Final project, Advanced Parallel Computing(240199), Fall, 2020

Dynamic Channel-wise Convolution: Channel-wise Attention over Convolution Kernels

JUN-UK JUNG[1]

[1]School of Computer Engineering, Korea University of Technology and Education, Cheonan-si, Rep. of Korea (e-mail: rnans33@koreatech.ac.kr)

Course Instructor: Duksu Kim (e-mail: bluekds@koreatech.ac.kr)

This is the report for the final project of the advanced parallel computing course in 2020 at the school of computer engineering, Korea University of Technology and Education (KOREATECH).

ABSTRACT Convolution neural networks are endlessly weighed up the pros and cons of either performance or computational budgets. In terms of lightweight, it has built efficient neural networks by re-designing the convolution operator and constraining representation capability, even at the cost of performance drop. To address this issue, we present Dynamic Channel-wise Convolution, a reformed Dynamic Convolution [1], that improves representation ability without increasing computation cost. Dynamic channel-wise convolution applies each output channel-specific attention in the process of aggregating multiple parallel convolution kernels. By simply replace dynamic convolution with dynamic channel-wise convolution for the architecture MobileNetV2 [2], the accuracy of CIFAR100 classification is boosted by 1.03% with only 0.01% additional GMacs. We also propose forward cuda optimization for dynamic channel-wise convolution available in depthwise convolution. it is faster than implemented with pytorch at high-resolution images.

INDEX TERMS Deep learning, Computer Vision and Pattern Recognition, Parallel Computing

JUN-UK JUNG Bachelor in Computer Engineering Korea University of Technology and Education(KOREATECH) 2013. Master(Enrolled) in Computer Engineering, Korea University of Technology and Education(KOREATECH), 2020. RESEARCH INTERESTS: Deep learning, Computer Vision and Pattern Recognition.

I. INTRODUCTION

THERE has been an explosion of interest in building lightweight and efficient neural networks. To provides optimized speed and inference by such as lightweight devices, mobile devices, industrial gateways, and IoT sensors, it commonly constrains both the depth (number of convolution layers) and the width (number of channels) of CNNs. Decreasing CNNs depth and width has faster speed but caused performance degradation because of resulting in limited representation capability. Many works(e.g. MobileNet [2] and ShuffleNet [3]) have shown that efficient operator designs (e.g. depthwise convolution, channel shuffle, squeeze-and-excitation [4], asymmetric convolution [5]) are important for preventing performance degradation and for efficient neural networks.

Recently, dynamic convolution [1] has achieved an enhanced performance than convolution and depthwise convolution without increasing the network depth or width. Instead of using a single convolution kernel per layer, dynamic convolution aggregates multiple parallel convolution kernels dynamically based upon their attentions, which are input dependent. Dynamic convolution uses a set of K parallel convolution kernels $\{\tilde{\boldsymbol{W}}_k, \tilde{\boldsymbol{b}}_k\}$ instead of using a single convolution kernel per layer. These convolution kernels are aggregated dynamically $\tilde{\boldsymbol{W}} = \sum_k \pi_k(\boldsymbol{x})\tilde{\boldsymbol{W}}_k$ for each individual input $\boldsymbol{x}$ via input dependent attention $\pi_k(\boldsymbol{x})$. The biases are aggregated using the same attention $\tilde{\boldsymbol{b}} = \sum_k \pi_k(\boldsymbol{x})\tilde{\boldsymbol{b}}_k$. Because of aggregating K more kernels, dynamic convolution has more representation power than static convolution.

However, dynamic convolution has a problem compressing the massive kernels. Dynamic convolution kernel size is too high to classify core information with only K attention probability because the attention mechanism(e.g. Seq2Seq-attention [6] and CBAM [7]) is performed in the typically lower feature dimension.

This paper proposes a reformed operator design, named dynamic channel-wise convolution, to more increase the attentive representation power with negligible extra GMacs. Dynamic channel-wise convolution uses a set of K parallel convolution kernels in the same way as dynamic convolution, but apply different attention score to each output channel. These convolution kernels are aggregated dynamically $\tilde{\boldsymbol{W}}_c = \sum_k \pi_{c,k}(\boldsymbol{x})\tilde{\boldsymbol{W}}_{c,k}$ per output channel where c means element of output channel. It only introduces extra computational cost to compute attentions $\pi_{c,k}$ and aggregate kernels, which is negligible compared to convolution and dynamic convolution. We demonstrate the effectiveness of dynamic channel-wise convolution on both image classification CIFAR10 and CIFAR100. Without bells and whistles, simply replacing dynamic convolution with dynamic channel-wise convolution in MobileNetV2 achieves solid improvement with only a slight increase (0.01%) of computational cost. We also perform custom forward cuda optimization for dynamic channel-wise convolution. Our forward optimization shows faster speed at higher resolution compared to implemented with pytorch.

II. YOUR APPROACH

A. DYNAMIC CHANNEL-WISE CONVOLUTION

Dynamic Convolution: Let us denote the traditional or static convolution as $\boldsymbol{y} = g(\boldsymbol{W} \otimes \boldsymbol{x} + \boldsymbol{b})$, where w and b are weight matrix and bias vector, g is an activation function (e.g. ReLU [8]), and $\otimes$ is convolution operator. We define the exist dynamic convolution(Figure 1.a) by aggregating multiple K convolution functions $\{\tilde{\boldsymbol{W}}_k \otimes \boldsymbol{x} + \tilde{\boldsymbol{b}}_k\}$ as follows:

$$\begin{aligned} &\boldsymbol{y} = g(\tilde{\boldsymbol{W}}(\boldsymbol{x}) \otimes \boldsymbol{x} + \tilde{\boldsymbol{b}}(\boldsymbol{x})) \\ &\tilde{\boldsymbol{W}}(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{x})\tilde{\boldsymbol{W}}_k, \tilde{\boldsymbol{b}}(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k(\boldsymbol{x})\tilde{\boldsymbol{b}}_k \\ &\text{s.t.} \qquad 0 <= \pi_k(\boldsymbol{x}) <= 1, \sum_{k=1}^{K} \pi_k(\boldsymbol{x}) = 1, \end{aligned} \tag{1}$$

where $\pi_k(\boldsymbol{x})$ is the attention weight for the k^{th} convolution function $\tilde{\boldsymbol{W}}_k \otimes \boldsymbol{x} + \tilde{\boldsymbol{b}}_k$. Note that the aggregated weight $\tilde{\boldsymbol{W}}(\boldsymbol{x})$ and bias $\tilde{\boldsymbol{b}}(\boldsymbol{x})$ are functions of input and share the same attention. Dynamic convolution has more representation power than its static counterpart but it has a lack of attentive representation capacity because it is difficult to control the massive kernels with only K attention weights.

Dynamic Channel-wise Convolution: Equally to dynamic convolution, Dynamic channel-wise convolution(Figure 1.b) has K convolution kernels that share the same kernel size and input/output dimensions. However, they apply attention differently than dynamic convolution. To develop effective attention, we decompose the huge kernels by each output channel and give them different attention weights. We define this process as channel-wise attention. The dynamic channel-wise convolution utilizing channel-wise attention can be formulated as:

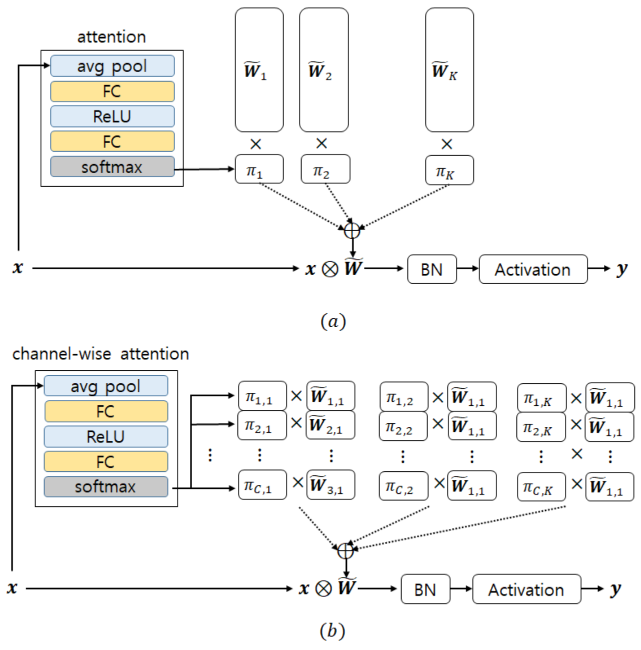

FIGURE 1. (a) A dynamic convolution layer, (b) A dynamic channel-wise convolution layer.

$$\begin{aligned} &\boldsymbol{y} = g(\hat{\boldsymbol{W}}(\boldsymbol{x}) \otimes \boldsymbol{x} + \hat{\boldsymbol{b}}(\boldsymbol{x})) \\ &\tilde{\boldsymbol{W}}_c(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_{c,k}(\boldsymbol{x})\tilde{\boldsymbol{W}}_{c,k}, \tilde{b}_c(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_{c,k}(\boldsymbol{x})\tilde{b}_{c,k} \\ &\tilde{\boldsymbol{W}} = [\tilde{\boldsymbol{W}}_1, \tilde{\boldsymbol{W}}_2, ..., \tilde{\boldsymbol{W}}_C], \tilde{\boldsymbol{b}} = [\tilde{b}_1, \tilde{b}_2, ..., \tilde{b}_C] \\ &\text{s.t.} \qquad 0 <= \pi_{c,k}(\boldsymbol{x}) <= 1, \sum_{k=1}^{K} \pi_{c,k}(\boldsymbol{x}) = 1, \end{aligned} \tag{2}$$

where $\pi_{c,k}(\boldsymbol{x})$ is the channel-wise attention weight for the $\{c^{th}, k^{th}\}$ partially convolution function $\tilde{\boldsymbol{W}}_{c,k} \otimes \boldsymbol{x} + \tilde{b}_{c,k}$ and C is output channel size.

Channel-wise attention: Equally to dynamic convolution, We apply squeeze-and-excitation [4] to compute kernel attentions (see Figure 3). The global spatial information is firstly squeezed by global average pooling. Then we use two fully connected layers (with a ReLU between them) and softmax. The first fully connected layer reduces the dimension by 4. Instead of using only K attention weights for convolution kernels, Channel-wise attention uses $C * K$ attention weights where * is times.

B. FORWARD OPTIMIZATION

Because our goal is to provide better trade-off between network performance and computation burden, we test our proposed module to MobileNetv2 [2], which is a very efficient and lightweight network. In MobileNetv2, depthwise convolution is the core part used in the inner inverted bottleneck block. We implement dynamic channel-wise convolution (only depthwise is available) as cuda to replace the depthwise convolution.

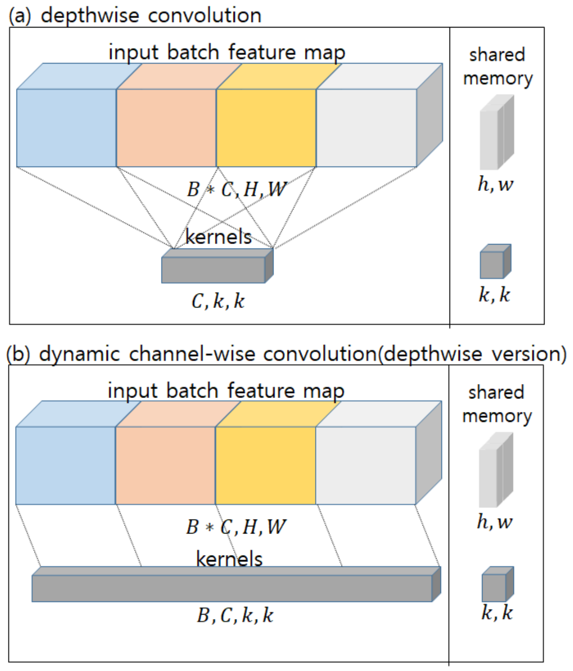

FIGURE 2. (a) depthwise convolution cuda optimization, (b) A dynamic channel-wise convolution cuda optimization (only depthwise is available).

Depthwise convolution cuda optimization: As shown in Figure 2.a, the input batch feature map size is B, C, H, W (B : batch size, C : channel size, H : Height, W : Width) and depthwise convolution kernel size is C, k, k. We first convert B, C, H, W to $B * C, H, W$ and then copy the tile $\{h, w\}$ and kernel $\{k, k\}$ of a specific channel of input batch feature map and depthwise convolution kernel to shared memory. Then, shared tile and shared kernel is calculated convolution. We set the shared memory tile h as 64, w as 16, grid as dim3$((H-1)/h+1, (W-1)/w+1, B*C)$, and block as dim3$(32*8, 1, 1)$

Dynamic channel-wise convolution cuda optimization: As shown in Figure 2.b, since it is aggregating with channel-wise attention, the kernels have the kernel value dependent on the input. Our proposed convolution kernel size is B, C, k, k. Because it is only needed to convert the manner of uploading the kernel value to shared memory, we transform the shared kernel index $C^{th}k^2 + ky + x$ into $(BC)^{th}k^2 + ky + x$ where x and y is kernel width and height index.

III. RESULTS

Implementation Details: We evaluate dynamic channel-wise convolution on MobileNetV2 [2], by replacing the depthwise convolution with our convolution. Each layer has $K = 4$ convolution kernels, which is best hyper-parameter in dynamic convolution. The batch size is 256. The initial learning rate is 0.1 and drops by 10 at epoch 30, 60, and 90. The weight decay is 5e-4. model are trained using SGD optimizer with 0.9 momentum for 100 epochs.

CIFAR10	
model	accuracy
MobileNetV2	0.9293
DY-MobileNetV2	0.9280
DYC-MobileNetV2	**0.9308**
CIFAR100	
model	accuracy
MobileNetV2	0.7236
DY-MobileNetV2	0.7263
DYC-MobileNetV2	**0.7366**

TABLE 1. Models performance comparison in CIFAR10 and CIFAR100.

model	Param	GMacs
MobileNetV2	2.3M	0.09
DY-MobileNetV2	3.58M	0.09
DYC-MobileNetV2	8.01M	0.10

TABLE 2. The number of model parameters and GMacs measurement.

A. ACCURACY COMPARISON

Table 1 shows the classification accuracy on the CIFAR10 and CIFAR100 datasets. DY-MobileNetV2 means dynamic convolution is used, and DYC-MobileNetV2 means dynamic channel-wise convolution is used. In CIFAR10, Compared with MobileNetV2, Accuracy for DYC-MobileNetV2 improves 0.015 but DY-MobileNetV2 drops 0.013. In CIFAR100, both DYC-MobileNetV2 and DY-MobileNetV2 enhance accuracy, but DYC-MobileNetV2 outperforms DY-MobileNetV2 by 0.0103. This suggests that channel-wise attention is more effective to aggregate kernels than attention.

B. PERFORMANCE AND EFFICIENCY COMPARISON

DY-MobileNetV2 vs DYC-MobileNetV2 Table 2 shows the model parameters size and GMacs. When comparing the size of parameters, DYC-MobileNetV2 has about twice as much memory as DY-MoileNetV2 but it is negligible as the network is very lightweight. In terms of accuracy (Table 1), DYC-MobileNetV2 outperforms DY-MobileNetV2 with small extra computation cost (0.01).

Forward speed: We measure the actual forward speed models as shown in Table 3. We set the batch size is 1. All forward speeds are measured in milliseconds. We generate random input samples with 32×32 resolution (32R), 256×256 resolution (256R), and 512×512 resolution (512R), then iterate 10,000 times to compute the forward average speed. When comparing MobileNetV2 and DY-MobileNetV2, they have equal GMacs (Table 2), but MobileNetV2 is 2.442 (ms) faster. However, DYC-MobileNetV2 has extra GMacs cost 0.01 than DY-MobileNetV2, but the actual speed is almost the same as it. This shows that DYC-MobileNetV2 is very competitive in terms of actual speed compared to DY-MobileNetV2.

We test the utility of our dynamic channel-wise forward cuda optimization in the models. In Table 3, our forward optimization is generally similar in speed to the implemented with pytorch, but our strategy is 0.107 (ms) faster at high

model	32R	256R	512R
MobileNetV2	7.609	-	-
DY-MobileNetV2	10.051	-	-
DYC-MobileNetV2	**10.464**	**10.549**	49.018
DYC-MobileNetV2*	10.554	10.635	**48.911**

TABLE 3. Forward speed experiments about models with various resolutions (* : forward optimization).

convolution	32R	256R	512R
depthwise convolution	0.028	-	-
dynamic convolution	0.299	-	-
dynamic channel-wise convolution	**0.317**	**0.533**	2.084
dynamic channel-wise convolution*	0.319	0.572	**1.956**

TABLE 4. Forward speed experiments about convolutions with various resolutions (* : forward optimization).

resolution (512R).

Table 4 shows the comparison of forwarding speed with different convolution operators in various resolutions. Depthwise convolution is absolutely faster than others because it doesn't calculate the squeeze and excitation [4] and attention. dynamic channel-wise convolution has slightly slower than dynamic convolution, but considering the accuracy improvement (Table 1), it is satisfactory. Also, forward optimization speeds up at high resolution.

IV. CONCLUSION

In this paper, we introduce dynamic channel-wise convolution, which converts attention into channel-wise attention in the process of aggregating multiple convolution kernels. Compared to dynamic convolution, it significantly improves the attentive representation power with negligible extra computation cost, thus is more friendly to efficient CNNs. We achieve solid improvement for both CIFAR10 and CIFAR100 image classification. We also implement the forward optimization of dynamic channel-wise convolution available in depthwise convolution. it achieves optimized performance at high resolution. In future work, we will test our module imagenet [9] classification and COCO [10] object detection. We expect enhanced performance like to as the results of our experiments using our dynamic channel-wise convolution.

REFERENCES

[1] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic convolution: Attention over convolution kernels," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[2] 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 (CVPR), June 2018.

[3] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[4] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 8, pp. 2011–2023, 2020.

[5] X. Ding, Y. Guo, G. Ding, and J. Han, "Acnet: Strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks," in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. IEEE, 2019, pp. 1911–1920.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[7] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[8] A. F. Agarap, "Deep learning using rectified linear units (relu)," CoRR, vol. abs/1803.08375, 2018.

[9] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA. IEEE Computer Society, 2009, pp. 248–255.

[10] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V, vol. 8693. Springer, 2014, pp. 740–755.