Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs
Appendix

Appendix A: Training Configurations

ImageNet-1K
For training MobileNet V2 models (Sec. 3), we use 8 GPUs, an SGD optimizer with momentum of 0.9, a batch size of 32 per GPU, input resolution of 224×224, weight decay of $4 \times 10^{-5}$, learning rate schedule with 5-epoch warmup, initial value of 0.1 and cosine annealing for 100 epochs. For the data augmentation, we only use random cropping and left-right flipping, as a common practice.

For training RepLKNet models (Sec. 4.2), we use 32 GPUs and a batch size of 64 per GPU to train for 120 epochs. The optimizer is AdamW [10] with momentum of 0.9 and weight decay of 0.05. The learning rate setting includes an initial value of $4 \times 10^{-3}$, cosine annealing and 10-epoch warmup. For the data augmentation and regularization, we use RandAugment [4] (“rand-m9-mstd0.5-inc1” as implemented by timm [15]), label smoothing coefficient of 0.1, mixup [18] with $\alpha = 0.8$, CutMix with $\alpha = 1.0$, Rand Erasing [19] with probability of 25% and Stochastic Depth with a drop-path rate of 30%, following the recent works [1, 8, 9, 12]. The RepLKNet-31B reported in Sec. 4.3 is trained with the same configurations except the epoch number of 300 and drop-path rate of 50%.

For finetuning the 224×224-trained RepLKNet-31B with 384×384, we use 32 GPUs, a batch size of 32 per GPU, initial learning rate of $4 \times 10^{-4}$, cosine annealing, 1-epoch warmup, 30 epochs, model EMA (Exponential Moving Average) with momentum of $10^{-4}$, the same RandAugment as above but no CutMix nor mixup.

For finetuning RepLKNet-31B/L with 384×384, we use 32 GPUs and a batch size of 16 per GPU, and the drop-path rate is raised to 30%.

RepLKNet-XL and Semi-supervised Pretraining
We continue to scale up our architecture and train a ViT-L [6] level model named RepLKNet-XL. We use $B = [2, 2, 18, 2], C = [256, 512, 1024, 2048], K = [27, 27, 27, 13]$, and introduce inverted bottleneck with expansion ratio of 1.5 to each RepLK Block. During pretraining, we use a private semi-supervised dataset named MegData73M, which contains 38 million labeled images and 35 million unlabeled ones. Labeled images come from public and private classification datasets such as ImageNet-1K, ImageNet-22K and Places365 [20]. Unlabeled images are selected from YFCC100M [11]. We design a multi-task label system according to [7], and utilize soft pseudo labels which are offline generated by multiple task-specific ViT-Ls wherever human annotations are unavailable. We pre-train our model for up to 15 epochs with similar configurations as ImageNet-1K pretraining. We do not use CutMix or mixup, decrease drop-path rate to 20%, and use a lower initial learning rate of $1.5 \times 10^{-4}$ and a total batch size of 2048. Structural Re-parameterization is omitted because it only brings less than 0.1% performance gain on such a large-scale dataset. In other words, we observe that the inductive bias (re-parameterization with small kernels) becomes less important as the data become bigger, which is similar to the discoveries reported by ViT [6].

We finetune on ImageNet-1K with input resolution of 320×320 for 30 epochs following BeiT [1], except for a higher learning rate of $10^{-4}$ and stage-wise learning rate decay of 0.4. Finetuning with a higher resolution of 384×384 brings no further improvements. For downstream tasks, we use the default training setting except for a drop-path rate of 50% and stage-wise learning rate decay.

Appendix B: Visualizing the ERF
Formally, let $I(n \times 3 \times h \times w)$ be the input image, $M(n \times c \times h' \times w')$ be the final output feature map, we desire to measure the contributions of every pixel on I to
the central points of every channel on \( M, \) i.e., \( M_{i,j,h'/2,w'/2} \), which can be simply implemented via taking the derivatives of \( M_{i,j,h'/2,w'/2} \) to \( I \) with the auto-grad mechanism. Concretely, we sum up the central points, take the derivatives to remove the negative parts (denoted by \( P \)). Then we aggregate the entries across all the examples and the three input channels, and take the logarithm for better visualization. Formally, the aggregated contribution score matrix \( A(h \times w) \) is given by

\[
P = \max\left(\frac{\partial}{\partial I}\sum_{i}^{n}\sum_{j}^{c} M_{i,j,h'/2,w'/2}, 0\right),
\]

\[
A = \log_{10}\left(\sum_{i}^{n}\sum_{j}^{c} P_{i,j,:,:} + 1\right).
\]  

Then we respectively rescale \( A \) of each model to \([0, 1]\) via dividing the maximum entry for the comparability across models.

Table 10 presents a quantitative analysis, where we report the high-contribution area ratio \( r \) of a minimum rectangle that covers the contribution scores over a given threshold \( t \). For examples, 20% of the pixel contributions (\( A \) values) of ResNet-101 reside within a 103 × 103 area at the center, so that the area ratio is \((103/1024)^2 = 1.0%\) with \( t = 20%\). We make several intriguing observations. 1) While being significantly deeper, ResNets have much smaller ERFs than RepLKNet-31. For example, over 99% of the contribution scores of ResNet-101 reside within a small area which takes up only 23.4% of the total area, while such area ratio of RepLKNet-31 is 98.6%, which means most of pixels considerably contribute to the final predictions. 2) Adding more layers to ResNet-101 does not effectively enlarge the ERF, while scaling up the kernels improves the ERF with marginal computational costs.

Appendix C: Large-Kernel Models have High Shape Bias

A recent work \([13]\) reported that vision transformers are more similar to the human vision systems in that they make predictions more based on the overall shapes of objects, while CNNs focus more on the local textures. We follow its methodology and use its toolbox \([2]\) to obtain the shape bias (e.g., the fraction of predictions made based on the shapes, rather than the textures) of RepLKNet-31B and Swin-B pretrained on ImageNet-1K or 22K, together two small-kernel baselines RepLKNet-3 and ResNet-152. Fig. 5 shows that RepLKNet has higher shape bias than Swin. Considering RepLKNet and Swin have similar overall architectures, we reckon shape bias is closely related to the Effective Receptive Field rather than the concrete formulation of self-attention (i.e., the query-key-value design). This also explains 1) the high shape bias of ViTs \([6]\) reported by \([13]\) (since ViTs employ global attention), 2) the low shape bias of 1K-pretrained Swin (attention within local windows), and 3) the shape bias of the small-kernel baseline RepLKNet-3, which is very close to ResNet-152 (both models are composed of 3 × 3 convolutions).

Appendix D: ConvNeXt + Very Large Kernels

We use the recently proposed ConvNeXt \([9]\) as the benchmark architecture to evaluate large kernel as a generic design element. We simply replace the 7 × 7 convolutions in ConvNeXt \([9]\) by kernels as large as 31 × 31. The training configurations on ImageNet (120 epochs) and ADE20K (80K iterations) are identical to the results shown in Sec. 4.2. Table. 11 shows that though the original kernels are already 7 × 7, further increasing the kernel sizes still brings significant improvements, especially on the downstream task: with kernels as large as 31 × 31, ConvNeXt-Tiny outperforms the original ConvNeXt-Small, and the large-kernel ConvNeXt-Small outperforms the original ConvNeXt-Base. Again, such phenomena demonstrate
Table 11. ConvNeXt 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.

<table>
<thead>
<tr>
<th>Kernel size</th>
<th>Architecture</th>
<th>ImageNet acc</th>
<th>Params</th>
<th>FLOPs</th>
<th>ADE20K mIoU</th>
<th>Params</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-7-7-7</td>
<td>ConvNeXt-Tiny</td>
<td>Top-1</td>
<td>81.0</td>
<td>29M</td>
<td>4.5G</td>
<td>44.6</td>
<td>60M</td>
</tr>
<tr>
<td>7-7-7-7</td>
<td>ConvNeXt-Small</td>
<td>Top-1</td>
<td>82.1</td>
<td>50M</td>
<td>8.7G</td>
<td>45.9</td>
<td>82M</td>
</tr>
<tr>
<td>7-7-7-7</td>
<td>ConvNeXt-Base</td>
<td>Top-1</td>
<td>82.8</td>
<td>89M</td>
<td>15.4G</td>
<td>47.2</td>
<td>122M</td>
</tr>
<tr>
<td>31-29-27-13</td>
<td>ConvNeXt-Tiny</td>
<td>Top-1</td>
<td>81.6</td>
<td>32M</td>
<td>6.1G</td>
<td>46.2</td>
<td>64M</td>
</tr>
<tr>
<td>31-29-27-13</td>
<td>ConvNeXt-Small</td>
<td>Top-1</td>
<td>82.5</td>
<td>48M</td>
<td>8.7G</td>
<td>48.2</td>
<td>90M</td>
</tr>
</tbody>
</table>

Table 12. MobileNet V2 with all regular DW 3×3 layers replaced by 3×3 dilated layers.

<table>
<thead>
<tr>
<th>Max RF</th>
<th>Kernel size</th>
<th>Dilation</th>
<th>ImageNet acc</th>
<th>Params</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>9×9</td>
<td>-</td>
<td>72.67</td>
<td>4.0M</td>
<td>319M</td>
</tr>
<tr>
<td>9</td>
<td>3×3</td>
<td>4</td>
<td>57.23</td>
<td>3.5M</td>
<td>300M</td>
</tr>
<tr>
<td>13</td>
<td>13×13</td>
<td>-</td>
<td>72.53</td>
<td>4.6M</td>
<td>361M</td>
</tr>
<tr>
<td>13</td>
<td>3×3</td>
<td>6</td>
<td>51.21</td>
<td>3.5M</td>
<td>300M</td>
</tr>
</tbody>
</table>

that kernel size is an important scaling dimension.

Appendix E: Dense Convolutions vs. Dilated Convolutions

As another alternative to implement large convolutions, dilated convolution [3, 17] is a common component to increase the receptive field (RF). However, Table 12 shows though a depth-wise dilated convolution may have the same maximum RF as a depth-wise dense convolution, its representational capacity is much lower, which is expected because it is mathematically equivalent to a sparse large convolution. Literature (e.g., [14, 16]) further suggests that dilated convolutions may suffer from gridding problem. We reckon the drawbacks of dilated convolutions could be overcome by mixture of convolutions with different dilations, which will be investigated in the future.

Appendix F: Visualizing the Kernel Weights with Small-Kernel Re-parameterization

We visualize the weights of the re-parameterized 13×13 kernels. Specifically, we investigate into the MobileNet V2 models both with and without 3×3 re-parameterization. As shown in Sec. 3 (Guideline 3), the ImageNet scores are 73.24% and 72.53%, respectively. We use the first stride-1 13×13 conv in the last stage (i.e., the stage with input resolution of 7×7) as the representative, and aggregate (take the absolute value and sum up across channels) the resultant kernel into a 13×13 matrix, and respectively rescale to [0, 1] for the comparability. For the model with 3×3 re-param, we show both the original 13×13 kernel (only after BN fusion) and the result after re-param (i.e., adding the 3×3 kernel onto the central part of 13×13). For the model without re-param, we also fuse the BN for the fair comparison.

We observe that every aggregated kernel shows a similar pattern: the central point has the largest magnitude; generally, points closer to the center have larger values; and the “skeleton” parameters (the 13×1 and 1×13 criss-cross parts) are relatively larger, which is consistent with the discovery reported by ACNet [5]. But the kernel with 3×3 re-param differs in that the central 3×3 part of the resultant kernel is further enhanced, which is found to improve the performance.

References


