

基于PSRT的窗口自适应改进

一些尝试中的结论

相较baseline, PSNR 0.59↑; ERGAS 0.915↑; SAM 0.066↓

20231028

- 最开始提出的对每个窗口计算卷积, 取每个窗口的卷积核计算注意力, 回卷窗口, 拼成feature map的方法是负提升。计算注意力后的卷积核去卷窗口不如去卷全图
- 所有窗口卷积核在通道上concat, 通过1*1卷积暴力生成global kernel (维度c, c, k, k, 普通卷积), 使用global kernel卷全图, global kernel有提升但有限
- 卷积核计算自注意力可能带来负提升 (还差一个实验跑出来验证)

20231116

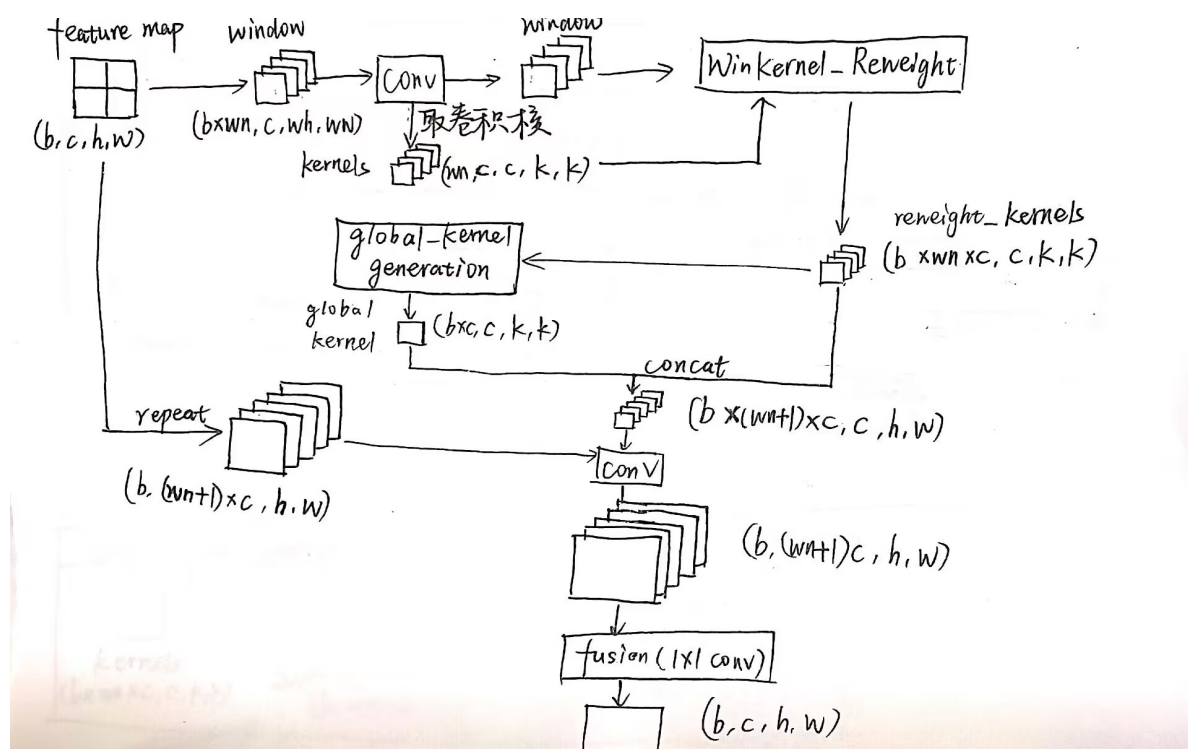
主要有四个改进点: 有无卷积核SA, 有无卷积核SE, 有无全局卷积核, 有无卷积核赋权

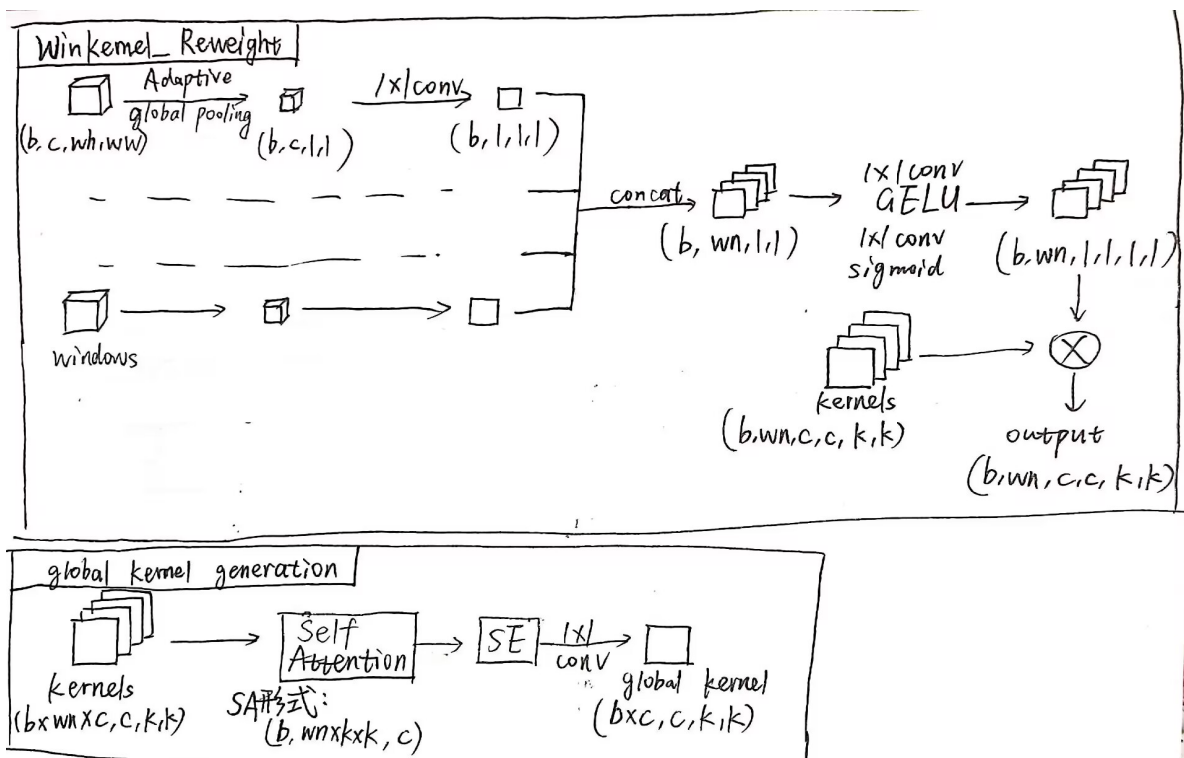
- 在没有全局卷积核时, SA带来效果提升, SE和卷积核赋权会使效果变差
- 在有全局卷积核时, 卷积核赋权带来效果提升, SA和SE会使效果变差

实验说明

基于PSRT的改进

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:

baseline

PSRT设置bs=32, lr=1e-4, embed_dim=48

- **PSRT_noshuffle**: 把PSRT的shuffle都变成普通的Swin Block

有global kernel

- **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_KAv11_noshuffle**: 基于KAv5和KAv7, 卷全图, 没有SA和SE, 有global kernel。并行
- **PSRT_KAv12_noshuffle**: 基于KAv10和KAv11, 卷全图, 有SA无SE, 有global kernel。并行
- **PSRT_KAv16_noshuffle**: 基于KAv5和KAv7, 卷全图, 没有SA有SE, SE的激活函数改为GELU; 有global kernel。并行
- **PSRT_KAv17_noshuffle**: 基于KAv11; 无SA和SE; 都和原图进行第二次卷积; 有global kernel; 窗口卷积核使用第一次卷积的window赋权

模型	SAM	ERGAS	PSNR	参数量	SA	SE	global_kernel	第二次卷积	卷积核赋权
baseline	2.1187720	2.1811231	50.4297113	0.538 M	-	-	-	-	-
5	2.1078129	2.2032974	50.5076604	1.002 M	×	×	√	原图	×
6	4.7182505	3.9199647	40.0239899	1.054 M	√	√	√	原图	×
11	2.1693590	1.4011621	50.8749442	0.881 M	×	×	√	原图	×
12	2.3742382	1.2469189	50.6505637	0.851 M	√	×	√	原图	×
16	2.3273963	1.2449526	50.4512170	0.901 M	×	×	√	原图	×
17	2.1851830	1.2657717	51.0142954	0.884 M	×	×	√	原图	√
21				0.554 M	×	×	√	原图	√

有global kernel、不进行卷积核的SA和SE效果最好

无global kernel

- **PSRT_KAv7_noshuffle**: 基于KAv2和KAv6, 卷积核共享SE参数。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_KAv10_noshuffle: 基于KAv7, 卷全图, 不加SE模块, 无global kernel。并行
- PSRT_KAv13_noshuffle: 基于KAv11, 卷全图, 不加SA, 无global kernel。并行, 加GELU
- PSRT_KAv18_noshuffle: 基于KAv11; 卷全图无SA和SE; 都和原图进行第二次卷积; 无global kernel; 窗口卷积核使用第一次卷积的window赋权
- PSRT_KAv19_noshuffle: 基于KAv11, 卷全图, 不加SA和SE, 无global kernel。

模型	SAM	ERGAS	PSNR	参数量	SA	SE	global_kernel	第二次卷积	卷积核赋权
baseline	2.1187720	2.1811231	50.4297113	0.538 M	-	-	-	-	-
7	2.1232879	2.1154806	50.4642246	0.894 M	√	√ (共享)	×	原图	×
8	2.1751094	2.4212308	50.3579216	0.946 M	√	√ (共享)	×	原图	×
10	2.2156852	1.4317201	50.7399171	0.894 M	√	×	×	原图	×
13	2.1941420	2.4338021	50.1611231	0.894 M	×	√	×	原图	×
18	2.3828535	1.3595995	50.2718298	0.832 M	×	×	×	原图	√
19	2.5441515	1.3270533	49.6788777	0.832 M	×	×	×	原图	×

Conv-GELU-Conv结构

- PSRT_KAv14_noshuffle: 基于KAv11, SE的激活函数改为GELU; SE放在SA前面; 都和reverse后的feature map进行第二次卷积。不收敛
- PSRT_KAv15_noshuffle: 基于KAv14, 在attention加shortcut, 不收敛

卷窗口

- [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

模型	SAM	ERGAS	PSNR	参数量	note
baseline	2.1187720	2.1811231	50.4297113	0.538 M	
2	2.2752936	2.0677896	49.6950313	0.854 M	
3	2.2756061	1.7408064	50.1445174	0.918 M	
4	2.1899021	2.3440072	50.2209833	1.002 M	

池化生成卷积核

- 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($c \rightarrow c^{**2}$), 与原图重新计算卷积, 1*1卷积融合得到的四张图。并行
- PSRT_KAv9_noshuffle: 基于KAv1, 生成卷积核增加c的维度的方法改为repeat, $c^{*2} \rightarrow c^{*c}$ 后进行一次参数为cc的linear

模型	SAM	ERGAS	PSNR	参数量	note
baseline	2.1187720	2.1811231	50.4297113	0.538 M	
1	2.2294778	1.3029419	50.7237681	0.779 M	
9	2.2132997	3.2366958	50.0673282	0.519 M	

需要讨论

20231028

- 为什么去掉PSRT的shuffle效果会有提升？ baseline没有滑动窗口
- 进入Kernel Attention的张量不保留LayerNorm，保留Window Attention的LayerNorm？ (EDSR)
- global kernel的有效性，生成global kernel时是否需要进行窗口kernels的注意力计算？ kernels注意力计算和生成global kernel的先后？ global kernel和kernels要不要一起计算注意力？
- 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？ 两个思路，b->c，这里b, c, k, k在b的维度上加起来； c->c，这里可以暴力拓展通道数、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，自注意力计算可能带来负面的影响。或者这里不计算空间自注意力，计算通道自注意力？ 3*3->5*5？
- kernels是否共享SE的参数？ global kernel是否共享？

20231109

- 卷积核的SA不能做B, win_num, c*wh*ww，这样全连接层参数爆炸
- 卷积核的Attention效果确实会下降？
- 卷积核赋权使用SA都不可行，(bs, 4, c*k*k)，生成qkv的过程参数量是 $k^4 \times c^2$ 。这部分我用类似SE的过程实现的
- GELU是对B C H W做还是对B L C做
- ConvNeXt为什么在Attention部分使用大核卷积？ 同理SegNeXt
- 如果使用两个卷积替换Attention，要不要加shortcut
- Conv-GELU-Conv结构为什么不收敛