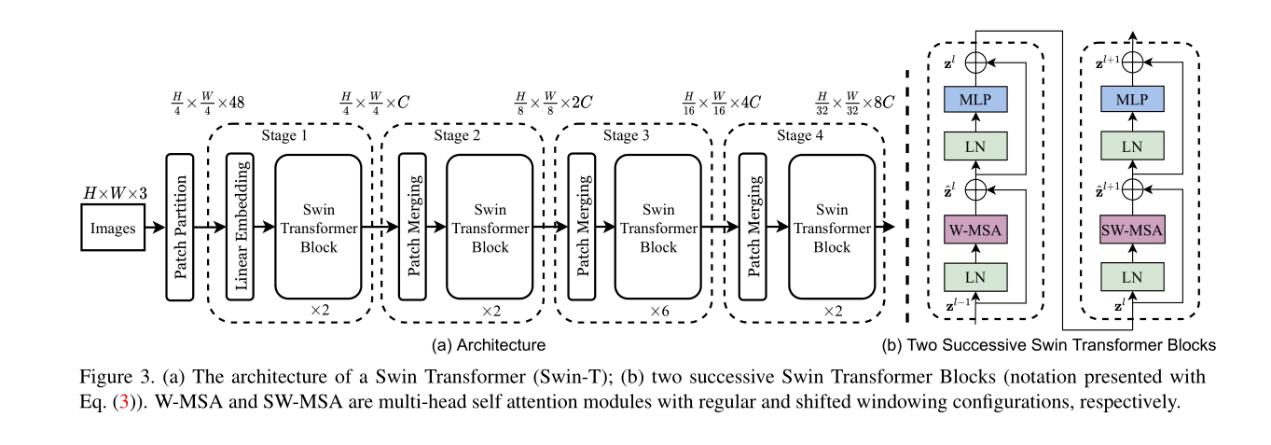
# 基于swin transformer分类器的调查

## 背景

Swin Transformer是2021年微软研究院发表在ICCV上的一篇文章，并且已经获得ICCV 2021 best paper的荣誉称号。Swin Transformer网络是Transformer模型在视觉领域的又一次碰撞。该论文一经发表就已在多项视觉任务中霸榜。该论文是在2021年3月发表的，根据官方提供的信息可以看到，现在还在COCO数据集的目标检测以及实例分割任务中是第一名。

## 整体框架

首先简单看一下论文中给出的swin transfoemer的网络框架。



图表 1

该网络模型主要分成三个模块。

首先将图片输入金patch partition模块中进行分块，每4x4相邻的像素作为一个patch，然后在通道方向展平。论文中以RGB图像为例，通道数为3，每个patch就有4x4x3=48个数据。经过次过程的图像就会由原来的[H, W ,3]转变为[H/4, W/4, 48]。然后经过一个Linear Embeding层对每个像素的channel数据做线性变换，由48变成C，即图像shape再由 [H/4, W/4, 48]变成了 [H/4, W/4, C]。[1]转变后的patch可以认为是一个像素，只不过像素的通道数较多。

之后通过四个相同的stage。除了stage1中一开始是Linear Embeding层，其余都是patch merging层进行下采样。需要注意的是每个swin transformer block都是包括两个部分，如如中右侧所示。两个部分不同之处仅在于第一个使用的是W-MSA，第二个使用的是SW-MSA。一个是对局部进行注意力运算，一个是对窗口间进行信息通信而进行的注意力计算。

因为swin transformer block的成对使用，所以个数为偶数。

最后经过计算机的结果会接上一个Layer Norm层，全池化层以及全连接层得到输出。论文中没有明确说明，但是在源码中却是这样操作的。

## 详细说明

对整个图像进行一个划分处理，首先是划分patch，之后根据相邻的patch综合成几个window。

### Patch merging

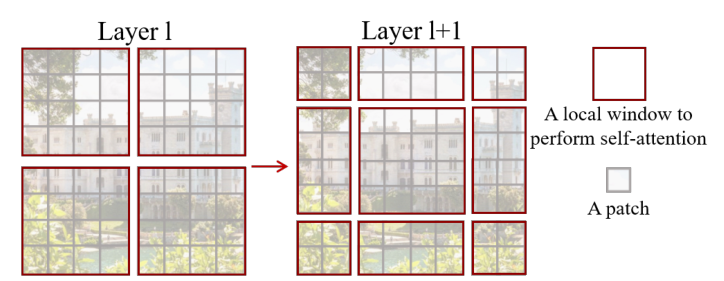
首先对每个patch merging层进行说明。该层主要是完成下采样任务。以一个4x4像素的图像来说，在实际操作过程中，先将2x2的相邻元素设置为一个patch，每个patch 取相同位置的的像素拼接在一起得到4个feature map，将这四个feature map在深度方向上拼接，之后通过一个LayerNorm层，最后通过一个全连接层在深度方向上做线性变化，将深度由C变为C/2。这样每次下采样，输入图像高度和宽度减倍，深度加倍。

### W-MSA和SW-MSA

W-MSA模块的全称是Windows Multi-head Self-Attention。相比于普通的MSA，为了减少计算量，在计算的时候需要将每一个feature map划分成MxM大小的windows，然后单独对每一个windows内部进行自注意力计算。相对来讲计算每个windows的注意力和直接计算全局的注意力，可以很明显减少计算量。

但是这样会导致一个后果，就是每个windows之间是独立的，没有办法计算一个全局的结果。为了让不同的windows之间能够沟通信息，论问题提出了一种shift机制。

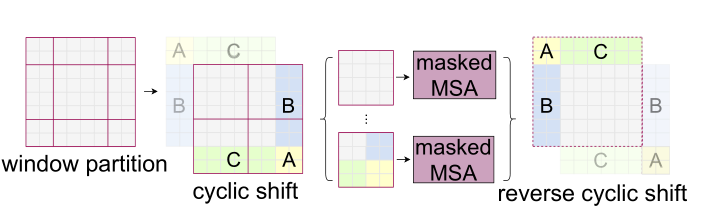
除了W-MSA，还有一种结构是SW-MSA（Shifted Windows Multi-Head Self-Attention）。总结来讲就是将将整体的图片进行一个二次划分，划分成不同的windows，这一次的划分是和第一次划分不同，如图所示：



图表 2

图中，左边的是第一次划分windows的结果，右侧是第二次划分的结果。划分完成之后依旧会对每个windows进行注意力计算。我们能很明显发现，右侧图中第一行第二个的windows能够沟通左侧的两个窗口，第二行的第二个能沟通四个窗口。窗口间的信息交流问题就这样解决了。

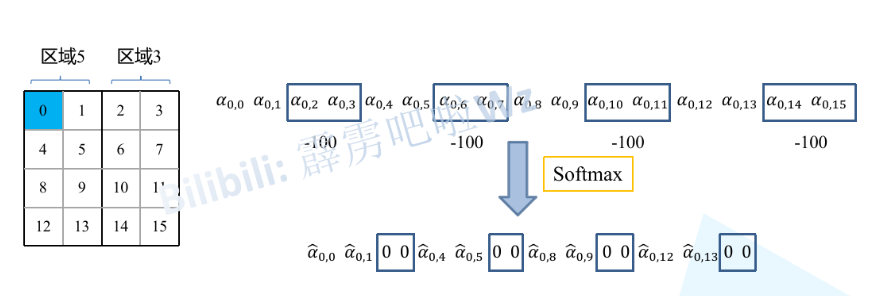
但是，第二次划分之后是9个窗口，需要对每个窗口进行注意力计算，这样来看，又变得更麻烦了。所以，为了便于计算，论文的作者又提出了一种更加高效的计算方法。下面是原论文的示意图：



这张图的意思就是，将第二次分好的窗口进行一个标号第一行的第一个记为A，第二个和第三个记为C，第二行第一个和第三行第一个共同记为B。将九个窗口从左到右依次从0标记。

先将C向下移动到对底部，将B移动到最右边，A移动到右下角。将移动好的图像进行第一次的分割，分成四个窗口，可以发现左上窗口对应的是原来的中间的窗口，右上窗口里包含了原来的第五个和第三个窗口，左下包含了原来第一个和第七个窗口，右下包含了第零个、第二个、第六个和第八个窗口。之后对新的4个窗口进行注意力计算，就相当于是计算了原来第二次划分的窗口注意力信息。

可是，这样划分再计算，是会导致不同窗口之间的信息乱窜，对最后的结果造成影响。所以，在实际使用过程中，使用的是masked MSA，即带有蒙版的MSA。通过设置不同的蒙版来隔绝不同区域内的信息。如图所示：

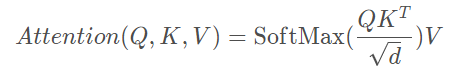


对于窗口内的每一个像素来说，在进行MSA计算时，都要先生成对应的Q,K,V值。假设对于上图的像素0来说，得到Q之后要与每一个像素的K进行匹配，图中右侧上方的数值代表的就是最后计算的匹配结果。但是，我们时不想有区域3的结果的。所以计算出结果之后，每个像素0和区域三像素的匹配结果都会-100.之后进行一个Softmax操作。一般来讲匹配结果都是零点几的数。将其-100之后再经过Softmax计算相当于权重都是0了。所以对于区域5的像素来说，只是计算了本区域内的MSA。

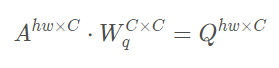
当计算出结果之后，需要将移动后的图像复原。

### 自注意力计算及偏执参数

在计算注意力结果的时候，我们一般使用的公式如下：



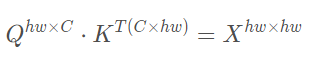
对于每个feature map内部的像素都要进行注意力计算。首先通过​,​,​生成对应的q,k,v。这里假设q,k,v的向量长度和feature map 的深度保持一致。那么计算对应所有像素生成Q的过程如下公式：



* 为将所有像素拼接在一起得到的矩阵。一共有hw个像素，每个像素深度为C。
* 表示生成query（q值）的变换矩阵。
* 为所有像素经过得到的q值拼接后的矩阵

生成K和C也是同理，只不过中间的转换成对应的和。在代码中，q,k,v是通过Linear层直接进行线性变换得到的。

之后计算Q和的相乘结果，计算公式如下：

**

接下来再经过除以以及Softmax的计算，假设得到,最后乘以V，得到结果B。公式如下：

此时，我们计算的过程中使用的还仅仅是单头的自注意力模块。计算我们需要的多头注意力模块时，我们的需要在次结果上加上一个融合矩阵。公式如下：



在这里计算的时候，论文中提到了偏执参数。加上偏执参数的公式如下：



论文中指出在一般情况下，有偏执参数的计算结果比之前的要好。但是这个B的计算是一个问题。B代表的是相对位置，那么应该怎样去描述相对位置就很重要。

以每个像素点为原点建立坐标系，向左为正，向上为正，描述出所有其他像素相对于此时作为原点的像素的位置。（代码中是这样写的）。将每一次选择原点计算得到的结果作为一行，将所有的计算结果铺成一个矩阵，每个矩阵的元素就是一个二维索引。

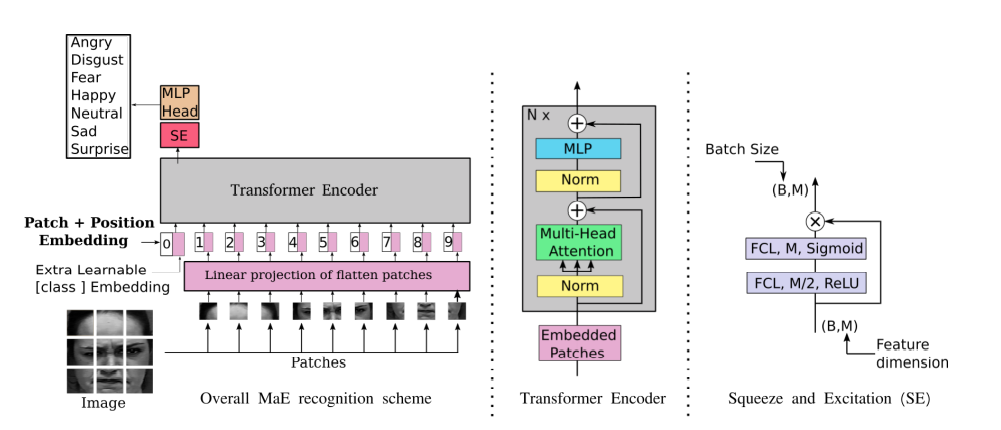
简单地将行索引、列索引相加可以转化为一维索引，但是如果索引中有（-1,0）和（0,-1），这在二维中明显代表不同位置，简单相加就会出现问题。为了解决此问题，先将每个相对位置索引都加上M-1(M为窗口的大小)，加上之后就不会出现负数。之后将所有额行标都乘以2M-1。最后才将行标和列标相加，这样既保证的相对位置关系，也不会出现上述问题。

此时还是相对位置索引，并不是相对位置偏执参数。真正的训练参数是保存在特定位置的表里的，可以根据上述的索引拿到偏执参数。

## 改进想法

阅读完整篇论文之后，我对于这个注意力模型有了一些改进想法。静态的图片实际就偏重于注意力的改进。以表情识别来说，可以探究的是表情不协调现象，也就是我们认为标准表情都是五官或者不同运动单元一起协调运动，而有可能有的表情就是不协调，不同的注意力区域产生了不协调，具体而言就是不同的注意力区域可能相对于正常的表情产生了不正确的分布（比如说笑就应该有眼角纹，但是个人完全没有眼部运动，比如假笑，或者是一些复杂情绪），我们是否能发现数据集中这种不协调情况，实际上也算是数据的噪音，然后克服这种不协调情况来增强模型效果。同理，对于其他分类的改进也是如此，找到数据集中的不协调现象，然后进行改进。

1. 视觉转换器。包括两个步骤：标记化和转换器编码器。在标记化步骤中，将图像裁剪到L =（h×h）维面片上，然后展平为向量。添加一个额外的可学习向量作为分类标记。每个向量都标记有一个位置值。Transformer编码器是注意模块的N个块的序列。注意块的主要部分是多头注意（MHA）。多头注意力机制由z个自我注意的头部组成，也称为内部注意。自我注意的概念是将单个序列的不同位置联系起来。那么就可以将一张图片裁剪后的结果再联系起来。[2]



挤压和激发块也是一种注意力机制。它包含的参数比自我注意块少得多，其中两个完全连接的层仅使用一个逐点乘法操作。具体来说，我们只使用激励部分，因为挤压部分是一个池层构建，以减少2d CNN层的尺寸。在我们的体系结构中使用SE的想法是通过在提取的局部注意特征之间学习更多的全局注意关系来优化ViT的学习。因此，在分类标记向量上更精确地将SE引入变压器编码器的顶部。

自注意力机制的概念是将单个序列的不同位置联系起来，以计算序列的表示。那么我们可以将一张图片分割成不同的window，只不过这一次的划分不再是像swin transformer那样不规则的，而是规则的。或者说是要每一个window尽量包括人脸有可能影响最后识别的一部分的，比如说眼睛、下巴等。就如图所示，对每一个window进行一定注意力处理之后才输入Transformer Encoder中，计算全局的注意力。或者将每一个window计算完注意力之后再通过滑动窗口计算全局的注意力。之后再经过一次裁剪展平之后进入swin transformer。相当于是先将图片经过一个注意力的预处理，再进入模型。

1. swin transformer主要是使用W-MSA和SW-MSA。那么应该可以使用SE模块的激发块，对进行swin transformer block的输出部分进行处理。SE模块的部分本来也是用来学习全局关系的。那么是否在进入SW-MSA模块之前，有一个这样的处理结果作为参考，对之后进行SW-MSA学习的时候可以作为一个参考。加在SW-MSA之前是为了在计算W-MSA之后，我们进行一个对全局的计算处理，之后进行SW-MSA计算的时候，是否就可以消除滑动模块的时候每一个window只通信全局的一部分的影响。 在SW-MSA模块中，虽然window之间进行了交流。但是每个window都是计算的该window局部的，shift之后的window也只是让不同window之间的patch之间有了交流。没有总体上的一个全局交流。那么是否可以在SW-MSA之后加上一个关注全局的模块，对总体的情况也进行一个关注。或者在某个地方加上这样的一个模块进行消除这种影响。

## 总结

在本次的调研中，我们对现在比较先进的图像分类器swin transformer进行了调查。可以明显发现这个结构是脱胎于VIT。但是和VIT不同的是这里划分了许多的windows和patch。便于实现进行注意力的计算，如果仅仅是按照原来的图像进行注意力计算，每个像素都要和图片中的其他像素进行注意力计算，那么计算量就会大很多。这个论文非常创新的点在于为了避免计算量的增加而做出的一系列改动，不管是window还是窗口之间的移动，计算注意力时使用的蒙版，都是需要理解的。

其实，在一开始这么复杂算法时有些难以理解的。但是，在明白它们的底层逻辑之后，就很容易理解为何这样做。整篇论文理解的难点就是W-MSA和SW-MSA。其实很容易发现，这个论文中的方法需要一开始设计windows的划分和蒙版的设计。因此，我们分类时需要对图像进行预处理，裁剪成合适的图片规格。或者，在挑选数据集的时候尽量选择和输入要求一致的图片。论文中给出了对不同规格的图片使用的不同的参数。如果，我们使用的规格不正确，需要重新对新规格进行窗口和蒙版设计。

## 参考文献

[1] Liu Z , Lin Y , Cao Y , et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows[C]// 2021.

[2] Aouayeb M , Hamidouche W , Soladie C , et al. Learning Vision Transformer with Squeeze and Excitation for Facial Expression Recognition[J]. 2021.