Work-in-Progress: Flexible Group-Level Pruning of Deep Neural Networks for Fast Inference on Mobile GPUs

Kwangbae Lee*, Hoseung Kim*, Hayun Lee*, Dongkun Shin*
*Sungkyunkwan University, Department of Electrical and Computer Engineering, South Korea
{kblee93,ghtmd123,lhy920806,dongkun}@skku.edu

ABSTRACT

Network pruning is a promising compression technique to reduce computation and memory access cost of deep neural networks. In this paper, we propose a novel group-level pruning method to accelerate deep neural networks on mobile GPUs, where several adjacent weights are pruned in a group while providing high accuracy. Although several group-level pruning techniques have been proposed, the previous techniques can not achieve the desired accuracy at high sparsity. In this paper, we propose a unaligned approach to improve the accuracy of compressed model.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable performance in a variety of problems. To attain high accuracy, DNN models are becoming larger and deeper. Therefore, recent DNN models require tremendous computing cost and energy consumption. Thus, executing those large DNN models at mobile devices which have limited resources is challenging. To resolve such a problem, network pruning techniques [1, 2] were proposed, which eliminate the weight connections having a small influence on accuracy.

The fine-grained pruning can prevent significant accuracy loss by eliminating a small magnitude of weights, but it generates irregular networks. Therefore, it is hard to get a speed-up unless the pruned network has sufficient sparsity. In contrast, the coarse-grained approach [3, 4] groups weight elements into regions and eliminates such regions by magnitude. Therefore, it generates a structured and regular network, the execution time of the pruned network can be reduced. However, the coarse-grained pruning incurs significant accuracy loss at high sparsity. Moreover, it is very difficult to attain 1%-2% accuracy improvement or maintain the desired accuracy using a coarse-grained approach at high sparsity.

The previous coarse-grained pruning approaches [3, 5] implemented with *aligned* approach, where a given weight matrix is first partitioned into multiple non-overlapping groups. Then, the group with small magnitude is removed until the target sparsity is satisfied. Although this algorithm can find victim groups simply, an important weight can be removed. Therefore, the aligned group-level pruning leads to accuracy loss at high sparsity.

In this paper, we introduce a novel group-level pruning approach, called *unaligned* group-level pruning, which can select victim groups without alignment constraint. The weight groups selected from arbitrary locations can improve accuracy without losing important weights.

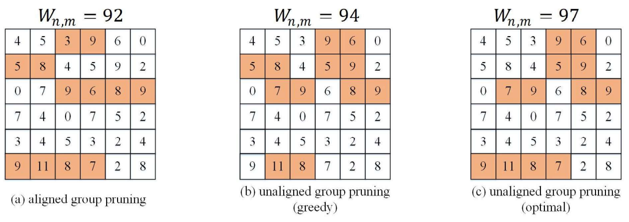

Figure 1: Aligned pruning vs. unaligned pruning.

2 UNALIGNED GROUP-LEVEL PRUNING

2.1 Group Size Determination

The weight group size for pruning must be determined considering underlying hardware factors such as SIMD units, shader cores and registers. For example, if one 128-bit vector register executed four 32-bit floating-point operations simultaneously by each SIMD unit. To make full use of the SIMD unit, the group size must be a multiple of four. Moreover, the number of shader cores is also important factor. It affects the degree of parallelism such as work-groups.

2.2 Unaligned Group Selection

Our group-level pruning determines whether a group is important or not based on the magnitude of the group. For example, as shown in Fig. 1, when the target sparsity is 66.6% and the group size is 2×1, the six groups can be preserved. The aligned group-level pruning yields a regular pruned matrix as shown in Fig. 1(a). However, due to the aligned group-level approach, several large magnitude of weights are eliminated. Therefore, the total value of remaining weights is reduced to 92. If we remove the alignment constraint, we can select weight groups at arbitrary locations and thus can preserve the number of significant elements. Fig. 1(b) shows the result of the unaligned group-level pruning. Under the same sparsity used in Fig. 1(a), the total values of preserved weights is increased.

As a simple solution for the unaligned group-level pruning, we can consider a greedy algorithm, which selects the largest magnitude of group from the candidate groups repeatedly. Although the greedy algorithm is fast and simple, the solution is not optimal when the matrix size and the group size are large. Fig. 1(c) shows the result of optimal solution. The total value of preserved weights in the optimal solution is larger compared with other solution.

To figure out the optimal algorithm, we need to define the unaligned group-level pruning problem. First, we transform the 2D weight matrix (K^2C by F) to 1D weight matrix (K^2CF by 1) to make a searching problem in one-dimensional space. The number of groups to be preserved, m, can be calculated by $m = n \times (1 - S)/G$. When n denotes the width of the transformed matrix (i.e., $n = K^2CF$) and the group size is G, for a given target sparsity ratio S.

CASES'19, October 13–18, 2019, New York, NY, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6925-1/19/10...$15.00
https://doi.org/10.1145/3349569.3351537

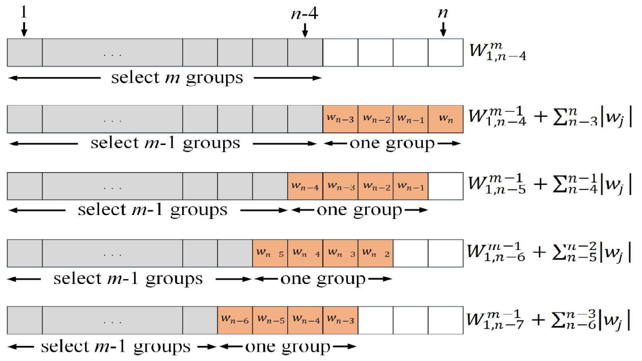
1
n-4
n
$W^{m}_{1,n-4}$
select m groups
$W^{m-1}_{1,n-4} + \Sigma^{n}_{n-3}|w_j|$
select m-1 groups
one group
$W^{m-1}_{1,n-5} + \Sigma^{n-1}_{n-4}|w_j|$
$W^{m-1}_{1,n-6} + \Sigma^{n-2}_{n-5}|w_j|$
$W^{m-1}_{1,n-7} + \Sigma^{n-3}_{n-6}|w_j|$

Figure 2: The five cases one of which can be a solution to maximize $W^m_{1,n}$ assuming the group size is 4.

If our algorithm satisfies the following equation, we can say that it can find the optimal solution.

$$W^k_{s,e} = \max_{G \le i < 2G} \left(W^{k-1}_{s,e-i} + \sum_{j\,=\,e-i+1}^{e-i+G} |w_j|,\ W^k_{s,e-G}\right) \qquad (1)$$

When $W^k_{s,e}$ denotes the total sum of weights in k number of non-overlapping groups selected from the sub-region of the target one-dimensional weight matrix, where the first index and the last index of the sub-region are s and e, respectively. In Equation (1), w_j represents the weight value with the index j. To help the understanding of Equation (1), we provide Fig. 2, which illustrates the five cases one of which can be a solution to maximize $W^m_{1,n}$ assuming the group size is 4. Using dynamic programming, the optimal solution satisfying Equation (1) can be found. When solving the problem, we must not select a group which is located across two rows. Since each row operation uses different output registers, such a boundary-crossing weight group invokes accesses to multiple output registers. Consequently, this situation makes an adverse impact on latency.

3 EVALUATION

Our target hardware platform is ODROID-XU4 board, which is equipped with ARM Mali-T628 GPU. We used ACL (Arm Compute Library 18.11) as a base. Since ACL doesn't support an optimized sparse BLAS library, we implemented our own sparse library to execute the sparse DNN. The pruning didn't modify the first layer and the last layer because those layers are sensitive to pruning. In experiment, the weight group size is set to 4 × 4 for the 2D group to make the best use of SIMD units on Mali GPUs. We compare the inference latency of sparse DNN models with that of the corresponding dense models which ran with the ACL.

We evaluated our technique with VGG-13 for CIFAR-10 dataset. The accuracy of the original dense networks of VGG-13 is 93.57%. Fig. 3(a) shows the accuracy change on the VGG-13 varying the sparsity under different 2D-group pruning methods. The optimal unaligned pruning has higher accuracy than other pruning methods. At a high sparsity, the difference among those three methods is even more pronounced. For example, with the sparsity of 94.9%, the optimal unaligned pruning achieved about 2.8% and 1.6% of improvements on accuracy over the aligned pruning and the greedy unaligned pruning, respectively. From this result, we can say that the proposed optimal pruning algorithm is superior to the aligned

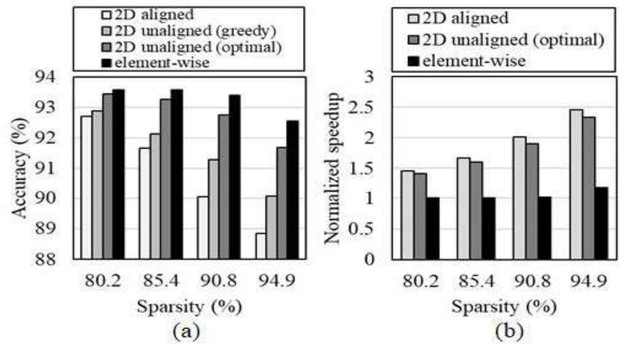

Figure 3: Experimental result of VGG-13 on CIFAR-10.

algorithm. However, our unaligned pruning may increase the inference latency since it generates more cache miss. But, the difference is small (3%-5%). Fig. 3(b) shows the inference latency of sparse networks normalized by the latency of the original dense network by ACL. Since the cache miss ratio of sparse networks generated by different unaligned pruning methods are similar, we show only the result of the optimal unaligned pruning methods. For example, at 90.8% of sparsity, the sparse network generated by the unaligned pruning shows almost 2 times faster performance than the dense network. However, the result of element-wise pruned networks are slower than dense networks with ACL at low sparsity.

4 CONCLUSION

The network pruning is an effective technique to reduce the model size and computation cost. Considering the trade-off between accuracy and performance, we proposed a novel pruning technique, called unaligned group-level pruning. Unlike the previous group-level pruning, our unaligned scheme can select weight groups to be preserved without alignment constraint. Due to its flexibility, the unaligned group sparse networks can provide higher accuracy. Based on experimental results, DNN models can be compressed by unaligned group-level pruning without significant accuracy loss such that they can be deployed at resource-limited mobile systems.

ACKNOWLEDGMENTS

This work was supported by Institute for Information & Communications Technology Planning and Evaluation(IITP) grant funded by the Korea government(MSIT) (No.IITP-2017-0-00914, Software Framework for Intelligent IoT Devices)

REFERENCES

[1] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*. 1135–1143.
[2] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*. 2736–2744.
[3] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally. 2017. Exploring the regularity of sparse structure in convolutional neural networks. *arXiv preprint arXiv:1705.08922* (2017).
[4] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*. 2074–2082.
[5] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. 2017. Scalpel: Customizing dnn pruning to the underlying hardware parallelism. In *ACM SIGARCH Computer Architecture News*, Vol. 45. ACM, 548–560.