**AugOp: Inject Transformation into Neural Operator**

**Longqing Ye**

**ABSTRACT**

In this paper, we propose a simple and general approach to augment regular convolution operator by injecting extra group-wise transformation during training and recover it during inference. Extra transformation is carefully selected to ensure it can be merged with regular convolution in each group and will not change the topological structure of regular convolution during inference. Compared with regular convolution operator, our approach (AugConv) can introduce larger learning capacity to improve model performance during training but will not increase extra computational overhead for model deployment. Based on ResNet, we utilize AugConv to build convolutional neural networks named AugResNet. Result on image classification dataset Cifar-10 shows that AugResNet outperforms its baseline in terms of model performance.

1 Introduction

![Figure 1: The schema of AugConv in the channel perspective](image)

In recent years, convolutional neural network (CNN) has achieved remarkable success in natural language processing, audio processing and computer vision domain. Deeper and wider neural networks [1–3] have been proposed to pursue higher performance at the cost of heavier computational overhead. Meanwhile, convolutional layer is recognized as a basic neural unit and stacking it repeatedly is still a common way to build CNNs. Therefore, convolutional layer almost dominates the computational overhead of a CNN model and how to develop a more powerful convolution operator is a crucial research topic.

It known that convolutional layer is a special case of fully connected layer, characterized by local connections and parameter-sharing mechanism. However, we observe that convolutional layer still follows fully connected topological structure in the channel dimension, which constrains the efficiency of wide CNNs. Group convolution alleviates this
issue by introducing more sparse channel-wise connections, greatly reducing the computational cost and parameter count. Depth-wise convolution with spatial size of $K \times K$, an extreme case of group convolution, combined with $1 \times 1$ regular convolution is widely applied in compact models including ShuffleNetV2 and MobileNets series [4–7] to approximate the expressivity of a regular convolution with spatial size of $K \times K$. Meanwhile, $1 \times 1$ regular convolution occupies the most of computational overhead in such models and still plays a key role in compact models which are deployed on CPU or GPU devices.

This paper aims to develop a more powerful variant of regular convolution which can utilize local channel-wise information as group convolution. To this end, as illustrated in Figure 1, we propose a quite simple but general design pattern to improve regular convolution operator:

1. During training, we first split feature maps along channel dimension into $g$ groups.
2. We then perform regular convolution and extra transformation over every group of input channels independently. Here, every convolution operator is only responsible for partial channels as group convolution.
3. During inference, we fuse all regular convolution operators and transformations across all groups into a single regular convolution operator with global channel-wise receptive field.

Modified convolution operator is called AugConv in this paper. Compared with original regular convolution, AugConv owns more powerful representivity as it introduces more trainable parameters via extra transformation during training and more local channel-wise information is considered. Another attractive property of AugConv is that its computational overhead for deployment exactly equals to corresponding regular convolution.

2 Related Work

Figure 2: Topological structure of different convolution operators in the channel dimension.

Group Convolution: Figure 2 shows channel-wise topological structure of different convolution operators. It seen that regular convolution has extremely dense connections in the channel dimension, which will lead to high consumption of computational resource when CNN becomes wide. Group convolution is equivalent to a collection of regular convolutions over local channels. It is used in AlexNet [1] to solve memory issue and applied in ResNeXt [8] to increase network width in an efficient way. Depth-wise convolution (DWConv) is a special group convolution where each group contains only a single channel and widely applied in compact models to extract spatial information due to its low computational cost in terms of computational cost and parameter count.

Re-parameterizing Structure: The basic unit of RepVGG [9] constructs three parallel branches before activation function: $3 \times 3$ regular convolution, $1 \times 1$ regular convolution and residual connection during training and fuses them into a single $3 \times 3$ regular convolution during inference. Here, residual connection can be regarded as a special $3 \times 3$ convolutional kernel [10]. Therefore, if ignoring batch normalization layers over all branches, the basic unit of RepVGG can be viewed as a static version of CondConv [11] whose kernel is a mixture of multiple convolutional kernels with different kernel size. RepOpt [12] also adopts similar strategy to build neural network and removes multi-branch structure. Instead of using multiple convolutional kernels, this paper chooses to decouple a large convolutional kernel into a collections of small kernels in the form of multiple convolution branches over different local channels with extra transformations. The goal of this paper is to explore the performance gain extra transformation can bring and provides a new tool to augment current neural operators.
3 Approach

3.1 Rethink Neuron Model

As illustrated in Figure 3, neuron activity can be modelled as weighted sum of its all inputs, which has exerted a profound influence on modern neural networks. Neural operators including fully connected, convolutional and self-attention layers can be regarded as complex extensions of a basic neuron. A neuron model can be formulated as:

\[ y = A\left(\sum_{i=1}^{n} w_i x_i\right) \]  

(1)

where \( w_i \) denotes associated weight for every input signal \( x_i \), \( A \) is a nonlinear function, \( n \) represents the number of input signals.

In this paper, we construct a similar formula as follows:

\[ y = A\left(f_1\left(\sum_{i=1}^{n/g} w_i x_i\right) + f_2\left(\sum_{i=n/g+1}^{2n/g} w_i x_i\right) \ldots + f_g\left(\sum_{i=n-n/g+1}^{n} w_i x_i\right)\right) \]

(2)

Ignoring the nonlinear function \( A \), we split \( n \) input signals into \( g \) groups (\( n \) is divisible by \( g \)), then perform transformation\((f_i)\) over every group, finally sum up results of all groups. Here, transformation \( f_j \) and weights within each group can be merged into another group of weights, for example, the fist group satisfies:

\[ f_1\left(\sum_{i=1}^{n/g} w_i x_i\right) = \sum_{i=1}^{n/g} w'_i x_i \]  

(3)

Other groups follow a similar pattern. It’s seen that the right of Equation 1 shares the same form with the last line of Equation 2.

3.2 AugConv

As most of neural operators such as fully connected layer, convolutional layer and self-attention layer can be regarded as various clusters of neurons and involve the operation of "weighted sum"(dot product), we can insert the auxiliary transformation \( f \) to enhance them during training and turn them into normal operators during inference. The modification allows neural network to learn more potential information.
In order to better understand above approach, we take convolutional layer as an example and choose batch normalization (BN) as auxiliary transformation $f$. Suppose a convolutional layer receives $n$ input feature maps with the spatial size of $H \times W$, its output can be formulated as:

$$Y = BN(F_1 \otimes X_1) + BN(F_2 \otimes X_2) \cdots + BN(F_g \otimes X_g)$$

(4)

Here, input feature maps are split into $g$ groups along channel-wise dimension and $X_i$ corresponds to $i^{th}$ group of feature maps ($n$ is divisible by $g$). $F_i$ represents convolutional weights for $i^{th}$ group and $\otimes$ denotes regular convolution operator. It is known that convolution and batch normalization can be merged into a single operator (another convolution with different weights), which is a common technique in deployment of CNN models. Therefore, we can obtain following formula:

$$Y = (F_1' \otimes X_1) + (F_2' \otimes X_2) \cdots + (F_g' \otimes X_g)$$

(5)

In addition, the summation of convolutions over all groups is equivalent to a single convolution with large convolutional kernel. We can further modify above formula:

$$Y = Concat[F_1, F_2, \cdots F_g] \otimes Concat[X_1, X_2, \cdots X_g]$$

(6)

$Concat$ will concatenate data along channel dimension. The convolution operator with extra transformation $f$ is called AugConv, as illustrated in Figure 1. Batch normalization is just a special case of transformation $f$ and can introduce more statistical information for every group.

### 4 Result

| Model           | Params       | MAdds       | Acc (%) |
|-----------------|--------------|-------------|---------|
| ResNet-20       | 275.6K/275.6K | 40.6M/40.6M | 91.88   |
| AugResNet-20 (g=2) | 275.6K/276.9K | 40.6M/40.6M | 92.40   |
| ResNet-56       | 858.9K/858.9K | 125.5M/125.5M | 93.34   |
| AugResNet-56 (g=2) | 858.9K/862.9K | 125.5M/125.5M | 93.44   |

Table 1: Performance of models on CIFAR-10. Params: parameter count during inference/training, MAdds: computational cost during inference/training.

**Datasets** All models are evaluated on the dataset Cifar-10. Cifar-10 consists of colored natural images with $32 \times 32$ pixels drawn from 10 classes. The training and test sets contain 50,000 and 10,000 images respectively. Images in the training set are augmented by random horizontal flip and random crop (4 pixels are padded on each side, and a 32x32 crop is randomly sampled from the padded image).

**Training Settings** All models are trained 300 epochs using cosine learning rate decay. 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 1e-4.

To demonstrate the effectiveness of our approach, we select ResNet-20 (20 layers) and ResNet-56 (56 layers) as baselines. We build a new model family named AugResNets by replacing all regular convolution (except the first convolutional layer) of ResNets with AugConv. For AugConv operator, we set $g = 2$. As show in Table 1, AugResNets obviously outperforms ResNets in terms of accuracy on Cifar-10. The result is not surprising as we introduce more parameters for AugResNets via extra transformations (batch normalization layers) within convolution operator during training. However, compared with ResNets, the extra overhead (parameter count and computational cost) of AugResNets during training can be ignored. During the inference, AugResNet and ResNet have the same amount of computational overhead as AugConv turns into a regular convolution via operators fusion. It means that our approach can improve model performance without extra deployment overhead for models but at the cost of a little more negligible training overhead and more training time due to more branches during training.
5 Conclusion

This paper presents a new design pattern to improve neural operators and demonstrates the advantage of AugConv to build convolutional neural network. Actually, batch normalization is not the only choice of transformation for AugConv. Any transformation which can be merged into original operators deserves to be explored and different transformations can be used for all groups at the same time to form heterogeneous structure. We can also adopt similar strategy to modify self-attention layer since it involves the operation of “weighted sum”. Furthermore, we can apply our approach in a recursive way to form fractal structures for neural operators, which will be explored in the feature.

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] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.
[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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 1492–1501, 2017.
[9] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13733–13742, 2021.
[10] Sergey Zagoruyko and Nikos Komodakis. Diracnets: Training very deep neural networks without skip-connections. arXiv preprint arXiv:1706.00388, 2017.
[11] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. Advances in Neural Information Processing Systems, 32, 2019.
[12] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. Re-parameterizing your optimizers rather than architectures. arXiv preprint arXiv:2205.15242, 2022.
[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.
[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pages 448–456. PMLR, 2015.
[15] Timothy I Murphy. Line spacing in latex documents. https://nenadmarkus.com/p/fusing-batchnorm-and-conv/. Accessed April 4, 2010.