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

Yunpeng Chen †‡ , Haoqi Fan † , Bing Xu † , Zhicheng Yan † , Yannis Kalantidis † , Marcus Rohrbach † , Shuicheng Yan ‡ ♭ , Jiashi Feng ‡

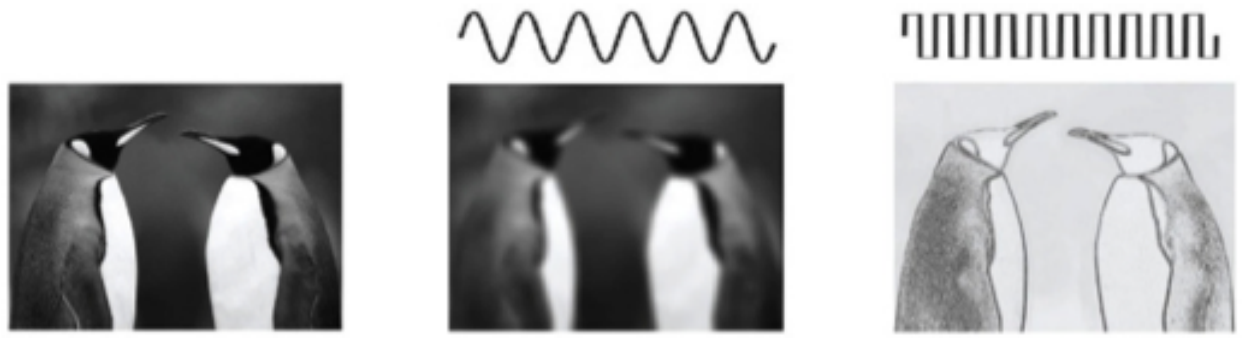
† Facebook AI, ‡ National University of Singapore, ♭ Yitu Technology

Presented by: Huidong Xie

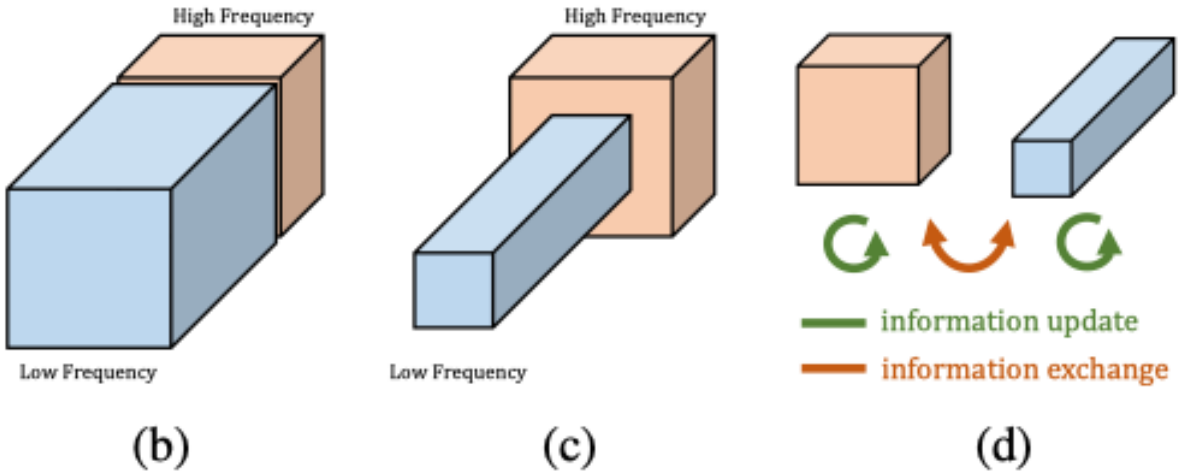
Key points

- They proposed Octave Convolution to replace vanilla convolutional layers.
- Octave CNN can be easily deployed in existing CNNs.
- No need to change network structure.
- No need to fine-tune hyper-parameters.
- Octave CNN is designed to reduce spatial dimension of feature maps.
- Consistently improved performance in popular CNNs for image and video recognitions. (ResNet, DenseNet, ect.)

Motivations



(a) Separating the low and high spatial frequency signal [1, 10].

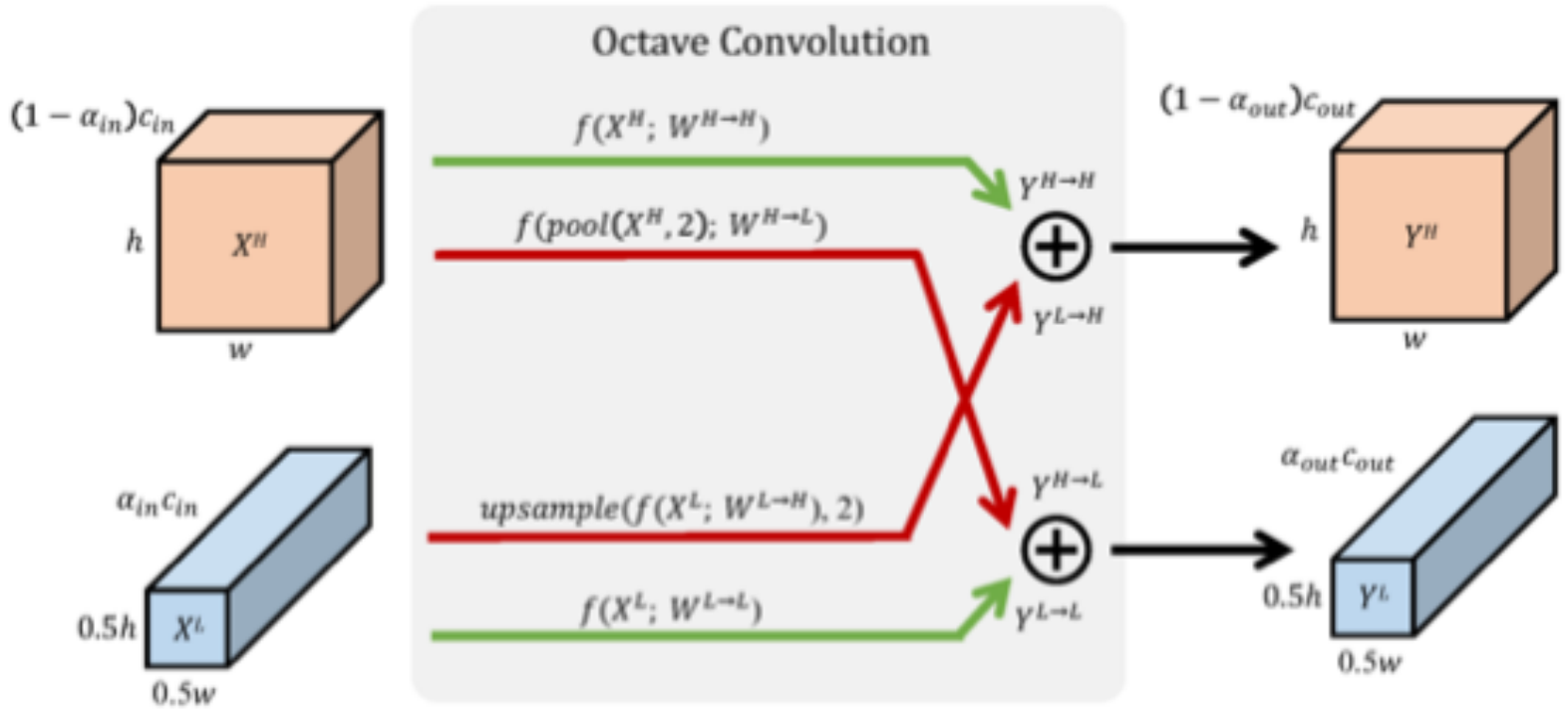


(b) (c) (d)

Related works

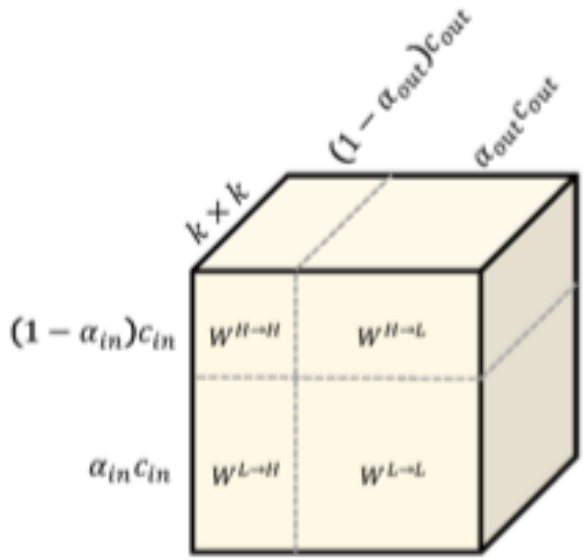
- ResNet and DenseNet improve the network topology by adding shrotcuts.
- ResNeXt reduces redundancy in inter-channel connectivity by using group convolution.
- Some pruning methods designed to reduce model parameters.
- Some other methods designed to find the optimal network topology for a given task.
- None of them focuses on the redundancy on the spatial dimension of feature-maps. Also, those previous methods need hyper-parameter tuning.

Octave convolution



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

Octave convolution



(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.

Octave convolution

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

Efficiency analysis

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%

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.

Experiments

- For image classification, they used the ImageNet. A set of most popular CNNs are used to test the effectiveness of Octave CNNs.
- For video action recognition. They used the Kinetics-400 and Kinetics-600 datasets for human action recognition.

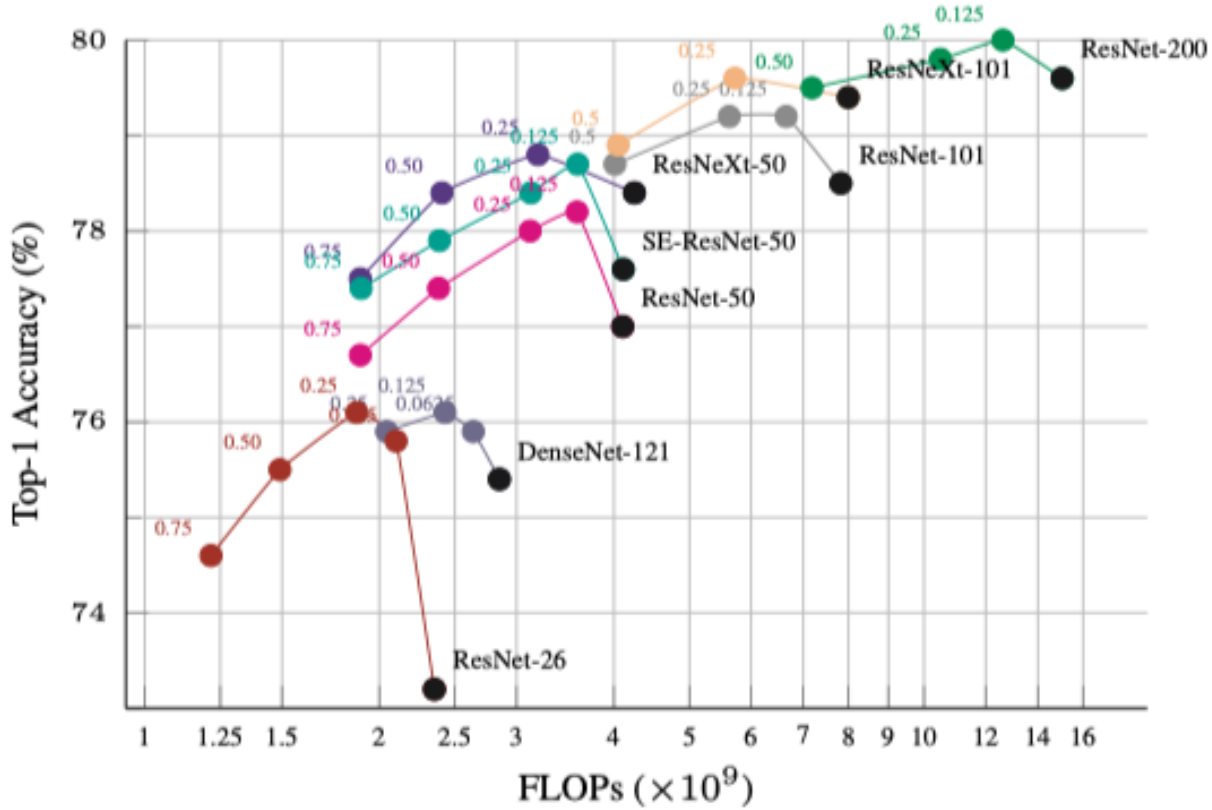


Figure 3: Ablation study results on ImageNet. OctConv-equipped models are more efficient and accurate than baseline models. Markers in black in each line denote the corresponding baseline models without OctConv. The colored numbers are the ratio α. Numbers in X axis denote FLOPs in logarithmic scale.

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	77.4	2.4	74	TVM
.75	76.7	1.9	61	TVM

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) [23] 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.

Frequency analysis

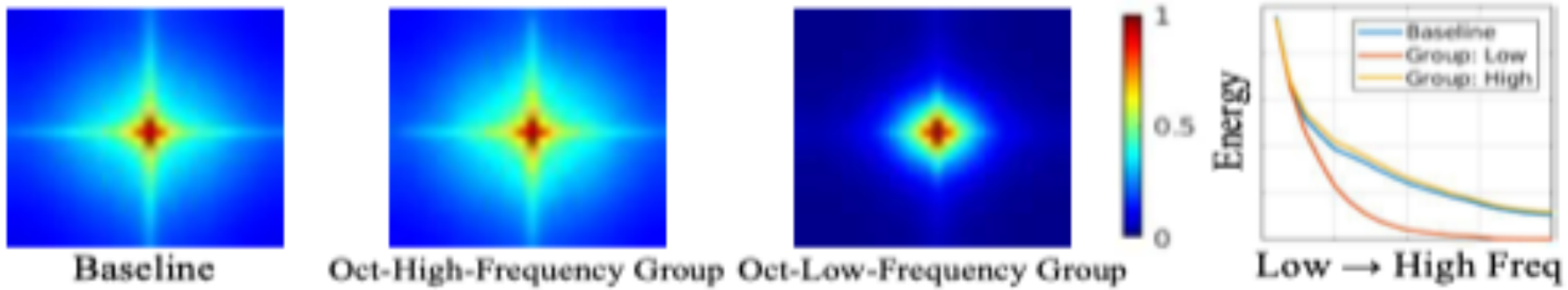


Figure 4: Frequency analysis for activation maps in different groups. 'Baseline' refers to vanilla ResNet. 10k activation maps are sampled from ResNet-101(Res3).

Method	ratio (α)	#Params (M)	#FLOPs (M)	CPU (ms)	Top-1 (%)
CondenseNet ($G = C = 8$) [21]	-	2.9	274	-	71.0
1.5 ShuffleNet (v1) [49]	-	3.4	292	-	71.5
1.5 ShuffleNet (v2) [32]	-	3.5	299	-	**72.6**
0.75 MobileNet (v1) [18]	-	2.6	325	13.4	70.3*
0.75 Oct-MobileNet (v1) (ours)	.375	2.6	**213**	**11.9**	70.5
1.0 Oct-MobileNet (v1) (ours)	.5	4.2	321	18.4	**72.5**
1.0 MobileNet (v2) [34]	-	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**

Method	ratio (α)	Depth	#Params (M)	#FLOPs (G)	Top-1 (%)
R-MG-34 [25]	-	34	32.9	5.8	75.5
Oct-ResNet-26 (ours)	.25	26	**16.0**	**1.9**	76.1
Oct-ResNet-50 (ours)	.5	50	25.6	2.4	**77.4**
ResNet-50 + GloRe [8] (+3 blocks Res4)	-	50	30.5	5.2	78.4
Oct-ResNet-50 (ours) + GloRe [8] (+3 blocks Res4)	.5	50	30.5	**3.1**	**78.8**
ResNeXt-50 + Elastic [43]	-	50	25.2	4.2	78.4
Oct-ResNeXt-50 (32×4d) (ours)	.25	50	**25.0**	**3.2**	**78.8**
ResNeXt-101 + Elastic [43]	-	101	44.3	7.9	79.2
Oct-ResNeXt-101 (32×4d) (ours)	.25	101	**44.2**	**5.7**	**79.6**
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	**77.8**
Oct-ResNet-50 (ours)	.5	50	**25.6**	**2.4**	77.4
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**	78.4
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	**79.4**
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

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) [51] ◇	88.9	331 × 331 / 320 × 320	> 32,480	43 ‡	-	-	-	23.8	82.7	96.2
AmoebaNet-A (N=6, F=190) [33] ◇	86.7		> 32,480	47 ‡	-	-	-	23.1	82.8	96.1
PNASNet-5 (N=4, F=216) [29] ◇	86.1		> 32,480	38 ‡	-	-	-	25.0	82.9	96.2
Squeeze-Excite-Net [19]	115.1		> 32,480	43 †	-	-	-	42.3	83.1	96.4
AmoebaNet-A (N=6, F=448) [33] ◇	469		> 32,480	15 §	-	-	-	104	83.9	96.6
Dual-Path-Net-131 [7]	79.5	224 × 224	31,844	83	16.0	80.1	94.9	32.0	81.5	95.8
SE-ShuffleNet v2-164 [32]	69.9		> 32,480	70 †	12.7	81.4	-	-	-	-
Squeeze-Excite-Net [19]	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[3], $\alpha = 0.125$ (ours)	66.8		21,885	95	**10.9**	**81.6**	**95.7**	**22.2**	**82.9**	**96.3**

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)**
SlowFast-R50 [12]		27.6 [5]	75.6	
Oct-SlowFast-R50, α=0.1, (ours)		**24.5**	**76.2**	**(+0.6)**
Oct-SlowFast-R50, α=0.2, (ours)		22.9	75.8	(+0.2)
(b) Kinetics-600 [2]				
I3D	✓	28.1	74.3	
Oct-I3D, α=0.1, (ours)	✓	**25.6**	**76.0**	**(+1.7)**