Joint Sparsity with Mixed Granularity for Efficient GPU Implementation

Chuliang Guo[1], Xingang Yan[1], Yufei Chen[1], He Li[2], Xunzhao Yin[1*], Cheng Zhuo[1*]
[1]Zhejiang University, Hangzhou, China
[2]University of Cambridge, Cambridge, UK
*Corresponding authors (e-mail: xzyin1@zju.edu.cn, czhuo@zju.edu.cn)

***Abstract*—Given the over-parameterization property in recent deep neural networks, sparsification is widely used to compress networks and save memory footprint. Unstructured sparsity, *i.e.*, fine-grained pruning, can help preserve model accuracy, while structured sparsity, *i.e.*, coarse-grained pruning, is preferred for general-purpose hardwares, *e.g.*, GPUs. This paper proposes a novel joint sparsity pattern using mixed granularity to take advantage of both unstructured and structured sparsity. We utilize a heuristic strategy to infer the joint sparsity pattern by mixing vector-wise fine-grained and block-wise coarse-grained pruning masks. Experimental results show that the joint sparsity can achieve higher model accuracy and sparsity ratio while consistently maintaining moderate inference speed for VGG-16 on CIFAR-100 in comparison to the commonly used block sparsity and balanced sparsity strategies.**

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

I. Introduction

Recently, deep learning, especially convolutional neural networks (CNNs), has achieved enormous success with state-of-the-art accuracy in the realms of computer vision, speech recognition, and language processing. Due to the over-parameterization of deep neural networks, large models usually demand high computational and storage resources during training and inference [1]–[4]. With the growing popularity of low power devices [5] and increasing demands for energy efficiency [6], [7], techniques including sparsification attract growing interest in compressing and accelerating neural networks to minimize the computational cost.

The sparsity patterns can be classified into fine-grained and coarse-grained sparsity regarding different data objects [8], both aiming to eliminate unimportant elements or connections. Fine-grained sparsity is generally preferred to preserve higher model accuracy as it elaborately prunes weight elements of less importance. However, it is practically challenging to measure the criticality of weight elements in the runtime. Thus, the fine-grained pruning frequently relies on a magnitude-based criterion and thus induces a random reshape of weight structure, which is poorly supported by the general-purpose accelerators such as GPUs.

On the other hand, structured-sparse training has demonstrated efficiency for GPU accelerations [9]–[11], which is considered as a promising coarse-grained alternative for the preservation of compact sub-structures—incorporating neural network semantics such as kernels, filters, and channels [12]–[14]. In addition, as the primary concern of adequate sparsity ratio, structured sparsity relies on a *user-specified* or *calculated* target sparsity threshold to gradually prune the network [15]–[17]. When conducting such sparsity patterns, a balance between the structure uniformity and sparsity ratio is always crucial. In this paper, we propose joint sparsity with a mixed granularity based method for efficient GPU implementation. The motivation is to associate sparsity patterns of different granularities for balanced and compact sub-structures with GPU execution efficiency and preserved model accuracy. There are a few challenges to reach the goal:

- *Model accuracy:* The sequential or combined use of more than one weight pruning scheme results in different weight structures and hence different model accuracies.
- *Achieved sparsity:* The target sparsity needs to be fully reached regardless of the sparsity granularity.
- *Hardware efficiency:* The GPU-implemented inference speedup should be observed under distinct sparsity granularities.

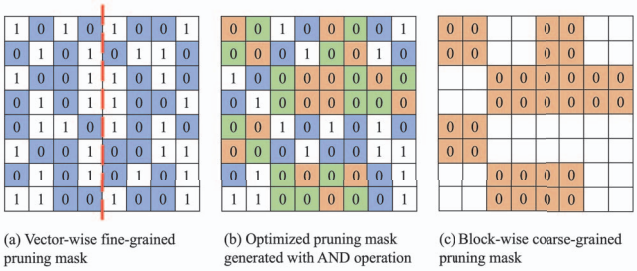


Fig. 1: Generation of pruning masks in joint sparsity.

As shown in Fig. 1, given a target sparsity ratio and the underlying GPU platform, we first independently generate a vector-wise fine-grained pruning mask in Fig. 1 (a), and a block-wise coarse-grained pruning mask in Fig. 1 (c) through incremental sparsity threshold controlled by a hyperparameter denoted as the mixed ratio p in the remainder of the paper. While the fine- and coarse-grained pruning masks function as the bounds, the optimized weight pruning mask in Fig. 1 (b) combines different granularities, thereby achieving balanced accuracy and sparsity.

Moreover, achieved sparsity in each layer is not fixed to a particular ratio when we apply the proposal in a layer-wise pruning manner. By combining pruning masks with tunable vector/block sizes, our proposal actually combines different sparsity patterns into one unified framework, ***i.e.*, the balanced sparsity, block sparsity, and channel-wise structured sparsity are sub-cases of the proposed joint sparsity.**

The contributions of this paper are briefly summarized as below:

- We propose a unified joint sparsity-based spasification approach without regularization terms, enabling mixed granularity for the desired model accuracy and reduced inference overhead.
- A sparsity compensation method is proposed to optimize the achieved sparsity ratio and provide the trade-off between model accuracy and sparsity ratio.
- Even with various vector sizes in fine-grained pruning or block sizes in coarse-grained pruning, joint sparsity can consistently achieve the desired inference speed.

Experimental results show that the proposed joint sparsity can achieve higher model accuracy and sparsity ratio while consistently maintaining the desired inference speed for VGG-16 on CIFAR-100 dataset in comparison to the commonly used block sparsity and balanced sparsity strategies.

II. Related Work

Weight pruning conducted on a lowered weight matrix aims at zeroing out unimportant weight values. There are generally two categories formulating a sparsification problem: heuristic methods

978-3-9819263-5-4/DATE21/©2021 EDAA

and optimization methods.Although formulating the sparsification as an optimization problem benefits model accuracy, *e.g.*, structured sparsity learning [18] and ADMM method [19]–[21], such methods usually demand regularization terms including ℓ_1 and ℓ_2 norm [22], which requires expensive arithmetic operations involving division and square root. Furthermore, they commonly generate a beforehand unknown and relatively low sparsity ratio, while the target sparsity in heuristic methods [15]–[17] is relatively simple to set and reach due to the magnitude-based pruning criterion. Thus, there emerges a visionary opportunity to eliminate the irregularity of weight distribution: weight importance and neural network semantics are jointly considered in weight pruning. Mao *et al* proposed vector sparsity [8], and Narang *et al.* [15] and Vooturiet *et al.* [23] proposed block sparsity pattern with single and multiple hierarchical block types, respectively. Yao *et al.* proposed balanced sparsity, introducing uniform-sized weight blocks across channels [16]. Improved inference time and preserved model accuracy have been reported in these works.

III. Methodology

Joint sparsity consists of independent vector-wise fine-grained and block-wise coarse-grained pruning. Each pruning scheme generates a weight pruning mask in an iterative manner of pruning and retraining. And then the optimized pruning mask is generated by combining these two masks with AND operation.

A. Vector-wise Fine-grained Pruning

Vector-wise fine-grained pruning is crucial to accuracy preservation due to few constraints on weight structure. It is rather efficient to implement sorting inside vectors in a row. Fig. 2 demonstrates the concept of vector-wise fine-grained sparsity within weight matrix rows. Each row in the weight matrix is partitioned into equal-sized 1-by-K vectors, where K can be specified as the maximized volume of shared memory on various GPU platforms. As for the number of weight matrix columns indivisible by K, edge columns are zero-padded for a magnitude-based pruning criterion. In this case, weight values located in edge columns will not be affected by zero-padding. Therefore, the same sparsity ratio can be achieved across vectors as well as channels. The benefit of vector-wise fine-grained sparsity also lies in a balanced workload suitable for shared memory supplies amongst parallel GPU threads, as reported by Yao *et al.* [16].

Dense weight matrix

1.2	1.4	-0.2	0.9	-0.3	1.8	-2.1	0.7	-0.4	1.3
0.5	-0.7	1.5	0.9	1.1	-1.9	1.2	0.2	0.5	-1.6

Vector-wise fine-grained pruned weight matrix

1.2	1.4				1.8	-2.1			
		1.5		1.1	-1.9				-1.6

Fig. 2: Vector-wise fine-grained sparsity within weight matrix rows (K = 5, sparsity = 0.6).

B. Block-wise Coarse-grained Pruning

Coarse-grained pruning typically outperforms fine-grained pruning in shaping compact sub-structures for trades-off between hardware efficiency and model accuracy. Nowadays, commodity GPUs deployed in deep learning application scenarios (*e.g.*, Volta, Turing, and Nvidia A100 GPUs) universally adopt specialized hardware termed Tensor Cores, which hold superiority in quick matrix multiplications of convolution (Conv) and fully-connected (FC) layers. When exploiting such computation parallelism of GPUs within sparsity constraints, the perfect case is that matrix dimensions are divisible by the tile size, and the number of tiles created is divisible by the streaming multiprocessor (SM) count. Given a certain neural network, the SM count is typically evenly divisible, while GPU tiles can be fully occupied by selecting the divisible R-by-S block size. Moreover, since additions are less time- and area-consuming than multiplications, and weight gradients are off-the-shelf in backpropagation, the first-order Taylor approximated local sum in Eq. (1) is utilized as the pruning criterion, which is proven more relevant to the weight importance than the absolute magnitude [24].

$$\widehat{\mathcal{I}}_{\mathcal{S}}^{(1)}(\mathbf{W}) \triangleq \sum_{s\in\mathcal{S}} \mathcal{I}_s^{(1)}(\mathbf{W}) = \sum_{s\in\mathcal{S}} (g_s w_s)^2 \tag{1}$$

where $\widehat{\mathcal{I}}_{\mathcal{S}}^{(1)}$ is the first-order Taylor approximated importance of a structural set of weight $\mathbf{W}$, and g_s is the gradient of weight w_s. The weight block whose local sum is beneath the sparsity threshold of the current epoch will be pruned. When matrix dimensions are indivisible by the block size, only the most extensive sub-weight matrix divisible by the block size is counted, and thus edge blocks will not be pruned.

C. Joint Sparsity with Mixed Granularity

Joint sparsity forms the optimized weight mask via AND operation on fine-grained and coarse-grained pruning masks. These two pruning masks are independently generated instead of sequentially since more weights in some valuable channels might be pruned in sequential pruning, resulting in potential accuracy degradation.

Aiming to achieve the mixed sparsity granularity, we set a human-decided hyperparameter, denoted as the mixed ratio p, to control the proportion that vector-wise fine-grained sparsity contributes to the target sparsity. For example, if the target sparsity is 0.7 (*i.e.*, 70% elements in the pruned weight matrix are zeros.) and p is 0.8, the sparsity contributed by fine- and coarse-grained sparsity should be 0.56 and 0.14, respectively. Nevertheless, the indeed achieved sparsity in each layer is observed lower than the target sparsity since fine- and coarse-grained masks can overlap some weight elements. This could be explained that some important weights are measured in high significance in both pruning criteria. Thus, we re-approximate the sparsity contributed by fine- and coarse-grained pruning:

$$s_f = s_t \times \frac{p}{\max(p, 1-p)} \tag{2}$$

$$s_c = s_t \times \frac{1-p}{\max(p, 1-p)} \tag{3}$$

where s_t, s_f and s_c are target sparsity, re-approximated target sparsity for fine- and coarse-grained sparsity respectively.

As shown in Fig. 3, when deploying the sparsity compensation method, the target sparsity can be fully achieved regardless of its value. Furthermore, with p around 0.5, as shown in Table I, the surplus sparsity obtains a trade-off between sparsity and accuracy from different initial dense training epochs. Alternatively, with p near 0 or 1, the achieved sparsity ratio is closer to the target sparsity. In particular, balanced sparsity can be achieved with p = 1, while block sparsity and channel-wise sparsity are sub-cases of p = 0.

We iteratively prune the weight matrix and retrain the network for several epochs when generating fine- and coarse-grained pruning masks. In practice, iterative pruning [25] typically prunes a more significant number of weights while maintaining model accuracy. An incremental exponential function with positive but decreasing first derivative is applied to calculate the current sparsity threshold:

$$s_{f\_thres} = s_f - s_f \times \left(1 - \frac{e_c - e_i}{e_{total}}\right)^r \tag{4}$$

$$s_{c\_thres} = s_c - s_c \times \left(1 - \frac{e_c - e_i}{e_{total}}\right)^r \tag{5}$$

TABLE I: Achieved sparsity and top-1 accuracy of VGG-16 on CIFAR-100 with various initial dense training epochs (p = 0.5).

R	S	s_t	Sparsity (20 epochs)	Accuracy (20 epochs)	Sparsity (40 epochs)	Accuracy (40 epochs)	Sparsity (60 epochs)	Accuracy (60 epochs)	Sparsity (80 epochs)	Accuracy (80 epochs)
4	4	0.5	57.94%	72.60%	57.53%	72.35%	57.22%	72.38%	53.89%	72.74%
4	4	0.7	76.96%	71.42%	76.50%	71.76%	74.75%	72.08%	72.41%	72.21%
8	8	0.5	59.05%	72.26%	58.66%	71.90%	56.37%	72.69%	54.47%	72.56%
8	8	0.7	78.13%	70.92%	77.78%	70.97%	75.91%	71.33%	71.35%	72.46%

where s_{f_thres} and s_{c_thres} are the sparsity threshold of fine- and coarse-grained pruning of the current epoch e_c, respectively. e_i is the initial pruning epoch, while r controls how fast or slow the threshold increases exponentially.

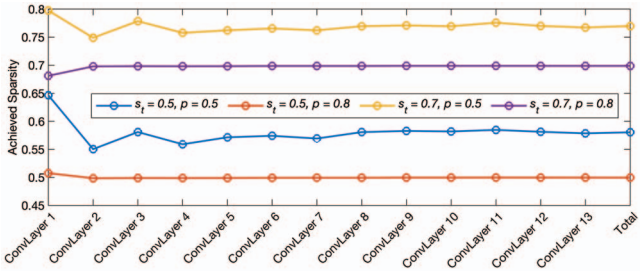


Fig. 3: The truly achieved sparsity of joint sparsity in Conv layers of VGG-16 (K = $\#col$, R = 4, S = 4).

IV. Experiments

We present experimental results of PyTorch [26] implemented image classification tasks on CIFAR-100 dataset [27] with various target sparsity s_t, mixed ratio p, vector size $1 \times K$, and block size $R \times S$. Three prevalent convolutional network architectures (VGG-16 [28], Resnet-164 [29], and Densenet-40 [30]) are trained from scratch, iteratively pruned and fine-tuned every 20 epochs with $r = 3$ in Eq. 4 and 5. The entire training epoch is 160 including 20 initial dense training epochs, with a learning rate of 0.1 and a batch size of 64 and 256 for the training and testing set, respectively. The optimization method is stochastic gradient descent with Nesterov momentum [31] of 0.9 without dampening, and the weight initialization method introduced in [32] is also applied.

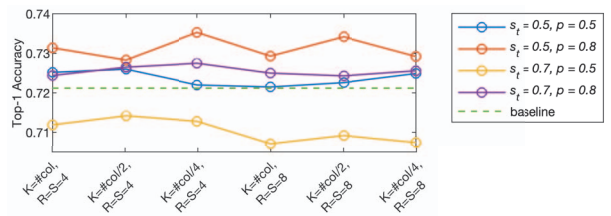
A. Impact on Accuracy

We first examine how top-1 accuracy of VGG-16 varies when p ranges from 0 to 1 with a step length of 0.1. As shown in Table II, the achieved sparsity displays diversity under the same target sparsity but different mixed ratio p. Note that block sparsity and balanced sparsity are also incorporated in Table II as sub-cases when p is 0 and 1, respectively. Fine-grained sparsity is generally considered best to preserve the accuracy, while the highest top-1 accuracy of VGG-16 appears in joint sparsity other than balanced sparsity, where p equals 0.7 under the target sparsity of 0.5, and p equals 0.8 under the target sparsity of 0.7. In the meantime, joint sparsity provides an even slightly higher sparsity ratio of 69.86% than that of 69.83% in balanced sparsity. Moreover, the achieved sparsity is closest to the target sparsity with p around 0.7 and 0.8.

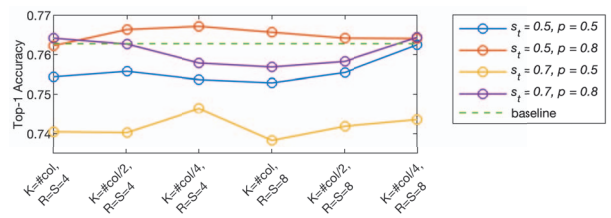
Fig. 4 (a) indicates the collective impact of vector size and block size on top-1 accuracy of VGG-16 on the CIFAR-100 dataset. Pruned models obtain accuracy above 72.12% of the baseline model except when the target sparsity is 0.7, and the mixed ratio is 0.5. Furthermore, pruned models under the sparsity of 0.7 can achieve better accuracy than the sparsity of 0.5 due to a finer sparsity granularity, while severe accuracy degradation also occurs as the achieved sparsity grows too high for error tolerance (up to 78.22% when s_t = 0.7, p = 0.5). The sparsity ratio also acts as a significant

TABLE II: Achieved sparsity and top-1 accuracy of VGG-16 on CIFAR-100 with various mixed ratios (R = S = 4, K = $\#col/2$).

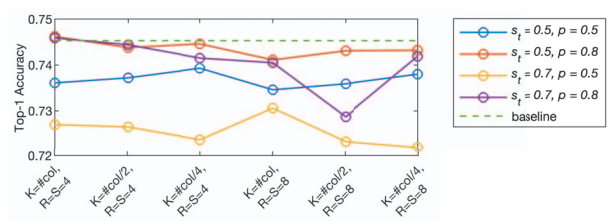
p	Sparsity ($s_t = 0.5$)	Accuracy ($s_t = 0.5$)	Sparsity ($s_t = 0.7$)	Accuracy ($s_t = 0.7$)
0.0	49.90%	72.69%	69.86%	71.51%
0.1	49.98%	72.78%	69.92%	71.85%
0.2	50.30%	72.66%	70.14%	71.68%
0.3	51.06%	72.45%	70.71%	71.76%
0.4	52.91%	72.23%	72.21%	71.44%
0.5	57.94%	72.60%	76.96%	71.42%
0.6	51.36%	72.60%	70.53%	72.61%
0.7	50.15%	**73.46%**	69.93%	72.50%
0.8	49.93%	72.83%	69.86%	**72.65%**
0.9	49.88%	72.84%	69.83%	72.64%
1.0	55.86%	73.14%	69.83%	72.63%
baseline	-	72.12%	-	72.12%



(a) Top-1 accuracy of VGG-16 on CIFAR-100.



(b) Top-1 accuracy of Resnet-164 on CIFAR-100.



(c) Top-1 accuracy of Densenet-40 on CIFAR-100.

Fig. 4: Top-1 accuracy comparisons with various vector sizes, block sizes, target sparsity, and mixed ratios.

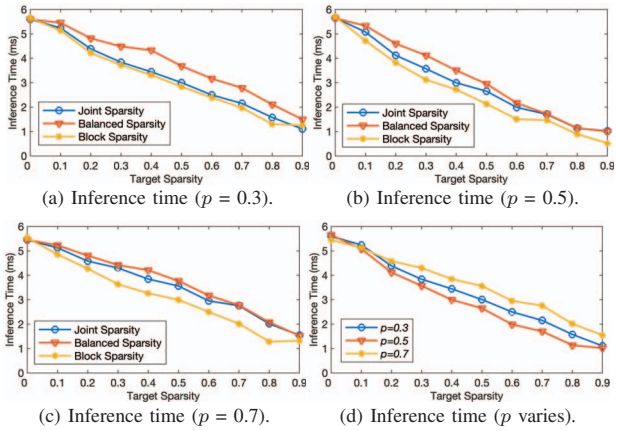
impact on accuracy in Resnet-164 and Densent-40, as shown in Fig. 4 (b) and (c), while the shrinkage of the block size or vector size does not always guarantee a better accuracy performance.

B. Inference Time

To demonstrate the hardware efficiency, we conduct a benchmark of GeMM in the size of 8192×8192 under various mixed ratio p. Both block sparsity and balanced sparsity are included when p is 0 and 1, respectively. Inference times of three sparsity patterns are examined of the same achieved sparsity, vector size, and block size

for a fair comparison. As shown in Fig. 5, GPU tiles are fully utilized since the block size is divisible by 16. Block sparsity, therefore, achieved a slightly faster inference speed than balanced sparsity. As p grows, the inference time of joint sparsity decreases closer to balanced sparsity away from block sparsity. Since block sparsity and balanced sparsity are sub-cases of joint sparsity, the mixed ratio p realizes the trade-off between model accuracy and inference time.

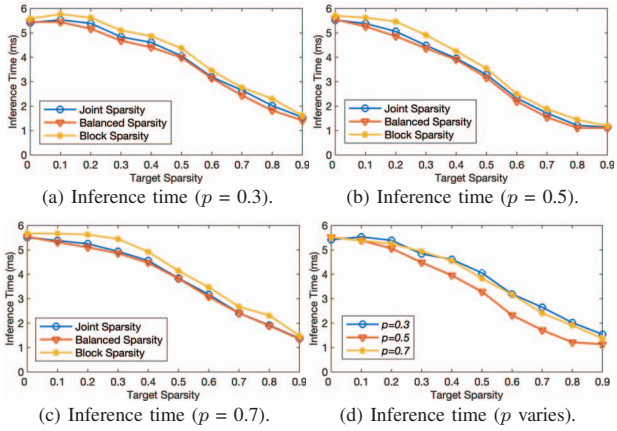
In the scenario of the vector size of 1×512 and the block size of 4×4, as shown in Fig. 6, balanced sparsity achieves shortened inference time due to fewer elements to sort in vectors. Correspondingly, the inference time of joint sparsity grows closer to block sparsity away from balanced sparsity. This simply means that under the same sparsity ratio, joint sparsity consistently acquires a moderate inference time without regard to the vector size of fine-grained pruning and the block size of coarse-grained pruning.



(a) Inference time (p = 0.3). (b) Inference time (p = 0.5).

(c) Inference time (p = 0.7). (d) Inference time (p varies).

Fig. 5: Inference time benchmark comparisons with various target sparsity and mixed ratio (K = 2048, R = S = 16).



(a) Inference time (p = 0.3). (b) Inference time (p = 0.5).

(c) Inference time (p = 0.7). (d) Inference time (p varies).

Fig. 6: Inference time benchmark comparisons with various target sparsity and mixed ratios (K = 512, R = S = 4).

V. Conclusion

In this paper, we propose joint sparsity by combining vector-wise fine-grained and block-wise coarse-grained pruning masks. The advantages of the two sparsity strategies are unified in our proposal. Experimental results demonstrate that the proposed strategy with a sparsity compensation method not only guarantees sufficient sparsity but provides additional sparsity for longer initial dense training epochs to maintain accuracy. Compared with block sparsity and balanced sparsity, the proposed joint sparsity further achieves the highest top-1 accuracy of VGG-16 on CIFAR-100, with a slightly higher sparsity ratio and moderate inference time.

Acknowledgment

This work was partially supported by NSFC with Grant No. 62034007 and 61974133, and Key Area RD Program of Guangdong Province with Grant No. 2018B030338001.

References

[1] EL. Denton et al., "Exploiting linear structure within convolutional networks for efficient evaluation," Proc. NIPS, 2014.
[2] J. Ba et al., "Do deep nets really need to be deep?," Proc. NIPS, 2014.
[3] M. Zhu et al., "To prune, or not to prune: exploring the efficacy of pruning for model compression," arXiv preprint arXiv:1710.01878, 2017.
[4] C. Li et al., "A Scalable Design of Multi-Bit Ferroelectric Content Addressable Memory for Data-Centric Computing," IEEE IEDM, 2020.
[5] X. Yin et al., "FeCAM: A universal compact digital and analog content addressable memory using ferroelectric," IEEE TED, 2020, 67(7), 2785-2792.
[6] C. Zhuo et al.,"Noise-Aware DVFS for Efficient Transitions on Battery-Powered IoT Devices," IEEE TCAD 39(7):1498-1510, 2020.
[7] C. Zhuo et al.,"From Layout to System: Early Stage Power Delivery and Architecture Co-Exploration," IEEE TCAD 38(7):1291-1304, 2019.
[8] H. Mao et al., "Exploring the granularity of sparsity in convolutional neural networks," Proc CVPR, 2017.
[9] S. Gray et al., "Gpu kernels for block-sparse weights," arXiv preprint arXiv:1711.09224, 2017.
[10] C. Chen et al., "Optimally approximated and unbiased floating-point multiplier with runtime configurability," ICCAD, 2020, 1-9.
[11] J. Deng et al.,"Energy Efficient Real-Time UAV Object Detection on Embedded Platforms," IEEE TCAD 39(10):3123-3127, 2020.
[12] Y. Hu et al., "A novel channel pruning method for deep neural network compression." arXiv preprint arXiv:1805.11394, 2018.
[13] H. Li et al., "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710, 2016.
[14] Y. He et al., "Channel pruning for accelerating very deep neural networks." Proc. ICCV, 2017.
[15] S. Narang et al., "Block-sparse recurrent neural networks," arXiv preprint arXiv:1711.02782, 2017.
[16] Z. Yao et al., "Balanced sparsity for efficient dnn inference on gpu," Proc. AAAI, 2019.
[17] N. Gamboa et al., "Campfire: Compressible, regularization-free, structured sparse training for hardware accelerators," arXiv preprint arXiv:2001.03253, 2020.
[18] W. Wen et al., "Learning structured sparsity in deep neural networks," Proc. NIPS, 2016.
[19] T. Zhang et al., "A systematic DNN weight pruning framework using alternating direction method of multipliers," Proc. ECCV, 2018.
[20] T. Zhang et al., "StructADMM: A systematic, high-efficiency framework of structured weight pruning for DNNs," arXiv preprint arXiv:1807.11091, 2018.
[21] S. Ye et al., "Progressive weight pruning of deep neural networks using ADMM," arXiv preprint arXiv:1810.07378, 2018.
[22] M. Yuan et al., "Model selection and estimation in regression with grouped variables," JRSS: Series B, 68(1):49-67, 2006.
[23] DT. Vooturiet al., "Hierarchical block sparse neural networks," arXiv preprint arXiv:1808.03420, 2018.
[24] P. Molchanov et al., "Importance estimation for neural network pruning," Proc. CVPR, 2019.
[25] S. Han et al., "Learning both weights and connections for efficient neural network," Proc. NIPS, 2015.
[26] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," Proc. NIPS, 2019.
[27] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," tech report, 2009.
[28] K. Simonyan et al., "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[29] K. He et al., "Deep residual learning for image recognition," Proc. CVPR, 2016.
[30] G.Huang et al., "Densely connected convolutional networks," Proc. CVPR, 2017.
[31] I. Sutskever et al., "On the importance of initialization and momentum in deep learning," Proc. ICML, 2013.
[32] K. He et al., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," Proc. ICCV, 2015.