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

Xiaohan Ding [1*]   Xiangyu Zhang [2†]   Yizhuang Zhou [2]
Jungong Han [3]   Guiguang Ding [1]   Jian Sun [2]

[1] Beijing National Research Center for Information Science and Technology (BNRist);
School of Software, Tsinghua University, Beijing, China
[2] MEGVII Technology
[3] Computer Science Department, Aberystwyth University, SY23 3FL, UK

dxh17@mails.tsinghua.edu.cn   zhangxiangyu@megvii.com   zhouyizhuang@megvii.com
jungonghan77@gmail.com   dinggg@tsinghua.edu.cn   sunjian@megvii.com

Abstract

*We revisit large kernel design in modern convolutional neural networks (CNNs). Inspired by recent advances in vision transformers (ViTs), in this paper, we demonstrate that using a few large convolutional kernels instead of a stack of small kernels could be a more powerful paradigm. We suggested five guidelines, e.g., applying re-parameterized large depth-wise convolutions, to design efficient high-performance large-kernel CNNs. Following the guidelines, we propose RepLKNet, a **pure** CNN architecture whose kernel size is as large as 31×31, in contrast to commonly used 3×3. RepLKNet greatly closes the performance gap between CNNs and ViTs, e.g., achieving comparable or superior results than Swin Transformer on ImageNet and a few typical downstream tasks, with lower latency. RepLKNet also shows nice scalability to big data and large models, obtaining **87.8%** top-1 accuracy on ImageNet and **56.0%** mIoU on ADE20K, which is very competitive among the state-of-the-arts with similar model sizes. Our study further reveals that, in contrast to small-kernel CNNs, large-kernel CNNs have much larger effective receptive fields and higher shape bias rather than texture bias. Code & models at https://github.com/megvii-research/RepLKNet.*

1. Introduction

Convolutional neural networks (CNNs) [42, 55] used to be a common choice of visual encoders in modern computer
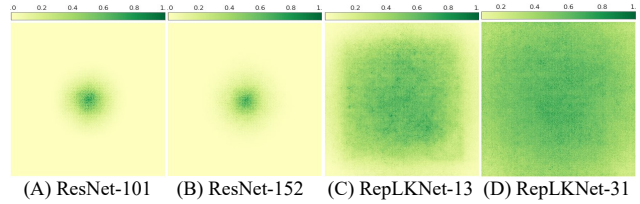

Figure 1. The *Effective Receptive Field (ERF)* of ResNet-101/152 and RepLKNet-13/31 respectively. A more widely distributed dark area indicates a larger ERF. More layers (*e.g.*, from ResNet-101 to ResNet-152) help little in enlarging ERFs. Instead, our large kernel model *RepLKNet* effectively obtains large ERFs.

(A) ResNet-101   (B) ResNet-152   (C) RepLKNet-13   (D) RepLKNet-31

vision systems. However, recently, CNNs [42, 55] have been greatly challenged by *Vision Transformers (ViTs)* [35, 61, 89, 98], which have shown leading performances on many visual tasks – not only image classification [35, 109] and representation learning [4, 10, 17, 105], but also many downstream tasks such as object detection [25, 61], semantic segmentation [98, 103] and image restoration [11, 56]. Why are ViTs super powerful? Some works believed that *multi-head self-attention (MHSA)* mechanism in ViTs plays a key role. They provided empirical results to demonstrate that, MHSA is more flexible [52], capable (less inductive bias) [21], more robust to distortions [68, 103], or able to model long-range dependencies [71, 93]. But some works challenge the necessity of MHSA [121], attributing the high performance of ViTs to the proper building blocks [34], and/or dynamic sparse weights [40, 116]. More works [21, 40, 44, 100, 121] explained the superiority of ViTs from different point of views.

In this work, we focus on one view: *the way of building up large receptive fields*. In ViTs, *MHSA* is usually designed to be either global [35, 80, 98] or local but with large kernels [61, 72, 92], thus each output from a *single* MHSA layer is able to gather information from a large region. However, large kernels are not popularly employed

in CNNs (except for the first layer [42]). Instead, a typical fashion is to use a stack of many small *spatial convolutions*[1] [42, 46, 49, 70, 79, 84, 114] (*e.g.*, 3×3) to enlarge the receptive fields in state-of-the-art CNNs. Only some old-fashioned networks such as *AlexNet* [55], *Inceptions* [81–83] and a few architectures derived from *neural architecture search* [39, 45, 58, 122] adopt large spatial convolutions (whose size is greater than 5) as the main part. The above view naturally lead to a question: what if we use *a few large* instead of *many small* kernels to conventional CNNs? Is large kernel or the way of building large receptive fields the key to close the performance gap between CNNs and ViTs?

To answer this question, we systematically explore the large kernel design of CNNs. We follow a very simple "philosophy": just introducing large *depth-wise* convolutions into conventional networks, whose sizes range from 3×3 to 31×31, although there exist other alternatives to introduce large receptive fields via a single or a few layers, *e.g.* feature pyramids [96], dilated convolutions [14, 106, 107] and deformable convolutions [24]. Through a series of experiments, we summarize five empirical guidelines to effectively employ large convolutions: **1)** very large kernels can still be efficient in practice; **2)** identity shortcut is vital especially for networks with very large kernels; **3)** reparameterizing [31] with small kernels helps to make up the optimization issue; **4)** large convolutions boost downstream tasks much more than *ImageNet*; **5)** large kernel is useful even on small feature maps.

Based on the above guidelines, we propose a new architecture named *RepLKNet*, a *pure*[2] CNN where reparameterized large convolutions are employed to build up large receptive fields. Our network in general follows the macro architecture of *Swin Transformer* [61] with a few modifications, while replacing the *multi-head self-attentions* with large depth-wise convolutions. We mainly benchmark middle-size and large-size models, since ViTs used to be believed to surpass CNNs on large data and models. On ImageNet classification, our baseline (similar model size with *Swin-B*), whose kernel size is as large as 31×31, achieves **84.8%** top-1 accuracy trained only on ImageNet-1K dataset, which is 0.3% better than Swin-B but much more efficient in latency.

More importantly, we find that the large kernel design is particularly powerful on *downstream* tasks. For example, our networks outperform *ResNeXt*-101 [104] or *ResNet*-101 [42] backbones by **4.4%** on *COCO* detection [57] and **6.1%** on *ADE20K* segmentation [120] under the similar complexity and parameter budget, which is also on par with

or even better than the counterpart *Swin Transformers* but with higher inference speed. Given more pretraining data (*e.g.*, 73M images) and more computational budget, our best model obtains very competitive results among the state-of-the-arts with similar model sizes, *e.g.* **87.8%** top-1 accuracy on ImageNet and **56.0%** on ADE20K, which shows excellent scalability towards large-scale applications.

We believe the high performance of RepLKNet is mainly because of the large *effective receptive fields (ERFs)* [65] built via large kernels, as compared in Fig. 1. Moreover, RepLKNet is shown to leverage more shape information than conventional CNNs, which partially agrees with human's cognition. We hope our findings can help to understand the intrinsic mechanism of both CNNs and ViTs.

2. Related Work

2.1. Models with Large Kernels

As mentioned in the introduction, apart from a few old-fashioned models like *Inceptions* [81–83], large-kernel models became not popular after *VGG-Net* [79]. One representative work is *Global Convolution Networks (GCNs)* [69], which uses very large convolutions of 1×K followed by K×1 to improve semantic segmentation task. However, large kernels are reported to harm the performance on ImageNet. *Local Relation Networks (LR-Net)* [47] proposes a spatial aggregation operator (LR-Layer) to replace standard convolutions, which can be viewed as a dynamic convolution. LR-Net could benefit from a kernel size of 7×7, but the performance decreases with 9×9. With a kernel size as large as the feature map, the top-1 accuracy significantly reduced from 75.7% to 68.4%.

Recently, *Swin Transformers* [61] propose to capture the spatial patterns with shifted window attention, whose window sizes range from 7 to 12, which can also be viewed as a variant of large kernel. The follow-ups [33, 60] employ even larger window sizes. Inspired by the success of those local transformers, a recent work [40] replaces *MHSA* layers with static or dynamic 7×7 depth-wise convolutions in [61] while still maintains comparable results. Though the network proposed by [40] shares similar design pattern with ours, the motivations are different: [40] does not investigate the relationship between *ERFs*, large kernels and performances; instead, it attributes the superior performances of vision transformers to sparse connections, shared parameters and dynamic mechanisms. Another three representative works are *Global Filter Networks (GFNets)* [74], *CK-Conv* [76] and *FlexConv* [75]. GFNet optimizes the spatial connection weights in the Fourier domain, which is equivalent to circular global convolutions in the spatial domain. CKConv formulates kernels as continuous functions to process sequential data, which can construct arbitrarily large kernels. FlexConv learns different kernel sizes for different

---

[1]Convolutional kernels (including the variants such as depthwise/group convolutions) whose spatial size is larger than 1×1.

[2]Namely CNNs free of any attention or dynamic mechanism, *e.g.*, *squeeze-and-excitation* [48], *multi-head self-attention*, *dynamic weights* [40, 100], and *etc*.

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

layers, which can be as large as the feature maps. Although they use very large kernels, they do not intend to answer the key questions we desire: why do traditional CNNs underperform ViTs, and how to apply large kernels in *common CNNs*. Besides, both [40] and [74] do not evaluate their models on strong baselines, *e.g.*, models larger than *Swin-L*. Hence it is still unclear whether large-kernel CNNs can scale up well as transformers.

Concurrent works. *ConvMixer* [90] uses up to 9×9 convolutions to replace the "mixer" component of *ViTs* [35] or *MLPs* [87, 88]. *MetaFormer* [108] suggests *pooling* layer is an alternate to self-attention. *ConvNeXt* [62] employs 7×7 depth-wise convolutions to design strong architectures, pushing the limit of CNN performances. Although those works show excellent performances, they do not show benefits from much larger convolutions (*e.g.*, 31×31).

2.2. Model Scaling Techniques

Given a small model, it is a common practice to scale it up for better performance, thus scaling strategy plays a vital role in the resultant accuracy-efficiency trade-offs. For CNNs, existing scaling approaches usually focus on model depth, width, input resolution [32, 70, 84], bottleneck ratio and group width [32, 70]. Kernel size, however, is often neglected. In Sec. 3, we will show that the kernel size is also an important scaling dimension in CNNs, especially for *downstream* tasks.

2.3. Structural Re-parameterization

Structural Re-parameterization [27–31] is a methodology of equivalently converting model structures via transforming the parameters. For example, RepVGG targeted at a deep inference-time VGG-like (*e.g.*, branch-free) model, and constructed extra ResNet-style shortcuts parallel to the 3×3 layers during training. In contrast to a real VGG-like model that is difficult to train [42], such shortcuts helped the model reach a satisfactory performance. After training, the shortcuts are absorbed into the parallel 3×3 kernels via a series of linear transformations, so that the resultant model becomes a VGG-like model. In this paper, we use this methodology to add a relatively small (*e.g.*, 3×3 or 5×5) kernel into a very large kernel. In this way, we make the

very large kernel capable of capturing small-scale patterns, hence improve the performance of the model.

3. Guidelines of Applying Large Convolutions

Trivially applying large convolutions to CNNs usually leads to inferior performance and speed. In this section, we summarize 5 guidelines for effectively using large kernels.

Guideline 1: large depth-wise convolutions can be efficient in practice. It is believed that large-kernel convolutions are computationally expensive because the kernel size quadratically increases the number of parameters and FLOPs. The drawback can be greatly overcome by applying depth-wise (DW) convolutions [18, 46]. For example, in our proposed *RepLKNet* (see Table 5 for details), increasing the kernel sizes in different stages from [3, 3, 3, 3] to [31, 29, 27, 13] only increases the FLOPs and number of parameters by 18.6% and 10.4% respectively, which is acceptable. The remaining 1×1 convolutions actually dominate most of the complexity.

One may concern that DW convolutions could be very inefficient on modern parallel computing devices like GPUs. It is true for conventional DW 3×3 kernels [46, 77, 114], because DW operations introduce low ratio of computation *vs.* memory access cost [66], which is not friendly to modern computing architecture. However, we find when kernel size becomes large, the computational density increases: for example, in a DW 11×11 kernel, each time we load a value from the feature map, it can attend at most 121 multiplications, while in a 3×3 kernel the number is only 9. Therefore, according to the roofline model, the actual latency should not increase as much as the increasing of FLOPs when kernel size becomes larger.

Remark 1. Unfortunately, we find off-the-shelf deep learning tools (such as Pytorch) support large DW convolutions poorly, as shown in Table 1. Hence we try several approaches to optimize the CUDA kernels. FFT-based approach [67] appears reasonable to implement large convolutions. However, in practice we find block-wise (inverse) *implicit gemm* algorithm is a better choice. The implementation has been integrated into the open-sourced framework MegEngine [1] and we omit the details here. We have also released an efficient implementation [2] for PyTorch. Ta-

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

Table 3. Results of 3×3 re-parameterization on MobileNet V2 with various kernel sizes.

Kernel	3×3 re-param	ImageNet top-1 acc (%)	Cityscapes val mIoU (%)
3×3	N/A	71.76	72.31
9×9		72.67	76.11
9×9	✓	**73.09**	**76.30**
13×13		72.53	75.67
13×13	✓	**73.24**	**76.60**

ble 1 shows that our implementation is far more efficient, compared with the Pytorch baseline. With our optimization, the latency contribution of DW convolutions in RepLKNet reduces from 49.5% to 12.3%, which is roughly in proportion to the FLOPs occupation.

Guideline 2: identity shortcut is vital especially for networks with very large kernels. To demonstrate this, we use *MobileNet V2* [77] to benchmark, since it heavily uses DW layers and has two published variants (with or without shortcuts). For the large-kernel counterparts, we simply replace all the DW 3×3 layers with 13×13. All the models are trained on ImageNet with the identical training configurations for 100 epochs (see Appendix A for details). Table 2 shows large kernels improve the accuracy of MobileNet V2 with shortcuts by 0.77%. However, without shortcuts, large kernels reduce the accuracy to only 53.98%.

Remark 2. The guideline also works for *ViTs*. A recent work [34] finds that without identity shortcut, attention loses rank doubly exponentially with depth, leading to over-smoothing issue. Although large-kernel CNNs may degenerate in a different mechanism from ViT's, we also observed without shortcut, it is difficult for the network to capture local details. From a similar perspective as [94], shortcuts make the model an implicit ensemble composed of numerous models with different receptive fields (RFs), so it can benefit from a much larger maximum RF while not losing the ability to capture small-scale patterns.

Guideline 3: re-parameterizing [31] with small kernels helps to make up the optimization issue. We replace the 3×3 layers of *MobileNet V2* by 9×9 and 13×13 respectively, and optionally adopt *Structural Re-parameterization* [27, 28, 31] methodology. Specifically, we construct a 3×3 layer parallel to the large one, then add up their outputs after *Batch normalization (BN)* [51] layers (Fig. 2). After training, we merge the small kernel as well as BN parameters into the large kernel, so the resultant model is equivalent to the model for training but no longer has small kernels. Table 3 shows directly increasing the kernel size from 9 to 13 reduces the accuracy, while re-parameterization addresses the issue.

We then transfer the ImageNet-trained models to semantic segmentation with DeepLabv3+ [16] on Cityscapes [22]. We only replace the backbone and keep all the default training settings provided by MMSegmentation [20]. The observation is similar to that on ImageNet: 3×3 re-param im-

proves the mIoU of the 9×9 model by 0.19 and the 13×13 model by 0.93. With such simple re-parameterization, increasing kernel size from 9 to 13 no longer degrades the performance on both ImageNet and Cityscapes.

Remark 3. It is known that *ViTs* have optimization problem especially on small datasets [35, 59]. A common workaround is to introduce *convolutional prior*, *e.g.*, add a DW 3×3 convolution to each self-attention block [19, 101], which is analogous to ours. Those strategies introduce additional *translational equivariance* and *locality* prior to the network, making it easier to optimize on small dataset without loss of generality. Similar to what ViT behaves [35], we also find when the pretraining dataset increases to 73 million images (refer to *RepLKNet-XL* in the next section), re-parameterization can be omitted without degradation.

Guideline 4: large convolutions boost downstream tasks much more than ImageNet classification. Table 3 (after *re-param*) shows increasing the kernel size of MobileNet V2 from 3×3 to 9×9 improves the *ImageNet* accuracy by 1.33% but the *Cityscapes* mIoU by 3.99%. Table 5 shows a similar trend: as the kernel sizes increase from [3, 3, 3, 3] to [31, 29, 27, 13], the ImageNet accuracy improves by only 0.96%, while the mIoU on *ADE20K* [120] improves by 3.12%. Such phenomenon indicates that models of similar ImageNet scores could have very different capability in downstream tasks (just as the bottom 3 models in Table 5).

Remark 4. What causes the phenomenon? First, large kernel design significantly increases the *Effective Receptive Fields (ERFs)* [65]. Numerous works have demonstrated "contextual" information, which implies large ERFs, is crucial in many downstream tasks like object detection and semantic segmentation [63, 69, 96, 106, 107]. We will discuss the topic in Sec. 5. Second, We deem another reason might be that large kernel design contributes more shape biases to the network. Briefly speaking, ImageNet pictures can be correctly classified according to either texture or shape, as proposed in [8, 36]. However, humans recognize objects mainly based on shape cue rather than texture, therefore a model with stronger shape bias may transfer better to downstream tasks. A recent study [91] points out ViTs are strong in shape bias, which partially explains why ViTs are super powerful in transfer tasks. In contrast, conventional CNNs trained on ImageNet tend to bias towards tex-
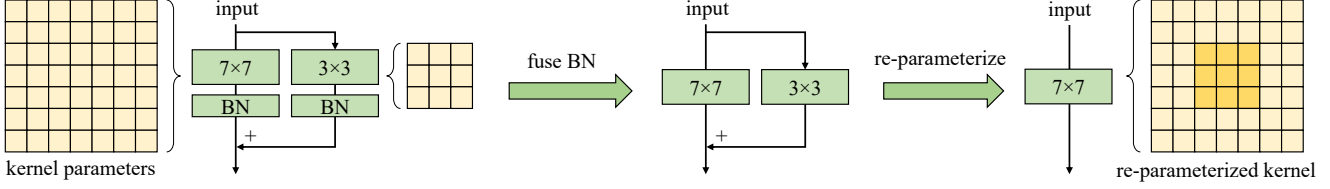
Figure 2. An example of re-parameterizing a small kernel (*e.g.*, 3×3) into a large one (*e.g.*, 7×7). See [28, 31] for details.
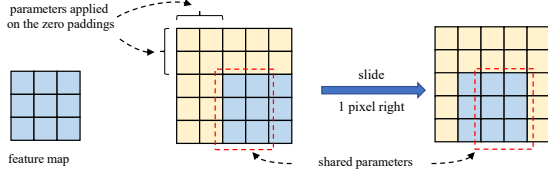

Figure 3. Illustration to convolution with small feature map and large kernel. Two outputs at adjacent locations only share a part of kernel weights. Translational equivariance does not strictly hold.

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**

ture [8, 36]. Fortunately, we find simply enlarging the kernel size in CNNs can effectively improve the shape bias. Please refer to Appendix C for details.

Guideline 5: large kernel (*e.g.*, 13×13) is useful even on small feature maps (*e.g.*, 7×7). To validate it, We enlarge the DW convolutions in the *last stage* of *MobileNet V2* to 7×7 or 13×13, hence the kernel size is on par with or even larger than feature map size (7×7 by default). We apply re-parameterization to the large kernels as suggested by Guideline 3. Table 4 shows although convolutions in the last stage already involve very large receptive field, further increasing the kernel sizes still leads to performance improvements, especially on downstream tasks such as *Cityscapes*.

Remark 5. When kernel size becomes large, notice that translational equivariance of CNNs does not strictly hold. As illustrated in Fig. 3, two outputs at adjacent spatial locations share only a fraction of the kernel weights, *i.e.*, are transformed by different mappings. The property also agrees with the "philosophy" of *ViTs* – relaxing the symmetric prior to obtain more capacity. Interestingly, we find 2D *Relative Position Embedding (RPE)* [5, 78], which is widely used in the transformer community, can also be viewed as a large depth-wise kernel of size $(2H-1) \times (2W-1)$, where H and W are feature map height and width respectively. Large kernels not only help to learn the relative positions between concepts, but also encode the *absolute position* information due to *padding effect* [53].

4. RepLKNet: a Large-Kernel Architecture

Following the above guidelines, in this section we propose RepLKNet, a *pure* CNN architecture with large kernel design. To our knowledge, up to now CNNs still dominate small models [113, 115], while *vision transformers* are believed to be better than CNNs under more complexity budget. Therefore, in the paper we mainly focus on relatively large models (whose complexity is on par with or larger than *ResNet-152* [42] or *Swin-B* [61]), in order to verify whether large kernel design could eliminate the performance gap between CNNs and ViTs.

4.1. Architecture Specification

We sketch the architecture of RepLKNet in Fig. 4:

Stem refers to the beginning layers. Since we target at high performance on downstream dense-prediction tasks, we desire to capture more details by several conv layers at the beginning. After the first 3×3 with 2× downsampling, we arrange a DW 3×3 layer to capture low-level patterns, a 1×1 conv, and another DW 3×3 layer for downsampling.

Stages 1-4 each contains several RepLK Blocks, which use shortcuts (Guideline 2) and DW large kernels (Guideline 1). We use 1×1 conv before and after DW conv as a common practice. Note that each DW large conv uses a 5×5 kernel for re-parameterization (Guideline 3), which is not shown in Fig. 4. Except for the large conv layers which provide sufficient receptive field and the ability to aggregate spatial information, the model's representational capacity is also closely related to the depth. To provide more nonlinearities and information communications across channels, we desire to use 1×1 layers to increase the depth. Inspired by the Feed-Forward Network (FFN) which has been widely used in transformers [35, 61] and MLPs [27, 87, 88], we use a similar CNN-style block composed of shortcut, BN, two 1×1 layers and GELU [43], so it is referred to as ConvFFN Block. Compared to the classic FFN which uses Layer Normalization [3] before the fully-connected layers, BN has an advantage that it can be fused into conv for efficient inference. As a common practice, the number of internal channels of the ConvFFN Block is 4× as the input. Simply following ViT and Swin, which interleave attention and FFN blocks, we place a ConvFFN after each RepLK Block.

Transition Blocks are placed between stages, which first increase the channel dimension via 1×1 conv and then con-
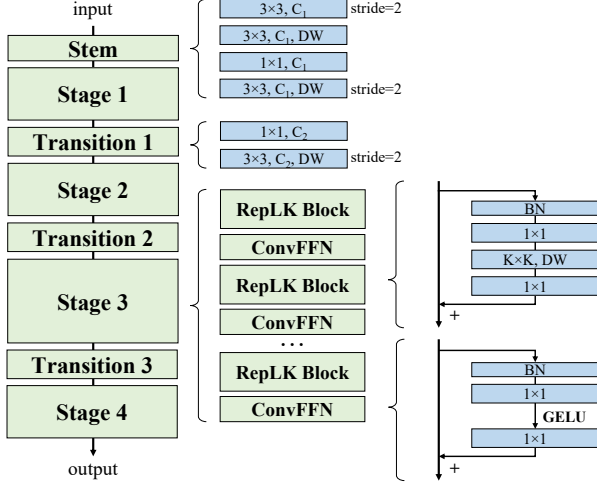
Figure 4. RepLKNet comprises Stem, Stages and Transitions. Except for depth-wise (DW) large kernel, the other components include DW 3×3, dense 1×1 conv, and batch normalization [51] (BN). Note that every conv layer has a following BN, which are not depicted. Such conv-BN sequences use ReLU as the activation function, except those before the shortcut-addition (as a common practice [42,77]) and those preceding GELU [43].

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.

Kernel size	ImageNet			ADE20K		
	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

duct 2× downsampling with DW 3×3 conv.

In summary, each stage has three architectural hyper-parameters: the number of RepLK Blocks B, the channel dimension C, and the kernel size K. So that a RepLKNet architecture is defined by $[B_1, B_2, B_3, B_4], [C_1, C_2, C_3, C_4], [K_1, K_2, K_3, K_4]$.

4.2. Making Large Kernels Even Larger

We continue to evaluate large kernels on RepLKNet via fixing $\mathbf{B}=[2, 2, 18, 2]$, $\mathbf{C}=[128, 256, 512, 1024]$, varying $\mathbf{K}$ and observing the performance of both classification and semantic segmentation. Without careful tuning of the hyper-parameters, we casually set the kernel sizes as $[13, 13, 13, 13]$, $[25, 25, 25, 13]$, $[31, 29, 27, 13]$, respectively, and refer to the models as RepLKNet-13/25/31. We also construct two small-kernel baselines where the kernel sizes are all 3 or 7 (RepLKNet-3/7).

On ImageNet, we train for 120 epochs with AdamW [64] optimizer, RandAugment [23], mixup [111], CutMix [110],

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

Rand Erasing [118] and Stochastic Depth [50], following the recent works [4, 61, 62, 89]. The detailed training configurations are presented in Appendix A.

For semantic segmentation, we use ADE20K [120], which is a widely-used large-scale semantic segmentation dataset containing 20K images of 150 categories for training and 2K for validation. We use the ImageNet-trained models as backbones and adopt UperNet [102] implemented by MMSegmentation [20] with the 80K-iteration training setting and test the *single-scale* mIoU.

Table 5 shows our results with different kernel sizes. On ImageNet, though increasing the kernel sizes from 3 to 13 improves the accuracy, making them even larger brings no further improvements. However, on ADE20K, scaling up the kernels from $[13, 13, 13, 13]$ to $[31, 29, 27, 13]$ brings 0.82 higher mIoU with only 5.3% more parameters and 3.5% higher FLOPs, which highlights the significance of large kernels for downstream tasks.

In the following subsections, we use RepLKNet-31 with stronger training configurations to compare with the state-of-the-arts on ImageNet classification, Cityscapes/ADE20K semantic segmentation and COCO [57] object detection. We refer to the aforementioned model as RepLKNet-31B (B for Base) and a wider model with $\mathbf{C} = [192, 384, 768, 1536]$ as RepLKNet-31L (Large). We construct another RepLKNet-XL with $\mathbf{C} = [256, 512, 1024, 2048]$ and 1.5× inverted bottleneck design in the RepLK Blocks (*i.e.*, the channels of the DW large conv layers are 1.5× as the inputs).

4.3. ImageNet Classification

Since the overall architecture of RepLKNet is akin to Swin, we desire to make a comparison at first. For RepLKNet-31B on ImageNet-1K, we extend the aforementioned training schedule to 300 epochs for a fair comparison. Then we finetune for 30 epochs with input resolution of 384×384, so that the total training cost is much lower

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 [38]. ‡ indicates ImageNet-22K pretraining.

Backbone	Method	mIoU (ss)	mIoU (ms)	Param (M)	FLOPs (G)
RepLKNet-31B	UperNet [102]	**83.1**	**83.5**	110	2315
ResNeSt-200 [112]	DeepLabv3 [15]	-	82.7	-	-
Axial-Res-XL	Axial-DL [95]	80.6	81.1	173	2446
Swin-B	UperNet	80.4	81.5	121	2613
Swin-B	UperNet + [38]	80.8	81.8	121	-
ViT-L ‡	SETR-PUP [117]	79.3	82.1	318	-
ViT-L ‡	SETR-MLA	77.2	-	310	-
Swin-L ‡	UperNet	82.3	83.1	234	3771
Swin-L ‡	UperNet + [38]	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 [102]	43.8	44.9	86	1029
ResNeSt-200 [112]	DeepLabv3 [15]	-	48.4	113	1752
Swin-B	UperNet	48.1	49.7	121	1188
Swin-B	UperNet + [38]	48.4	50.1	121	-
ViT-Hybrid	DPT-Hybrid [73]	-	49.0	90	-
ViT-L	DPT-Large	-	47.6	307	-
ViT-B	SETR-PUP [117]	46.3	47.3	97	-
ViT-B	SETR-MLA [117]	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

than the Swin-B model, which was trained with 384×384 from scratch. Then we pretrain RepLKNet-B/L models on ImageNet-22K and finetune on ImageNet-1K. RepLKNet-XL is pretrained on our private semi-supervised dataset named *MegData73M*, which is introduced in the Appendix. We also present the throughput tested with a batch size of 64 on the same 2080Ti GPU. The training configurations are presented in the Appendix.

Table 6 shows that though very large kernels are not intended for ImageNet classification, our RepLKNet models show a a favorable trade-off between accuracy and efficiency. Notably, with only ImageNet-1K training, RepLKNet-31B reaches 84.8% accuracy, which is 0.3% higher than Swin-B, and runs 43% faster. And even though RepLKNet-XL has higher FLOPs than Swin-L, it runs

4.4. Semantic Segmentation

We then use the pretrained models as the backbones on Cityscapes (Table 7) and ADE20K (Table 8). Specifically, we use the UperNet [102] implemented by MMSegmentation [20] with the 80K-iteration training schedule for Cityscapes and 160K for ADE20K. Since we desire to evaluate the backbone only, we do not use any advanced techniques, tricks, nor custom algorithms.

On Cityscapes, ImageNet-1K-pretrained RepLKNet-31B outperforms Swin-B by a significant margin (single-scale mIoU of 2.7), and even outperforms the ImageNet-22K-pretrained Swin-L. Even equipped with DiverseP-atch [38], a technique customized for vision transformers, the single-scale mIoU of the 22K-pretrained Swin-L is still lower than our 1K-pretrained RepLKNet-31B, though the former has 2× parameters.

On ADE20K, RepLKNet-31B outperforms Swin-B with both 1K and 22K pretraining, and the margins of single-scale mIoU are particularly significant. Pretrained with our semi-supervised dataset *MegData73M*, RepLKNet-XL achieves an mIoU of 56.0, which shows feasible scalability towards large-scale vision applications.

4.5. Object Detection

For object detection, we use RepLKNets as the backbone of FCOS [86] and Cascade Mask R-CNN [9,41], which are representatives of one-stage and two-stage detection methods, and the default configurations in MMDetection [13]. The FCOS model is trained with the 2x (24-epoch) training schedule for a fair comparison with the X101 (short for ResNeXt-101 [104]) baseline from the same code base [20], and the other results with Cascade Mask R-CNN all use 3x (36-epoch). Again, we simply replace the backbone and do not use any advanced techniques. Table 9 shows RepLKNets outperform ResNeXt-101-64x4d by up to 4.4 mAP while have fewer parameters and lower FLOPs. Note that the results may be further improved with the advanced techniques like HTC [12], HTC++ [61], Soft-NMS [7] or a 6x (72-epoch) schedule. Compared to Swin, RepLKNets achieve higher or comparable mAP with fewer parameters and lower FLOPs. Notably, RepLKNet-XL achieves an mAP of 55.5, which demonstrates the scalability again.

5. Discussions

5.1. Large-Kernel CNNs have Larger ERF than Deep Small-Kernel Models

We have demonstrated large kernel design can significantly boost *CNNs* (especially on *downstream* tasks). However, it is worth noting that large kernel can be expressed

Table 9. Object detection on COCO. The FLOPs is computed with 1280×800 inputs. The results of ResNeXt-101-64x4d + Cas Mask are reported by [61]. The results of 22K-pretrained Swin (without HTC++ [61]) are reported by [62]. ‡ indicates ImageNet-22K pre-training. ⋄ indicates pretrained with extra data.

Backbone	Method	AP^{box}	AP^{mask}	Param (M)	FLOPs (G)
RepLKNet-31B	FCOS	**47.0**	-	87	437
X101-64x4d	FCOS	42.6	-	90	439
RepLKNet-31B	Cas Mask	**52.2**	**45.2**	137	965
X101-64x4d	Cas Mask	48.3	41.7	140	972
ResNeSt-200	Cas R-CNN [9]	49.0	-	-	-
Swin-B	Cas Mask	51.9	45.0	145	982
RepLKNet-31B ‡	Cas Mask	**53.0**	**46.0**	137	965
Swin-B ‡	Cas Mask	**53.0**	45.8	145	982
RepLKNet-31L ‡	Cas Mask	**53.9**	46.5	229	1321
Swin-L ‡	Cas Mask	**53.9**	**46.7**	254	1382
RepLKNet-XL ⋄	Cas Mask	**55.5**	**48.0**	392	1958

by a series of small convolutions [79], *e.g.*, a 7×7 convolution can be decomposed into a stack of three 3×3 kernels without information loss (more channels are required after the decomposition to maintain the degree of freedom). Given that fact, a question naturally comes up: why do conventional CNNs, which may contain tens or hundreds of small convolutions (*e.g.*, *ResNets* [42]), still behave inferior to large-kernel networks?

We argue that in terms of obtaining large receptive field, a single large kernel is much more *effective* than many small kernels. **First**, according to the theory of *Effective Receptive Field (ERF)* [65], ERF is proportion to $\mathcal{O}(K\sqrt{L})$, where K is the kernel size and L is the depth, *i.e.*, number of layers. In other words, ERF grows linearly with the kernel size while sub-linearly with the depth. **Second**, the increasing depth introduces optimization difficulty [42]. Although ResNets seem to overcome the dilemma, managing to train a network with hundreds of layers, some works [26, 94] indicate ResNets might not be as deep as they appear to be. For example, [94] suggests ResNets behave like ensembles of shallow networks, which implies the ERFs of ResNets could still very limited even if the depth dramatically increases. Such phenomenon is also empirically observed in previous works [54]. To summarize, large kernels design requires fewer layers to obtain large ERFs and avoids the optimization issue brought by the increasing depth.

To support our viewpoint, we choose ResNet-101/152 and the aforementioned RepLKNet-13/31 as the representatives of small-kernel and large-kernel models, which are all well-trained on ImageNet, and test with 50 images from the ImageNet validation set resized to 1024×1024. To visualize the ERF, we use a simple yet effective method (code released at [2]) as introduced in Appendix B, following [54]. Briefly, we produce an *aggregated contribution score matrix* A (1024×1024), where each entry a ($0 \leq a \leq 1$) measures the contribution of the corresponding pixel on the in-

Table 10. Quantitative analysis on the ERF with the high-contribution area ratio r. A larger r suggests a smoother distribution of high-contribution pixels, hence larger ERF.

	$t = 20\%$	$t = 30\%$	$t = 50\%$	$t = 99\%$
ResNet-101	0.9%	1.5%	3.2%	22.4%
ResNet-152	1.1%	1.8%	3.9%	34.4%
RepLKNet-13	11.2%	17.1%	30.2%	96.3%
RepLKNet-31	16.3%	24.7%	43.2%	98.6%

put image to the central point of the feature map produced by the last layer. Fig. 1 shows the high-contribution pixels of ResNet-101 gather around the central point, but the outer points have very low contributions, indicating a limited ERF. ResNet-152 shows a similar pattern, suggesting the more 3×3 layers do not significantly increase the ERF. On the other hand, the high-contribution pixels in Fig. 1 (C) are more evenly distributed, suggesting RepLKNet-13 attends to more outer pixels. With larger kernels, RepLKNet-31 makes the high-contribution pixels spread more uniformly, indicating an even larger ERF.

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 RepLKNets. 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.

5.2. Large-Kernel Models are More Similar to Human in Shape Bias

We have found out that RepLKNet-31B has much higher shape bias than Swin Transformer and small-kernel CNNs. A recent work [91] 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 [6] 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 with 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
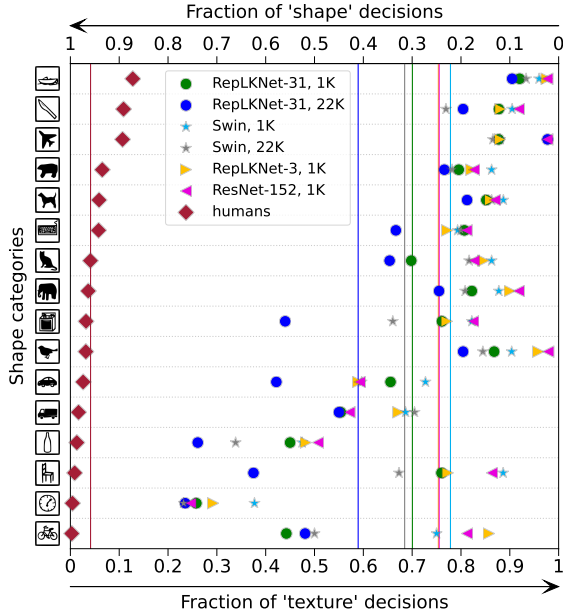
Figure 5. Shape bias of RepLKNet, Swin, and ResNet-152 pretrained on ImageNet-1K or 22K. The scatters represent the shape bias of 16 categories, and the vertical lines are the averages across categories (note RepLKNet-3 and ResNet-152 are very close).

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* [35] reported by [91] (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).

5.3. Large Kernel Design is a Generic Design Element that Works with ConvNeXt

Replacing the 7×7 convolutions in ConvNeXt [62] by kernels as large as 31×31 brings significant improvements, *e.g.*, ConNeXt-Tiny + large kernel >ConNeXt-Small , and ConNeXt-Small + large kernel >ConNeXt-Base.

We use the recently proposed ConvNeXt [62] as the benchmark architecture to evaluate large kernel as a *generic design element*. We simply replace the 7×7 convolutions in ConvNeXt [62] 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

that kernel size is an important scaling dimension.

5.4. Large Kernels Outperform Small Kernels with High Dilation Rates

Please refer to Appendix C for details.

6. Limitations

Although large kernel design greatly improves CNNs on both ImageNet and downstream tasks, however, according to Table 6, as the scale of data and model increases, *RepLKNets* start to fall behind *Swin Transformers*, *e.g.*, the ImageNet top-1 accuracy of RepLKNet-31L is 0.7% lower than Swin-L with ImageNet-22K pretraining (while the downstream scores are still comparable). It is not clear whether the gap is resulted from suboptimal hyperparameter tuning or some other fundamental drawback of CNNs which emerges when data/model scales up. We are working in progress on the problem.

7. Conclusion

This paper revisits large convolutional kernels, which have long been neglected in designing *CNN* architectures. We demonstrate that using a few large kernels instead of many small kernels results in larger *effective receptive field* more efficiently, boosting CNN's performances especially on downstream tasks by a large margin, and greatly closing the performance gap between CNNs and ViTs when data and models scale up. We hope our work could advance both studies of CNNs and ViTs. On one hand, for CNN community, our findings suggest that we should pay special attention to *ERFs*, which may be the key to high performances. On the other hand, for ViT community, since large convolutions act as an alternative to *multi-head self-attentions* with similar behaviors, it may help to understand the intrinsic mechanism of self-attentions.

References

[1] Megengine:a fast, scalable and easy-to-use deep learning framework. https://github.com/MegEngine/MegEngine, 2020. 3

[2] Official pytorch implementation of replknet. https://github.com/DingXiaoH/RepLKNet-pytorch, 2022. 3, 8

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[4] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 1, 6, 15

[5] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019. 5

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.

		ImageNet			ADE20K		
Kernel size	Architecture	Top-1	Params	FLOPs	mIoU	Params	FLOPs
7-7-7-7	ConvNeXt-Tiny	81.0	29M	4.5G	44.6	60M	939G
7-7-7-7	ConvNeXt-Small	82.1	50M	8.7G	45.9	82M	1027G
7-7-7-7	ConvNeXt-Base	82.8	89M	15.4G	47.2	122M	1170G
31-29-27-13	ConvNeXt-Tiny	81.6	32M	6.1G	**46.2**	64M	973G
31-29-27-13	ConvNeXt-Small	82.5	58M	11.3G	**48.2**	90M	1081G

[6] bethgelab. Toolbox of model-vs-human. https://github.com/bethgelab/model-vs-human, 2022. 8

[7] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017. 7

[8] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019. 4, 5

[9] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 7, 8

[10] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 1

[11] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021. 1

[12] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019. 7

[13] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 2, 16

[15] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 7

[16] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 4

[17] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised visual transformers. *arXiv e-prints*, pages arXiv–2104, 2021. 1

[18] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 3

[19] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021. 4

[20] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 4, 6, 7

[21] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019. 1

[22] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3213–3223. IEEE Computer Society, 2016. 4

[23] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 6, 15

[24] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2

[25] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7373–7382, 2021. 1

[26] Soham De and Samuel L Smith. Batch normalization biases residual blocks towards the identity function in deep networks. *arXiv preprint arXiv:2002.10444*, 2020. 8

[27] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Repmlpnet: Hierarchical vision mlp with re-parameterized locality. *arXiv preprint arXiv:2112.11081*, 2021. 3, 4, 5

[28] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1911–1920, 2019. 3, 4, 5, 16

[29] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520, 2021. 3

[30] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10886–10895, 2021. 3

[31] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vggstyle convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021. 2, 3, 4, 5

[32] Piotr Dollár, Mannat Singh, and Ross Girshick. Fast and accurate model scaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 924–932, 2021. 3

[33] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv preprint arXiv:2107.00652*, 2021. 2

[34] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*, 2021. 1, 4

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1, 3, 4, 5, 9, 15

[36] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018. 4, 5

[37] Golnaz Ghiasi, Barret Zoph, Ekin D Cubuk, Quoc V Le, and Tsung-Yi Lin. Multi-task self-training for learning general representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8856–8865, 2021. 15

[38] Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Improve vision transformers training by suppressing over-smoothing. *arXiv preprint arXiv:2104.12753*, 2021. 7

[39] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path oneshot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 2

[40] Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. Demystifying local vision transformer: Sparse connectivity, weight sharing, and dynamic weight. *arXiv preprint arXiv:2106.04263*, 2021. 1, 2, 3

[41] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 7

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 3, 5, 6, 8

[43] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 5, 6

[44] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021. 1

[45] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1314–1324. IEEE, 2019. 2

[46] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3

[47] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019. 2

[48] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[49] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society, 2017. 2

[50] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European*

Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV, volume 9908 of *Lecture Notes in Computer Science*, pages 646–661. Springer, 2016. 6

[51] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 4, 6

[52] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021. 1

[53] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020. 5

[54] Bum Jun Kim, Hyeyeon Choi, Hyeonah Jang, Dong Gu Lee, Wonseok Jeong, and Sang Woo Kim. Dead pixel test using effective receptive field. *arXiv preprint arXiv:2108.13576*, 2021. 8

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 2

[56] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 1

[57] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 6

[58] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2

[59] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020. 4

[60] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021. 2

[61] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1, 2, 5, 6, 7, 8, 15

[62] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022. 3, 6, 8, 9, 15

[63] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Pro-*

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3431–3440, 2015. 4

[64] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6, 15

[65] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard S. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4898–4906, 2016. 2, 4, 8

[66] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 3

[67] Michaël Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 3

[68] Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners. *arXiv preprint arXiv:2105.07581*, 2021. 1

[69] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2017. 2, 4

[70] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 2, 3

[71] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *arXiv preprint arXiv:2108.08810*, 2021. 1

[72] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 1

[73] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021. 7

[74] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *arXiv preprint arXiv:2107.00645*, 2021. 2, 3

[75] David W Romero, Robert-Jan Bruintjes, Jakub M Tomczak, Erik J Bekkers, Mark Hoogendoorn, and Jan C van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *arXiv preprint arXiv:2110.08059*, 2021. 2

[76] David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021. 2

[77] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 3, 4, 6

[78] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 5

[79] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 8

[80] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16519–16529, 2021. 1

[81] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 2

[82] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[83] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2

[84] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2, 3

[85] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 15

[86] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9626–9635. IEEE, 2019. 7

[87] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 3, 5

[88] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image

[89] classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021. 3, 5

[89] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 6, 15

[90] Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022. 3

[91] Shikhar Tuli, Ishita Dasgupta, Erin Grant, and Thomas L Griffiths. Are convolutional neural networks or transformers more like human vision? *arXiv preprint arXiv:2105.07197*, 2021. 4, 8, 9

[92] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12894–12904, 2021. 1

[93] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1

[94] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pages 550–558, 2016. 4, 8

[95] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Standalone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020. 7

[96] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2, 4

[97] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. Ieee, 2018. 16

[98] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1

[99] Ross Wightman. Timm implementation of randaugment. https://github.com/rwightman/pytorch-image-models/blob/master/timm/data/auto_augment.py, 2022. 15

[100] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019. 1, 2

[101] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing

convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021. 4

[102] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018. 6, 7

[103] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv:2105.15203*, 2021. 1, 16

[104] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 2, 7

[105] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *arXiv preprint arXiv:2105.04553*, 2021. 1

[106] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2, 4, 16

[107] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017. 2, 4

[108] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. *arXiv preprint arXiv:2111.11418*, 2021. 3

[109] Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. Volo: Vision outlooker for visual recognition. *arXiv preprint arXiv:2106.13112*, 2021. 1

[110] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 6

[111] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 6, 15

[112] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 7

[113] Mingda Zhang, Chun-Te Chu, Andrey Zhmoginov, Andrew Howard, Brendan Jou, Yukun Zhu, Li Zhang, Rebecca Hwa, and Adriana Kovashka. Basisnet: Two-stage model synthesis for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3081–3090, 2021. 5

[114] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 2, 3

[115] Yikang Zhang, Zhuo Chen, and Zhao Zhong. Collaboration of experts: Achieving 80% top-1 accuracy on imagenet with 100m flops. *arXiv preprint arXiv:2107.03815*, 2021. 5

[116] Yucheng Zhao, Guangting Wang, Chuanxin Tang, Chong Luo, Wenjun Zeng, and Zheng-Jun Zha. A battle of network structures: An empirical study of cnn, transformer, and mlp. *arXiv preprint arXiv:2108.13002*, 2021. 1

[117] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021. 7

[118] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. 6, 15

[119] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 15

[120] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 2, 4, 6

[121] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6688–6697, 2019. 1

[122] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2

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 4×10^{-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 [64] with momentum of 0.9 and weight decay of 0.05. The learning rate setting includes an initial value of 4×10^{-3}, cosine annealing and 10-epoch warm-up. For the data augmentation and regularization, we use RandAugment [23] ("rand-m9-mstd0.5-inc1" as implemented by timm [99]), label smoothing coefficient of 0.1, mixup [111] with $\alpha = 0.8$, CutMix with $\alpha = 1.0$, Rand Erasing [118] with probability of 25% and Stochastic Depth with a drop-path rate of 30%, following the recent works [4, 61, 62, 89]. 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 4×10^{-4}, cosine annealing, 1-epoch warm-up, 30 epochs, model EMA (Exponential Moving Average) with momentum of 10^{-4}, the same RandAugment as above but *no CutMix nor mixup*.

ImageNet-22K Pretraining and 1K Finetuning

For pretraining RepLKNet-31B/L on ImageNet-22K, we use 128 GPUs and a batch size of 32 per GPU to train for 90 epochs with a drop-path rate of 10%. The other configurations are the same as the aforementioned ImageNet-1K pretraining.

Then for finetuning RepLKNet-31B with 224×224, we use 16 GPUs, a batch size of 32 per GPU, drop-path rate of 20%, initial learning rate of 4×10^{-4}, cosine annealing, model EMA with momentum of 10^{-4} to finetune for 30 epochs. Note again that we use 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 [35] 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 pretrain-

ing, we use a private semi-supervised dataset named *Meg-Data73M*, 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 [119]. Unlabeled images are selected from YFCC100M [85]. We design a multi-task label system according to [37], and utilize soft *pseudo* labels which are offline generated by multiple task-specific ViT-Ls wherever human annotations are unavailable. We pretrain our model for up to 15 epochs with similar configurations as ImageNet-1K pretraining. We do *not use Cut-Mix or mixup*, decrease drop-path rate to 20%, and use a lower initial learning rate of 1.5×10^{-3} 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 [35].

We finetune on ImageNet-1K with input resolution of 320×320 for 30 epochs following BeiT [4], 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.*, $M_{:,:,h'/2,w'/2}$, which can be simply implemented via taking the derivatives of $M_{:,:,h'/2,w'/2}$ to I with the auto-grad mechanism. Concretely, we sum up the central points, take the derivatives to the input as the pixel-wise contribution scores and 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

$$P = \max\left(\frac{\partial(\sum_i^n \sum_j^c M_{i,j,h'/2,w'/2})}{\partial I}, 0\right), \quad (1)$$

$$A = \log_{10}\left(\sum_i^n \sum_j^3 P_{i,j,:,:} + 1\right). \quad (2)$$

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

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

| Max RF	Kernel size	Dilation	ImageNet acc	Params	FLOPs
9	9×9	-	72.67	4.0M	319M
9	3×3	4	57.23	3.5M	300M
13	13×13	-	72.53	4.6M	361M
13	3×3	6	51.21	3.5M	300M

Appendix C: Dense Convolutions *vs.* Dilated Convolutions

As another alternative to implement large convolutions, *dilated convolution* [14, 106] 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.*, [97, 103]) 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 D: 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* [28]. 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.
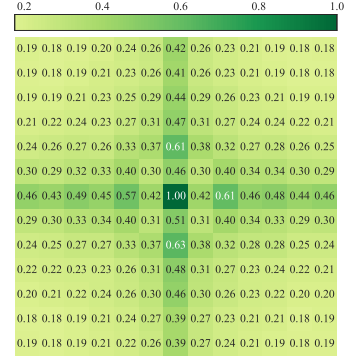
(a) Before re-param.
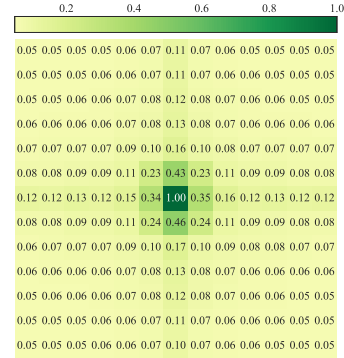
(b) After re-param.
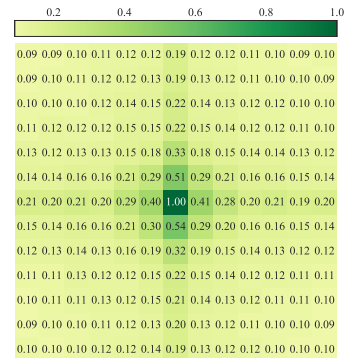
(c) Without re-param.

Figure 6. Parameters of 13×13 kernels in MobileNet V2 aggregated into 13×13 matrices.