter Efficient Visual Backbones

论文地址：https://arxiv.org/pdf/2103.12731.pdf

官方代码：未开源

核心代码：https://github.com/xmu-xiaoma666/External-Attention-pytorch/blob/master/attention/HaloAttention.py

2. Motivation

CNN最开始是CV任务的基础结构，Self-Attention（SA）最早是基于NLP任务被提出。近期，SA也逐渐被运用在了CV任务中。在CV领域中，SA的结构主要有以下几个优点：1）基于输入内容的信息交互；2）参数的数量与感受野无关；3）能够捕获long-range的关系；4）能够处理CV任务中多种类型的数据（e.g., 像素、点云、序列、图）。

在本篇文章之前，SASA[1]也提出了将卷积的内部操作换成SA，但是在性能上还是跟SOTA的CNN模型还有很大差距。除此之外，以一种非常高效的方式实现像卷积那样实现局部二维的SA会有一定困难（主要体现在CNN找neighborhood是深度学习库自带的，而局部SA找neighborhood的高效实现是不太容易的，naive的实现很容易导致显存爆炸、计算非常慢等问题）。

3. 方法

3.1. CNN和SA的滤波器

带有足够数量head和几何空间bias的SA，在本质上和CNN非常相似（在计算上非常相似，都是根据neighborhood的信息，更新当前位置的信息。但是CNN在训练完成后，所有的参数都是固定的，对于每一个window都是用相同的权重矩阵进行计算的；而SA的权重矩阵是根据Q和K之间的相似度动态计算的，而每一个window中的Q和K都是不同的，所以这个权重矩阵也是不同的，因此局部的SA也是一个Data-dependent的操作）。如果采用的局部的SA。局部的SA和CNN都可以被建模成下面的公式：

$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} f(i,j,a,b) x_{ab},$$

其中x为当前位置的信息，$f()$就是用来求权重矩阵的函数，就是对应上面所说的Local SA和 CNN。

$$f(i,j,a,b)^{conv} = W_{a-i,b-j} \tag{1}$$

$$f(i,j,a,b)^{self-att} = \mathrm{softmax}_{ab}\Big((W_Q x_{ij})^\top W_K x_{ab} + (W_Q x_{ij})^\top r_{a-i,b-j}\Big) W_V \tag{2}$$

$$= p^{ij}_{a-i,b-j} W_v \tag{3}$$

CNN和Local SA的$f()$具体可以由上面的公式表示，卷积这部分非常好理解，就是一个固定的矩阵W。Local SA这部分和传统的SA还是有一点不一样的，这里的Local SA计算权重矩阵时考虑了两个部分：第一，根据Q和K来计算权重矩阵的一部分，这一部分是“内容-内容”之间的交互；第二，根据Q和其他点的相对位置距离来计算权重矩阵的另一部分，这一部分是“内容-距离”之间的交互（这一部分其实就是Transformer中的相对位置编码）。

Noting：

在参数量方面：正常的SA参数量其实适合感受野大小无关的，只和通道数有关；而本文中引入了带相对位置的编码的SA，就需要学习一个相对位置距离的参数矩阵（kxk），所以带相对位置编码SA的参数量和感受野还是相关的，不过这个部分的参数和原始SA中来源于FC的参数相比，其实是非常小的。而CNN中的参数量和感受野是呈平方关系的，比如感受野为5的卷积核参数量是感受野为3的卷积核参数量的25/9倍。

在计算量方面：CNN的计算量和感受野大小是线性关系，而SA和感受野是呈平方关系的。

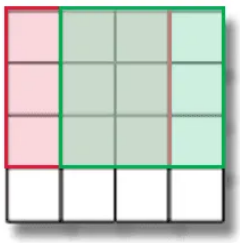
3.2.Block Self-Attention

首先举两个比较极端的例子：如果采用全局的SA，由于计算量与输入的大小呈平方关系，所以对于较大的输入就回导致计算量非常大；如果采用非常小的局部SA，由于每次滑动窗口的结果在显存上得不到释放，就会导致OOM（Out-Of-Memery）的问题。

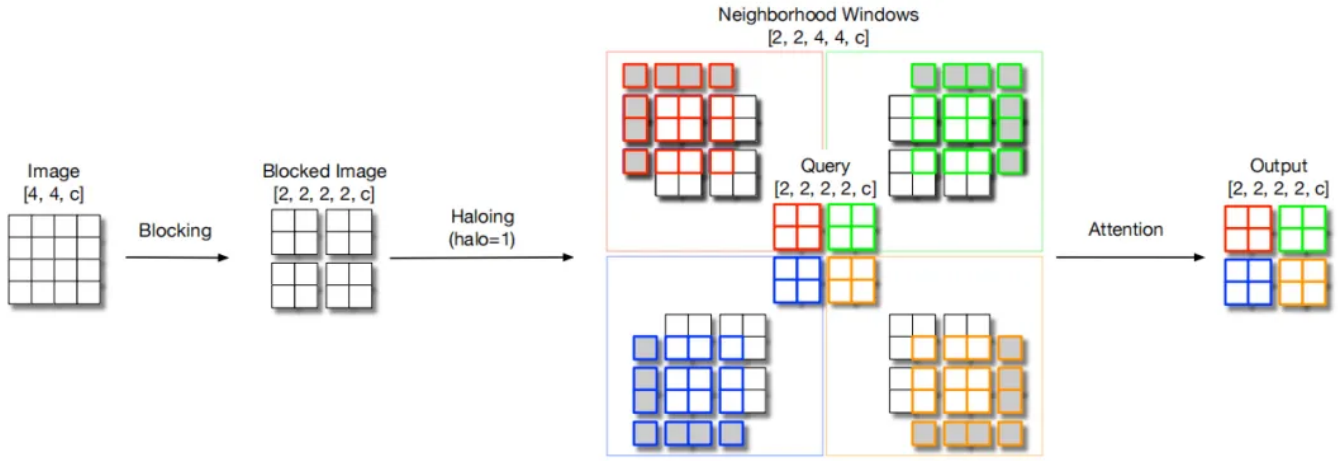
如何减少计算量的同时，又能降低显存呢。

如下图所示，红色区域和绿色区域分别代表滑动窗口前后经过的区域，可以看出，其实这两个局部中有一大部分是重合的，所以在计算时就会导致了重复的计算，如果能减少这部分重复的计算，就能减少计算量。

下图的窗口在滑动时采用的是步长为1，所以遍历完整张图片需要滑动(W-k+1)*(H-K+1)次，在这期间显存都得不到释放；如果能够增大滑动窗口的步长，就能非常有效的减少显存的使用。



基于以上的观察，作者提出了将整张图片分为多个Block，并对每个Block进行SA（Blocked Local Self-Attention）。



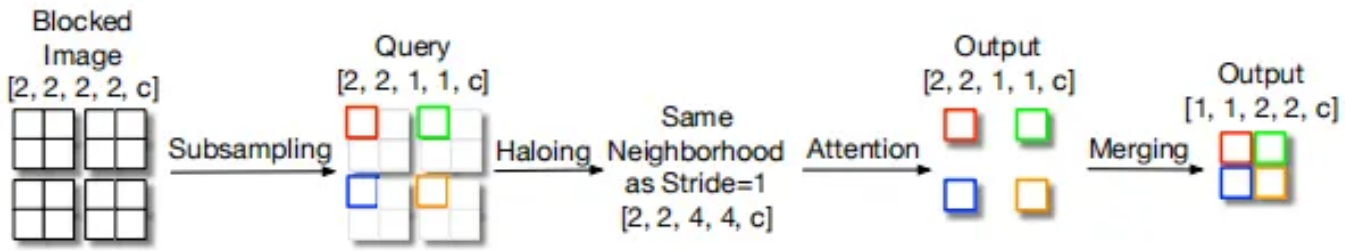
如上图所示，如果每次只考虑block内的信息，必然会导致信息的损失，因此在计算Local Self-Attention之前，作者先对每个block进行的haloing操作。

也就是在每个block外，再用原始图片的信息padding一圈（这个操作叫做Halo），使得每个block的感受野能够适当变大，关注更多的信息。

为什么这个操作叫Halo?

Halo其实就是光环、日晕的意思（大概就是这张图这样），halo操作在原始的block上再外加一圈额外的信息，就类似在block之外再加了一层光环，起到了增加感受野的作用。

3.3. Downsampling



ResNet等CNN结构在进行下采样的时候通常就是用Mean Pooling或者Max Pooling。本文采用了一种更加节省计算量的方法，对每个block的分别进行采样，然后对这些采样后的信息进行Attention操作，从而达到down sample的效果。

这样的操作能够减少4倍的FLOPs，因为这里是对每个block的信息进行采样后计算，而不是对每个block内的信息进行计算，并且这样的操作也是不会影响模型精度的。

3.4. HaloNet

基于以上的结构，作者提出了HaloNet，模型参数设置如下表：

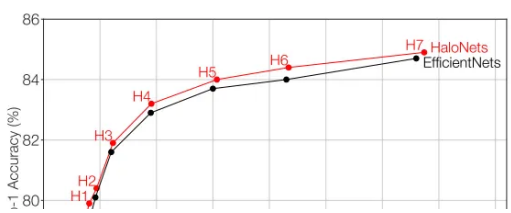
Output Resolution	Layers
$\frac{s}{4} \times \frac{s}{4}$	7×7 conv stride 2, 64 3×3 max pool stride 2
$\frac{s}{4} \times \frac{s}{4}$	$\left\{\begin{array}{c} 1 \times 1, 64 \\ \text{attention}(b, h), 64 \cdot r_v \\ 1 \times 1, 64 \cdot r_b \end{array}\right\} \times 3$
$\frac{s}{8} \times \frac{s}{8}$	$\left\{\begin{array}{c} 1 \times 1, 128 \\ \text{attention}(b, h), 128 \cdot r_v \\ 1 \times 1, 128 \cdot r_b \end{array}\right\} \times 3$
$\frac{s}{16} \times \frac{s}{16}$	$\left\{\begin{array}{c} 1 \times 1, 256 \\ \text{attention}(b, h), 256 \cdot r_v \\ 1 \times 1, 256 \cdot r_b \end{array}\right\} \times l_3$
$\frac{s}{32} \times \frac{s}{32}$	$\left\{\begin{array}{c} 1 \times 1, 512 \\ \text{attention}(b, h), 512 \cdot r_v \\ 1 \times 1, 512 \cdot r_b \end{array}\right\} \times 3$
$\frac{s}{32} \times \frac{s}{32}$	$1 \times 1, d_f$
1×1	global average pooling fc, 1000

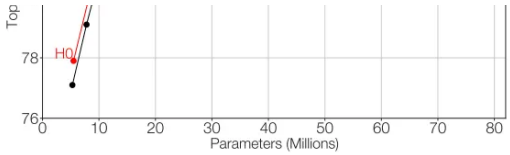
为了能和SOTA模型进行更加公平的比较，作者采用了与EfficientNet类似的参数设置，提出8个HaloNet的变种：

HaloNet Model	b	h	r_v	r_b	Total Layers	l_3	s	d_f	Params (M)	EfficientNet Params (M)	EfficientNet Image Size (M)
H0	8	3	1.0	0.5	50	7	256	–	5.5	B0: 5.3	224
H1	8	3	1.0	1.0	59	10	256	–	8.1	B1: 7.8	240
H2	8	3	1.0	1.25	62	11	256	–	9.4	B2: 9.2	260
H3	10	3	1.0	1.5	65	12	320	1024	12.3	B3: 12	300
H4	12	2	1.0	3	65	12	384	1280	19.1	B4: 19	380
H5	14	2	2.5	2	98	23	448	1536	30.7	B5: 30	456
H6	8	4	3	2.75	101	24	512	1536	43.4	B6: 43	528
H7	10	3	4	3.5	107	26	600	2048	67	B7: 66	600

4.实验

4.1. 分类任务





可以看出，在相似的参数量下，HaloNet的性能能够超过EfficientNet。

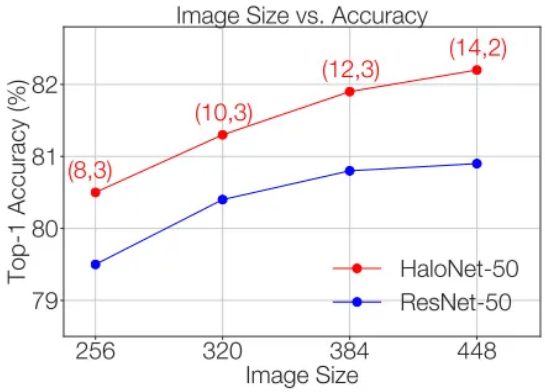
4.2. 卷积结构对于SA的影响

Components	HaloNet Accuracy	Baseline Δ	ResNet Accuracy	Baseline Δ
Baseline	78.6	0.0	77.6	0.0
+ LS	79.7	1.1	78.1	0.5
+ LS, RA	79.9	1.3	78.4	0.8
+ SE	78.6	0.0	78.6	1.0
+ SE, SiLU/Sw1	79.0	0.4	78.9	1.3
+ LS, SE	79.7	1.1	78.9	1.3
+ LS, SE, SiLU/Sw1	79.9	1.3	79.1	1.5
+ LS, SE, SiLU/Sw1, RA	80.5	1.9	79.5	1.9

作者探究了SE，Label Smooth（LS），RandAugment (RA)等在卷积上能够明显提点的结构在SA上有效性。

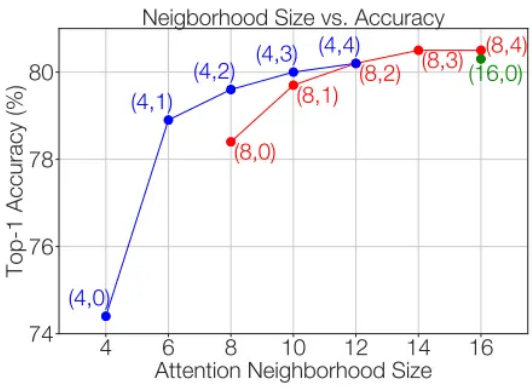
从上表中可以看出，这些结构在HaloNet上依旧有效，LS结构在HaloNet上的提点效果明显优于ResNet上的提点效果。

4.3. 图片大小的影响



随着图片size的增大，HaloNet的性能持续提高，并始终优于ResNet。

4.4. Window size和Halo Size的影响



上图展示了不同（window size，halo size）下，HaloNet的实验结果。可以看出，更大的window size，halo size能提高模型的性能，halo size从0到1的过程，性能有明显提高。

4.5. 卷积和SA的 速度-精度 tradeoff

Conv Stages	Attention Stages	Top-1 Acc (%)	Norm. Train Time
-	1, 2, 3, 4	84.9	1.9
1	2, 3, 4	84.6	1.4
1, 2	3, 4	84.7	1.0
1, 2, 3	4	83.8	0.5

可以看出，更多的SA能够提高精度，但也会带来速度上的损失。

4.6. 目标检测和实例分割的结果

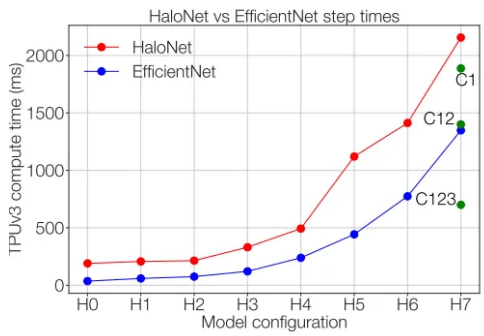
Model	AP^{bb}	AP^{bb}_s	AP^{bb}_m	AP^{bb}_l	AP^{mk}	AP^{mk}_s	AP^{mk}_m	AP^{mk}_l	Speed (ms)	Train time (hrs)
R50 baseline in lit	42.1	22.5	44.8	59.1	37.7	18.3	40.5	54.9	409	14.6
R50 + SE (our baseline)	44.5 (+2.4)	25.5	47.7	61.2	39.6 (+1.9)	20.4	42.6	57.6	446	15.2
R50 + SE + Local Att ($b = 8$)	45.2 (++0.7)	25.4	48.1	63.3	40.3 (++0.7)	20.5	43.1	59.0	540	15.8
R50 + SE + Local Att ($b = 32$)	45.4 (++0.9)	25.9	48.2	63.0	40.5 (++0.9)	21.2	43.5	58.8	613	16.5
R101 + SE (our baseline)	45.9 (+3.8)	25.8	49.5	62.9	40.6 (+2.9)	20.9	43.7	58.7	740	17.9
R101 + SE + Local Att ($b = 8$)	46.8 (++0.9)	26.3	50.0	64.5	41.2 (++0.6)	21.4	44.3	59.8	799	18.4

可以看出，与ResNet结构相比，在检测和分割任务上，HaloNet在性能上都能取得一定的提高。

5. 总结

本文采用了CNN的网络结构，SA的计算方式，提出了一个新的网络结构HaloNet。与以前的工作相比，本文的创新点主要体现在两个结构，blocked local attention和attention downsampling。HaloNet在分类、检测、分割任务上都取得不错的效果。

个人认为，HaloNet其实是一种介于CNN和Transformer之间结构，虽然它融合了CNN和SA的优点，但是在一定程度上也具有他们的缺点，比如训练上比CNN（EfficientNet）更慢（如下图所示）。因此，后期对HaloNet计算量和显存的优化也是一个非常重要的工作。



参考文献

[1]. Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan
Bello, Anselm Levskaya, and Jon Shlens. Stand-alone selfattention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors,
Advances in Neural Information Processing Systems 32, pages
68–80. Curran Associates, Inc., 2019.

[2]. Yuan, Li, et al. "VOLO: Vision Outlooker for Visual Recognition." arXiv preprint arXiv:2106.13112 (2021).

[3]. Dai, Zihang, et al. "CoAtNet: Marrying Convolution and Attention for All Data Sizes." arXiv preprint arXiv:2106.04803 (2021).

本文亮点总结

1.本文虽然也还是通过局部注意力来提升性能，但是不同于VOLO、CoATNet，只是将局部和全局结构进行串联换成并联，这篇文章就跟操作流程就CNN比较相似，在每一个窗口中都采用了SA的方式进行计算，以此来解决了CNN的两个缺点（parameter-dependent scaling、content-independent interactions）。

2.本文采用了一种更加节省计算量的方法，对每个block的分别进行采样，然后对这些采样后的信息进行Attention操作，从而达到down sample的效果。这样的操作能够减少4倍的FLOPs，因为这里是对每个block的信息进行采样后计算，而不是对每个block内的信息进行计算，并且这样的操作也是不会影响模型精度的。

阅读原文