**AugShuffleNet: Improve ShuffleNetV2 with more information communication**

LongQing Ye

**Abstract**

Based on ShuffleNetV2, we build a more powerful and efficient model family, termed as AugShuffleNets, by introducing higher frequency of cross-layer information communication for better model performance. Evaluated on the CIFAR-10 and CIFAR-100 datasets, AugShuffleNet consistently outperforms ShuffleNetV2 in terms of accuracy, with less computational cost, fewer parameter count.

1 Introduction

Since the success of AlexNet [1], convolutional neural networks (CNNs, or ConvNets) have become dominant in various vision tasks including image classification, object detection and semantic segmentation. Typical models including VGG, GoogLeNet, Resnet [2-4] achieve remarkable performance. However, the accuracy improvement usually involves scaling up network in the form of more layers or more channels per layer, further leading to an increasing amount of computational cost and parameters. In recent years, many real-time CNN-based applications are deployed on resource-constrained platforms such as sensors and smart phones, requiring CNN models to be computationally efficient and of rapid response. How to design efficient models becomes an important research topic.

Many efforts has been dedicated to improve the efficiency of ConvNets. Currently, there are two main representative and complementary schemes: model compression and compact model design. The former kind aims to reduce the model redundancy without significant degeneration in performance via pruning, quantization, low-rank, etc. However, the upper limit of model performance tends to be determined by pre-trained networks. The latter kind build compact models trained from scratch. Typical instances like MobileNetV1-V3 and ShuffleNetV1-V2 utilize efficient operators, building block or network architecture search algorithms, providing significant insights for compact model design.

In this paper, we revisit the design of building block in ShuffleNetV2 and design a more efficient and powerful CNN model named AugShuffleNet

2 Related Work

2.1 Compact Model Design

Composition of low-rank (1 × 3, 3 × 1) filters [5] or sparse filters [6] are proposed to approximate dense convolution filter. Those work indicates that less redundant filters could bring a great reduction of FLOPs and parameters while maintaining the performance of models. Group convolution has been viewed as a standard operator in modern compact models [6][7]. Depth-wise convolution is an extreme form of group convolution, in which each channel presents a group. ShuffleNetV1 [6] uses group convolution to replace 1 × 1 convolution and further introduces the operation of "Channel Shuffle" to improve cross-group information communication. MobilenetV1 [8] utilizes depth-wise convolution and pointwise convolution to construct a lightweight model for mobile platforms. ResNet [4] and MobileNetV2 [9] adopt a bottleneck structure to alleviate the burden of heavy computation for channel expansion. MobileNetV3 and other work [10][12] introduce the neural architecture search algorithms into compact model design, significantly reducing the cost of manual design.
2.2 Model Compression

Complementary to compact model design, model compression is another approach to further shrink pre-trained models. Network pruning [13] removes redundant and non-informative connections or channels. Model quantization [14] aims to represent stored weights at a low cost for model compression and calculation acceleration. Knowledge distillation [15] transfers refined knowledge from “teacher network” into “student network”, simplifying the process of suppressing redundant information. In addition, efficient convolution algorithms like FFT [16] and winograd [17] are explored to speed up the implement of convolutional layer without any modification of network design.

2.3 Short Connection

ResNets and Highway Networks [18] introduce skip connections to allow training deeper neural networks. Deep networks with stochastic depth [19] reduces training depth of ResNets by randomly skipping layers. SkipNet further utilizes gating mechanism to learn how to skip layers for both training and inference procedures.

Residual connection proposed by ResNet has been a popular technique to construct deep neural networks [7, 20] in vision and natural language processing domain. Research [21] regards ResNet as a collection of many paths of different length and discovers that deep ResNet updates weights mainly via short paths during the training procedure. DenseNet [22] introduces densely connections by connecting each layer to every other layer in the networks to increase more short path. ShuffleNetV2 [23] combines “Channel Split” and “Channel Shuffle” operations to achieve cross-layer information communication, resulting in a similar effect of DenseNet. All of those work enjoy the benefit of short paths to alleviate “gradient vanishing” problem in the training process.

3 Approach

We introduce two global hyper-parameter: communication frequency and split ratio to this end. Higher communication frequency could utilize more intermediate information from neural layers and increase more short connections. Split ratio is used to adjust the computational cost of inference network.

3.1 Rethink Shuffle Block in ShuffleNetV2

Distinct from common ConvNets, ShufflenetV2 does not use residual connection and adopts split-transform-fuse strategy to trade off efficiency and performance for model design. Illustrated in Figure 1a the whole pipeline of shuffle block in ShuffleNetV2 could be concluded as three stages:

Split: Channels of feature maps are split into two parts. One part is conserved, another is fed into transformation stage for high-level information. Note that, the conserved part is named “Feature Bank” in this paper. The number of feature maps should be equal to the split ratio.
maps in "Feature Bank" is determined by the parameter split ratio. It is seen that split ratio is a fixed parameter 0.5 in ShuffleNetV2.

**Transform**: Transformation stage consists of three layers: $1 \times 1$ regular convolution, $3 \times 3$ depth-wise convolution and regular $1 \times 1$ convolution in order, serving as a learnable "Feature Extractor".

**Fuse**: After transformation is performed, feature maps come from feature bank and transformation stage would be merged by concatenating along channel-wise dimension. Then, those merged channels would be rearranged in an interleaving way, which is called "Channel Shuffle". "Channel Shuffle" enables information communication between the two branches.

The success of ShuffleNetV2 may lie in its use of "Feature Bank" via which to control the information communication across different layers. Essentially, "Feature Bank" could be regarded as auxiliary memory component where feature maps are stored. "Channel Split" and "Channel Shuffle" are two main interactive operations between "Feature Bank" and "Feature Extractor" to achieve cross-layer information communication. Partial Channels from shallow layers could be conserved in "Feature Bank" and periodically fused into deeper layers of the network, achieving a similar effect of feature reuse in DenseNet [22].

### 3.2 Improved Shuffle Block with More Information Communication

Benefited from "Feature Bank" with "Channel Split" and "Channel Shuffle" operations, ShuffleNetV2 shows expressive balance between performance and efficiency. However, it is noticed that there still exists remaining room to strike a better balance:

1. For ShuffleNetV2, information communication is only limited to the exit of the shuffle block via the operation of "Channel Shuffle". Intermediate information from the first and second layer of shuffle block is not well exploited.
2. In ShuffleNetV2, keeping the same number of channels for every layer in the transformation stage is not an absolute principle for the lowest memory access cost. When the width (the number of channels) of network increases, shuffle block in ShuffleNetV2 still would produce more channel-wise redundancy.

We propose another more powerful and efficient shuffle block illustrated in Figure[1b]. Compared to the original shuffle block shown in[1a] this paper makes following modifications:

- **M1.** Split ratio $r$ is set as a variable, which could flexibly adjust the number of channels fed into "Feature Extractor" and further control the efficiency of the whole building block in terms of computational cost, parameter consumption and inference speed. For simplicity, the first and second layers ($1 \times 1$ regular convolution and $3 \times 3$ depth-wise convolution) keep the same number of channels for input and output.

- **M2.** We introduce another operation termed "Channel Crossover" to exchange information after the depth-wise convolutional layer. "Channel Crossover" deposit partial new feature maps into "Feature Bank" and withdraw more old feature maps to achieve channel-wise expansion without computational cost. By adding interaction mechanism between intermediate layers and "Feature Bank", information is fully utilized.

M1 and M2 work together to improve efficiency and make model gain better representational ability. when $r < 0.5$, the first and second layers in shuffle block would be more efficient than original shuffle block.

### 4 Network Design

The architecture of AugShuffleNet is shown in Table[1] basically following the configuration of ShuffleNetV2.

### 5 Experiment and Result

**Datasets** All models are evaluated on the datasets CIFAR-10 and CIFAR-100. The two CIFAR datasets consist of colored natural images with $32 \times 32$ pixels. CIFAR-10 consists of images drawn from 10 classes and CIFAR-100 from 100 classes. The training and test sets contain 50,000 and 10,000 images respectively for both two datasets. Images in the training set are augmented by random horizontal flip and random crop (4 pixels are padded on each side, and a $32 \times 32$ crop is randomly sampled from the padded image).

**Training Settings** All models are trained 300 epochs using cosine learning rate decay for both of CIFAR-10 and CIFAR-100. The initial learning rate is set as 0.1. we adopt Stochastic Gradient Descend (SGD) optimizer (momentum parameter is 0.9, nesterov is set to True) with the batch size of 128. The weight decay is set to 5e-4 for regularization.
Table 1: Architecture of AugShuffleNet

| Layer            | Output size | KSize | Stride | Repeat | Output channels |
|------------------|-------------|-------|--------|--------|-----------------|
| Image            | 32 × 32     |       |        |        |                 |
| Conv1            | 32 × 32     | 3 × 3 | 1      | 1      | 24              |
| Stage2 DownBlock | 16 × 16     | 2     | 1      | 120    | 176             |
| BasicBlock1      | 8 × 8       | 2     | 1      | 240    | 352             |
| BasicBlock2      | 4 × 4       | 2     | 1      | 480    | 704             |
| Conv2            | 4 × 4       | 1 × 1 | 1      | 1024   | 1024            |
| GlobalPool       | 1 × 1       |       |        |        |                 |
| FC               |             |       |        | num_classes| num_classes |

Table 2: Comparison of models on Cifar10

| Model            | Params | FLOPs  | Acc (%)     | #run1 | #run2 | #run3 | #run4 | #run5 | #Average |
|------------------|--------|--------|-------------|-------|-------|-------|-------|-------|----------|
| ShuffleNetV2 1.5x| 2.49M  | 94.27M | 93.93      | 93.96 | 93.86 | 93.98 | 94.11 | 93.97 |
| ShuffleNetV2 1.0x| 1.26M  | 45.01M | 93.12      | 93.21 | 93.17 | 93.30 | 93.42 | 93.24 |
| Ours(Large)      | 2.22M  | 85.38M | 94.31      | 94.41 | 94.21 | 94.44 | 94.62 | 94.40 |
| Ours(small)      | 1.21M  | 43.56M | 93.63      | 93.78 | 93.95 | 93.96 | 93.87 | 93.84 |

5.1 Comparison to Other Models

To verify our approach, we choose ResNet and ShuffleNetV2 as baselines, which are classic backbone networks. All models are trained in the same way for fair comparison. Extensive experiments are conducted on image classification datasets Cifar10 and Cifar100. The result for every model are obtained via 5 runs to ensure the reliability of experiments. Result are reported in Table 2 and Table 3. It is seen that our model consistently outperforms other models by a large margin with better efficiency in terms of computational cost and parameter count.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.
[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.
[5] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866, 2014.

Table 3: Comparison of models on Cifar100

| Model            | Params | FLOPs  | Acc (%)     | #run1 | #run2 | #run3 | #run4 | #run5 | #Average |
|------------------|--------|--------|-------------|-------|-------|-------|-------|-------|----------|
| ShuffleNetV2 1.5x| 2.58M  | 94.36M | 74.53       | 74.66 | 74.83 | 74.76 | 74.70 | 74.70 |
| ShuffleNetV2 1.0x| 1.36M  | 45.10M | 73.24       | 73.06 | 72.91 | 72.70 | 73.49 | 73.08 |
| Ours(Large)      | 2.32M  | 85.47M | 75.65       | 75.75 | 75.98 | 76.44 | 75.75 | 75.91 |
| Ours(small)      | 1.30M  | 43.65M | 74.07       | 74.10 | 74.31 | 74.45 | 73.88 | 74.16 |
[6] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian 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.

[7] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1492–1500, 2017.

[8] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520, 2018.

[10] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In Proceedings of the IEEE International Conference on Computer Vision, pages 1314–1324, 2019.

[11] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791, 2019.

[12] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

[13] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 1389–1397, 2017.

[14] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4820–4828, 2016.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[16] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. arXiv preprint arXiv:1312.5851, 2013.

[17] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4013–4021, 2016.

[18] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. arXiv preprint arXiv:1505.00387, 2015.

[19] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In European conference on computer vision, pages 646–661. Springer, 2016.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[21] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. Advances in neural information processing systems, 29:550–558, 2016.

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

[23] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European conference on computer vision (ECCV), pages 116–131, 2018.