Involution: Inverting the Inherence of Convolution for Visual Recognition

Duo Li[1] Jie Hu[2] Changhu Wang[2] Xiangtai Li[3] Qi She[2] Lei Zhu[3] Tong Zhang[1] Qifeng Chen[1]
The Hong Kong University of Science and Technology[1] ByteDance AI Lab[2] Peking University[3]
duo.li@connect.ust.hk {hujie.frank, wangchanghu}@bytedance.com {tongzhang, cqf}@ust.hk

Abstract

Convolution has been the core ingredient of modern neural networks, triggering the surge of deep learning in vision. In this work, we rethink the inherent principles of standard convolution for vision tasks, specifically spatial-agnostic and channel-specific. Instead, we present a novel atomic operation for deep neural networks by inverting the aforementioned design principles of convolution, coined as involution. We additionally demystify the recent popular self-attention operator and subsume it into our involution family as an over-complicated instantiation. The proposed involution operator could be leveraged as fundamental bricks to build the new generation of neural networks for visual recognition, powering different deep learning models on several prevalent benchmarks, including ImageNet classification, COCO detection and segmentation, together with Cityscapes segmentation. Our involution-based models improve the performance of convolutional baselines using ResNet-50 by up to 1.6% top-1 accuracy, 2.5% and 2.4% bounding box AP, and 4.7% mean IoU absolutely while compressing the computational cost to 66%, 65%, 72%, and 57% on the above benchmarks, respectively. Code and pre-trained models for all the tasks are available at https://github.com/d-li14/involution.

1. Introduction

Albeit the rapid advance of neural network architectures, convolution remains the building mainstay of deep neural networks. Drawn inspiration from the classical image filtering methodology, convolution kernels enjoy two remarkable properties that contribute to its magnetism and popularity, namely, spatial-agnostic and channel-specific. In the spatial extent, the former property guarantees the efficiency of convolution kernels by reusing them among different locations and pursues translation equivalence [52]. In the channel domain, a spectrum of convolution kernels is responsible for collecting diverse information encoded in different channels, satisfying the latter property. Furthermore, modern neural networks appreciate the compactness of convolution kernels via restricting their spatial span to no more than 3×3, since the advent of the seminal VGGNet [34].

On the one hand, although the nature of spatial-agnostic along with spatial-compact makes sense in enhancing the efficiency and interpreting the translation equivalence, it deprives convolution kernels of the ability to adapt to diverse visual patterns with respect to different spatial positions. Besides, locality constrains the receptive field of convolution, posing challenges for capturing long-range spatial interactions in a single shot. On the other hand, as is known to us all, inter-channel redundancy inside convolution filters stands out in many successful deep neural networks [19], casting the large flexibility of convolution kernels with respect to different channels into doubt.

To conquer the aforementioned limitations, we present the operation coined as *involution* that has symmetrically *inverse inherent* characteristics compared to convolution, namely, spatial-specific and channel-agnostic. Concretely speaking, involution kernels are distinct in the spatial extent but shared across channels. Being subject to its spatial-specific peculiarity, if involution kernels are parameterized as fixed-sized matrices like convolution kernels and updated using the back-propagation algorithm, the learned involution kernels would be impeded from transferring between input images with variable resolutions. To the end of handling variable feature resolutions, an involution kernel belonging to a specific spatial location is possible to be generated solely conditioned on the incoming feature vector at the corresponding location itself, as an intuitive yet effective instantiation. Besides, we alleviate the redundancy of kernels by sharing the involution kernel along the channel dimension. Taken the above two factors together, the computational complexity of an involution operation scales up linearly with the number of feature channels, based on which an extensive coverage in the spatial dimension is allowed for the dynamically parameterized involution kernels. By virtue of an inverted designing scheme, our proposed involution has two-fold privileges over convolution: (i) involution could summarize the context in a wider spatial arrangement, thus overcome the difficulty of modeling long-range interactions well; (ii) involution could adaptively allocate

the weights over different positions, so as to prioritize the most informative visual elements in the spatial domain.

Analogously, recent approaches have spoken for going beyond convolution with the preference of self-attention for the purpose of capturing long-range dependencies [31, 53]. Among these works, pure self-attention could be utilized to construct stand-alone models with promising performance. Intriguingly, we reveal that self-attention particularizes our generally defined involution through a sophisticated formulation concerning kernel construction. By comparison, the involution kernel adopted in this work is generated conditioned on a single pixel, rather than its relationship with the neighboring pixels. To take one step further, we prove in our experiments that even with our embarrassingly simple version, involution could achieve competitive accuracy-cost trade-offs to self-attention. Being fully aware that the affinity matrix acquired by comparing query with each key in self-attention is also an instantiation of the involution kernel, we question the necessity of composing query and key features to produce such a kernel, since our simplified involution kernel could also attain decent performance while avoiding the superfluous attendance of key content, let alone the dedicated positional encoding in self-attention.

The presented involution operation readily facilitates visual recognition by embedding extendable and switchable spatial modeling into the representation learning paradigm, in a fairly lightweight manner. Built upon this redesigned visual primitive, we establish a backbone architecture family, dubbed as RedNet, which could achieve superior performance over convolution-based ResNet and self-attention based models for image classification. On the downstream tasks including detection and segmentation, we comprehensively perform a step-by-step study to inspect the effectiveness of involution on different components of detectors and segmentors, such as their backbone and neck. Involution is proven to be helpful for each of the considered components, and the combination of them leads to the greatest efficiency.

Summarily, our primary contributions are as follows:

1. We rethink the inherent properties of convolution, associated with the spatial and channel scope. This motivates our advocate of other potential operators embodied with discrimination capability and expressiveness for visual recognition as an alternative, breaking through existing inductive biases of convolution.
2. We bridge the emerging philosophy of incorporating self-attention into the learning procedure of visual representation. In this context, the desiderata of composing pixel pairs for relation modeling is challenged. Furthermore, we unify the view of self-attention and convolution through the lens of our involution.
3. The involution-powered architectures work universally well across a wide array of vision tasks, including image classification, object detection, instance and semantic segmentation, offering significantly better performance than the convolution-based counterparts.

2. Sketch of Convolution

We initiate from introducing the standard convolution operation to make the definition of our proposed involution self-contained. Let $\mathbf{X} \in \mathbb{R}^{H \times W \times C_i}$ denote the input feature map, where H, W represent its height, width and C_i enumerates the input channels. Inside the cube of a feature tensor $\mathbf{X}$, each feature vector $\mathbf{X}_{i,j} \in \mathbb{R}^{C_i}$ located in a cell of the image lattice can be considered as a *pixel* representing certain high-level semantic patterns, with a little abuse of notation.

A cohort of C_o **convolution filters** with the fixed kernel size of $K \times K$ is denoted as $\boldsymbol{\mathcal{F}} \in \mathbb{R}^{C_o \times C_i \times K \times K}$, where each filter $\boldsymbol{\mathcal{F}}_k \in \mathbb{R}^{C_i \times K \times K}, k = 1, 2, \cdots, C_o$, contains C_i **convolution kernels** $\boldsymbol{\mathcal{F}}_{k,c} \in \mathbb{R}^{K \times K}, c = 1, 2, \cdots, C_i$ and executes Multiply-Add operations on the input feature map in a sliding-window manner to yield the output feature map $\mathbf{Y} \in \mathbb{R}^{H \times W \times C_o}$, defined as

$$\mathbf{Y}_{i,j,k} = \sum_{c=1}^{C_i} \sum_{(u,v) \in \boldsymbol{\Delta}_K} \boldsymbol{\mathcal{F}}_{k,c,u+\lfloor K/2 \rfloor, v+\lfloor K/2 \rfloor} \mathbf{X}_{i+u,j+v,c}, \tag{1}$$

where $\boldsymbol{\Delta}_K \in \mathbb{Z}^2$ refers to the set of offsets in the neighborhood considering convolution conducted on the center pixel, written as ($\times$ indicates Cartesian product here)

$$\boldsymbol{\Delta}_K = [-\lfloor K/2 \rfloor, \cdots, \lfloor K/2 \rfloor] \times [-\lfloor K/2 \rfloor, \cdots, \lfloor K/2 \rfloor]. \tag{2}$$

Moreover, depth-wise convolution [6] pushes the formulation of group convolution [23, 46] to the extreme, where each filter (virtually degenerated into a single kernel) $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{K \times K}, k = 1, 2, \cdots, C_o$, strictly performs convolution on an individual feature channel indexed by k, so the first dimension is eliminated from $\boldsymbol{\mathcal{F}}_k$ to form $\boldsymbol{\mathcal{G}}_k$, under the assumption that the number of output channels equals the input ones. As it stands, the convolution operation becomes

$$\mathbf{Y}_{i,j,k} = \sum_{(u,v) \in \boldsymbol{\Delta}_K} \boldsymbol{\mathcal{G}}_{k,u+\lfloor K/2 \rfloor, v+\lfloor K/2 \rfloor} \mathbf{X}_{i+u,j+v,k}. \tag{3}$$

Note that the kernel $\boldsymbol{\mathcal{G}}_k$ is specific to the k^{th} feature slice $\mathbf{X}_{\cdot,\cdot,k}$ from the view of channel and shared among all the spatial locations within this slice.

3. Design of Involution

Compared to either standard or depth-wise convolution described above, **involution kernels** $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{H \times W \times K \times K \times G}$ are devised to embrace transforms with *inverse* characteristics in the spatial and channel domain, hence its name. Specifically, **an involution kernel** $\boldsymbol{\mathcal{H}}_{i,j,\cdot,\cdot,g} \in \mathbb{R}^{K \times K}, g =$

Algorithm 1 Pseudo code of involution in a PyTorch-like style.

```
# B: batch size, H: height, W: width
# C: channel number, G: group number
# K: kernel size, s: stride, r: reduction ratio

################### initialization ###################
o = nn.AvgPool2d(s, s) if s > 1 else nn.Identity()
reduce = nn.Conv2d(C, C//r, 1)
span = nn.Conv2d(C//r, K*K*G, 1)
unfold = nn.Unfold(K, dilation, padding, s)
################### forward pass ###################
x_unfolded = unfold(x) # B,CxKxK,HxW
x_unfolded = x_unfolded.view(B, G, C//G, K*K, H, W)
# kernel generation, Eqn.(6)
kernel = span(reduce(o(x))) # B,KxKxG,H,W
kernel = kernel.view(B, G, K*K, H, W).unsqueeze(2)
# Multiply-Add operation, Eqn.(4)
out = mul(kernel, x_unfolded).sum(dim=3) # B,G,C/G,H,W
out = out.view(B, C, H, W)
return out
```

$1, 2, \cdots, G$, is specially tailored for the pixel $\mathbf{X}_{i,j} \in \mathbb{R}^C$ (the subscript of C is omitted for notation brevity) located at the corresponding coordinate (i, j), but shared over the channels. G counts the number of groups where each group shares the same involution kernel. The output feature map of involution is derived by performing Multiply-Add operations on the input with such involution kernels, defined as

$$\mathbf{Y}_{i,j,k} = \sum_{(u,v)\in\Delta_K} \boldsymbol{\mathcal{H}}_{i,j,u+\lfloor K/2\rfloor,v+\lfloor K/2\rfloor,\lceil kG/C\rceil} \mathbf{X}_{i+u,j+v,k}. \tag{4}$$

Different from convolution kernels, the shape of involution kernels $\boldsymbol{\mathcal{H}}$ depends on that of the input feature map $\mathbf{X}$. A natural thought is to generate the involution kernels conditioned on (part of) the original input tensor, so that the output kernels would be comfortably aligned to the input. We symbolize the kernel generation function as ϕ and abstract the functional mapping at each location (i, j) as

$$\boldsymbol{\mathcal{H}}_{i,j} = \phi(\mathbf{X}_{\boldsymbol{\Psi}_{i,j}}), \tag{5}$$

where $\boldsymbol{\Psi}_{i,j}$ indexes the set of pixels $\boldsymbol{\mathcal{H}}_{i,j}$ is conditioned on.

Implementation Details Respectful of the conciseness of convolution, we make involution conceptually as simple as possible. Note that our target is to firstly provide a design space for the kernel generation function ϕ and then fast prototype some effective designing instances for practical usage. In this work, we choose to span each involution kernel $\boldsymbol{\mathcal{H}}_{i,j}$ from a single pixel $\mathbf{X}_{i,j}$ for incarnation. More exquisite designs under exploration may have the potential of further pushing the performance boundary, but are left as future work. Besides, we are conscious that self-attention falls into this design space while being a more complicated materialization than our default choice, which is to be discussed in more detail in Section 4.2. Formally, we have the kernel generation function $\phi\colon \mathbb{R}^C \mapsto \mathbb{R}^{K\times K\times G}$ with $\boldsymbol{\Psi}_{i,j} = \{(i, j)\}$ taking the following form:

$$\boldsymbol{\mathcal{H}}_{i,j} = \phi(\mathbf{X}_{i,j}) = \mathbf{W}_1\sigma(\mathbf{W}_0\mathbf{X}_{i,j}). \tag{6}$$

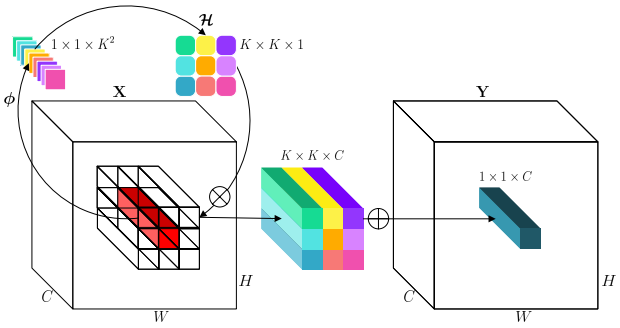


Figure 1: Schematic illustration of our proposed involution. The involution kernel $\boldsymbol{\mathcal{H}}_{i,j} \in \mathbb{R}^{K\times K\times 1}$ ($G = 1$ in this example for ease of demonstration) is yielded from the function ϕ conditioned on a single pixel at (i, j), followed by a channel-to-space rearrangement. The Multiply-Add operation of involution is decomposed into two steps, with $\bigotimes$ indicating multiplication broadcast across C channels and $\bigoplus$ indicating summation aggregated within the $K \times K$ spatial neighborhood. Best viewed in color.

In this formula, $\mathbf{W}_0 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $\mathbf{W}_1 \in \mathbb{R}^{(K\times K\times G)\times\frac{C}{r}}$ represent two linear transformations that collectively constitute a bottleneck structure, where the intermediate channel dimension is under the control of a reduction ratio r for efficient processing, and σ implies Batch Normalization and non-linear activation functions that interleave two linear projections. We refer to Eqn. 4 with the materialized kernel generation function of Eqn. 6 as involution hereinafter. The pseudo code shown in Alg. 1 delineates the computation flow of involution, which is visualized in Figure 1.

For building the entire network with involution, we mirror the design of ResNet [15] by stacking residual blocks, since the elegant architecture of ResNet makes it apt for incubating new ideas and making comparisons. We replace involution for 3×3 convolution at all bottleneck positions in the stem (using 3×3 or 7×7 involution for classification or dense prediction) and trunk (using 7×7 involution for all tasks) of ResNet, but retain all the 1×1 convolution for channel projection and fusion. These delicately redesigned entities unite to shape a new species of highly efficient backbone networks, termed as RedNet.

Once spatial and channel information interweaves, heavy redundancy tends to occur inside the neural networks. However, the information interactions are tactfully decoupled in our RedNet towards a favorable accuracy-efficiency trade-off, as empirically evidenced in Figure 2. To be specific, the information encoded in the channel dimension of one pixel is implicitly scattered to its spatial vicinity in the kernel generation step, after which the information in an enriched receptive field is gathered thanks to the vast and dynamic involution kernels. Indispensably, linear transformations (realized by 1×1 convolutions) are interspersed for channel information exchange. In a word, channel-spatial, spatial-alone, and channel-alone interactions alternately and independently act on the stream of information propagation, collaboratively facilitating the miniaturization of network architectures while ensuring the representation capability.

4. In Context of Prior Literature

This section relates to several important aspects revolving around neural architecture in prior literature. We clarify their similarities and differences compared to our method.

4.1. Convolution and Variants

As the *de facto* standard operator of modern vision systems, convolution [24] possesses two principal characteristics, spatial-agnostic and channel-specific. Convolution kernels are location-independent in the spatial extent for translation equivalence but privatized at different channels for information discrimination. Along another research line, depth-wise convolution demonstrates wide applicability in efficient neural network architecture design [6, 33, 28, 40]. The depth-wise convolution is a pioneering attempt towards factorizing the spatial and channel entanglement of standard convolution, which is symmetric to our proposed involution operation in that depth-wise convolution contains a set of kernels specific to each channel and spatially-shared while our invented involution kernels are shared over channels and dedicated to each planar location in the image lattice.

Until most recently, dynamic convolutions emerge as powerful variants of the stationary ones. These approaches either straightforwardly generate the entire convolution filters [13, 21, 47], or parameterize the sampling grid associated with each convolution kernel [9, 20, 55]. Regarding the former category [13, 21, 47], unlike us, their dynamically generated convolution filters still conform to the two properties of standard convolution, thus incurring significant memory or computation consumption for filter generation. Regarding the latter category [9, 20, 55], only certain attributes, *e.g.*, the footprint of convolution kernels, are determined in an adaptive fashion.

Actually, early in the field of face recognition, DeepFace [39] and DeepID [37] have explored locally connected layers without weight sharing in the spatial domain, enlightened by apparently different regional distributions of statistics in the face imagery. Nevertheless, such excessive relaxation of convolution parameters can be problematic in knowledge transfer from one position to others. Resembling dynamic convolutions, our involution tackles this dilemma through sharing *meta-weights* of the kernel generation function across different positions, though not directly the *weights* of kernel instances. There also exist previous works that adopt pixel-wise dynamic kernels for feature aggregation, but they mainly capitalize on the context information for feature up-sampling [35, 43] and still rely on convolution for basic feature extraction. The most relevant work towards substituting convolution rather than up-sampling might be [50], but the pixel-wise generated filters still inherit one original property of convolution, to perform feature aggregation in a distinct manner over each channel.

4.2. Attention Mechanism

The attention mechanism originates from the field of machine translation [41] and exhibits blossoming development in the arena of natural language processing [10, 48]. Its success has also been translated to a plethora of vision tasks, including image recognition [2, 17, 31, 53], image generation [29, 51], video understanding [36, 44], object detection [3, 16, 54], and semantic segmentation [12, 18, 42]. Some works sparingly insert self-attention as plugin modules into the backbone neural network [4, 49] or attach them on the top of the backbone to extract high-level semantic relationships [3, 36], retaining the substratum of convolutional features. More aggressively, other works adopt the off-the-shell self-attention layer as the fundamental backbone component for vision [2, 17, 31, 42, 53]. Still, limited emphasis has been laid on delving deep into the learning dynamics of this functional form compared to convolution [7].

Our proposed involution in Eqn. 4 is reminiscent of self-attention and essentially could become a generalized version of it. The self-attention pools *values* $\mathbf{V}$ depending on the affinities obtained by computing correspondences between the *query* and *key* content, $\mathbf{Q}$ and $\mathbf{K}$, formulized as

$$\mathbf{Y}_{i,j,k} = \sum_{(p,q)\in\mathbf{\Omega}} (\mathbf{Q}\mathbf{K}^\top)_{i,j,p,q,\lceil kH/C \rceil} \mathbf{V}_{p,q,k}, \qquad (7)$$

where $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are linearly transformed from the input $\mathbf{X}$, and H is the number of heads in multi-head self-attention [41]. The similarity lies in that both operators collect pixels in the neighborhood $\mathbf{\Delta}$ or a less bounded scope $\mathbf{\Omega}$ through a weighted sum. On the one hand, the computing regime of involution can be considered as an attentive aggregation over the spatial domain. On the other hand, the attention map, or say affinity matrix $\mathbf{Q}\mathbf{K}^\top$ in the self-attention, can be viewed as a kind of involution kernel $\boldsymbol{\mathcal{H}}$.

However, with the particulars of kernel generation comes the differences between self-attention and our materialized involution form with Eqn. 6. Regrading previous endeavor on replacing convolution with local self-attention [17, 31, 53] to establish backbone models, they have to derive the affinity matrix (equivalent to involution kernel in our context) based on the relationship between the *query* and *key content*, optionally with hand-crafted *relative positional encoding* for permutation-variance. From this point of view, for self-attention, the input to the kernel generation function in Eqn. 5 would become a set of pixels indexed by $\mathbf{\Psi}_{i,j} = (i,j) + \mathbf{\Delta}_K$[1], including both the pixel of interest and its surrounding ones. Subsequently, the function could compose all these attended pixels, in an either ordered [53] or unordered [17, 31, 53] manner, and exploit complex relationships between them. In stark contrast to above, we constitute the involution kernel via operating solely on the orig-

[1] $+$ indicates adding a variable vector to each element in a set here.

inal input pixel itself with $\mathbf{\Psi}_{i,j} = \{(i,j)\}$, as expressed by Eqn. 6. From the perspective of self-attention, our involution kernels only explicitly rely on the *query content*, while the *relative positional information* is implicitly encoded in the organized output form of our kernel generation function. We sacrifice the pixel-paired relationship modeling, but the final performance of our RedNet is on par with those heavily relation-based models. Therefore, we may reach a conclusion that it is the macro design principles of involution instead of its micro setup nuances that are instrumental in the representation learning for visual understanding, corroborated by our empirical results in the experimental part. Another strong evidence supporting our hypothesis is that only using position encoding (by replacing $\mathbf{QK}^\top$ in Eqn. 7 with $\mathbf{QR}^\top$, where $\mathbf{R}$ is the position embedding matrix) retains descent performance of self-attention based models [31, 1]. Previously, the above observation is interpreted as the crucial role of position encoding in self-attention, but now a reinterpretation of the root cause behind might be $\mathbf{QR}^\top$ is still a form of dynamically parameterized involution kernel.

More importantly, precedent self-attention based works seldom show their versatility in multifarious vision tasks, but our involution paves a viable pathway for a great variety of tasks, as we shall find soon in Section 5.1.

5. Experiments

5.1. Main Results

We conduct comprehensive experiments from conceptual prediction to (semi-)dense prediction. All the network models are implemented with the PyTorch library [30].

5.1.1 Image Classification

We perform the backbone training from scratch on the ImageNet [11] training set that is one of the most challenging benchmarks for object recognition up to date. For a fair comparison, we adhere to the training protocol of Stand-Alone Self-Attention [31] and Axial Attention [42], except that we *do not* use exponential moving average (EMA) over the trainable parameters during training. Following the identical recipe, we re-implement both pairwise and patchwise SAN [53] with their open-source code[2] as a stronger baseline, and show our reproduced results in the corresponding tables and figures respectively. The detailed training setup is provided in the Appendix. We apply the Inception-style preprocessing for data augmentation [38], *i.e.*, random resized cropping and horizontal flipping. For evaluation, we use the single-crop testing method on the validation set following the common practice.

In the same spirit of ResNet, we scale the network depth to establish our RedNet family. The comparison to convolution and self-attention based vision models are summarized in Table 1. Almost within each group of the table, RedNet achieves the highest recognition accuracy whilst with the most parsimonious parameter storage and computational budget. RedNet could substantially outperform ResNet across all depths. For example, RedNet-50 achieves a margin of 1.6% higher accuracy over ResNet-50, using 39.5% fewer parameters and 34.1% lower computational consumption. Moreover, RedNet-50 is on par with ResNet-101 regrading to the top-1 recognition accuracy, while saving 65.2% and 65.8% storage and computation resources respectively. For an intuitive demonstration, the corresponding accuracy-complexity envelope is illustrated in Figure 2a, where our RedNet shows the top-performing Pareto frontier, in abreast with other state-of-the-art self-attention models, while being free from more complex relation modeling. Likewise, we could observe a similar trend in the accuracy-parameter envelope shown in Figure 2b. It is noteworthy that RedNet strikes a better balance between

Architecture	#Params (M)	FLOPs (G)	Top-1 Acc. (%)
ResNet-26 [15]	13.7	2.4	73.6
LR-Net-26 [17]	14.7	2.6	75.7
Stand-Alone ResNet-26 [31]	10.3	2.4	74.8
SAN10† [53]	10.5	2.2	75.5
RedNet-26	**9.2**	**1.7**	**75.9**
ResNet-38 [15]	19.6	3.2	76.0
Stand-Alone ResNet-38 [31]	14.1	3.0	76.9
SAN15† [53]	14.1	3.0	77.1
RedNet-38	**12.4**	**2.2**	**77.6**
ResNet-50 [15]	25.6	4.1	76.8
LR-Net-50 [17]	23.3	4.3	77.3
AA-ResNet-50 [2]	25.8	4.2	77.7
Stand-Alone ResNet-50 [31]	18.0	3.6	77.6
SAN19† [53]	17.6	3.8	77.4
Axial ResNet-S‡ [42]	**12.5**	3.3	78.1
RedNet-50	15.5	**2.7**	**78.4**
ResNet-101 [15]	44.6	7.9	78.5
LR-Net-101 [17]	42.0	8.0	78.5
AA-ResNet-101 [2]	45.4	8.1	78.7
RedNet-101	**25.6**	**4.7**	**79.1**
ResNet-152 [15]	60.2	11.6	**79.3**
AA-ResNet-152 [2]	61.6	11.9	79.1
Axial ResNet-M‡ [42]	**26.5**	**6.8**	79.2
Axial ResNet-L‡ [42]	45.8	11.6	**79.3**
RedNet-152	34.0	**6.8**	**79.3**

Table 1: The architecture profiles on ImageNet val set. Single-crop testing with 224×224 crop size is adopted. We compare with improved re-implementations if available and extract the other results from their original publications.

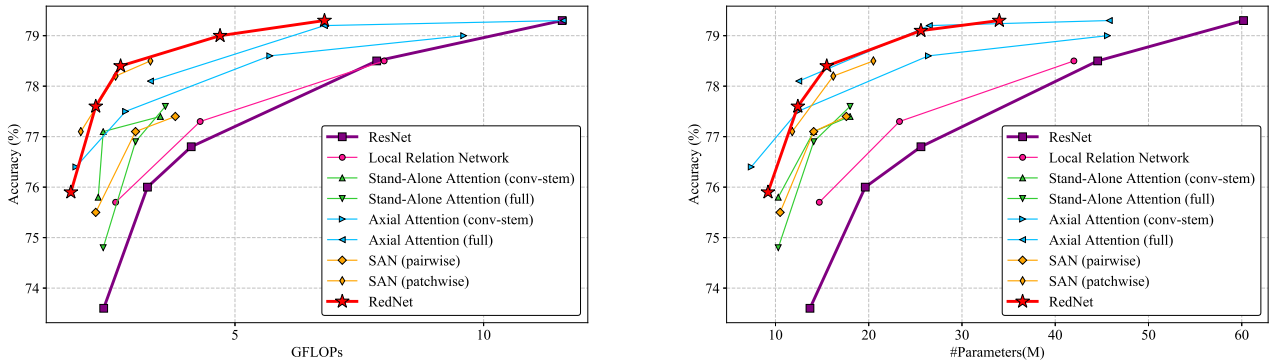
† *The improved re-implementation results of pairwise SAN models are listed here.*

‡ *Axial ResNet modifies the architecture setup of ResNet by changing the reduction ratio in each bottleneck block from 4 to 2.*

Architecture	GPU time (ms)	CPU time (ms)	Top-1 Acc. (%)
ResNet-50 [15]	11.4	895.4	76.8
ResNet-101 [15]	18.9	967.4	78.5
SAN19 [53]	33.2	N/A	77.4
Axial ResNet-S [42]	35.9	377.0	78.1
RedNet-38	11.4	156.3	77.6
RedNet-50	14.3	211.2	78.4

Table 2: Runtime analysis for representative networks. The speed benchmark is on a single NVIDIA TITAN Xp GPU and Intel® Xeon® CPU E5-2660 v4@2.00GHz.

[2] https://github.com/hszhao/SAN



(a) The accuracy-complexity envelope on ImageNet.

(b) The accuracy-parameter envelope on ImageNet.

Figure 2: The accuracy-efficiency envelopes on ImageNet val set. This figure visualizes Table 1. In general, RedNet achieves the optimal trade-off in comparison with all the other convolution and self-attention based architectures.

parameters and complexities, compared to the top competitors like SAN and Axial ResNet, as they are enveloped by the curve of RedNet series either in Figure 2a or 2b.

To reflect the practical runtime, we measure the inference time of different architectures with the comparable performance for a single image with the shape of 224×224. We report the running time on GPU/CPU in Table 2, where RedNet demonstrates its merits in terms of wall-clock time under the same level of accuracy. A customized CUDA kernel implementation with optimized memory scheduling for involution is highly anticipated for further acceleration on GPU. Depending on the extent to which optimizing hardware accelerators is contributed to this new involution operator, on-device speedup might approach the theoretical speedup compared to convolution in the future.

5.1.2 Object Detection and Instance Segmentation

Beyond fundamental image classification, we demonstrate the generalization ability of our proposed involution on downstream vision tasks, such as object detection and instance segmentation. For object detection, we employ the representative one- and two-stage detectors, RetinaNet [26] and Faster R-CNN [32], both equipped with the FPN [25] neck. For instance segmentation, we adopt the main-stream detection system, Mask R-CNN [14], also in companion with FPN. These three detectors with the underlying backbones, ResNet-50 or RedNet-50, are fine-tuned on the Microsoft COCO [27] `train2017` set for transferring the learned representations of images. More training details are reported in the appendix. During quantitative evaluation, we test on the `val2017` set and report the COCO-style mean Average Precision (mAP) under different IoU thresholds ranging from 0.5 to 0.95 with an increment of 0.05.

Table 3 compares our models against the baseline of ResNet backbone with the convolution-based neck and head. First, with the RedNet backbone, all the three detectors excel their ResNet-based counterparts with considerable performance gains, *i.e.*, 1.7%, 1.8%, and 1.8% higher in bounding box AP, while being more parameter- and computation-conserving. Second, additionally swapping involution for convolution in the FPN neck brings about healthy margins for Faster/Mask R-CNN, while further reducing their parameters and computational cost to 71%/73% and 65%/72%. In particular, the margins with respect to bounding box AP are enlarged to 2.5% and 2.4% respectively. Third, to build fully involution-based detectors, we further replace convolution in the task-specific heads of Faster/Mask R-CNN with involution, which could cut down more than half of the computational complexity while retaining the superior or on-par performance. This kind of fully involution-based detectors may stand out especially in cases where computational resource is the major bottleneck. Forth, we pay special attention to the scores of small/medium/large objects and notice that the most compelling performance improvement appears in the measurement of AP_L. Our best detection models could surpass the baselines by more than 3% bounding box AP in this regard, specifically 3.4%, 4.3%, and 3.3% for RetinaNet, Faster R-CNN, and Mask R-CNN. We hypothesize that the success of detecting large-scale objects arises from the design of spread-out and position-aware involution kernels. Besides AP_L, the performance gains are consistent under the fine-grained taxonomy of AP evaluation metrics, demonstrated in different columns of Table 3.

5.1.3 Semantic Segmentation

To further exploit the versatility of involution, we also conduct experiments on the task of semantic image segmentation. We choose the segmentation frameworks of Semantic FPN [22] and UPerNet [45], loaded with ImageNet pre-trained backbone weights. We fine-tune these segmentors on the finely-annotated part of the Cityscapes dataset [8], which contains a split of 2975 and 500 images for training and validation respectively, divided into 19 classes. More training details can be found in the appendix. After training, we perform the evaluation on the validation set un-

Detector	Backbone	Neck	#Params (M)	FLOPs (G)	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}_{S}	AP^{bbox}_{M}	AP^{bbox}_{L}
RetinaNet [26]	ResNet-50	convolution	37.7	239.3	36.6	55.8	39.2	20.9	40.6	47.5
	RedNet-50	convolution	27.8	210.1	38.3 (+1.7)	58.2 (+2.4)	40.5 (+1.3)	21.1 (+0.2)	41.8 (+1.2)	50.9 (+3.4)
	RedNet-50	involution	26.3	199.9	38.2 (+1.6)	58.2 (+2.4)	40.4 (+1.2)	21.8 (+0.9)	41.6 (+1.0)	50.7 (+3.2)

Detector	Backbone	Neck	Head	#Params (M)	FLOPs (G)	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}_{S}	AP^{bbox}_{M}	AP^{bbox}_{L}
Faster R-CNN [32]	ResNet-50	convolution	convolution	41.5	207.1	37.7	58.7	40.8	21.7	41.6	48.4
	RedNet-50	convolution	convolution	31.6	177.9	39.5 (+1.8)	60.9 (+2.2)	42.8 (+2.0)	23.3 (+1.6)	42.9 (+1.3)	52.2 (+3.8)
	RedNet-50	involution	convolution	29.5	135.0	40.2 (+2.5)	62.1 (+3.4)	43.4 (+2.6)	24.2 (+2.5)	43.3 (+1.7)	52.7 (+4.3)
	RedNet-50	involution	involution	29.0	91.5	39.2 (+1.5)	61.0 (+2.3)	42.4 (+1.6)	23.1 (+1.4)	43.0 (+1.4)	50.7 (+2.3)

Detector	Backbone	Neck	Head	#Params (M)	FLOPs (G)	AP	AP_{50}	AP_{75}	AP_{S}	AP_{M}	AP_{L}
Mask R-CNN [14]	ResNet-50	convolution	convolution	44.2	253.4	38.4	59.2	41.9	21.9	42.3	49.7
						35.1	56.3	37.3	18.5	38.6	46.9
	RedNet-50	convolution	convolution	34.2	224.2	40.2 (+1.8)	61.4 (+2.2)	43.7 (+1.8)	24.2 (+2.3)	43.4 (+1.1)	52.5 (+2.8)
						36.1 (+1.0)	58.1 (+1.8)	38.2 (+0.9)	19.9 (+1.4)	39.3 (+0.7)	48.9 (+2.0)
	RedNet-50	involution	convolution	32.2	181.3	40.8 (+2.4)	62.3 (+3.1)	44.3 (+2.4)	24.2 (+2.3)	44.0 (+1.7)	53.0 (+3.3)
						36.4 (+1.3)	59.0 (+2.7)	38.5 (+1.2)	19.9 (+1.4)	39.4 (+0.8)	49.1 (+2.2)
	RedNet-50	involution	involution	29.5	104.6	39.6 (+1.2)	60.7 (+1.5)	42.7 (+0.8)	23.5 (+1.6)	43.1 (+0.8)	51.1 (+1.4)
						35.1 (+0.0)	57.1 (+0.8)	37.3 (+0.0)	19.2 (+0.7)	38.5 (−0.1)	47.3 (+0.4)

Table 3: Performance comparison on COCO detection and segmentation. The bounding box AP is reported for the object detection track in the upper table. The bounding box and mask AP are simultaneously reported for the instance segmentation track in the lower table, listed in the two separate lines following a single detector. In the parentheses are the gaps to the fully convolution-based counterparts. Highlighted in green are the gaps of at least +2.0 points, the same in Table 4 and 5.

Segmentor	Backbone	Neck	#Params (M)	FLOPs (G)	mean IoU (%)	wall	truck	bus
Semantic FPN [22]	ResNet-50	convolution	28.5	362.8	74.5	39.4	58.6	72.2
	RedNet-50	convolution	18.5	293.9	78.3 (+3.8)	52.7 (+13.3)	77.3 (+18.7)	87.6 (+15.4)
	RedNet-50	involution	16.4	205.2	79.2 (+4.7)	56.9 (+17.5)	82.1 (+23.5)	88.5 (+16.3)

Table 4: Performance comparison on Cityscapes segmentation based on Semantic FPN. We showcase the mean IoU averaged over all classes, as well as IoUs of the top three classes with the most evident performance amelioration.

Segmentor	Backbone	#Params (M)	FLOPs (G)	mIoU (%)
UPerNet [45]	ResNet-50	66.4	1894.5	78.2
	RedNet-50	56.4	1825.6	80.6 (+2.4)
Panoptic-DeepLab [5]	Axial-DeepLab-S	12.1	220.8	80.5
	Axial-DeepLab-M	25.9	419.6	80.3
	Axial-DeepLab-XL	173.0	2446.8	80.6

Table 5: Performance comparison on Cityscapes segmentation based on UPerNet. The efficiency of UPerNet is greatly boosted by the RedNet backbone, showing competitive performance to Axial-DeepLab-XL with only 32.6% parameter counts and 75.7% computational cost.

der the single-scale mode and adopt the Intersection-over-Union (IoU) as the evaluation metric.

Based on the Semantic FPN framework, we are able to achieve 3.8% higher mean IoU over all classes, taking advantage of RedNet over ResNet as the backbone. Consequent to further infusing involution into the FPN neck to replace convolution, the gain in mean IoU is elevated to 4.7% but the parameters and FLOPs are cut down to 57.5% and 56.6% of the baseline model accordingly. The detailed comparison results are shown in Table 4. To take one step further, we investigate the effectiveness of our method on different object classes. Aligned with the discovery in object detection, we notice that the segmentation effects of those objects with a large spatial arrangement are improved by more than 10%, *e.g.*, wall, truck, and bus, while slight improvements are observed in classes of relatively small objects, *e.g.*, traffic light, traffic sign, person, and bicycle. Once again, the involution operation effectively aids the large object perception by endowing the representation process with dynamic and distant interactions. In addition, we replace the ResNet backbone of UPerNet with RedNet and evaluate the final performance, as displayed in Table 5. Though not an apple-to-apple comparison using the same segmentor and training strategy, RedNet-based UPerNet appears more efficient than Axial-DeepLab, which is dedicatedly designed for segmentation tasks by converting the original Axial ResNet backbone network.

5.2. Ablation Analysis

We present several ablation studies designed to understand the contributions of individual components, taking RedNet-50 as an example.

Stem First of all, we isolate the impact of involution on the network stem. Following the practice of recent self-attention based architectures [53, 42], the network stem is decomposed into three consecutive operations to save memory cost. In accordance with our practice of integrating involution into the trunk, we place 3×3 involution at the bottleneck position of the stem. This act improves the accuracy from 77.7% to 78.4% with marginal cost, leading to our default setting of RedNet in the main experiments.

Otherwise explicitly mentioned, we use RedNet-50 with 7×7 *convolution stem* for the following ablation analysis.

Kernel Size In the spatial dimension, we probe the effect of kernel size. Steady improvement is observed in Table 6a when increasing the spatial extent up to 7×7 with negligible computational overheads. The improvement somewhat plateaus when further expanding the spatial extent, which is possibly relevant to the feature resolution in the network.

Kernel Size	#Params (M)	FLOPs (G)	Top-1 Acc. (%)
3×3	14.7	2.4	76.9
5×5	15.1	2.5	77.4
7×7	15.5	2.6	77.7
9×9	16.2	2.7	77.8

(a) Accuracy saturates with **kernel size** increasing.

#Group Channel	#Params (M)	FLOPs (G)	Top-1 Acc. (%)
1	30.2	5.0	77.9
4	18.5	3.0	77.7
16	15.5	2.6	77.7
C	14.6	2.4	76.5

(b) Appropriate **grouping channels** improves efficiency.

Function Form	#params (M)	FLOPs (G)	Top-1 Acc. (%)
$\mathbf{W}$	18.1	3.0	77.8
$\mathbf{W}_1\sigma\mathbf{W}_0, r = 1$	19.4	3.2	77.8
$\mathbf{W}_1\sigma\mathbf{W}_0, r = 4$	15.5	2.6	77.7
$\mathbf{W}_1\sigma\mathbf{W}_0, r = 16$	14.6	2.4	77.4

(c) Introducing the **bottleneck structure** reduces complexity.

Table 6: We examine the role of different components in the design of involution, including kernel size, group channels, and the form of kernel generation function. In gray are entries with the default setting as our main experiments. When we adjust one hyper-parameter for ablation, we leave the others as the default setting. The final performance is not sensitive to each hyper-parameter configuration.

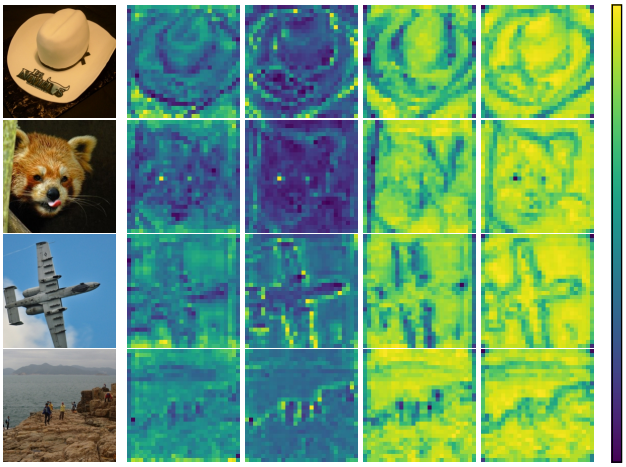


Figure 3: The heat maps in each row interpret the generated kernels for an image instance from the ImageNet validation set, drawn from four different classes, including cowboy hat, lesser panda, warplane, and cliff (from top to bottom).

This set of controlled experiments shows the superiority of harnessing large involution kernels over compact and static convolution, while avoiding to introduce prohibitive memory and computational cost.

Group Channel In the channel dimension, we assess the feasibility of sharing an involution kernel. As can be seen in Table 6b, sharing a kernel per 16 channels halves the parameters and computational cost compared to the non-shared one, only sacrificing 0.2% accuracy. However, sharing a single kernel across all the C channels obviously underperforms in accuracy. Considering the channel redundancy of involution kernels, as long as setting the channels shared in a group to an acceptable range, the channel-agnostic behavior will not only reserve the performance, but also reduce the parameter count and computational cost. This will also permit a larger kernel size under the same budget.

Kernel Generation Function Next, we validate the utility of bottleneck architecture for the kernel generation process in Table 6c. Adopting a single linear transform $\mathbf{W}$ or two transforms without bottleneck ($r = 1$) as the kernel generation function incurs more parameters and FLOPs but only performs marginally better, compared to the default setting ($r = 4$). Moreover, inferior performance could be ascribed to aggressive channel reduction ($r = 16$).

Further attaching activation functions such as softmax, sigmoid to the kernel generation function, would constrain the kernel values, thus restrict its expressive capability, and ends up hindering the performance by over 1%. So we opt not to insert any additional functions at the output end of the kernel generation function, allowing the generated kernel to span the entire subspace of $K \times K$ matrices.

5.3. Visualization

For dissecting the learned involution kernels, we take the sum of $K \times K$ values from each involution kernel as its representative value. All the representatives at different spatial locations frame the corresponding heat map. Some selected heat maps are plotted in Figure 3, where the columns following the original image indicate different involution kernels in the last block of the third stage (conv3_4 following the naming convention of [15]), separated by groups. On the one hand, involution kernels automatically attend to crucial parts of objects in the spatial range for correct image recognition. On the other hand, in a single involution layer, different kernels from different groups focus on varying semantic concepts of the original image, by highlighting peripheral parts, sharp edges or corners, smoother regions, outline of the foreground and background objects, respectively (from left to right in each row).

6. Conclusion and Prospect

In this work, we present involution, an effective and efficient operator for visual representation learning, reversing the design principles of convolution and generalizing the formulation of self-attention. Thanks to the medium of involution, we are able to disclose the underlying relationship between self-attention and convolution and empirically ascertain the essential driving force of self-attention for its recent progress in vision. Our proposed involution is benchmarked on several standard vision benchmarks, consistently delivering enhanced performance at reduced cost compared to convolution-based counterparts and self-attention based models. Furthermore, careful ablation analysis helps us better understand that such performance enhancement is rooted in the core contributions of involution, from the efficacy of spatial modeling to the efficiency of architecture design.

We believe that this work could foster future research enthusiasm on simple yet effective visual primitives beyond convolution, which is expected to make inroads into fields of neural architecture engineering where uniform and local spatial modeling has prestigiously dominated.

References

[1] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *ICLR*, 2021. 5

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *ICCV*, 2019. 4, 5

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 4

[4] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ2-nets: Double attention networks. In *NeurIPS*, 2018. 4

[5] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 7

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 2, 4

[7] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020. 4

[8] 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 *CVPR*, 2016. 6

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 4

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, 2019. 4

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 5

[12] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019. 4

[13] David Ha, Andrew Dai, and Quoc Le. Hypernetworks. In *ICLR*, 2017. 4

[14] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 6, 7

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 5, 8

[16] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018. 4

[17] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019. 4, 5

[18] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019. 4

[19] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014. 1

[20] Yunho Jeon and Junmo Kim. Active convolution: Learning the shape of convolution for image classification. In *CVPR*, 2017. 4

[21] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NIPS*, 2016. 4

[22] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic feature pyramid networks. In *CVPR*, 2019. 6, 7

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[25] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *ICCV*, 2017. 6, 7

[27] 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 *ECCV*, 2014. 6

[28] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 4

[29] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018. 4

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 5

[31] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019. 2, 4, 5

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 6, 7

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 4

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[35] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *CVPR*, 2019. 4

[36] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *ICCV*, 2019. 4

[37] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *CVPR*, 2014. 4

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 5

[39] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 4

[40] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 4

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 4

[42] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020. 4, 5, 7

[43] Jiaqi Wang, Kai Chen, Rui Xu, Ziwei Liu, Chen Change Loy, and Dahua Lin. Carafe: Content-aware reassembly of features. In *ICCV*, 2019. 4

[44] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 4

[45] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 6, 7

[46] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2

[47] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019. 4

[48] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019. 4

[49] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In *NeurIPS*, 2018. 4

[50] Julio Zamora Esquivel, Adan Cruz Vargas, Paulo Lopez Meyer, and Omesh Tickoo. Adaptive convolutional kernels. In *ICCV Workshops*, 2019. 4

[51] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019. 4

[52] Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019. 1

[53] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020. 2, 4, 5, 7

[54] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *ICCV*, 2019. 4

[55] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 4