ShuffleNet for Mobile Devices

Hongzhi Liu

Jun 10, 2018

Abstract

Building deeper and larger convolutional neural networks (CNNs) is a primary trend for solving major visual recognition tasks. The most accurate CNNs usually have hundreds of layers and thousands of channels, thus requiring computation at billions of FLOPs. Today, I read a thesis written by Xiangyu Zhang, who is from Megvii Inc (Face++). His team introduce an extremely computation-efficient CNN architecture named ShuffleNet which is designed specially for mobile devices with very limited computing power. The new architecture utilizes two new operations, pointwise group convolution and channel shuffle to greatly reduce computation cost while maintaining accuracy.

1. Overview of ShuffleNet

The last few years have seen the success of deep neural networks in computer vision tasks, in which model designs play an important role. The increasing needs of running high quality deep neural networks on embedded devices encourage the study on efficient model designs [2]. Concurrent with them, a very recent work [10] employs reinforcement learning and model search to explore efficient model designs. The proposed mobile *NASNet* model achieves comparable performance with their counterpart ShuffleNet model.

The concept of group convolution, which was first introduced in *AlexNet* [6] for distributing the model over two GPUs because a single block of GPU GTX 580 graphics cards is only 3GB that could not be put down the complete model, has been well demonstrated its effectiveness in ResNeXt [8]. To overcome the side effects brought by group convolutions, the team come up with a novel *channel shuffle* operation to help the information flowing across feature channels. Based on the two techniques, they build a highly efficient architecture called *ShuffleNet* [9]. Their work generalizes group convolution and depthwise separable convolution.

2. ShuffleNet Models

The core idea of ShuffleNet lies in pointwise group convolution and channel shuffle operation. In tiny networks, expensive pointwise convolutions result in limited number of channels to meet the complexity constraint, which might significantly damage the accuracy. To address the issue, a straightforward solution is to apply group convolutions. Channel shuffle operation makes it possible to build more powerful structures with multiple group convolutional layers.

2.1. Group Convolutions

Modern convolutional neural networks [4] usually consist of repeated building blocks with the same structure. Among them, state-of-the-art networks such as Xception [1] and ResNeXt [8] introduce efficient group convolutions into the building blocks to strike an excellent trade-off between representation capability and computational cost. However, Zhang notices that both designs do not fully take the 1×1 convolutions into account, which require considerable complexity.

To address the issue, a straightforward solution is to apply group convolutions. By ensuring that each convolution operates only on the corresponding input channel group, group convolution significantly reduces computation cost. However, if multiple group convolutions stack together, there is one side effect that is outputs from a certain channel are only derived from a small fraction of input channels. Fig. 1 (a) illustrates a situation of two stacked group convolution layers. It is clear that outputs from a certain group only relate to the inputs within the group. This property blocks information flow between channel groups and weakens representation.

2.2. Channel Shuffle

If we allow group convolution to obtain input data from different groups as shown in Fig. 1 (b), the input and output channels will be fully related. Specifically, for the feature map generated from the previous group layer, we can first divide the channels in each group into several subgroups,

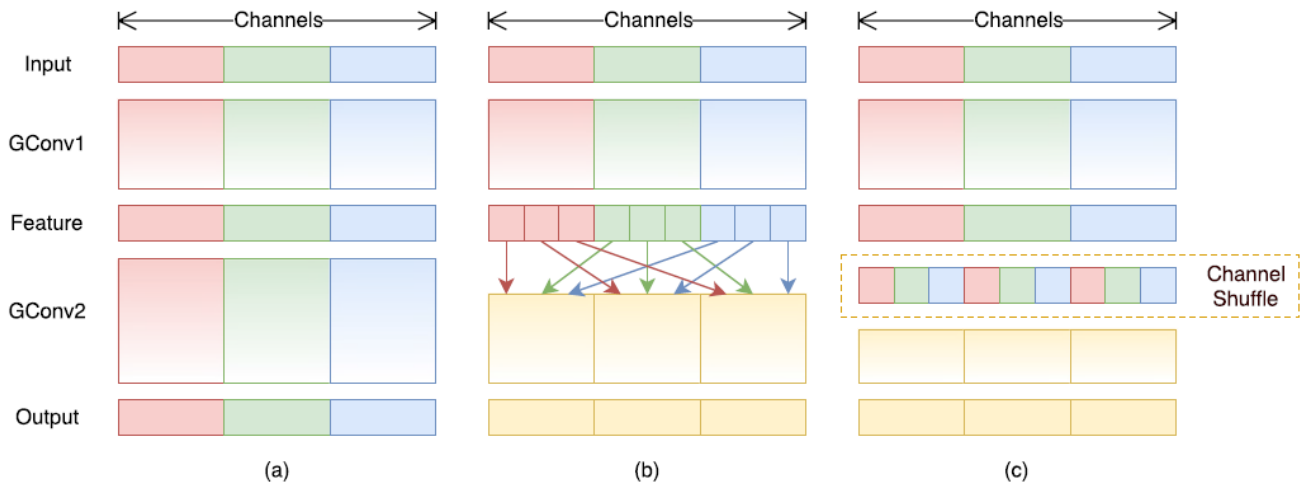

Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle.

then feed each group in the next layer with different subgroups. This can be efficiently and elegantly implemented by a channel shuffle operation as shown in Fig. 1 (c): suppose a convolutional layer with g groups whose output has $g \times n$ channels; the team first reshape the output channel dimension into (g, n), transposing and then flattening it back as the input of next layer. Moreover, channel shuffle is also differentiable, which means it can be embedded into net-

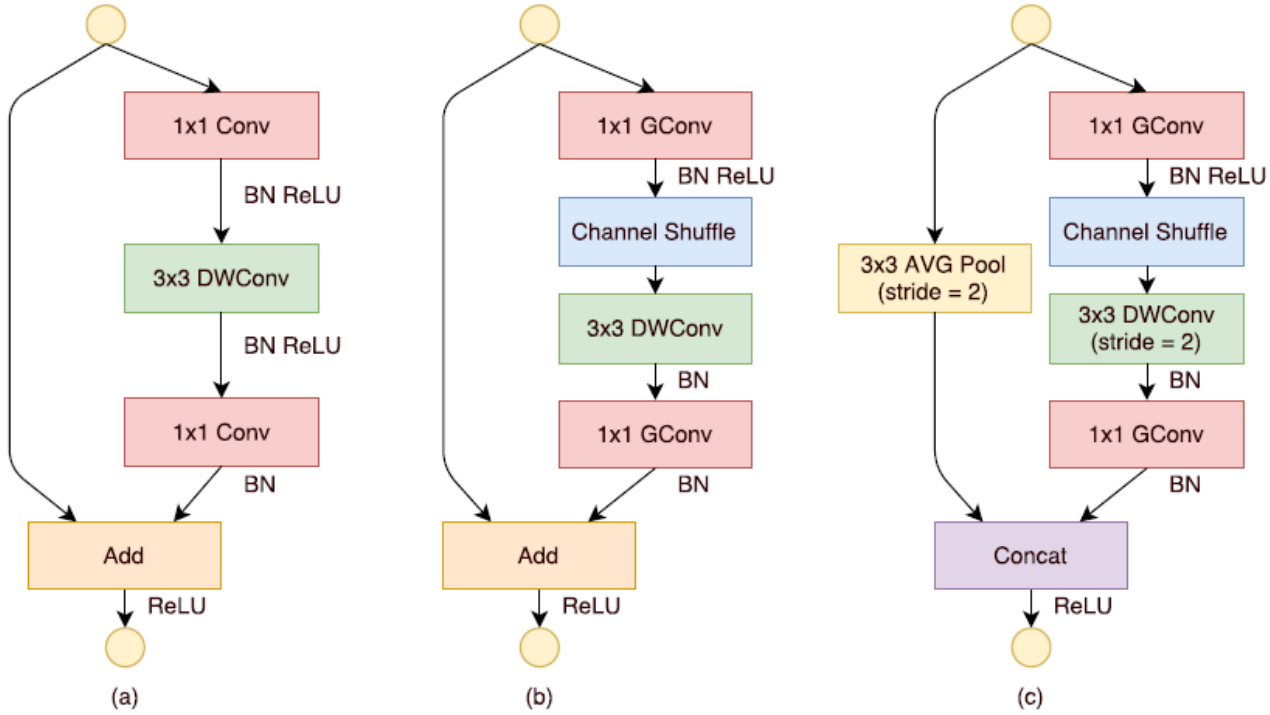

Figure 2. ShuffleNet Units. a) bottleneck unit [3] with depthwise convolution (DWConv) [5]; b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2.

Table 1. ShuffleNet architecture. The complexity is evaluated with FLOPs, *i.e.* the number of floating-point multiplication-adds. Note that for Stage 2, they do not apply group convolution on the first pointwise layer because the number of input channels is relatively small.

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					g=1	g=2	g=3	g=4	g=8
Image	224×224	-	-	-	3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2	-	-	-	-	-	-
Stage2	28×28	-	2	1	144	200	240	272	384
	28×28	-	1	3	144	200	240	272	384
Stage3	14×14	-	2	1	288	400	480	544	768
	14×14	-	1	7	288	400	480	544	768
Stage4	7×7	-	2	1	576	800	960	1088	1536
	7×7	-	1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7	-	-	-	-	-	-	-
FC	-	-	-	-	1000	1000	1000	1000	1000
Complexity	-	-	-	-	143M	140M	137M	133M	137M

Table 2. Classification error vs. number of groups g

Model	Complexity (MFLOPs)	Classification error (%)				
		g=1	g=2	g=3	g=4	g=8
ShuffleNet $1\times$	140	33.6	32.7	32.6	32.8	32.4
ShuffleNet $0.5\times$	38	45.1	44.4	43.2	41.6	42.3
ShuffleNet $0.25\times$	13	57.1	56.8	55.0	54.2	52.7

work structures for end-to-end training. Channel shuffle operation makes it possible to build more powerful structures with multiple group convolutional layers.

2.3. ShuffleNet Unit

Taking advantage of the channel shuffle operation, Zhang and his team propose a novel ShuffleNet unit specially designed for small networks. They start from the design principle of bottleneck unit [3] in Fig. 2 (a). It is a residual block. In its residual branch, for the 3×3 layer, they apply a computational economical 3×3 depthwise convolution [1] on the bottleneck feature map. Then they replace the first 1×1 layer with pointwise group convolution followed by a channel shuffle operation, to form a ShuffleNet unit as shown in Fig. 2 (b). The purpose of the second pointwise group convolution is to recover the channel dimension to match the shortcut path. As for the case where ShuffleNet is applied with stride, they simply make two modifications (see Fig. 2 (c)): (i) add a 3×3 average pooling on the shortcut path; (ii) replace the element-wise addition with channel concatenation, which makes it easy to enlarge channel dimension with little extra computation cost. Thanks to pointwise group convolution with channel shuffle, all components in ShuffleNet unit can be computed efficiently.

2.4. Network Architecture

Built on ShuffleNet units, Zhang presents the overall ShuffleNet architecture in Table 1. The proposed network is mainly composed of a stack of ShuffleNet units grouped into three stages. Intent of his team is to provide a reference design as simple as possible, although they find that further hyper-parameter tunning might generate better results. In ShuffleNet units, group number g controls the connection sparsity of pointwise convolutions. Table 1 explores different group numbers and we adapt the output channels to ensure overall computation cost roughly unchanged. Obviously, larger group numbers result in more output channels for a given complexity constraint which helps to encode more information.

3. Evaluation of Models

The team mainly evaluate their models on the ImageNet 2012 classification dataset [7]. They follow most of the training settings and hyper-parameters used in [8]. To benchmark, they compare single crop top-1 performance on ImageNet validation set, *i.e.* cropping 224×224 center view from $256\times$ input image and evaluating classification accuracy. They use exactly the same settings for all models to ensure fair comparisons.

To evaluate the importance of pointwise group convolu-

tions, Zhang and his team compare ShuffleNet models of the same complexity whose numbers of groups range from 1 to 8. If the group number equals 1, no pointwise group convolution is involved. For better understanding, they also scale the width of the networks to 3 different complexities and compare their classification performance respectively. Results are shown in Table 2.

From the results, we see that models with group convolutions (g > 1) consistently perform better than the counterparts without pointwise group convolutions (g = 1). Smaller models tend to benefit more from groups. Table 2 also shows that for some models (*e.g.* ShuffleNet $0.5\times$) when group numbers become relatively large (*e.g.* $g = 8$), the classification score saturates or even drops. Furthermore, they also notice that for smaller models such as ShuffleNet $0.25\times$ larger group numbers tend to better results consistently, which suggests wider feature maps bring more benefits for smaller models.

References

[1] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016. 1, 3

[2] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *CVPR*, 2015. 1

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3

[4] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 1

[5] 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. 2

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 3

[8] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 3

[9] X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017. 1

[10] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017. 1