Recent Research Paper on Convolution

Xiangtai Li
EECS@PKU
2019.4.15
lxtpku@pku.edu.cn

Outline:

Outline:

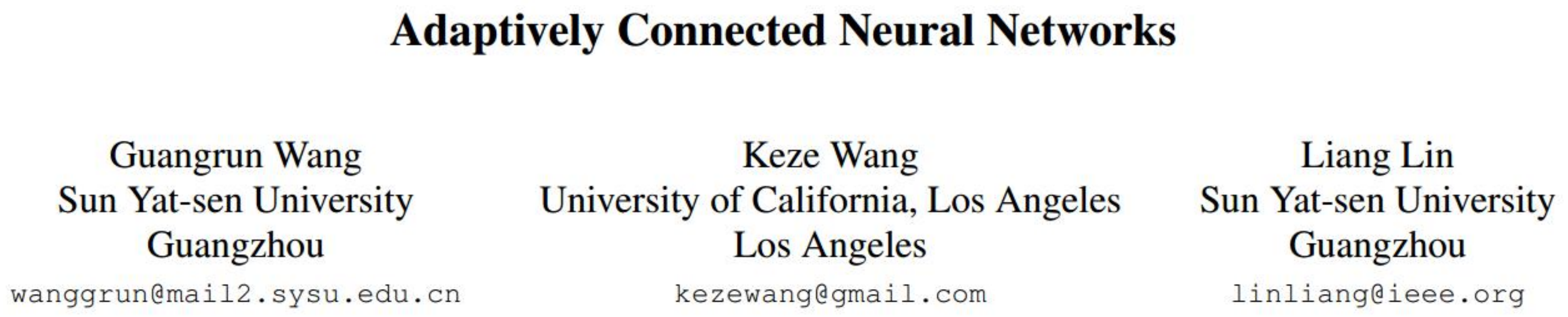
1. Adaptively Connected Neural Networks (CVPR 2019)

2. Pixel-Adaptive Convolutional Neural Networks (CVPR 2019)

3. HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs (CVPR 2019)

4. Drop an Octave: Reducing Spatial Redundancy in
Convolutional Neural Networks with Octave Convolution (arxiv)

5. Res2Net: A New Multi-scale Backbone Architecture (arxiv)



Adaptively Connected Neural Networks

Guangrun Wang
Sun Yat-sen University
Guangzhou
wanggrun@mail2.sysu.edu.cn

Keze Wang
University of California, Los Angeles
Los Angeles
kezewang@gmail.com

Liang Lin
Sun Yat-sen University
Guangzhou
linliang@ieee.org

Hightlight: proposes a novel adaptively connected network (ACNet)

Imporves traditional convolutional neural networks in two aspects:

1. a flexible way to switch global and local inference in
processing the internal feature representations.

(Eg:show that existing CNNs, the classical multilayer perceptron (MLP), and
the recently proposed non-local network (NLN) are all
special cases of ACNet.)

2. is also capable of handling non-Euclidean data.

Motivation

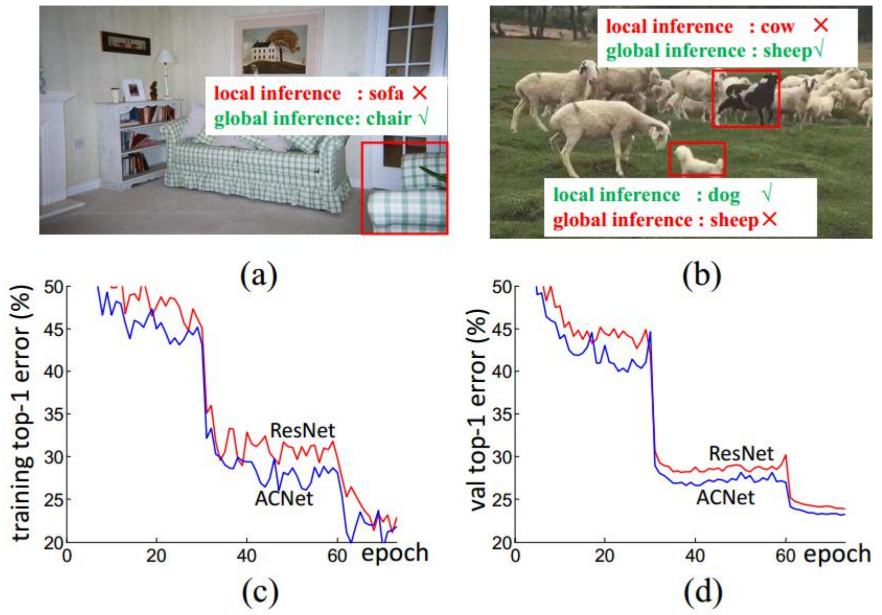


Figure 1: Some pixels prefer global dependencies, while others prefer local inference. For example, without global inference we cannot recognize the *chair* in (a). While in (b), the representation capacity of the *dog* is weakened by global information. Thanks to the adaptively determining the global/local inference, our AC-Net achieves lower top-1 training/validation error than ResNet on ImageNet-1k shown in (c) and (d).

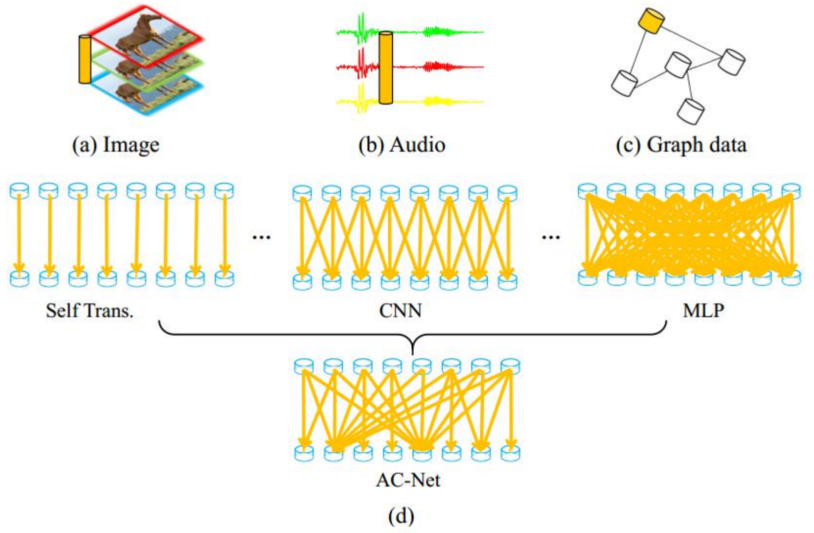


Figure 2: "Nodes" are presented in form of orange cylinder in (a) an image, (b) an audio, and (c) a general graph. (d) ACNet can be considered as a generalization of MLP and CNN on these "nodes".

MLP: fully global information.
CNN: local information processing unit

Methods

share the same idea with Switchbale Normalization (SN)

$$\mathbf{y}_i = \alpha_i \sum_{j=i} \mathbf{x}_j \mathbf{u}_{ij} + \boxed{\beta_i \sum_{j \subseteq N(i)} \mathbf{x}_j \mathbf{v}_{ij}} + \boxed{\gamma_i \sum_{\forall j} \mathbf{x}_j \mathbf{w}_{ij},} \qquad (1)$$

dataset-aware ACNet

$$\alpha = \frac{e^{\lambda_\alpha}}{e^{\lambda_\alpha} + e^{\lambda_\beta} + e^{\lambda_\gamma}}. \qquad (2)$$

$$\gamma_i = \gamma_i(\mathbf{x}) = \mathbf{w}_{\gamma_i,2} f(\mathbf{w}_{\gamma_i,1} \Big[\sum_{j=i} \mathbf{x}_j \mathbf{u}_{ij}; \sum_{j \subseteq N(i)} \mathbf{x}_j \mathbf{v}_{ij}; \sum_{\forall j} \mathbf{x}_j \mathbf{w}_{ij} \Big]), \qquad (3)$$

pixel-aware ACNet

Notice:
1. 3rd iterm is first transformed by an average pooling for downsampling.
2. All parameters are learned end-to-end.
3. obtained y in Eqn.1 can be activated by a non-linear function f(·), such as BatchNorm+ReLU.
4. Eqn.3 is another implementation formula if alpha/beta are scalars.

Genaeralization to Non-Euclidean Data

$$\mathbf{y}_i = \alpha_i \sum_{j=i} \mathbf{x}_j \mathbf{u} + \beta_i \sum_{j \subseteq N(i)} p_{ij}(\mathbf{x}_j \mathbf{v}) + \gamma_i \sum_{\forall j} q_{ij}(\mathbf{x}_j \mathbf{w}). \quad (5)$$

$$p_{ij}(\mathbf{x}_j \mathbf{v}) = \mathbf{x}_j \mathbf{v} \zeta_{ij}, \quad q_{ij}(\mathbf{x}_j \mathbf{w}) = \mathbf{x}_j \mathbf{w} \xi_{ij} \quad (6)$$

where ζ_{ij} and ξ_{ij} are constant variables sampled from a Gaussian noise.

Experiments:

Table 1: Comparison of ImageNet *val* top-1 accuracies and parameter numbers on ResNet50. **ACNet‡:** pixel-aware ACNet using Eqn. 3; **ACNet:** dataset-aware ACNet with α, β, γ being scalar variables;

	top-1 accuracies	#params
CNN-ResNet50	$76.4^{\uparrow 0.0}$	$25.56\text{M}^{\times 1.00}$
ACNet-ResNet50	$\mathbf{77.5}^{\uparrow 1.1}$	$29.38\text{M}^{\times 1.15}$
ACNet‡-ResNet50	$\mathbf{77.5}^{\uparrow 1.1}$	$31.85\text{M}^{\times 1.25}$
generalized ACNet	$\mathbf{76.2}^{\downarrow 0.2}$	$\mathbf{19.80M}^{\times 0.77}$

Table 2: Comparison between ACNet-Resnet50 and CNN-ResNet60 in terms of ImageNet val top-1 accuracies and parameter numbers.

	top-1 accuracies (%)	#params
CNN-ResNet60	$76.7^{\uparrow 0.0}$	$30.03\text{M}^{\times 1.00}$
ACNet-ResNet50	$\mathbf{77.5}^{\uparrow 0.8}$	$\mathbf{29.38M}^{\times 0.98}$

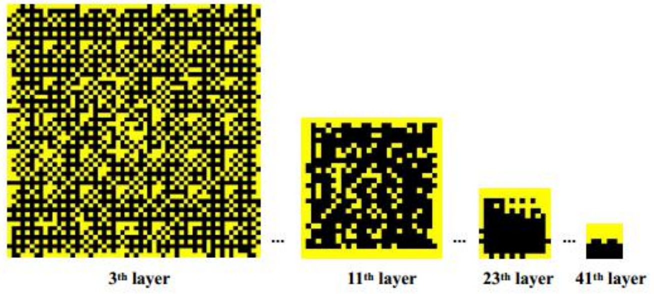


Figure 3: Visualization of the nodes with different types of inference generated by our ACNet, which is trained on ImageNet. One node painted by the yellow color indicates its the output of the global inference from the preceding layer (i.e., it connects to all nodes in the preceding layer), while the opposite black nodes indicate the outputs of the local inference from the preceding layer.

Table 4: Ablation studies on CIFAR10.

Method	Error (%)
Standard ACNet	$6.0^{\downarrow 0.0}$
w/o global inference	$7.1^{\downarrow 1.1}$
w/o local inference	$24.0^{\downarrow 18}$
Fixed global+local	$6.8^{\downarrow 0.8}$

Table 5: Detection and segmentation ablation studies on COCO2017 using Mask RCNN.

backbone	AP^{bbox}	AP^{mask}
CNN	$38.0^{\uparrow 0.0}$	$34.6^{\uparrow 0.0}$
NLN	$39.0^{\uparrow 1.0}$	$35.5^{\uparrow 0.9}$
ACNet	$\mathbf{39.5}^{\uparrow 1.5}$	$\mathbf{35.2}^{\uparrow 0.6}$

Table 6: Comparison on a Person Re-identification task (CUHK03, where 'bs' denotes batch size.)

	Rank-1
BOW+XQDA [53]	6.4
PUL [7]	9.1
LOMO+XQDA [25]	12.8
IDE [54]	21.3
IDE+DaF [51]	26.4
IDE+XQ.+Re-ranking [55]	34.7
PAN	36.3
DPFL [4]	40.7
SVDNet [39]	41.5
TriNet + Era. [56]	55.5
TriNet + Era.(Our reproduction)	$62.0^{\uparrow 0.0}$
TriNet + Era. + ACNet	$64.3^{\uparrow 2.3}$
TriNet + Era. + reranking(bs = 32)	$61.2^{\uparrow 0.0}$
TriNet + Era. + reranking + ACNet(bs = 32)	$64.8^{\uparrow 3.6}$

Implementation:

dataset-aware ACNet

pixel-aware ACNet

1. Simplely combine the conv and fc

2. Downsample operation use GAP(gobal average pooling) with 2 fc(full conected layer) behind.

3. dataset-aware means the combination parameters are leanable in data driven ways.

Summary:

1. Very well writen but the implementation is very simple. The core idea is embedding global information into convolution operation in cascade manner.

2. A lot of experiments but like analysis on important role of learned parameters.

Outline:

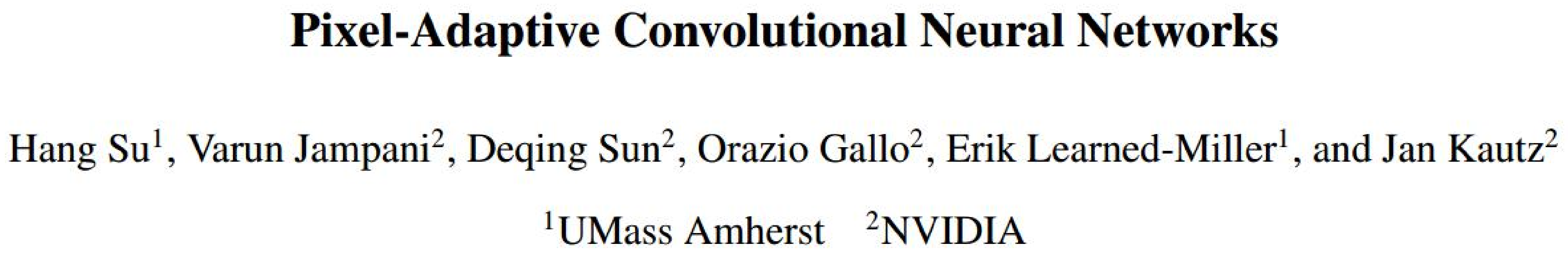
1. Adaptively Connected Neural Networks (CVPR 2019)

2. Pixel-Adaptive Convolutional Neural Networks (CVPR 2019)

3. HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs (CVPR 2019)

4. Drop an Octave: Reducing Spatial Redundancy in
Convolutional Neural Networks with Octave Convolution (arxiv)

5. Res2Net: A New Multi-scale Backbone Architecture (arxiv)



Pixel-Adaptive Convolutional Neural Networks

Hang Su[1], Varun Jampani[2], Deqing Sun[2], Orazio Gallo[2], Erik Learned-Miller[1], and Jan Kautz[2]

[1]UMass Amherst [2]NVIDIA

Highlight: propose a **pixel-adaptive convolution** (PAC) operation, in which the filter weights are **multiplied** with a **spatially varying kernel that depends on learnable, local pixel features.**

CNN drawbacks: Spatial Sharing and its ContentAgnostic nature

1. Spatial Sharing:
Although spatially invariant convolutions significantly reduce the number of parameters.
For dense pixel predition tasks, the loss is spatially varying because of varying scene elements on a pixel grid.

2. Content Agnostic: Trained CNN shares the same parameters with different images as inputs.

Content-Adaptive Convolutional Networks

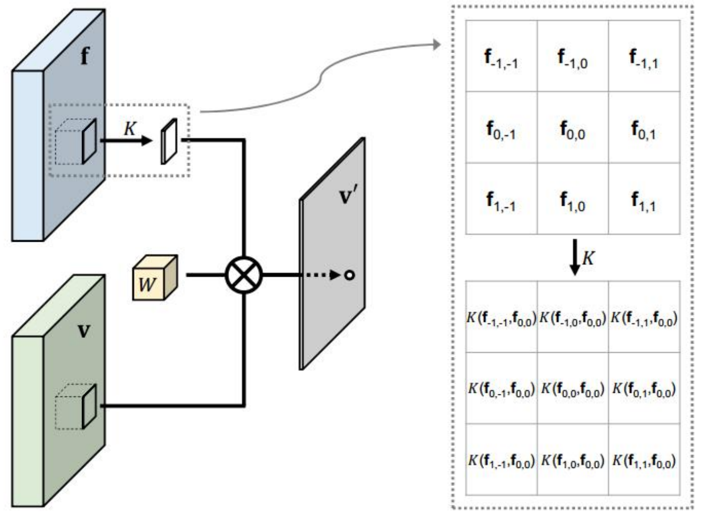

f
K
v
W
v′
$\mathbf{f}_{-1,-1}$ $\mathbf{f}_{-1,0}$ $\mathbf{f}_{-1,1}$
$\mathbf{f}_{0,-1}$ $\mathbf{f}_{0,0}$ $\mathbf{f}_{0,1}$
$\mathbf{f}_{1,-1}$ $\mathbf{f}_{1,0}$ $\mathbf{f}_{1,1}$
K
$K(\mathbf{f}_{-1,-1},\mathbf{f}_{0,0})$ $K(\mathbf{f}_{-1,0},\mathbf{f}_{0,0})$ $K(\mathbf{f}_{-1,1},\mathbf{f}_{0,0})$
$K(\mathbf{f}_{0,-1},\mathbf{f}_{0,0})$ $K(\mathbf{f}_{0,0},\mathbf{f}_{0,0})$ $K(\mathbf{f}_{0,1},\mathbf{f}_{0,0})$
$K(\mathbf{f}_{1,-1},\mathbf{f}_{0,0})$ $K(\mathbf{f}_{1,0},\mathbf{f}_{0,0})$ $K(\mathbf{f}_{1,1},\mathbf{f}_{0,0})$

Figure 1: **Pixel-Adaptive Convolution.** PAC modifies a standard convolution on an input $\mathbf{v}$ by modifying the spatially invariant filter $\mathbf{W}$ with an adapting kernel K. The adapting kernel is constructed using either pre-defined or learned features $\mathbf{f}$. $\otimes$ denotes element-wise multiplication of matrices followed by a summation. Only one output channel is shown for the illustration.

Traditional Convolution

$$\mathbf{v}_i' = \sum_{j\in\Omega(i)} \mathbf{W}\left[\mathbf{p}_i - \mathbf{p}_j\right]\mathbf{v}_j + \mathbf{b} \qquad (1)$$

Pixel-Adaptive Convolution

$$\mathbf{v}_i' = \sum_{j\in\Omega(i)} K\left(\mathbf{f}_i, \mathbf{f}_j\right)\mathbf{W}\left[\mathbf{p}_i - \mathbf{p}_j\right]\mathbf{v}_j + \mathbf{b} \qquad (3)$$

The **adapting features f** can be either hand-specified such as position and color features f = (x; y; r; g; b) or can be deep features that are learned end-to-end.

Deep Joint Upsampling Networks share the same idea with Sifei Liu's work

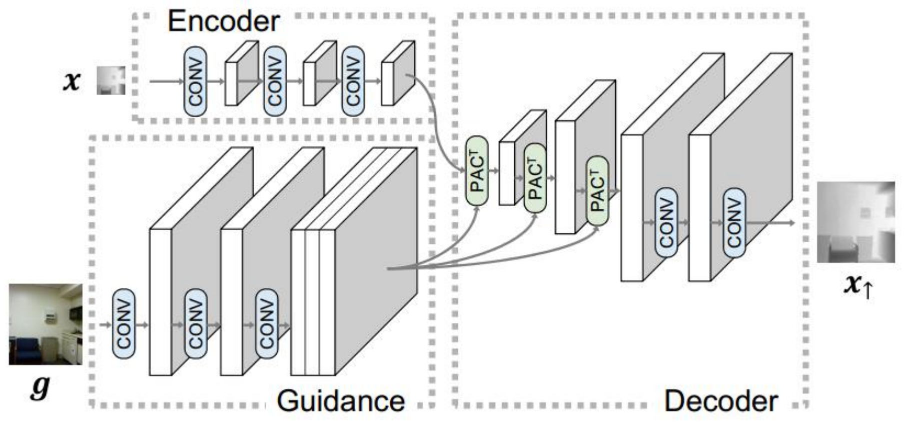


Figure 2: **Joint upsampling with PAC.** Network architecture showing encoder, guidance and decoder components. Features from the guidance branch are used to adapt PAC$^{\mathsf{T}}$ kernels that are applied on the encoder output resulting in upsampled signal.

- *Encoder* branch operates directly on the low-resolution signal with convolution (CONV) layers.
- *Guidance* branch operates solely on the guidance image, and generates adapting features that will be used in all PAC$^{\mathsf{T}}$ layers later in the network.
- *Decoder* branch starts with a sequence of PAC$^{\mathsf{T}}$, which perform transposed pixel-adaptive convolution, each of which upsamples the feature maps by a factor of 2. PAC$^{\mathsf{T}}$ layers are followed by two CONV layers to generate the final upsampled output.

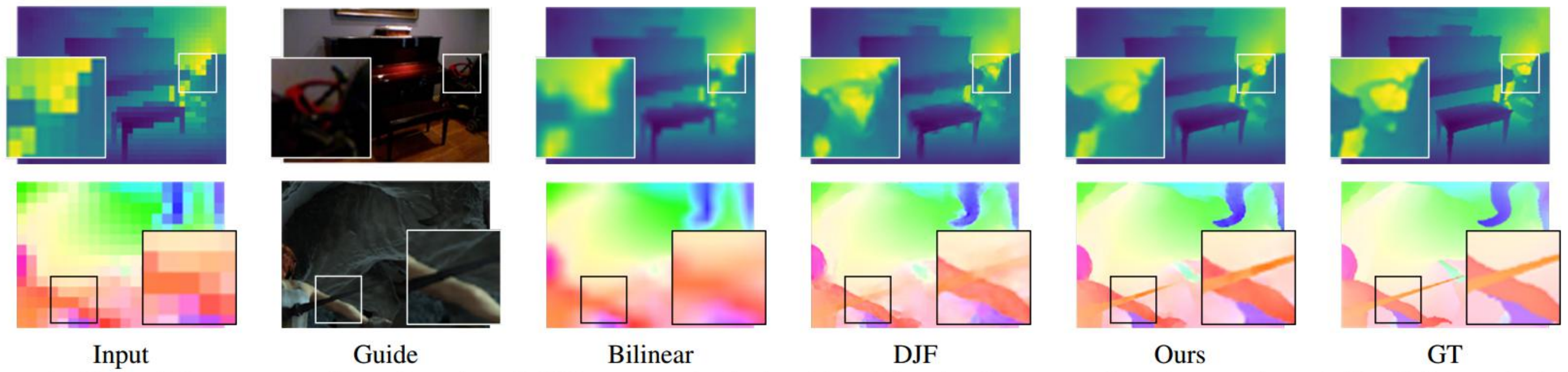


Figure 3: **Deep joint upsampling.** Results of different methods for 16× joint depth upsampling (top row) and 16× joint optical flow upsampling (bottom row). Our method produces results that have more details and are more faithful to the edges in the guidance image.

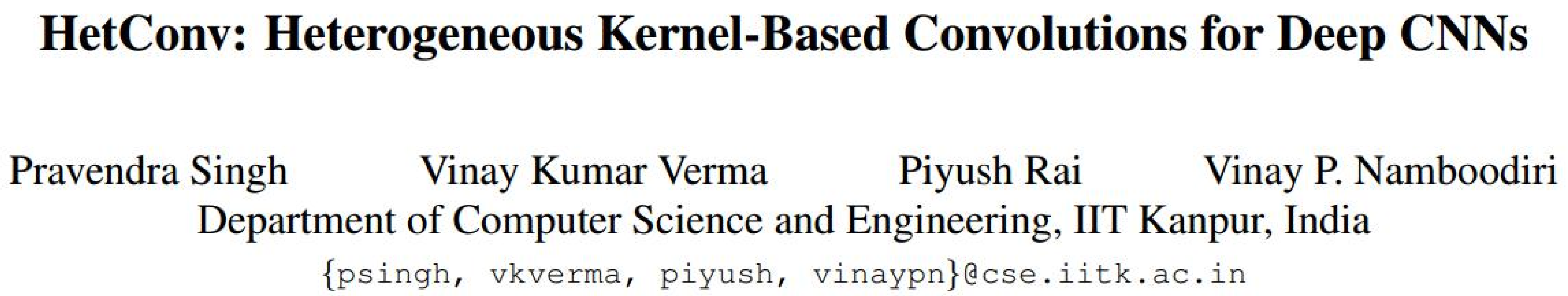
Table 1: **Joint depth upsampling.** Results (in RMSE) show that our upsampling network consistently outperforms other techniques for different upsampling factors.

Method	4×	8×	16×
Bicubic	8.16	14.22	22.32
MRF	7.84	13.98	22.20
GF [18]	7.32	13.62	22.03
JBU [24]	4.07	8.29	13.35
Ham *et al*. [15]	5.27	12.31	19.24
DMSG [19]	3.78	6.37	11.16
FBS [5]	4.29	8.94	14.59
DJF [27]	3.54	6.20	10.21
DJF+ [28]	3.38	5.86	10.11
DJF (Our impl.)	2.64	5.15	9.39
Ours-lite	2.55	4.82	8.52
Ours	**2.39**	**4.59**	**8.09**

Table 2: **Joint optical flow upsampling.** End-Point-Error (EPE) showing the improved performance compared with DJF [27].

	4×	8×	16×
Bilinear	0.465	0.901	1.628
DJF [27]	0.176	0.438	1.043
Ours	**0.105**	**0.256**	**0.592**

Outline:



HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs

Pravendra Singh Vinay Kumar Verma Piyush Rai Vinay P. Namboodiri
Department of Computer Science and Engineering, IIT Kanpur, India
{psingh, vkverma, piyush, vinaypn}@cse.iitk.ac.in

Highligt: Propose a novel convolution opeartion call HetConv

The proposed HetConv (Heterogeneous Kernel-Based Convolution) reduces the computation (FLOPs) and the number of parameters as compared to standard convolution operation while still maintaining representational efficiency.

Homogeneous: All the kernels are the **same** size

Heterogeneours: contains **different** sizes of kernels

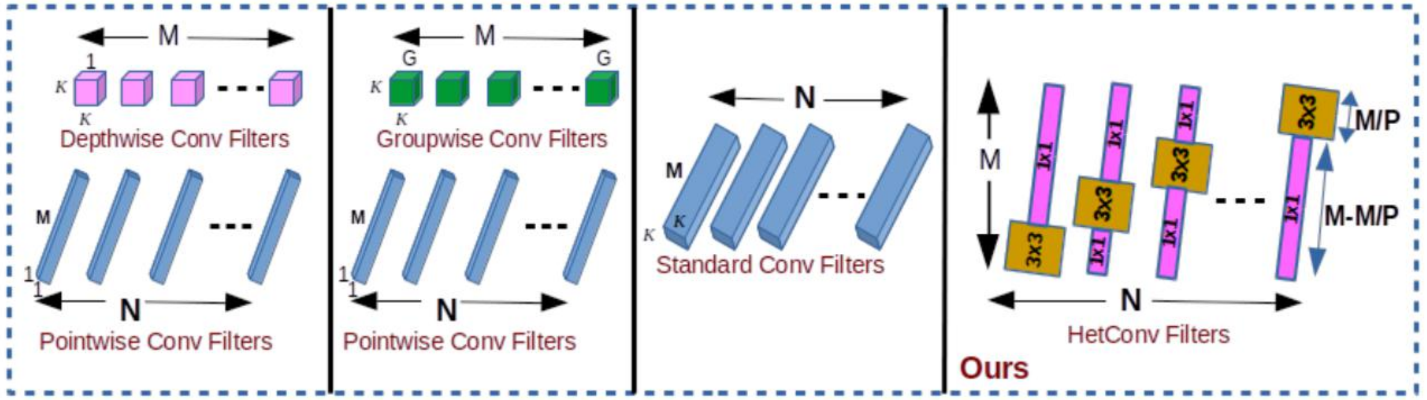


Figure 2. Comparison between the proposed Convolutional filters (HetConv) with other efficient convolutional filters. Our heterogeneous filters have zero latency while other (GWC+PWC or DWC+PWC) have a latency of one unit.

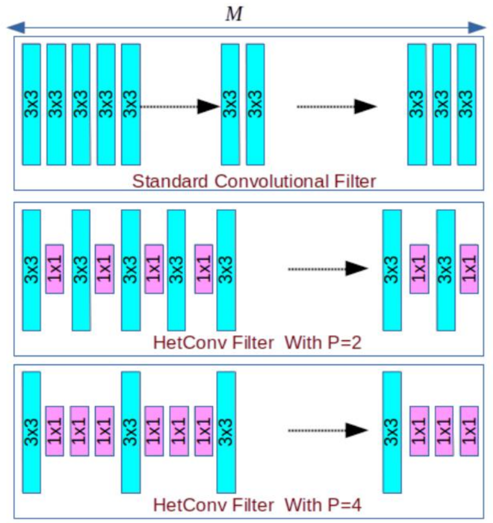


Figure 1. Difference between standard convolutional filter (homogeneous) and heterogeneous convolutional filter (HetConv). Here M is the input depth (number of input channels), and P is the part. Out of M kernels, M/P kernels will be of size 3×3 and remaining will be 1×1 kernels.

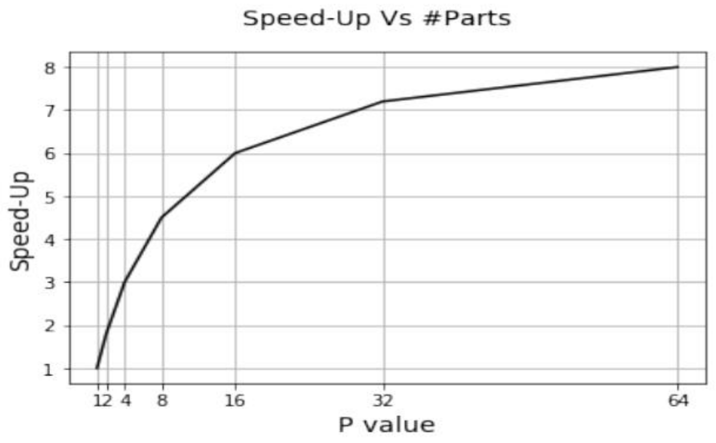


Figure 5. Speedup over standard convolution for different values of P for a HetConv Filter with 3×3 and 1×1 kernels.

4.2.3 Comparison with FLOPs compression methods

Method	Error%	FLOPs Reduced(%)
Li-pruned [21]	6.60	34.20
SBP [25]	7.50	56.52
SBPa [25]	9.00	68.35
AFP-E [3]	7.06	79.69
AFP-F [3]	7.13	81.39
VGG-16_P32 (Ours)	**6.27**	**85.54**
VGG-16_P64 (Ours)	**6.58**	**86.92**

Table 2. The table shows the comparison of our models with state-of-art model compression methods for VGG-16 architecture on the CIFAR-10 dataset.

4.5. VGG-16 on ImageNet

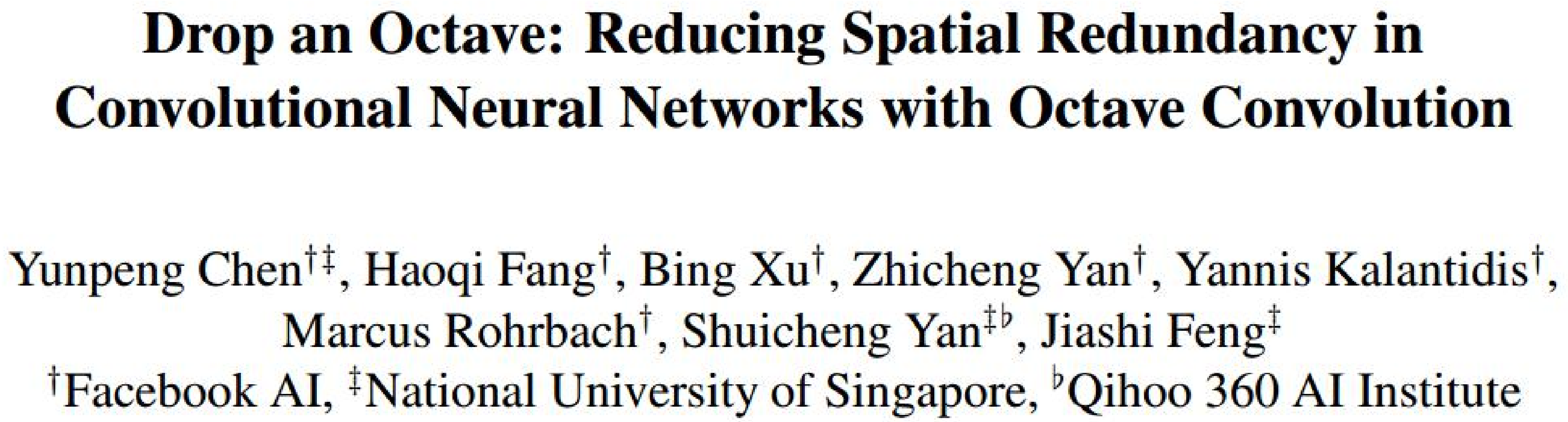
Method	Acc%(Top-1)	Acc%(Top-5)	FLOPs Reduced %
RNP (3X)[22]	–	87.57	66.67
ThiNet-70 [24]	69.8	89.53	69.04
CP 2X [11]	–	89.90	50.00
VGG-16_P1	71.3	90.2	–
VGG-16_P4	**71.2**	**90.2**	**65.8**

Table 5. Table shows the results for the VGG-16 on ImageNet [29]. Our model has no loss in accuracy as compare to state-of-art [11, 24] pruning approaches while significantly higher FLOPs reduction.

4.7. ResNet-50 on ImageNet

Method	Error (top-1)%	FLOPs	FLOPs Reduced(%)
ThiNet-70 [24]	27.90	–	36.8
NISP [41]	27.33	–	27.31
ResNet-50_P1	23.86	4.09G	–
ResNet-50_P4	**23.84**	2.85G	**30.32**

Table 7. Table shows the results for ResNet-50 on ImageNet [29]. Our model has no loss in accuracy as compare to state-of-art [24, 41] flop pruning approaches.



Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution

Yunpeng Chen[†‡], Haoqi Fang[†], Bing Xu[†], Zhicheng Yan[†], Yannis Kalantidis[†],
Marcus Rohrbach[†], Shuicheng Yan[‡♭], Jiashi Feng[‡]
[†]Facebook AI, [‡]National University of Singapore, [♭]Qihoo 360 AI Institute

Highlight: propose to factorize the mixed feature maps by their frequencies, and design a **novel Octave Convolution (OctConv) operation** to store and process feature maps that vary spatially "slower" at a lower spatial resolution reducing both memory and computation cost.

Outline:

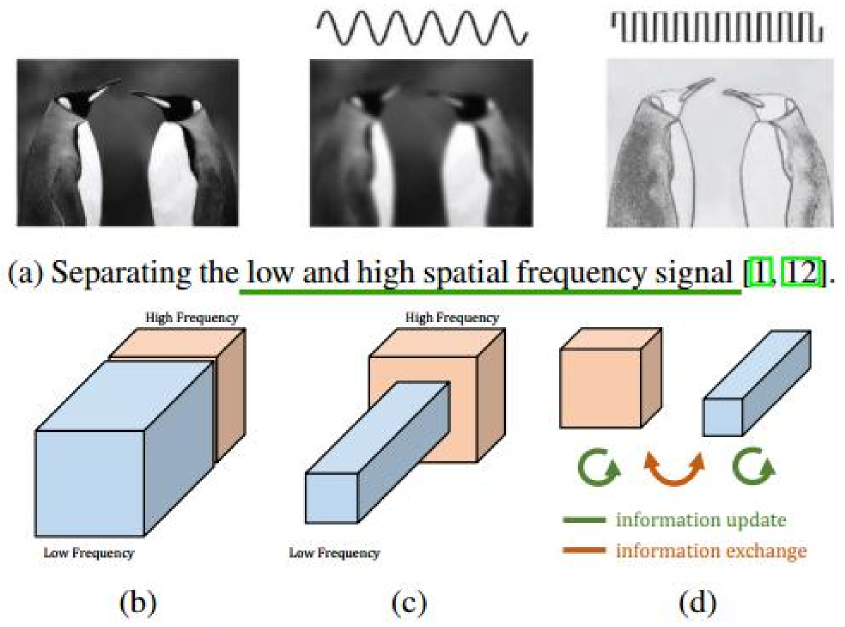
1. Adaptively Connected Neural Networks (CVPR 2019)

2. Pixel-Adaptive Convolutional Neural Networks (CVPR 2019)

3. HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs (CVPR 2019)

4. Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution (arxiv)

5. Res2Net: A New Multi-scale Backbone Architecture (arxiv)



(a) Separating the low and high spatial frequency signal [1, 12].

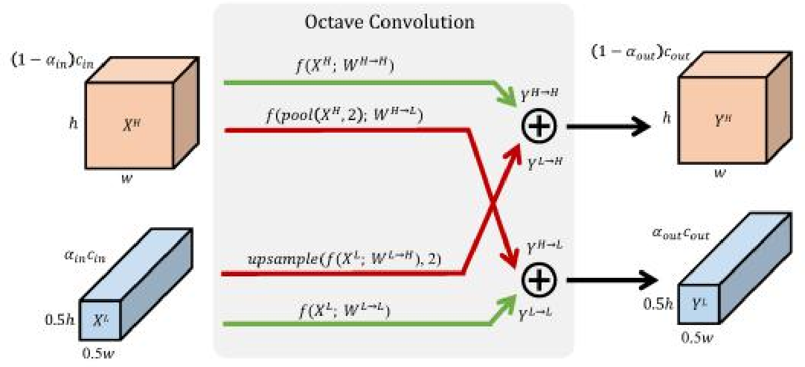
(b) (c) (d)

Figure 1: (a) Motivation. The spatial frequency model for vision [1, 12] shows that natural image can be decomposed into a low and a high spatial frequency part. (b) The output maps of a convolution layer can also be factorized and grouped by their spatial frequency. (c) The proposed multi-frequency feature representation stores the smoothly changing, low-frequency maps in a low-resolution tensor to reduce spatial redundancy. (d) The proposed Octave Convolution operates directly on this representation. It updates the information for each group and further enables information exchange between groups.

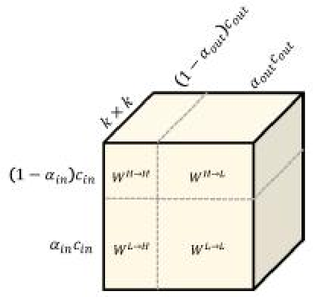
Motivation(or writing):

1. Factorize the convolution operation on input signal on high frequency part and low frequency part.

2. Perform information exchange between these parts as well as perfrom information individually.



(a) Detailed design of the Octave Convolution. Green arrows correspond to information updates while red arrows facilitate information exchange between the two frequencies.



(b) The Octave Convolution kernel. The $k \times k$ Octave Convolution kernel $W \in \mathbb{R}^{c_{in} \times c_{out} \times k \times k}$ is equivalent to the vanilla convolution kernel in the sense that the two have the exact same number of parameters.

Figure 2: Octave Convolution. We set $\alpha_{in} = \alpha_{out} = \alpha$ throughout the network, apart from the first and last OctConv of the network where $\alpha_{in} = 0, \alpha_{out} = \alpha$ and $\alpha_{in} = \alpha, \alpha_{out} = 0$, respectively.

$$Y_{p,q} = \sum_{i,j \in \mathcal{N}_k} W_{i+\frac{k-1}{2}, j+\frac{k-1}{2}}{}^{\top} X_{p+i,q+j}, \tag{1}$$

$$\begin{aligned} Y^H_{p,q} =& Y^{H\to H}_{p,q} + Y^{L\to H}_{p,q} \\ =& \sum_{i,j \in \mathcal{N}_k} W^{H\to H}_{i+\frac{k-1}{2}, j+\frac{k-1}{2}}{}^{\top} X^H_{p+i,q+j} \\ &+ \sum_{i,j \in \mathcal{N}_k} W^{L\to H}_{i+\frac{k-1}{2}, j+\frac{k-1}{2}}{}^{\top} X^L_{(\lfloor \frac{p}{2} \rfloor + i),(\lfloor \frac{q}{2} \rfloor + j)}, \end{aligned} \tag{2}$$

Two line above in (a)

$$\begin{aligned} Y^L_{p,q} =& Y^{L\to L}_{p,q} + Y^{H\to L}_{p,q} \\ =& \sum_{i,j \in \mathcal{N}_k} W^{L\to L}_{i+\frac{k-1}{2}, j+\frac{k-1}{2}}{}^{\top} X^L_{p+i,q+j} \\ &+ \sum_{i,j \in \mathcal{N}_k} W^{H\to L}_{i+\frac{k-1}{2}, j+\frac{k-1}{2}}{}^{\top} X^H_{(2*p+0.5+i),(2*q+0.5+j)}, \end{aligned} \tag{3}$$

Two line below in (a)

$$\begin{aligned} Y^H =& f(X^H; W^{H\to H}) + \texttt{upsample}(f(X^L; W^{L\to H}), 2) \\ Y^L =& f(X^L; W^{L\to L}) + f(\texttt{pool}(X^H, 2); W^{H\to L})), \end{aligned} \tag{4}$$

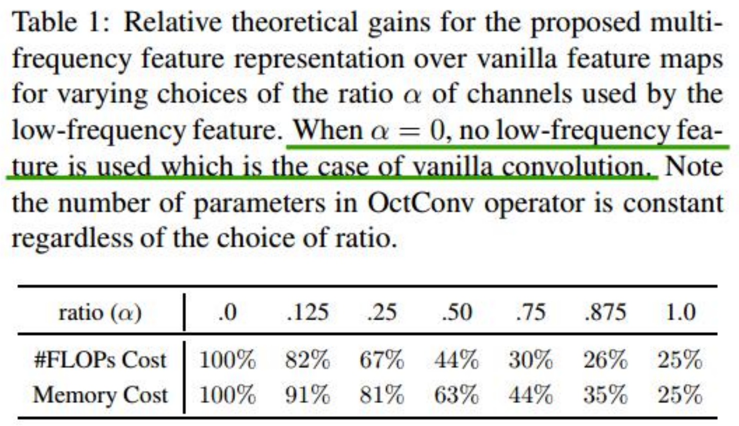


Table 1: Relative theoretical gains for the proposed multi-frequency feature representation over vanilla feature maps for varying choices of the ratio α of channels used by the low-frequency feature. When $\alpha = 0$, no low-frequency feature is used which is the case of vanilla convolution. Note the number of parameters in OctConv operator is constant regardless of the choice of ratio.

ratio (α)	.0	.125	.25	.50	.75	.875	1.0
#FLOPs Cost	100%	82%	67%	44%	30%	26%	25%
Memory Cost	100%	91%	81%	63%	44%	35%	25%

alpha means the ratio of low-frequency part of one covolution.

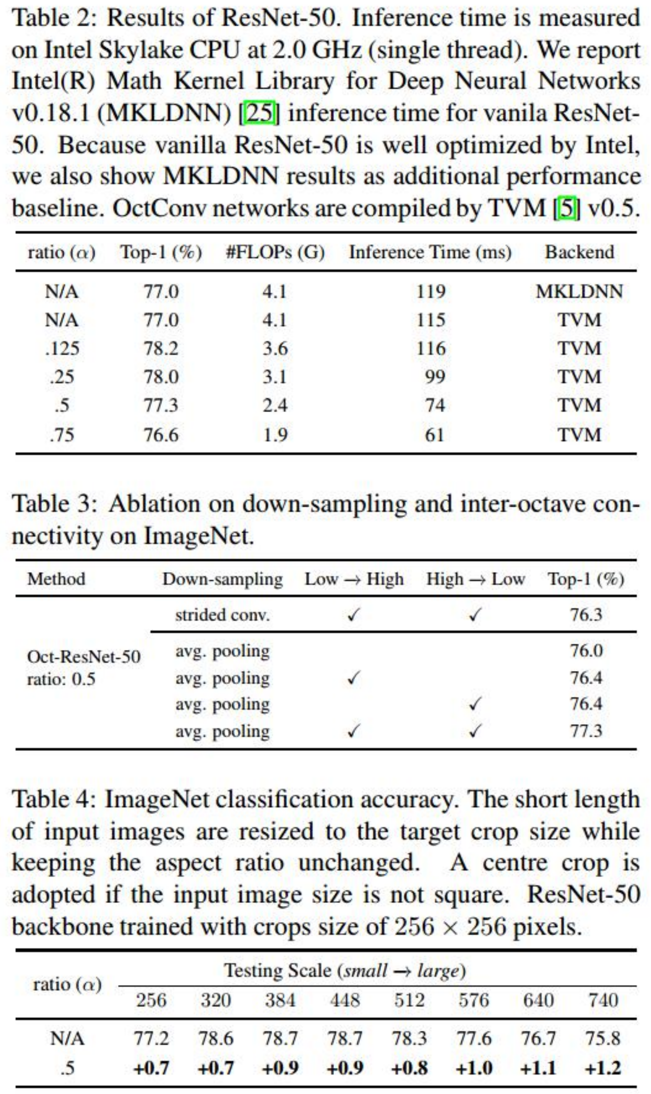


Table 2: Results of ResNet-50. Inference time is measured on Intel Skylake CPU at 2.0 GHz (single thread). We report Intel(R) Math Kernel Library for Deep Neural Networks v0.18.1 (MKLDNN) [25] inference time for vanila ResNet-50. Because vanilla ResNet-50 is well optimized by Intel, we also show MKLDNN results as additional performance baseline. OctConv networks are compiled by TVM [5] v0.5.

ratio (α)	Top-1 (%)	#FLOPs (G)	Inference Time (ms)	Backend
N/A	77.0	4.1	119	MKLDNN
N/A	77.0	4.1	115	TVM
.125	78.2	3.6	116	TVM
.25	78.0	3.1	99	TVM
.5	77.3	2.4	74	TVM
.75	76.6	1.9	61	TVM

Table 3: Ablation on down-sampling and inter-octave connectivity on ImageNet.

Method	Down-sampling	Low → High	High → Low	Top-1 (%)
Oct-ResNet-50 ratio: 0.5	strided conv.	✓	✓	76.3
	avg. pooling			76.0
	avg. pooling	✓		76.4
	avg. pooling		✓	76.4
	avg. pooling	✓	✓	77.3

Table 4: ImageNet classification accuracy. The short length of input images are resized to the target crop size while keeping the aspect ratio unchanged. A centre crop is adopted if the input image size is not square. ResNet-50 backbone trained with crops size of 256×256 pixels.

ratio (α)	Testing Scale (*small* → *large*)							
	256	320	384	448	512	576	640	740
N/A	77.2	78.6	78.7	78.7	78.3	77.6	76.7	75.8
.5	**+0.7**	**+0.7**	**+0.9**	**+0.9**	**+0.8**	**+1.0**	**+1.1**	**+1.2**

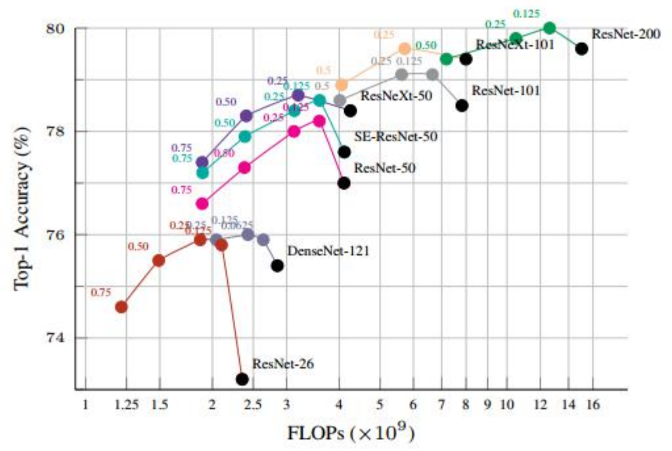


Figure 4: Ablation study results on ImageNet. OctConv-equipped models are more efficient and accurate than baseline models. Markers in black in each line denote the corresponding baseline models without OctConv. The colored numbers are the ratio α. Numbers in X axis denote FLOPs in logarithmic scale.

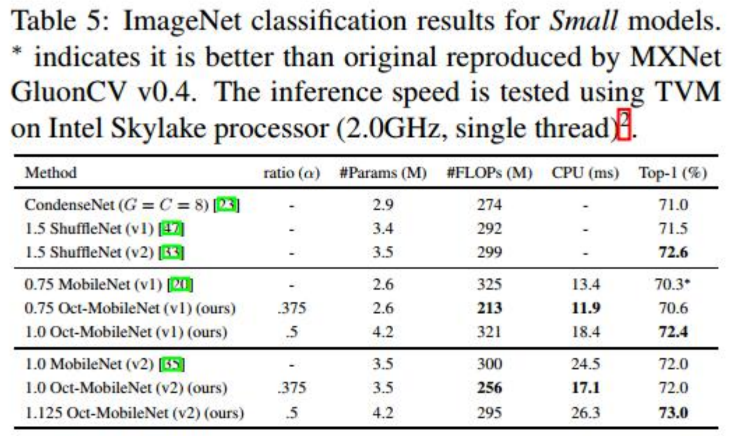


Table 5: ImageNet classification results for *Small* models. * indicates it is better than original reproduced by MXNet GluonCV v0.4. The inference speed is tested using TVM on Intel Skylake processor (2.0GHz, single thread)[2].

Method	ratio (α)	#Params (M)	#FLOPs (M)	CPU (ms)	Top-1 (%)
CondenseNet ($G = C = 8$) [23]	-	2.9	274	-	71.0
1.5 ShuffleNet (v1) [47]	-	3.4	292	-	71.5
1.5 ShuffleNet (v2) [33]	-	3.5	299	-	**72.6**
0.75 MobileNet (v1) [20]	-	2.6	325	13.4	70.3*
0.75 Oct-MobileNet (v1) (ours)	.375	2.6	**213**	**11.9**	70.6
1.0 Oct-MobileNet (v1) (ours)	.5	4.2	321	18.4	**72.4**
1.0 MobileNet (v2) [35]	-	3.5	300	24.5	72.0
1.0 Oct-MobileNet (v2) (ours)	.375	3.5	**256**	**17.1**	72.0
1.125 Oct-MobileNet (v2) (ours)	.5	4.2	295	26.3	**73.0**

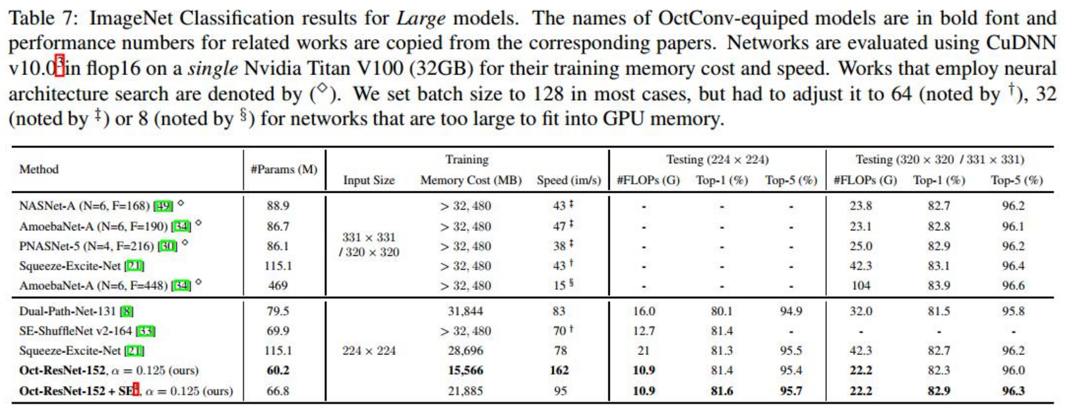


Table 7: ImageNet Classification results for *Large* models. The names of OctConv-equiped models are in bold font and performance numbers for related works are copied from the corresponding papers. Networks are evaluated using CuDNN v10.0[3] in flop16 on a *single* Nvidia Titan V100 (32GB) for their training memory cost and speed. Works that employ neural architecture search are denoted by ($^\diamond$). We set batch size to 128 in most cases, but had to adjust it to 64 (noted by †), 32 (noted by ‡) or 8 (noted by §) for networks that are too large to fit into GPU memory.

Method	#Params (M)	Training			Testing (224 × 224)			Testing (320 × 320 / 331 × 331)		
		Input Size	Memory Cost (MB)	Speed (im/s)	#FLOPs (G)	Top-1 (%)	Top-5 (%)	#FLOPs (G)	Top-1 (%)	Top-5 (%)
NASNet-A (N=6, F=168) [49] $^\diamond$	88.9	331 × 331 / 320 × 320	> 32, 480	43 ‡	-	-	-	23.8	82.7	96.2
AmoebaNet-A (N=6, F=190) [34] $^\diamond$	86.7		> 32, 480	47 ‡	-	-	-	23.1	82.8	96.1
PNASNet-5 (N=4, F=216) [30] $^\diamond$	86.1		> 32, 480	38 ‡	-	-	-	25.0	82.9	96.2
Squeeze-Excite-Net [21]	115.1		> 32, 480	43 †	-	-	-	42.3	83.1	96.4
AmoebaNet-A (N=6, F=448) [34] $^\diamond$	469		> 32, 480	15 §	-	-	-	104	83.9	96.6
Dual-Path-Net-131 [8]	79.5	224 × 224	31,844	83	16.0	80.1	94.9	32.0	81.5	95.8
SE-ShuffleNet v2-164 [33]	69.9		> 32, 480	70 †	12.7	81.4	-	-	-	-
Squeeze-Excite-Net [21]	115.1		28,696	78	21	81.3	95.5	42.3	82.7	96.2
Oct-ResNet-152, $\alpha = 0.125$ (ours)	**60.2**		**15,566**	**162**	**10.9**	81.4	95.4	**22.2**	82.3	96.0
Oct-ResNet-152 + SE[4], $\alpha = 0.125$ (ours)	66.8		21,885	95	**10.9**	**81.6**	**95.7**	**22.2**	**82.9**	**96.3**

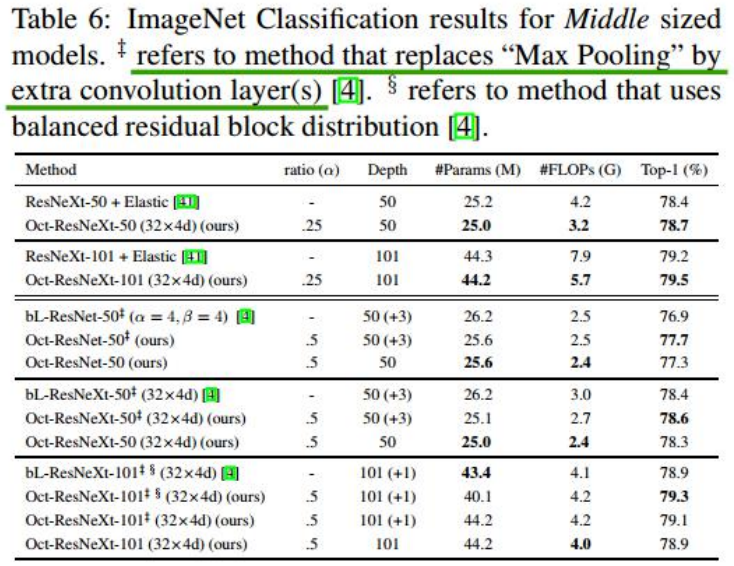


Table 6: ImageNet Classification results for *Middle* sized models. ‡ refers to method that replaces "Max Pooling" by extra convolution layer(s) [4]. § refers to method that uses balanced residual block distribution [4].

Method	ratio (α)	Depth	#Params (M)	#FLOPs (G)	Top-1 (%)
ResNeXt-50 + Elastic [41]	-	50	25.2	4.2	78.4
Oct-ResNeXt-50 (32×4d) (ours)	.25	50	**25.0**	**3.2**	**78.7**
ResNeXt-101 + Elastic [41]	-	101	44.3	7.9	79.2
Oct-ResNeXt-101 (32×4d) (ours)	.25	101	**44.2**	**5.7**	**79.5**
bL-ResNet-50‡ ($\alpha = 4, \beta = 4$) [4]	-	50 (+3)	26.2	2.5	76.9
Oct-ResNet-50‡ (ours)	.5	50 (+3)	25.6	2.5	**77.7**
Oct-ResNet-50 (ours)	.5	50	**25.6**	**2.4**	77.3
bL-ResNeXt-50‡ (32×4d) [4]	-	50 (+3)	26.2	3.0	78.4
Oct-ResNeXt-50‡ (32×4d) (ours)	.5	50 (+3)	25.1	2.7	**78.6**
Oct-ResNeXt-50 (32×4d) (ours)	.5	50	**25.0**	**2.4**	78.3
bL-ResNeXt-101$^{\ddagger\,\S}$ (32×4d) [4]	-	101 (+1)	**43.4**	4.1	78.9
Oct-ResNeXt-101$^{\ddagger\,\S}$ (32×4d) (ours)	.5	101 (+1)	40.1	4.2	**79.3**
Oct-ResNeXt-101‡ (32×4d) (ours)	.5	101 (+1)	44.2	4.2	79.1
Oct-ResNeXt-101 (32×4d) (ours)	.5	101	44.2	**4.0**	78.9

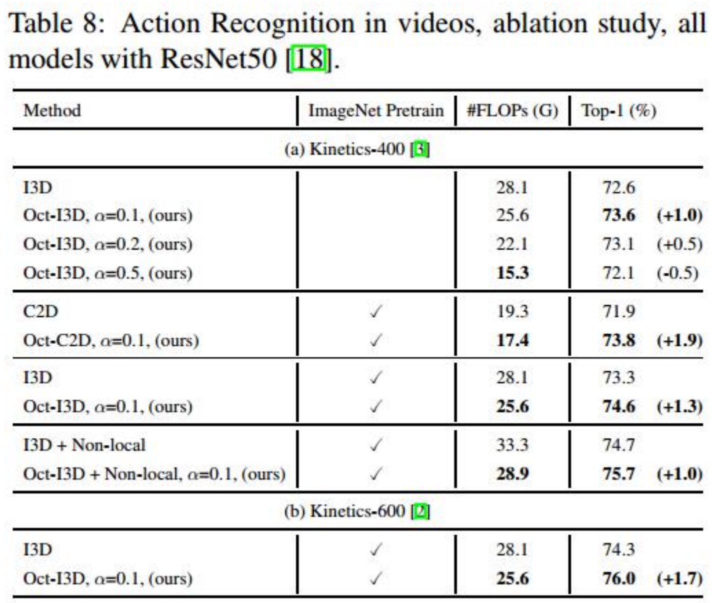


Table 8: Action Recognition in videos, ablation study, all models with ResNet50 [18].

Method	ImageNet Pretrain	#FLOPs (G)	Top-1 (%)
(a) Kinetics-400 [3]			
I3D		28.1	72.6
Oct-I3D, α=0.1, (ours)		25.6	**73.6** (+1.0)
Oct-I3D, α=0.2, (ours)		22.1	73.1 (+0.5)
Oct-I3D, α=0.5, (ours)		**15.3**	72.1 (-0.5)
C2D	✓	19.3	71.9
Oct-C2D, α=0.1, (ours)	✓	**17.4**	**73.8** (+1.9)
I3D	✓	28.1	73.3
Oct-I3D, α=0.1, (ours)	✓	**25.6**	**74.6** (+1.3)
I3D + Non-local	✓	33.3	74.7
Oct-I3D + Non-local, α=0.1, (ours)	✓	**28.9**	**75.7** (+1.0)
(b) Kinetics-600 [2]			
I3D	✓	28.1	74.3
Oct-I3D, α=0.1, (ours)	✓	**25.6**	**76.0** (+1.7)

Lots of experiments on recongition task.

Outline:

Res2Net: A New Multi-scale Backbone Architecture

Shang-Hua Gao*, Ming-Ming Cheng*, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr

HighLight:
Most existing methods represent the multi-scale features in a layerwise manner. In this paper, we propose a novel building block for CNNs, namely Res2Net, **by constructing hierarchical residual-like connections within one single residual block.**

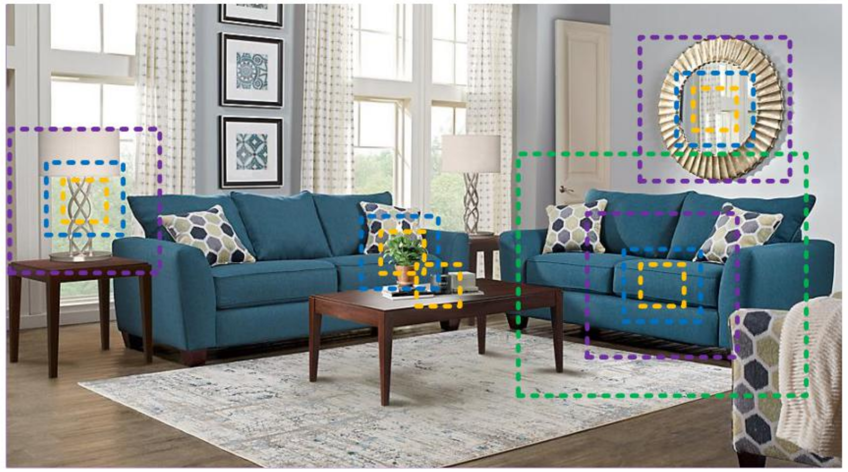


Fig. 1: Multi-scale representations are essential for various vision tasks, such as perceiving boundaries, regions, and semantic categories of the target objects. Even for the simplest recognition tasks, perceiving information from very different scales is essential to understand parts, objects (*e.g.* , sofa, table, and cup in this example), and their surrounding context (*e.g.* , 'on the table' context contributes to recognizing the black blob).

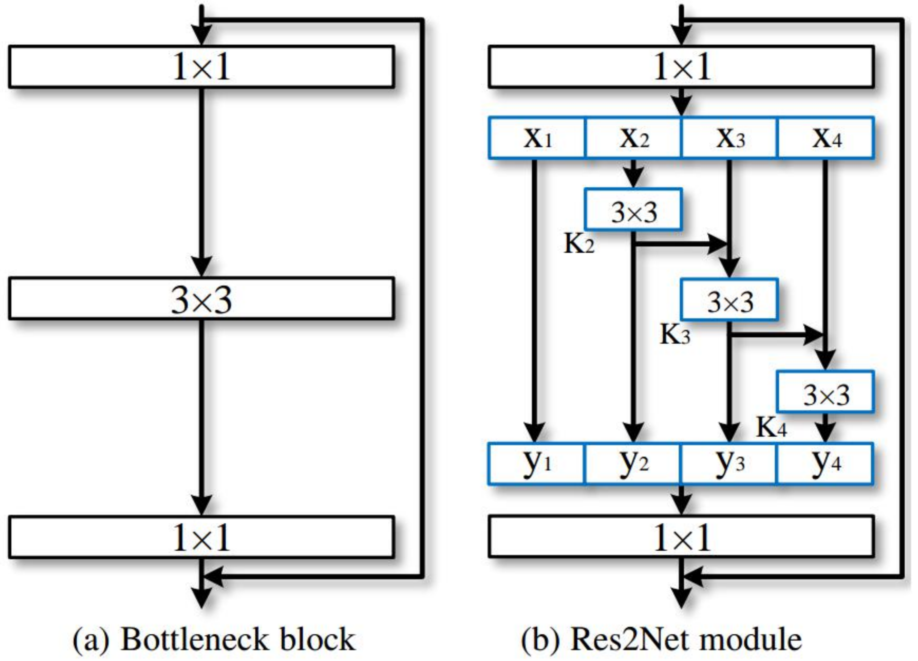


Fig. 2: Comparison between the bottleneck block and the proposed Res2Net module (the scale dimension $s = 4$).

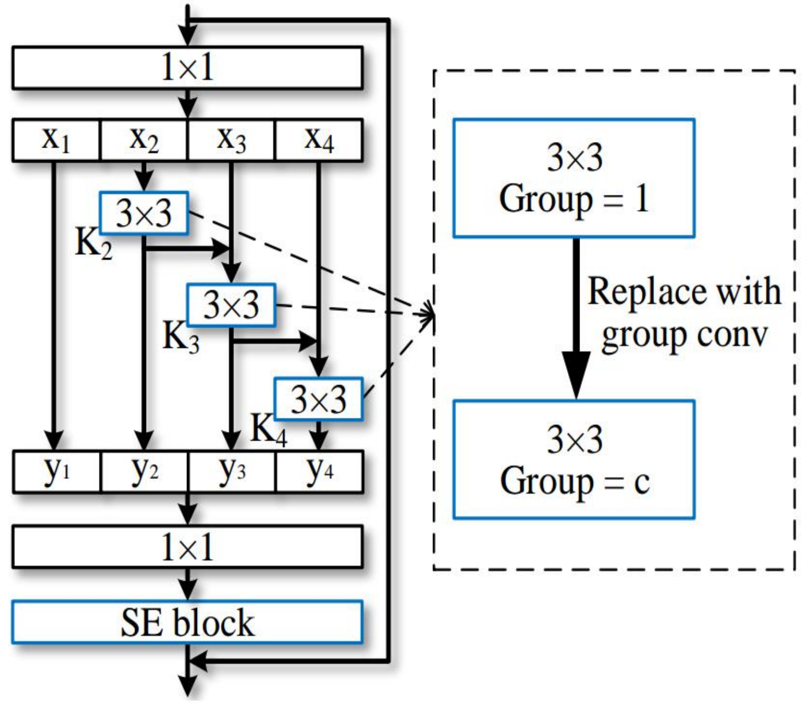


Fig. 3: The Res2Net module can be integrated with the dimension cardinality [43] (replace conv with group conv) and SE [19] blocks.

TABLE 1: Top-1 and Top-5 test error on the ImageNet dataset.

	top-1 err. (%)	top-5 err. (%)
ResNet-50 [17]	23.85	7.13
Res2Net-50	**22.01**	**6.15**
InceptionV3 [40]	22.55	6.44
Res2Net-50-299	**21.41**	**5.88**
ResNeXt-50 [43]	22.61	6.50
Res2NeXt-50	**21.76**	**6.09**
DLA-60 [47]	23.32	6.60
Res2Net-DLA-60	**21.53**	**5.80**
DLA-X-60 [47]	22.19	6.13
Res2NeXt-DLA-60	**21.55**	**5.86**
SENet-50 [19]	23.24	6.69
SE-Res2Net-50	**21.56**	**5.94**

TABLE 2: Top-1 and Top-5 test error (%) of deeper networks on the ImageNet dataset.

	top-1 err.	top-5 err.	Memory
DenseNet-161 [20]	22.35	6.20	268M
ResNet-101 [17]	22.63	6.44	162M
Res2Net-101	**20.81**	**5.57**	179M

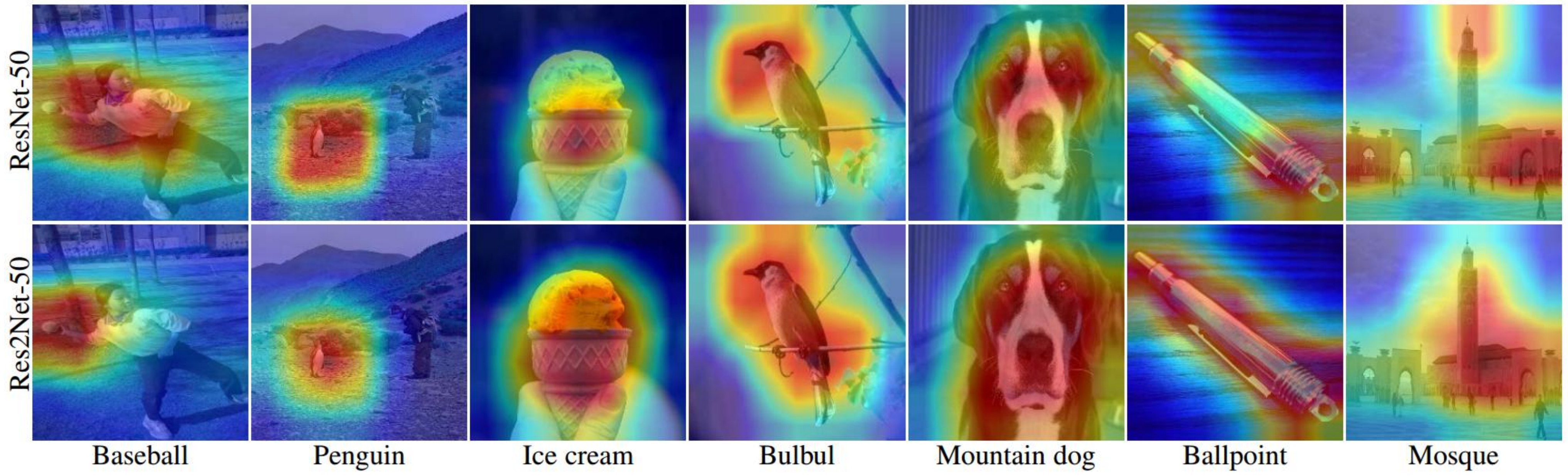


Fig. 5: Visualization of class activation mapping [35], using ResNet-50 and Res2Net-50 as backbone networks.

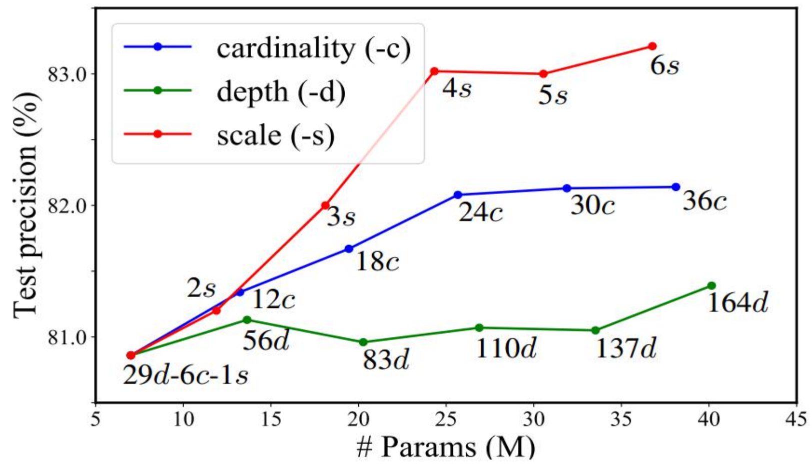


Fig. 4: Test precision on the CIFAR-100 dataset with regard to the model size, by changing cardinality (ResNeXt-29), depth (ResNeXt), and scale (Res2Net-29).

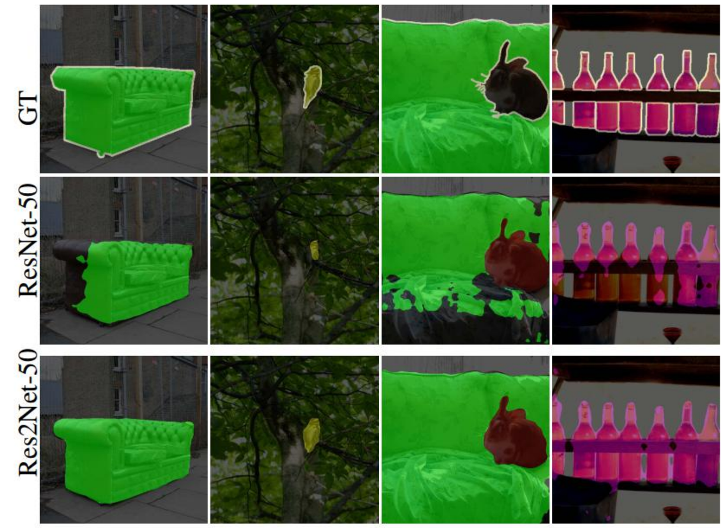


Fig. 6: Visualization of semantic segmentation results [5], using ResNet-101 and Res2Net-101 as backbone networks.

Summary:

1. Adaptively Connected Neural Networks (CVPR 2019)
2. Pixel-Adaptive Convolutional Neural Networks (CVPR 2019)
3. HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs (CVPR 2019)
4. Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution (arxiv)
5. Res2Net: A New Multi-scale Backbone Architecture (arxiv)

1,3,4,5: learning multi-scale feature repersentation rather than single scale in block level.

2: more like data-driven filter for different data as input.

Thanks for Watching!

Q & A