Embedding

由于one-hot向量没有任何的语义信息，而且在这个one-hot中，空间庞大。我们需要一个低维稠密的向量来代替one-hot向量。这时就可以使用Embedding来进行编码，得到低维稠密的向量。torch.nn.Embedding就是用来将一个数字变成一个指定维度的向量的。在使用时Embedding层将该数字直接转化为你想要的维度的one-hot向量，这样就不需要存储一个one-hot的矩阵来占用内存。由此可以看出对one-hot向量进行编码时，Embedding和Linear几乎是一样的，区别就在于：输入不同，一个是输入数字，后者是输入one-hot向量。习惯上，我们在模型的第一层使用的是Embedding，而不是Linear。模型的后续不会再使用Embedding，而是使用Linear。

例如初始化时embedding=torch.nn.Embedding(vocab\_size,emb\_size)，其中vocab\_size为需要的one-hot向量的维度长，emb\_size为编码的维度。实际使用时只需要传入一个整数即可。如embedding(torch.tensor([0],dtype=torch.int64))

对于Embedding的初始化，还有特殊的初始化方法，即如下方法：embedding=nn.Embedding.from\_pretrained(torch.tensor([0],dtype=torch.int64))。即使用预训练好的词向量，并可以将该词向量的参数冻结住。即将其requires\_grad = False。

Transformer Position Embedding

在Transformer中，其位置编码为一个一维绝对的编码，根据位置所决定的sin，cos常量，是不可以学习的,其实质为不同维度使用不同频率的正/余弦公式进而生成不同位置的高维位置向量。采用奇偶区分可以通过全连接层帮助重排坐标，所以可以直接简单地分为两段(前256维使用sin，后 256维使用cos)。本文是交替进行，其公式如下：

式中为token在序列中的绝对位置（单词在句子中的绝对位置），它的取值是0到序列最大长度max\_seq\_len-1之间的整数。是位置向量（单词向量）的维度（等同于Channel），与整个模型的隐藏状态维度值相同。是从0到之间的整数值。

采用三角函数的方式来对位置进行编码是由于三角函数的性质：

因此可以推导出，两个位置向量的点积是一个与他们两个位置差值（即相对位置）有关，而与绝对位置无关。这个性质使得在计算注意力权重的时候(两个向量做点积)，使得相对位置对注意力发生影响，而绝对位置变化不会对注意力有任何影响，这更符合常理。但是这里有个缺陷，就是这个相对位置没有正负之分，无法区位置的前后顺序。

ViT Position Embedding

在ViT（Vision Transformer）中，其位置编码为一个一维绝对的编码，是可以学习的。其主要是将Transformer中的编码公式使用一个Embedding来学习。

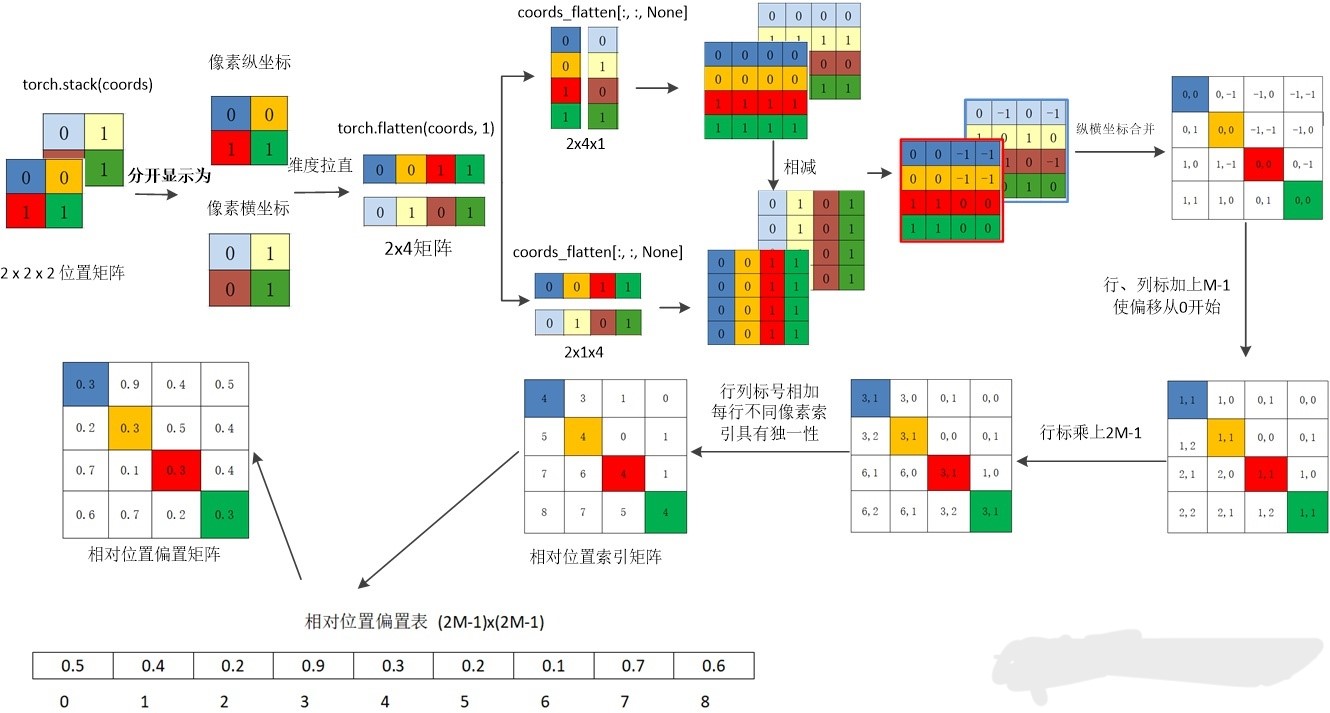
MAE Position Embedding

在MAE（Masked Autoencoder）中，其位置编码为一个二维绝对的编码，根据位置所决定的sin，cos常量，是不可以学习的。其编码公式与Transformer的Position Embedding类似。首先计算各行的绝对索引，展平后将其送入Transformer的Position Embedding中得到输出。然后计算各列的绝对索引，同样展平后将其送入另一个Position Embedding中得到输出，最后将两个输出拼接起来得到结果。

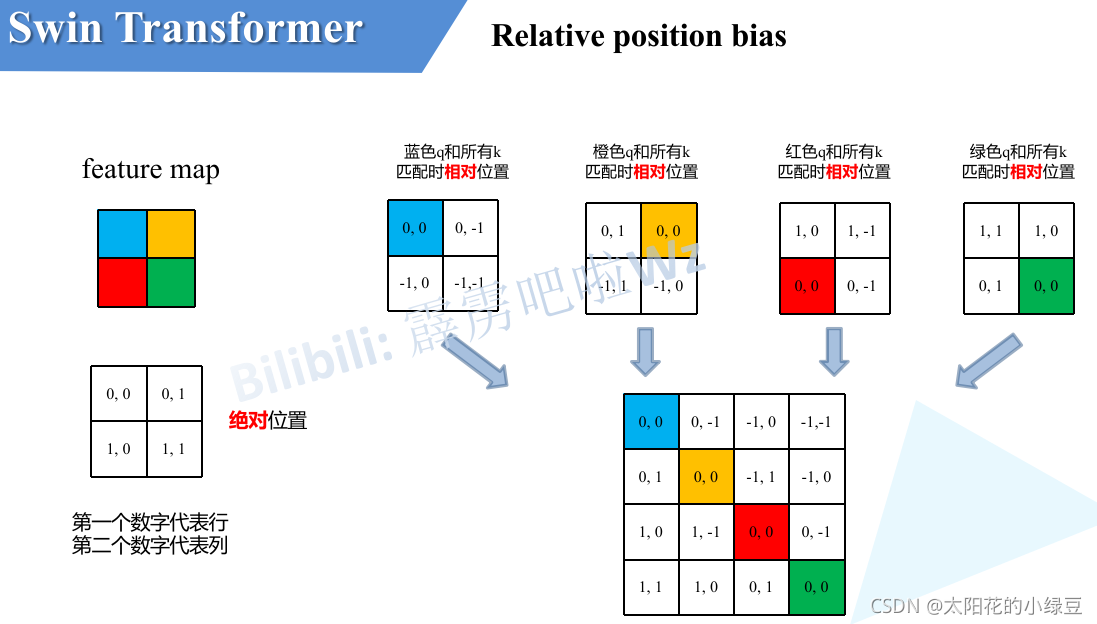
Swin Transformer Position Embedding

在Swin Transformer中，其位置编码为一个二维相对的编码，是可以学习的。假设获取的图片高和宽为height和width（在Swin Transformer中实际为包含patch的window）。由于将二维相对坐标(i, j)映射为一维相对坐标时，最简单的映射方式是将i和j相加，但这样无法区分(i, j)和(j, i)，因此作者采用了i\*height+j的方法来计算相对距离，最终得到一个[height×width，height×width]的矩阵，将矩阵中的值在相对位置编码表中取得对应的值得到最终的结果，最后展平后记作也计算相对位置偏置，用于计算注意力时使用，即如下公式所示：

对于其他的Position Embedding则是在attention之前注入。对于其编码的具体步骤，可以参考如下示意图：



其中设置height=width=2。计算得到一个4×4的距离表。



对于索引矩阵，有一个简单的表示方法，即通过对应蓝、黄、红、绿的绝对位置进行相见，得到对应的相对位置，再将四个矩阵展平后拼接起来。

注入Position Embedding

常见的注入Position Embedding均为采取词向量和位置向量直接相加的方式。对于词向量 E 和位置向量 P 相加再经过后续的线性变换，可以理解为：

而当两个向量拼接后经过后续的线性变换，可以理解为:

那么当时，二者的效果是等价的。我们可以由此发现，拼接总是不会比相加差的，但是由于参数量的增加，其学习难度也会进一步上升。并且通常，在词向量和位置向量相加之前，还会进行线性映射，那么公式（2）中的就可以视为线性映射的矩阵。

在Swin-Transformer中，其注入Position Embedding是在attention中加入，稍有不同。