RepLKNet:2022

Scaling Up Your Kernels to 31×31：Revisiting Large Kernel Design in CNNs

论文出发点和背景

受vision transformer的启发，然后使用大卷积核对视觉任务进行处理

transformer中的MHSA的长建模能力是transformer能work的重要原因

只有之前老式的网络和一些通过NAS得到的网络结构使用了大卷积核，因而就产生一个问题，如果我们使用大卷积核会有什么效果，会增大还是缩小CNN和ViT之间的差距?

之前的网络尝试用稍微大点的卷积核进行实验，但是并没有显示处更大卷积的好处

论文创新思路

大卷积核

结构重参数化

对比ViT说明增大卷积核的好处与优势

论文方法介绍

使用31×31的卷积核——>大卷积核具有更大的有效感受野和形状偏差，而不是纹理偏差

结构重参数化

将相对较小的内核嵌入一个非常大的内核中，来捕获小规模的patter，提高模型性能

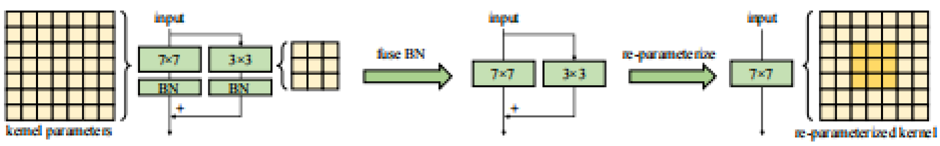

Figure 2. An example of re-parameterizing a small kernel (*e.g.*, 3×3) into a large one (*e.g.*, 7×7). See [27, 30] for details.

通过卷积核大小设计（只是引入大型深度卷积到传统网络），总结出了 5个有效使用大卷积的经验准则：1.在实践时，使用非常大的卷积是很有效的（通过使用深度可分离卷积减少计算上的消耗）；2.identity支路在超大卷积核网络中非常重要（在mobile v2中将所有的DW3×3替换为13×13）；3.用小卷积核重新参数化可以缓解优化问题；4.大卷积核在下游任务中表现更好(大卷积核的有效感受野更大，人类识别物体的方式主要是基于形状线索，而不是基于纹理)；5.大卷积核在小的特征映射上面也是有效的。

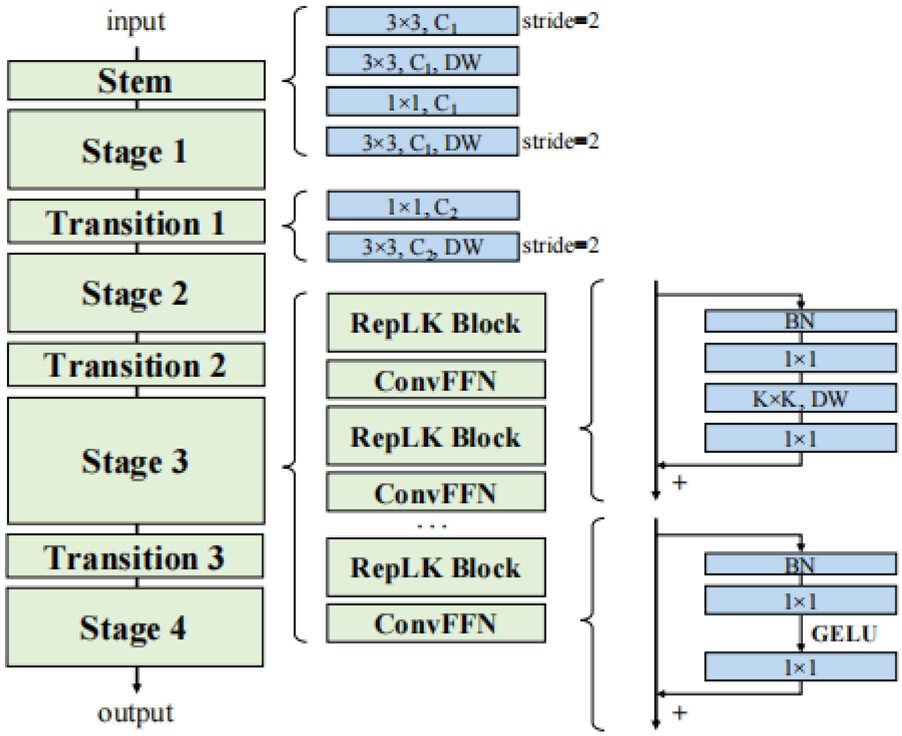

Figure 4. RepLKNet comprises Stem, Stages and Transitions. Except for depth-wise (DW) large kernel, the other components include DW 3×3, dense 1×1 conv, and batch normalization [49] (BN). Note that every conv layer has a following BN, which are not depicted. Such conv-BN sequences use ReLU as the activation function, except those before the shortcut-addition (as a common practice [40, 75]) and those preceding GELU [41].

实际效果

Table 1. Inference speed of a stack of 24-layer depth-wise convolutions with various kernel sizes and resolutions on a single GTX 2080Ti GPU. The input shape is (64, 384, R, R). Baselines are evaluated with Pytorch 1.9.0 + cuDNN 7.6.5, in FP32 precision.

Resolution R	Impl	Latency (ms) @ Kernel size 3	5	7	9	13	17	21	27	29	31
16 × 16	Pytorch	5.6	11.0	14.4	17.6	36.0	57.2	83.4	133.5	150.7	171.4
	Ours	5.6	6.5	6.4	6.9	7.5	8.4	8.4	8.4	8.3	8.4
32 × 32	Pytorch	21.9	34.1	54.8	76.1	141.2	230.5	342.3	557.8	638.6	734.8
	Ours	21.9	28.7	34.6	40.6	52.5	64.5	73.9	87.9	92.7	96.7
64 × 64	Pytorch	69.6	141.2	228.6	319.8	600.0	977.7	1454.4	2371.1	2698.4	3090.4
	Ours	69.6	112.6	130.7	152.6	199.7	251.5	301.0	378.2	406.0	431.7

Table 2. Results of different kernel sizes in normal/shortcut-free MobileNet V2.

Shortcut	Kernel size	ImageNet top-1 accuracy (%)
✓	3×3	71.76
✓	13×13	**72.53**
	3×3	**68.67**
	13×13	53.98

reduces from 49.5% to 12.3%, which is roughly in proportion to the FLOPs occupation.

Table 4. Results of various kernel sizes in the *last stage* of MobileNet V2. Kernel sizes in previous stages remain to be 3×3.

Kernel size	ImageNet acc (%)	Cityscapes mIoU (%)
3×3	71.76	72.31
7×7	**72.00**	74.30
13×13	71.97	**74.62**

Table 5. RepLKNet with different kernel sizes. The models are pretrained on ImageNet-1K in 120 epochs with 224×224 input and finetuned on ADE20K with UperNet in 80K iterations. On ADE20K, we test the *single-scale* mIoU, and compute the FLOPs with input of 2048×512, following Swin.

	ImageNet			ADE20K		
Kernel size	Top-1	Params	FLOPs	mIoU	Params	FLOPs
3-3-3-3	82.11	71.8M	12.9G	46.05	104.1M	1119G
7-7-7-7	82.73	72.2M	13.1G	48.05	104.6M	1123G
13-13-13-13	83.02	73.7M	13.4G	48.35	106.0M	1130G
25-25-25-13	83.00	78.2M	14.8G	48.68	110.6M	1159G
31-29-27-13	83.07	79.3M	15.3G	49.17	111.7M	1170G

K. So that a RepLKNet architecture is defined by $[B_1, B_2, B_3, B_4]$,$[C_1, C_2, C_3, C_4]$,$[K_1, K_2, K_3, K_4]$.

Table 6. ImageNet results. The throughput is tested with FP32 and a batch size of 64 on 2080Ti. ‡ indicates ImageNet-22K pretraining. ⋄ indicates pretrained with extra data.

Model	Input resolution	Top-1 acc	Params (M)	FLOPs (G)	Throughput examples/s
RepLKNet-31B	224×224	83.5	79	15.3	295.5
Swin-B	224×224	83.5	88	15.4	226.2
RepLKNet-31B	384×384	84.8	79	45.1	97.0
Swin-B	384×384	84.5	88	47.0	67.9
RepLKNet-31B ‡	224×224	85.2	-	-	-
Swin-B ‡	224×224	85.2	-	-	-
RepLKNet-31B ‡	384×384	86.0	-	-	-
Swin-B ‡	384×384	86.4	-	-	-
RepLKNet-31L ‡	384×384	86.6	172	96.0	50.2
Swin-L ‡	384×384	87.3	197	103.9	36.2
RepLKNet-XL ⋄	320×320	87.8	335	128.7	39.1

Table 7. Cityscapes results. The FLOPs is computed with 1024×2048 inputs. The mIoU is tested with single-scale (ss) and multi-scale (ms). The results with Swin are implemented by [36]. ‡ indicates ImageNet-22K pretraining.

Backbone	Method	mIoU (ss)	mIoU (ms)	Param (M)	FLOPs (G)
RepLKNet-31B	UperNet [97]	**83.1**	**83.5**	110	2315
ResNeSt-200 [107]	DeepLabv3 [14]	-	82.7	-	-
Axial-Res-XL	Axial-DL [92]	80.6	81.1	173	2446
Swin-B	UperNet	80.4	81.5	121	2613
Swin-B	UperNet + [36]	80.8	81.8	121	-
ViT-L ‡	SETR-PUP [112]	79.3	82.1	318	-
ViT-L ‡	SETR-MLA	77.2	-	310	-
Swin-L ‡	UperNet	82.3	83.1	234	3771
Swin-L ‡	UperNet + [36]	82.7	83.6	234	-

Table 8. ADE20K results. The mIoU is tested with single-scale (ss) and multi-scale (ms). The results with 1K-pretrained Swin are cited from the official GitHub repository. ‡ indicates ImageNet-22K pretraining and 640×640 finetuning on ADE20K. ⋄ indicates pretrained with extra data. The FLOPs is computed with 2048×512 for the ImageNet-1K pretrained models and 2560×640 for the ImageNet-22K and larger, following Swin.

Backbone	Method	mIoU (ss)	mIoU (ms)	Param (M)	FLOPs (G)
RepLKNet-31B	UperNet	**49.9**	**50.6**	112	1170
ResNet-101	UperNet [97]	43.8	44.9	86	1029
ResNeSt-200 [107]	DeepLabv3 [14]	-	48.4	113	1752
Swin-B	UperNet	48.1	49.7	121	1188
Swin-B	UperNet + [36]	48.4	50.1	121	-
ViT-Hybrid	DPT-Hybrid [71]	-	49.0	90	-
ViT-L	DPT-Large	-	47.6	307	-
ViT-B	SETR-PUP [112]	46.3	47.3	97	-
ViT-B	SETR-MLA [112]	46.2	47.7	92	-
RepLKNet-31B ‡	UperNet	**51.5**	**52.3**	112	1829
Swin-B ‡	UperNet	50.0	51.6	121	1841
RepLKNet-31L ‡	UperNet	**52.4**	52.7	207	2404
Swin-L ‡	UperNet	52.1	**53.5**	234	2468
ViT-L ‡	SETR-PUP	48.6	50.1	318	-
ViT-L ‡	SETR-MLA	48.6	50.3	310	-
RepLKNet-XL ⋄	UperNet	**55.2**	**56.0**	374	3431

个人理解

当时一看到题目就想到RepVGG，然后文章就用了结构重参数化，感觉和昨晚看的A Convnet for the 2020s的方向都很相似，通过一些“现代化”的设计进行改造卷积神经网络，然后将卷积神经网络拿去和transformer比性能，证明现在CNN还是挺能造的。感觉这两篇文章都有点像实验报告的感觉，其实都是在说大卷积核是有用的，和vision transformer可以相比较，然后用一些例如深度可分离卷积、结构重参数化等操作来减轻运算消耗。