Regularization-Free Structural Pruning for GPU Inference Acceleration

Chuliang Guo\textsuperscript{1}, Yanbing Yang\textsuperscript{2}, Li Zhang\textsuperscript{1}, Shaodi Wang\textsuperscript{3}, He Li\textsuperscript{4}, Keyu Long\textsuperscript{2}, Xunzhao Yin\textsuperscript{1*}, Cheng Zhuo\textsuperscript{1*}

\textsuperscript{1}Zhejiang University, Hangzhou, China
\textsuperscript{2}The Second Research Institute of Civil Aviation Administration of China, Chengdu, China
\textsuperscript{3}WITIN Tech Co. Ltd., Beijing, China
\textsuperscript{4}University of Cambridge, Cambridge, UK

Email: \{xzyin1, czhuo\}@zju.edu.cn

Abstract—Pruning is recently prevalent in deep neural network compression to save memory footprint and accelerate network inference. Unstructured pruning, i.e., fine-grained pruning, helps preserve model accuracy, while structural pruning, i.e., coarse-grained pruning, is preferred for general-purpose platforms such as GPUs. This paper proposes a regularization-free structural pruning scheme to take advantage of both unstructured and structural pruning by heuristically mixing vector-wise fine-grained and block-wise coarse-grained pruning masks with an AND operation. Experimental results demonstrate that the proposal can achieve higher model accuracy and higher sparsity ratio of VGG-16 on CIFAR-10 and CIFAR-100 compared with commonly applied block and balanced sparsity.

Index Terms—Structured Sparsity, Network Pruning, Machine Learning, Lightweight Architecture

I. INTRODUCTION

Machine learning, in particular, deep neural networks (DNNs), has achieved success in various artificial intelligence applications [1], such as computer vision and speech recognition. However, with the growing size of DNN, its feature abstraction and learning has high memory consumption and computational complexity, which hinders the development of deep neural network model, especially in the power constrained IoT applications [2]. Therefore, DNN compression techniques have been intensively studied in both academia and industry, pursuing a satisfying trade-off between application model accuracy and hardware execution efficiency. Prevalent approaches include low-rank decomposition [3]-[5], tensorized decomposition [6]-[8], quantization [9]-[12], and pruning [13]-[16].

Pruning aims to alleviate storage demands by eliminating redundant connections in DNNs, wherein sparsification data objects can be errors and gradients in the training process, or weights and activations in the inference process. Han et al. [17] first proposed an element-wise pruning scheme, i.e., fine-grained or unstructured, where non-zero parameters are penalized with $\ell_1$ or $\ell_2$ norm-based regularization during iterative pruning and retraining. Although fine-grained pruning can achieve a significant sparsity ratio, it hardly accelerates model inference due to highly irregular weight distribution, especially in general-purpose processors with insufficient support to sparse matrix computations. Therefore, coarse-grained pruning, i.e., structural pruning, has been proposed to produce more hardware-friendly dense sub-matrix. The idea in [17] of incorporating pruning into training objective functions inspires structural pruning works that formulate pruning as an optimization problem. The optimization then can achieve good GPU execution speed-up with limited accuracy degradation. However, such methods typically involve regularization terms like $\ell_1$ or $\ell_2$ norm [18] with expensive arithmetic operations of high computation complexity. Moreover, the methods typically prefer to preserve the sparsification data objects instead of achieving a relatively high sparsity ratio.

To achieve low computational complexity and high sparsity ratio, the structural pruning can also adopt the heuristic problem formulation, which commonly relies on magnitude-based pruning criteria with a user-specified or predefined target sparsity [19], [20]. Furthermore, neural network semantics, e.g., filters and channels, are usually jointly considered with pruning data objects for more efficient compression and execution. Thus, in this paper, we propose a regularization-free, structural pruning scheme for DNN inference on GPUs by combining vector-wise fine-grained pruning and block-wise coarse-grained pruning. Since the conventional heuristic structural pruning lacks a mechanism to preserve accuracy when compared with the optimization-based methods [17], the balance between structure uniformity and sparsity ratio is rather crucial, which affects:

- \textit{Model accuracy}: The mixed use of multiple pruning schemes in a sequential or combined manner shapes distinct weight distributions and thus different model accuracies.
- \textit{Achieved sparsity}: The achievement of the user-specified target sparsity is demanded under various sparsity granularities.

Given these challenges, the proposed regularization-free structural pruning independently generates fine- and coarse-grained pruning masks and then combines them later with an AND operation. Nevertheless, the combined use of multiple pruning schemes causes insufficient sparsity ratio as the two pruning masks can get overlapped on some weights. We then propose a sparsity compensation method to ensure desired sparsity ratio.

In addition to the advantage of relatively low computational expense from regularization-free and GPU friendly weight distribution in the structural pruning, the proposed pruning scheme is capable of achieving diverse sparsity ratio in each layer when deployed in a layer-wise pruning manner. Moreover, with tunable vector size, block size, and mixed ratio, the proposal unifies different structured sparsity patterns into one framework. In other words, balanced sparsity, block sparsity, and channel-wise structural sparsity can be considered as the special cases of the proposal.

The contributions of this paper are briefly summarized as below:

- We propose a unified structural sparsity-based pruning scheme without regularization terms, enabling mixed granularity for the desired model accuracy and reduced inference overhead.
- We propose a sparsity compensation method to optimize the achieved sparsity and provide the opportunity for trade-offs between model accuracy and sparsity ratio.

Experimental results show that the proposal achieves higher model accuracy and higher sparsity of VGG-16 on CIFAR-10 and CIFAR-100 dataset when in comparison with the commonly used block sparsity and balanced sparsity.

II. RELATED WORK

Pruning aims to reduce the memory footprint of neural networks by zeroing out unimportant data objects, such as errors and gradients in the training process or weights and activations in the inference process. In this work, we mainly consider the network acceleration on inference and thus focus on weight pruning since similar techniques can be extended to neuron pruning, which usually suffers more accuracy degradation [21]. The operations in neural networks are
commonly abstracted in the modality of multiplications [22], which can be processed as general matrix multiplication (GeMM) by lowering high dimensional tensors to matrices. Therefore, weight pruning is typically conducted on a lowered weight matrix to ablate individual elements, kernels, channels, or even layers. In weight pruning, the sparsity ratio mainly determines the degree of memory saving, while the acceleration on hardware execution is related to the pruned structure of the weight matrix. Distinct weight structures can be generated by applying varied sparsity granularities. There are mainly two categories of problem formulation for weight pruning: heuristics and optimizations [21]. Although optimization pruning methods benefit model accuracy, e.g., structured sparsity learning [23] and ADMM method [24]-[26], such methods usually require regularization terms including $\ell_1$ and $\ell_2$ norm [18]. This requires expensive arithmetic operations involving division and square root, while the guarantee of predefined sparsity is not the priority. On the other hand, for heuristic methods [19], [20], [27], a higher sparsity ratio is relatively simple to reach due to the magnitude-based pruning criterion. Therefore, heuristic structural pruning, where weight importance is considered in association with neural network semantics, offers an alternative to address the dilemma of weight matrix irregularity. Mao et al proposed vector sparsity [15] while Narang et al. [27] and Vooturiet et al. [28] proposed block sparsity of single and multiple hierarchies, respectively. Yao et al. proposed balanced sparsity of uniform-sized partition blocks inside channels [19], which can achieve accelerated inference process and maintained application accuracy.

III. METHODOLOGY

The proposed regularization-free structural pruning combines independent vector-wise fine-grained pruning and block-wise coarse-grained pruning. Each of the two pruning schemes iteratively forms a pruning mask. An AND operation between the two pruning masks then generates the optimized pruning mask. In the following, we will detail the proposed methodology.

A. Vector-wise Fine-grained Pruning

Fine-grained pruning is capable of achieving a high sparsity ratio with low computation complexity, while preserving acceptable model accuracy due to its fewer constraints on the weight distribution. Vector-wise fine-grained pruning conducts magnitude-based pruning in vector-wise inside a row, whereas sorting weight elements costs limited computational and storage resources. Each row in the weight matrix is partitioned into equal-sized vectors in the size of $1 \times K$. In the corner cases where $K$ is indivisible, the vectors in the column are zero-padded and will not be improperly pruned in a magnitude-based pruning criterion. The advantage of such a vector-wise fine-grained pruning can help achieve the same sparsity across vectors and channels, as well as the balanced workloads, as reported by Yao et al. [19], which is suitable to the shared memory supplies of parallel GPU threads.

B. Block-wise Coarse-grained Pruning

Current commodity GPUs for DNN applications possess superiority in quick GeMM with rich support to diverse data types and bit-widths. This brings benefits to the coarse-grained pruning, which typically generates more hardware-friendly dense sub-structures with trade-off in model accuracy. The perfect case for GPU execution is that the matrix dimensions are divisible by the tile size, while the generated GPU tiles are divisible by streaming multi-processor (SM) count. Under block-wise constraints, we mainly focus on selecting $R \times S$ block size suitable for GPU tiles as SM count is generally evenly divisible for a given neural network. Additionally, since the coarse-grained pruning naturally suffers more degradation on model accuracy and the fine-grained pruning relies on an absolute magnitude-based criterion, we utilize the first-order Taylor expansion in Eq. (1) as the coarse-grained pruning criterion, which is proven to be a good representation of weight importance by Molchanov et al. [29]. Since operations of additions are less resource- and time-consuming than multiplications, such a pruning criterion will not heavily stress the computation and storage.

$$\tilde{I}_{S}^{(1)}(W) \triangleq \sum_{x \in S} I_{S}^{(1)}(W) = \sum_{x \in S} (g_{x}w_{x})^2 \quad (1)$$

where $\tilde{I}_{S}^{(1)}$ is the approximated first-order Taylor local sum of a structural set of weight $W$ and $g_{x}$ is the gradient of weight $w_{x}$. Similar to the magnitude-based fine-grained pruning, weight blocks with local sum beneath the sparsity threshold of the current epoch will be pruned. Note that for indivisible matrix dimensions by block size, only the largest divisible sub-matrix will be counted for local sum calculations.

C. Regularization-free Structural Pruning

The proposed regularization-free structural pruning unifies different structural prunings into one framework by combining vector-wise fine-grained pruning and block-wise coarse-grained pruning. These two pruning schemes are independently employed instead of in a sequential order. The two pruning masks are later combined with an AND operation. This is because pruning a greater number of weights in the more critical channels may induce potential accuracy degradation.

To achieve the flexibility of structural pruning on distinct semantic levels, e.g., vector-wise, block-wise or channel-wise, a manually selected hyperparameter, the mixed ratio $p$, is introduced to denote the proportion of sparsity ratio contributed by the vector-wise fine-grained pruning. For example, when the target sparsity is 0.7 and $p$ is 0.8, the sparsity ratio contributed by fine- and coarse-grained pruning is 0.56 and 0.14, respectively. However, the combination of the two pruning schemes in an independent manner inevitably results in an insufficient sparsity ratio since some weight elements can be considered as important entries in both pruning. Thus, as shown in Eq. (2) and (3), we propose a simple yet effective sparsity compensation method to guarantee adequate achieved sparsity ratio under any user-specified target sparsity.

$$s_{f} = s_{f} \times \frac{p}{\max(p, 1-p)} \quad (2)$$

$$s_{c} = s_{c} \times \frac{1-p}{\max(p, 1-p)} \quad (3)$$

where $s_{f}, s_{f}$ and $s_{c}$ are the target sparsity, the compensated target sparsity for fine-grained and coarse-grained sparsity, respectively. The neural networks are trained from scratch with $e_{i}$ initial dense training epochs, and then in an iterative manner of pruning followed by retraining. For the current training epoch $e_{i}$, the sparsity threshold is decided by an incremental exponential function:

$$s_{f,\text{thres}} = s_{f} - s_{f} \times \left(1 - \frac{e_{r} - e_{i}}{e_{\text{total}}} \right)^{r} \quad (4)$$

$$s_{c,\text{thres}} = s_{c} - s_{c} \times \left(1 - \frac{e_{r} - e_{i}}{e_{\text{total}}} \right)^{r} \quad (5)$$

where $s_{f,\text{thres}}$ and $s_{c,\text{thres}}$ are the sparsity thresholds of fine-grained and coarse-grained pruning, respectively; $r$ controls how fast or slow the threshold increases. The generation of the optimized pruning mask of the proposed regularization-free structural pruning is demonstrated in Algorithm 1-3. Fine- and coarse-grained pruning masks are generated based on their own target sparsity and threshold ($r=3$ in Eq.
4 and 5), and then combined with an AND operation to prune the original weight matrix.

**Algorithm 1** Vector-wise Fine-grained Sparsity

**Input:**
Original Weight Matrix, \( W(\#row, \#col) \);
Target Sparsity, \( s_t \);
Mixed Ratio, \( p \);
Vector Size, \( K \);

**Output:**
Vector-wise Fine-grained Pruning Mask, \( fmask \);
for \( W_i \in W.rows \) do
  partition \( W_i \) into \( vector_{i,j}, J = 1 \) to \( \#col/K + 1 \);
end

calculate \( sf \) by Eq.2;
\( sf_{\text{thres}} = 0 \);
while \( sf_{\text{thres}} < sf \) do
  increase \( sf_{\text{thres}} \) by Eq.4;
\( n = sf_{\text{thres}} \times K \);
\( \text{thres}_{i,j} = n^{th} \) largest element in \( vector_{i,j} \);
for \( W_i \in W \) do
  \( fmask_{i,j} = 1 \) if \( W_{i,j} > \text{thres}_{i,j} \), else 0;
end

**Algorithm 2** Block-wise Coarse-grained Sparsity

**Input:**
Original Weight Matrix, \( W(\#row, \#col) \);
Target Sparsity, \( s_t \);
Mixed Ratio, \( p \);
Block Size, \( R, S \);

**Output:**
Block-wise Coarse-grained Pruning Mask, \( cmask \);
for \( W_{i,j} \in W \) do
  partition \( W_{i,j} \) into \( block_{i,j}, I = 1 \) to \( \#row/R, J = 1 \) to \( \#col/S \);
end

calculate \( sc \) by Eq.3;
\( sc_{\text{thres}} = 0 \);
while \( sc_{\text{thres}} < sc \) do
  increase \( sc_{\text{thres}} \) by Eq.5;
\( n = sc_{\text{thres}} \times R \times S \);
calculate \( psum_{i,j} \) of each block by Eq.1;
\( \text{thres}_{i,j} = n^{th} \) largest \( psum_{i,j} \);
for \( W_{i,j} \in W \) do
  \( cmask_{i,j} = 1 \) if \( psum_{i,j} > \text{thres}_{i,j} \), else 0;
end

**Algorithm 3** Joint Sparsity

**Input:**
Original Weight Matrix, \( W \);
Vector-wise Fine-grained Pruning Mask, \( fmask \);
Block-wise Coarse-grained Pruning Mask, \( cmask \);

**Output:**
Pruned Weight Matrix, \( W_p \);
\( mask = fmask \ AND cmask \);
\( W_p = W \times mask \);

A. Achieved Sparsity

Since the independently generated vector-wise fine-grained and block-wise coarse-grained pruning masks can overlap some important weight elements, the achieved sparsity ratio may be beneath the target sparsity. Thus we propose the sparsity compensation method to guarantee sufficient sparsity under various sparsity granularity. Take compensated achieved sparsity in Conv layers of VGG-16 for example (target sparsity \( s_t = 0.5 \)), when \( p = 0.5 \) as shown in Fig. 1 (a), the sparsity compensation method detailed in section III provides sufficient and even surplus sparsity, while achieved sparsity in each Conv layer is diverse under the same target sparsity. As for \( p = 0.8 \) in 1 (b), the proposal behaves more similar to vector-wise fine-grained pruning wherein achieved sparsity in each layer is of less difference. Note that the achieved sparsity in early shallow Conv layers varies in a wider range since the vector/block size is fixed while the size of the weight matrix is much smaller than deeper layers. Moreover, when \( p \) is located around 0.5, the surplus sparsity can be utilized by applying longer initial dense training epochs to recover model accuracy [20].

IV. EXPERIMENTS

Experimental results are obtained from image classification tasks on CIFAR-10 and CIFAR-100 dataset [30] in PyTorch [31] implementation. Three prevalent convolutional neural networks including VGG-16 [32], Resnet-164 [33], and Densenet-40 [34] are examined with considerations of model accuracy and sparsity. We set the entire training epoch as 160, including 20 initial dense training epochs followed by iterative pruning and retraining every 20 epochs. The learning rate is 0.1, while the batch size is 64 and 256 for the training and testing sets, respectively. The stochastic gradient descent method with Nesterov momentum [35] of 0.9 is adopted as the optimization method, and the weight initialization is the same as the proposal in [36].

B. Model Accuracy

The proposed regularization-free structural pruning features with a mixed ratio \( p \) for a tunable sparsity granularity since the sparsity granularity is usually reckoned as a crucial factor besides target sparsity. The target sparsity is empirically set at 0.5 and 0.7. We first demonstrate the impact of the mixed ratio on the model accuracy, from 0 to 1 with a step-wise of 0.1. Block sparsity and balanced sparsity are also included in comparison where \( p \) is 0 and 1,

Fig. 1: Achieved sparsity in Conv layers of VGG-16 with various vector size, block size, and mixed ratio.
TABLE I: Top-1 accuracy (and achieved sparsity) of VGG-16 with various mixed ratios ($R = S = 4$, $K = \#col/2$, $s_t = 0.5, 0.7$).

<table>
<thead>
<tr>
<th>p</th>
<th>Top-1 accuracy on CIFAR-10</th>
<th>Top-1 accuracy on CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>93.65% (49.90%)</td>
<td>72.69% (49.90%)</td>
</tr>
<tr>
<td>0.1</td>
<td>93.64% (50.00%)</td>
<td>72.78% (49.98%)</td>
</tr>
<tr>
<td>0.2</td>
<td>93.30% (50.37%)</td>
<td>72.66% (50.30%)</td>
</tr>
<tr>
<td>0.3</td>
<td>93.47% (51.32%)</td>
<td>72.45% (51.06%)</td>
</tr>
<tr>
<td>0.4</td>
<td>93.37% (53.64%)</td>
<td>72.23% (52.91%)</td>
</tr>
<tr>
<td>0.5</td>
<td>93.41% (59.21%)</td>
<td>72.60% (57.94%)</td>
</tr>
<tr>
<td>0.6</td>
<td>93.45% (52.56%)</td>
<td>72.60% (51.36%)</td>
</tr>
<tr>
<td>0.7</td>
<td>93.53% (50.48%)</td>
<td>72.36% (70.53%)</td>
</tr>
<tr>
<td>0.8</td>
<td>93.67% (49.97%)</td>
<td>72.83% (49.93%)</td>
</tr>
<tr>
<td>0.9</td>
<td><strong>93.82%</strong> (49.88%)</td>
<td><strong>72.84%</strong> (49.88%)</td>
</tr>
<tr>
<td>1.0</td>
<td>93.66% (49.86%)</td>
<td>73.14% (55.86%)</td>
</tr>
</tbody>
</table>

respectively. As shown in Table I, with a fine pruning granularity, the proposal obtains the best VGG-16 top-1 accuracy on both CIFAR-10 and CIFAR-100, with a slightly higher sparsity ratio than block sparsity or balanced sparsity.

V. CONCLUSION

This paper proposes a regularization-free structural pruning strategy for GPU inference. Vector-wise fine-grained pruning and block-wise coarse-grained pruning are combined together to generate the optimized pruning mask using an AND operation. The advantages of the fine- and coarse-grained pruning, and many other pruning strategies (e.g., block sparsity, balanced sparsity, and channel-wise structural sparsity) are unified in our proposal. Experimental results demonstrate that the proposal guarantees sufficient sparsity, while
achieving the highest accuracy in comparison to the balanced and block sparsity strategies.

ACKNOWLEDGMENT

This work was partially supported by Sichuan Science and Technology Program with Grant No. 2019YFSY0033, National Natural Science Foundation of China with Grant No. 62034007 and 61974133, and National Key RD Program of China with Grant No. 2018YFE0126300.

REFERENCES