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

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) operation1 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

The efficiency of Convolutional Neural Networks (CNNs) keeps increasing with recent efforts to reduce the inherent redundancy in dense model parameters [15, 31, 42] and in the channel dimension of feature maps [47, 18, 6, 9]. 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.

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, 10, 37, 39]. 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 convolution, and propose Octave Convolution (OctConv) which

---

1https://github.com/facebookresearch/OctConv

Figure 1: (a) Motivation. The spatial frequency model for vision [1, 10] 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.
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 [22, 41, 35, 33, 29], reducing channel-wise redundancy in convolutional feature maps [47, 9, 34, 32, 21] and reducing redundancy in dense model parameters [42, 15, 31]. Moreover, different from methods that exploit multi-scale information [4, 43, 12], 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. Compared to the closely related Multi-grid convolution [25], OctConv provides more insights on reducing the spatial redundancy in CNNs based on the frequency model and adopts more efficient inter-frequency information exchange strategy with better performance. We further integrate the OctConv into a wide variety of backbone architectures (including the ones featuring group, depth-wise, and 3D convolutions) and demonstrate universality of OctConv.

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 [16, 17], ResNeXt [47], DenseNet [22], MobileNet [18, 34] and SE-Net [19] on 2D image recognition on ImageNet [11], as well as 3D CNN backbones C2D [44] and I3D [44] on video action recognition on Kinetics [24, 3, 2]. The OctConv-equipped Oct-ResNet-152 can match or outperform state-of-the-art manually designed networks [32, 19] 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 OctConv 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 [26] and VGG [35], researchers have made substantial efforts to improve the efficiency of CNNs. ResNet [16, 17] and DenseNet [22] improve the network topology by adding shortcut connections to early layers. ResNeXt [47] and ShuffleNet [49] use sparsely connected group convolutions to reduce redundancy in inter-channel connectivity. Xception [9] and MobileNet [18, 34] adopt depth-wise convolutions that further reduce the connection density. Meanwhile, NAS [51], PNAS [29] and AmoebaNet [33] propose to atomically find the best network topology for a given task. Pruning methods, such as DSD [15] and ThiNet [31], focus on reducing the redundancy in the model parameters by eliminating the least significant weight or connections in CNNs. Besides, HetConv [36] propose to replace the vanilla convolution filters with heterogeneous convolution filters that are in different sizes. 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 complementary to these previous methods. Noticeably, OctConv does not change the connectivity between feature maps, making it also different from inception-like multi-path designs [41, 40, 47].

**Multi-scale Representation Learning.** Prior to the success of deep learning, multi-scale representation has long been applied for local feature extraction, such as the SIFT features [30]. In the deep learning era, multi-scale representation also plays a prominent role due to its strong robustness and generalization ability. FPN [27] and PSP [50] merge convolutional features from different depths at the end of the networks for object detection and segmentation tasks. MSDNet [20] and HR-Nets [38], proposed carefully designed network architectures that contain multiple branches where each branch has its own spatial resolution. The B-Net [4] and ELASTIC-Net [43] adopt similar idea, but are designed as a replacement of residual block for ResNet [16, 17] and thus are more flexible and easier to use. But extra expertise and hyper-parameter tuning are still required when adopt them to architectures beyond ResNet, such as MobileNetV1 [18], DenseNet [22]. Multi-grid CNNs [25] propose a multi-grid pyramid feature representation and define the MG-Conv operator as a replacement of convolution operator, which is conceptually similar to our method but is motivated for exploiting multi-
scale features. Compared with MG-Conv, OctConv adopts more efficient design to exchange inter-frequency information with higher performance as can be found in Sec. 3.3 and Sec. 4.3. For video models, the recently proposed Slow-Fast Networks [12] introduce multi-scale pathways on the temporal dimension. As we show in Section 4.4, this is complementary to OctConv which operates on the spatial dimensions.

In a nutshell, OctConv focuses on reducing the spatial redundancy in CNNs and is designed to replace vanilla convolution operations without needing to adjust backbone CNN architecture. We extensively compare OctConv to closely related methods in the sections of method and experiment and show that OctConv CNNs give top results on a number of challenging benchmarks.

3. Method

In this section, we first introduce the octave feature representation and then describe Octave Convolution, which operates directly on it. We also discuss implementation details and show how to integrate OctConv into group and depth-wise convolution architectures.

3.1. Octave Feature Representation

For the vanilla convolution, all input and output feature maps have the same spatial resolution, which may not be necessary since some of the feature maps may represent low-frequency information which is spatially redundant and can be further compressed as illustrated in Figure 1.

To reduce the 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 [28] 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 follow this fashion and reduce 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 (OctConv).

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 Octave Convolution, which can directly operate on factorized tensors $X = \{X^h, X^l\}$ without requiring any extra computational or memory overhead.

Vanilla Convolution. Let $W \in \mathbb{R}^{k \times k \times c}$ 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

$$Y_{p,q} = \sum_{i,j \in N_k} W_{i+p, j+q}^T X_{i+1, j+1},$$  \hspace{1cm} (1)

where $(p, q)$ denotes the location coordinate and $N_k = \{(i, j) : i = -\frac{k-1}{2}, \ldots, \frac{k-1}{2}, j = -\frac{k-1}{2}, \ldots, \frac{k-1}{2}\}$ 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 inter-frequency communication. 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 $Y^h = Y^{h \rightarrow h} + Y^{h \rightarrow l}$ and $Y^l = Y^{l \rightarrow l} + Y^{l \rightarrow h}$, respectively, where $Y^{A \rightarrow B}$ denotes the convolutional update from feature map group $A$ to group $B$. Specifically, $Y^{h \rightarrow h}, Y^{l \rightarrow l}$ denote intra-frequency update, while $Y^{h \rightarrow l}, Y^{l \rightarrow h}$ 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: $W^h = [W^{h \rightarrow h}, W^{h \rightarrow l}]$ and $W^l = [W^{l \rightarrow l}, W^{l \rightarrow h}]$ 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-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
k × 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, as shown in Appendix A. and Appendix C. We can now rewrite the output \(Y = \{Y^H, Y^L\}\) of the Octave Convolution using average pooling for down-sampling as:

\[
Y^H = f(X^H; W^H) + \text{upsample}(f(X^L; W^L), 2)
\]

\[
Y^L = f(X^L; W^L) + f(\text{pool}(X^H, 2), W^H)
\]

(4)

where \(f(X; W)\) denotes a convolution with parameters \(W\), \(\text{pool}(X, k)\) is an average pooling operation with kernel size \(k \times k\) and stride \(k\). \(\text{upsample}(X, k)\) 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 [47] or depth-wise [18] convolutions. For the group convolution case, we simply set all four convolution operations that appear inside the OctConv of the network where \(\alpha_{in} = 0, \alpha_{out} = \alpha\) throughout the network, apart from the first and last OctConv of the network where \(\alpha_{in} = \alpha, \alpha_{out} = 0\), respectively.

---

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.

---

**Feature maps as follows:**

\[
Y^H_{p,q} = Y^H_{p,q} + Y^L_{p,q} + Y^H_{p,q} + Y^L_{p,q} + Y^H_{p,q} + Y^L_{p,q} + Y^H_{p,q} + Y^L_{p,q}
\]

\[
= \sum_{i,j \in \mathbb{N}} W^H_{H} T_{i+1,k, j+1} X^H_{p+i,q+j}
\]

\[
+ \sum_{i,j \in \mathbb{N}} W^L_{H} T_{i+1,k, j+1} X^L_{((2/2)+i), ((2/2)+j)}
\]

where \([\cdot]\) 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:

\[
Y^L_{p,q} = Y^L_{p,q} + Y^H_{p,q}
\]

\[
= \sum_{i,j \in \mathbb{N}} W^L_{H} T_{i+1,k, j+1} X^L_{p+i,q+j}
\]

\[
+ \sum_{i,j \in \mathbb{N}} W^H_{L} T_{i+1,k, j+1} X^H_{((2*p+0.5*i)+1), ((2*q+0.5*j)+1)}
\]

\[
(3)
\]

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 \((2*p+0.5+i, 2*q+0.5+j)\) 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. ??, 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...
tions 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 convolutional 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.

**Comparison to Multi-grid Convolution [25].** The multi-grid conv (MG-Conv) [25] is a bi-directional and cross-scale convolution operator. Though being conceptually similar, our OctConv is different from MG-Conv in both the core motivation and design. MG-Conv aims to exploit multi-scale information in existing CNNs, while OctConv is focusing on reducing spatial redundancy among neighborhood pixels. In terms of design, MG-Conv adopts max-pooling for down-sampling. This requires extra memory for storing the index of the maximum value during training and further yields lower accuracy (see Appendix C.). MG-Conv also first up-samples and then convolves with the enlarged feature maps. Differently, OctConv aims for reducing spatial redundancy and is a naive extension of convolution operator. It uses average pooling to distill low-frequency features without extra memory cost and its upsampling operation follows the convolution, and is thus more efficient than MG-Conv. The meticulous design of the lateral paths are essential for OctConv to be much more memory and computationally efficient than MG-Conv and improve accuracy without increasing the network complexity. We compare OctConv to MG-Conv in Table 4.

### 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 [11] and then compare it with the state-of-the-art. Then, we show the proposed OctConv also works in 3D CNNs using Kinetics-400 [24, 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 [18, 34, 16, 17, 22, 47, 19] 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 $\alpha$, 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 naive softmax cross entropy loss except that the MobileNetV2 also adopts the label smoothing [40], and the best ResNet-152 adopts both label smoothing and mixup [48] to prevent overfitting. Same as [4], all networks are trained from scratch and optimized by SGD with cosine learning rate [13]. Standard accuracy of single central crop [16, 17, 47, 4, 43] on validation set is reported.

**Video action recognition.** We use both Kinetics-400 [24, 3] and Kinetics-600 [2] for human action recognition. We choose standard baseline backbones from Inflated 3D ConvNet [44] and compare them with the OctConv counterparts. We follow the setting from [45] using frame length of 8 as standard input size, training 300k iterations in total, and averaging the predictions over 30 crops during inference time. To make fair comparison, we report the performance of the baseline and OctConv under precisely the same settings.

#### 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?*

<table>
<thead>
<tr>
<th>ratio ($\alpha$)</th>
<th>0</th>
<th>.125</th>
<th>.25</th>
<th>.50</th>
<th>.75</th>
<th>.875</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>#FLOPs Cost</td>
<td>100%</td>
<td>82%</td>
<td>67%</td>
<td>44%</td>
<td>30%</td>
<td>26%</td>
<td>25%</td>
</tr>
<tr>
<td>Memory Cost</td>
<td>100%</td>
<td>91%</td>
<td>81%</td>
<td>63%</td>
<td>44%</td>
<td>35%</td>
<td>25%</td>
</tr>
</tbody>
</table>
Results on ResNet-50. We begin with using the popular ResNet-50 [17] 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) [16,17,47,7] with the baseline. The results are shown in Figure 3 in pink.

![Figure 3](image3.png)

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 $\alpha$. Numbers in X axis denote FLOPs in logarithmic scale.

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 $\alpha$, 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 0.3% 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 OctConv 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.

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) [17], ResNeXt-(50,32\times4d;101,32\times4d) [47], DenseNet-121 [22] and SE-ResNet-50 [19]. The ResNeXt is chosen for assessing the OctConv on group convolution, while the SE-Net [19] 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 3, 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.

Frequency Analysis. Figure 4 shows the frequency analysis results. We conducted the Fourier transform for each group of feature maps and visualized the averaged results. From the energy map, the low frequency group does not contain high frequency signal, while the high frequency group contains both low and high frequency signals. This confirms that low-frequency group indeed captures low-frequency information as expected. Note that OctConv gives the high frequency group the flexibility to store both low and high frequency signals for better learning capacity.

<table>
<thead>
<tr>
<th>ratio ($\alpha$)</th>
<th>Top-1 (%)</th>
<th>#FLOPs (G)</th>
<th>Inference Time (ms)</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>77.0</td>
<td>4.1</td>
<td>119</td>
<td>MKLDNN</td>
</tr>
<tr>
<td>N/A</td>
<td>77.0</td>
<td>4.1</td>
<td>115</td>
<td>TVM</td>
</tr>
<tr>
<td>.125</td>
<td>78.2</td>
<td>3.6</td>
<td>116</td>
<td>TVM</td>
</tr>
<tr>
<td>.25</td>
<td>78.0</td>
<td>3.1</td>
<td>99</td>
<td>TVM</td>
</tr>
<tr>
<td>.5</td>
<td>77.4</td>
<td>2.4</td>
<td>74</td>
<td>TVM</td>
</tr>
<tr>
<td>.75</td>
<td>76.7</td>
<td>1.9</td>
<td>61</td>
<td>TVM</td>
</tr>
</tbody>
</table>

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 vanilla 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.

![Figure 4](image4.png)

Figure 4: Frequency analysis for activation maps in different groups. ‘Baseline’ refers to vanilla ResNet. 10k activation maps are sampled from ResNet-101(Res3).
shows the results of OctConv in reducing the FLOPs of MobileNetV2 by 15%, achieving the same accuracy and faster speed in practice; it is able to reduce the FLOPs of MobileNetV1 by 7%, particularly we use the “0.75 MobileNet (v1)” [18] and “1.0 MobileNet (v2)” [34] as baseline and replace the regular convolution with our proposed OctConv. The results are shown in Table 3. 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 accuracy 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 [16, 17, 47, 22, 18, 34, 19]. In this part, we compare OctConv with MG-Conv [25], GloRe [8], Elastic [43] and bl-Net [4] which share a similar idea as our method. Seven groups of results are shown in Table 4. In group 1, our Oct-ResNet-26 shows 0.6% better accuracy than R-MG-34 while costing only one third of FLOPs and half of #Params. Also, our Oct-ResNet-50, which costs less than half of FLOPS, achieves 1.9% higher accuracy than R-MG-34. In group 2, adding our OctConv to GloRe network reduces the FLOPs with better accuracy. In group 3, our Oct-ResNeXt-50 achieves better accuracy than the Elastic [43] based method (78.8% v.s. 78.4%) while reducing the computational cost by 31%. In group 4, the Oct-ResNeXt-101 also achieves higher accuracy than the Elastic based method (79.6% 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 0.9% higher accuracy than bL-ResNet-50 under the same computational budget (group 5), and Oct-ResNeXt-50 (group 6) and Oct-ResNeXt-101 (group 7) get better accuracy under comparable or even lower computational budget. This is because MG-Conv [25], ElasticNet [43] and bl-Net [4] are designed following the principle of introducing multi-scale features without considering reducing the spatial redundancy. In contrast, OctConv is born for solving the high spatial redundancy problem in CNNs, uses more efficient strategies to store and process the information throughout the network, and can thus achieve better efficiency and performance.

### Large models.
Table 5 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 [19]. 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.

---

For small models, we should notice according to arithmetic intensity [46], real execution time is not only bounded by FLOPS.

---

**Table 3:** ImageNet classification results for Small models. * indicates it is better than original reproduced by MNEx GluonCV v0.4 [14]. The inference speed is tested using TVM on Intel Skylake processor (2.0GHz, single thread)².  

<table>
<thead>
<tr>
<th>Method</th>
<th>ratio (v)</th>
<th>#Params (M)</th>
<th>#FLOPs (M)</th>
<th>CPU (ms)</th>
<th>Top-1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CondenseNet (G = C = 8) [21]</td>
<td>-</td>
<td>294</td>
<td>71.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5 ShuffleNet (v1) [19]</td>
<td>-</td>
<td>3.4</td>
<td>292</td>
<td>71.5</td>
<td></td>
</tr>
<tr>
<td>1.5 ShuffleNet (v2) [19]</td>
<td>-</td>
<td>3.5</td>
<td>299</td>
<td>72.6</td>
<td></td>
</tr>
<tr>
<td>0.75 MobileNet (v1) [18]</td>
<td>-</td>
<td>2.6</td>
<td>325</td>
<td>73.8</td>
<td></td>
</tr>
<tr>
<td>0.75 Oct-MobileNet (v1) (ours)</td>
<td>.75</td>
<td>2.6</td>
<td>213</td>
<td>70.5</td>
<td></td>
</tr>
<tr>
<td>1.0 Oct-MobileNet (v1) (ours)</td>
<td>.5</td>
<td>4.2</td>
<td>321</td>
<td>72.2</td>
<td></td>
</tr>
<tr>
<td>1.0 MobileNet (v2) [34]</td>
<td>-</td>
<td>3.5</td>
<td>300</td>
<td>72.0</td>
<td></td>
</tr>
<tr>
<td>1.0 Oct-MobileNet (v2) (ours)</td>
<td>.75</td>
<td>3.5</td>
<td>256</td>
<td>72.0</td>
<td></td>
</tr>
<tr>
<td>1.125 Oct-MobileNet (v2) (ours)</td>
<td>.5</td>
<td>4.2</td>
<td>295</td>
<td>73.0</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4:** 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].

<table>
<thead>
<tr>
<th>Method</th>
<th>ratio (v)</th>
<th>Depth</th>
<th>#Params (M)</th>
<th>#FLOPs (G)</th>
<th>Top-1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-MG-34 [25]</td>
<td>-</td>
<td>34</td>
<td>12.9</td>
<td>75.5</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-26 (ours)</td>
<td>25</td>
<td>16.0</td>
<td>75.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-50 (ours)</td>
<td>5</td>
<td>50</td>
<td>25.6</td>
<td>77.4</td>
<td></td>
</tr>
<tr>
<td>OctNet-50 + GloRe (v1)</td>
<td>-</td>
<td>50</td>
<td>30.5</td>
<td>78.4</td>
<td></td>
</tr>
<tr>
<td>R-ResNeXt-50 + Elastic [43]</td>
<td>-</td>
<td>50</td>
<td>30.4</td>
<td>78.4</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-50 (32×4d) (ours)</td>
<td>.5</td>
<td>50</td>
<td>25.6</td>
<td>78.4</td>
<td></td>
</tr>
<tr>
<td>OctResNet-101 + Elastic [43]</td>
<td>-</td>
<td>101</td>
<td>44.5</td>
<td>79.2</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-101 (32-4d) (ours)</td>
<td>25</td>
<td>101</td>
<td>44.2</td>
<td>79.6</td>
<td></td>
</tr>
<tr>
<td>B-ResNet-50 (α = 0.75 Mobile) [4]</td>
<td>-</td>
<td>50</td>
<td>26.2</td>
<td>79.9</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-50 (ours)</td>
<td>5</td>
<td>50</td>
<td>25.6</td>
<td>77.8</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-50 (ours) + GloRe</td>
<td>5</td>
<td>50</td>
<td>25.6</td>
<td>77.4</td>
<td></td>
</tr>
<tr>
<td>B-ResNet-50 (32×4d) [4]</td>
<td>-</td>
<td>50</td>
<td>26.2</td>
<td>79.4</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-50 (32-4d) (ours)</td>
<td>5</td>
<td>50</td>
<td>25.1</td>
<td>78.6</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-50 (32-4d) (ours) + GloRe</td>
<td>5</td>
<td>50</td>
<td>25.2</td>
<td>78.4</td>
<td></td>
</tr>
<tr>
<td>B-ResNet-101 (32×4d) [4]</td>
<td>-</td>
<td>101</td>
<td>43.4</td>
<td>80.9</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-101 (32-4d) (ours)</td>
<td>5</td>
<td>101</td>
<td>40.1</td>
<td>79.4</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-101 (32-4d) (ours) + GloRe</td>
<td>5</td>
<td>101</td>
<td>44.2</td>
<td>79.1</td>
<td></td>
</tr>
<tr>
<td>Oct-ResNet-101 (32-4d) (ours) + GloRe + Elastic [43]</td>
<td>5</td>
<td>101</td>
<td>44.2</td>
<td>80.0</td>
<td></td>
</tr>
</tbody>
</table>

---

1. For small models, we should notice according to arithmetic intensity 46, real execution time is not only bounded by FLOPS.

---
due to the use of group and depth-wise convolutions. Our proposed method is also complementary to Squeeze-and-excitation [19], where the accuracy can be further boosted when the SE-Block is added (last row).

4.4. Experiments of Video Recognition on Kinetics

In this subsection, we evaluate the effectiveness of OctConv 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 6, OctConv consistently decreases FLOPs and meanwhile improves the accuracy when added to C2D and I3D [44, 45], and is also complementary to the Non-local [44]. This is observed for models pre-trained on ImageNet [11] as well as models trained from scratch on Kinetics. The higher accuracy, lower FLOPs and the ability of being complimentary to existing methods, e.g. Non-local method, confirm the effectiveness of the proposed OctConv method. Performance further increases when combining OctConv with the SlowFast Networks [12]. Specifically, we apply OctConv on the spatial dimensions and SlowFast on the temporal dimension.

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 recognition 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.

Acknowledgement. We would like to thank Min Lin and Xin Zhao for helpful discussions of the code development.
References


