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

Abstract

In natural images, information is conveyed at different frequencies where higher frequencies are usually encoded with fine details and lower frequencies are usually encoded with global structures. Similarly, the output feature maps of a convolution layer can also be seen as a mixture of information at different frequencies. In this work, we 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. Unlike existing multi-scale methods, OctConv is formulated as a single, generic, plug-and-play convolutional unit that can be used as a direct replacement of (vanilla) convolutions without any adjustments in the network architecture. It is also orthogonal and complementary to methods that suggest better topologies or reduce channel-wise redundancy like group or depth-wise convolutions. We experimentally show that by simply replacing convolutions with OctConv, we can consistently boost accuracy for both image and video recognition tasks, while reducing memory and computational cost. An OctConv-equipped ResNet-152 can achieve 82.9% top-1 classification accuracy on ImageNet with merely 22.2 GFLOPs.

1. Introduction

Convolutional Neural Networks (CNNs) have achieved remarkable success in many computer vision tasks [18, 17, 42] and their efficiency keeps increasing with recent efforts to reduce the inherent redundancy in dense model parameters [16, 32, 40] and in the channel dimension of feature maps [45, 20, 7, 10]. However, substantial redundancy also exists in the spatial dimension of the feature maps produced by CNNs, where each location stores its own feature descriptor independently, while ignoring common information between adjacent locations that could be stored and processed together.

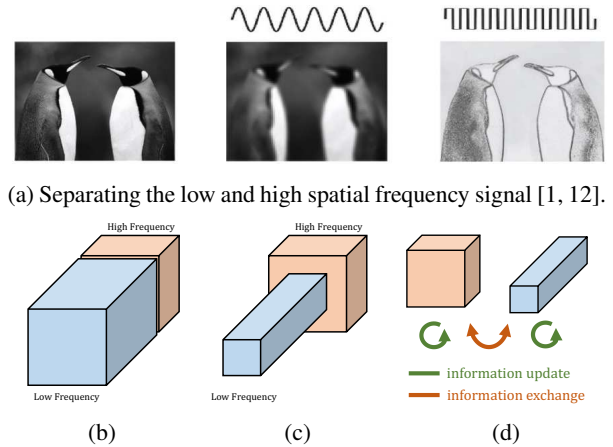(a) Separating the low and high spatial frequency signal [1, 12].

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.

As shown in Figure 1(a), a natural image can be decomposed into a low spatial frequency component that describes the smoothly changing structure and a high spatial frequency component that describes the rapidly changing fine details [1, 12]. Similarly, we argue that the output feature maps of a convolution layer can also be decomposed into features of different spatial frequencies and propose a novel multi-frequency feature representation which stores high- and low-frequency feature maps into different groups as shown in Figure 1(b). Thus, the spatial resolution of the low-frequency group can be safely reduced by sharing information between neighboring locations to reduce spatial redundancy as shown in Figure 1(c). To accommodate the novel feature representation, we generalize the vanilla con-

volution, and propose *Octave Convolution (OctConv)* which takes in feature maps containing tensors of two frequencies one octave apart, and extracts information directly from the low-frequency maps without the need of decoding it back to the high-frequency as shown in Figure 1(d). As a replacement of vanilla convolution, OctConv consumes substantially less memory and computational resources. In addition, OctConv processes low-frequency information with corresponding (low-frequency) convolutions and effectively enlarges the receptive field in the original pixel space and thus can improve recognition performance.

We design the OctConv in a generic way, making it a plug-and-play replacement for the vanilla convolution. Since OctConv mainly focuses on processing feature maps at multiple spatial frequencies and reducing their spatial redundancy, it is orthogonal and complementary to existing methods that focus on building better CNN topology [24, 38, 36, 34, 30], reducing channel-wise redundancy in convolutional feature maps [45, 10, 35, 33, 23] and reducing redundancy in dense model parameters [40, 16, 32]. We further discuss the integration of OctConv into the group, depth-wise, and 3D convolution cases. Moreover, different from methods that exploit multi-scale information [4, 41, 14], OctConv can be easily deployed as a plug-and-play unit to replace convolution, without the need of changing network architectures or requiring hyper-parameters tuning.

Our experiments demonstrate that by simply replacing the vanilla convolution with OctConv, we can consistently improve the performance of popular 2D CNN backbones including ResNet [18, 19], ResNeXt [45], DenseNet [24], MobileNet [20, 35] and SE-Net [21] on 2D image recognition on ImageNet [13], as well as 3D CNN backbones C2D [42] and I3D [42] on video action recognition on Kinetics [26, 3, 2]. The OctConv-equipped Oct-ResNet-152 can match or outperform state-of-the-art manually designed networks [33, 21] at lower memory and computational cost. Our contributions can be summarized as follows:

- We propose to factorize convolutional feature maps into two groups at different spatial frequencies and process them with different convolutions at their corresponding frequency, one octave apart. As the resolution for low frequency maps can be reduced, this saves both storage and computation. This also helps each layer gain a larger receptive field to capture more contextual information.

- We design a plug-and-play operation named OctConv to replace the vanilla convolution for operating on the new feature representation directly and reducing spatial redundancy. Importantly, OctConv is fast in practice and achieves a speedup close to the theoretical limit.

- We extensively study the properties of the proposed Oct-Conv on a variety of backbone CNNs for image and video tasks and achieve significant performance gain even comparable to the best AutoML networks.

2. Related Work

Improving the efficiency of CNNs. Ever since the pioneering work on AlexNet [27] and VGG [36] which achieve astonishing results by stacking a set of convolution layers, researchers have made substantial efforts to improve the efficiency of CNNs. ResNet [18, 19] and DenseNet [24] improve the network topology by adding shortcut connections to early layers to enhance the feature reusing mechanism and alleviate optimization difficulties. ResNeXt [45] and ShuffleNet [47] use sparsely connected group convolutions to reduce redundancy in inter-channel connectivity, making it feasible to adopt deeper or wider networks under the same computational budget. Xception [10] and MobileNet [20, 35] adopt depth-wise convolutions that further reduce the connection density. Besides these manually designed networks, researchers also tried to atomically find the best network topology for a given task. NAS [49], PNAS [30] and AmoebaNet [34] successfully discovered typologies that perform better than manually designed networks. Another stream of work focuses on reducing the redundancy in the model parameters. DSD [16] reduces the redundancy in model connections by pruning connections of low weights. ThiNet [32] prunes convolutional filters based on statistics computed from its next layer. However, all of these methods ignore the redundancy on the spatial dimension of feature maps, which is addressed by the proposed OctConv, making OctConv orthogonal and complimentary to the previous methods.

Multi-scale Representation Learning. The proposed Oct-Conv performs convolution on feature maps at different spatial resolutions, resulting in a multi-scale feature representation with an enlarged receptive field. Scale-spaces have long been applied for local feature extraction, such as the once popular SIFT features [31]. In the deep learning era, existing methods focus on merging multi-scale features [28, 48, 14, 22] and better capturing long range information [42, 6, 9]. Such approaches however, aggregate information only at a small number of depths (usually in the middle or close to the end) of the network by inserting newly proposed blocks. The bL-Net [4] and ELASTIC-Net [41] frequently down- and up-sample the feature maps throughout the network to automatically learn multi-scale features. However, both methods are designed as a replacement of residual block which requires extra expertise and hyper-parameter tuning, especially when applied to different network architectures like MobileNetV1 [20], DenseNet [24]. Besides, these methods only synchronize multi-scale information at the end of each building block and synchronize all information on high resolution maps. In [22], Huang *et al*. keep multi-scale features throughout the network and have inter-scale connections at each layer of a DenseNet. Aiming at a reduced computational cost,

they use only the coarse features as inputs to multiple classifiers at different depths. In contrast, OctConv is designed as a replacement of vanilla convolution and can be applied to existing CNNs directly without network architecture adjustment. For OctConv, multi-scale information is synchronized at every layer in each group, delivering higher learning capacity and efficiency. We extensively compare OctConv to all closely related approaches in the experiments section and show that OctConv CNNs give the top results on a number of challenging benchmarks.

3. Method

In this section, we first introduce the octave feature representation for reducing the spatial redundancy in feature maps and then describe the Octave Convolution that operates directly on it. We also discuss implementation details and show how to integrate OctConv into group and depthwise convolution architectures.

3.1. Octave Feature Representation

For the vanilla convolution, all input and output feature maps have the same spatial resolution. However, the spatial-frequency model [1, 12] argues that a natural image can be factorized into a low-frequency signal that captures the global layout and coarse structure, and a high-frequency part that captures fine details, as shown in Figure 1(a). In an analogous way, we argue that there is a subset of the feature maps that capture spatially low-frequency changes and contain spatially redundant information.

To reduce such spatial redundancy, we introduce the *octave feature representation* that explicitly factorizes the feature map tensors into groups corresponding to low and high frequencies. The scale-space theory [29] provides us with a principled way of creating scale-spaces of spatial resolutions, and defines an *octave* as a division of the spatial dimensions by a power of 2 (we only explore 2^1 in this work). We define the low- and high-frequency spaces in this fashion, *i.e.* by reducing the spatial resolution of the low-frequency feature maps by an octave.

Formally, let $X \in \mathbb{R}^{c \times h \times w}$ denote the input feature tensor of a convolutional layer, where h and w denote the spatial dimensions and c the number of feature maps or channels. We explicitly factorize X along the channel dimension into $X = \{X^H, X^L\}$, where the high-frequency feature maps $X^H \in \mathbb{R}^{(1-\alpha)c \times h \times w}$ capture fine details and the low-frequency maps $X^L \in \mathbb{R}^{\alpha c \times \frac{h}{2} \times \frac{w}{2}}$ vary slower in the spatial dimensions (w.r.t. the image locations). Here $\alpha \in [0, 1]$ denotes the *ratio* of channels allocated to the low-frequency part and the low-frequency feature maps are defined *an octave lower* than the high frequency ones, *i.e.* at half of the spatial resolution as shown in Figure 1(c).

In the next subsection, we introduce a convolution operator that operates directly on this multi-frequency feature

representation and name it *Octave Convolution*, or *OctConv* for short.

3.2. Octave Convolution

The octave feature representation presented in Section 3.1 reduces the spatial redundancy and is more compact than the original representation. However, the vanilla convolution cannot directly operate on such a representation, due to differences in spatial resolution in the input features. A naive way of circumventing this is to up-sample the low-frequency part X^L to the original spatial resolution, concatenate it with X^H and then convolve, which would lead to extra costs in computation and memory and diminish all the savings from the compression. In order to fully exploit our compact multi-frequency feature representation, we introduce the Octave Convolution that can directly operate on factorized tensors $X = \{X^H, X^L\}$ without requiring any extra computational or memory overhead.

Vanilla Convolution. Let $W \in \mathbb{R}^{c \times k \times k}$ denote a $k \times k$ convolution kernel and $X, Y \in \mathbb{R}^{c \times h \times w}$ denote the input and output tensors, respectively. Each feature map in $Y_{p,q} \in \mathbb{R}^c$ can be computed by

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

where (p, q) denotes the location coordinate and $\mathcal{N}_k = \{(i, j) : i = \{-\frac{k-1}{2}, \ldots, \frac{k-1}{2}\}, j = \{-\frac{k-1}{2}, \ldots, \frac{k-1}{2}\}\}$ defines a local neighborhood. For simplicity, in all equations we omit the padding, we assume k is an odd number and that the input and output data have the same dimensionality, *i.e.* $c_{in} = c_{out} = c$.

Octave Convolution. The goal of our design is to effectively process the low and high frequency in their corresponding frequency tensor but also enable efficient communication between the high and low frequency component of our Octave feature representation. Let X, Y be the factorized input and output tensors. Then the high- and low-frequency feature maps of the output $Y = \{Y^H, Y^L\}$ will be given by $Y^H = Y^{H \to H} + Y^{L \to H}$ and $Y^L = Y^{L \to L} + Y^{H \to L}$, respectively, where $Y^{A \to B}$ denotes the convolutional update from feature map group A to group B. Specifically, $Y^{H \to H}, Y^{L \to L}$ denote intra-frequency information update, while $Y^{H \to L}, Y^{L \to H}$ denote inter-frequency communication.

To compute these terms, we split the convolutional kernel W into two components $W = [W^H, W^L]$ responsible for convolving with X^H and X^L respectively. Each component can be further divided into intra- and inter-frequency part: $W^H = [W^{H \to H}, W^{L \to H}]$ and $W^L = [W^{L \to L}, W^{H \to L}]$ with the parameter tensor shape shown in Figure 2(b). Specifically for high-frequency feature map, we compute it at location (p, q) by using a regular convolution for the intra-frequency update, and for the inter-
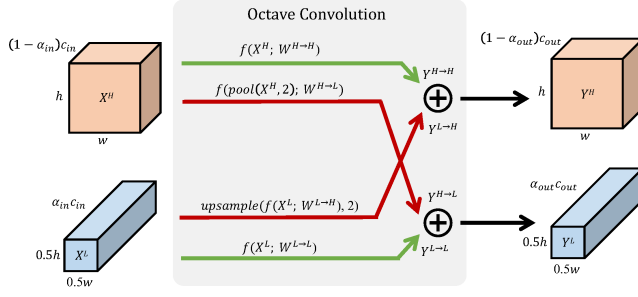
(a) Detailed design of the Octave Convolution. Green arrows correspond to information updates while red arrows facilitate information exchange between the two frequencies.
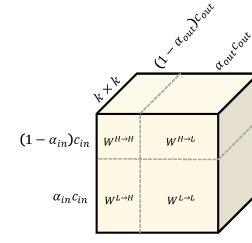
(b) The Octave Convolution kernel. The $k \times k$ Octave Convolution kernel $W \in \mathbb{R}^{c_{in} \times c_{out} \times k \times k}$ is equivalent to the vanilla convolution kernel in the sense that the two have the exact same number of parameters.

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.

frequency communication we can fold the up-sampling over the feature tensor X^L into the convolution, removing the need of explicitly computing and storing the up-sampled feature maps as follows:

$$
\begin{aligned}
Y_{p,q}^H =& Y_{p,q}^{H \to H} + Y_{p,q}^{L \to H} \\
=& \sum_{i,j \in \mathcal{N}_k} W_{i + \frac{k-1}{2}, j + \frac{k-1}{2}}^{H \to H}{}^\top X_{p+i,q+j}^H \\
&+ \sum_{i,j \in \mathcal{N}_k} W_{i + \frac{k-1}{2}, j + \frac{k-1}{2}}^{L \to H}{}^\top X_{(\lfloor \frac{p}{2} \rfloor + i),(\lfloor \frac{q}{2} \rfloor + j)}^L,
\end{aligned}
\tag{2}
$$

where $\lfloor \cdot \rfloor$ denotes the floor operation. Similarly, for the low-frequency feature map, we compute the intra-frequency update using a regular convolution. Note that, as the map is in one octave lower, the convolution is also low-frequency w.r.t. the high-frequency coordinate space. For the inter-frequency communication we can again fold the down-sampling of the feature tensor X^H into the convolution as follows:

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

where multiplying a factor 2 to the locations (p, q) performs down-sampling, and further shifting the location by half step is to ensure the down-sampled maps well aligned with the input. However, since the index of X^H can only be an integer, we could either round the index to $(2*p+i, 2*q+j)$ or approximate the value at $(2*p+0.5+i, 2*q+0.5+j)$ by averaging all 4 adjacent locations. The first one is also known as strided convolution and the second one as average pooling. As we discuss in Section 3.3 and Fig. 3, strided convolution leads to misalignment; we therefore use average pooling to approximate this value for the rest of the paper.

An interesting and useful property of the Octave Convolution is the larger receptive field for the low-frequency feature maps. Convolving the low-frequency part X^L with $k \times k$ convolution kernels, results in an effective enlargement of the receptive field by a factor of 2 compared to vanilla convolutions. This further helps each OctConv layer capture more contextual information from distant locations and can potentially improve recognition performance.

3.3. Implementation Details

As discussed in the previous subsection, the index $\{(2 * p + 0.5 + i), (2 * q + 0.5 + j)\}$ has to be an integer for Eq. 3. Instead of rounding it to $\{(2 * p + i), (2 * q + j)\}$, i.e. conduct convolution with stride 2 for down-sampling, we adopt average pooling to get more accurate approximation. This helps alleviate misalignments that appear when aggregating information from different scales [11], as shown in Figure 3 and validated in Table 3. We can now rewrite the output $Y = \{Y^H, Y^L\}$ of the Octave Convolution using average pooling for down-sampling as:

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

where $f(X; W)$ denotes a convolution with parameters W, $\texttt{pool}(X, k)$ is an average pooling operation with kernel size $k \times k$ and stride k. $\texttt{upsample}(X, k)$ is an up-sampling operation by a factor of k via nearest interpolation.

The details of the OctConv operator implementation are shown in Figure 2. It consists of four computation paths that correspond to the four terms in Eq. (4): two green paths correspond to information updating for the high- and low-frequency feature maps, and two red paths facilitate information exchange between the two octaves.

Group and Depth-wise convolutions. The Octave Convolution can also be adopted to other popular variants of the vanilla convolution such as group [45] or depth-wise [20] convolutions. For the group convolution case, we simply

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%

Figure 3: Strided convolution causes misaligned feature maps after up-sampling. As the example shows, up-sampling after the strided convolution will cause the entire feature map to move to the lower right, which is problematic when we add the shifted map with the unshifted map.

set all four convolution operations that appear inside the design of the OctConv to group convolutions. Similarly, for the depth-wise convolution case, the convolution operations are depth-wise and therefore the information exchange paths are eliminated, leaving only two depth-wise convolution operations. We note that both group OctConv and depth-wise OctConv reduce to their respective vanilla versions if we do not compress the low-frequency part.

Efficiency analysis. Table 1 shows the theoretical computational cost and memory consumption of OctConv over the vanilla convolution and vanilla feature map representation. More information on deriving the theoretical gains presented in Table 1 can be found in the supplementary material. We note the theoretical gains are calculated per convolutional layer. In Section 4 we present the corresponding practical gains on real scenarios and show that our OctConv implementation can sufficiently approximate the theoretical numbers.

Integrating OctConv into backbone networks. OctConv is backwards compatible with vanilla convolution and can be inserted to regular convolution networks without special adjustment. To convert a vanilla feature representation to a multi-frequency feature representation, *i.e.* at the first OctConv layer, we set $\alpha_{in} = 0$ and $\alpha_{out} = \alpha$. In this case, OctConv paths related to the low-frequency input is disabled, resulting in a simplified version which only has two paths. To convert the multi-frequency feature representation back to vanilla feature representation, *i.e.* at the last OctConv layer, we set $\alpha_{out} = 0$. In this case, OctConv paths related to the low-frequency output is disabled, resulting in a single full resolution output.

4. Experimental Evaluation

In this section, we validate the effectiveness and efficiency of the proposed Octave Convolution for both 2D and 3D networks. We first present ablation studies for image classification on ImageNet [13] and then compare it with the state-of-the-art. Then, we show the proposed OctConv also works in 3D CNNs using Kinetics-400 [26, 3] and Kinetics-
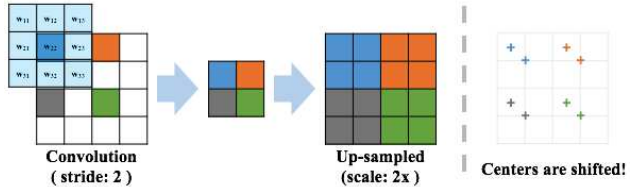
600 [2] datasets. The best results per category/block are highlighted in bold font throughout the paper.

4.1. Experimental Setups

Image classification. We examine OctConv on a set of most popular CNNs [20, 35, 18, 19, 24, 45, 21] by replacing the regular convolutions with OctConv (except the first convolutional layer before the max pooling). The resulting networks only have one global hyper-parameter α, which denotes the ratio of low frequency part. We do apple-to-apple comparison and reproduce all baseline methods by ourselves under the same training/testing setting for internal ablation studies. All networks are trained with naïve softmax cross entropy loss except that the MobileNetV2 also adopts the label smoothing [37], and the best ResNet-152 adopts both label smoothing and mixup [46] to prevent overfitting. Same as [4], all networks are trained from scratch and optimized by SGD with cosine learning rate [15]. Standard accuracy of single centeral crop [18, 19, 45, 4, 41] on validation set is reported[1].

Video action recognition. We use both Kinetics-400 [26, 3] and Kinetics-600 [2] for human action recognition. We choose standard baseline backbones from Inflated 3D ConvNet [42] and compare them with the OctConv counterparts. We follow the setting from [43] using frame length of 8 as standard input size and training 300k iterations in total. To make fair comparison, we report the performance of the baseline and OctConv under precisely the same settings. For the inference time, we average the predictions over 30 crops (each of (left, center, right) $\times$ 10 crops along temporal dimension), again following prior work [42].

4.2. Ablation Study on ImageNet

We conduct a series of ablation studies aiming to answer the following questions: 1) *Does OctConv have better FLOPs-Accuracy trade-off than vanilla convolution?* 2) *In which situation does the OctConv work the best?*

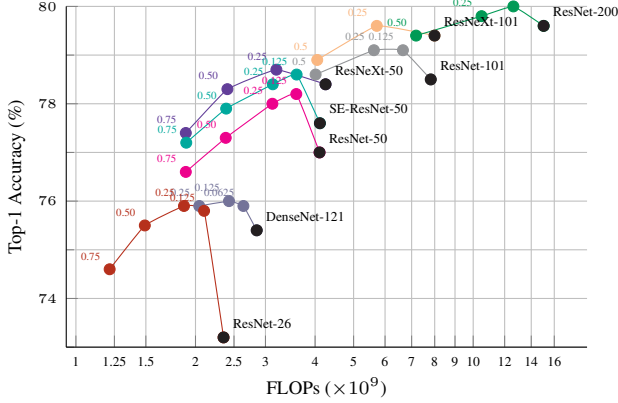[1]Code and trained models will be released on GitHub soon.

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.

Results on ResNet-50. We begin with using the popular ResNet-50 [19] as the baseline CNN and replacing the regular convolution with our proposed OctConv to examine the flops-accuracy trade-off. In particular, we vary the global ratio $\alpha \in \{0.125, 0.25, 0.5, 0.75\}$ to compare the image classification accuracy versus computational cost (i.e. FLOPs) [18, 19, 45, 8] with the baseline. The results are shown in Figure 4 in pink.

We make following observations. 1) The flops-accuracy trade-off curve is a concave curve, where the accuracy first rises up and then slowly goes down. 2) We can see two sweet spots: The first at $\alpha = 0.5$, where the network gets similar or better results even when the FLOPs are reduced by about half; the second at $\alpha = 0.125$, where the network reaches its best accuracy, 1.2% higher than baseline (black circle). We attribute the increase in accuracy to OctConv's effective design of multi-frequency processing and the corresponding enlarged receptive field which provides more contextual information to the network. While reaching the accuracy peak at 0.125, the accuracy does not suddenly drop but decreases slowly for higher ratios α, indicating reducing the resolution of the low frequency part does not lead to significant information loss. Interestingly, 75% of the feature maps can be compressed to half the resolution with only 0.4% accuracy drop, which demonstrates effectiveness of grouping and compressing the smoothly changed feature maps for reducing the spatial redundancy in CNNs. In Table 2 we demonstrate the theoretical FLOPs saving of Oct-Conv is also reflected in the actual CPU inference time in practice. For ResNet-50, we are close to obtaining theoretical FLOPs speed up. These results indicate OctConv is able to deliver important practical benefits, rather than only saving FLOPs in theory.

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 (%)
	strided conv.	✓	✓	76.3
Oct-ResNet-50	avg. pooling			76.0
ratio: 0.5	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**

Results on more CNNs. To further examine if the proposed OctConv works for other networks with different depth/wide/topology, we select the currently most popular networks as baselines and repeat the same ablation study. These networks are ResNet-(26;50;101;200) [19], ResNeXt-(50,32×4d;101,32×4d) [45], DenseNet-121 [24] and SE-ResNet-50 [21]. The ResNeXt is chosen for assessing the OctConv on group convolution, while the SE-Net [21] is used to check if the gain of SE block found on vanilla convolution based networks can also be seen on OctConv. As shown in Figure 4, OctConv equipped networks for different architecture behave similarly to the Oct-ResNet-50, where the FLOPs-Accuracy trade-off is in a concave curve and the performance peak also appears at ratio $\alpha = 0.125$ or $\alpha = 0.25$. The consistent performance gain on a variety of backbone CNNs confirms that OctConv is a good replacement of vanilla convolution.

Besides, we also have some intriguing findings: 1) Oct-Conv can help CNNs improve the accuracy while decreas-

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 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‡§ (32×4d) [4]	-	101 (+1)	**43.4**	4.1	78.9
Oct-ResNeXt-101‡§ (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

ing the FLOPs, deviating from other methods that reduce the FLOPs with a cost of lower accuracy. 2) At test time, the gain of OctConv over baseline models increases as the test image resolution grows because OctConv can detect large objects better due to its larger receptive field, as shown in Table 4. 3) Both the information exchanging paths are important, since removing any of them can lead to accuracy drop as shown in Table 3. 4) Shallow networks, *e.g.* ResNet-26, have a rather limited receptive field, and can especially benefit from OctConv, which greatly enlarges their receptive field.

4.3. Comparing with SOTAs on ImageNet

Small models. We adopt the most popular light weight networks as baselines and examine if OctConv works well on these compact networks with depth-wise convolution. In particular, we use the "0.75 MobileNet (v1)" [20] and "1.0 MobileNet (v2)" [35] as baseline and replace the regular convolution with our proposed OctConv. The results are shown in Table 5. We find that OctConv can reduce the FLOPs of MobileNetV1 by 34%, and provide better accuracy and faster speed in practice; it is able to reduce the FLOPs of MobileNetV2 by 15%, achieving the same ac-

curacy with faster speed. When the computation budget is fixed, one can adopt wider models to increase the learning capacity because OctConv can compensate the extra computation cost. In particular, our OctConv equipped networks achieve 2% improvement on MobileNetV1 under the same FLOPs and 1% improvement on MobileNetV2.

Medium models. In the above experiment, we have compared and shown that OctConv is complementary with a set of state-of-the-art CNNs [18, 19, 45, 24, 20, 35, 21]. In this part, we compare OctConv with Elastic [41] and bL-Net [4] which share a similar idea as our method. Five groups of results are shown in Table 6. In group 1, our Oct-ResNeXt-50 achieves better accuracy than the Elastic [41] based method (78.7% v.s. 78.4%) while reducing the computational cost by 31%. In group 2, the Oct-ResNeXt-101 also achieves higher accuracy than the Elastic based method (79.5% v.s. 79.2%) while costing 38% less computation. When compared to the bL-Net [4], OctConv equipped methods achieve better FLOPs-Accuracy trade-off without bells and tricks. When adopting the tricks used in the baseline bL-Net [4], our Oct-ResNet-50 achieves 0.8% higher accuracy than bL-ResNet-50 under the same computational budget (group 3), and Oct-ResNeXt-50 (group 4) and Oct-ResNeXt-101 (group 5) get better accuracy under comparable or even lower computational budget. This is because both the Elastic-Net [41] and bL-Net [4] are designed to exploit multi-scale features instead of reducing the spatial redundancy. In contrast, OctConv uses a more compact feature representation to store and process the information throughout the network, and can thus achieve better efficiency and performance.

Large models. Table 7 shows the results of OctConv in large models. Here, we choose the ResNet-152 as the backbone CNN, replacing the first 7×7 convolution by three 3×3 convolution layers and removing the max pooling by a lightweight residual block [4]. We report results for Oct-ResNet-152 with and without the SE-block [21]. As can be seen, our Oct-ResNet-152 achieves accuracy comparable to the best manually designed networks with less FLOPs (10.9G v.s. 12.7G). Since our model does not use group or depth-wise convolutions, it also requires significantly less GPU memory, and runs faster in practice compared to the SE-ShuffleNet v2-164 and AmoebaNet-A (N=6, F=190) which have low FLOPs in theory but run slow in practice due to the use of group and depth-wise convolutions. Our proposed method is also complementary to Squeeze-and-excitation [21], where the accuracy can be further boosted when the SE-Block is added (last row).

---

[2]For small models, we should notice according to arithmetic intensity [44], real execution time is not only bounded by FLOPS.

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**

4.4. Experiments of Video Recognition on Kinetics

In this subsection, we evaluate the effectiveness of Oct-Conv for action recognition in videos and demonstrate that our spatial OctConv is sufficiently generic to be integrated into 3D convolution to decrease #FLOPs and increase accuracy at the same time. As shown in Table 8, OctConv consistently decreases FLOPs and meanwhile improves the accuracy when added to C2D and I3D [42, 43], and is also complimentary to the Nonlocal building block [42]. This is observed for models pre-trained on ImageNet [13] as well as models trained from scratch on Kinetics.

Specifically, we first investigate the behavior of training OctConv equipped I3D models from scratch on Kinetics. We use a learning rate $10\times$ larger than the standard and train it 16 times longer than finetuning setting for a better convergence. Compared to the vanilla I3D model, Oct-I3D achieves 1.0% higher accuracy with 91% of the FLOPs.

We then explore the behavior of finetuning a OctConv on ImageNet pre-trained model with step-wise learning schedule. For this, we train an OctConv ResNet-50 model [18] on ImageNet [13] and then inflate it into a network with 3D convolutions [39] (over space and time) using the I3D technique [3]. After the inflation, we finetune the inflated Oct-Conv following the schedule described in [43] on Kinetics-400. Compared to the 71.9% Top-1 accuracy of the C2D baseline on the Kinetics-400 validation set, the OctConv counterpart achieves 73.8% accuracy, using 90% of the FLOPs. For I3D, adding OctConv improves accuracy from 73.3% to 74.6% accuracy, while using only 91% of the FLOPs. We also demonstrate that the gap is consistent when adding Non-local [42]. Finally, we repeat the I3D experiment on Kinetics-600 [2] dataset and have a consistent finding, which further confirms the effectiveness of our method.

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)**

5. Conclusion

In this work, we address the problem of reducing spatial redundancy that widely exists in vanilla CNN models, and propose a novel Octave Convolution operation to store and process low- and high-frequency features separately to improve the model efficiency. Octave Convolution is sufficiently generic to replace the regular convolution operation in-place, and can be used in most 2D and 3D CNNs without model architecture adjustment. Beyond saving a substantial amount of computation and memory, Octave Convolution can also improve the recognition performance by effective communication between the low- and high-frequency and by enlarging the receptive field size which contributes to capturing more global information. Our extensive experiments on image classification and video action recog-

---

[3]The auto-tune is set to *off* when evaluating the memory cost for more accurate result, and is set to *on* when measuring speed for fastest speed.

[4]An extra BatchNorm is added at the beginning of each residual func-

tion, otherwise the gradient will easily diverged due to the newly added SE module. This costs more memory and slows down the speed but can provide higher accuracy.

nition confirm the superiority of our method for striking a much better trade-off between recognition performance and model efficiency, not only in FLOPs, but also in practice.

References

[1] F. W. Campbell and J. Robson. Application of fourier analysis to the visibility of gratings. *The Journal of physiology*, 197(3):551–566, 1968.

[2] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[4] C.-F. Chen, Q. Fan, N. Mallinar, T. Sercu, and R. Feris. Big-little net: An efficient multi-scale feature representation for visual and speech recognition. *Proceedings of the Seventh International Conference on Learning Representations*, 2019.

[5] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, et al. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 578–594, 2018.

[6] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. Aˆ2-nets: Double attention networks. In *Advances in Neural Information Processing Systems*, pages 350–359, 2018.

[7] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. Multi-fiber networks for video recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 352–367, 2018.

[8] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In *Advances in Neural Information Processing Systems*, pages 4467–4475, 2017.

[9] Y. Chen, M. Rohrbach, Z. Yan, S. Yan, J. Feng, and Y. Kalantidis. Graph-based global reasoning networks. *arXiv preprint arXiv:1811.12814*, 2018.

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

[11] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[12] R. L. De Valois and K. K. De Valois. Spatial vision. *Oxford psychology series, No. 14.*, 1988.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slow-fast networks for video recognition. *arXiv preprint arXiv:1812.03982*, 2018.

[15] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[16] S. Han, J. Pool, S. Narang, H. Mao, E. Gong, S. Tang, E. Elsen, P. Vajda, M. Paluri, J. Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[21] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[22] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense networks for resource efficient image classification. *ICLR*, 2018.

[23] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2752–2761, 2018.

[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[25] Intel. Math kernel library for deep neural networks (mkldnn). https://github.com/intel/mkl-dnn/tree/7de7e5d02bf687f971e7668963649728356e0c20, 2018.

[26] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[28] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[29] T. Lindeberg. *Scale-space theory in computer vision*, volume 256. Springer Science & Business Media, 2013.

[30] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[31] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[32] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin. Thinet: pruning cnn filters for a thinner net. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[33] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[34] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[39] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[40] F. Tung and G. Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018.

[41] H. Wang, A. Kembhavi, A. Farhadi, A. Yuille, and M. Rastegari. Elastic: Improving cnns with instance specific scaling policies. *arXiv preprint arXiv:1812.05262*, 2018.

[42] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[43] X. Wang, R. Girshick, A. Gupta, and K. He. https://github.com/facebookresearch/video-nonlocal-net, 2018.

[44] S. W. Williams. *Auto-tuning performance on multicore computers*. University of California, Berkeley, 2008.

[45] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[46] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *Proceedings of the Sixth International Conference on Learning Representations*, 2018.

[47] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[48] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[49] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

Supplementary Material

1. Relative Theoretical Gains of OctConv

In Table 1 of the main paper, we reported the relative theoretical gains of the proposed multi-frequency feature representation over regular feature representation with respect to memory footprint and computational cost, as measured in FLOPS (*i.e.* multiplications and additions). In this section, we show how the gains are estimated in theory.

Memory cost. The proposed OctConv stores the feature representation in a multi-frequency feature representation as shown in Figure 5, where the low frequency tensor is stored in $2\times$ lower spatial resolution and thus cost 75% less space to store the low frequency maps compared with the conventional feature representation. The relative memory cost is conditional on the ratio (α) and is calculated by

$$1 - \frac{3}{4}\alpha. \tag{5}$$

Computational cost. The computational cost of OctConv is proportional to the number of locations and channels that are needed to be convolved on. Following the design shown in Figure 2 in the main paper, we need to compute four paths, namely $H \to H$, $H \to L$, $L \to H$, and $L \to L$.

We assume the convolution kernel size is $k \times k$, the spatial resolution of the high-frequency feature is $h \times w$, and there are $(1 - \alpha)c$ channels in the high-frequency part and αc channels in the low-frequency part. Then the FLOPS for computing each paths are calculated as below.

$$FLOPS(Y^{H \to H}) = h \times w \times k^2 \times (1 - \alpha)^2 \times c^2$$
$$FLOPS(Y^{H \to L}) = \frac{h}{2} \times \frac{w}{2} \times k^2 \times \alpha \times (1 - \alpha) \times c^2$$
$$FLOPS(Y^{L \to H}) = \frac{h}{2} \times \frac{w}{2} \times k^2 \times (1 - \alpha) \times \alpha \times c^2$$
$$FLOPS(Y^{L \to L}) = \frac{h}{2} \times \frac{w}{2} \times k^2 \times \alpha^2 \times c^2$$
$$\tag{6}$$

We omit FLOPS for adding $Y^{H \to H}$ and $Y^{L \to H}$ together, as well as that of adding $Y^{L \to L}$ and $Y^{H \to H}$ together, since the FLOPS of such addition is less than $h \times w \times c$, and is negligible compared with other computational costs. The computational cost of the pooling operation is also ignorable compared with other computational cost. The nearest neighborhood up-sampling is basically duplicating values which does not involves any computational cost. Therefore, by adding up all FLOPS in Eqn 6, we can estimate the overall FLOPS for compute Y^H and Y^L in Eqn 7.

$$FLOPS([Y^H, Y^L]) = (1 - \frac{3}{4}\alpha(2 - \alpha)) \times h \times w \times k^2 \times c^2 \tag{7}$$
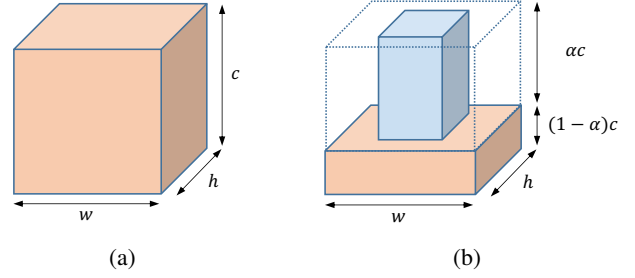

(a) (b)

Figure 5: (a) The conventional feature representation used by vanilla convolution. (c) The proposed multi-frequency feature representation stores the smoothly changing, low-frequency maps in a low-resolution tensor to reduce spatial redundancy, used by Octave Convolution. The figure is rotated compared to the one in the main paper for clarity.

For vanilla convolution, the FLOPS for computing output feature map Y of size $c \times h \times w$ with the kernel size $k \times k$, and input feature map of size $c \times h \times w$, can be estimated as below.

$$FLOPS(Y) = h \times w \times k^2 \times c^2 \tag{8}$$

three out of four internal convolution operations are conducted on the lower resolution tensors except the first convolution, *i.e.* $f(X^H, W^{H \to H})$. Thus, the relative computational cost compared with vanilla convolution using the same kernel size and number of input/out channels is: Therefore, the computational cost ratio between the OctConv and vanilla convolution is $(1 - \frac{3}{4}\alpha(2 - \alpha))$.

$$\frac{(1 - \alpha)^2 c^2 + \frac{1}{2}\alpha(1 - \alpha)c^2 + \frac{1}{4}\alpha^2 c^2}{c^2}$$
$$= 1 - \frac{3}{4}\alpha(2 - \alpha). \tag{9}$$

Note that the computational cost of the pooling operation is ignorable and thus is not considered. The nearest neighborhood up-sampling is basically duplicating values which does not involves any computational cost.

2. ImageNet ablation study results

For clarity of presentation and to allow future work to compare to the precise numbers, we further report in Table 9 the values that are plotted in Figure 4 of the main text.

Backbone		baseline	$\alpha = 0.125$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$
ResNet-26	GFLOPs	2.353	2.102	1.871	1.491	1.216
	Top-1 acc.	73.2	75.8	75.9	75.5	74.6
DenseNet-121	GFLOPs	2.852	2.428	2.044	-	-
	Top-1 acc.	75.4	76.0	75.9	-	-
ResNet-50	GFLOPs	4.105	3.587	3.123	2.383	1.891
	Top-1 acc.	77.0	78.2	78.0	77.3	76.6
SE-ResNet-50	GFLOPs	4.113	3.594	3.130	2.389	1.896
	Top-1 acc.	77.6	78.6	78.4	77.9	77.2
ResNeXt-50	GFLOPs	4.250	-	3.196	2.406	1.891
	Top-1 acc.	78.4	-	78.7	78.3	77.4
ResNet-101	GFLOPs	7.822	6.656	5.625	4.012	-
	Top-1 acc.	78.5	79.1	79.1	78.6	-
ResNeXt-101	GFLOPs	7.993	-	5.719	4.050	-
	Top-1 acc.	79.4	-	79.6	78.9	-
ResNet-200	GFLOPs	15.044	12.623	10.497	7.183	-
	Top-1 acc.	79.6	80.0	79.8	79.4	-

Table 9: Ablation study on ImageNet in table form corresponding to the plots in Figure 4 in the main paper. Note: All networks are trained with naïve softmax loss without label smoothing [37] or mixup [46]