背景

将 Transformer 应用到计算机视觉领域存在一些问题，比如说尺寸问题，现有的基于 Transformer 的方法先将图片分块，再映射为 token，而 token 的尺寸都是固定的；另一个问题就是分辨率的问题，高分辨率的图片导致了计算复杂度随着图片的尺寸呈二次式增长（后面又详细的计算过程）。

而 Swin Transformer 通过构建一个 hierarchical 表示方法，来解决尺寸固定问题，通过在不重叠的窗口中使用 self-attention 来解决计算复杂度的问题。

方法

使用 W-MSA 模块减小计算量

MSA 的计算量

先看一下单个 self-attention 模块的计算量。

输入时序向量 $\boldsymbol{x}=\{x_1,x_2,x_3,x_4\}$ 经过 input Embedding 映射到高维度向量 $\boldsymbol{a}=\{a_1,a_2,a_3,a_4\}$，然后向量 $\boldsymbol{a}^{hw\times C}$ 通过共享的参数矩阵 $\left\{W_q^{C\times C},W_k^{C\times C},W_v^{C\times C}\right\}$ 生成各自的 $\{q^i,k^i,v^i\}$。

这一步对应的计算量为：

$$3\times h\times w\times C^2 \tag{1}$$

根据 attention 的计算公式：

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

可以看出，随后进行的是 Q（$hw\times C$）和 K 的转置矩阵（$C\times hw$）相乘，这一步的计算量是：

$$(h\times w)^2\times C \tag{2}$$

忽略掉除以 $\sqrt{d_k}$ 和 *softmax* 操作所带来的计算量，接下来就是最后一步矩阵相乘，将刚才得到的注意力分数（大小为 $hw\times hw$）和矩阵 $V_{hw\times C}$ 相乘，这一步的计算量也是：

$$(h\times w)^2\times C \tag{3}$$

所以综上，self-attention 操作所需要的计算量有：

$$3hwC^2+2(hw)^2C \tag{4}$$

多头注意力机制相比于单个 self-attention 机制的计算量，多了最后的融合多头的操作，融合操作需要的计算量有：

$$h\times w\times C^2 \tag{5}$$

所以，MSA 模块需要的计算量有：

$$4hwC^2+2(hw)^2C \tag{6}$$

也就是原论文中公式(1)的由来

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \quad (1)$$
$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \quad (2)$$

W-MSA 的计算量

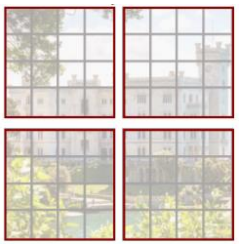

图 1 W-MSA 划分效果示意图

如上图所示，W-MSA 模块是先把特征图划分为一个个没有重叠的窗口，然后对每个窗口内部使用 self-attention，那么假设将特征图划分为了 $M \times M$ 大小的窗口，那么一个特征图包含了 $\frac{h}{M} \times \frac{w}{M}$ 个窗口，那么计算量有：

$$\begin{aligned} &\frac{h}{M} \times \frac{w}{M} \times (4 \times M \times M \times C^2 + 2 \times \left(M^2\right)^2 \times C) \\ &= 4hwC^2 + 2M^2hwC \end{aligned} \quad (7)$$

也就是原论文中的公式(2)。

使用 SW-MSA 进行信息交互

W-MSA 虽然可以有效的减少计算量，但是不同窗口之间没有办法进行信息交互，为此提出了 SW-MSA，位移窗口多头注意力机制。

第一个位移窗口使用的是常规的窗口分区方法，如下图所示，将 8×8 的特征图划分为边长为 4（即 M=4）的窗口（红色），然后整个特征图被划分为 2×2 个窗口。接下来，使用的窗口配置方法是将前一层的窗口进行左移 $\left\lfloor \frac{M}{2} \right\rfloor$ 个像素点，上移 $\left\lfloor \frac{M}{2} \right\rfloor$ 个像素点，得到新的窗口划分区域（绿色框）。

新得到的绿色框中，上方绿色矩形框 1 融合了上一层窗口中的 1、2 两个窗口的信息，绿色矩形框 2 融合了上一层窗口中的 1、3 两个窗口的信息，绿色矩

形框 3 融合了上一层窗口中的 4 个窗口的信息，绿色矩形框 4 融合了上一层窗口中的 2、4 两个窗口的信息，绿色矩形框 5 融合了上一层窗口中的 3、4 两个窗口的信息。

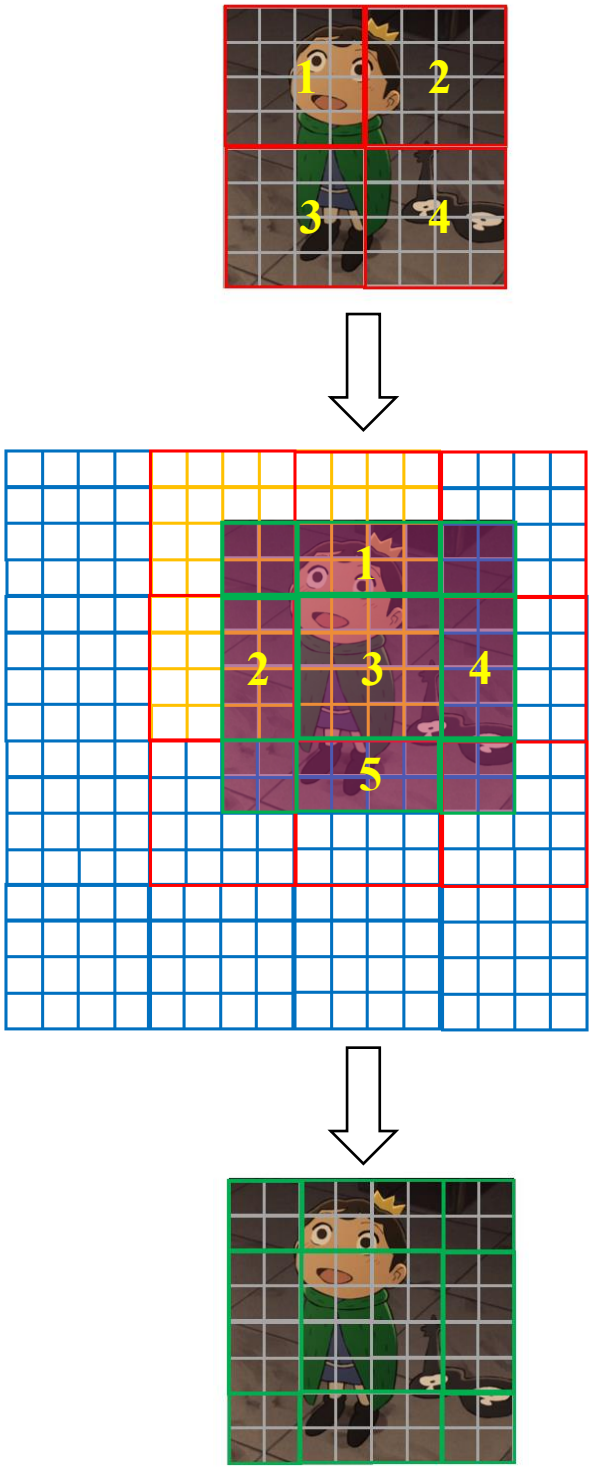

图 2

SW-MSA 中的高效批量计算

位移窗口方法划分区域带来的问题就是会产生很多的窗口，而且窗口尺寸会比原来的 $M\times M$ 都小，一个直接的办法就是将尺寸较小的窗口（比如上图中的绿色左上角的框）进行扩充，扩充到 $M\times M$ 大小再进行计算，但是这样计算量就增加了，因为本文针对位移窗口划分方式提出了一种更加**高效的批量计算**方法，其过程示意图如下图所示。

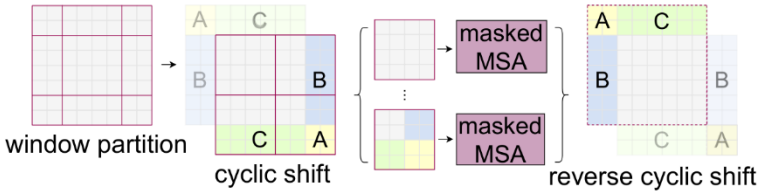

图 3

其中 **cyclic shift 操作**的具体过程如下所示：

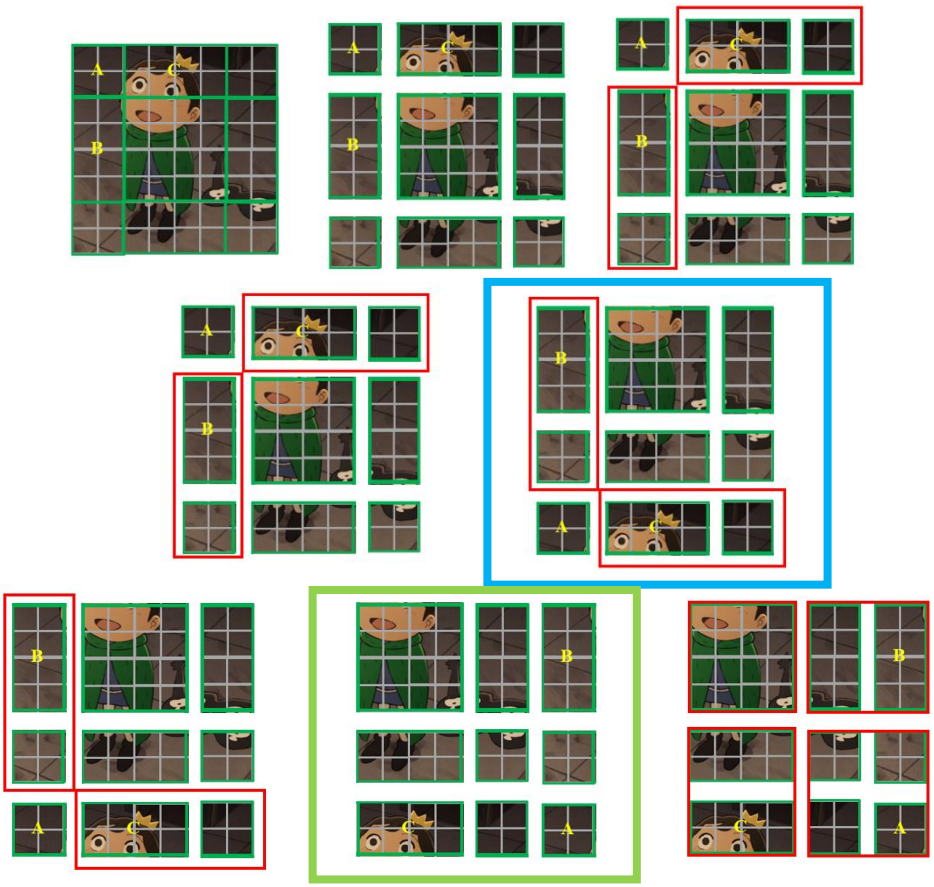

图 4 cyclic shift 过程示意图

先将绿 A 框和第一行的红色框区域内的框整体平移到最后一行，得到蓝色框中的结果，然后再将左侧红色框区域内的框和绿 A 框整体平移到最右侧，得到绿色框中的结果。最后由一开始的 9 个划分区域，又变回了 4 个划分区域（最后的红色框）。

这样位移完之后又带来了新的问题，新组合而成的 4×4 划分区域，比如说下图中带问号的区域，相比于左侧 4×4 的划分区域，该区域是由两个子窗口组合而成的，而且这两个子窗口的区域不连续，那么在之后对每一个窗口进行 self-attention 计算的时候，如下图右侧的图例所示，位置 0 处的 q 会与另一个原本不相邻的子窗口中的每一个 k 进行相关性运算时又该如何处理？

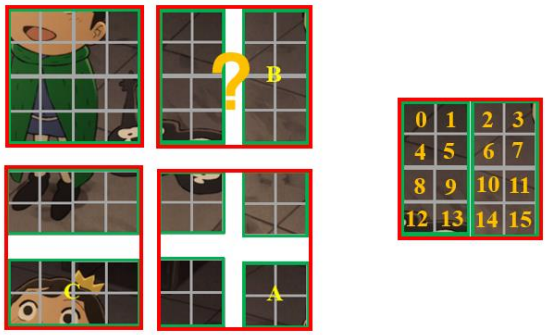

图 5 组合区域的相关性计算

针对这一问题，提出了一个**掩膜机制**，其示意图如下所示：

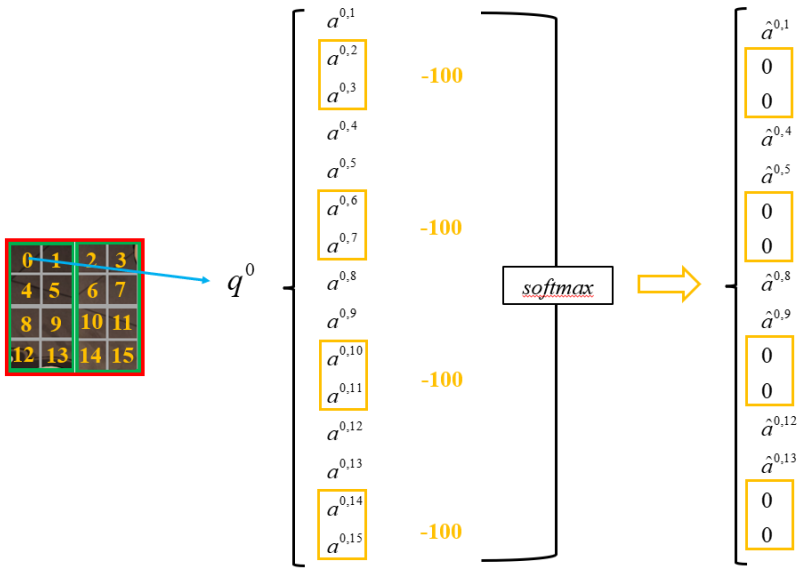

图 6 掩膜机制计算示意图

将位置 0 的 q 与特征图上其他位置像素点的 k 进行相关性计算，得到$a^{0,1}$～$a^{0,15}$，把与另一个子窗口计算得到的$a^{0,2}$、$a^{0,3}$、$a^{0,6}$、$a^{0,7}$、$a^{0,10}$、$a^{0,11}$、$a^{0,14}$、$a^{0,15}$减去一个比较大的数，这样使得经过 *softmax* 处理后，其值被抑制为 0，这样就将两个子窗口的相关性计算分隔开了。

计算完所有的 attention 后，再将刚才位移后的特征图进行位置还原。

相对位置偏置

	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

图 7 位置编码对比分析

不使用位置编码、使用 ViT 中的绝对位置编码、相对位置偏置在 ImageNet 数据集上进行测试，其对比结果如上图中的红色框所示，可以看出，使用相对位置偏置的效果最好。

相对位置偏移参数是原论文中，公式(4)里的参数 B：

$$\text{Attention}(Q,K,V)=\text{SoftMax}(QK^T/\sqrt{d}+B)V$$

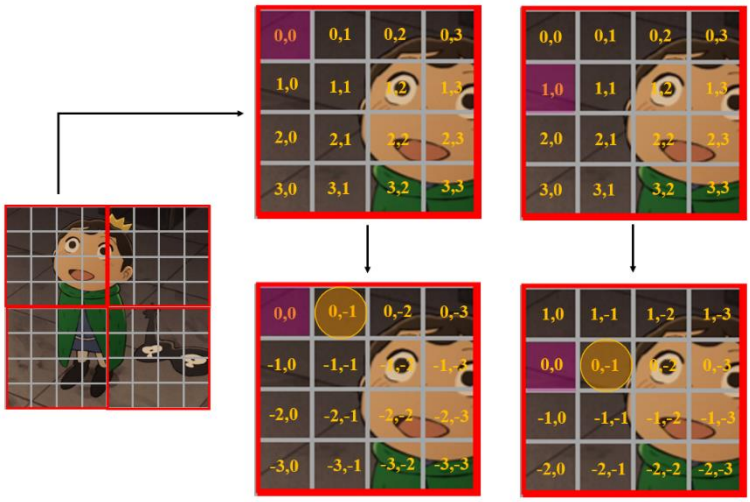

图 8 相对位置示意图

将上图中最左侧的特征图的最左上角的窗口取出，从左上角开始位置坐标编码，然后计算每个位置相对于其他位置的坐标，图中只展示了两个像素点相对坐标的计算过程，可以看出橙色圆圈内两个位置的相对坐标相同，其分别对应取出的窗口特征图中(0,0)位置和(1,0)位置右侧的相对位置坐标。而且相对位置坐标范围为[-M+1, M-1]，经过一系列操作，得到相对位置索引，相对位置偏移参数保存在相对位置偏置表中，其长度为(2M-1)×(2M-1)，以上图为例的话，其对应的相对位置偏置表的长度为 9×9，相对位置偏置参数 B 是根据上面的相对位置索引查相对位置偏置表得到的。

整体框架参数配置

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 128, head 4] × 2	[win. sz. 7×7, dim 192, head 6] × 2
stage 2	8× (28×28)	concat 2×2, 192-d , LN	concat 2×2, 192-d , LN	concat 2×2, 256-d , LN	concat 2×2, 384-d , LN
		[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 256, head 8] × 2	[win. sz. 7×7, dim 384, head 12] × 2
stage 3	16× (14×14)	concat 2×2, 384-d , LN	concat 2×2, 384-d , LN	concat 2×2, 512-d , LN	concat 2×2, 768-d , LN
		[win. sz. 7×7, dim 384, head 12] × 6	[win. sz. 7×7, dim 384, head 12] × 18	[win. sz. 7×7, dim 512, head 16] × 18	[win. sz. 7×7, dim 768, head 24] × 18
stage 4	32× (7×7)	concat 2×2, 768-d , LN	concat 2×2, 768-d , LN	concat 2×2, 1024-d , LN	concat 2×2, 1536-d , LN
		[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 1024, head 32] × 2	[win. sz. 7×7, dim 1536, head 48] × 2

Table 7. Detailed architecture specifications.

图 9 整体框架参数配置

如上图所示，就是不同 Swin Transformer 的参数配置。