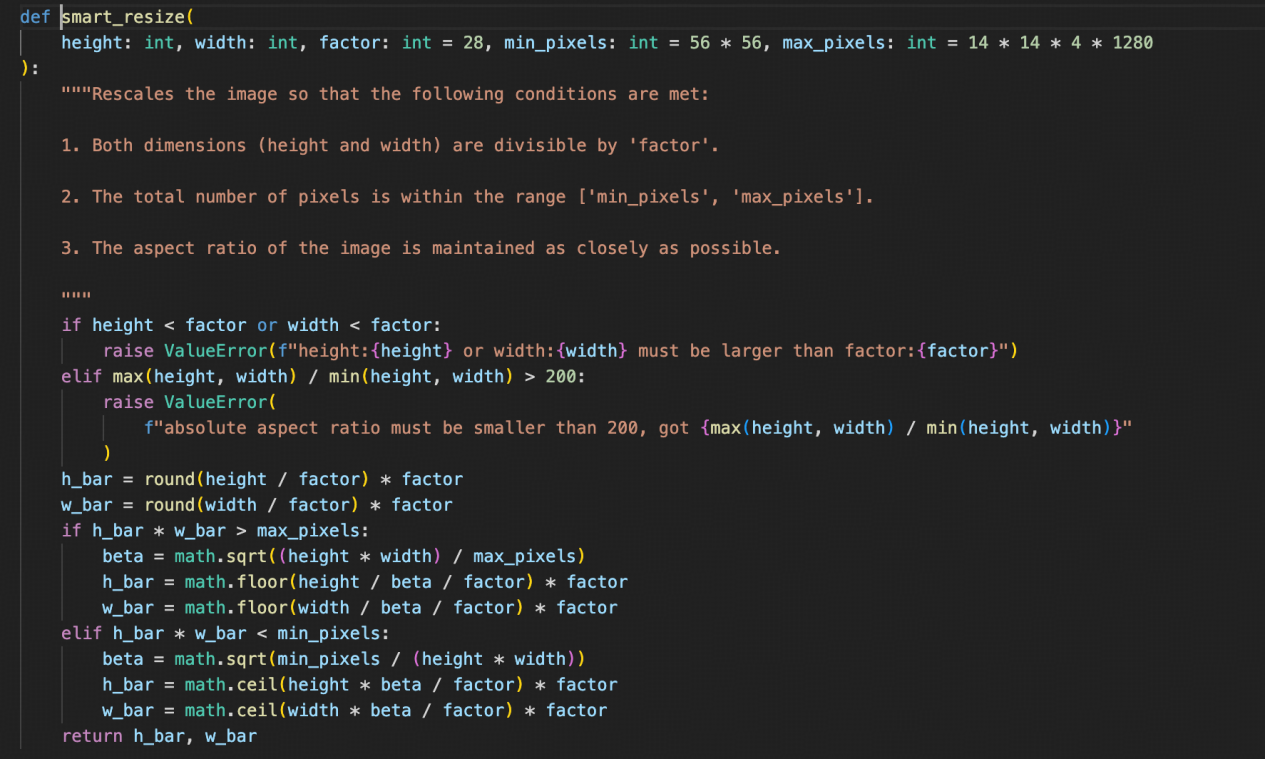
**qwen2-vl**

**smart\_resize**

这个部分的代码主要做的是图片的h和w能被28整除。并控制在一定范围内。

为什么呢？因为当vision model处理完图片数据后，会把2\*2个token会压缩成1一个token送入语言模型，一个token的大小对应14\*14分辨率（patch\_size是14），因此需要最后不多不少，正好全部转换。



|  |  |
| --- | --- |
|  |  |

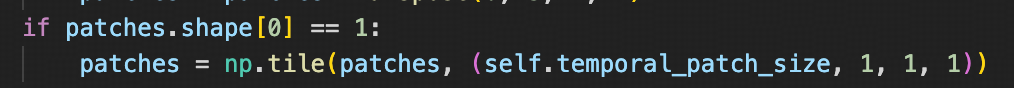
**图片预处理的reshape**

为了能处理video，图片除了(c,h,w)还加了时间维度t，变为(t,c,h,w)。

懂vit的都知道一张图片转为token，使用个conv2d即可完成

但是多了一个时间维度t，在转换token的时候需要考虑时间t这个维度，因此用的是conv3d

图片的时间维度一般是1，也是给时间维度进行了扩充，就是多张一样的图片叠一起，并且也是希望到时候卷积卷的时候不需要在t维度上进行padding，直接给扩了t\_patch\_size倍



之后就是这个无敌的reshape➕transpose，估计空间想象能力差的直接被干懵逼了。

主要分为三个部分吧，第一个部分是vision\_token的排列顺序，第二个部分需要merge的范围，第三个是转token时所需的信息。分别来看

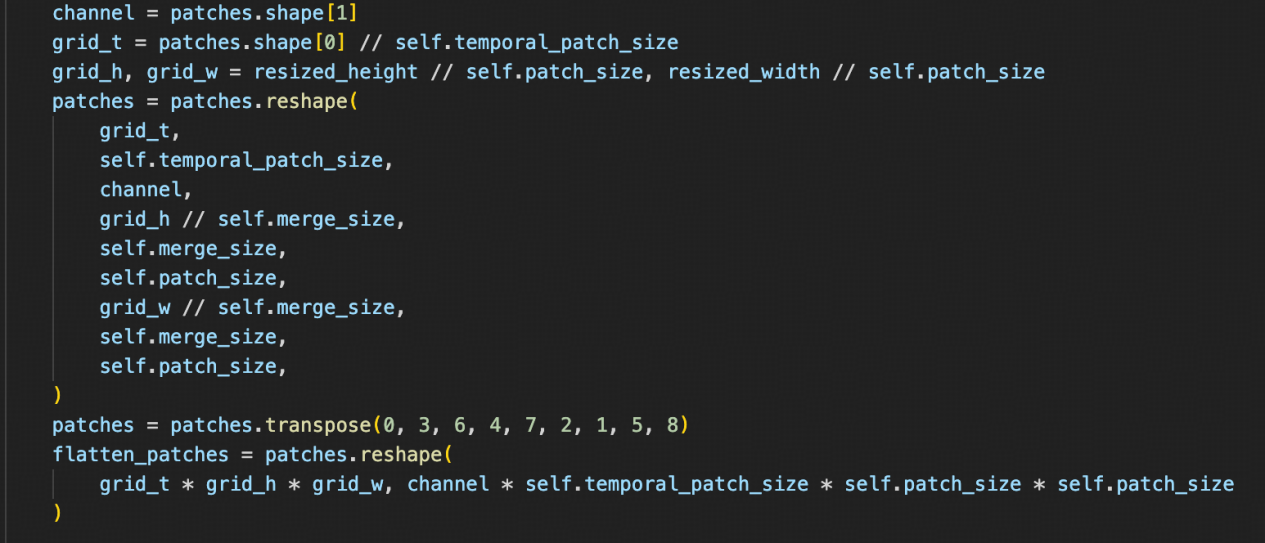
1.视觉token在语言模型中还是在视觉模型中token排列顺序都是（t，h，w）顺序进行排序的，这三个需要放前面

即0，3，6

2.前面说了最后需要每4个视觉token压缩为1个送入语言模型，将两个merge\_size（等于2）放在了中间，即4，7

3.剩下就是卷积需要卷的了，Channel放第一，作为输入通道数，然后后面每t\_size\*p\_size\*p\_size个像素转换为一个token。即2，1，5，8.

最后转为2维的tensor，因为图片分辨率是变化的，一个batch进来，没法保证所有图片的token数都一样，无法使用高维度tensor进行表示



**vision model**

Vision model主要由3个部分组成，embed、encoder和merger

1.embed：

将后三个信息转换为一个维度是embed\_dim的token



2.encoder

先看下vision model的forward

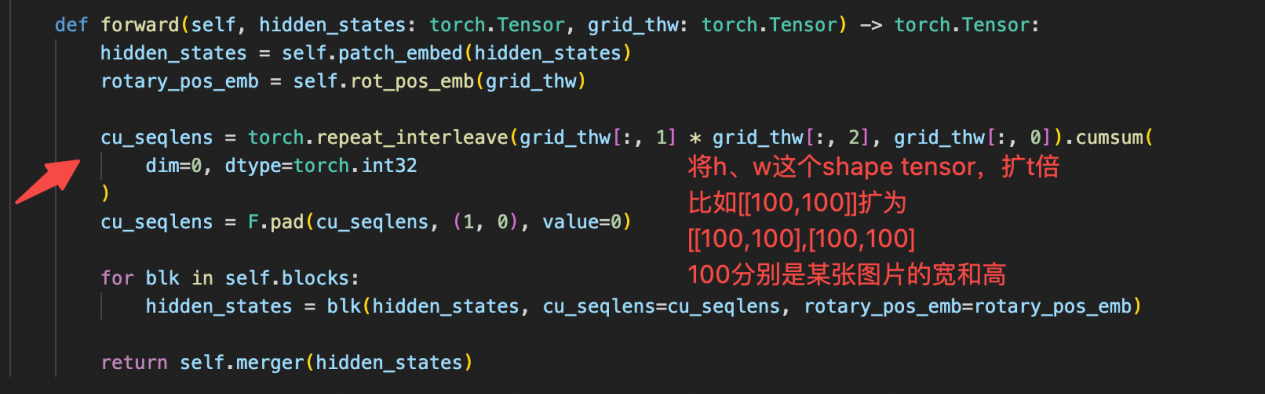
会使用var\_len\_flash\_attn都知道cu\_seqlens主要记录中batch中每个句子结束位置（将batch去掉padding，拼为一个长句子）

在这里他使用cu\_seqlens的意义是什么呢？

他希望同一张图片（video）在不同时间t上，分别单独的计算注意力。而图片的t\_size为1，这个主要处理video。

因此所有图片（video）的token转换为一个长序列token，序列很长，因此不建议使用原生的attn，而使用flash-attn

剩下的和vit一样（最后讲rotary embedding）



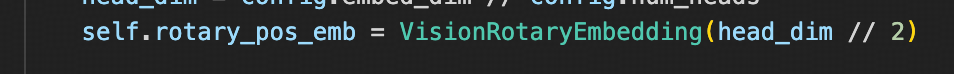
3.patch\_merger

看代码吧



4.2d rotary

首先是rotary的dim是head\_dim的一半。这是为什么呢？



pos\_ids的代码也在下图。我主要说下两个操作吧，一个是stack一个是flatten(1)。

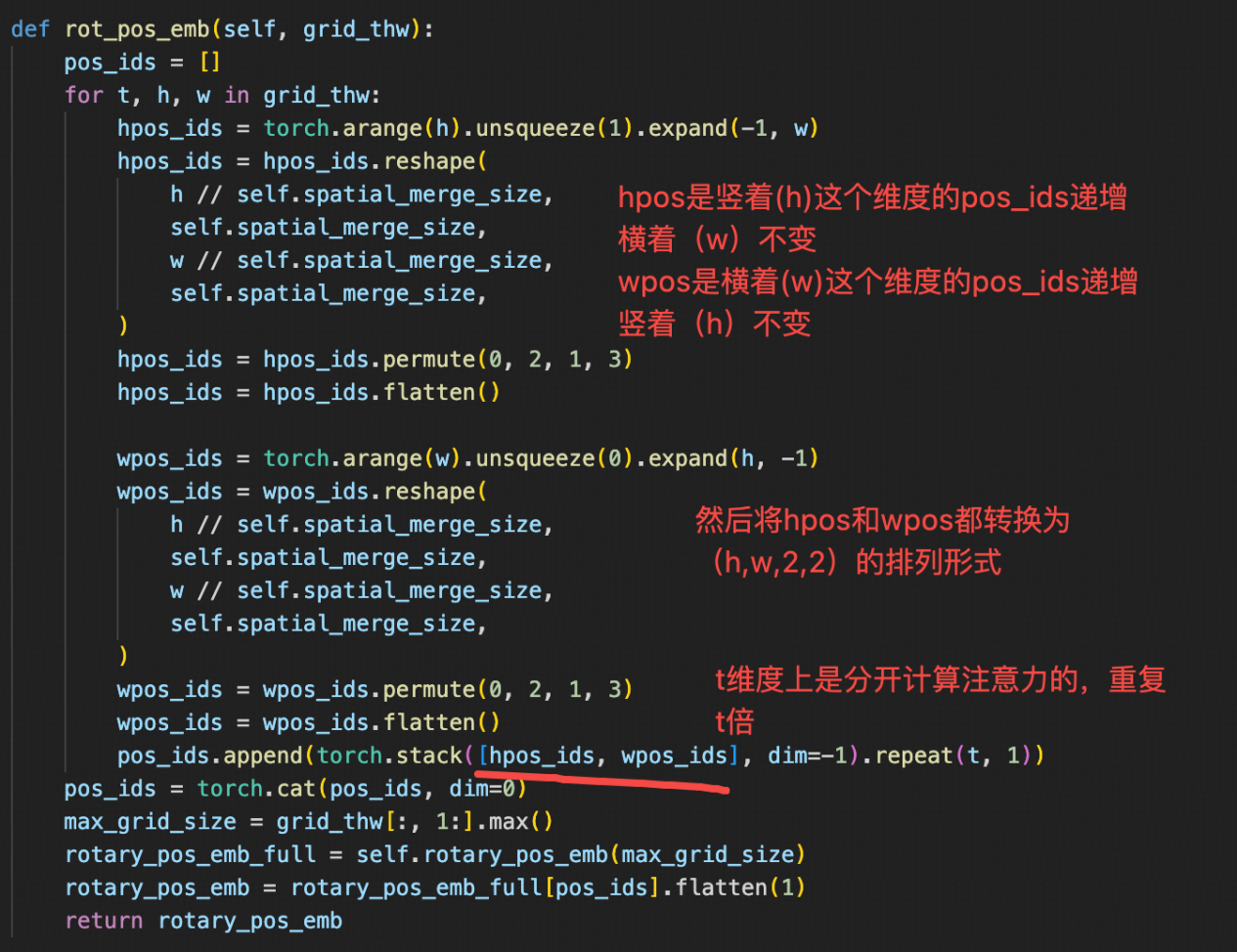
stack后的维度是（total\_len, 2）

过rotary后是(total\_len,2, head\_dim//2)

拉平就是(total\_len, head\_dim) 得到最终的旋转信息

什么含义呢？

比如一个视觉token的hd\_size是64，前32维度是使用hpos这个信息进行旋转，后32维使用wpos这个信息进行旋转。因此一个token前32维描述了该token在h中的位置，后32维描述了token在w中的位置



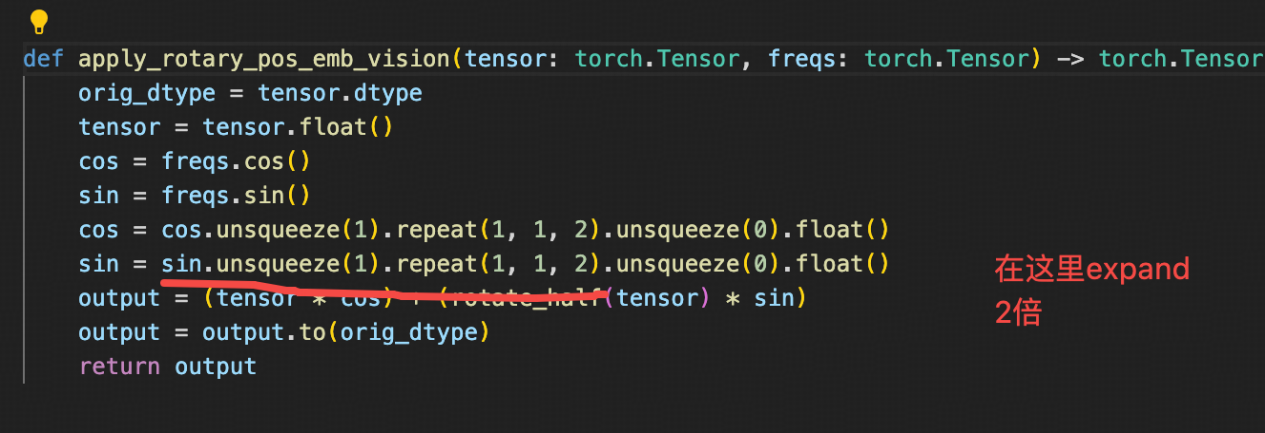
想要这么做还有两个细节就是如下图所示

还以64维举例，对1和33旋转同一个（关于h）角度，对17和49旋转同一个（关于w）角度

。。。。。。。。。.。。。。。。。。。。。。。。。。。。。。。。。

对16和48旋转同一个（关于h）角度，对32和64旋转同一个（关于w）角度



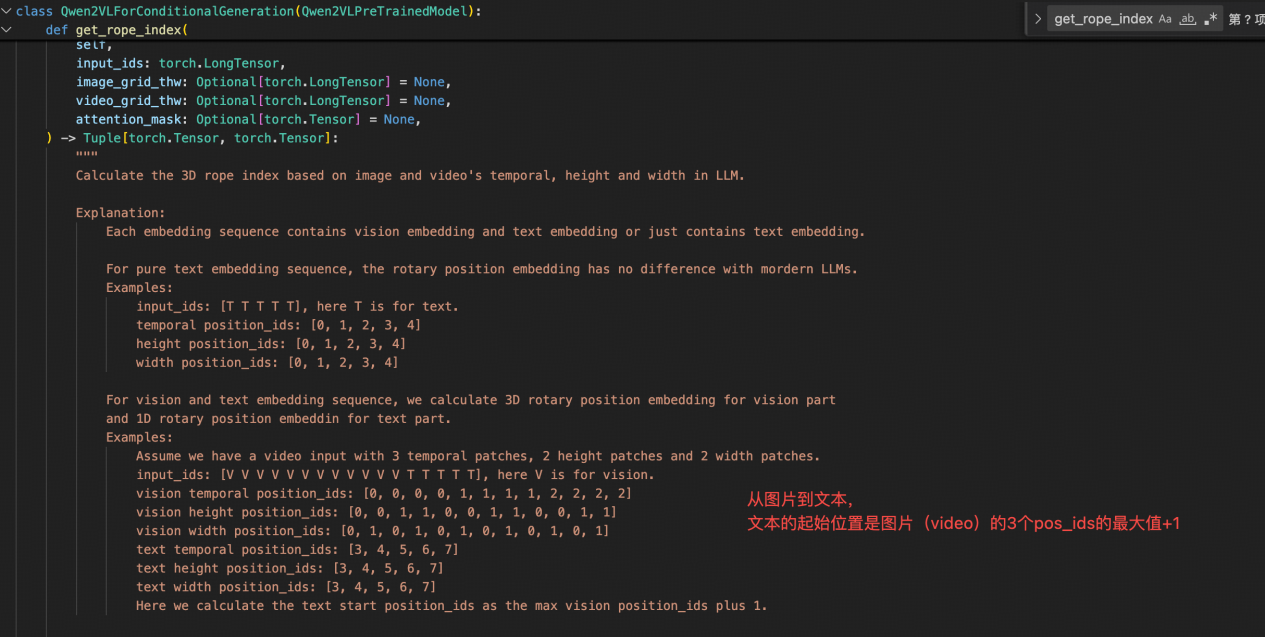


**3d rotary**

剩下一个技术就是3d rotary了

position\_ids一共有三套

即文本+时间t、文本+高度h，文本+宽度w



3套position\_ids如何联合使用呢？

cos和sin都是(3， bs, seq\_len, head\_dim)

mcrope\_section = [16,24,24,16,24,24]加起来刚好128等于head\_dim

下面代码主要表达的是

1-16，65-80使用信息t进行旋转（注意1和64旋转一个角度， 。。。。16和80旋转一个角度）

25-48，81-104使用信息h进行旋转

49-64，105-128使用信息w进行旋转



**结束**

到此就结束了，没有什么了特别的了

感觉模型专门用来推理的，如果要训练的话（transformers的版本），两个建议

1.看看get\_rope\_index是怎么实现的，弄清position\_ids的原理，加入数据处理函数