HRBP: Hardware-friendly Regrouping towards Block-based Pruning for Sparse CNN Training

CPAL 2024

Haoyu Ma[1*], Chengming Zhang[2*], Lizhi Xiang[3*], Xiaolong Ma[4], Geng Yuan[5], Wenkai Zhang[1], Shiwei Liu[6,7,8], Tianlong Chen[9,10,11], Dingwen Tao[2], Yanzhi Wang[12], Zhangyang Wang[6], Xiaohui Xie[1]

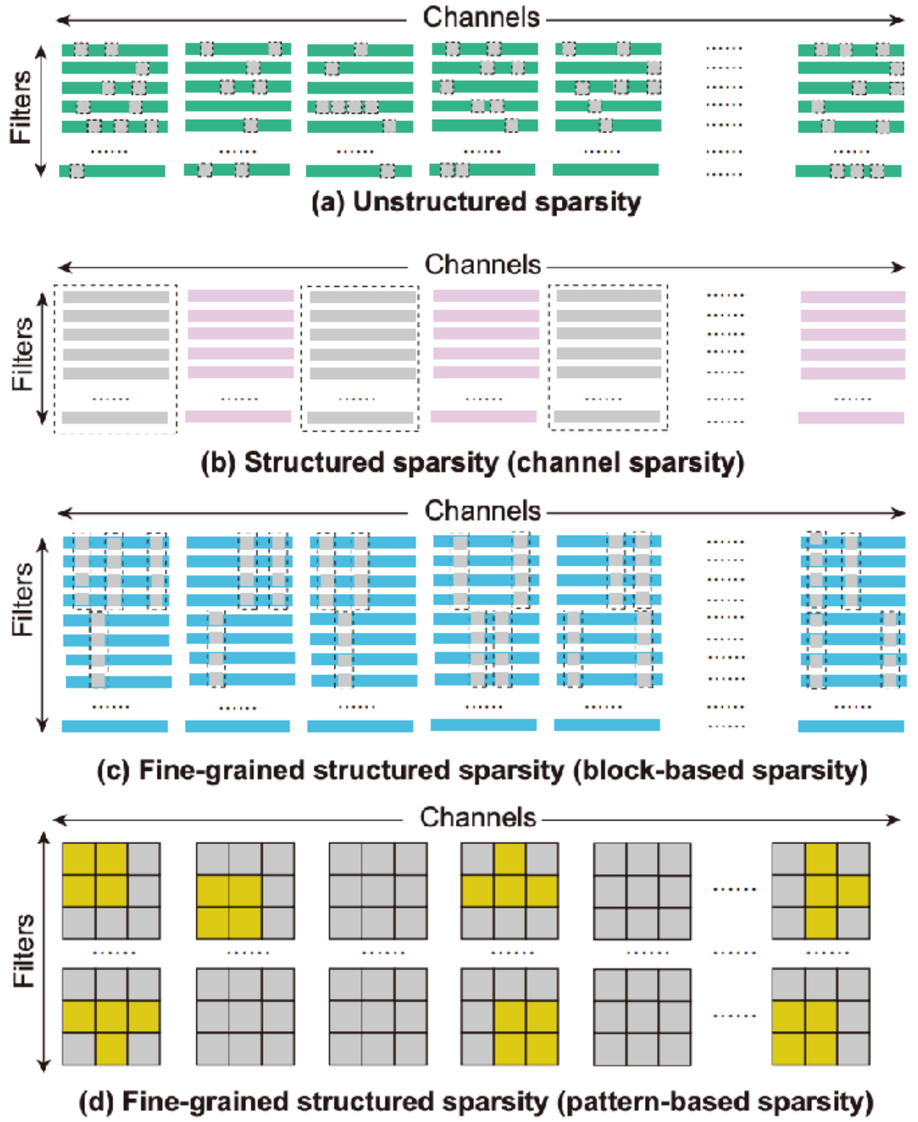
[1]University of California, Irvine, [2]Indiana University Bloomington, [3]University of Utah, [4]Clemson University, [5]University of Georgia, [6]University of Texas at Austin, [7]Eindhoven University of Technology, [8]University of Oxford, [9]MIT, [10]Harvard University, [11]The University of North Carolina at Chapel Hill, [12]Northeastern University

Background: Network Sparsity

- Unstructured Sparsity
  - maintain accuracy at large sparsity
  - almost no acceleration on hardware

- Structured Sparsity
  - suffer accuracy drop at large sparsity
  - real acceleration on hardware

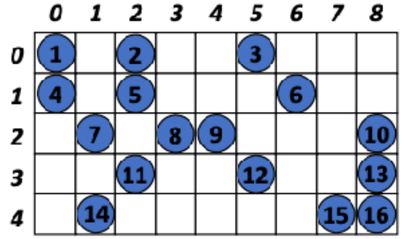
- Fine-grained Structured Sparsity
  - retain the benefits of unstructured and structured sparsity
  - pattern-based
  - **block-based**



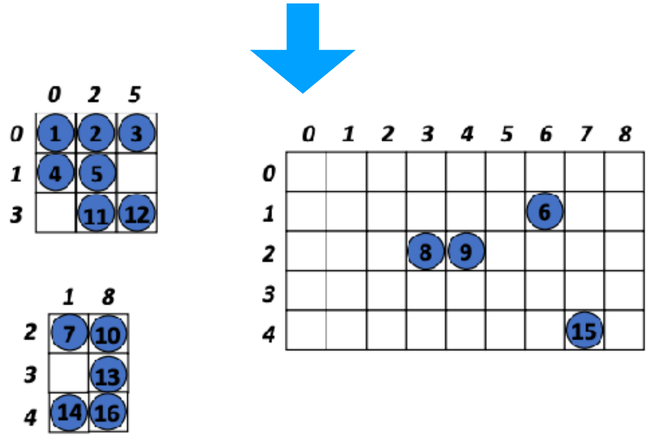
Yuan, Geng, et al. "Mest: Accurate and fast memory-economic sparse training framework on the edge." *Advances in Neural Information Processing Systems* 34 (2021): 20838-20850.

Background: Weights Regrouping on Unstructured Sparsity

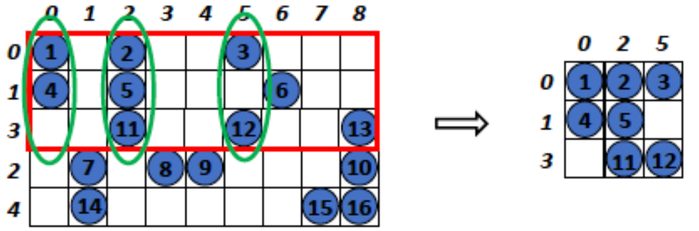
- extract dense blocks from sparse weights with weights regrouping algorithm [1]
- accelerate the inference (forward pass) of unstructured sparsity on hardware



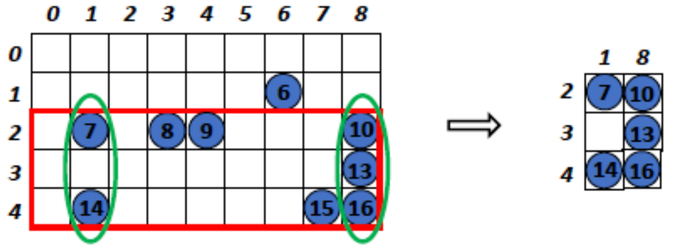
(a) A pruned convolutional layer with five kernels



(b) Weights are reorganized into two dense blocks and a sparse block



(a) Row 0, row 1 and row 3 are grouped together, and columns with at least two non-zeros are selected



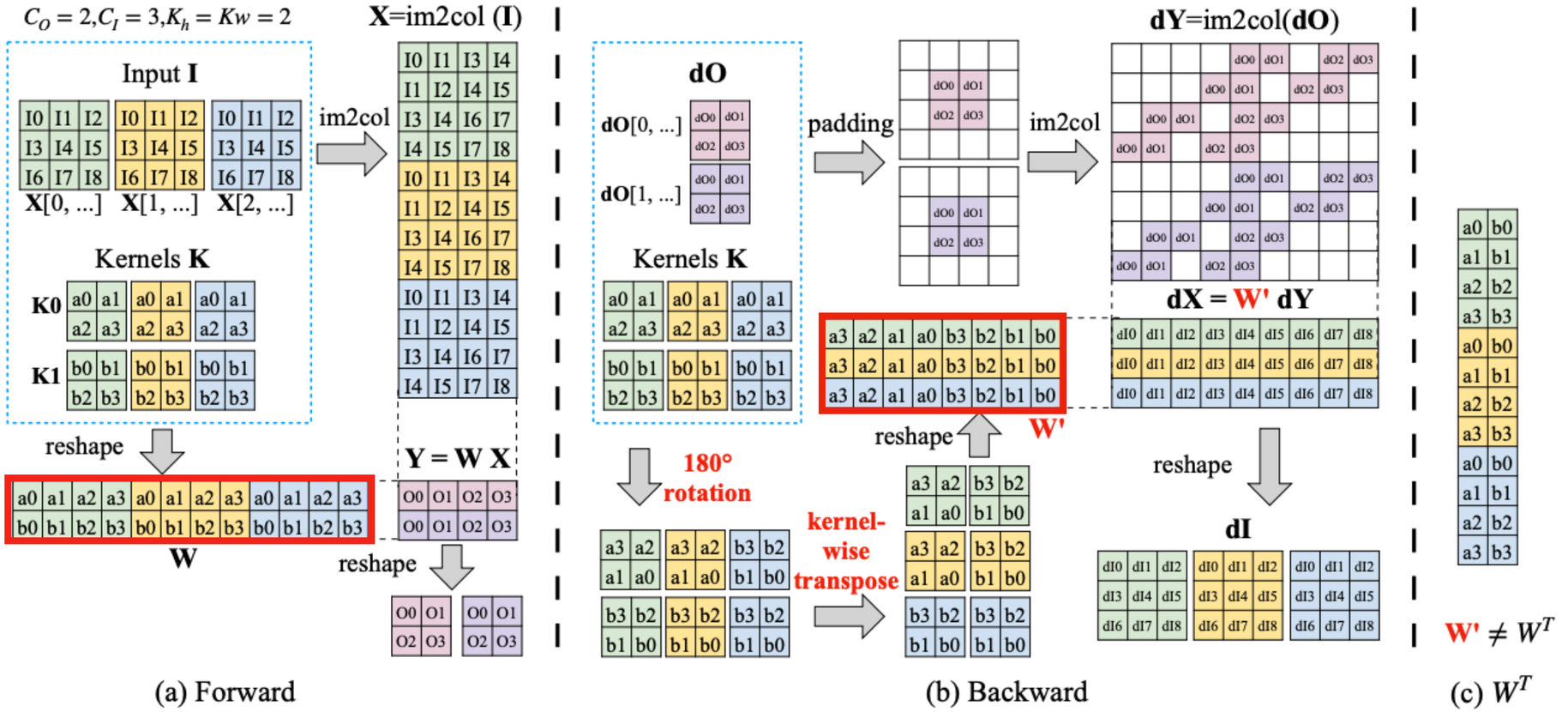
(b) Row 2, row 3 and row 4 are grouped together, and columns with at least two non-zeros are selected

[1] Rumi, Masuma Akter, et al. "Accelerating sparse cnn inference on gpus with performance-aware weight pruning." *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*. 2020.

Background: GEMM Implementation of CNN

- CNN is implemented with general matrix multiplication (GEMM) $\mathbf{Y} = \mathbf{WX}$ on hardware
- the backward pass of CNN cannot be linearly simplified to $\mathbf{dX} = \mathbf{W^T dY}$ since $\mathbf{W}' \neq \mathbf{W^T}$

- Linear layer:
  - forward:
    - $\mathbf{Y} = \mathbf{WX}$
  - backward:
    - $\mathbf{dW} = \mathbf{dY} \cdot \mathbf{X^T}$
    - $\mathbf{dX} = \mathbf{W^T dY}$

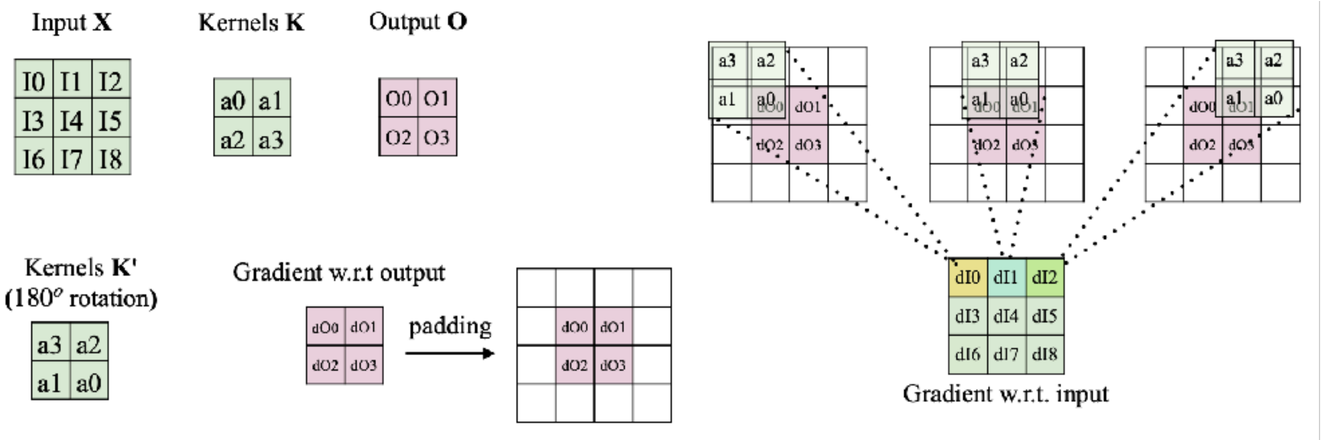


Background: GEMM Implementation of CNN (cont.)

- full convolution between a 180-degree rotated filter (**K**′) and the gradient w.r.t. the output **dO**

$$\mathbf{O}_{i,j} = \sum_{m=1}^{K_h} \sum_{n=1}^{K_w} K_{m,n} X_{i+m,j+n}$$

$$\frac{\partial L}{\partial X_{a,b}} = \sum_{i=1}^{H_O} \sum_{j=1}^{W_O} \frac{\partial L}{\partial O_{i,j}} \frac{\partial O_{i,j}}{\partial X_{a,b}}$$



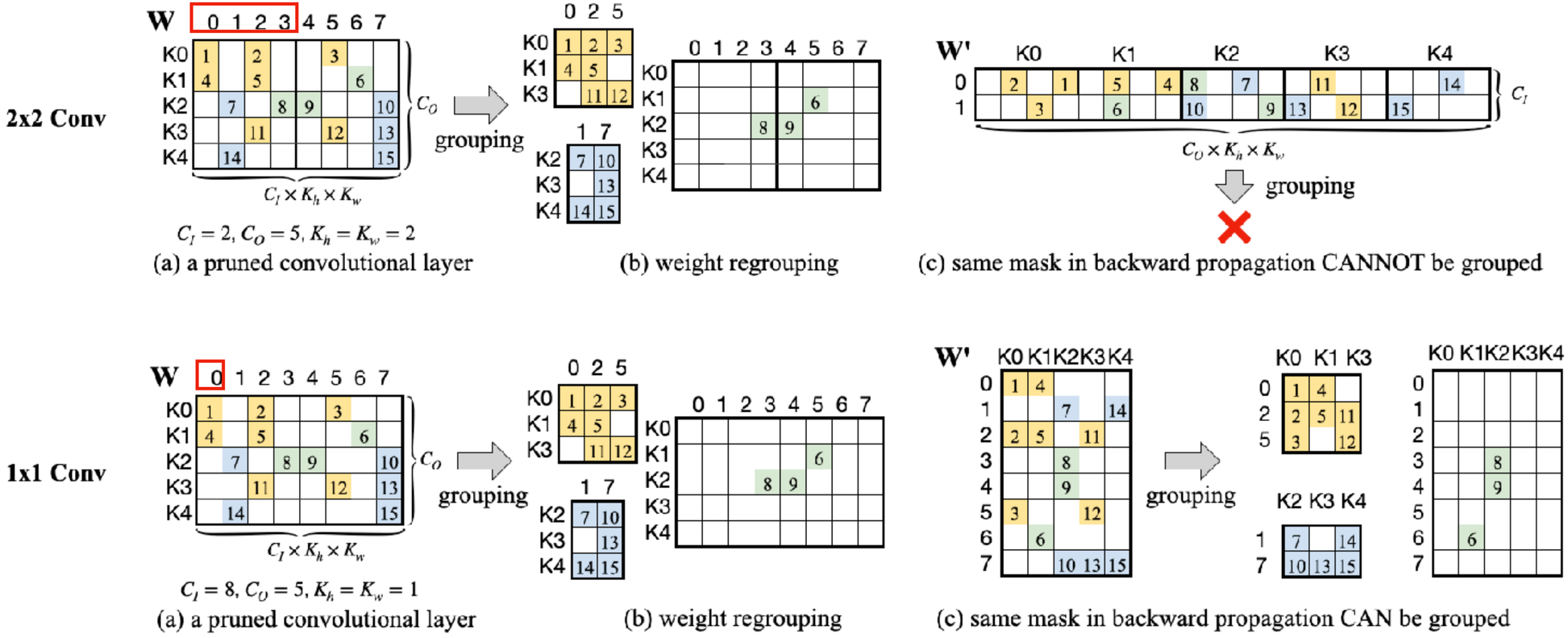
$$O_0 = I_0 * a_0 + I_1 * a_1 + I_3 * a_2 + I_4 * a3$$
$$O_1 = I_1 * a_0 + I_2 * a_1 + I_4 * a_2 + I_5 * a3$$
$$O_2 = I_3 * a_0 + I_4 * a_1 + I_6 * a_2 + I_7 * a3$$
$$O_3 = I_4 * a_0 + I_5 * a_1 + I_7 * a_2 + I_8 * a3$$

$$\frac{\partial L}{\partial I_0} = \frac{\partial L}{\partial O_0} * a_0$$
$$\frac{\partial L}{\partial I_1} = \frac{\partial L}{\partial O_0} * a_1 + \frac{\partial L}{\partial O_1} * a_0$$
$$\frac{\partial L}{\partial I_2} = \frac{\partial L}{\partial O_1} * a_1$$
$$\frac{\partial L}{\partial I_3} = \frac{\partial L}{\partial O_0} * a_2 + \frac{\partial L}{\partial O_2} * a_0$$
$$\frac{\partial L}{\partial I_4} = \frac{\partial L}{\partial O_0} * a_3 + \frac{\partial L}{\partial O_1} * a_2 + \frac{\partial L}{\partial O_2} * a_1 + \frac{\partial L}{\partial O_3} * a_0$$
$$\frac{\partial L}{\partial I_5} = \frac{\partial L}{\partial O_1} * a_3 + \frac{\partial L}{\partial O_3} * a_1$$
$$\frac{\partial L}{\partial I_6} = \frac{\partial L}{\partial O_2} * a_2$$
$$\frac{\partial L}{\partial I_7} = \frac{\partial L}{\partial O_2} * a_3 + \frac{\partial L}{\partial O_3} * a_2$$
$$\frac{\partial L}{\partial I_8} = \frac{\partial L}{\partial O_3} * a_3$$

Reference: https://deeplearning.cs.cmu.edu/F21/document/recitation/Recitation5/CNN_Backprop_Recitation_5_F21.pdf

Problems: Dense Blocks in CNN Backpropagation

- Extracted dense blocks [1] are broken in backward pass due to the transformation from **W** to **W′**
  - 1x1 convolution (reduced to linear layer) is an exception
- Cannot accelerate the backward pass when performing sparse training

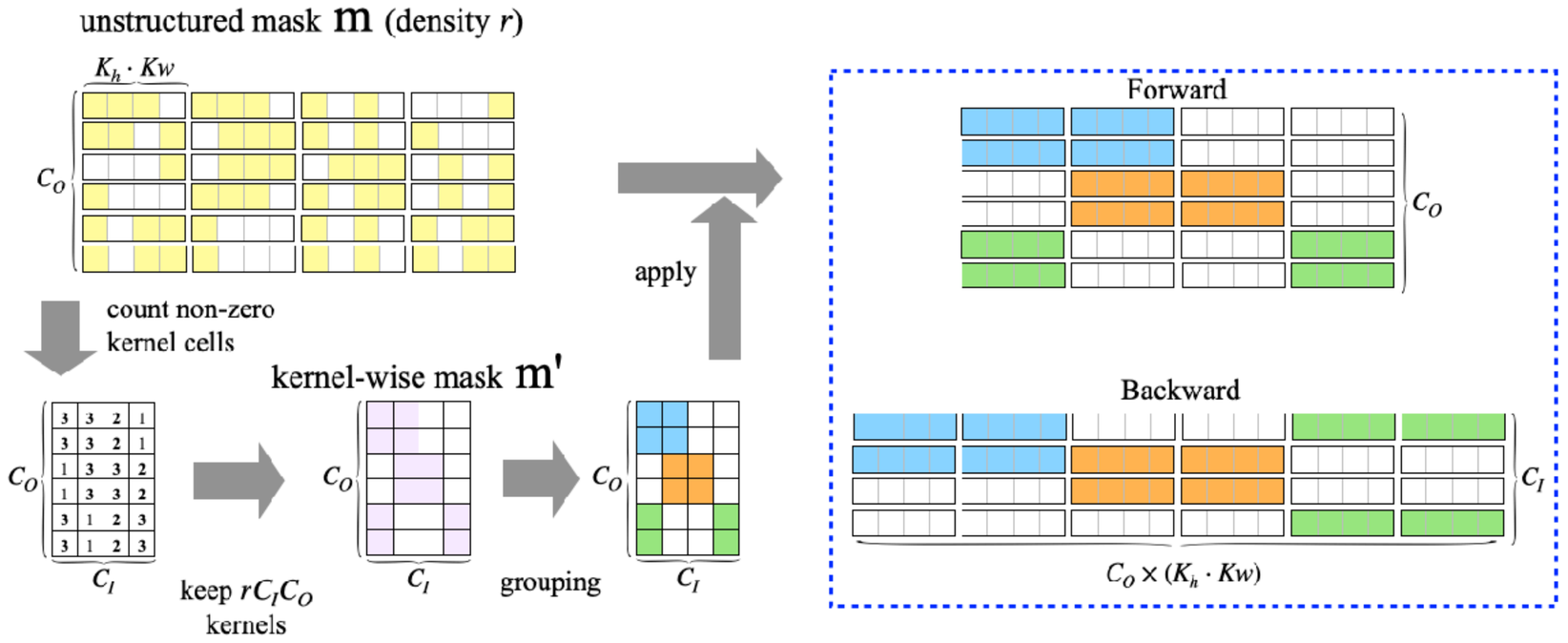


[1] Rumi, Masuma Akter, et al. "Accelerating sparse cnn inference on gpus with performance-aware weight pruning." *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*. 2020.

Method: Hardware-friendly Regrouping for Block-wise Pruning

---- Kernel-wise mask grouping

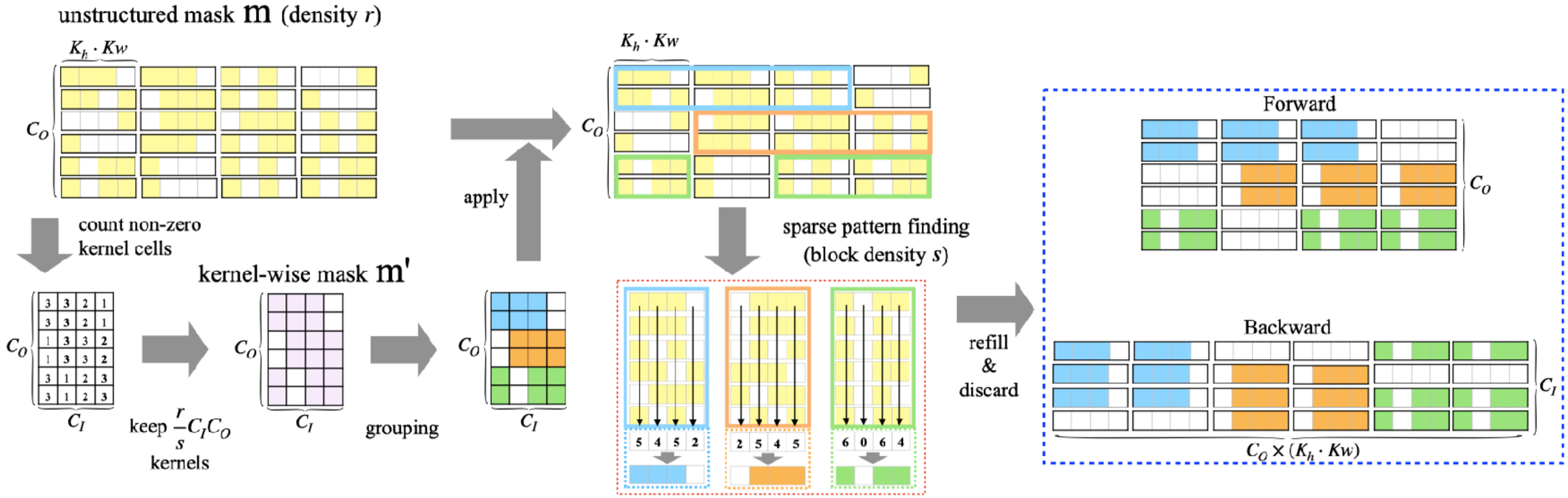
- The fact that fragmentary blocks in CNN can be naturally solved in 1x1 convolution (reduced to linear layer) suggests that we should locate identical elements in different input channels within one group
- kernel-wise mask enables "1x1 convolution" style grouping



Method: Hardware-friendly Regrouping for Block-wise Pruning

---- Exclusive block sparse pattern within each block

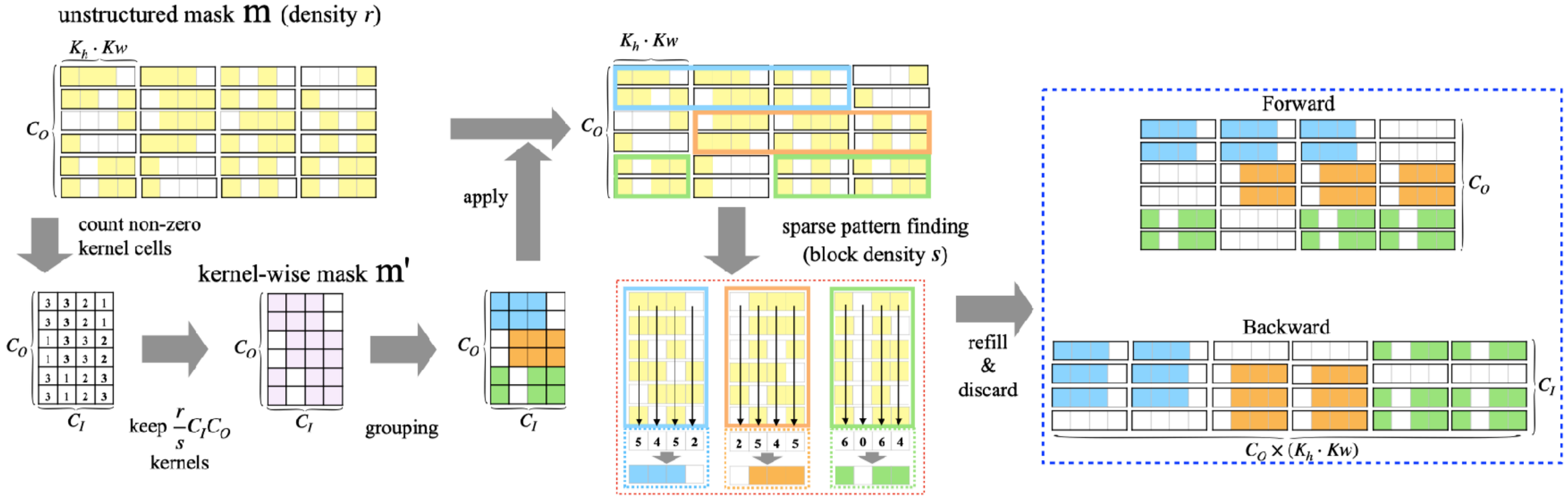
- Simply applying kernel-wise mask can easily discard all cells within a kernel and produce a large number of zero kernels
- Introduce sparsity (block density s) within each dense block to reduce all-zero kernels



Method: Hardware-friendly Regrouping for Block-wise Pruning

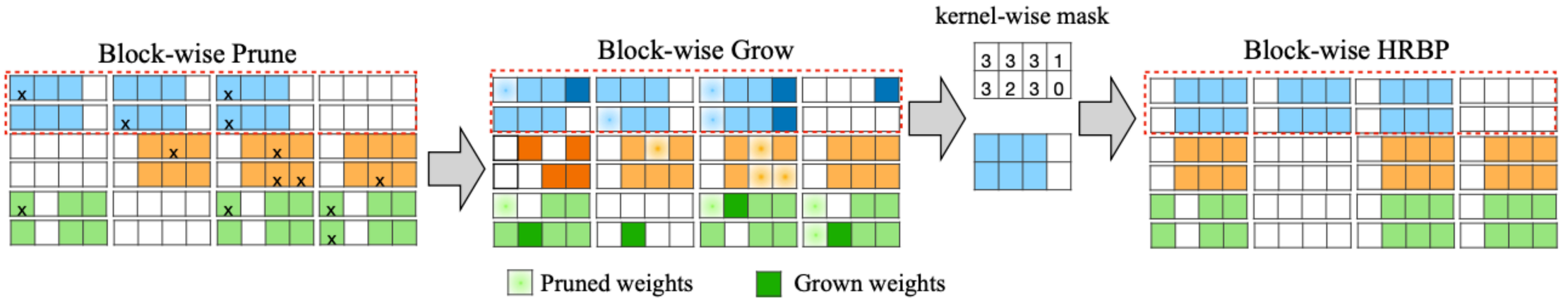
---- Summary of HRBP

- HRBP: a novel sparse pattern that preserve extracted dense blocks in backpropagation, enabling acceleration in both the forward and backward passes
  - Kernel-wise mask grouping
  - Exclusive block sparse pattern within each block



Method: Extend HRBP to Dynamic Sparse Training

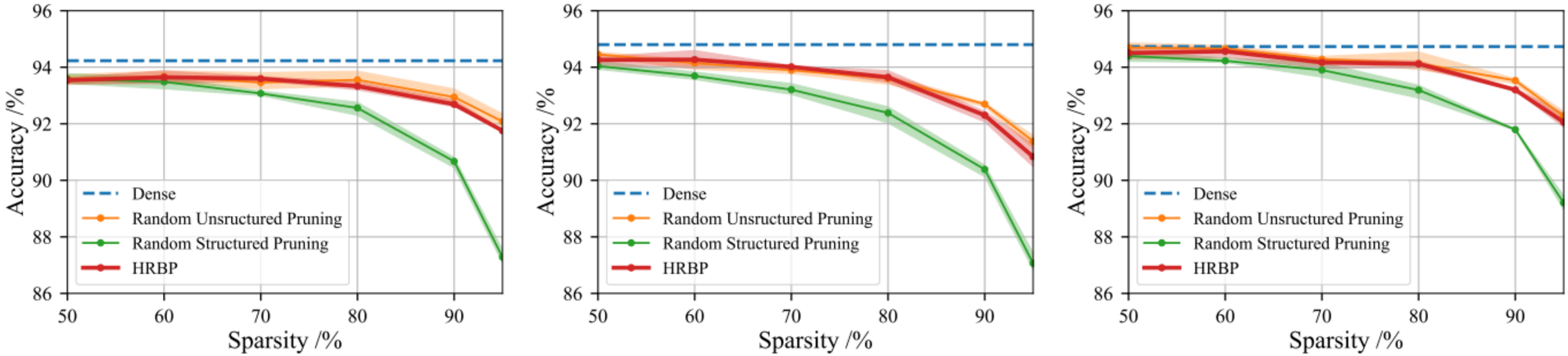
- Prune: magnitude-based
- Grow: gradient-based
- Block-wise updating: preserve the groups of rows (filters) and only updates the column-wise masks within each group



Results on Static Sparse Training

---- Accuracy

- HRBP achieves comparable performance to random unstructured pruning techniques and significantly outperforms channel-wise pruning across all sparsity levels.



(a) VGG-19 (b) ResNet-32 (c) ResNet-56

Results on Static Sparse Training

---- Acceleration

- HRBP can nearly match the training acceleration rate of channel-wise pruning at large sparsity ratios

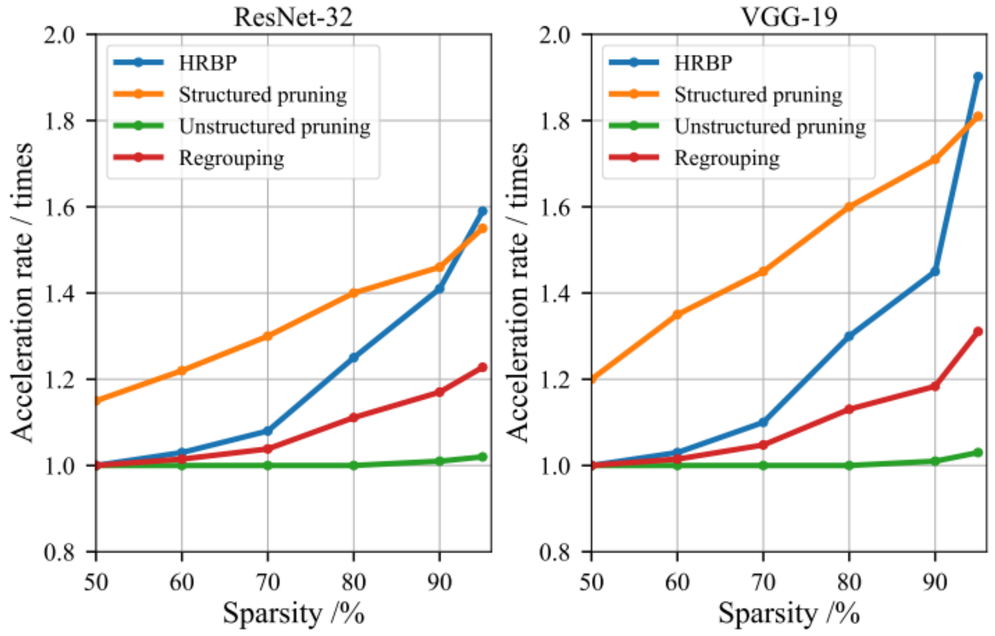


Figure 6: End-to-end training acceleration rate v.s. sparsity ratio of different sparsity schemes.

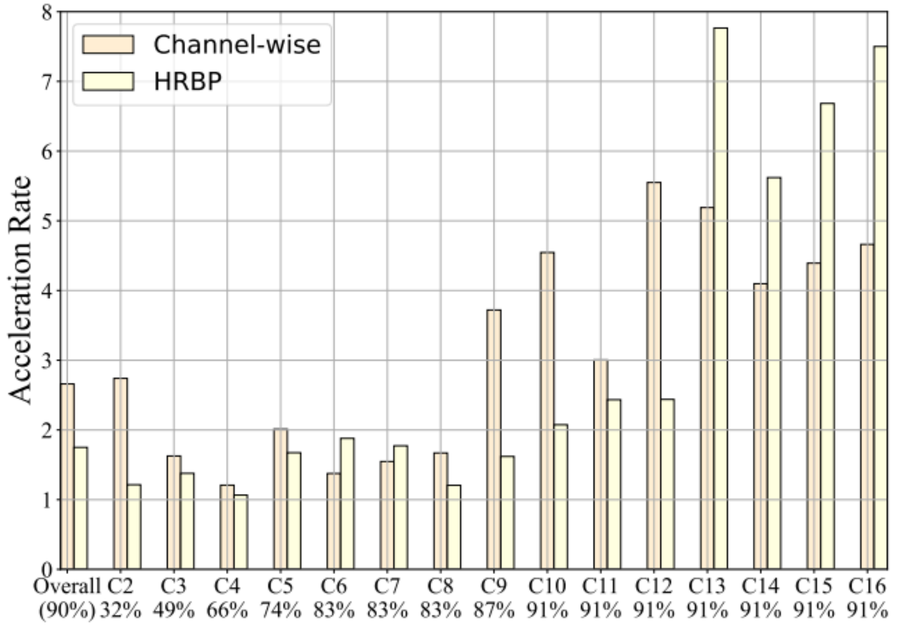


Figure 7: The layer-wise speedup of VGG-19 at inference time. The number in brackets is the corresponding sparsity.

Results on Static Sparse Training

---- Comparison

- HRBP can achieve similar performance as state-of-the-art carefully designed unstructured pruning approaches

Table 1: Comparison of static sparse training methods on CIFAR-10 and CIFAR-100. The number in brackets is the relative end-to-end training time speedup compared to a dense model.

Network	ResNet-32				VGG-19			
Dataset	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Dense	94.80		74.64		94.23		74.16	
Sparsity	90%	95%	90%	95%	90%	95%	90%	95%
RU	92.81±0.19 (1.0x)	91.38 ±0.04	69.48±0.21 (1.0x)	67.03±0.66 (1.0x)	92.91±0.10 (1.0x)	91.91±0.13 (1.0x)	70.39±0.43 (1.0x)	68.63±0.40 (1.0x)
LTH [5]	92.31 (1.0x)	91.06 (1.0x)	68.99 (1.0x)	65.02 (1.0x)	93.51 (1.0x)	92.92 (1.0x)	72.78 (1.0x)	71.44 (1.0x)
SNIP [6]	92.59 (1.0x)	91.01 (1.0x)	68.89 (1.0x)	65.22 (1.0x)	93.63 (1.0x)	93.43 (1.0x)	72.84 (1.0x)	71.83 (1.0x)
GraSP [7]	92.38 (1.0x)	91.39 (1.0x)	69.24 (1.0x)	66.50 (1.0x)	93.30 (1.0x)	93.04 (1.0x)	71.95 (1.0x)	71.23 (1.0x)
RC	90.27±0.24 (1.5x)	87.19±0.48 (1.6x)	62.71±0.22 (1.5x)	56.33 ±0.37 (1.6x)	90.84±0.38 (1.7x)	87.12±0.18 (1.8x)	59.61±0.52 (1.7x)	49.31±1.17 (1.8x)
Grouping [21]	91.63±0.11 (1.1x)	90.77±0.07 (1.2x)	66.97±0.18 (1.1x)	64.35±0.45 (1.2x)	92.81±0.25 (1.2x)	91.86±0.24 (1.2x)	70.52±0.36 (1.2x)	68.60±0.08 (1.2x)
HRBP	92.30±0.20 (1.4x)	90.84±0.41 (1.6x)	69.22±0.50 (1.4x)	65.94±0.25 (1.6x)	92.88±0.12 (1.4x)	91.66±0.14 (1.9x)	70.25±0.29 (1.4x)	67.89±0.49 (1.9x)

Table 2: SST on ImageNet. The dense ResNet-50 has 75.70% top-1 accuracy.

Sparsity	60%		80%	
Accuracy	top-1	top-5	top-1	top-5
SNIP [6]	73.95 (1.0×)	91.97	69.67 (1.0×)	89.24
GraSP [7]	74.02 (1.0×)	91.86	72.06 (1.0×)	90.82
HRBP	74.84 (1.17×)	92.35	70.90 (1.26×)	89.93

Results on Dynamic Sparse Training

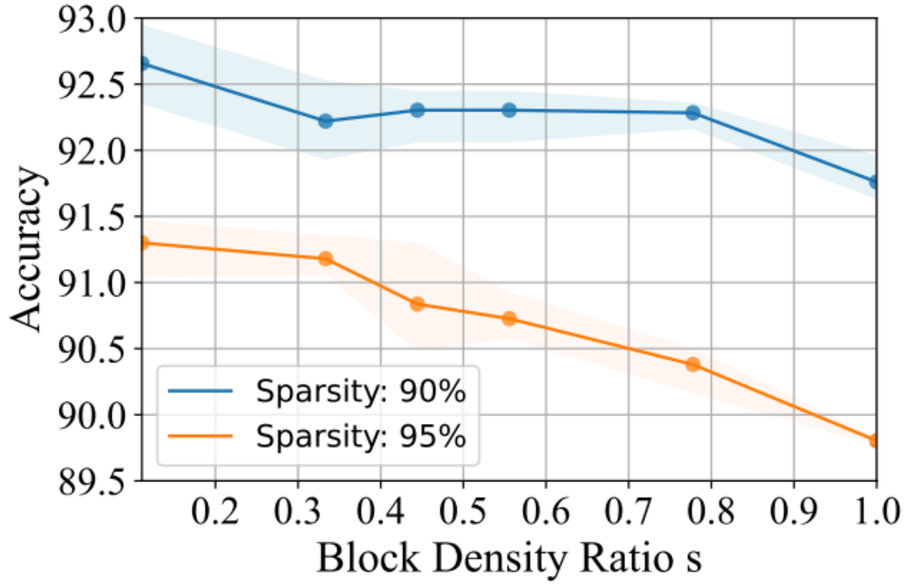
- HRBP with blockwise updating can match the accuracy of several unstructured DST methods

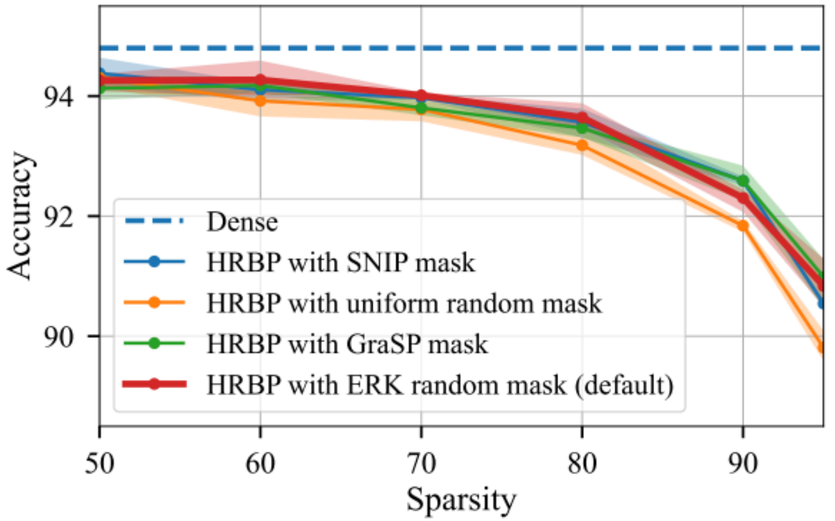
Table 3: Comparison of DST methods on CIFAR-10/100.

Network	ResNet-32				VGG-19			
Dataset	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Sparsity	90%	95%	90%	95%	90%	95%	90%	95%
Deep-R [39]	91.62	89.84	66.78	63.90	90.81	89.59	66.83	63.46
DSR [37]	92.97	91.61	69.63	68.20	93.75	93.86	72.31	71.98
SET [9]	92.30	90.76	69.66	67.41	92.46	91.73	72.36	69.81
RigL [10]	92.84±0.13	92.02±0.29	70.98±0.30	68.50±0.15	93.15±0.09	92.30±0.43	71.63±0.28	69.13±0.46
HRBP	92.72±0.23	91.25±0.12	69.51±0.52	66.41±0.30	93.07±0.12	91.79±0.18	70.49±0.19	68.04±0.37

Ablation Studies

- Block Density Ratio
  - smaller block density ratio leads to better performance
- Mask initialization
  - HRBP achieves similar performance with different mask initialization methods





Thanks!