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

Paper Author: Yunpeng Chen
Reproducer: Suixin Ou
SUN YAT-SEN UNIVERSITY
ousx@ mail2.sysu.edu.cn

Abstract

Although Convolutional Neural Networks have already got excellent performance on many image classification or prediction tasks, they need huge computing resource. Inspired by image compression technique, the author proposes Octave Convolution (OctConv for short). Images can be compressed by discarding some high frequency info, so are the feature maps. The author factorizes feature maps into high frequency and low frequency parts, and reduces low frequency part's dimension. By adjusting the hyper parameter alpha from 0 to 1, we can control the ratio of low frequency part which will be compressed to half the resolution. In my retrieval for this paper, I replace the convolution in ResNet, ResNeXt and DenseNet with OctConv. I find that OctConv method gets similar or better performance on Tiny ImageNet Dataset than baseline model with less computing resource and storage space. What's more, OctConv method reaches best accuracy which is at least 0.7% higher than baseline model when alpha is around 0.25

1. Introduction

Convolution Neural Networks have achieved remarkable success in many computer vision tasks. The accuracy of classification and prediction tasks becomes higher and higher with the more and more complex model from manually designed networks VGG, DenseNet, ResNet, ResNeXt, ShuffleNet, MobileNet to automatically found networks such as NAS, PNAS and AmoebaNet. The complex network leads to redundancy and computational overhead that cannot be ignored. We should find a method which has both powerful performance and low computation cost. Many researchers have made lots of efforts to reduce redundancy of feature maps without obviously degrading the performance, but all of these methods ignore the redundancy on the spatial dimension of feature maps, and this paper proposes Octave Convolution (OctConv) to deal with the problem, which is orthogonal and complimentary to the previous methods.

1.1. Related Exiting Methods

ResNeXt [1], Xception [2], MobileNet [3], ShuffleNet [4] and CondenseNet [5] reduce the redundancy in CNNs by reducing channel-wise redundancy in convolution feature maps.

[6, 7, 8] reducing redundancy in dense model parameters. DNC [6] do deep network compression learning by in-parallel pruning-quantization. DSD [7] reduce the redundancy in model connections by pruning connections of low weights. ThiNet [8] prunes convolutional filters based on statistics computed to reduce the redundancy in model from its next layer.

1.2. Main Idea of This Paper

To explain the idea of this method, which reduces the redundancy on the spatial dimension of feature maps. We can make an analogy between natural images and feature maps: information of the images is conveyed at different frequencies where higher frequencies are usually encoded with fine details and lower frequencies are usually encoded with global structures, so we can compress the image by dropping some high frequency info. In this way, the image will lose some detail, but we can still get overall meaning for identifying it. Similarly, the output feature maps of a convolution layer can also be seen as a mixture of information at different frequencies. We can divide a convolution layer into high-frequency part and low frequency part, then down sample the low frequency part to reduce the redundancy in networks. At last, the high frequency part contains more detail info while the low frequency part contains more overall info (some detail info is dropped), they are complimentary to each other, which makes the method has both powerful performance and low computation cost.

1.3. Reproduction Experiment Key Result

I use the Tiny ImageNet Dataset to do the retrieval experiment, which was a subset of ImageNet. I apply the OctConv method to ResNet, ResNeXt and DenseNet by replacing the regular convolutions with OctConv. And I adjust hyper parameter α (0<α<1), which denotes the ratio of low frequency part. At last I find that all of Oct-ResNet, Oct-ResNeXt, Oct-DenseNet can have better performance than baseline model under appropriate α. Particularly, they all get better performance than baseline when α=0.25. And at other value for α, the prediction accuracy is similar to baseline while the computation cost is less than baseline. More Detail for my experiment is in 'Experiment' part. All results above indicate that OctConv can improve origin model under overall consideration for both performance and computational cost.

2. Problem formulation

CNNs have achieved great success in many computer vision tasks. There are more and more researchers throwing themselves into this field. More and more powerful models has been proposed, but at the same time, these strong models cost more and more computation resources which is hard to afford in some way. So we should also consider the cost it may take when we design models. In this paper, the author takes this into full consideration and proposes a new way to replace origin convolution, this method brings similar or even better performance compared to baseline model while takes less computation resources.

3. Method

In this section, I will describe how to implement OctConv. Firstly, I will introduce the octave feature representation for reducing the spatial redundancy in feature maps. Secondly, I will describe the Octave Convolution that operates directly on octave feature maps.

3.1. Octave Feature Representation

In common convolution, all the feature maps in the same layer will have same spatial resolution. However, in this method we will divide feature maps into high and low frequency part and reduce the spatial resolution of feature maps in low frequency part. In this way we can reduce the computation cost for the lower resolution in low frequency feature maps. Besides because of lower resolution, low frequency feature maps contain large receptive field, which leads to better performance.

So the first step in this method is to factorize one feature map into two part. The scale space theory [9] provides a way of creating scale spaces of spatial resolutions, and an octave is defined as a division of the spatial dimensions by a power of 2, so the low and high frequency spaces can be obtained by reducing the spatial resolution of the low frequency feature maps by an octave.

Formally, let

$$\mathrm{X} \in R^{c\times h\times w}$$

denote the input feature tensor of a convolutional layer,

$$\alpha\in[0,1]$$

denotes the ratio of channels allocated to the low frequency part, X can be factorized into

$$\mathrm{X} = \{X^H, X^L\}$$

Where

$$X^H \in R^{(1-\alpha)c\times h\times w}$$
$$X^L \in R^{\alpha\times c\times\frac{h}{2}\times\frac{w}{2}}$$

3.2. Octave Convolution

Because Octave feature representation is different from vanilla convolution, we should also change the computation for Octave. So the authors define a new octave convolution to replace the vanilla convolution due to differences in spatial resolution in the octave features, which is expected to not only effectively process the low frequency and high frequency parts in their corresponding frequency tensor, but also enable efficient communication between them.

Formally, let Y denote the output feature tensor of a convolutional layer,

$$Y^{\mathrm{A}\to\mathrm{B}}$$

denotes the convolutional update from feature map group A to group B,

$$\mathbf{W}$$

denote the convolutional kernel, then Y can be factorized into

$$\mathrm{Y} = \{Y^H, Y^L\}$$

Where

$$Y^H = Y^{\mathrm{H}\to\mathrm{H}} + Y^{\mathrm{L}\to\mathrm{H}}$$
$$Y^L = Y^{\mathrm{H}\to\mathrm{L}} + Y^{\mathrm{L}\to\mathrm{L}}$$

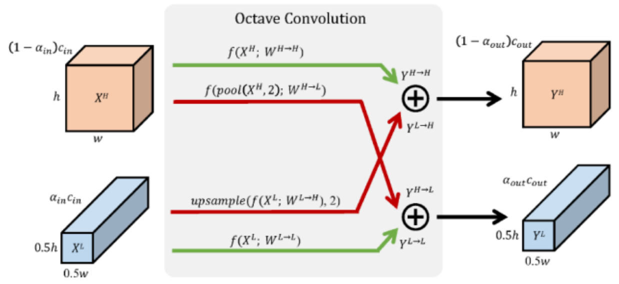
respectively, W can be factorized into

$$\mathrm{W} = \{W^H, W^L\}$$

Where

$$W^H = W^{\mathrm{H}\to\mathrm{H}} + W^{\mathrm{L}\to\mathrm{H}}$$
$$W^L = W^{\mathrm{H}\to\mathrm{L}} + W^{\mathrm{L}\to\mathrm{L}}$$

The total design is shown in the figure below.



In the calculation process, the authors compute the intra frequency update using a regular convolution. But when it comes to the inter frequency communication, folding the up sampling over the feature tensor

$$\mathrm{X}^{\mathrm{L}}$$

or the down sampling of the feature tensor

$$\mathrm{X}^{\mathrm{H}}$$

removing the need of explicitly computing and storing the up sampled feature maps or down sampled feature maps as follows:

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

where $\lfloor\cdot\rfloor$ denotes the floor operation

4. Implementation

Base on the description in this paper and refer to some code for Oct-ResNet, I try to implement OctConv, Oct-ResNet, Oct-ResNeXt and Oct-DenseNet with Pytorch framework. During the implementation, I find that there are some minors in the official code and mark them out in the form if comments.

For the implementation of OctConv, I use four smaller convolutions to finish it: conv_high_to_high, conv_high_to_low, conv_low_to_high, conv_low_to_low, they corresponding to $W^{\mathbf{H}\to\mathbf{H}}$, $W^{\mathbf{H}\to\mathbf{L}}$, $W^{\mathbf{L}\to\mathbf{H}}$, $W^{\mathbf{L}\to\mathbf{L}}$ in the formulas above, and the input feature maps are divided into high and low parts, I denote them as x_high, x_low, then calculate the output feature as the formula in 'Method' part:

y_high = conv_high_to_high*x_high + conv_low_to_high *upsample(x_low),

and it is the same as y_low:

y_low = conv_high_to_low*downsample(x_high) + conv_low_to_low *x_low

and output the { y_high , y_low } together.

That's all, implementing OctConv is not difficult, what difficult is how to apply it to models such as ResNet, DenseNet and so on. And I will describe how I apply OctConv to these models in the following part 'Difficulties & Solutions'

4.1. Difficulties & Solutions

If we want to apply OctConv to a model with vanilla convolution, we should pay attention that feature maps is divided into high and low parts in every layer except first, second and last convolution layer.

For the first convolution layer, its input feature is the source picture, and its channels only mean different color channel, not means frequency, so we should not apply OctConv to it. At early experiment, I have not understood this key point, and could not get good results until I found that source pictures color channels don't have frequency meaning. After this convolution, feature maps can contain frequency meaning, so we can start OctConv at second convolution layer.

For the second convolution, its input feature maps only have one part (it have not been divided yet), so we can regard it as a high part, and no low part, alpha_in=0 so that no $W^{\mathbf{L}\to\mathbf{H}}$ and $W^{\mathbf{L}\to\mathbf{L}}$, so we should judge weather alpha_in =0 and depend existence for $W^{\mathbf{L}\to\mathbf{H}}$ and $W^{\mathbf{L}\to\mathbf{L}}$,

For the last convolution, its output feature maps should only have one part (because we should do an average pool after that convolution), if output feature maps have two parts, we can't do average pooling for it. so we can regard it as a high part, and no low part, alpha_out=0 so that no $W^{\mathbf{H}\to\mathbf{L}}$ and $W^{\mathbf{H}\to\mathbf{L}}$, so we should judge weather alpha_out =0 and depend existence for $W^{\mathbf{H}\to\mathbf{L}}$ and $W^{\mathbf{H}\to\mathbf{L}}$.

If we don't deal with this detail well, the code will have bugs or the final performance will be very bad.

5. Experiments

I use the Tiny ImageNet Dataset to do the retrieval experiment, which was a subset of ImageNet. This dataset selects 20 different classes from ImageNet, the train set has 10000 pictures, 500 pictures for each class, and there are 1000 pictures in the test set, 50 pictures for each class. As what the author did in the paper, I also do the ablation studies for classification on ImageNet (actually a subset of it for time and resource limits), compare the improved model with OctConv with baseline model, and present the result in the following 'Ablation Study On Tiny ImageNet' part.

5.1. Experimental Setups

I apply OctConv to a set off most popular CNNs including ResNet [1], ResNeXt and DenseNet [10] by replacing the regular convolutions with OctConv (except first convolution as we mentioned in '4.1 Difficulties & Solutions' part). All the improved networks only have one global hyper parameter α (0<α<1), which denotes the ratio of low frequency part. I do apple to apple comparison and reproduce all baseline methods (ResNet, ResNeXt and DenseNet) by myself under the same training/testing setting with OctConv model for internal ablation studies. All networks are trained with cross entropy loss. All networks are trained from scratch and optimized by SGD for 40 epochs with exponentially decreasing learning rate every 10 epoch and momentum equal to 0.9. Standard accuracy of single central crop on validation set is reported. What's more, in order to facilitate vertical comparisons between Oct-ResNet, Oct-ResNeXt and Oct-DenseNet, I choose the similar scale for these 3 models (all of them have around 50 layers).

5.2. Ablation Study on Tiny ImageNet

I conduct a series of ablation studies aiming to reproduce the results & conclusion in this paper: 1) OctConv have better accuracy trade off than vanilla convolution 2) When α is 0.25, OctConv work better than other α value (0.5, 0.75).

ResNet-50 vs Oct-ResNet-50 (α = 0.25, 0.5, 0.75). I begin with using the popular ResNet-50 as the baseline CNN and replacing the regular convolution with the proposed OctConv to examine the time-accuracy trade off. In particular, I vary the global ratio α ∈ {0, 0.25, 0.5, 0.75} to compare the image classification accuracy versus computational cost (weighted by time used) with the baseline. The results are as below:

ResNet-50 vs Oct-ResNet-50				
α	0 (baseline)	0.25	0.5	0.75
accuracy	66.93%	68.26%	70.28%	61.88%
time used (s)	1700	7138	6611	5215

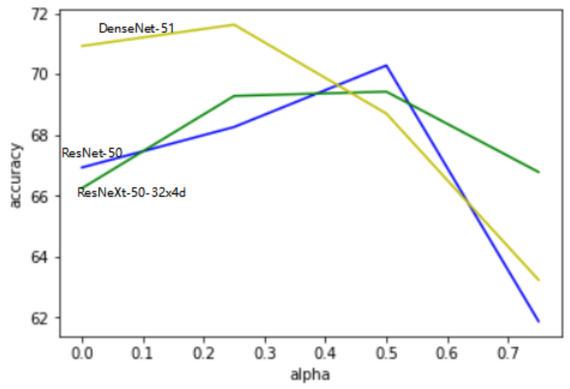
ResNeXt-50-32x4d vs Oct-ResNeXt-50-32x4d (α = 0.25, 0.5, 0.75). To further examine if the proposed OctConv in the paper works for other networks with different topology, I select the ResNeXt-50-32x4d as baselines and repeat the same ablation study. Here is the result:

ResNeXt-50-32x4d vs Oct-ResNeXt-50-32x4d				
α	0 (baseline)	0.25	0.5	0.75
accuracy	66.25%	69.28%	69.42%	66.78%
time used (s)	3052	8529	6342	6305

DenseNet-51 vs Oct-DenseNet-51 (α = 0.25, 0.5, 0.75). At last, I do the same ablation study on DenseNet-51. Here is the result:

DenseNet-51 vs Oct-DenseNet-51				
α	0 (baseline)	0.25	0.5	0.75
accuracy	70.92%	71.62%	68.70%	63.24%
time used (s)	1415	6254	4672	2789

Summarize the results in the three tables above into a fig:



According to result above, we can make the following observation:

Good result & reasons:

1) when we set α from 0 to 1, the accuracy firstly rises up and then goes down, just like a concave curve. This phenomenon is the same as what the authors of this paper find. 2) we can see when α= 0.5, the network can get similar or better result while the theory time used is reduced by about half. When α is around 0.25, the Oct-networks reaches its best accuracy, and get at 0.7% higher than baseline. Combine the analysis in the paper and my understanding, these increase in accuracy can be attributed to OctConv's effective design of multi-frequency processing and the corresponding enlarged receptive field which provides more contextual information to the network.

Bad result & failure analysis:

1) After α= 0.5, the accuracy drops down quickly, which is different from the description in the paper: only 0.4% accuracy drop in the paper while over 4% drop compared to baseline when α=0.75. The defeat of this part may have many possible reasons such as insufficient data, preprocess not good enough, inadequate training (I only train these models for 40 epochs), lack of small but key tricks, and even implement failed. 2) According to the paper, OctConv model can save much computational cost by compressing low frequency feature maps to half the resolution. But in my experiment, the time used for OctConv-model is more than baseline model, which means the computational cost is not reduced. I guess that might because I use four vanilla convolutions to implement one OctConv as mentioned in '4. Implementation' part, and in my code, these four convolutions can't be calculated parallelly, therefore the time used is almost 4 times than baseline. That is because my programming skill is not good enough, and I will try to improve it through more coding and learning in the feature.

6. Conclusion

Through my reproduction experiment, I find that the OctConv method have at least 3 advantages: First of all, it is easy to implement, I just write small amount of code to implement OctConv apply it to baseline model like ResNet. Secondly, this method is powerful, for any model you want to apply it to, you can always find appropriate α to get better performance compared to the baseline model. And in my experiment, I find when α= 0.25 or 0.5, we can get similar or better results. Finally, OctConv won't conflict with other methods improving the network topology or reducing the connection density, it is a complement to them. For example, I can apply OctConv to popular network with well-designed topology such as ResNet in my experiment and don't modify its original structure. The fly in the ointment is that I can't reproduce the time saved against baseline model as described in the paper. I think that is because my programming technique is not good enough so that four convolutions in one OctConv can't be calculated parallelly. I will implement OctConv with parallel ways in the future.

7. Source Code

Source code was in OctConv.ipynb, there are also some visible experiment results in the notebook.

References

[1] S. Xie, R. Girshick, P. Dolla´r, 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

[2] 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

[3] 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

[4] 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

[5] 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.

[6] 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

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

[8] 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.

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

[10] 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