# 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

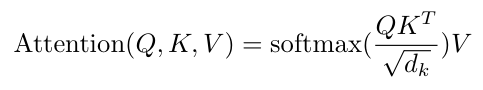
### MSA的计算量

先看一下单个self-attention模块的计算量。

输入时序向量经过input Embedding映射到高维度向量，然后向量通过共享的参数矩阵生成各自的。

这一步对应的计算量为：

 (1)

根据attention的计算公式：

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

 (2)

忽略掉除以和*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*×*hw*）和矩阵*Vhw×C*相乘，这一步的计算量也是：

 (3)

所以综上，self-attention操作所需要的计算量有：

 (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：

 (5)

所以，MSA模块需要的计算量有：

 (6)

也就是原论文中公式(1)的由来

### 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大小的窗口，那么一个特征图包含了个窗口，那么计算量有：

 (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个像素点，上移个像素点，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。

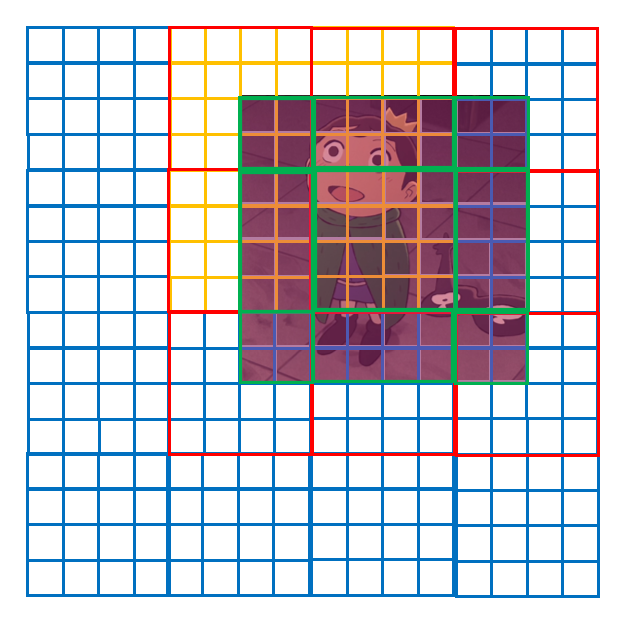


图 2

**1**

**2**

**3**

**4**

**4**

**1**

**2**

**3**

**5**

## 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*×*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*×*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。

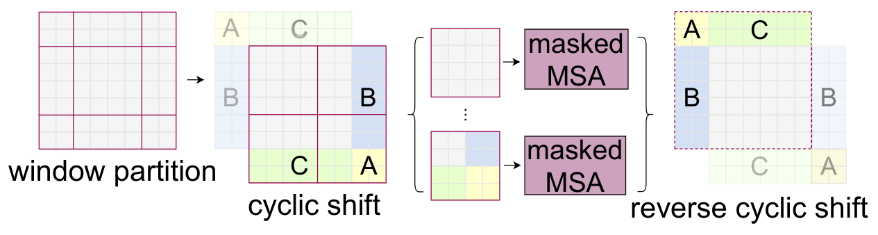
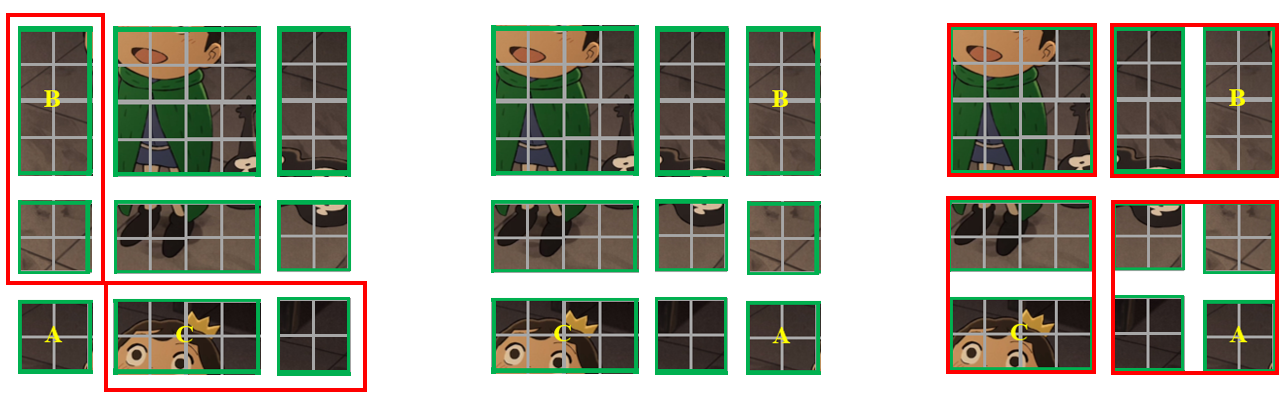
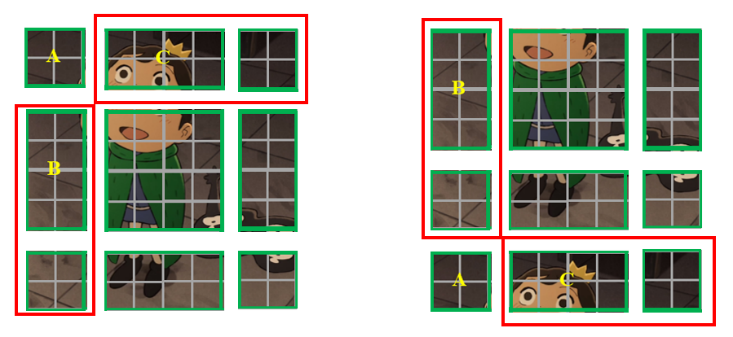
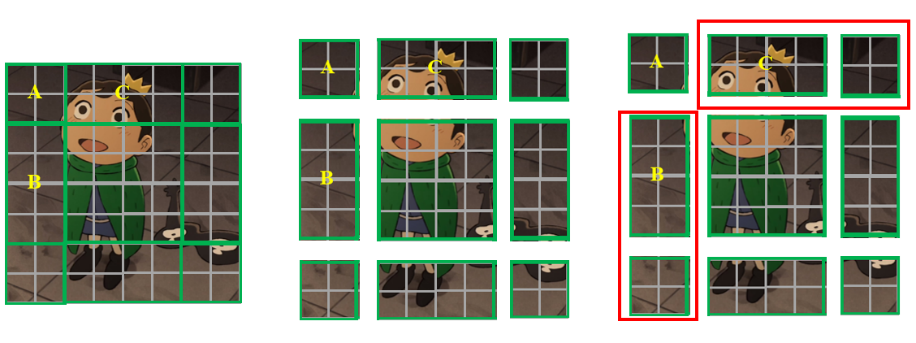


图 3

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？

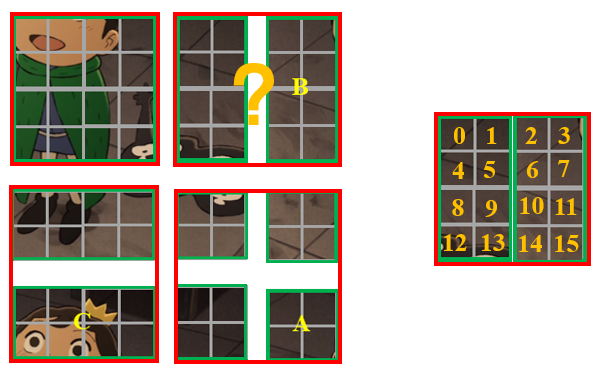


图 5 组合区域的相关性计算

针对这一问题，提出了一个**掩膜机制**，其示意图如下所示：

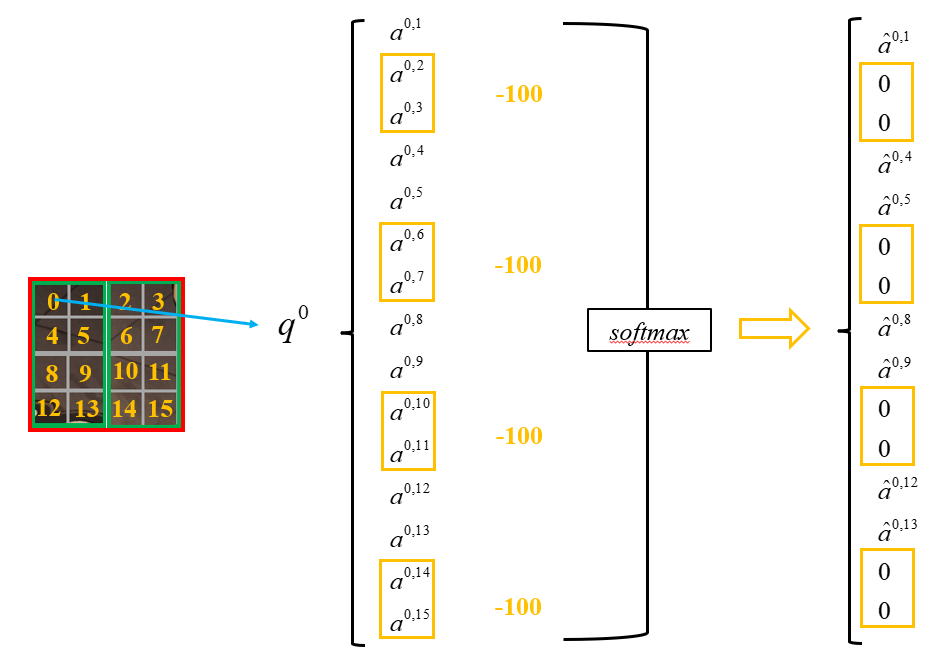


图 6 掩膜机制计算示意图

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~，把与另一个子窗口计算得到的、、、、、、、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

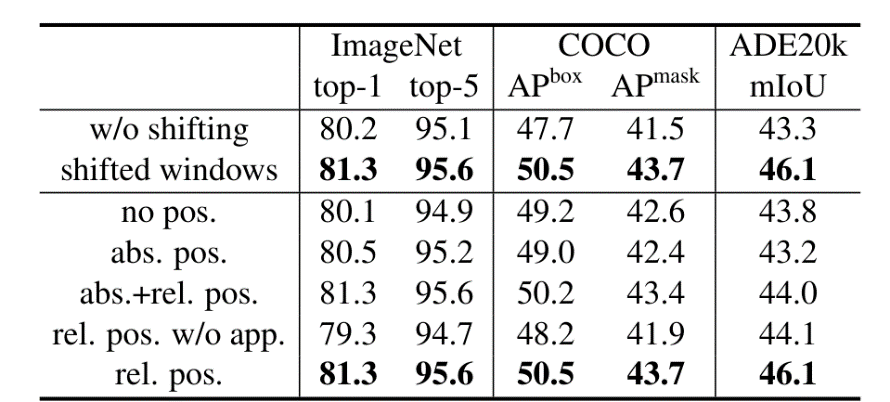


图 7 位置编码对比分析

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*：

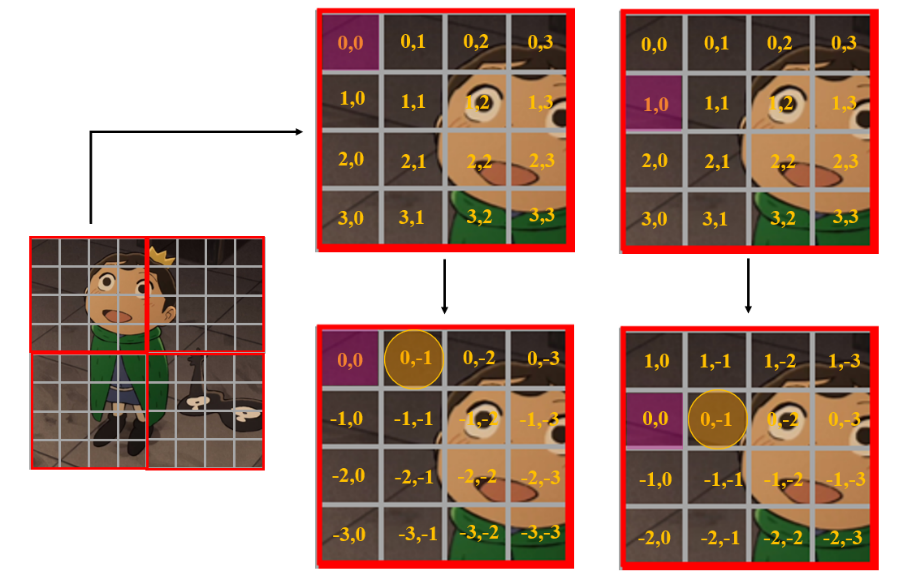


图 8 相对位置示意图

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(2*M*-1)×(2*M*-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

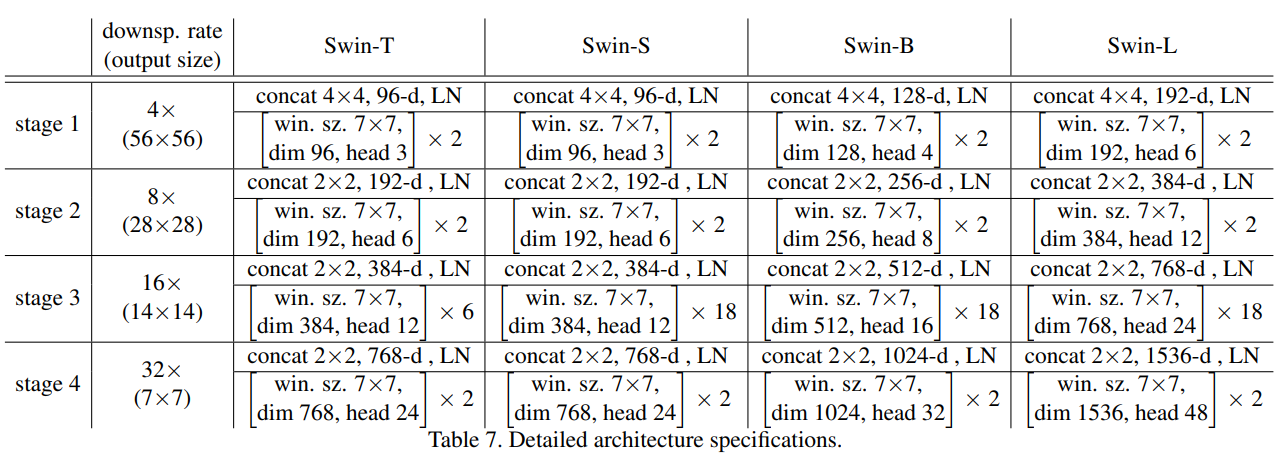


图 9 整体框架参数配置

如上图所示，就是不同Swin Transformer的参数配置。