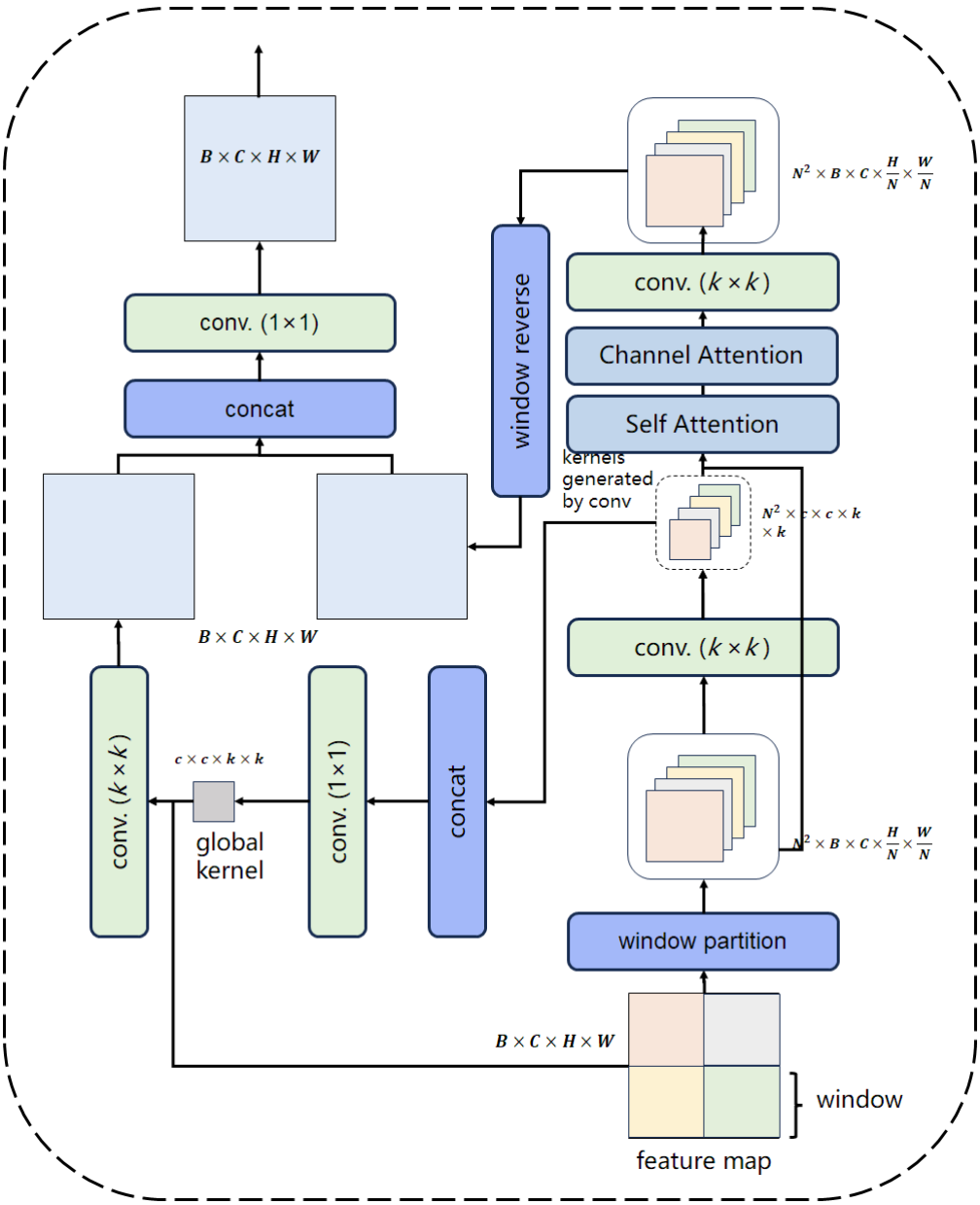
基于PSRT的窗口自适应改进

实验说明

基于PSRT的改进

之前写代码是用for实现窗口卷积的串行运行结构；后来用分组卷积实现了并行运行，但是并行的代码一开始写错了，所以这里区分开。串行、并行代码除了速度上的区别，还有SE模块参数是否窗口共享的区别。

下面是一张窗口生成的卷积核与窗口进行卷积的示意图；左下角是将窗口生成卷积核融合成一个全局卷积核，新卷积核与全图卷积。最后融合窗口卷积和全图卷积得到的feature map。模型解释里写的卷窗口、卷全图特指第二次卷积。



双分支结构（加粗为效果相对较好的尝试，PSRT_noshuffle是baseline）：

- **PSRT_noshuffle**：把PSRT的shuffle都变成普通的Swin Block
- **PSRT_KAv1_noshuffle**：卷积核由池化生成，自注意力、SE计算后去卷全图，卷积核暴力升维（c->c**2），与原图重新计算卷积，1*1卷积融合得到的四张图。并行

- [code error] PSRT_KAv2_noshuffle：把卷窗口的KA放进noshuffle的PSRT中，并行。KAv2的代码有错误，一个维度转换有问题；需要注意，没有for会比有for少三个SE，SE的参数是共享的
- PSRT_KAv3_noshuffle：卷窗口的KA，串行
- PSRT_KAv4_noshuffle：卷窗口的KA，窗口生成卷积核融合成一个全局卷积核（记为global kernel，1*1卷积实现），窗口卷积核与窗口卷积，全局卷积核与全图卷积，融合得到的五张图为一张图（1*1卷积）。串行
- **PSRT_KAv5_noshuffle**：卷窗口的KA，kernels融合成global kernel，只用global kernel与全图卷积。串行
- PSRT_KAv6_noshuffle：卷窗口的KA，kernels融合成global kernel，窗口核和全局核都卷全局，然后fusion。串行
- **PSRT_KAv7_noshuffle**：基于KAv2和KAv6，尝试解决了维度转换的错误，SE的参数依然是多卷积核共享。窗口生成的卷积核与全图计算卷积，然后融合
- PSRT_KAv8_noshuffle：与KAv6思想相同，se的参数是窗口核和全局核共享
- PSRT_KAv9_noshuffle：基于KAv1，生成卷积核增加c的维度的方法改为repeat，c*2->c*c后进行一个参数为cc的linear

测试结果

PSRT模型改进的测试结果

PSRT设置bs=32，lr=1e-4，embed_dim=48

模型	SAM	ERGAS	PSNR	参数量	note
PSRT(embed_Dim=32)	-	-	-	0.248 M	-
PSRT(embed_Dim=64)	-	-	-	0.939 M	-
PSRT(embed_Dim=48)	2.2407495	2.4452974	50.0313946	0.538 M	
PSRT_noshuffle	**2.1245276**	**2.2309420**	**50.4692293**	**0.538 M**	
PSRT_KAv1_noshuffle	**2.2294778**	**1.3029419**	**50.7237681**	**0.779 M**	
PSRT_KAv2_noshuffle	2.2752936	2.0677896	49.6950313	0.854 M	
PSRT_KAv3_noshuffle	2.2756061	1.7408064	50.1445174	0.918 M	
PSRT_KAv4_noshuffle	2.1899021	2.3440072	50.2209833	1.002 M	
PSRT_KAv5_noshuffle	**2.1078129**	**2.2032974**	**50.5076604**	**1.002 M**	
PSRT_KAv6_noshuffle	4.7182505	3.9199647	40.0239899	1.054 M	
PSRT_KAv7_noshuffle	**2.1232879**	**2.1154806**	**50.4642246**	**0.894 M**	
PSRT_KAv8_noshuffle	2.1751094	2.4212308	50.3579216	0.946 M	
PSRT_KAv9_noshuffle				0.519 M	

PSRT_KAv6_noshuffle怀疑是过拟合了，2000epoch时，PSNR只有40；1999epoch时，PSNR有50.26；1998epoch时，PSNR有50.43；1500epoch时，PSNR有50.24

一些暂未尝试的改进方向

- 进入Kernel Attention前的LayerNorm去掉，保留Window Attention的LayerNorm?
- 并行代码里，SE的参数是否需要共享？共享的话global kernel是否共享?
- 通过池化生成卷积核，卷积核缺失一个c的维度，这个维度从何而来？暴力拓展通道数、repeat，应该有其他更好的方法吧
- 对卷积核计算自注意力和通道注意力，这里能不能只计算通道自注意力。因为3*3卷积核太小了，自注意力计算可以带来负面的影响。或者这里不计算空间自注意力，计算通道自注意力?