Foundations of Edge AI

Lecture08

Neural Network Pruning (Cont'd)

Lanyu (Lori) Xu

Email: lxu@oakland.edu
Homepage: https://lori930.github.io/
Office: EC 524

Formulate Pruning

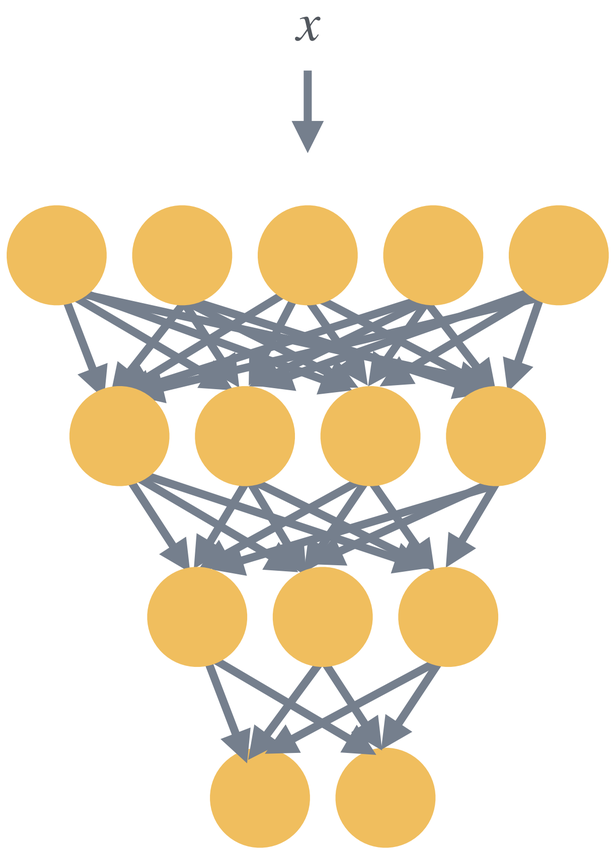
- In general, we could formulate the pruning as follows:

$$arg \min_{W_P} L(X; W_P)$$

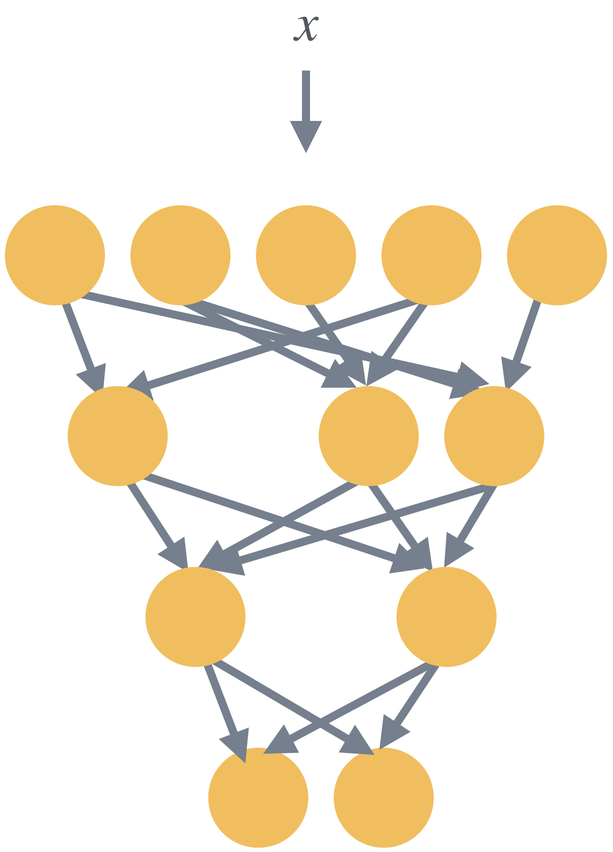
Subject to

$$\| W_P \|_0 \leq N$$

- L represents the objective function for neural network training
- X is input, W is the original weights, W_P is pruned weights
- $\| W_P \|_0$ calculates the #nonzeros in W_P, N is the target #nonzeros



$$\arg \min_{W} L(X; W)$$

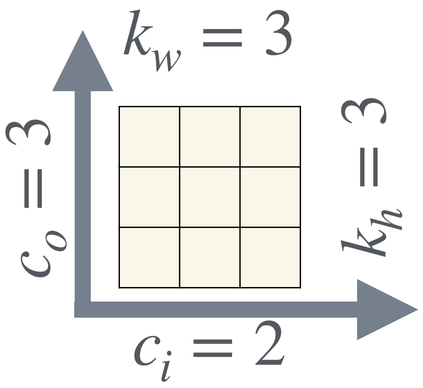


$$\arg \min_{W_P} L(X; W_P)$$
$$s.t. \| W_P \|_0 \leq N$$

Pruning at Different Granularities

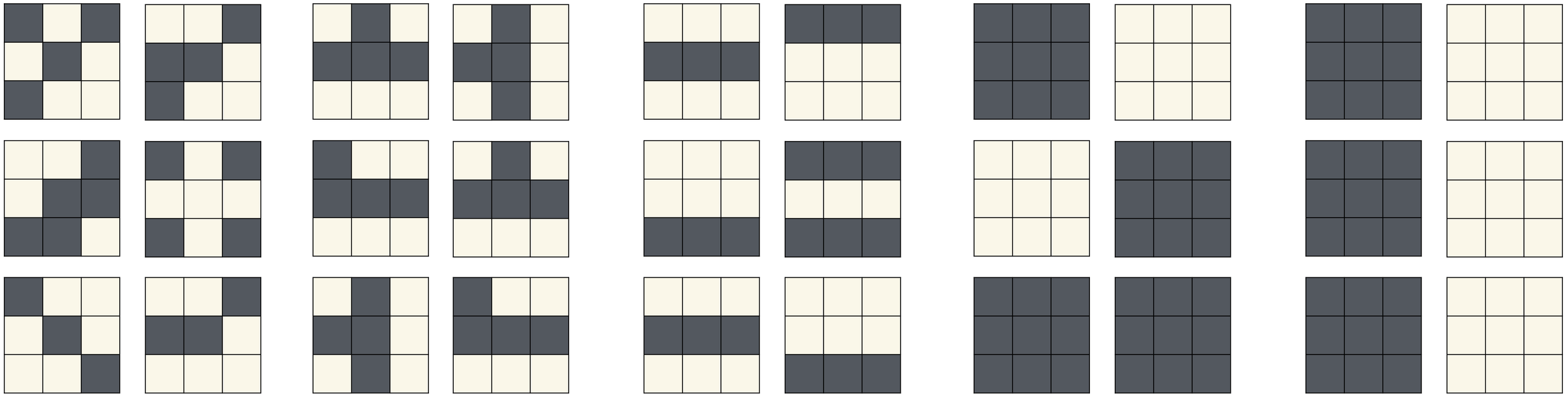
The case of convolutional layers

- Some of the commonly used pruning granularities



Preserved **Pruned**

Irregular **Regular**

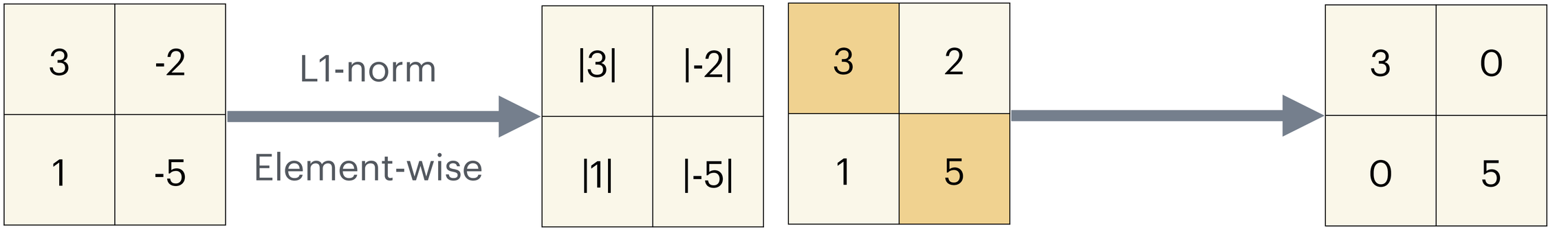


Fine-grained Pruning **Pattern-based Pruning** **Vector-level Pruning** **Kernel-level Pruning** **Channel-level Pruning**

Mao, H., Han, S., Pool, J., Li, W., Liu, X., Wang, Y., & Dally, W. J. (2017). Exploring the granularity of sparsity in convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 13-20).

Selection of Synapses to Prune

- When removing **parameters** from a neural network model
  - **The less important** the parameters being removed are, the better the performance of the pruned neural network is.
- **Magnitude-based pruning** considers weights with **larger absolute values** are more important than other weights
  - For element-wise pruning: Importance $= |W|$

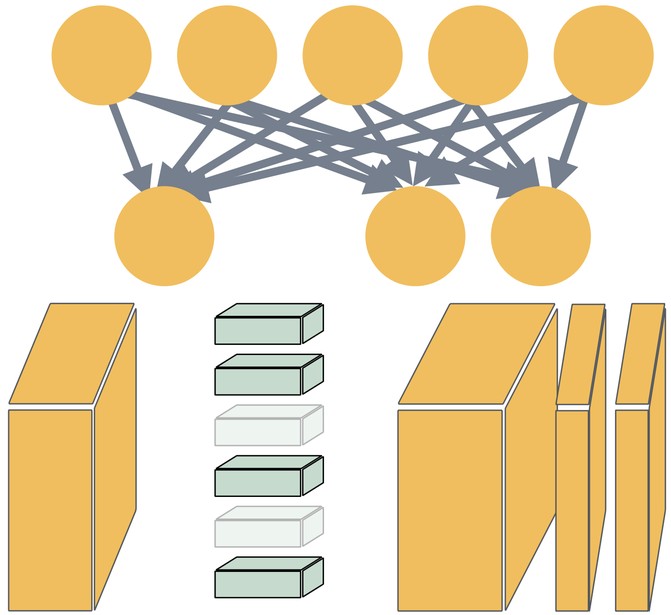


Selection of Neurons to Prune

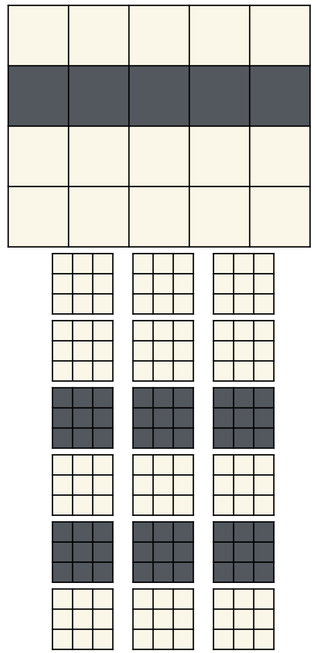
- When removing **neurons** from a neural network model
  - The **less useful** the neurons being removed are, the better the performance of the pruned neural network is.
- Neuron pruning is coarse-grained weight pruning

Neuron Pruning in Linear Layer

Channel Pruning in Convolution Layer



Weight Matrix



Lecture Plan

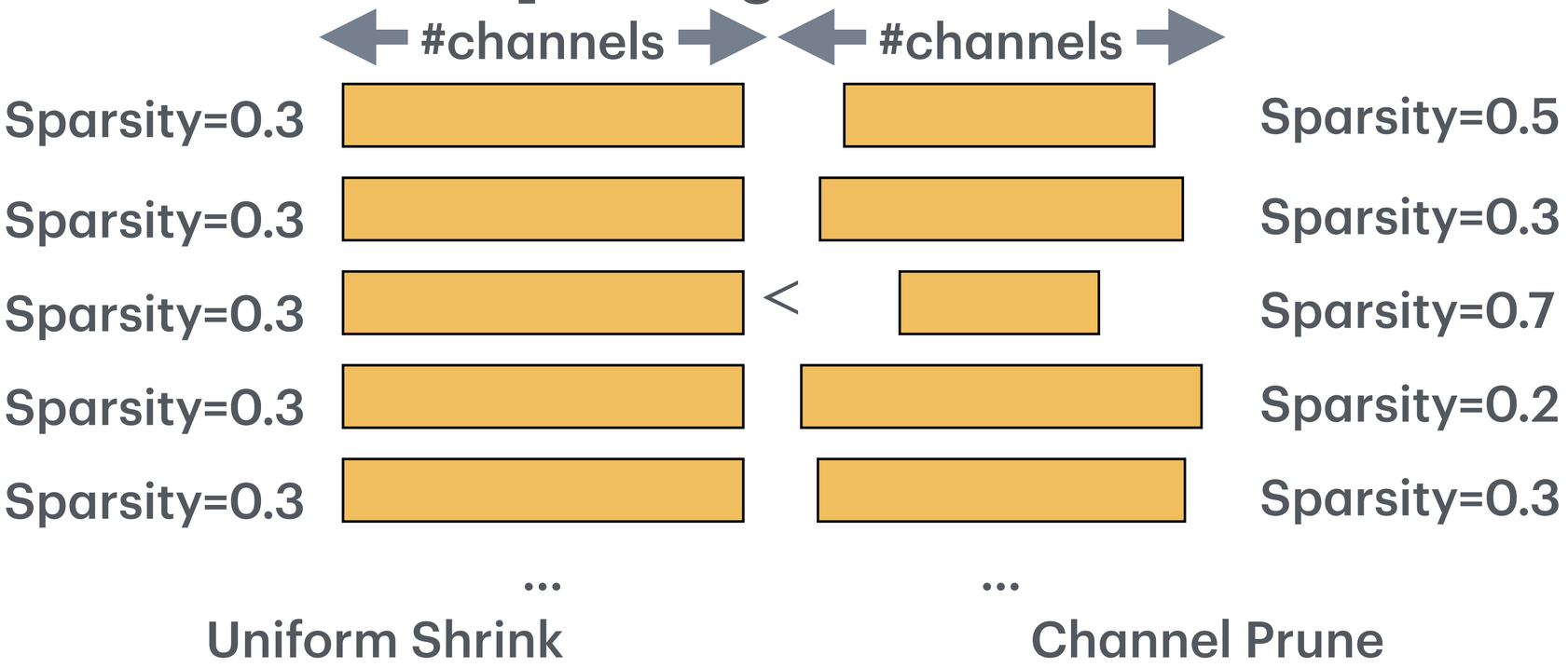
Today we will:

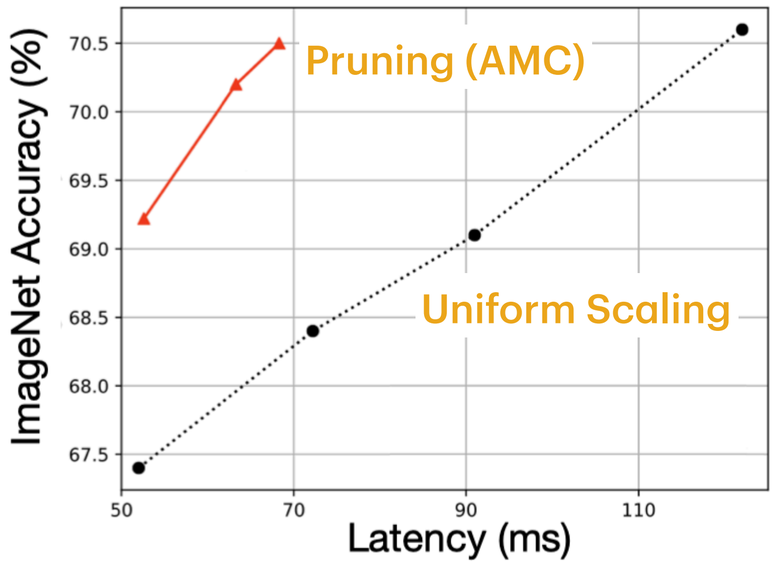
- Go through all steps of pruning
  - How to select pruning ratio
  - How to fine tune in NN pruning
- Learn system and hardware support for sparsity
  - Weight sparsity support
  - Activation sparsity support
- Lab 2 is out
  - Use the previous two colab files as your references

How to Choose Pruning Ratio?

Recap: Channel-Level Pruning

Non-uniform pruning is better than uniform shrinking



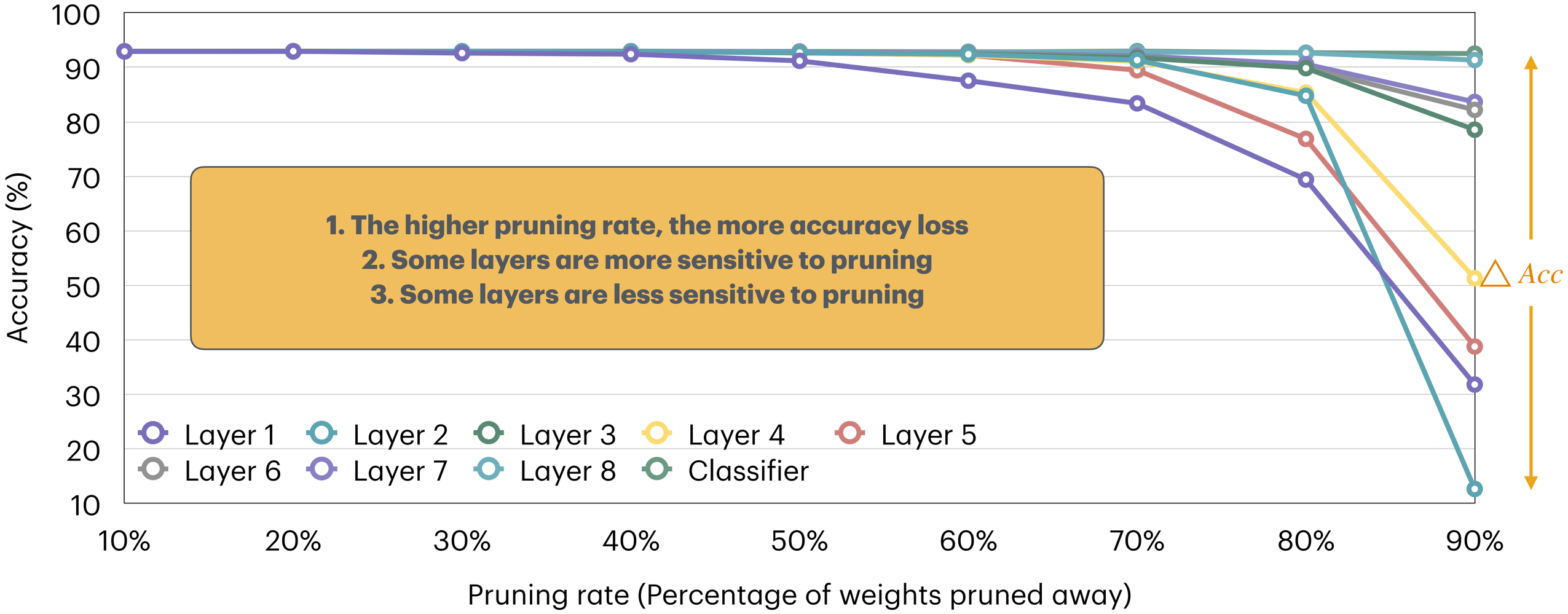


How should we find ratios for each layer?

He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). AMC: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

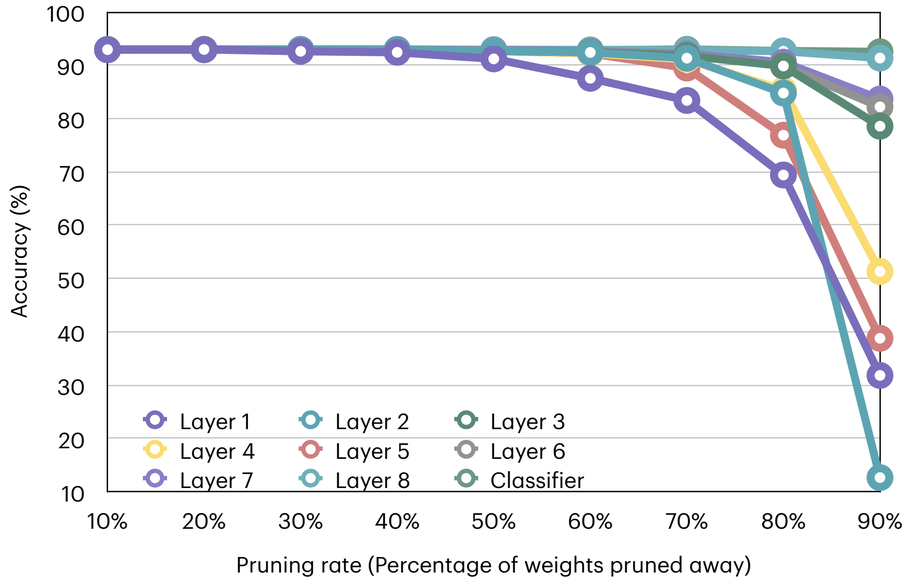
Pruning Layers of VGG on CIFAR-10 Dataset

Sensitivity Analysis



Finding Pruning Ratios

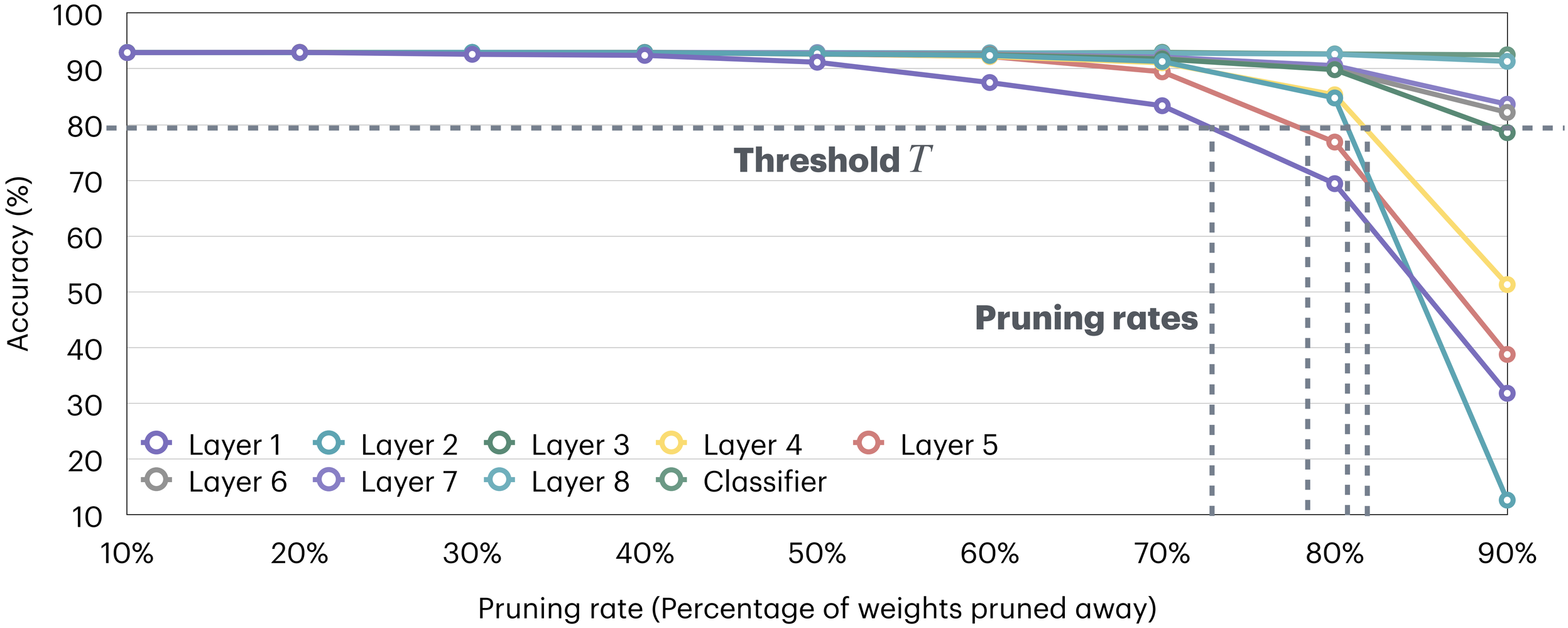
Analyze the sensitivity of each layer



- Each layers have different sensitivity
  - Some layers are more sensitive
  - Some layers are more redundant
- Perform **sensitivity analysis** to determine the per-layer pruning ratio
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other stides)
    - Observe the accuracy degrade $\triangle Acc_r^i$ for each pruning ratio
  - Repeat the process for all layers
  - Pick a degradation threshold T such that the overall pruning rate is desired

Finding Pruning Ratios

Pick a degradation threshold T



Finding Pruning Ratios

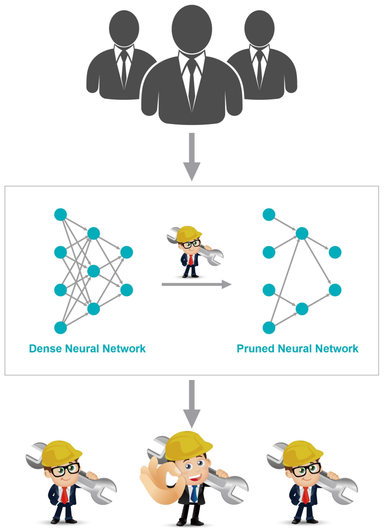
Analyze the sensitivity of each layer

- Is this optimal?
  - Maybe not. We do not consider the interaction between layers. 😔
- Can we go beyong the heuristics?

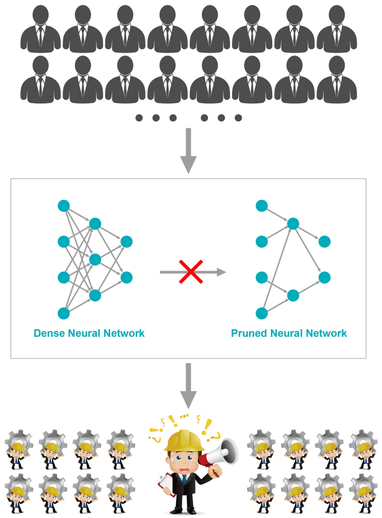
Automatic Pruning

- Given an overall compression ratio, how do we choose per-layer pruning ratios?
- Conventionally, such process relies on human expertise and trials and errors

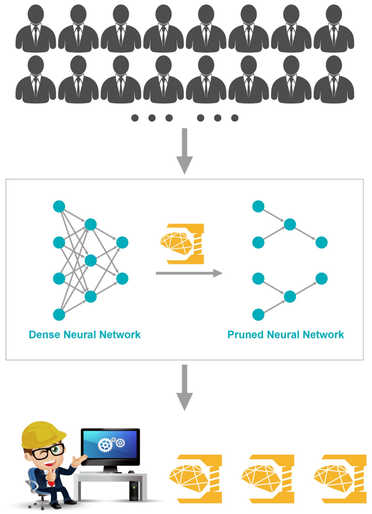
AlexNet: 8 layers
VGG-16: 16 layers



ResNet: 50 layers



Larger models?



Automatic Pruning

AMC: AutoML for Model Compression and Acceleration on Mobile Devices

Yihui He[‡*], Ji Lin[†*], Zhijian Liu[†], Hanrui Wang[†], Li-Jia Li[↕], and Song Han[†]
{jilin, songhan}@mit.edu

[†]Massachusetts Institute of Technology
[‡]Carnegie Mellon University
[↕]Google

Abstract. Model compression is a critical technique to efficiently deploy neural network models on mobile devices which have limited computation resources and tight power budgets. Conventional model compression techniques rely on hand-crafted heuristics and *rule-based* policies that require domain experts to explore the large design space trading off among model size, speed, and accuracy, which is usually sub-optimal and time-consuming. In this paper, we propose AutoML for Model Compression (AMC) which leverage reinforcement learning to provide the model compression policy. This *learning-based* compression policy outperforms conventional *rule-based* compression policy by having higher compression ratio, better preserving the accuracy and freeing human labor. Under 4× FLOPs reduction, we achieved **2.7%** better accuracy than the hand-crafted model compression policy for VGG-16 on ImageNet. We applied this automated, push-the-button compression pipeline to MobileNet and achieved **1.81**× speedup of measured inference latency on an Android phone and **1.43**× speedup on the Titan XP GPU, with only 0.1% loss of ImageNet Top-1 accuracy.

NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications

Tien-Ju Yang[1⋆[0000−0003−4728−0321]], Andrew Howard[2], Bo Chen[2], Xiao Zhang[2], Alec Go[2], Mark Sandler[2], Vivienne Sze[1], and Hartwig Adam[2]

[1] Massachusetts Institute of Technology
[2] Google Inc.
{tjy,sze}@mit.edu, {howarda,bochen,andypassion,ago,sandler,hadam}@google.com

Abstract. This work proposes an algorithm, called NetAdapt, that *automatically adapts* a pre-trained deep neural network to a mobile platform given a resource budget. While many existing algorithms simplify networks based on the number of MACs or weights, optimizing those indirect metrics may not necessarily reduce the direct metrics, such as latency and energy consumption. To solve this problem, NetAdapt incorporates direct metrics into its adaptation algorithm. These direct metrics are evaluated using *empirical measurements*, so that detailed knowledge of the platform and toolchain is not required. NetAdapt automatically and progressively simplifies a pre-trained network until the resource budget is met while maximizing the accuracy. Experiment results show that NetAdapt achieves better accuracy versus latency trade-offs on both mobile CPU and mobile GPU, compared with the state-of-the-art automated network simplification algorithms. For image classification on the ImageNet dataset, NetAdapt achieves up to a 1.7× speedup in *measured inference latency* with equal or higher accuracy on MobileNets (V1&V2).

Automatic Pruning

AMC: AutoML for Model Compression and Acceleration on Mobile Devices

Yihui He[‡*], Ji Lin[†*], Zhijian Liu[†], Hanrui Wang[†], Li-Jia Li[↕], and Song Han[†]
{jilin, songhan}@mit.edu

[†]Massachusetts Institute of Technology
[‡]Carnegie Mellon University
[↕]Google

Abstract. Model compression is a critical technique to efficiently deploy neural network models on mobile devices which have limited computation resources and tight power budgets. Conventional model compression techniques rely on hand-crafted heuristics and *rule-based* policies that require domain experts to explore the large design space trading off among model size, speed, and accuracy, which is usually sub-optimal and time-consuming. In this paper, we propose AutoML for Model Compression (AMC) which leverage reinforcement learning to provide the model compression policy. This *learning-based* compression policy outperforms conventional *rule-based* compression policy by having higher compression ratio, better preserving the accuracy and freeing human labor. Under 4× FLOPs reduction, we achieved **2.7%** better accuracy than the hand-crafted model compression policy for VGG-16 on ImageNet. We applied this automated, push-the-button compression pipeline to MobileNet and achieved **1.81**× speedup of measured inference latency on an Android phone and **1.43**× speedup on the Titan XP GPU, with only 0.1% loss of ImageNet Top-1 accuracy.

NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications

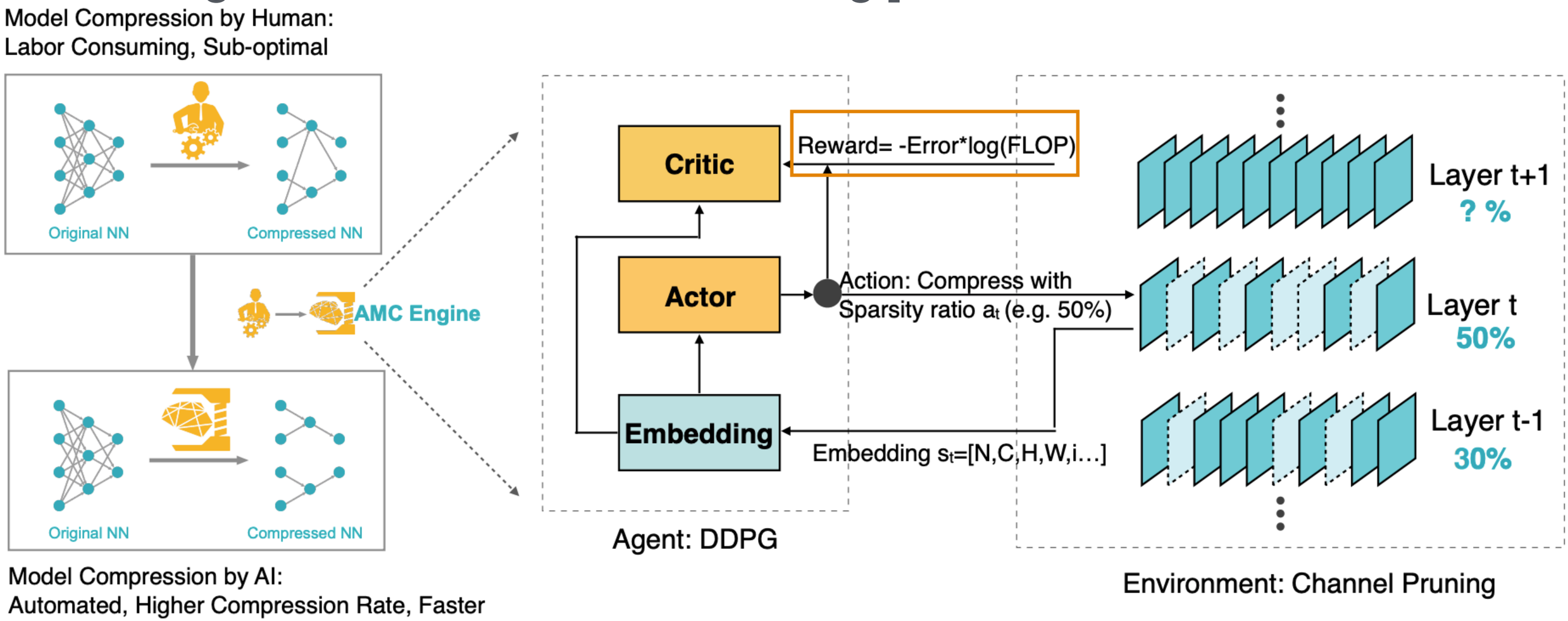
Tien-Ju Yang[1⋆[0000−0003−4728−0321]], Andrew Howard[2], Bo Chen[2], Xiao Zhang[2], Alec Go[2], Mark Sandler[2], Vivienne Sze[1], and Hartwig Adam[2]

[1] Massachusetts Institute of Technology
[2] Google Inc.
{tjy,sze}@mit.edu, {howarda,bochen,andypassion,ago,sandler,hadam}@google.com

Abstract. This work proposes an algorithm, called NetAdapt, that *automatically adapts* a pre-trained deep neural network to a mobile platform given a resource budget. While many existing algorithms simplify networks based on the number of MACs or weights, optimizing those indirect metrics may not necessarily reduce the direct metrics, such as latency and energy consumption. To solve this problem, NetAdapt incorporates direct metrics into its adaptation algorithm. These direct metrics are evaluated using *empirical measurements*, so that detailed knowledge of the platform and toolchain is not required. NetAdapt automatically and progressively simplifies a pre-trained network until the resource budget is met while maximizing the accuracy. Experiment results show that NetAdapt achieves better accuracy versus latency trade-offs on both mobile CPU and mobile GPU, compared with the state-of-the-art automated network simplification algorithms. For image classification on the ImageNet dataset, NetAdapt achieves up to a 1.7× speedup in *measured inference latency* with equal or higher accuracy on MobileNets (V1&V2).

AMC: AutoML for Model Compression

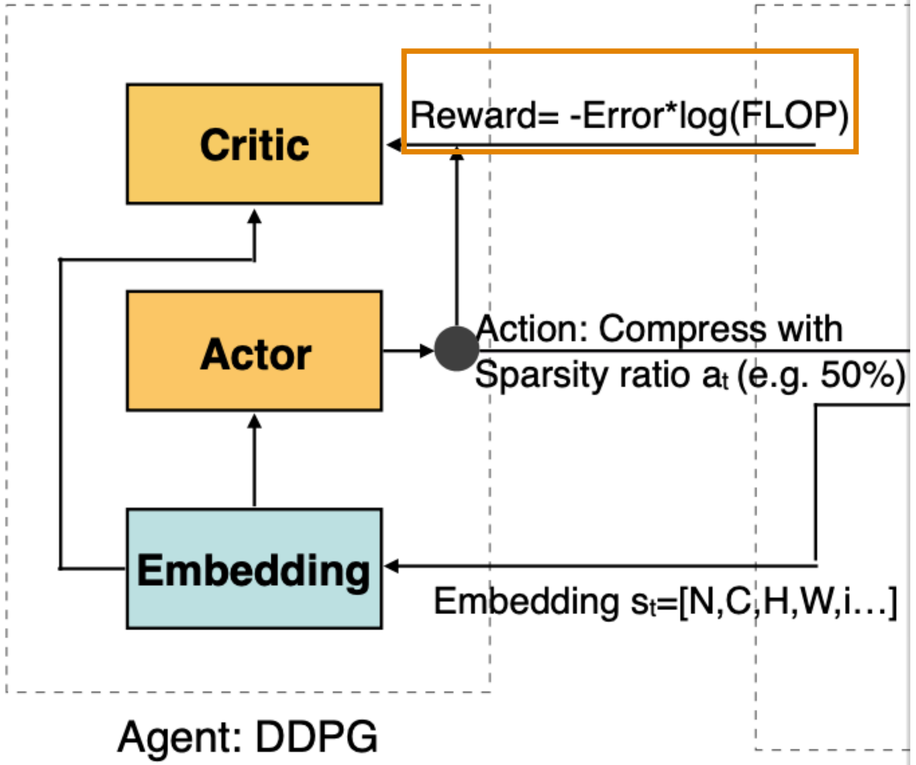
Pruning as an reinforcement learning problem

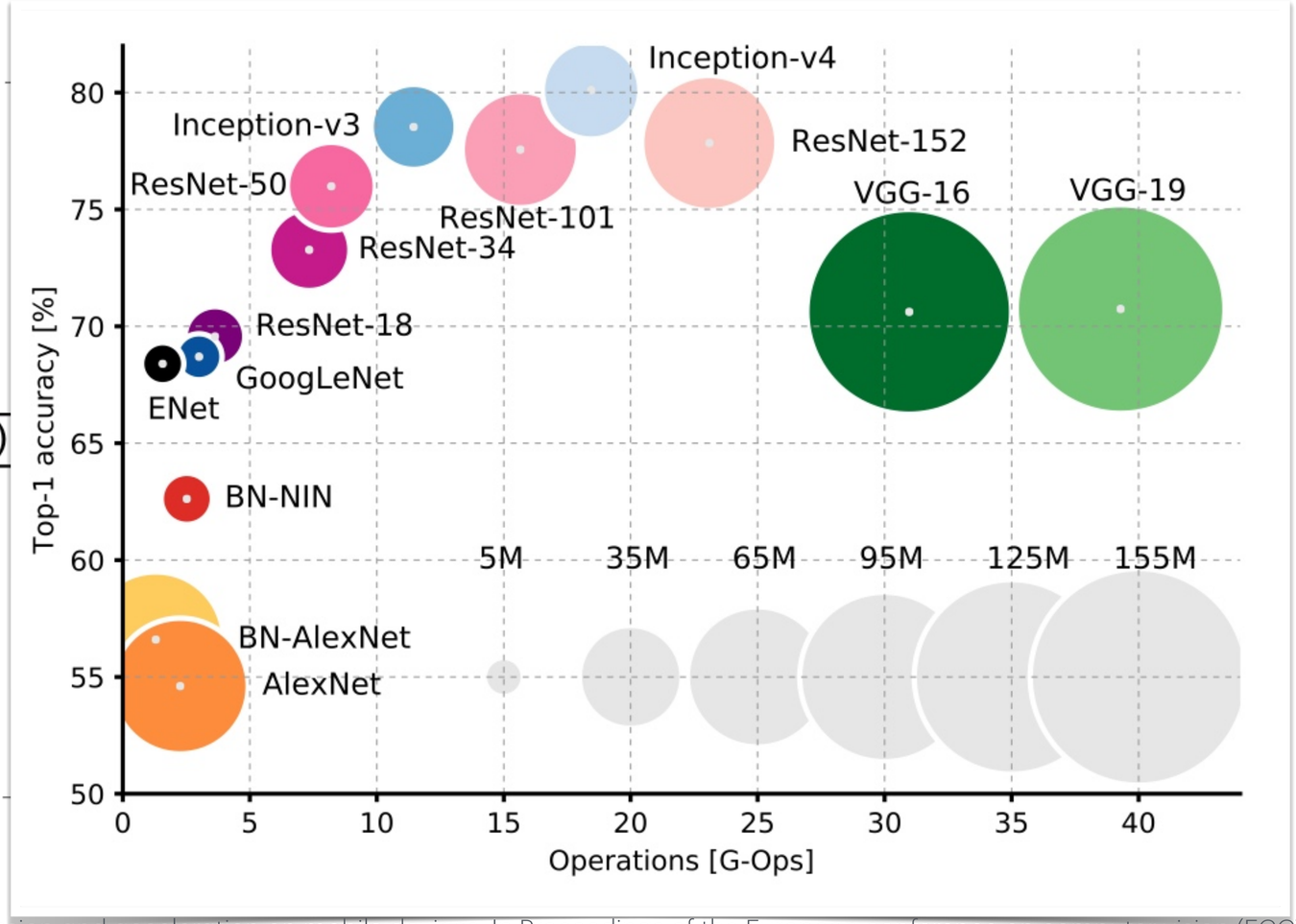


He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

AMC: AutoML for Model Compression

Pruning as a reinforcement learning problem





He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

AMC: AutoML for Model Compression

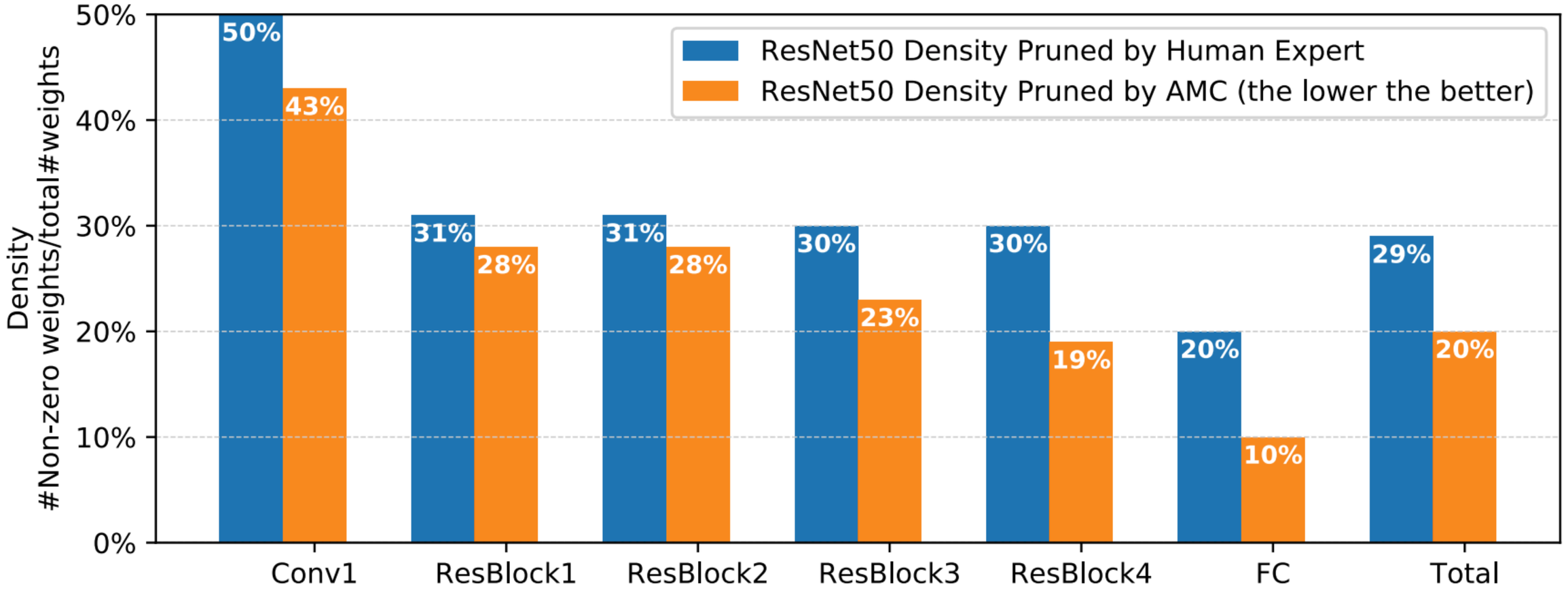
Pruning as a reinforcement learning problem

- Setup for the reinforcement learning (RL) problem
  - **State**: features including layer indices, channel numbers, kernel sizes, FLOPs, ...
  - **Action**: A continuous number (pruning ratio) $a \in [0,1)$
  - **Agent**: DDPG agent, since it supports continuous action output
  - **Reward**: $R = \begin{cases} -\text{Error}, & \text{if satisfies constrains} \\ -\infty, & \text{if not} \end{cases}$
    - Optimize **latency** constrains by a pre-built lookup table (LUT)

He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

AMC: AutoML for Model Compression

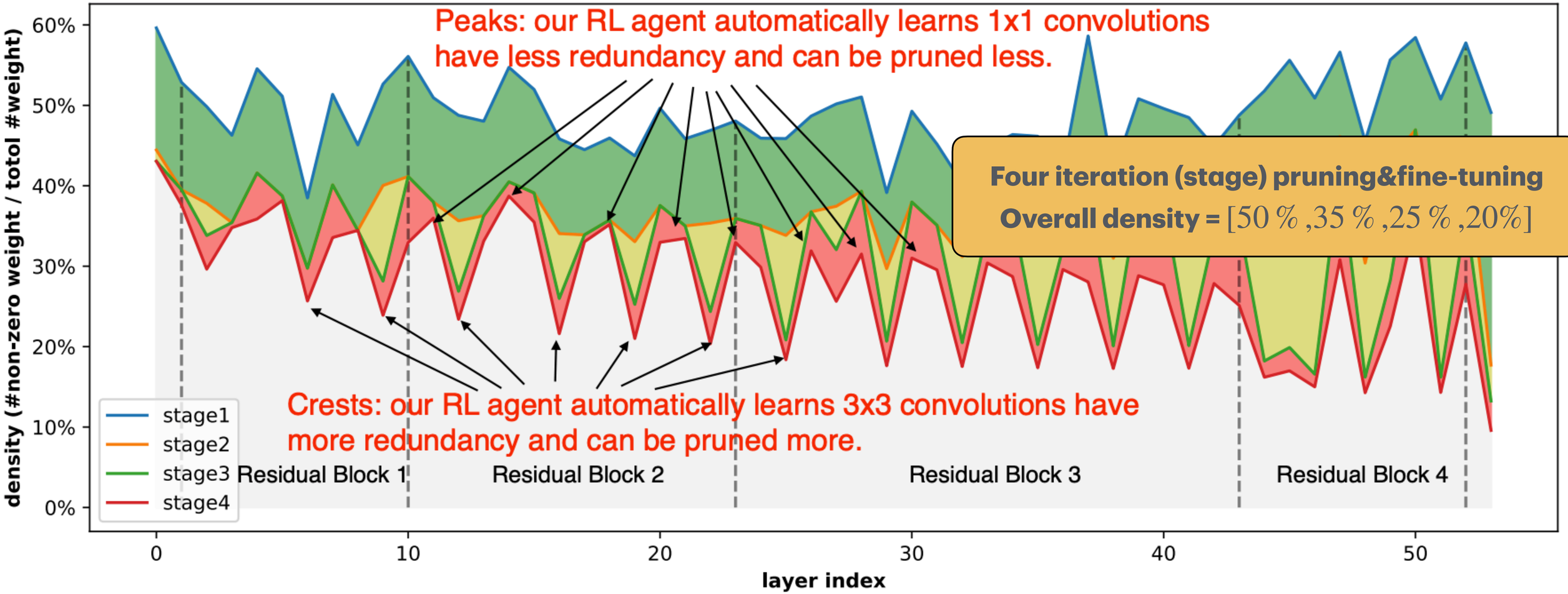
AMC prunes the model to a lower density



He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

AMC: AutoML for Model Compression

The pruning policy (sparsity ratio) given by the RL agent for ResNet-50



He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

AMC: AutoML for Model Compression

Latency oriented results

Model	MAC	Top-1	Latency	Speedup	Memory
1.0 MobileNet	569M	70.6%	119.0ms	1X	20.1MB
AMC (50% FLOPs)	285M	**70.5%**			MB
AMC (50% Time)	272M	**70.2%**			MB
0.75 MobileNet	325M	68.4%	69.5ms	1.7X	14.8MB

🤔 **Why 25% pruning gets 70% speedup?**

Measured with TF-Lite on Samsung Galaxy S7 Edge (which/Qualcomm Snapdragon SoC single core).
Batch size = 1

He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

AMC: AutoML for Model Compression

Latency oriented results

- Why 25% pruning gets 70% speedup?
- Recall: FLOPS calculation of a conv layer?
- $c_i \rightarrow 0.75c_i,\ c_o \rightarrow 0.75c_o$
- FLOPS $\rightarrow 0.75 \times 0.75$ FLOPS ≈ 0.7 FLOPS

Layer	#MACs (batch size n = 1)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_i \cdot k_h \cdot k_w \cdot c_o \cdot h_o \cdot w_o$
Grouped Convolution	$c_i/g \cdot k_h \cdot k_w \cdot c_o \cdot h_o \cdot w_o$
Depthwise Convolution	$k_h \cdot k_w \cdot c_o \cdot h_o \cdot w_o$

0.75 MobileNet	325M	68.4%	69.5ms	1.7X	14.8MB

Measured with TF-Lite on Samsung Galaxy S7 Edge (which/Qualcomm Snapdragon SoC single core).
Batch size = 1

He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

Automatic Pruning

AMC: AutoML for Model Compression and Acceleration on Mobile Devices

Yihui He[‡*], Ji Lin[†*], Zhijian Liu[†], Hanrui Wang[†], Li-Jia Li[‡], and Song Han[†]
{jilin, songhan}@mit.edu

[†]Massachusetts Institute of Technology
[‡]Carnegie Mellon University
[‡]Google

Abstract. Model compression is a critical technique to efficiently deploy neural network models on mobile devices which have limited computation resources and tight power budgets. Conventional model compression techniques rely on hand-crafted heuristics and *rule-based* policies that require domain experts to explore the large design space trading off among model size, speed, and accuracy, which is usually sub-optimal and time-consuming. In this paper, we propose AutoML for Model Compression (AMC) which leverage reinforcement learning to provide the model compression policy. This *learning-based* compression policy outperforms conventional *rule-based* compression policy by having higher compression ratio, better preserving the accuracy and freeing human labor. Under 4× FLOPs reduction, we achieved **2.7%** better accuracy than the hand-crafted model compression policy for VGG-16 on ImageNet. We applied this automated, push-the-button compression pipeline to MobileNet and achieved **1.81**× speedup of measured inference latency on an Android phone and **1.43**× speedup on the Titan XP GPU, with only 0.1% loss of ImageNet Top-1 accuracy.

NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications

Tien-Ju Yang[1⋆[0000−0003−4728−0321]], Andrew Howard[2], Bo Chen[2], Xiao Zhang[2], Alec Go[2], Mark Sandler[2], Vivienne Sze[1], and Hartwig Adam[2]

[1] Massachusetts Institute of Technology
[2] Google Inc.
{tjy,sze}@mit.edu, {howarda,bochen,andypassion,ago,sandler,hadam}@google.com

Abstract. This work proposes an algorithm, called NetAdapt, that *automatically adapts* a pre-trained deep neural network to a mobile platform given a resource budget. While many existing algorithms simplify networks based on the number of MACs or weights, optimizing those indirect metrics may not necessarily reduce the direct metrics, such as latency and energy consumption. To solve this problem, NetAdapt incorporates direct metrics into its adaptation algorithm. These direct metrics are evaluated using *empirical measurements*, so that detailed knowledge of the platform and toolchain is not required. NetAdapt automatically and progressively simplifies a pre-trained network until the resource budget is met while maximizing the accuracy. Experiment results show that NetAdapt achieves better accuracy versus latency trade-offs on both mobile CPU and mobile GPU, compared with the state-of-the-art automated network simplification algorithms. For image classification on the ImageNet dataset, NetAdapt achieves up to a 1.7× speedup in *measured inference latency* with equal or higher accuracy on MobileNets (V1&V2).

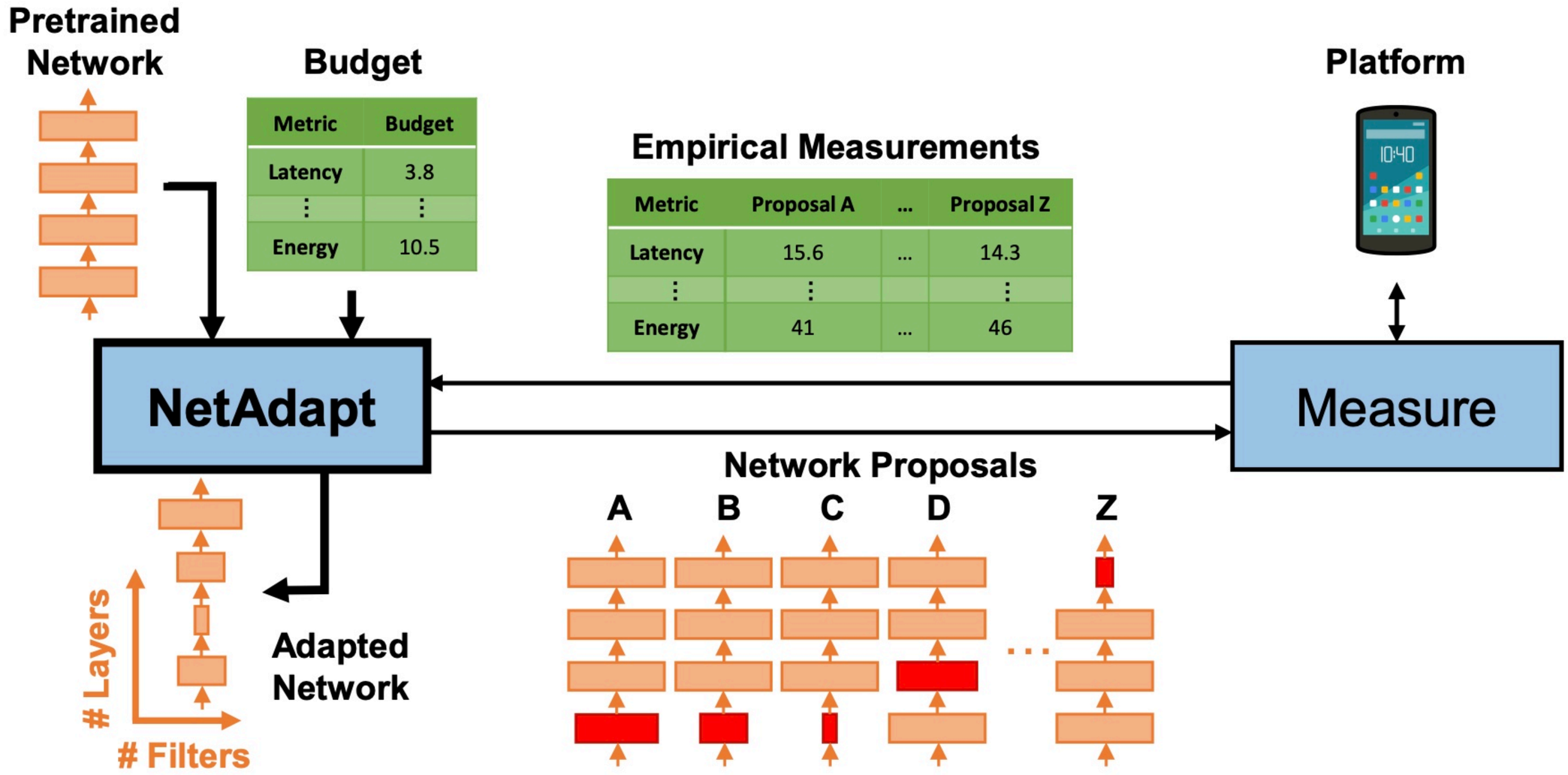
NetAdapt

Platform-Aware Neural Network Adaptation for Mobile Applications

- A rule-based **iterative**/progressive method
- Goal: find a per-layer pruning ratio to meet a global resource constraint (e.g., latency, energy)
- We will take **latency** constraint as an example to illustrate

Yang, T. J., Howard, A., Chen, B., Zhang, X., Go, A., Sandler, M., ... & Adam, H. (2018). Netadapt: Platform-aware neural network adaptation for mobile applications. In Proceedings of the European conference on computer vision (ECCV) (pp. 285-300).

NetAdapt

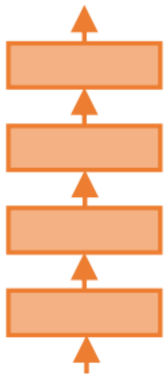


Yang, T. J., Howard, A., Chen, B., Zhang, X., Go, A., Sandler, M., ... & Adam, H. (2018). Netadapt: Platform-aware neural network adaptation for mobile applications. In Proceedings of the European conference on computer vision (ECCV) (pp. 285-300).

NetAdapt

Platform-Aware Neural Network Adaptation for Mobile Applications

1. For each iteration, reduce the latency by a certain amount $\triangle R$ (manually defined)



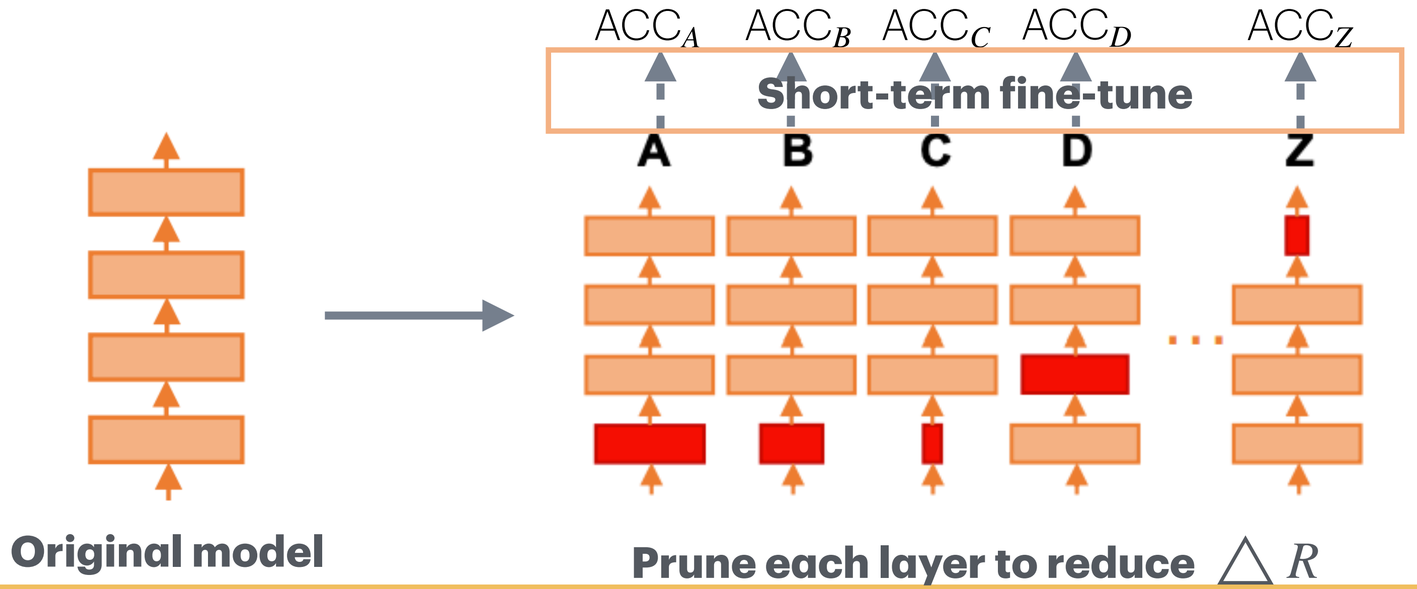
Original model

NetAdapt

Platform-Aware Neural Network Adaptation for Mobile Applications

(1) For each layer L_k (k in the $A - Z$ in the figure)

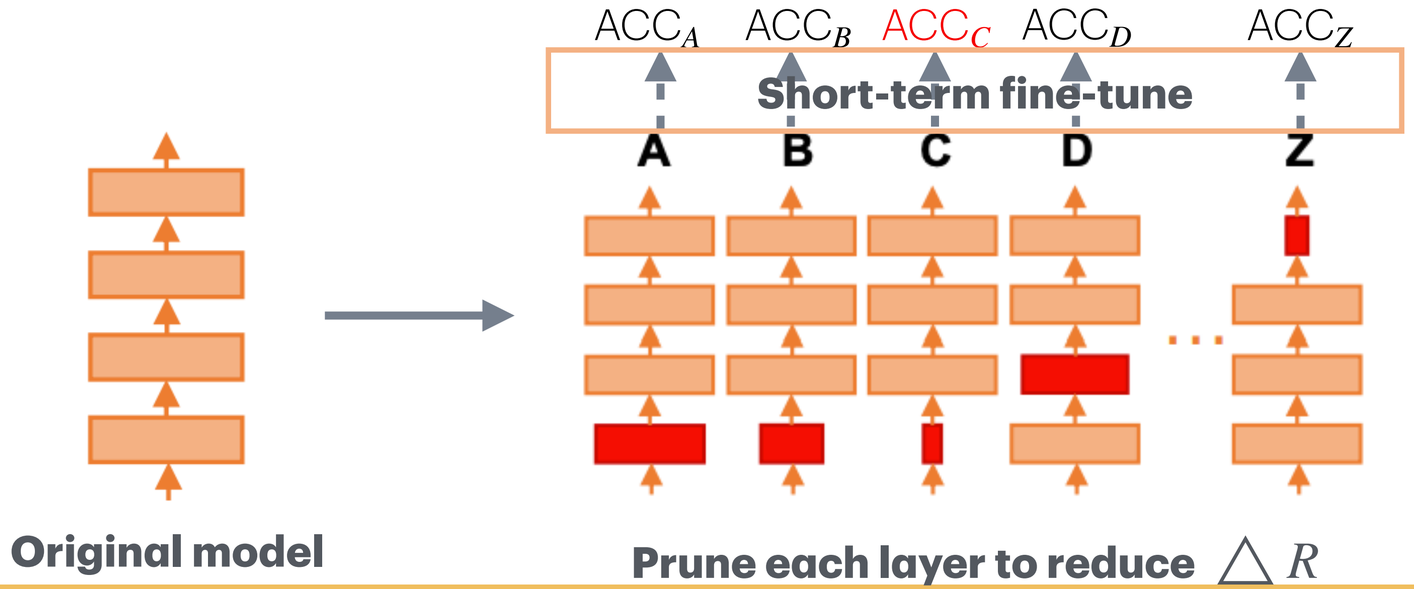
- Prune the layer s . t . the latency reduction meets $\triangle R$ (based on a pre-built lookup table)
- Short-term fine-tune model (10k iterations); measure accuracy after fine-tuning



NetAdapt

Platform-Aware Neural Network Adaptation for Mobile Applications

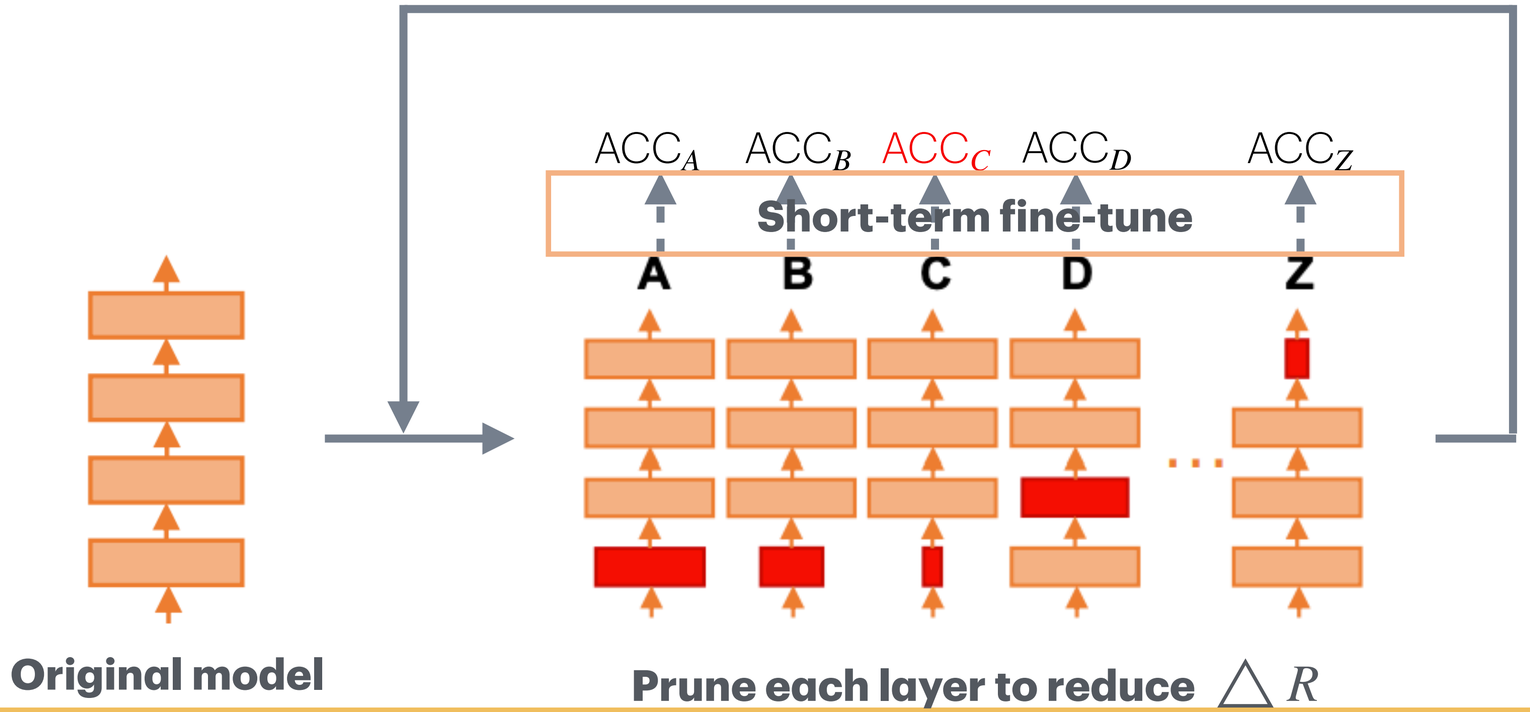
(2) Choose and prune the layer with the highest accuracy



NetAdapt

Platform-Aware Neural Network Adaptation for Mobile Applications

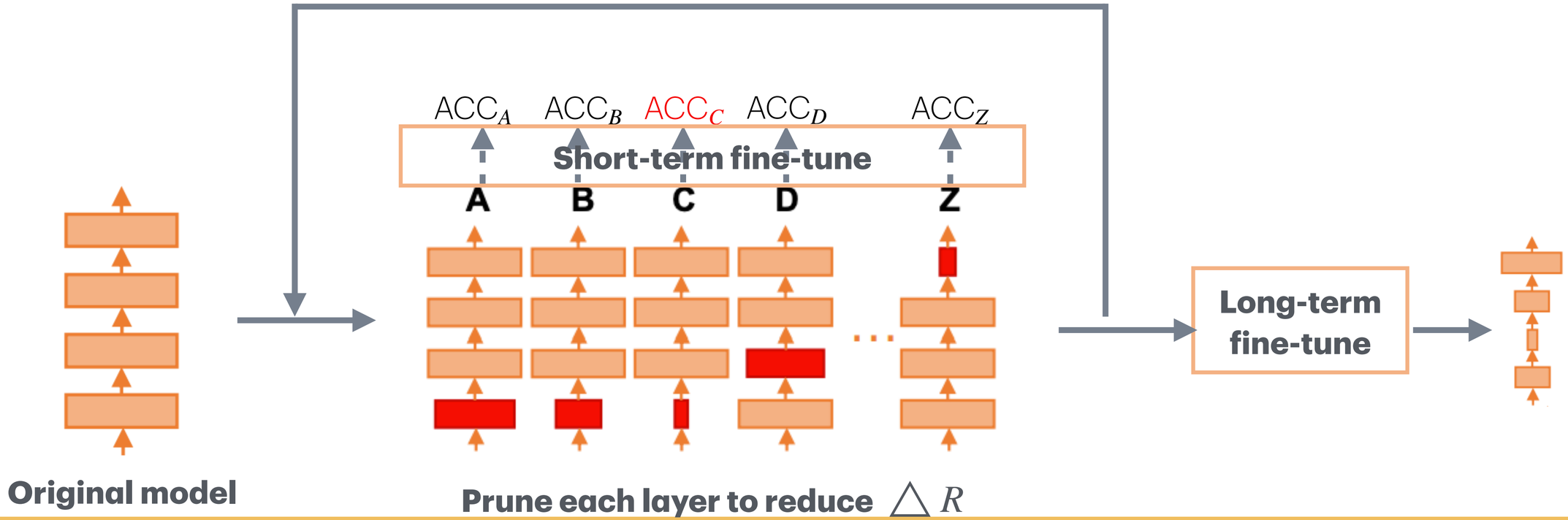
2. Repeat until the total latency reduction satifies the constraint



NetAdapt

Platform-Aware Neural Network Adaptation for Mobile Applications

3. Long-term fine-tune to recover accuracy



NetAdapt (Workflow)

Platform-Aware Neural Network Adaptation for Mobile Applications

1. For each iteration, reduce the latency by a certain amount $\triangle R$ (manually defined)

   (1) For each layer L_k (k in the $A - Z$ in the figure)

   - Prune the layer s . t . the latency reduction meets $\triangle R$ (based on a pre-built lookup table)
   - Short-term fine-tune model (10k iterations); measure accuracy after fine-tuning

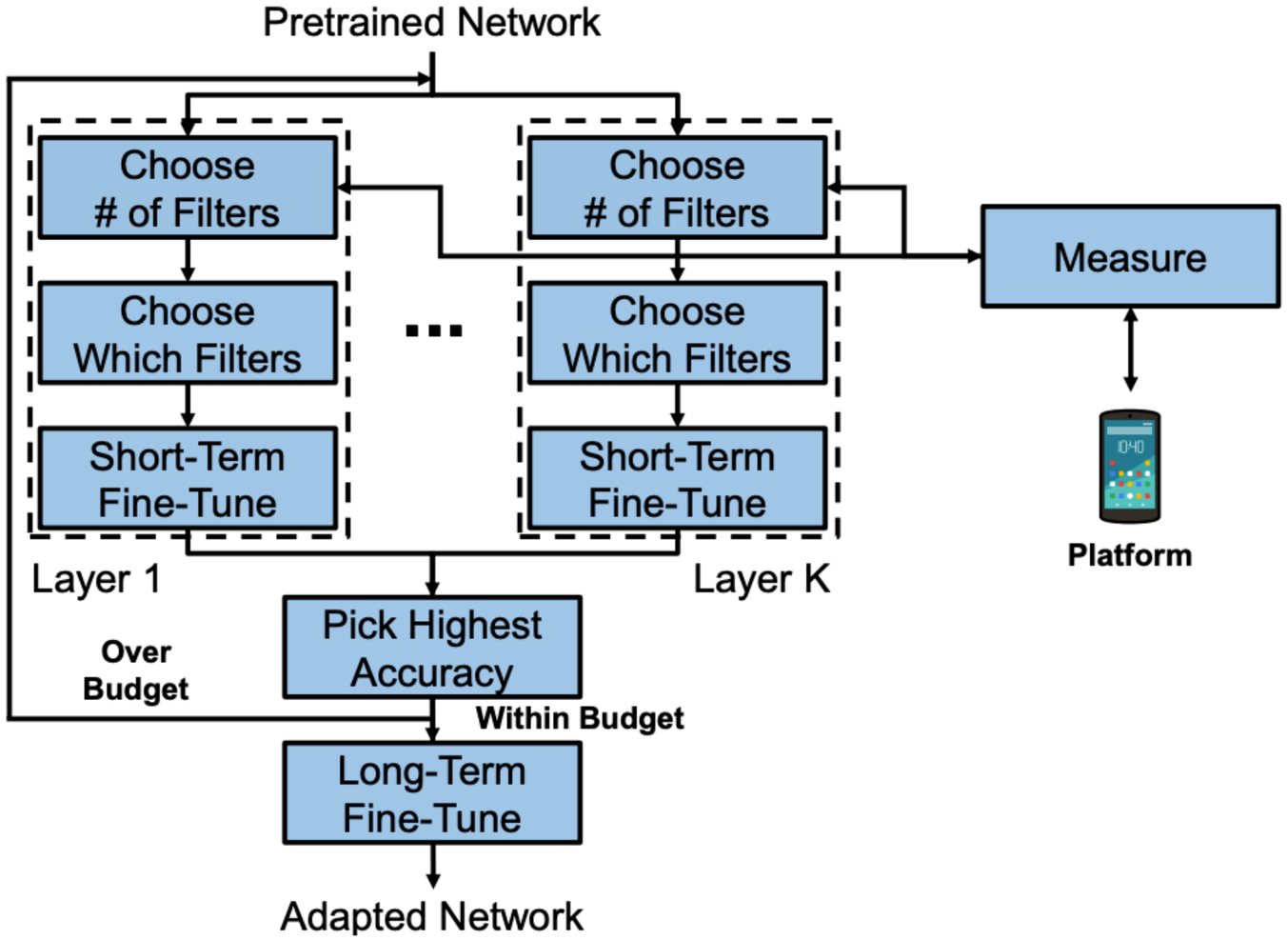
   (2) Choose and prune the layer with the highest accuracy

2. Repeat until the total latency reduction satifies the constraint

3. Long-term fine-tune to recover accuracy

NetAdapt

Visualization of the Workflow

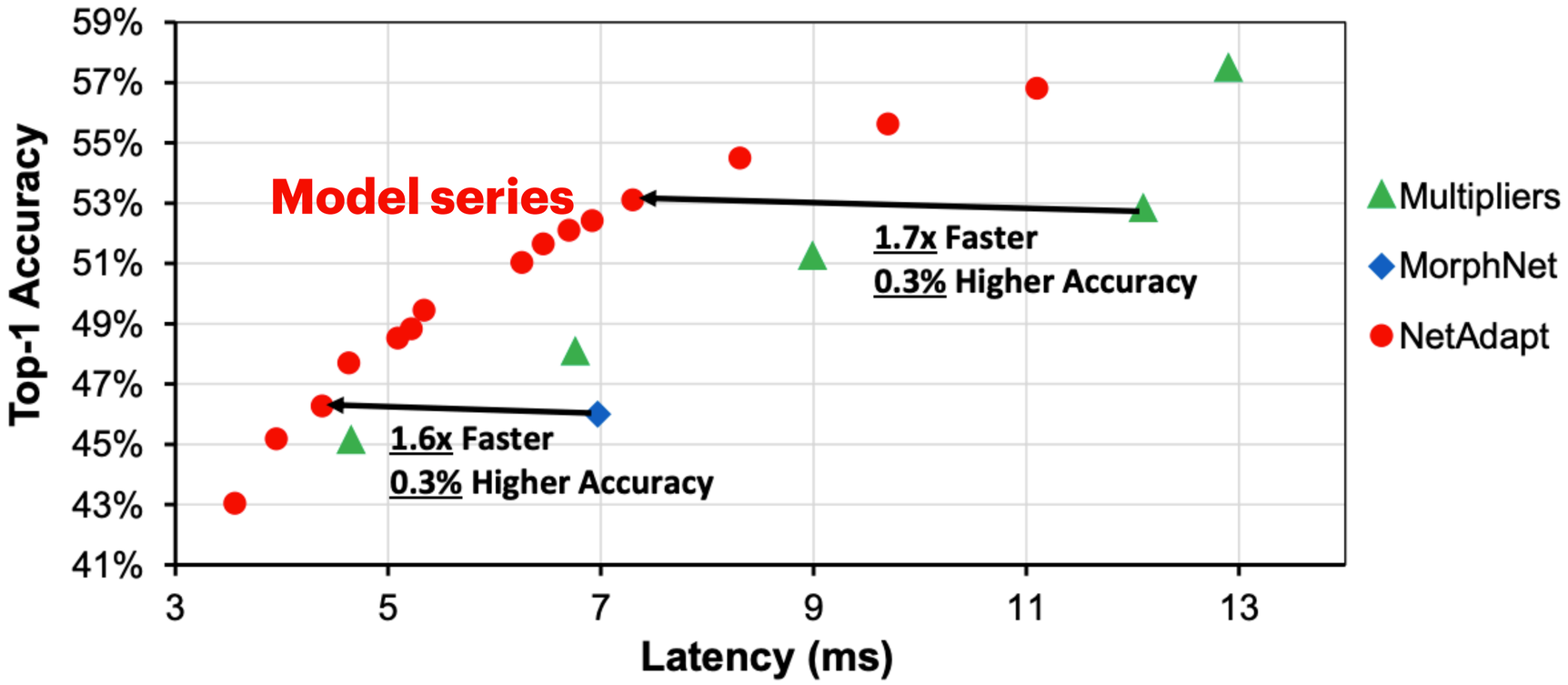


- Empirical lookup table
  - Efficiency
  - Hardware-awareness
  - Scalability
- Greedy and iterative

NetAdapt

Platform-Aware Neural Network Adaptation for Mobile Applications

- The iterative nature allows NetAdapt to obtain a serial of models with different costs
  - # models = # iterations



After pruning, the model accuracy may decrease, especially for larger pruning rratio

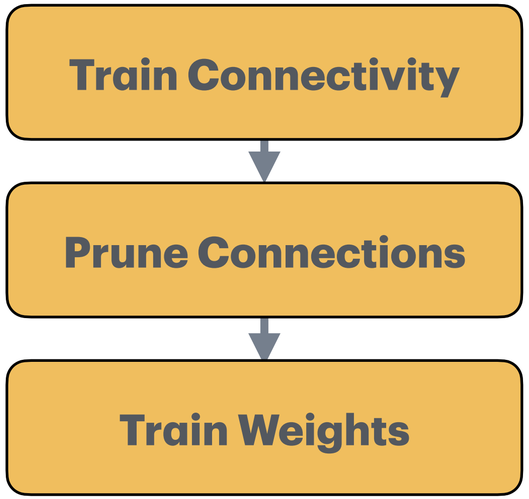
How should we improve performance of sparse models?

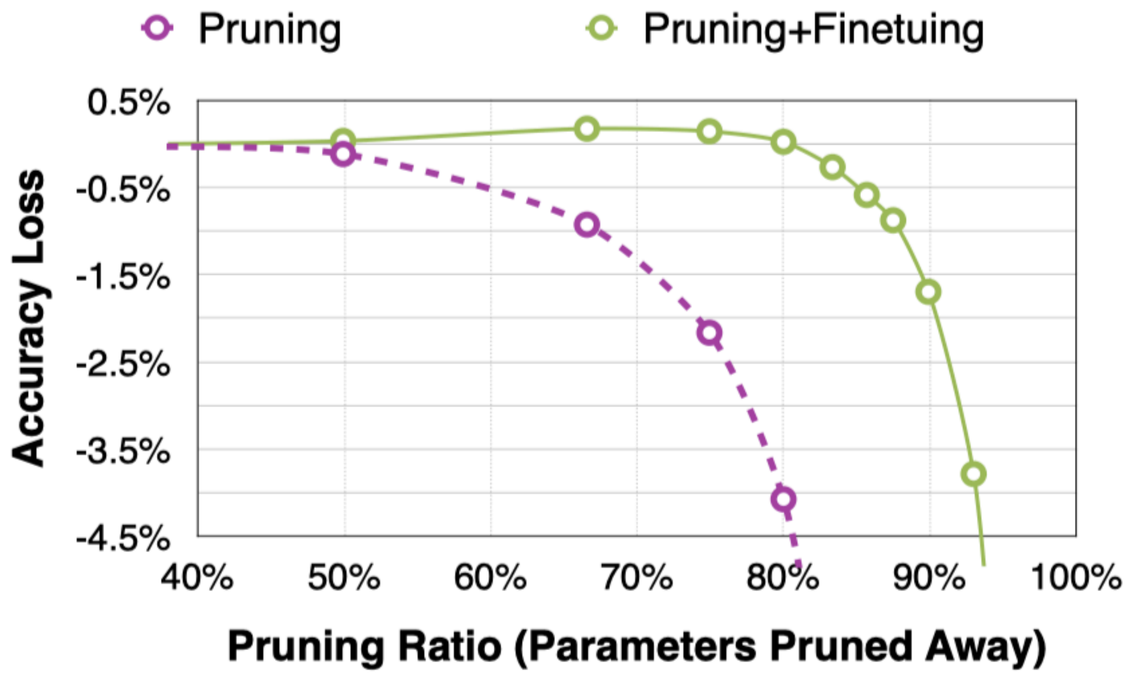
Fine-tuning / Training

How to fine-tune the pruned model to recover accuracy?

Fine-tuning Pruned Neural Networks

- Fine-tuning the pruned neural networks will help recover the accuracy and push the pruning ratio higher
  - **Learning rate for fune-tuning is usually 1/100 or 1/10 of the original learning rate**



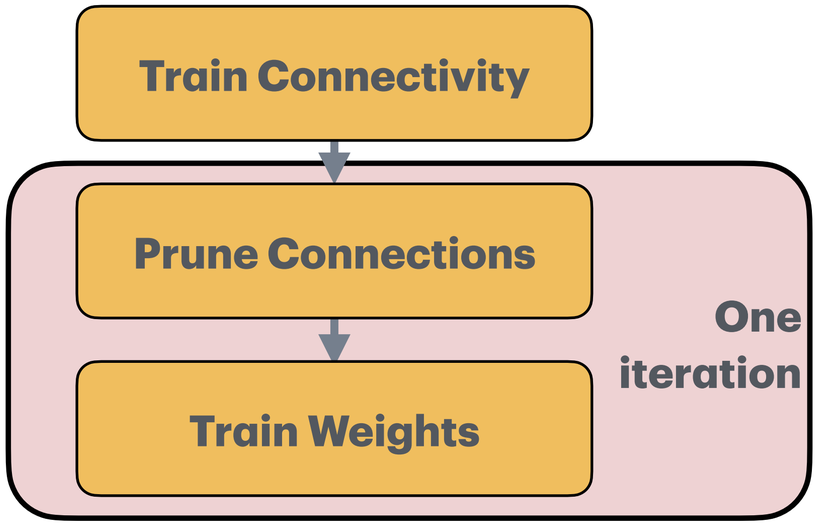


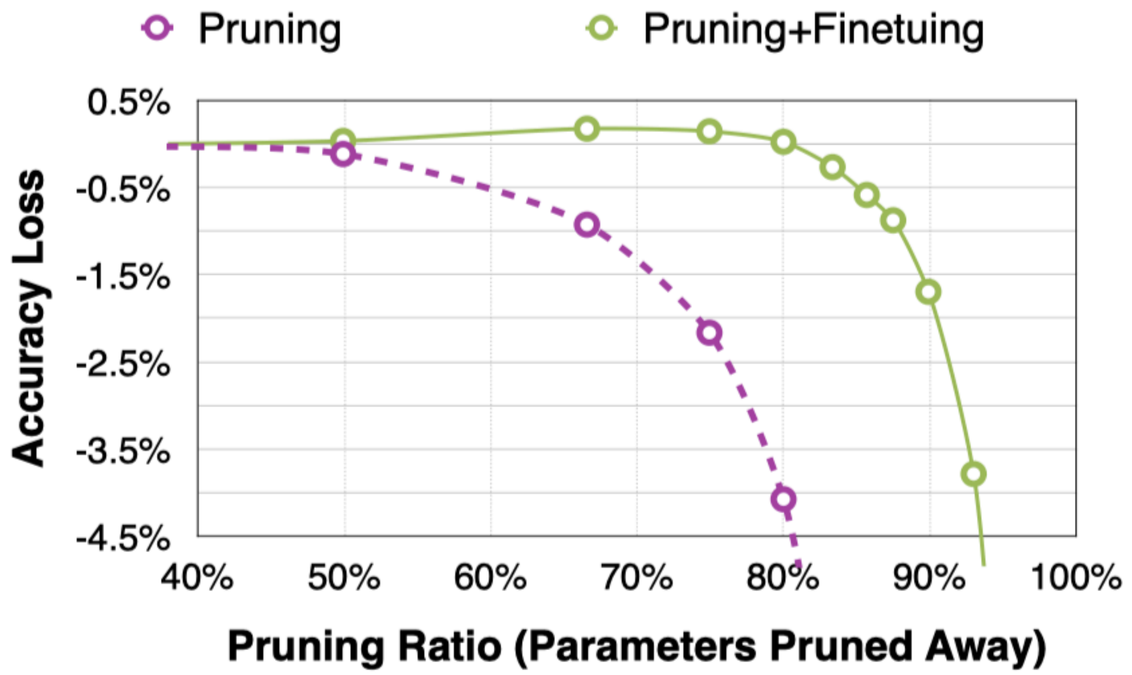
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider **pruning followed by a fune-tuning** is one iteration
- Iterative pruning **gradually increase the target sparsity** in each iteration



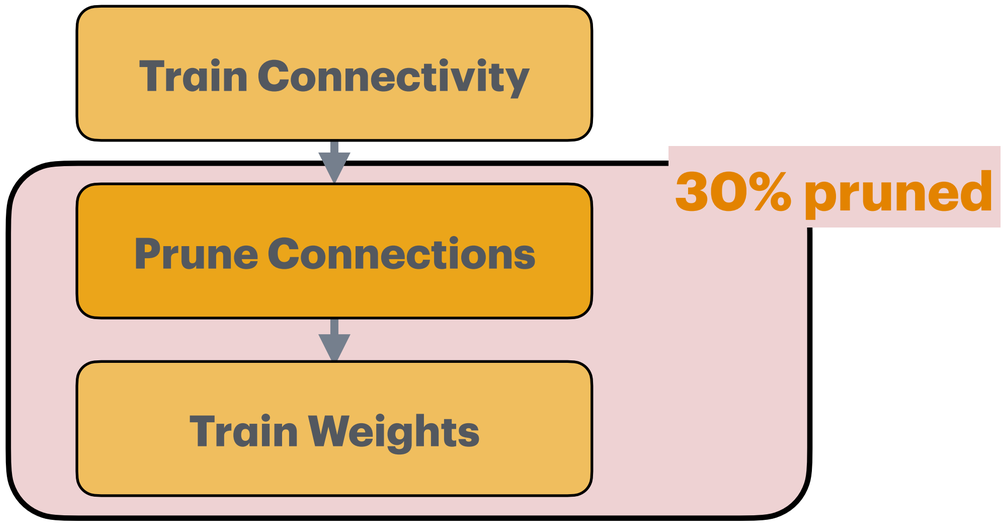


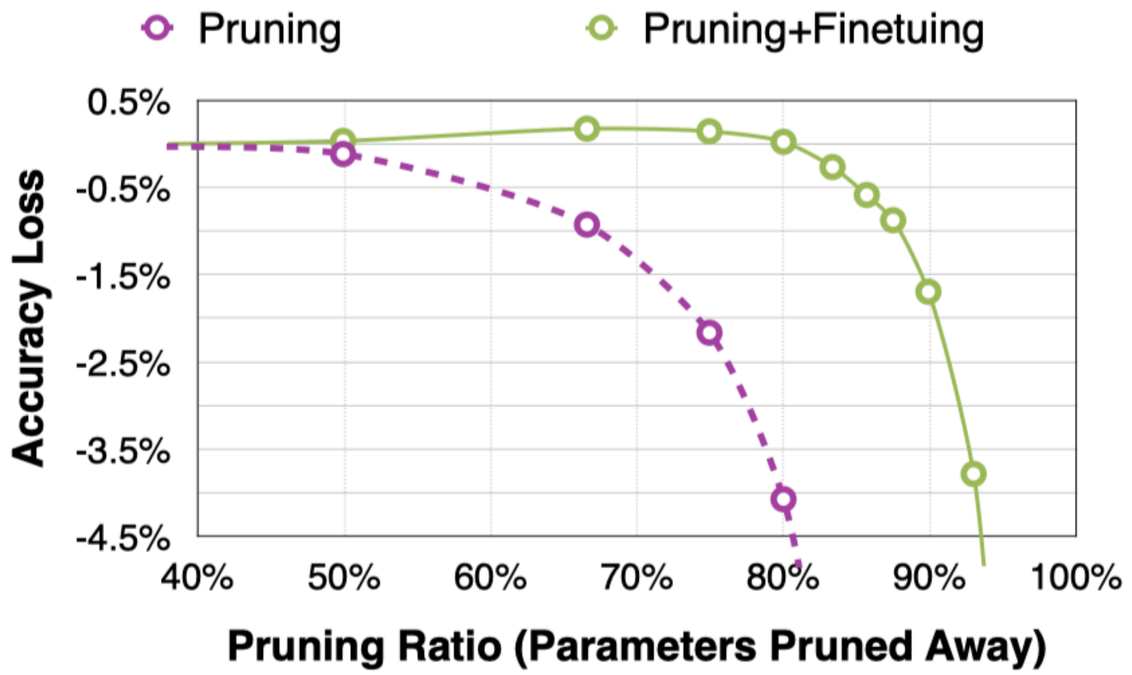
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider **pruning followed by a fune-tuning** is one iteration
- Iterative pruning **gradually increase the target sparsity** in each iteration



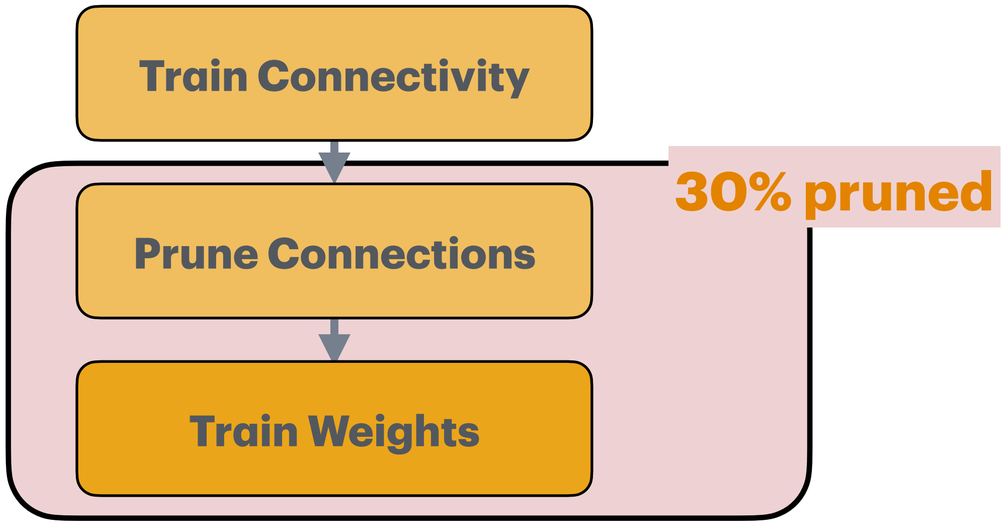


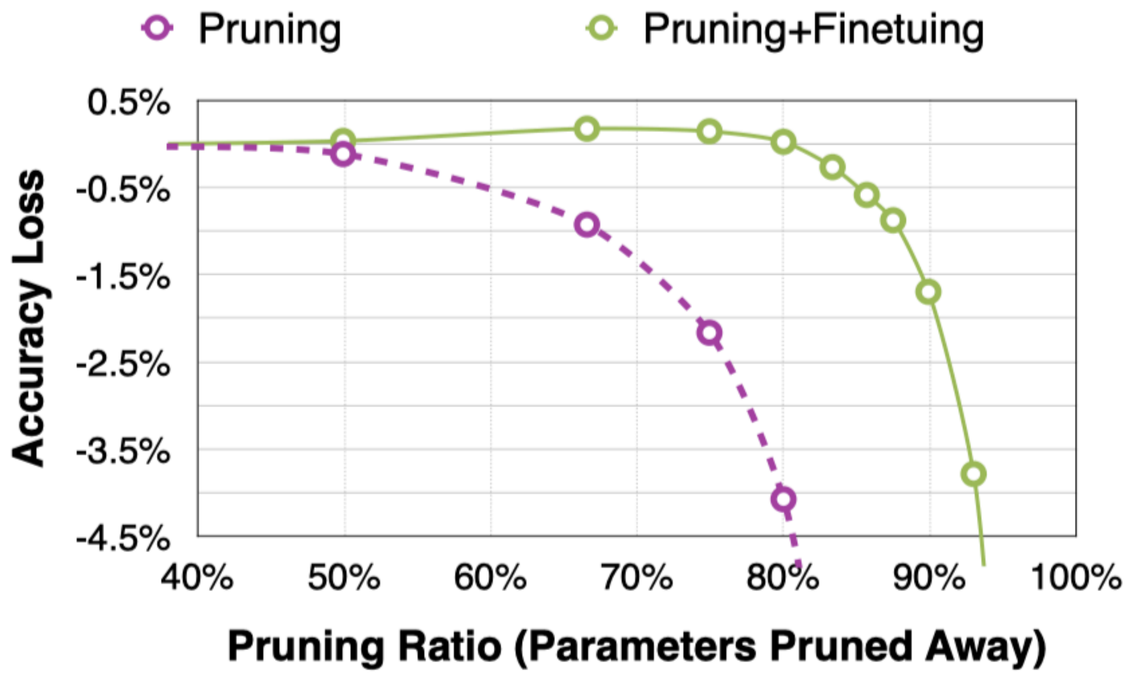
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider **pruning followed by a fune-tuning** is one iteration
- Iterative pruning **gradually increase the target sparsity** in each iteration



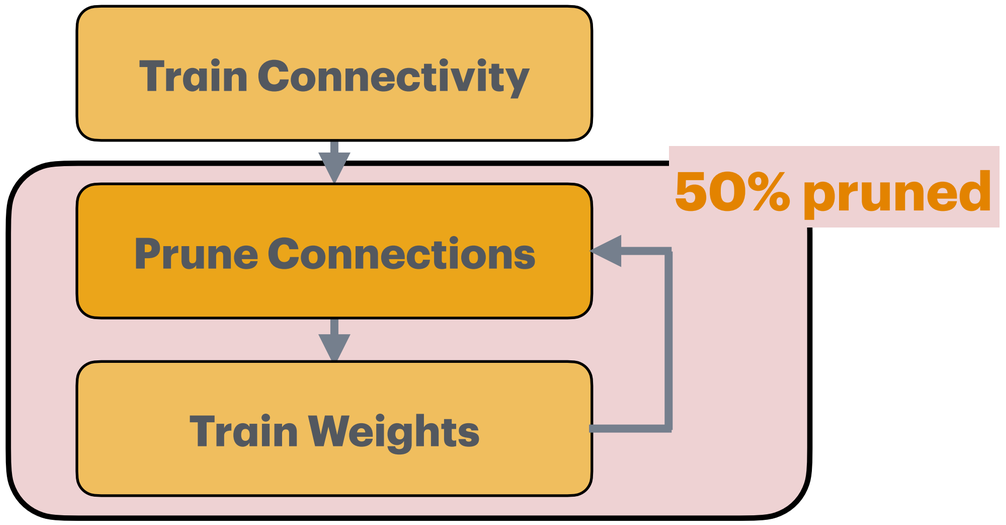


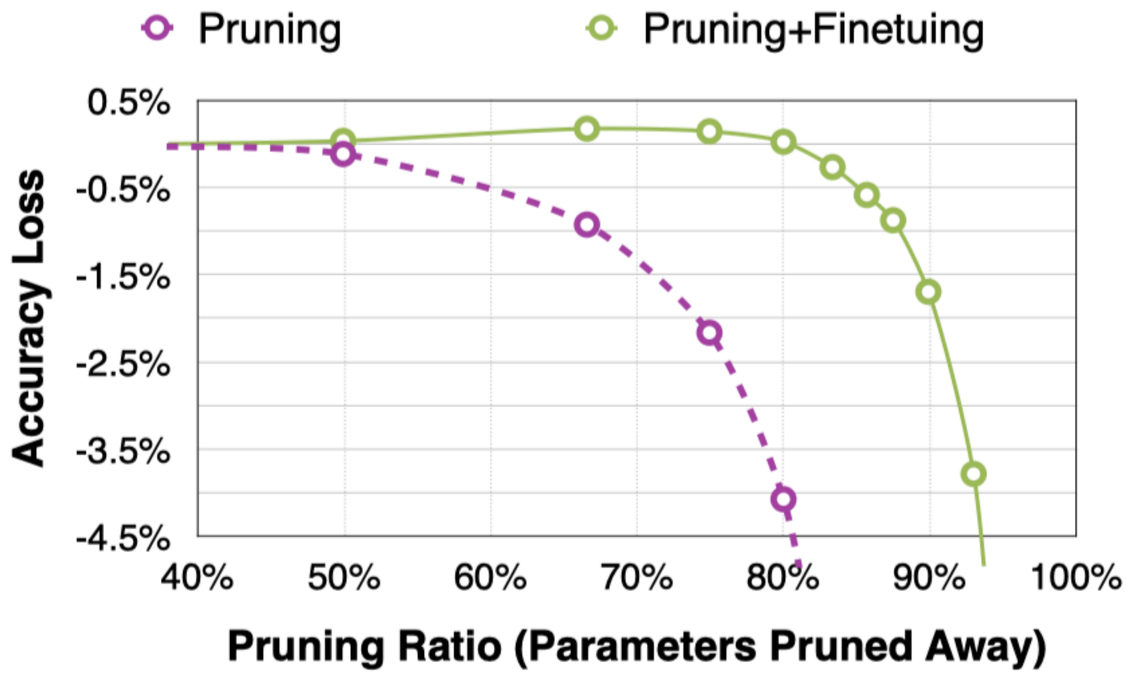
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider **pruning followed by a fune-tuning** is one iteration
- Iterative pruning **gradually increase the target sparsity** in each iteration



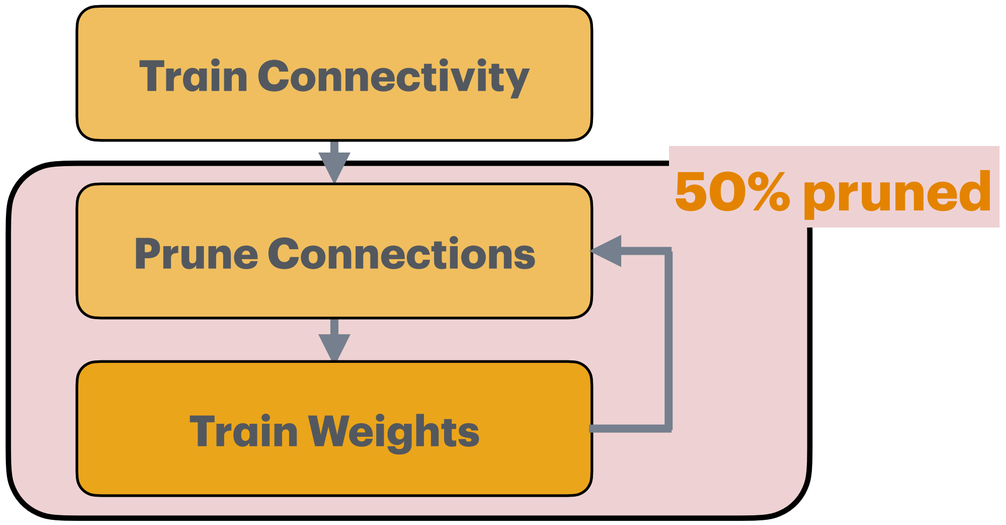


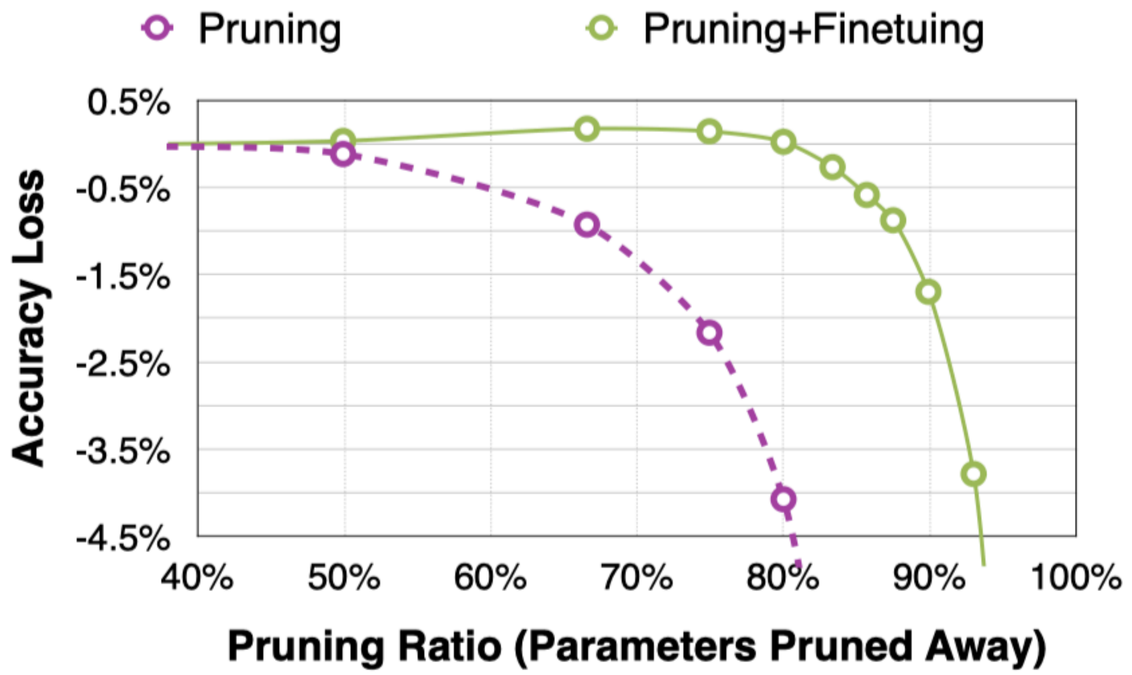
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider **pruning followed by a fune-tuning** is one iteration
- Iterative pruning **gradually increase the target sparsity** in each iteration



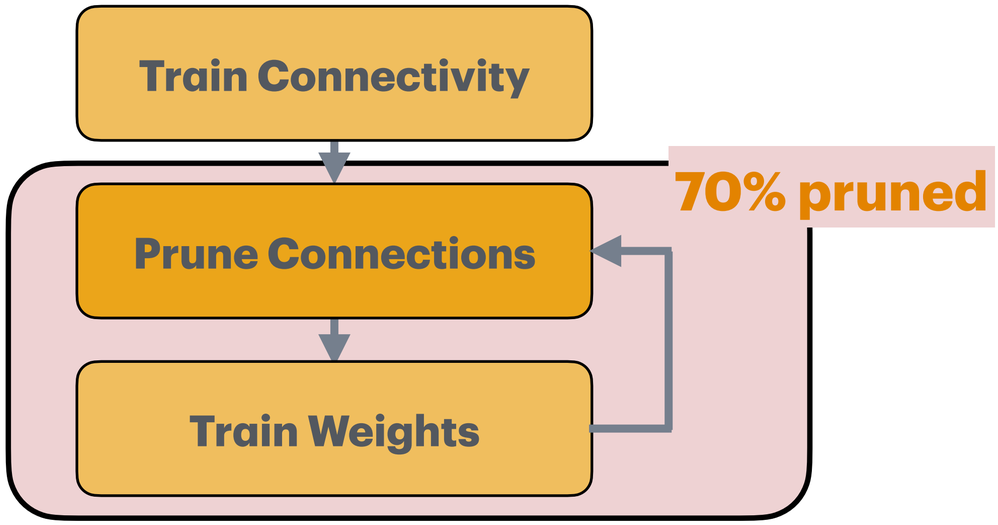


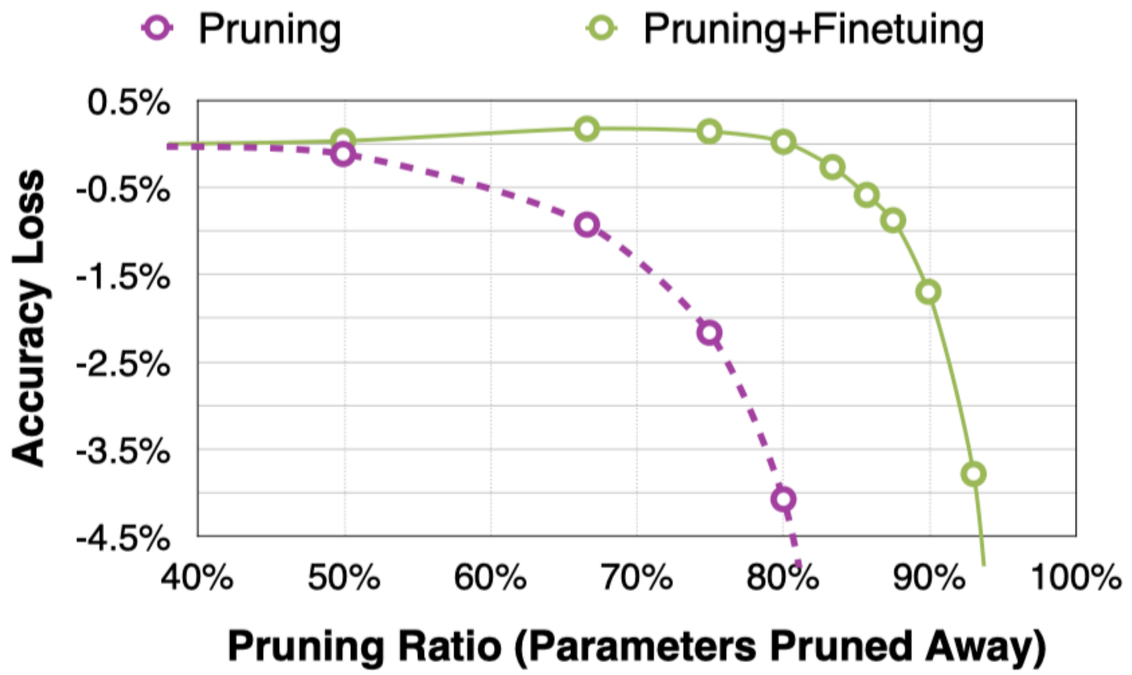
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider **pruning followed by a fune-tuning** is one iteration
- Iterative pruning **gradually increase the target sparsity** in each iteration



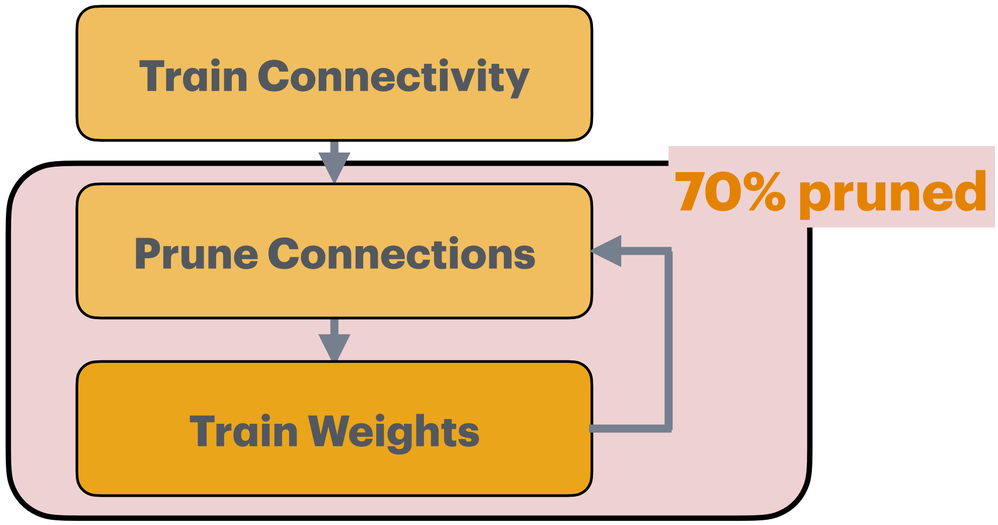


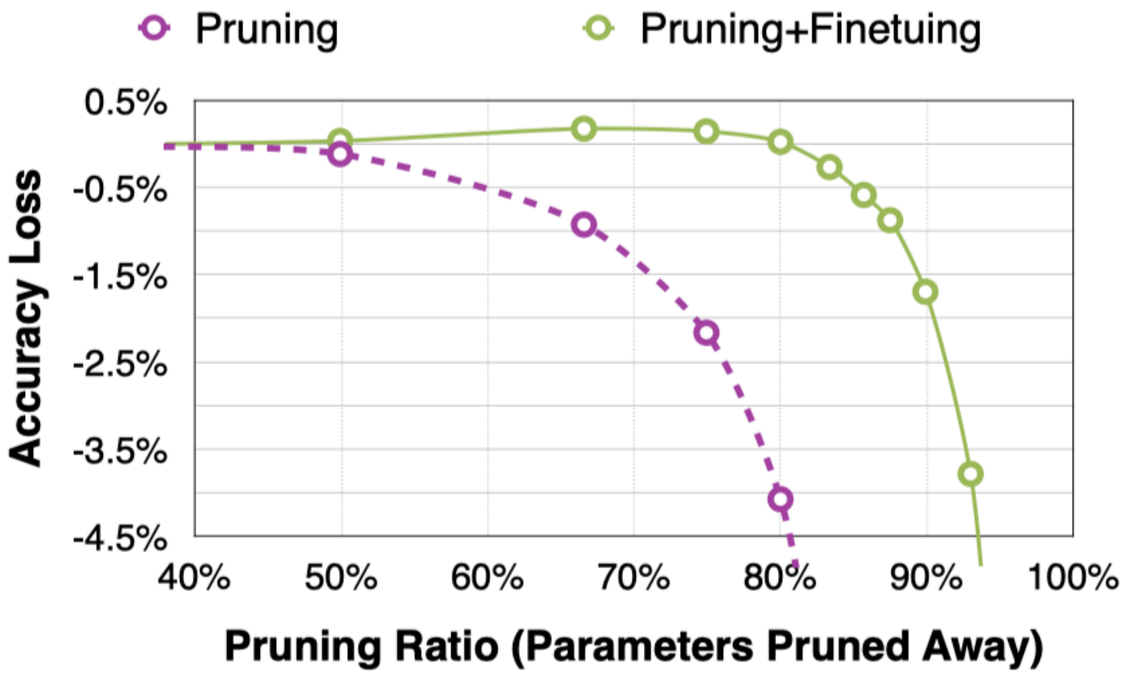
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider **pruning followed by a fune-tuning** is one iteration
- Iterative pruning **gradually increase the target sparsity** in each iteration



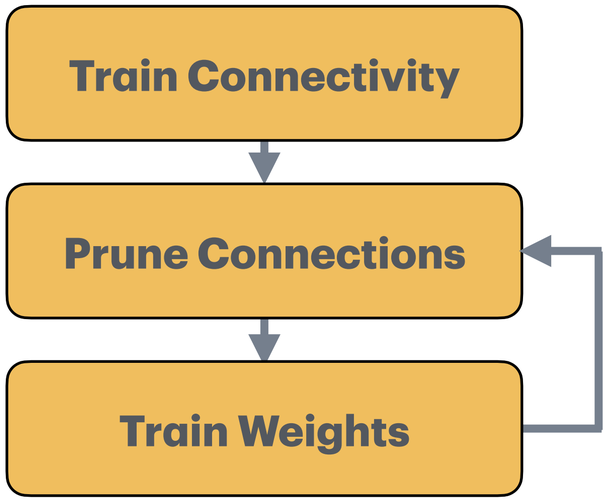


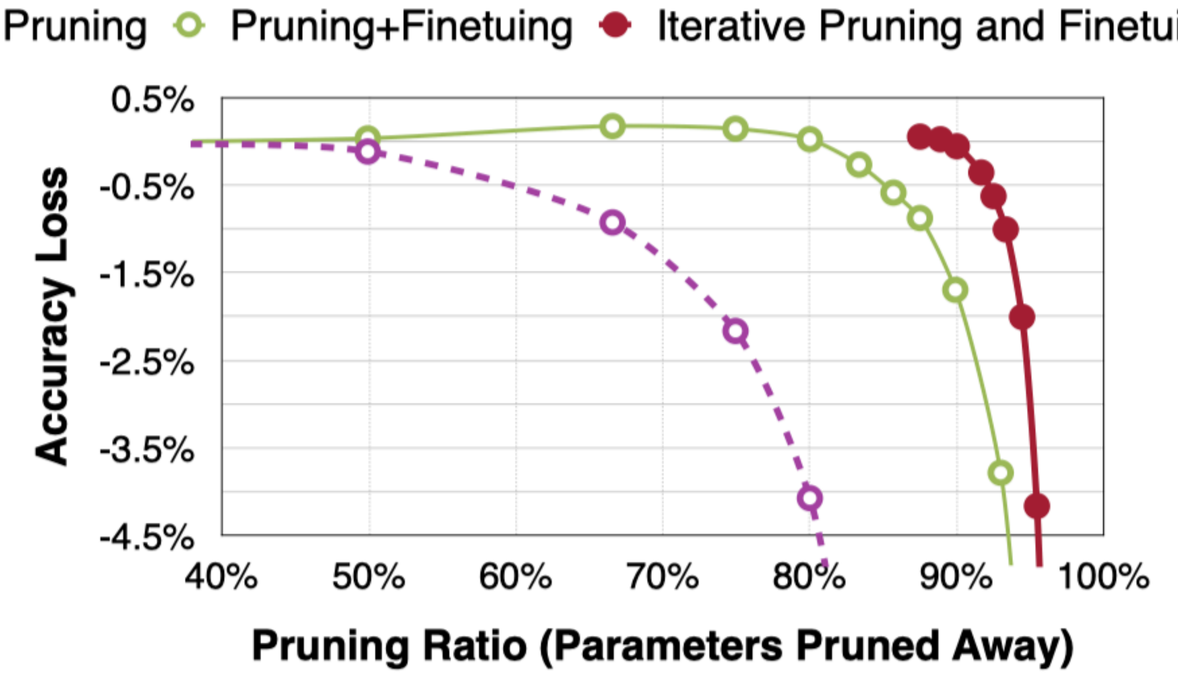
Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Iterative Pruning

- Consider pruning followed by a fune-tuning is one iteration
- Iterative pruning gradually increase the target sparsity in each iteration
  - Boost pruning ratio from 5X to 9X on AlexNet compared to single-step aggressive pruning





Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Figure source

Regularization

Prevent overfitting & Encourage simplicity

- When training neural networks or fine-tuning quantized neural networks, regularization is added to the **loss** term to (1) **Penalize non-zero parameters** (2) **Encourage smaller parameters**

- The most common regularization for improving performance of pruning is L1/L2 regularization.

  - L1 Regularization: $L' = L(x; W) + \lambda \mid W \mid$ *(λ balances data loss and regularization loss)*

  - L2 Regularization" $L' = L(x; W) + \lambda \parallel W \parallel^2$

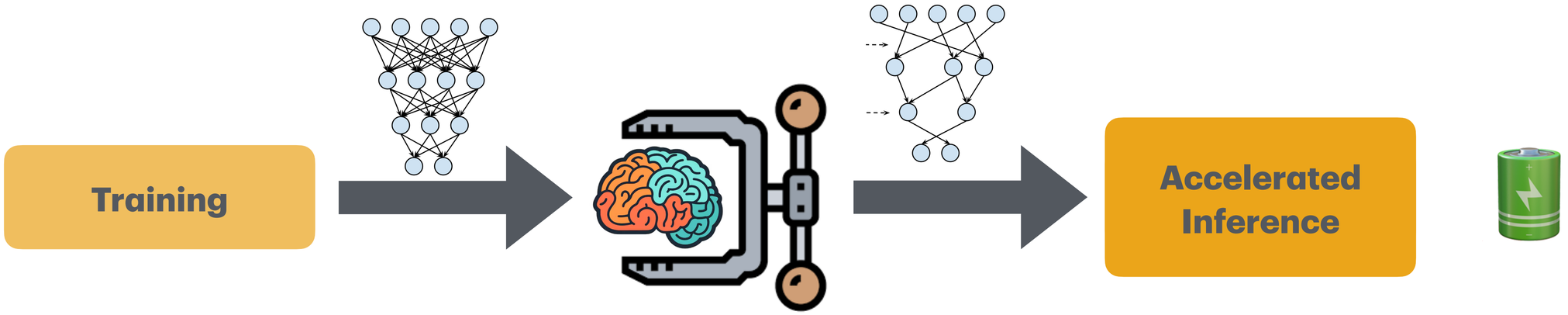
- Examples

  - Magnitude-based fine-grained pruning applies L2 regularization on weights

  - Network sliming applies smooth-L1 regularization on channel scaling factors

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.
Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE international conference on computer vision (pp. 2736-2744).

System & Hardware Support for Sparsity



Model compression
(pruning, sparsity, quantization, etc.)

Weight+Activation Sparsity For FC Layers

EIE: Efficient Inference Engine on Compressed Deep Neural Network

Song Han* Xingyu Liu* Huizi Mao* Jing Pu* Ardavan Pedram*
Mark A. Horowitz* William J. Dally*†
*Stanford University, †NVIDIA
{songhan,xyl,huizi,jingpu,perdavan,horowitz,dally}@stanford.edu

***Abstract*—State-of-the-art deep neural networks (DNNs) have hundreds of millions of connections and are both computationally and memory intensive, making them difficult to deploy on embedded systems with limited hardware resources and power budgets. While custom hardware helps the computation, fetching weights from DRAM is two orders of magnitude more expensive than ALU operations, and dominates the required power.**

Previously proposed 'Deep Compression' makes it possible to fit large DNNs (AlexNet and VGGNet) fully in on-chip SRAM. This compression is achieved by pruning the redundant connections and having multiple connections share the same weight. We propose an energy efficient inference engine (EIE) that performs inference on this compressed network model and accelerates the resulting sparse matrix-vector multiplication with weight sharing. Going from DRAM to SRAM gives EIE 120× energy saving; Exploiting sparsity saves 10×; Weight sharing gives 8×; Skipping zero activations from ReLU saves another 3×. Evaluated on nine DNN benchmarks, EIE is 189× and 13× faster when compared to CPU and GPU implementations of the same DNN without compression. EIE has a processing power of 102 GOPS/s working directly on a compressed network, corresponding to 3 TOPS/s on an uncompressed network, and processes FC layers of AlexNet at 1.88×10^4 frames/sec with a power dissipation of only 600mW. It is 24,000× and 3,400× more energy efficient than a CPU and GPU respectively. Compared with DaDianNao, EIE has 2.9×, 19× and 3× better throughput, energy efficiency and area efficiency.

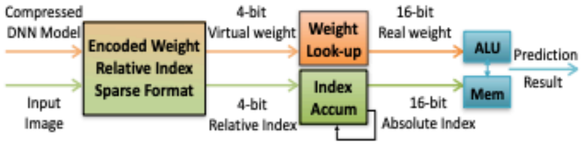
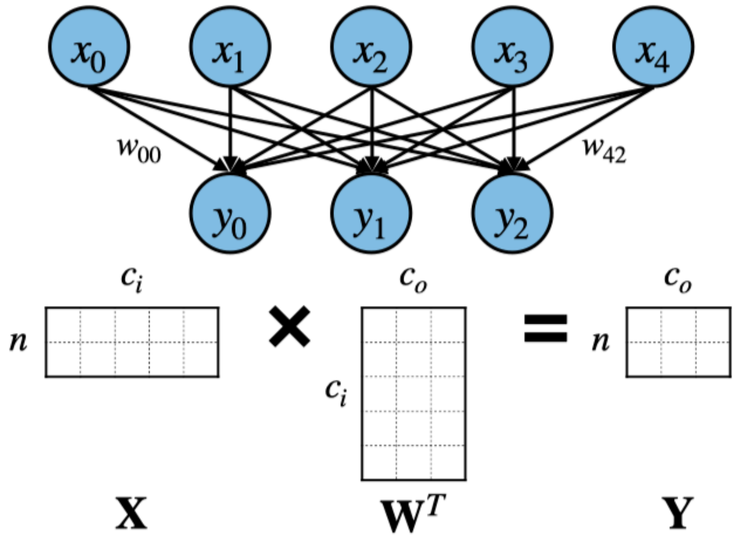


Figure 1. Efficient inference engine that works on the compressed deep neural network model for machine learning applications.

word, or speech sample. For embedded mobile applications, these resource demands become prohibitive. Table I shows the energy cost of basic arithmetic and memory operations in a 45nm CMOS process [9]. It shows that the total energy is dominated by the required memory access if there is no data reuse. The energy cost per fetch ranges from 5pJ for 32b coefficients in on-chip SRAM to 640pJ for 32b coefficients in off-chip LPDDR2 DRAM. Large networks do not fit in on-chip storage and hence require the more costly DRAM accesses. Running a 1G connection neural network, for example, at 20Hz would require $(20Hz)(1G)(640pJ) = 12.8W$ just for DRAM accesses, which is well beyond the power envelope of a typical mobile device.

Previous work has used specialized hardware to accelerate DNNs [10]–[12]. However, these efforts focus on acceler-



Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network

Song Han[1,3], Xingyu Liu[4], Huizi Mao[3], Jing Pu[5], Ardavan Pedram[2,6], Mark A. Horowitz[2], William J. Dally[2,3],
[1]MIT [2]Stanford [3] NVIDIA [4] CMU [5] Google [6] Samsung

***Abstract*—EIE proposed to accelerate pruned and compressed neural networks, exploiting weight sparsity, activation sparsity, and 4-bit weight-sharing in neural network accelerators. Since published in ISCA'16, it opened a new design space to accelerate pruned and sparse neural networks and spawned many algorithm-hardware co-designs for model compression and acceleration, both in academia and commercial AI chips. In retrospect, we review the background of this project, summarize the pros and cons, and discuss new opportunities where pruning, sparsity, and low-precision can accelerate emerging deep learning workloads.**

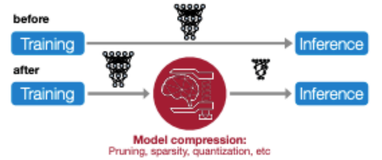


Fig. 1. EIE opened a new opportunity to build hardware accelerator for sparse and compressed neural networks.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.
Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2023). Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network. arXiv preprint arXiv:2306.09552.

EIE: Efficient Inference Engine

Top-5 most sicted papers in 50 years of ISCA

Rank	*Citations*	*Year*	*Title (★ means it won the ISCA Influential Paper Award)*	First Author + *HOF Authors*	*Type*	*Topic*
1	5351	1995	The SPLASH-2 programs: Characterization and methodological considerations	Stephen Woo, *Anoop Gupta*	Tool	Benchmark
2	4214	2017	In-datacenter performance analysis of a Tensor Processing Unit	*Norm Jouppi, David Patterson*	Arch	Machine Learning
3	3834	2000	★ Wattch: A framework for architectural-level power analysis and optimizations	*David Brooks, Margaret Martonosi*	Tool	Power
4	3386	1993	★ Transactional memory: Architectural support for lock-free data structures	Maurice Herlihy	Micro	Parallelism
5	2690	2016	EIE: Efficient inference engine on compressed deep neural network	Song Han, *Bill Dally, Mark Horowitz*	Arch	Machine Learning

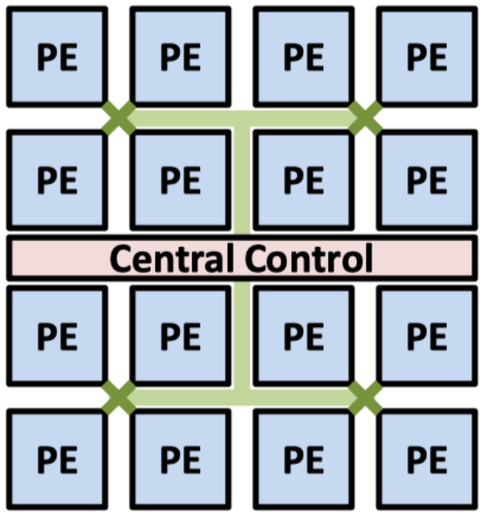
Source

EIE: Parallelization on Sparsity

EIE is a scalable array of processing elements (PEs)

- Every PE stores a partition of network in SRAM and performs the computations associated with that part.

- **Matrix** W and **vectors** a and b are interleaved over 4 PEs. Elements of the same color are stored in the same PE.



$$\vec{a} \begin{pmatrix} 0 & 0 & \boldsymbol{a_2} & 0 & a_4 & a_5 & 0 & a_7 \end{pmatrix}$$

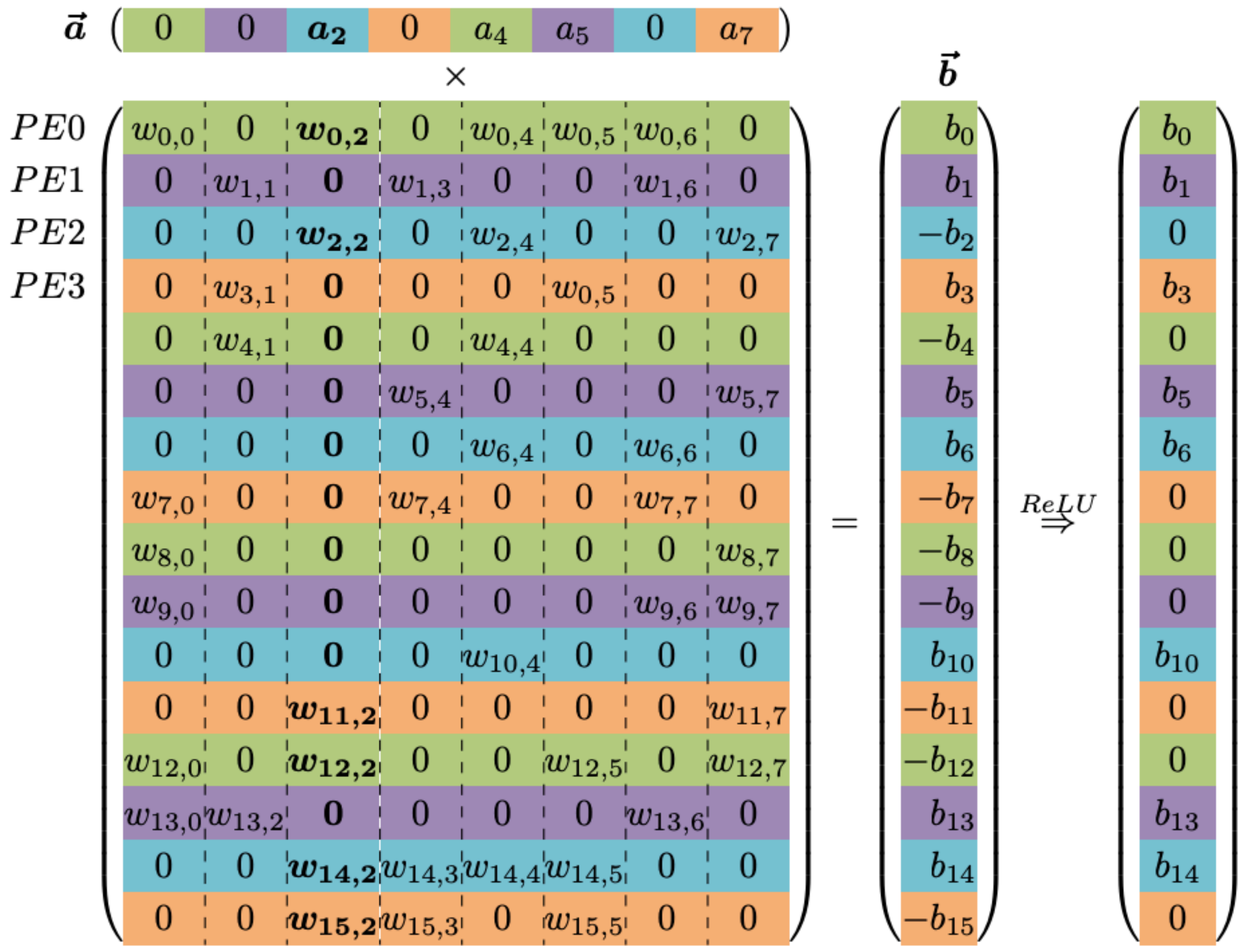
$$\times$$

$$\begin{array}{l} PE0 \\ PE1 \\ PE2 \\ PE3 \end{array} \begin{pmatrix} w_{0,0} & 0 & \boldsymbol{w_{0,2}} & 0 & w_{0,4} & w_{0,5} & w_{0,6} & 0 \\ 0 & w_{1,1} & \boldsymbol{0} & w_{1,3} & 0 & 0 & w_{1,6} & 0 \\ 0 & 0 & \boldsymbol{w_{2,2}} & 0 & w_{2,4} & 0 & 0 & w_{2,7} \\ 0 & w_{3,1} & \boldsymbol{0} & 0 & 0 & w_{0,5} & 0 & 0 \\ 0 & w_{4,1} & \boldsymbol{0} & 0 & w_{4,4} & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{0} & w_{5,4} & 0 & 0 & 0 & w_{5,7} \\ 0 & 0 & \boldsymbol{0} & 0 & w_{6,4} & 0 & w_{6,6} & 0 \\ w_{7,0} & 0 & \boldsymbol{0} & w_{7,4} & 0 & 0 & w_{7,7} & 0 \\ w_{8,0} & 0 & \boldsymbol{0} & 0 & 0 & 0 & 0 & w_{8,7} \\ w_{9,0} & 0 & \boldsymbol{0} & 0 & 0 & 0 & w_{9,6} & w_{9,7} \\ 0 & 0 & \boldsymbol{0} & 0 & w_{10,4} & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{w_{11,2}} & 0 & 0 & 0 & 0 & w_{11,7} \\ w_{12,0} & 0 & \boldsymbol{w_{12,2}} & 0 & 0 & w_{12,5} & 0 & w_{12,7} \\ w_{13,0} & w_{13,2} & \boldsymbol{0} & 0 & 0 & 0 & w_{13,6} & 0 \\ 0 & 0 & \boldsymbol{w_{14,2}} & w_{14,3} & w_{14,4} & w_{14,5} & 0 & 0 \\ 0 & 0 & \boldsymbol{w_{15,2}} & w_{15,3} & 0 & w_{15,5} & 0 & 0 \end{pmatrix}$$

$$\vec{b} \quad = \quad \begin{pmatrix} b_0 \\ b_1 \\ -b_2 \\ b_3 \\ -b_4 \\ b_5 \\ b_6 \\ -b_7 \\ -b_8 \\ -b_9 \\ b_{10} \\ -b_{11} \\ -b_{12} \\ b_{13} \\ b_{14} \\ -b_{15} \end{pmatrix} \quad \overset{ReLU}{\Rightarrow} \quad \begin{pmatrix} b_0 \\ b_1 \\ 0 \\ b_3 \\ 0 \\ b_5 \\ b_6 \\ 0 \\ 0 \\ 0 \\ b_{10} \\ 0 \\ 0 \\ b_{13} \\ b_{14} \\ 0 \end{pmatrix}$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Parallelization on Sparsity



Logically

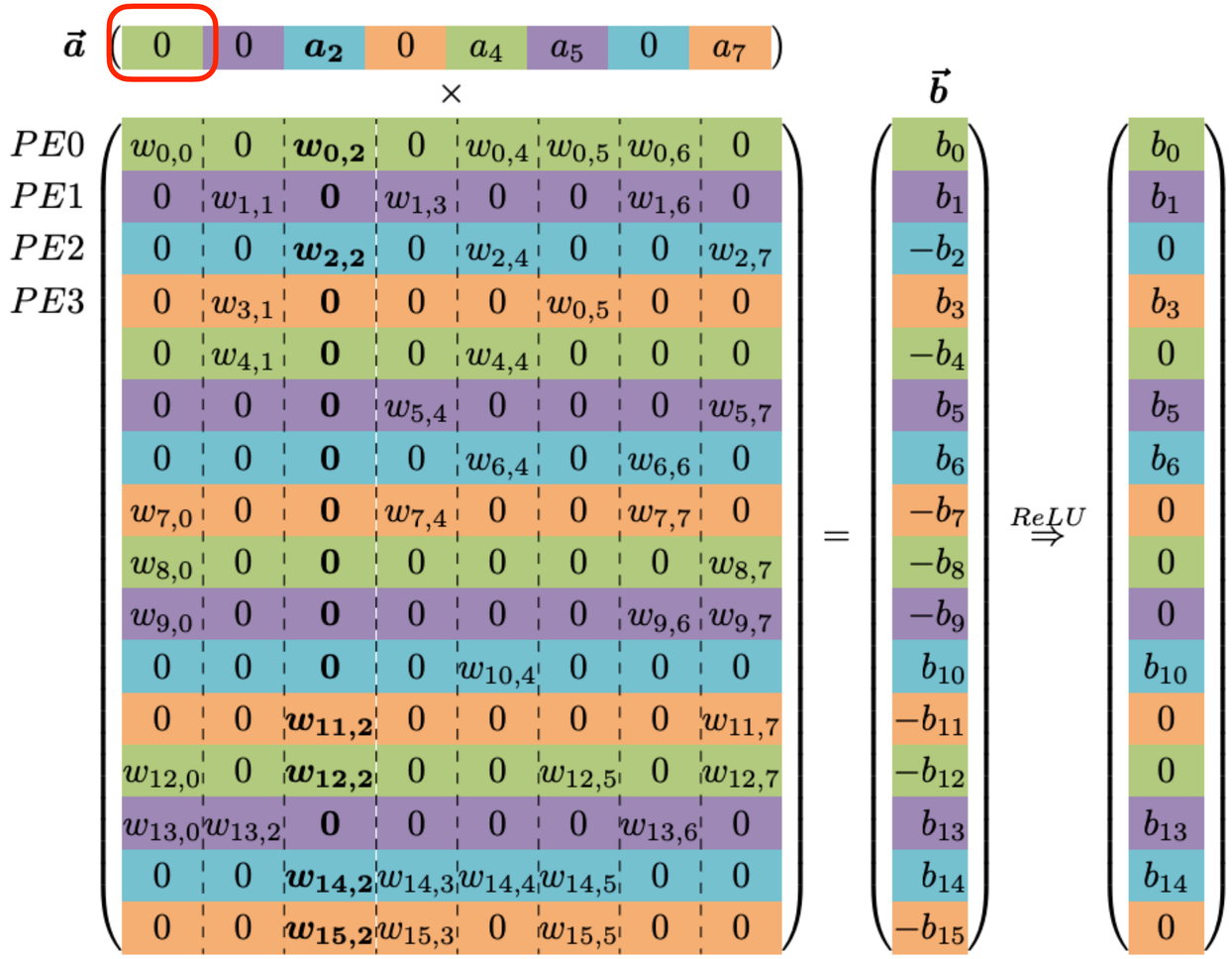
- Memory layout for the relative indexed, indirect weighted and interleaved CSC format, corresponding to PE0
  - CSC (Compressed Sparse Column): store sparse matrices by only keeping the **non-zero elements** and their positions

Virtual Weight	$W_{0,0}$	$W_{8,0}$	$W_{12,0}$	$W_{4,1}$	$W_{0,2}$	$W_{12,2}$	$W_{0,4}$	$W_{4,4}$	$W_{0,5}$	$W_{12,5}$	$W_{0,6}$	$W_{8,7}$	$W_{12,7}$
Relative Row Index	0	1	0	1	0	2	0	0	0	2	0	2	0
Column Pointer	0	3	4	6	6	8	10	11	13				

Physically

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



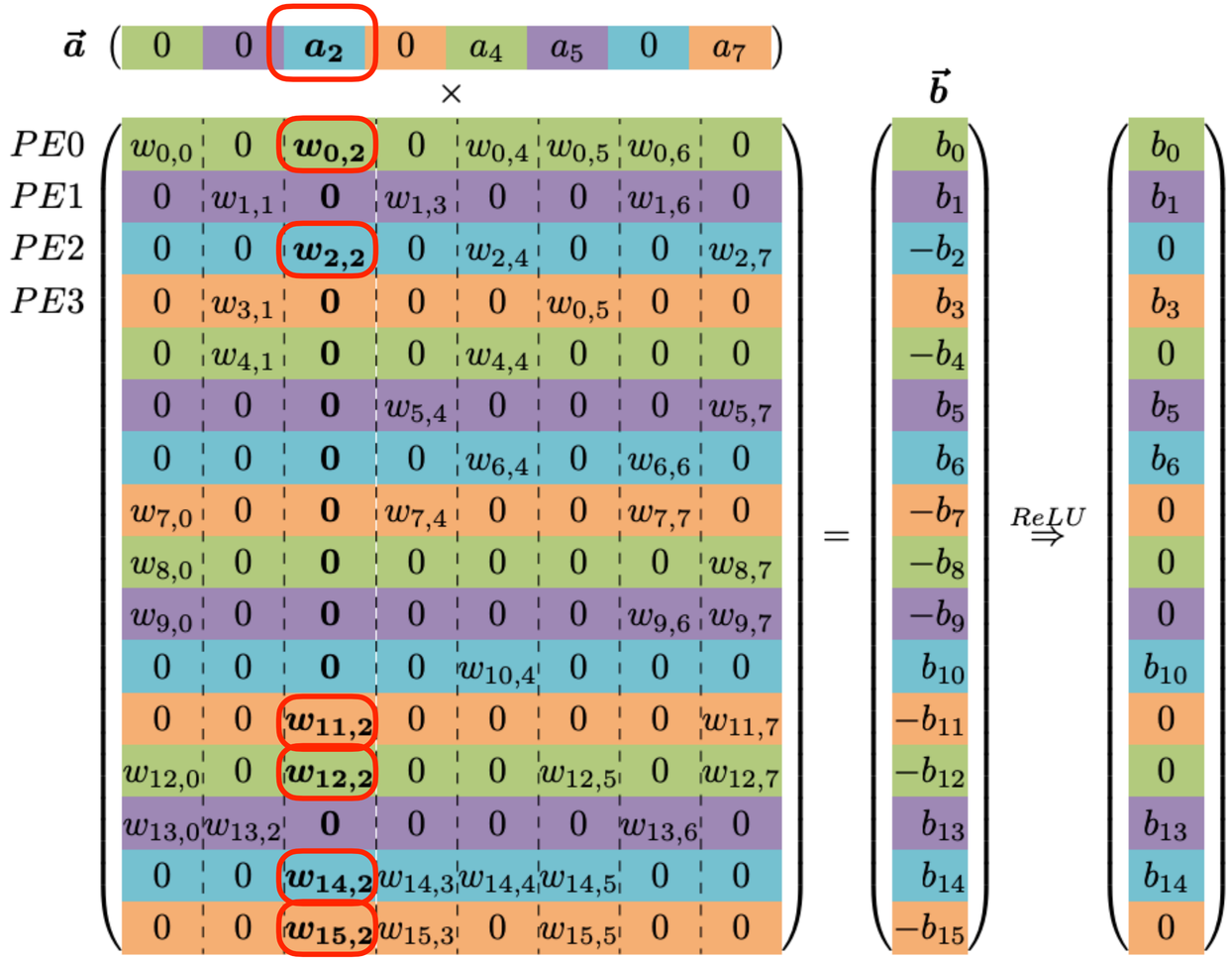
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



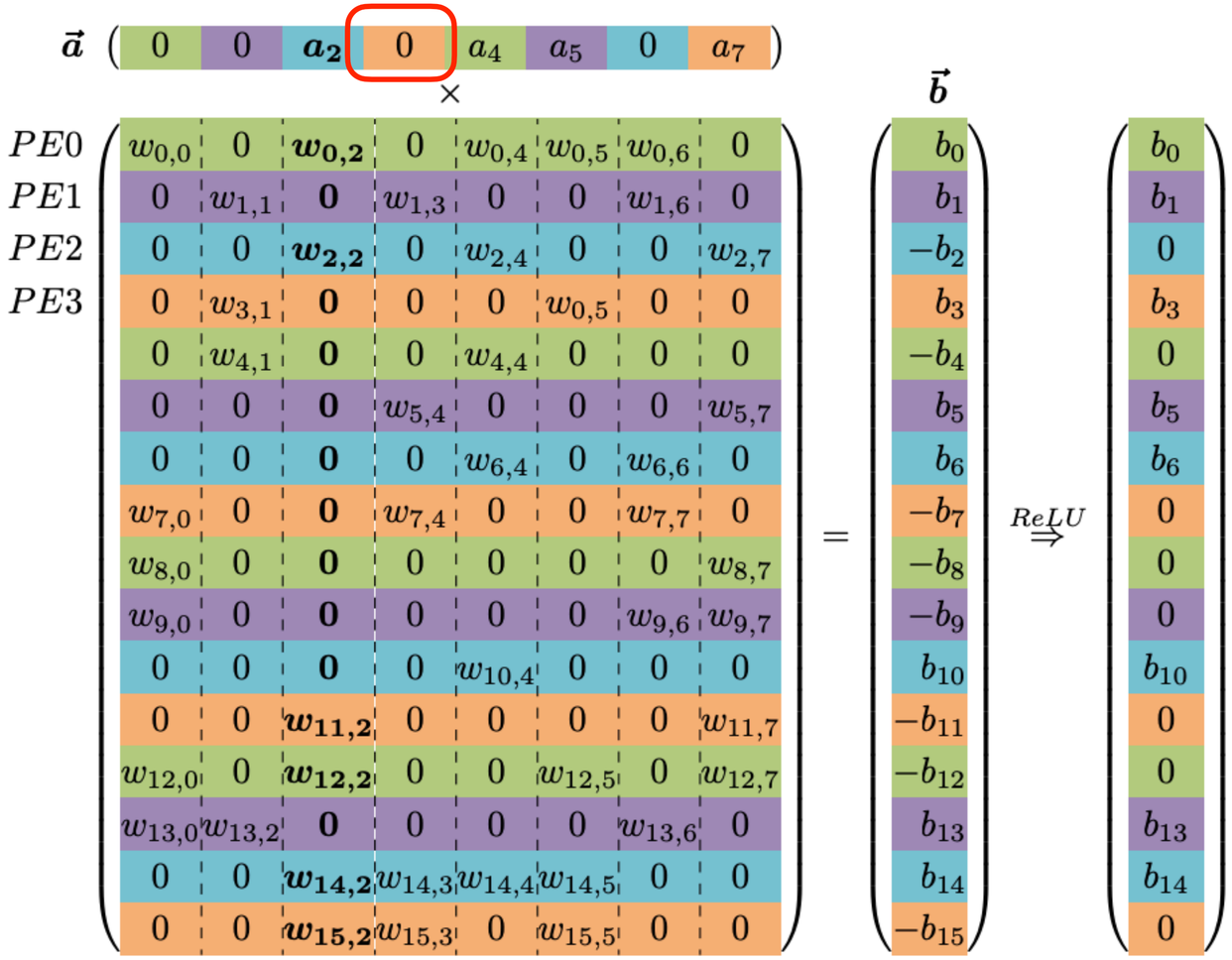
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



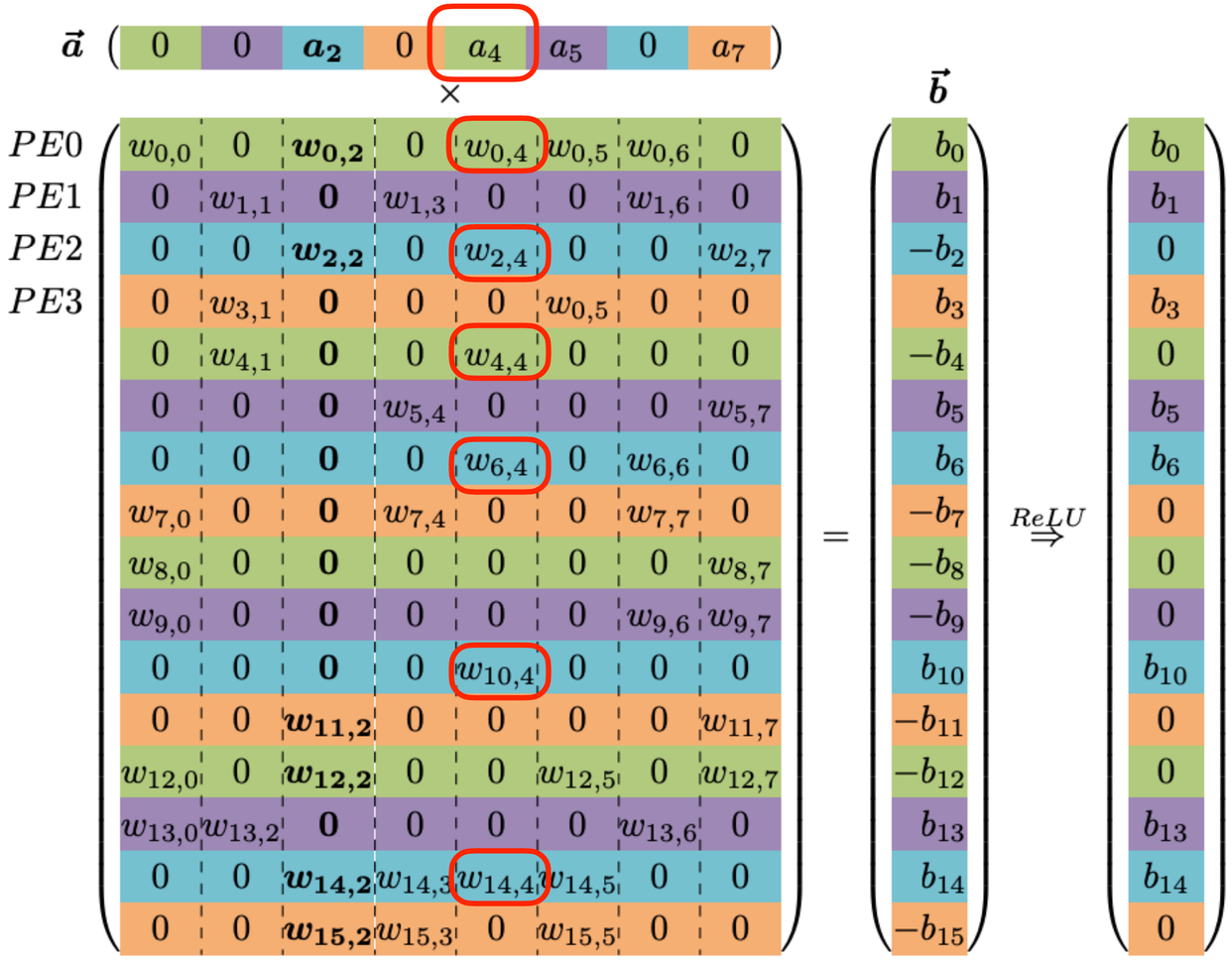
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



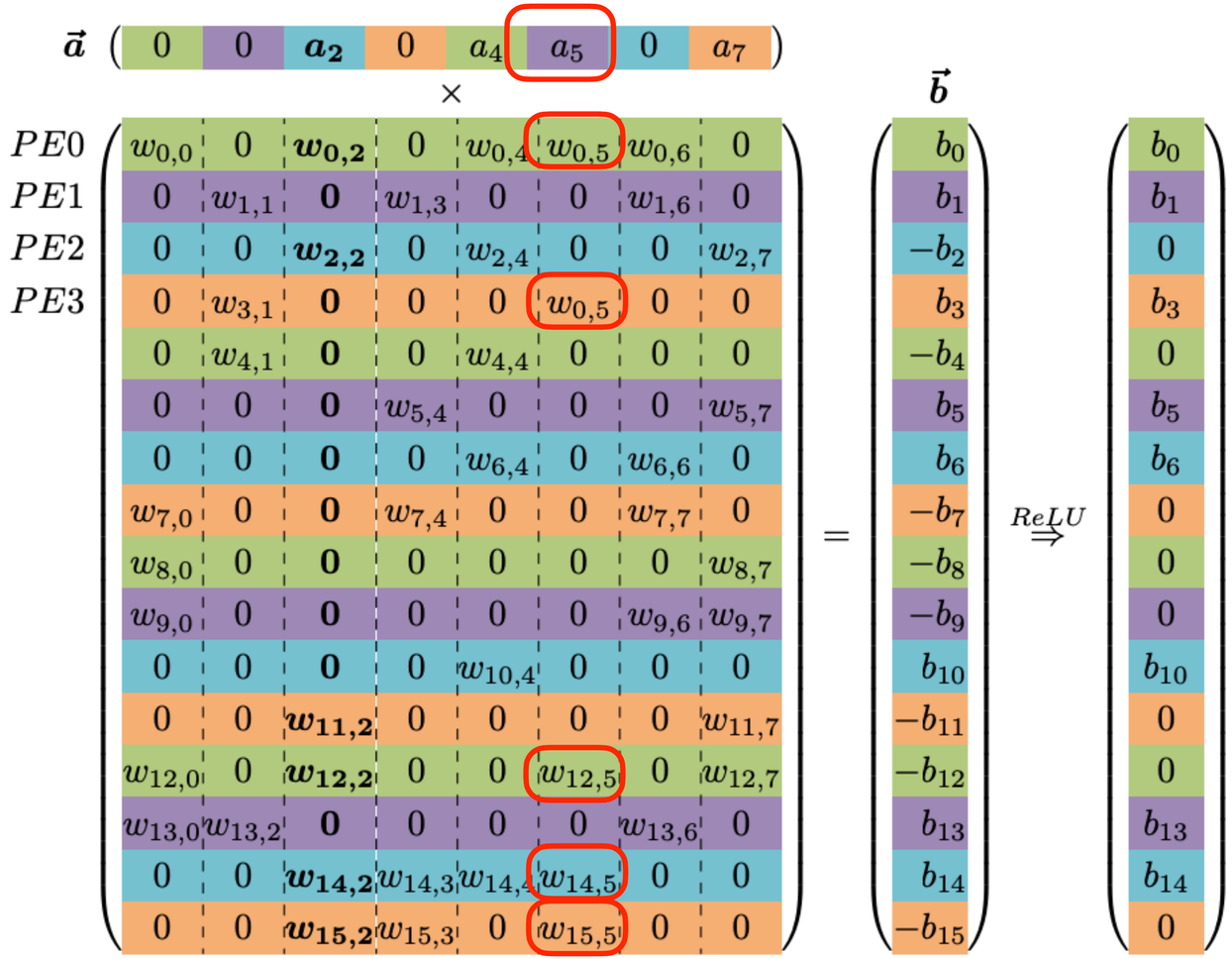
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



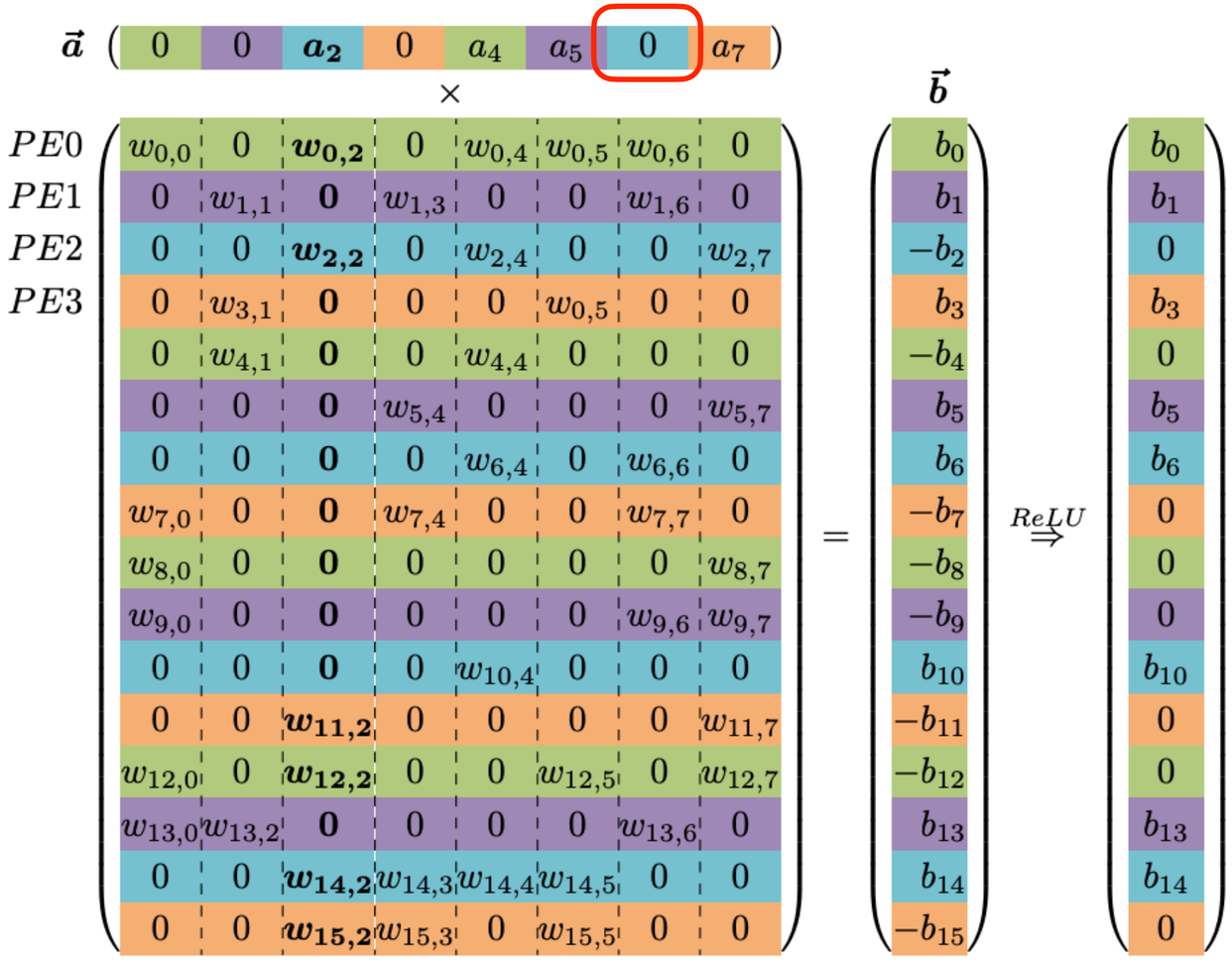
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



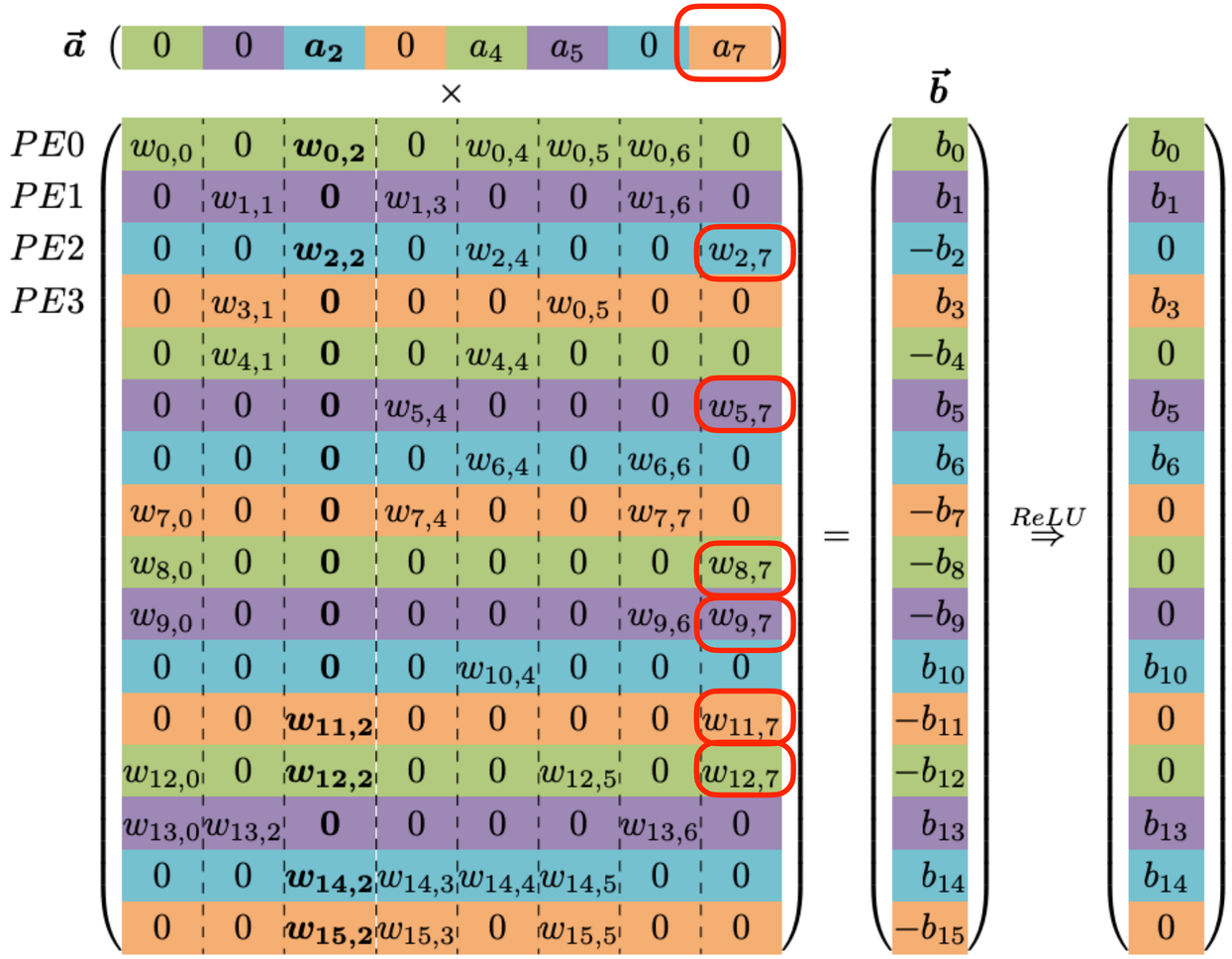
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



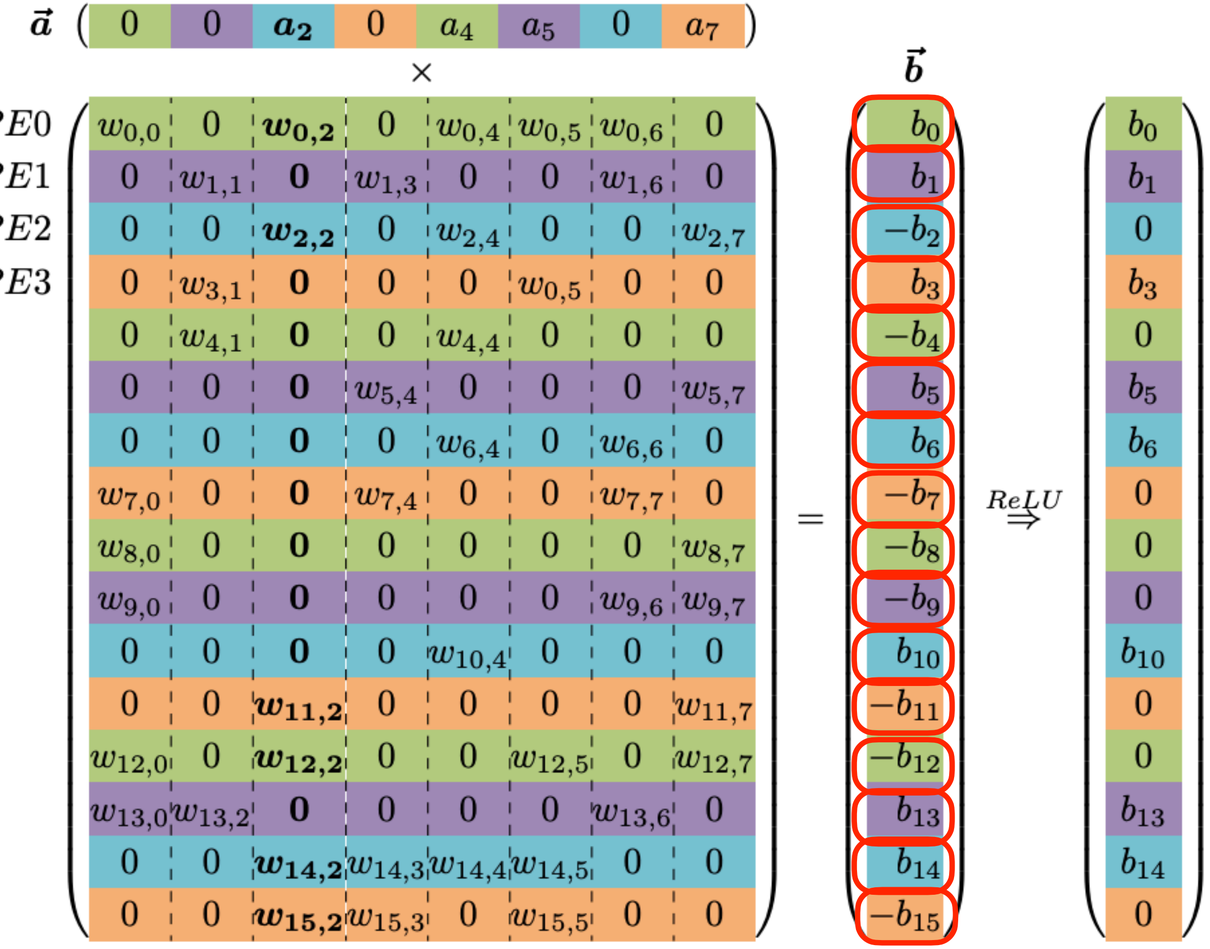
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Dataflow



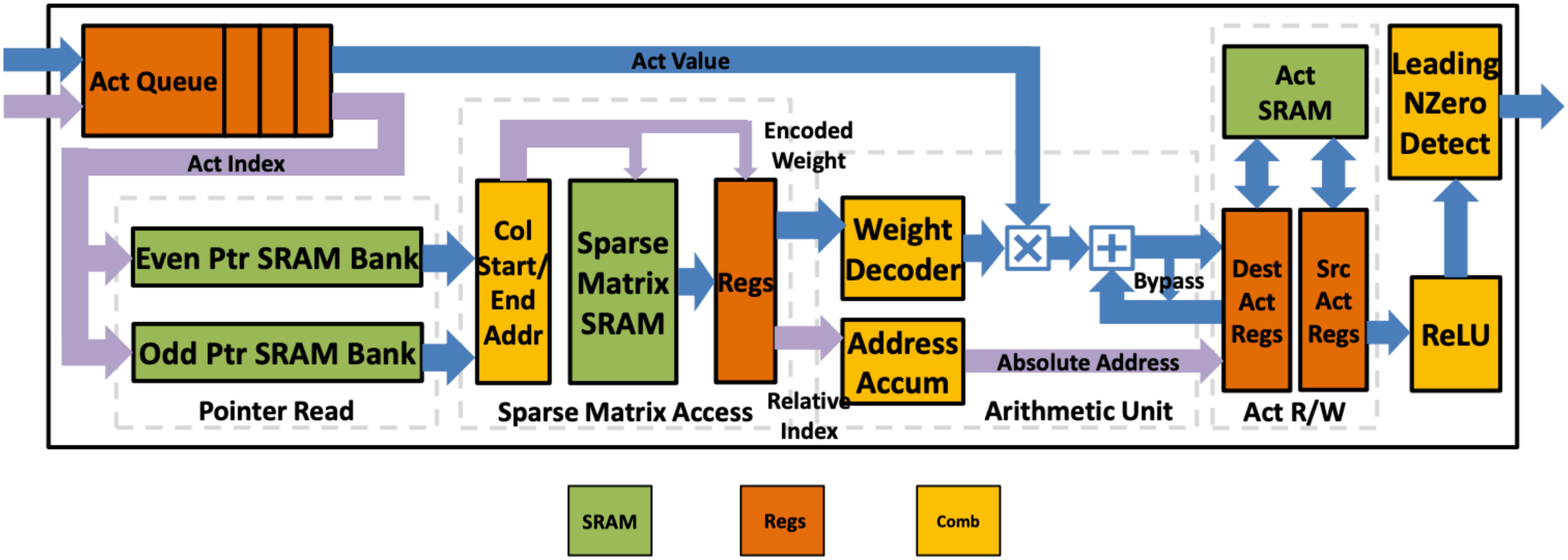
- Rule of thumb

$$0 \times A = 0$$

$$W \times 0 = 0$$

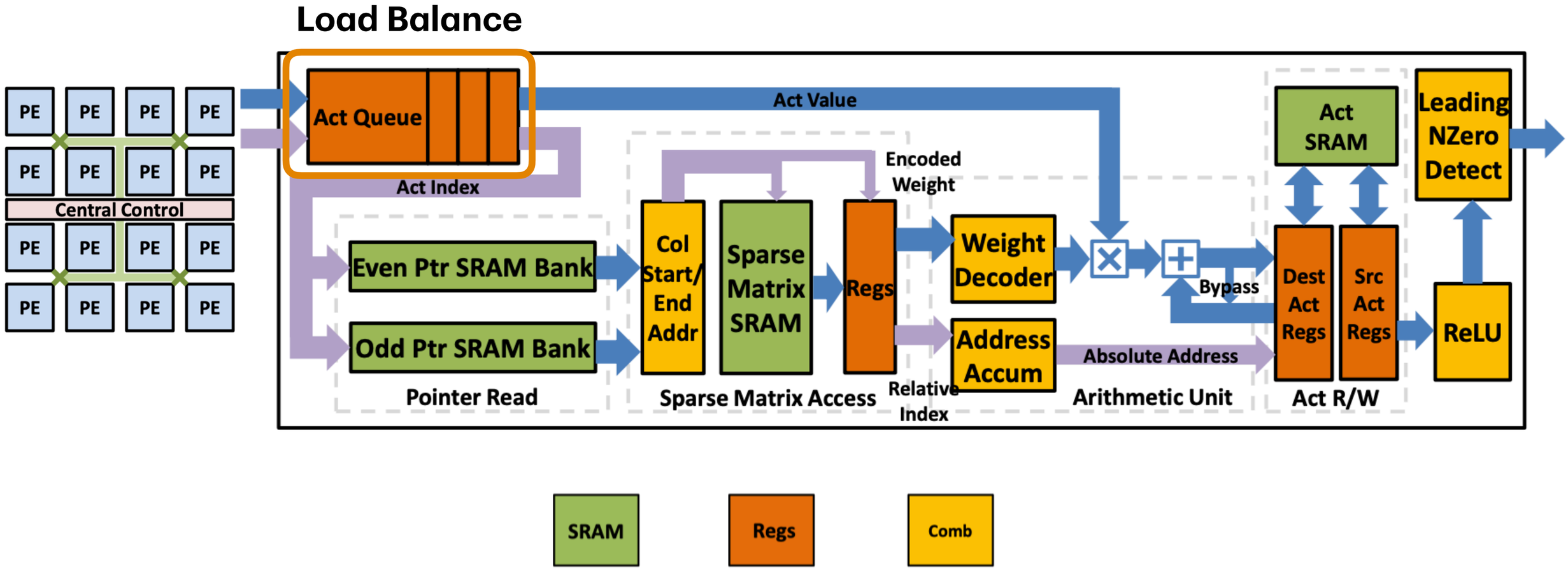
Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: The Micro Architecture of Each PE



Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

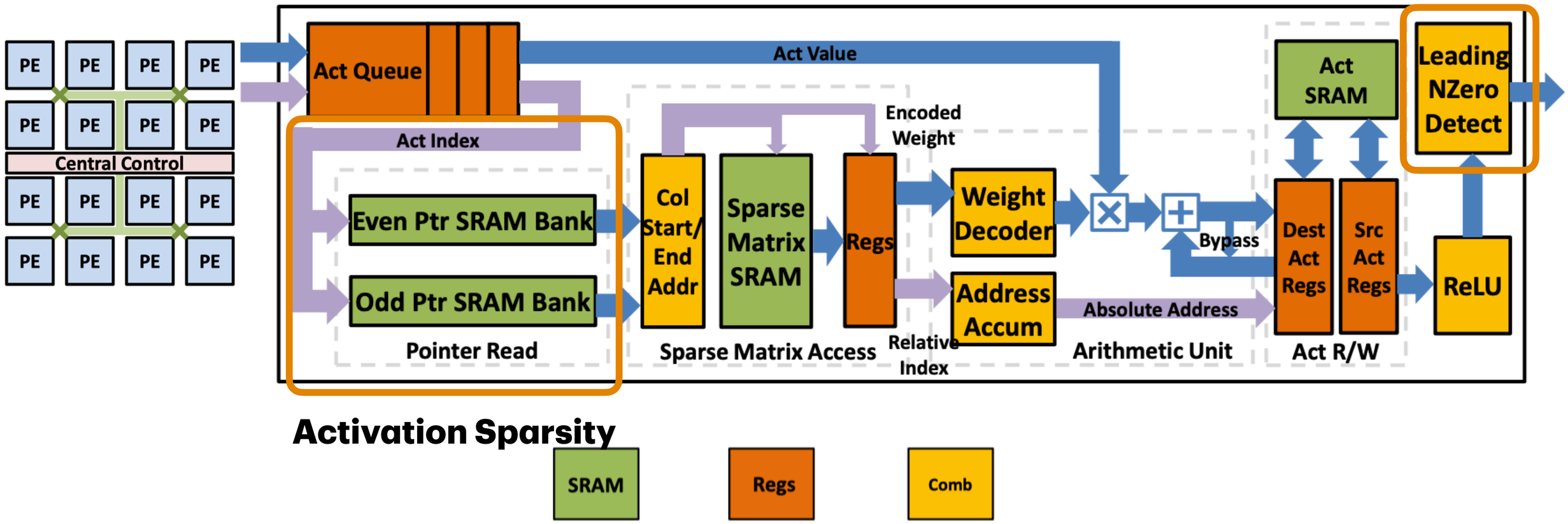
EIE: The Micro Architecture of Each PE



Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: The Micro Architecture of Each PE

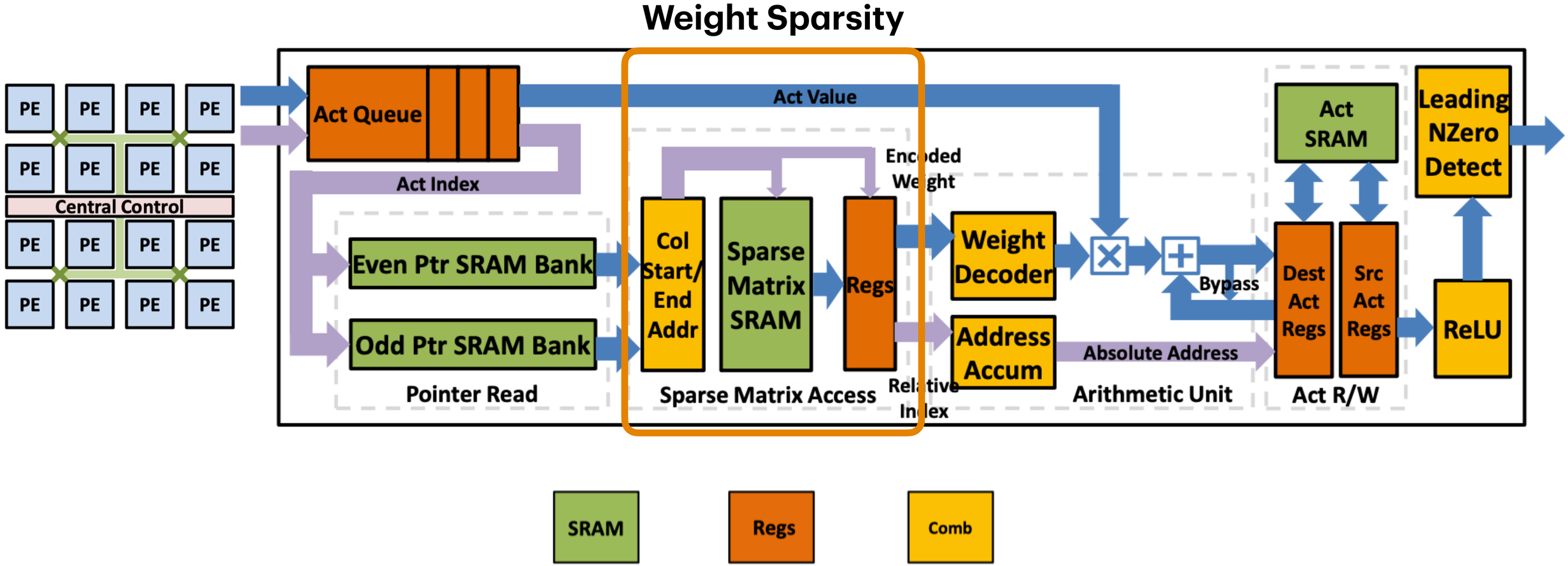
Take advantage of activation sparsity by
Leading NZero Detect and **Act Queue**



Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: The Micro Architecture of Each PE

Take advantage of weight sparsity by
only **store the non-zero weights** in **SRAM**.

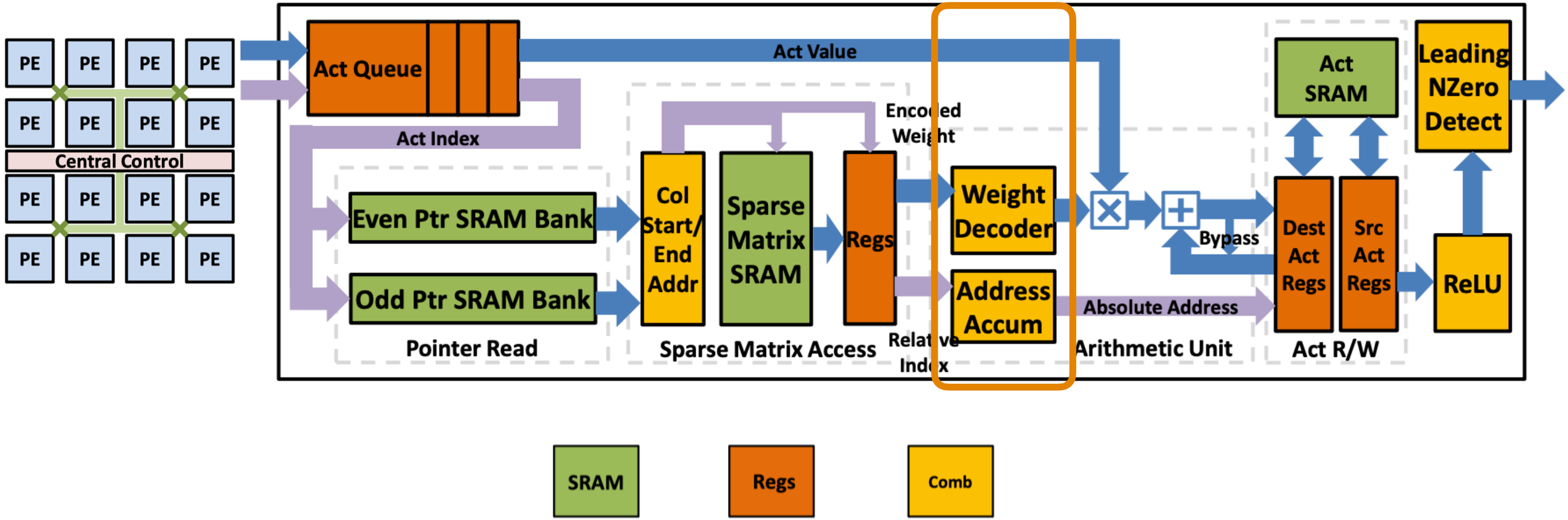


Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: The Micro Architecture of Each PE

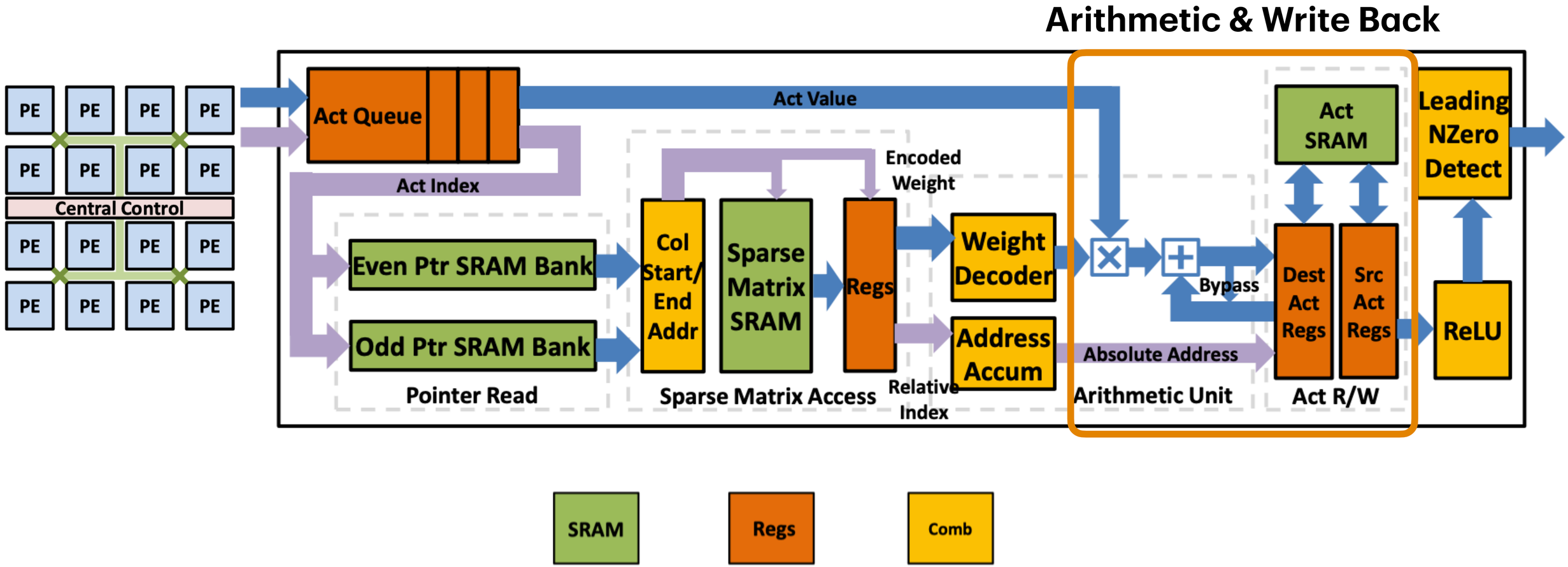
K-means **quantization** to store 4-bit of weight;
Use **relative index** to save #bits.
Weight Decoder and **Address Accum** in parallel

Weight Sharing



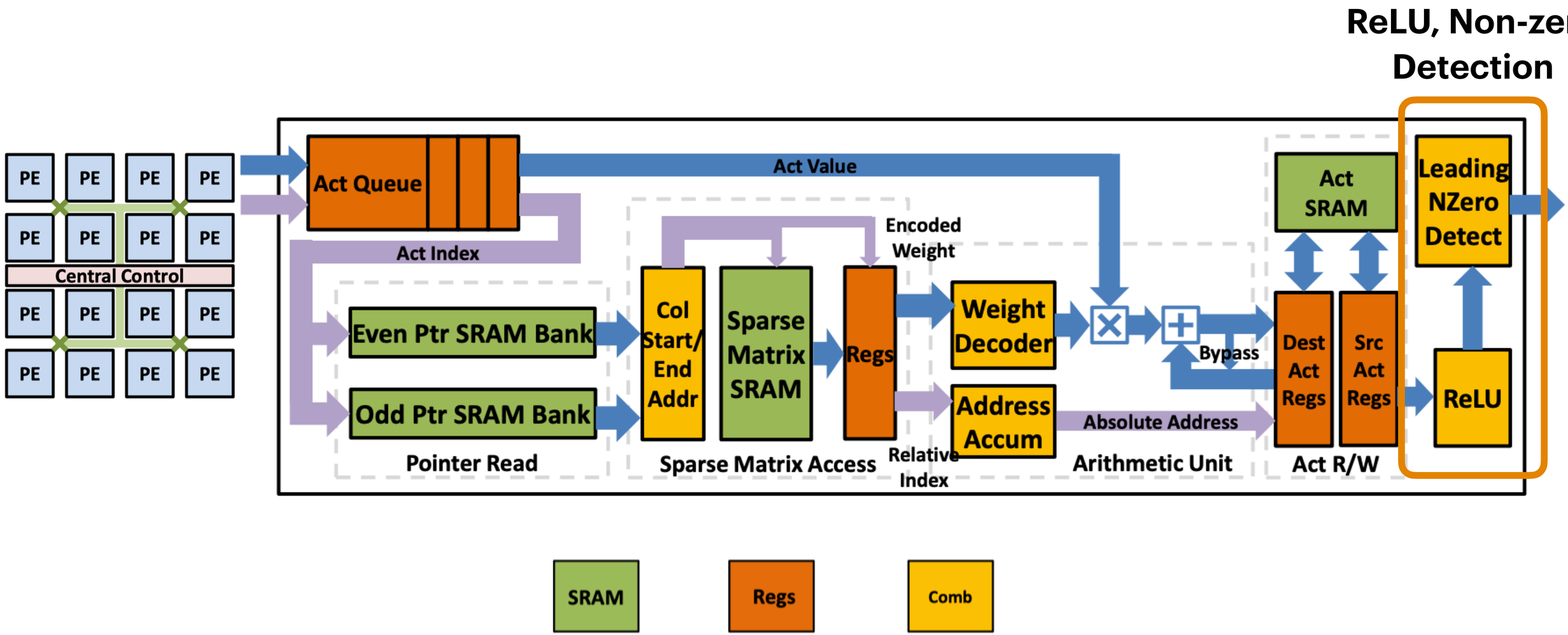
Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: The Micro Architecture of Each PE



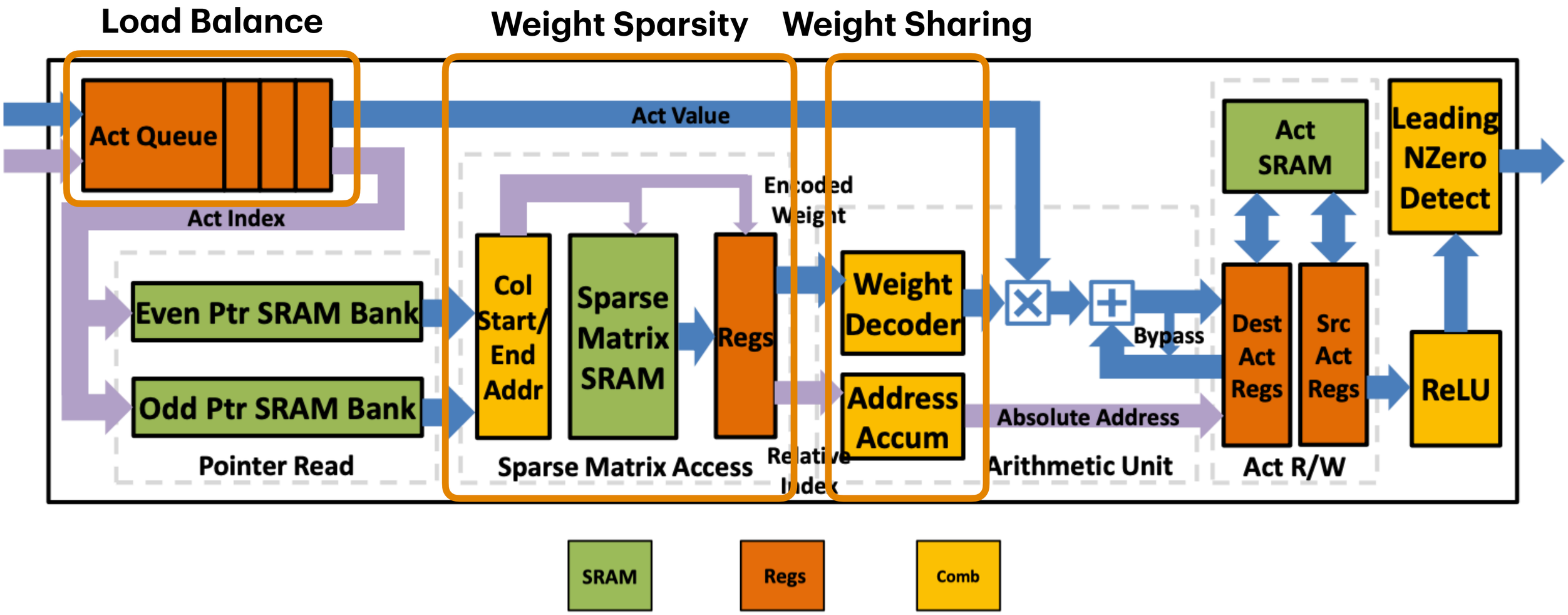
Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: The Micro Architecture of Each PE



Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: What's Special?



Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Performance

Benchmark (FC layers only)

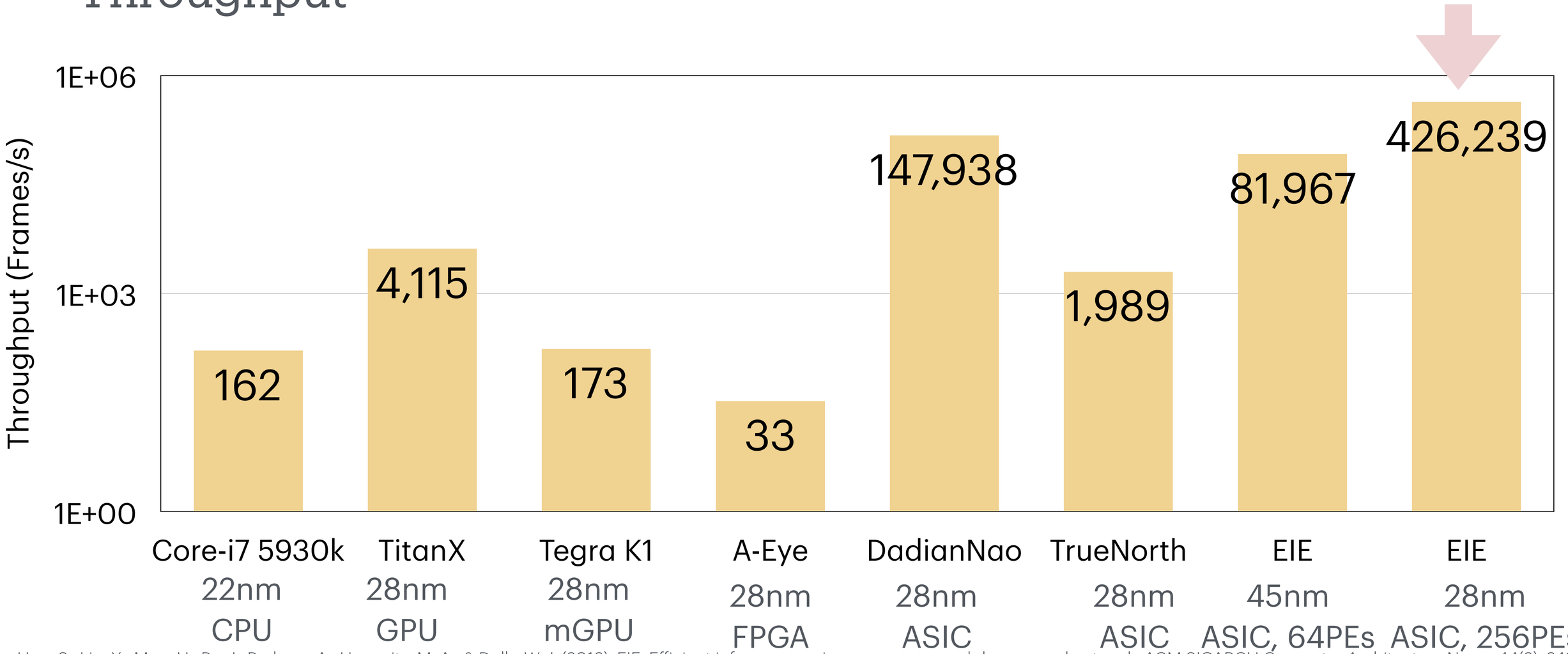
- CPU: Intel Core-i7 5930K
- GPU: NVIDIA TitanX
- Mobile GPU: NVIDIA Jetson TK1

Layer	Size	Weight Density	Activation Density	FLOP Reduction	Description
AlexNet-6	4096x9216	9%	35%	**33x**	AlexNet for image classification
AlexNet-7	4096x4096	9%	35%	**33x**	
AlexNet-8	1000x4096	25%	38%	**10x**	
VGG-6	4096x25088	4%	18%	**100x**	VGG-16 for image classification
VGG-7	4096x4096	4%	37%	**50x**	
VGG-8	1000x4096	23%	41%	**10x**	
NeuralTalk-We	600x4096	10%	100%	**10x**	RNN and LSTM for image caption
NeuralTalk-Wd	8791x600	11%	100%	**10x**	
NeuralTalk-LSTM	2400x1201	10%	100%	**10x**	

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Performance

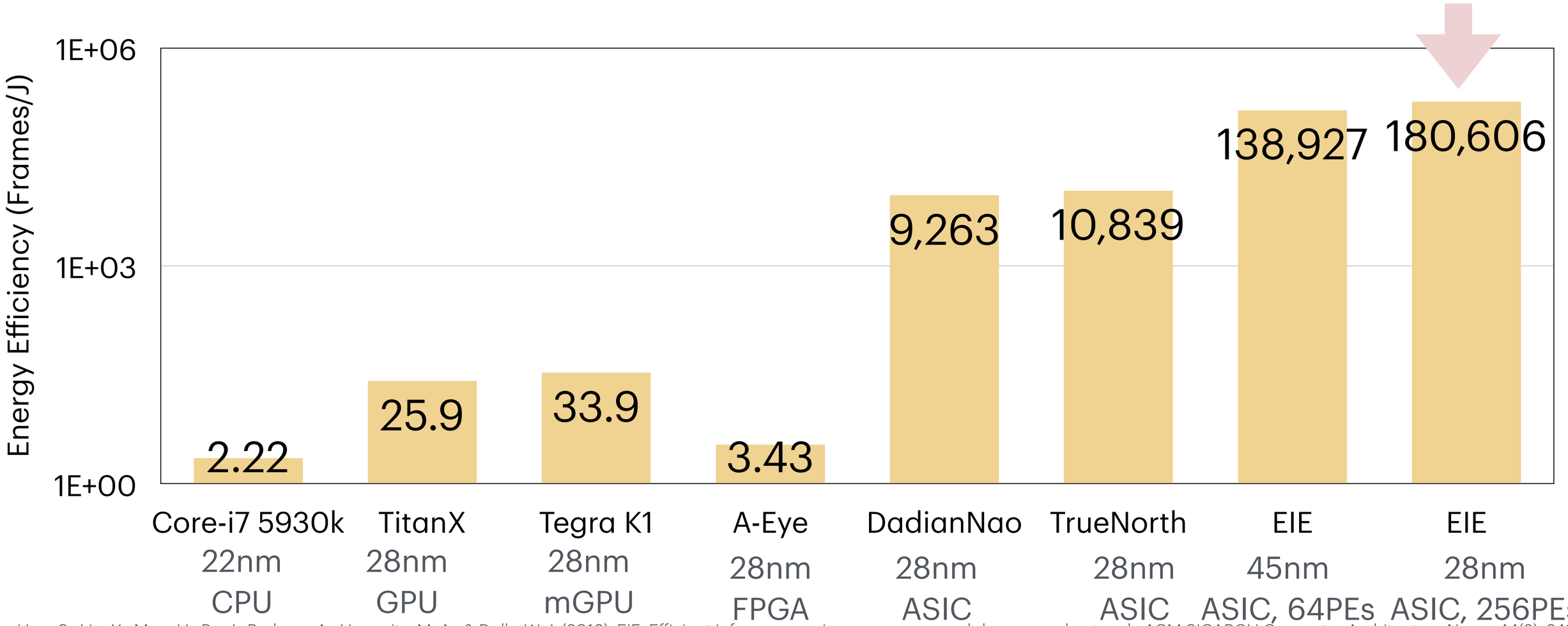
Throughput



Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Performance

Energy Efficiency



Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.

EIE: Efficient Inference Engine

The first DNN accelerator for sparse, compressed model

$0 \times A = 0$

Sparse Weight

90% static sparsity

10X less computation

5X less memory footprint

$W \times 0 = 0$

Sparse Activation

70% dynamic sparsity

3X less computation

$2.09, 1.92 \rightarrow 2$

Weight Sharing

4-bit weights

8X less memory footprint

Retrospective EIE

- **Pro**
  - Demonstrate that special-purpose hardware can make it cost-effiective to do sparse operations with matrices that are up to 50% dense
  - Exploits both weight sparsity and activation sparsity, not only saves energy by skipping zero weights, but also saves the cycle by not computing it
  - Supports fine-grained sparsity, and allows pruning to achieve a higher pruning ratio
  - Aggressive weight quantization (4bit) to save memory footprint. Decodes the weight to 16bit and uses 16bit arithmetic to maintain accuracy. (W4A16 approach is reborn in LLM)
- **Con**
  - Not as easily applied to arrays of vector processors. Improve: structured sparsity (N:M sparsity)
  - Control flow overhead, storage overhead. Improve: coarse grain sparsity
  - Only support FC layers. Reborn in LLM
  - Fits everythhing in SRAM. Practical for TinyML, not LLM.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2023). Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network. arXiv preprint arXiv:2306.09552.

Retrospective EIE

- **The first prinicple of efficient AI computing is to be lazy:**
  - Avoid redundant computation
  - Quickly reject the work
  - Or delay the work
- The future AI models will be sparse at various granularity and strcutures. Co-designed with specializaed accelerators, sparse models will become more efficient and accessible.
  - Generative AI: spatial sparsity [SIGE, NeurIPS'22]
  - Transformer: token sparsity, progressive quantization [SpAtten, HPCA'21]
  - Video: temporal sparsity [TSM, ICCV'19]
  - Point cloud: spatial sparsity [TorchSparse, MLSys'22 & PointAcc, Micro'22]

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2023). Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network. arXiv preprint arXiv:2306.09552.

M:N Weight Sparsity

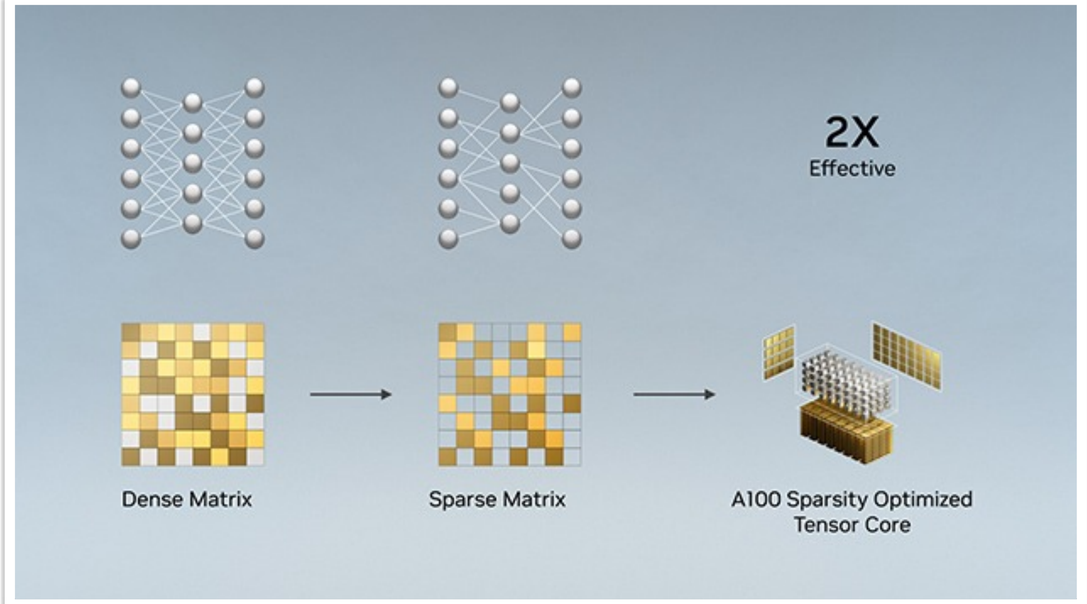
Accelerating Sparse Deep Neural Networks

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic,
Dusan Stosic, Ganesh Venkatesh, Chong Yu, Paulius Micikevicius

NVIDIA

{asitm, jalbericiola, jpool, darkos, dstosic, chongy, pauliusm}
@nvidia.com

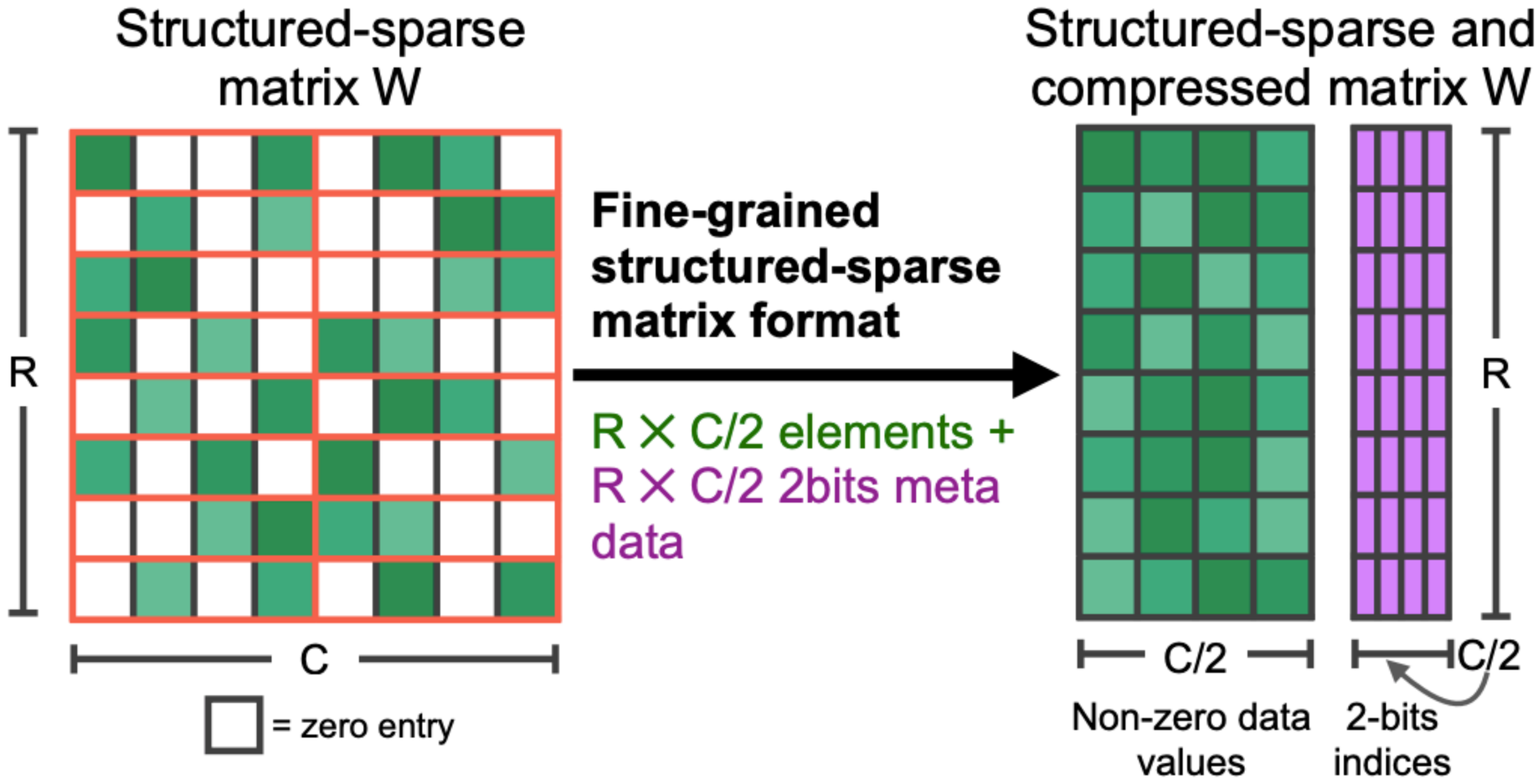
Abstract. As neural network model sizes have dramatically increased, so has the interest in various techniques to reduce their parameter counts and accelerate their execution. An active area of research in this field is sparsity – encouraging zero values in parameters that can then be discarded from storage or computations. While most research focuses on high levels of sparsity, there are challenges in universally maintaining model accuracy as well as achieving significant speedups over modern matrix-math hardware. To make sparsity adoption practical, the NVIDIA Ampere GPU architecture introduces sparsity support in its matrix-math units, Tensor Cores. We present the design and behavior of Sparse Tensor Cores, which exploit a 2:4 (50%) sparsity pattern that leads to twice the math throughput of dense matrix units. We also describe a simple workflow for training networks that both satisfy 2:4 sparsity pattern requirements *and* maintain accuracy, verifying it on a wide range of common tasks and model architectures. This workflow makes it easy to prepare accurate models for efficient deployment on Sparse Tensor Cores.



Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., ... & Micikevicius, P. (2021). Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378.

M:N Sparsity

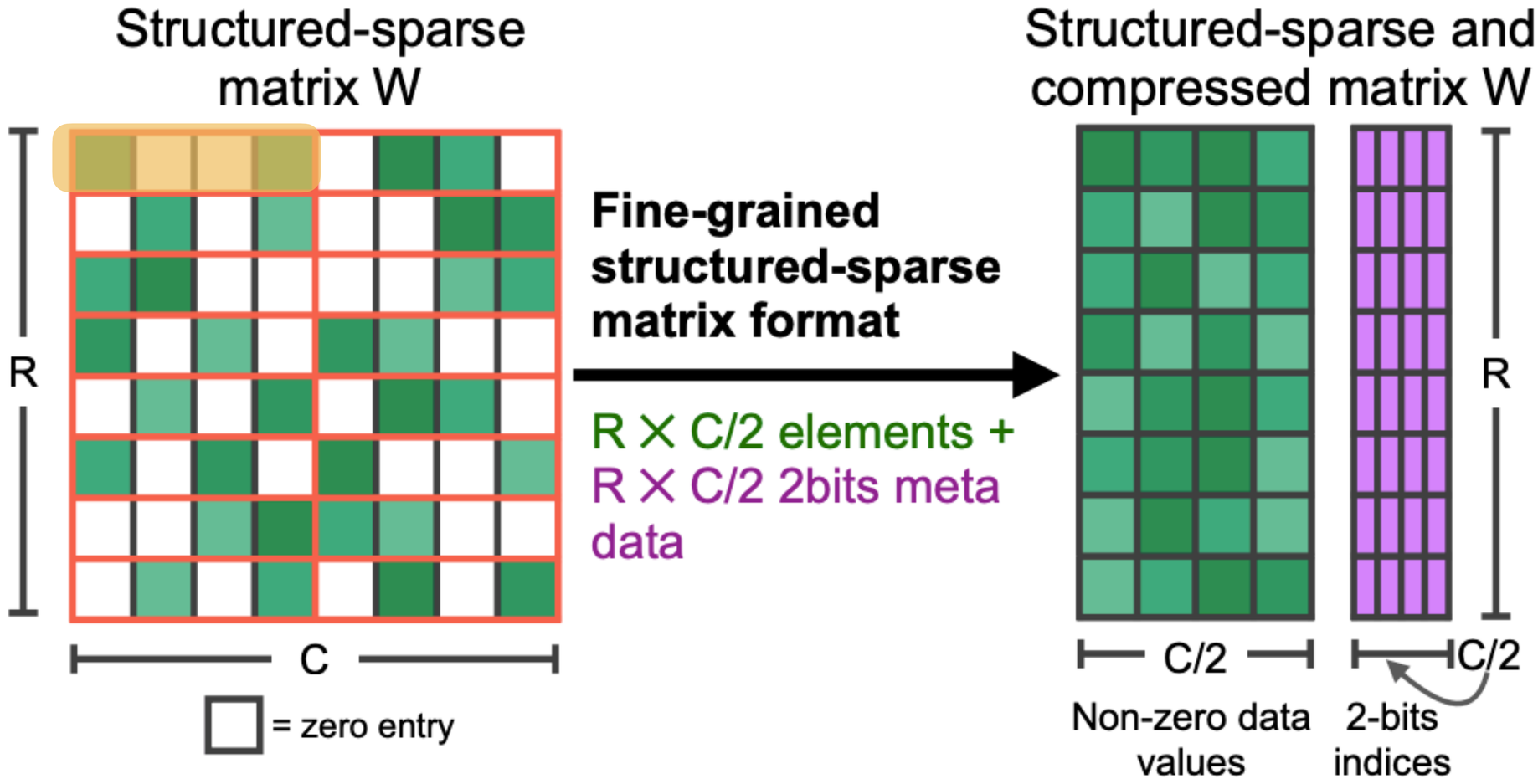
Two weights are nonzero out of four consecutive weights (2:4 sparsity)



Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., ... & Micikevicius, P. (2021). Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378.

M:N Sparsity

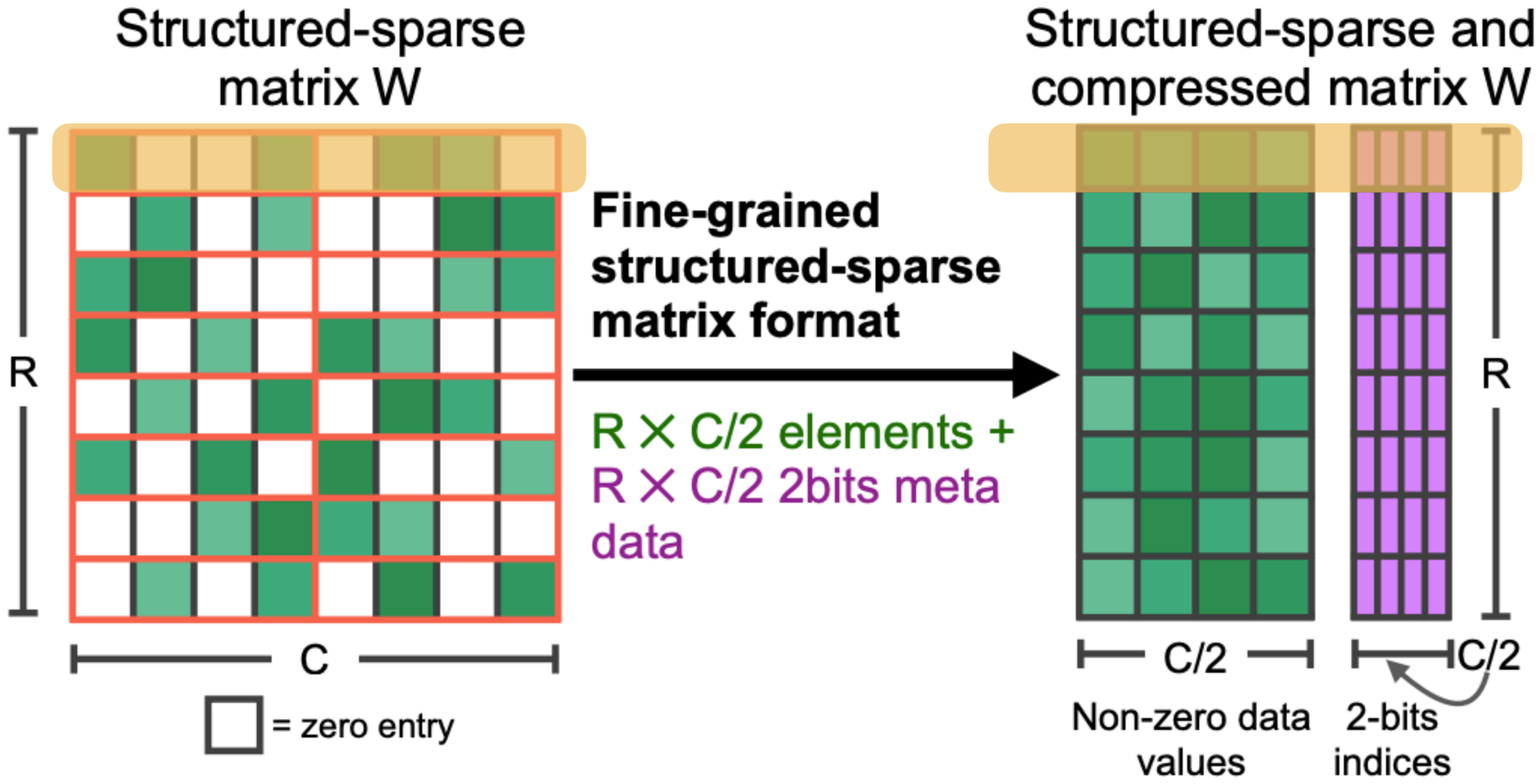
Two weights are nonzero out of four consecutive weights (2:4 sparsity)



Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., ... & Micikevicius, P. (2021). Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378.

M:N Sparsity

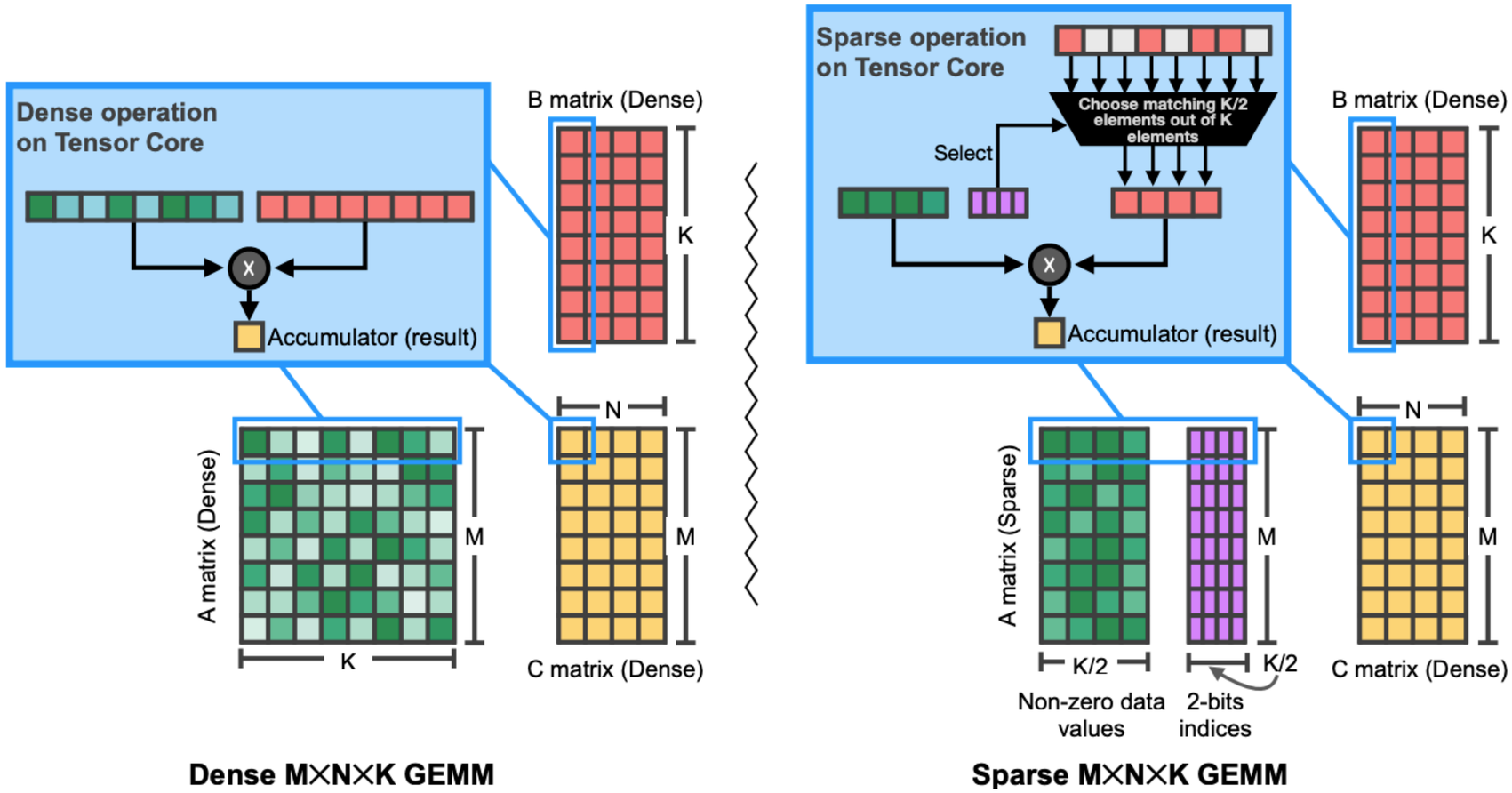
Two weights are nonzero out of four consecutive weights (2:4 sparsity)



Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., ... & Micikevicius, P. (2021). Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378.

System Support for M:N Sparsity

Mapping M:N sparse matrices onto NVIDIA tensor cores



- Indices are used to mask out the inputs. Only 2 multiplications will be done out of four.

Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., ... & Micikevicius, P. (2021). Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378.

System Support for M:N Sparsity

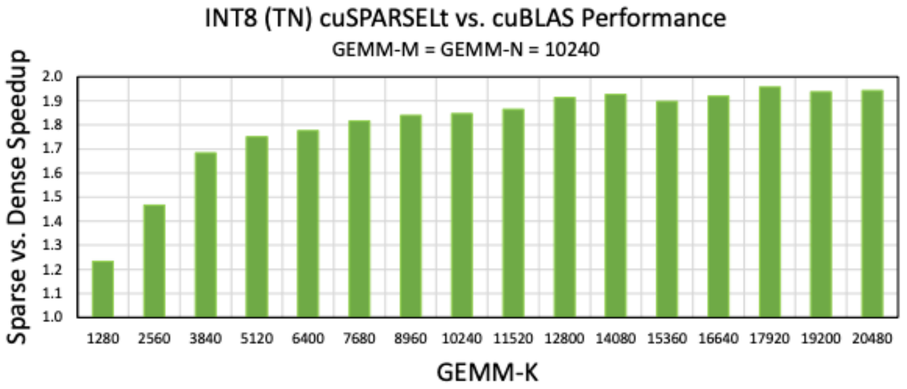


Fig. 3. Comparison of sparse and dense INT8 GEMMs on NVIDIA A100 Tensor Cores. Larger GEMMs achieve nearly a 2× speedup with Sparse Tensor Cores.

Pruning CNNs with 2:4 sparsity will bring about large speedup for GEMM workloads and will not incur performance drop for DNN models

Table 2. Top-1 accuracy of image classification networks evaluated on the ImageNet ILSVRC2012 dataset with 2:4 sparsity.

Network	Accuracy		
	Dense FP16	Sparse FP16	Sparse INT8
ResNet-34	73.7	73.9	73.7
ResNet-50	76.1	76.2	76.2
ResNet-50 (SWSL)	81.1	80.9	80.9
ResNet-101	77.7	78.0	77.9
ResNeXt-50-32x4	77.6	77.7	77.7
ResNeXt-101-32x16	79.7	79.9	79.9
ResNeXt-101-32x16 (WSL)	84.2	84.0	84.2
DenseNet-121	75.5	75.3	75.3
DenseNet-161	78.8	78.8	78.9
Wide ResNet-50	78.5	78.6	78.5
Wide ResNet-101	78.9	79.2	79.1
Inception v3	77.1	77.1	77.1
Xception	79.2	79.2	79.2
VGG-11	70.9	70.9	70.8
VGG-16	74.0	74.1	74.1
VGG-19	75.0	75.0	75.0
SUNet-128	75.6	76.0	75.4
SUNet-7-128	76.4	76.5	76.3
DRN26	75.2	75.3	75.3
DRN-105	79.4	79.5	79.4

Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., ... & Micikevicius, P. (2021). Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378.

Activation Sparsity

Point-cloud Specific

TORCHSPARSE: EFFICIENT POINT CLOUD INFERENCE ENGINE

Haotian Tang [*1] **Zhijian Liu** [*1] **Xiuyu Li** [*2] **Yujun Lin** [1] **Song Han** [1]
https://torchsparse.mit.edu

ABSTRACT

Deep learning on point clouds has received increased attention thanks to its wide applications in AR/VR and autonomous driving. These applications require low latency and high accuracy to provide real-time user experience and ensure user safety. Unlike conventional dense workloads, the sparse and irregular nature of point clouds poses severe challenges to running sparse CNNs efficiently on the general-purpose hardware. Furthermore, existing sparse acceleration techniques for 2D images do not translate to 3D point clouds. In this paper, we introduce TorchSparse, a high-performance point cloud inference engine that accelerates the sparse convolution computation on GPUs. TorchSparse directly optimizes the two bottlenecks of sparse convolution: **irregular computation** and **data movement**. It applies *adaptive matrix multiplication grouping* to trade computation for better regularity, achieving 1.4-1.5× speedup for matrix multiplication. It also optimizes the data movement by adopting *vectorized*, *quantized* and *fused locality-aware memory access*, reducing the memory movement cost by 2.7×. Evaluated on seven representative models across three benchmark datasets, TorchSparse achieves **1.6**× and **1.5**× measured end-to-end speedup over the state-of-the-art MinkowskiEngine and SpConv, respectively.

Sparse Convolution Library

PointAcc: Efficient Point Cloud Accelerator

Yujun Lin, Zhekai Zhang, Haotian Tang, Hanrui Wang, Song Han
MIT
{yujunlin, zhangzk, kentang, hanrui, songhan}@mit.edu
https://pointacc.mit.edu

ABSTRACT

Deep learning on point clouds plays a vital role in a wide range of applications such as autonomous driving and AR/VR. These applications interact with people in real time on edge devices and thus require low latency and low energy. Compared to projecting the point cloud to 2D space, directly processing 3D point cloud yields higher accuracy and lower #MACs. However, the extremely sparse nature of point cloud poses challenges to hardware acceleration. For example, we need to explicitly determine the nonzero outputs and search for the nonzero neighbors (mapping operation), which is unsupported in existing accelerators. Furthermore, explicit gather and scatter of sparse features are required, resulting in large data movement overhead.

In this paper, we comprehensively analyze the performance bottleneck of modern point cloud networks on CPU/GPU/TPU. To address the challenges, we then present PointAcc, a novel point cloud deep learning accelerator. PointAcc maps diverse mapping operations onto one versatile ranking-based kernel, streams the sparse computation with configurable caching, and temporally fuses consecutive dense layers to reduce the memory footprint. Evaluated on 8 point cloud models across 4 applications, PointAcc achieves 3.7× speedup and 22× energy savings over RTX 2080Ti GPU. Co-designed with light-weight neural networks, PointAcc rivals the prior accelerator Mesorasi by 100× speedup with 9.1% higher accuracy running segmentation on the S3DIS dataset. PointAcc paves the way for efficient point cloud recognition.

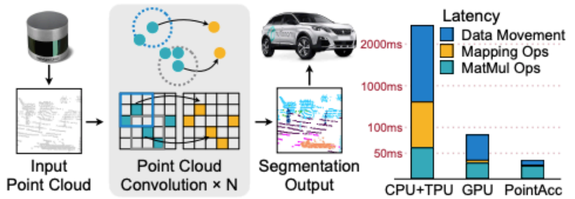


Figure 1: Point cloud deep learning is crucial for real-time AI applications. PointAcc accelerates point cloud computations by resolving sparsity and data movement bottlenecks.

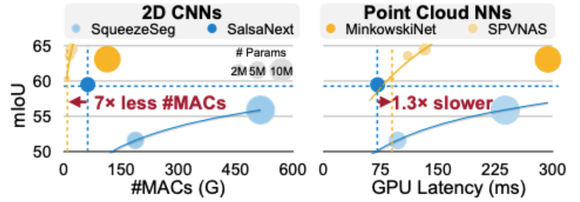


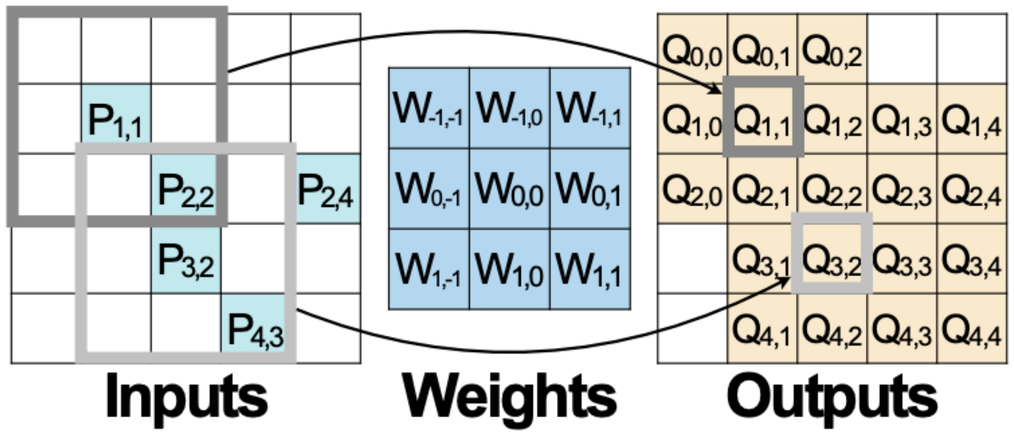
Figure 2: Compared to 2D CNNs, point cloud networks have higher accuracy and lower #MACs, but higher GPU latency due to low utilization brought by sparsity and irregularity.

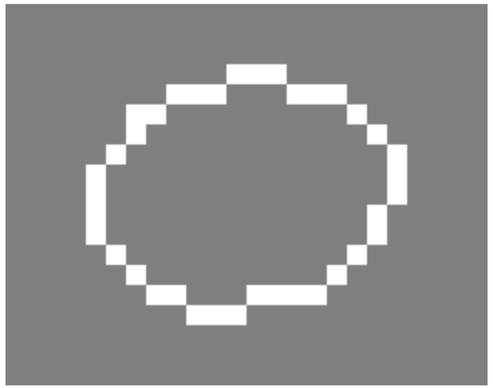
Hardware Accelerator for Sparse Convolution

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.
Lin, Y., Zhang, Z., Tang, H., Wang, H., & Han, S. (2021, October). Pointacc: Efficient point cloud accelerator. In MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 449-461).

Conventional Convolution

Each nonzero input is multiplied with all nonzero weights





- Input sparsity from ReLU 🤩
- Nonzeros will dilate after multiplication 😢

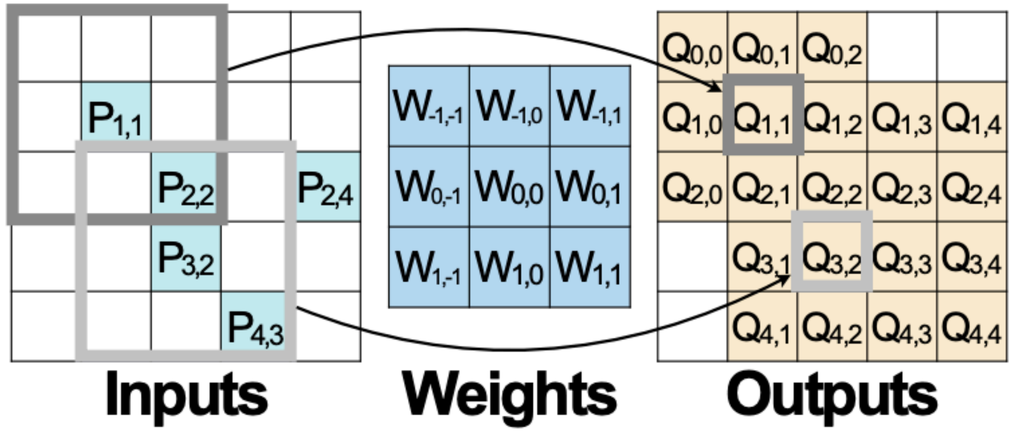
Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

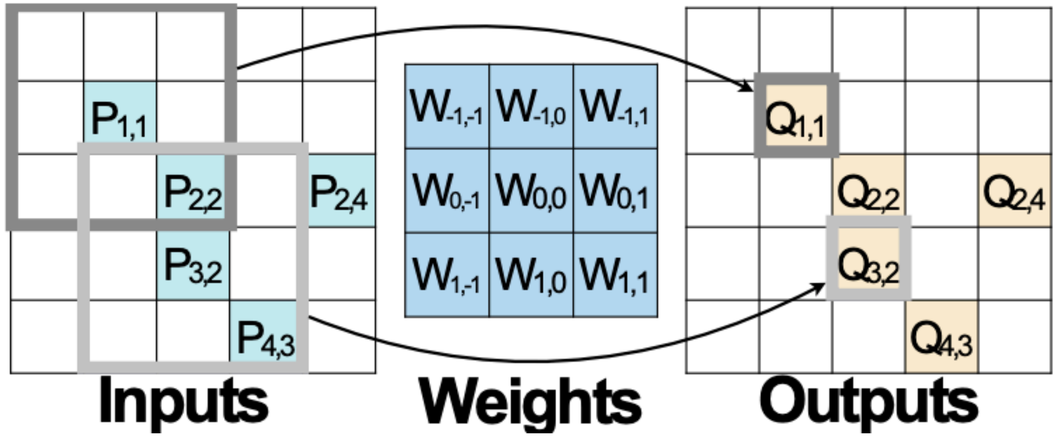
Figure source: 1, 2

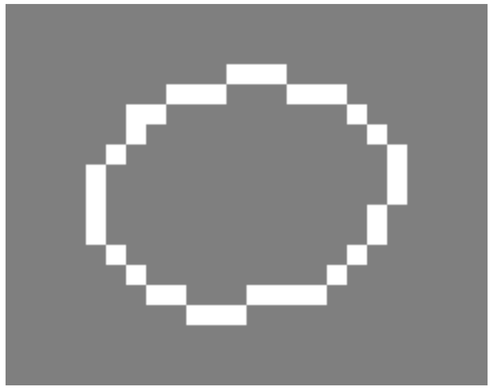
Sparse Convolution on Sparse Inputs

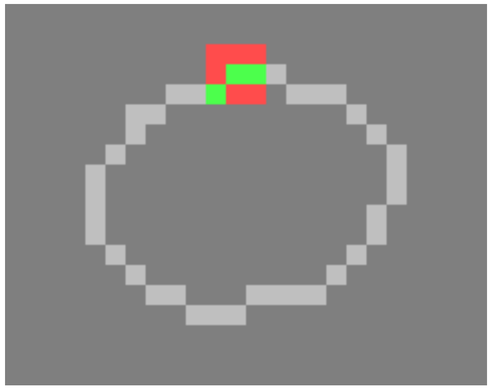
Sparse computation determined by maps

Map								
IN	$P_{1,1}$	$P_{3,2}$	$P_{2,2}$	$P_{1,1}$	$P_{2,2}$	…	$P_{3,2}$	$P_{2,2}$ $P_{4,3}$
WGT	$W_{-1,-1}$	$W_{-1,-1}$	$W_{-1,}$	$W_{0,0}$	$W_{0,0}$	…	$W_{1,0}$	$W_{1,1}$ $W_{1,1}$
OUT	$Q_{2,2}$	$Q_{4,3}$	$Q_{3,2}$	$Q_{1,1}$	$Q_{2,2}$	…	$Q_{2,2}$	$Q_{1,1}$ $Q_{3,2}$









Graham, B., & Van der Maaten, L. (2017). Submanifold sparse convolutional networks. arXiv preprint arXiv:1706.01307.

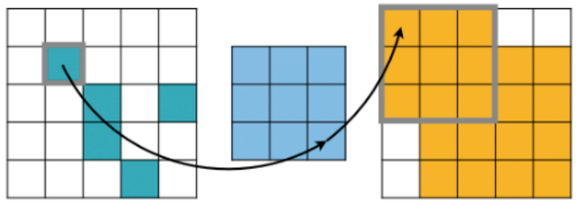
Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

Figure source: 1, 2

Sparse Convolution Computation

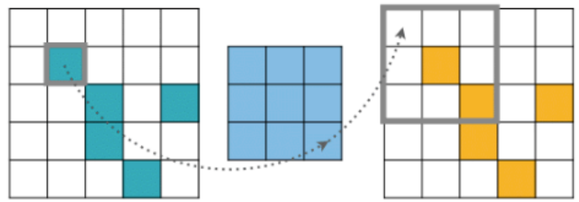
A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



(P_0, Q_0, $W_{1,1}$)

Sparse Convolution



No compute

Maps
(In, Out, Wgt)

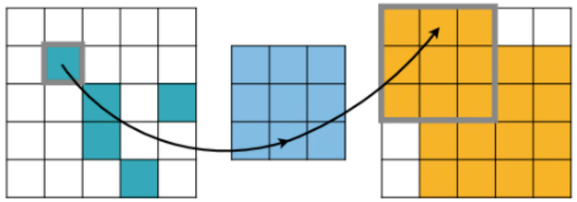
Computation
($f_{Out} = f_{Out} + f_{In} \times W_{Wgt}$) for each entry in the maps

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

Sparse Convolution Computation

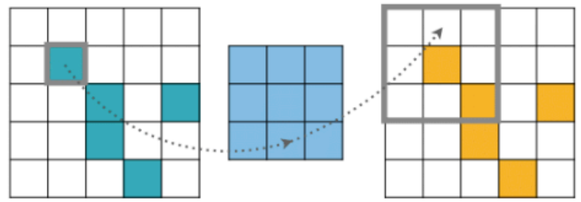
A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



$(P_0, Q_0, W_{1,1})$
$(P_0, Q_1, W_{1,0})$

Sparse Convolution



No compute
No compute

Maps
(In, Out, Wgt)

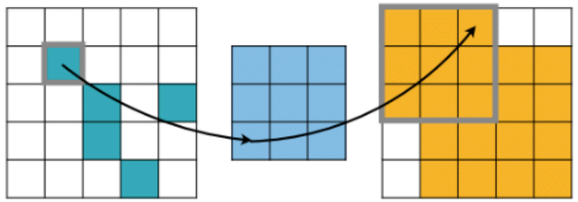
Computation
($f_{Out} = f_{Out} + f_{In} \times W_{Wgt}$) for each entry in the maps

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

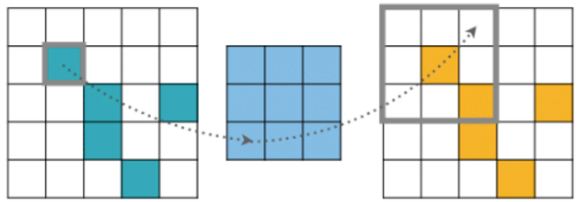
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

$(P_0, Q_0, W_{1,1})$
$(P_0, Q_1, W_{1,0})$
$(P_0, Q_2, W_{1,-1})$

No compute
No compute
No compute

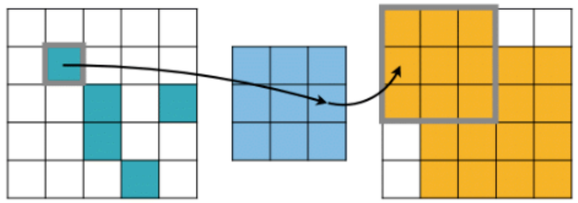
Computation
$(f_{Out} = f_{Out} + f_{In} \times W_{Wgt})$ for each entry in the maps

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

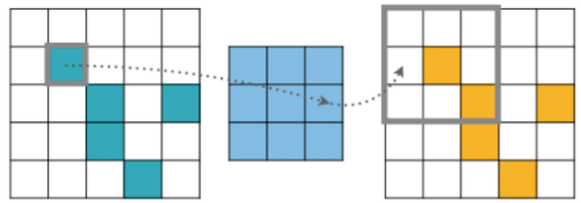
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

$(P_0, Q_0, W_{1,1})$
$(P_0, Q_1, W_{1,0})$
$(P_0, Q_2, W_{1,-1})$
$(P_0, Q_3, W_{0,1})$

No compute
No compute
No compute
No compute

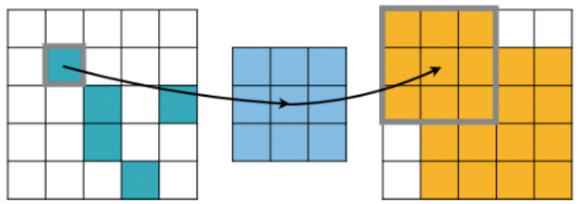
Computation
$(f_{Out} = f_{Out} + f_{In} \times W_{Wgt})$ for each entry in the maps

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

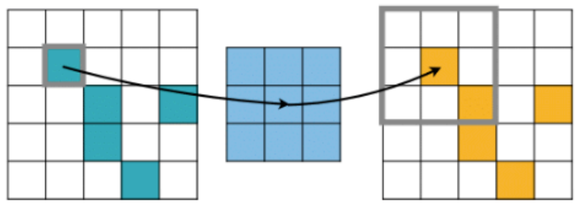
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

$(P_0, Q_0, W_{1,1})$
$(P_0, Q_1, W_{1,0})$
$(P_0, Q_2, W_{1,-1})$
$(P_0, Q_3, W_{0,1})$
$(P_0, Q_4, W_{0,0})$

No compute
No compute
No compute
No compute
$(P_0, Q_0, W_{0,0})$

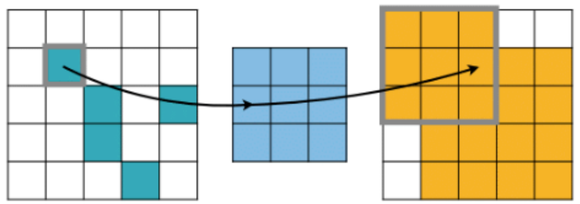
Computation
$(f_{Out} = f_{Out} + f_{In} \times W_{Wgt})$ for each entry in the maps

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

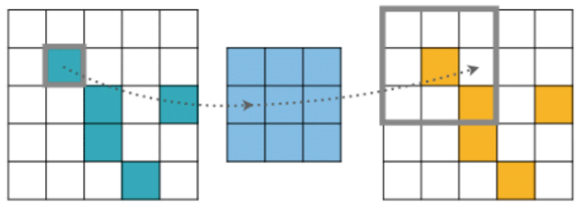
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

Computation
($f_{Out} = f_{Out} + f_{In} \times W_{Wgt}$) for each entry in the maps

(P_0, Q_0, $W_{1,1}$)
(P_0, Q_1, $W_{1,0}$)
(P_0, Q_2, $W_{1,-1}$)
(P_0, Q_3, $W_{0,1}$)
(P_0, Q_4, $W_{0,0}$)
(P_0, Q_5, $W_{0,-1}$)

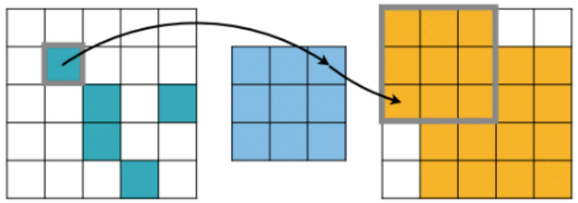
No compute
No compute
No compute
No compute
No compute
(P_0, Q_0, $W_{0,0}$)
No compute

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

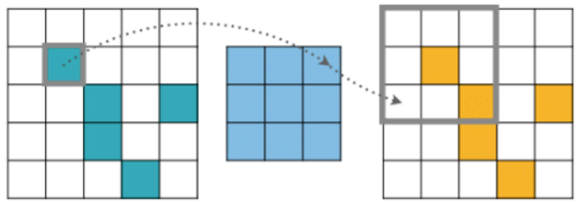
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

Computation
($f_{Out} = f_{Out} + f_{In} \times W_{Wgt}$) for each entry in the maps

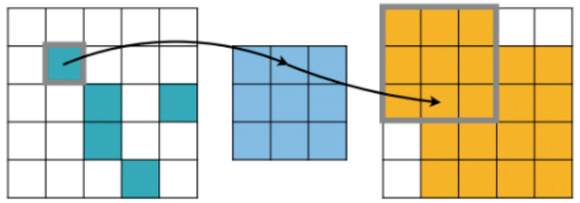
Conventional Convolution	Sparse Convolution
(P_0, Q_0, $W_{1,1}$)	No compute
(P_0, Q_1, $W_{1,0}$)	No compute
(P_0, Q_2, $W_{1,-1}$)	No compute
(P_0, Q_3, $W_{0,1}$)	No compute
(P_0, Q_4, $W_{0,0}$)	(P_0, Q_0, $W_{0,0}$)
(P_0, Q_5, $W_{0,-1}$)	No compute
(P_0, Q_8, $W_{-1,1}$)	No compute

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

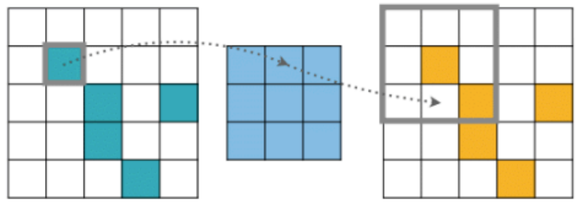
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

Computation
($f_{Out} = f_{Out} + f_{In} \times W_{Wgt}$) for each entry in the maps

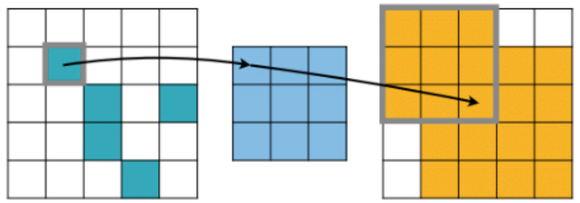
Conventional Convolution	Sparse Convolution
$(P_0, Q_0, W_{1,1})$	No compute
$(P_0, Q_1, W_{1,0})$	No compute
$(P_0, Q_2, W_{1,-1})$	No compute
$(P_0, Q_3, W_{0,1})$	No compute
$(P_0, Q_4, W_{0,0})$	$(P_0, Q_0, W_{0,0})$
$(P_0, Q_5, W_{0,-1})$	No compute
$(P_0, Q_8, W_{-1,1})$	No compute
$(P_0, Q_9, W_{-1,0})$	No compute

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

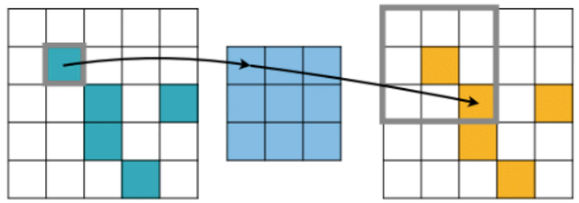
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

Computation
($f_{Out} = f_{Out} + f_{In} \times W_{Wgt}$) for each entry in the maps

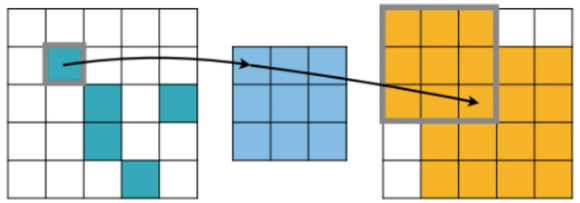
Conventional Convolution	Sparse Convolution
(P_0, Q_0, $W_{1,1}$)	No compute
(P_0, Q_1, $W_{1,0}$)	No compute
(P_0, Q_2, $W_{1,-1}$)	No compute
(P_0, Q_3, $W_{0,1}$)	No compute
(P_0, Q_4, $W_{0,0}$)	(P_0, Q_0, $W_{0,0}$)
(P_0, Q_5, $W_{0,-1}$)	No compute
(P_0, Q_8, $W_{-1,1}$)	No compute
(P_0, Q_9, $W_{-1,0}$)	No compute
(P_0, Q_{10}, $W_{-1,-1}$)	(P_0, Q_1, $W_{-1,-1}$)

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

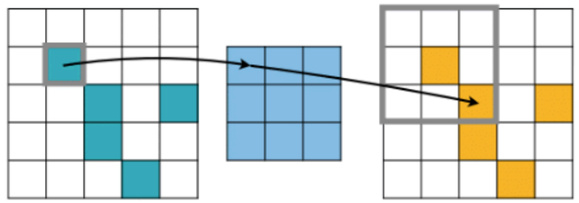
Sparse Convolution Computation

A sparse set of dense MMA, with rules defined by maps

Conventional Convolution



Sparse Convolution



Maps
(In, Out, Wgt)

Computation
($f_{Out} = f_{Out} + f_{In} \times W_{Wgt}$) for each entry in the maps

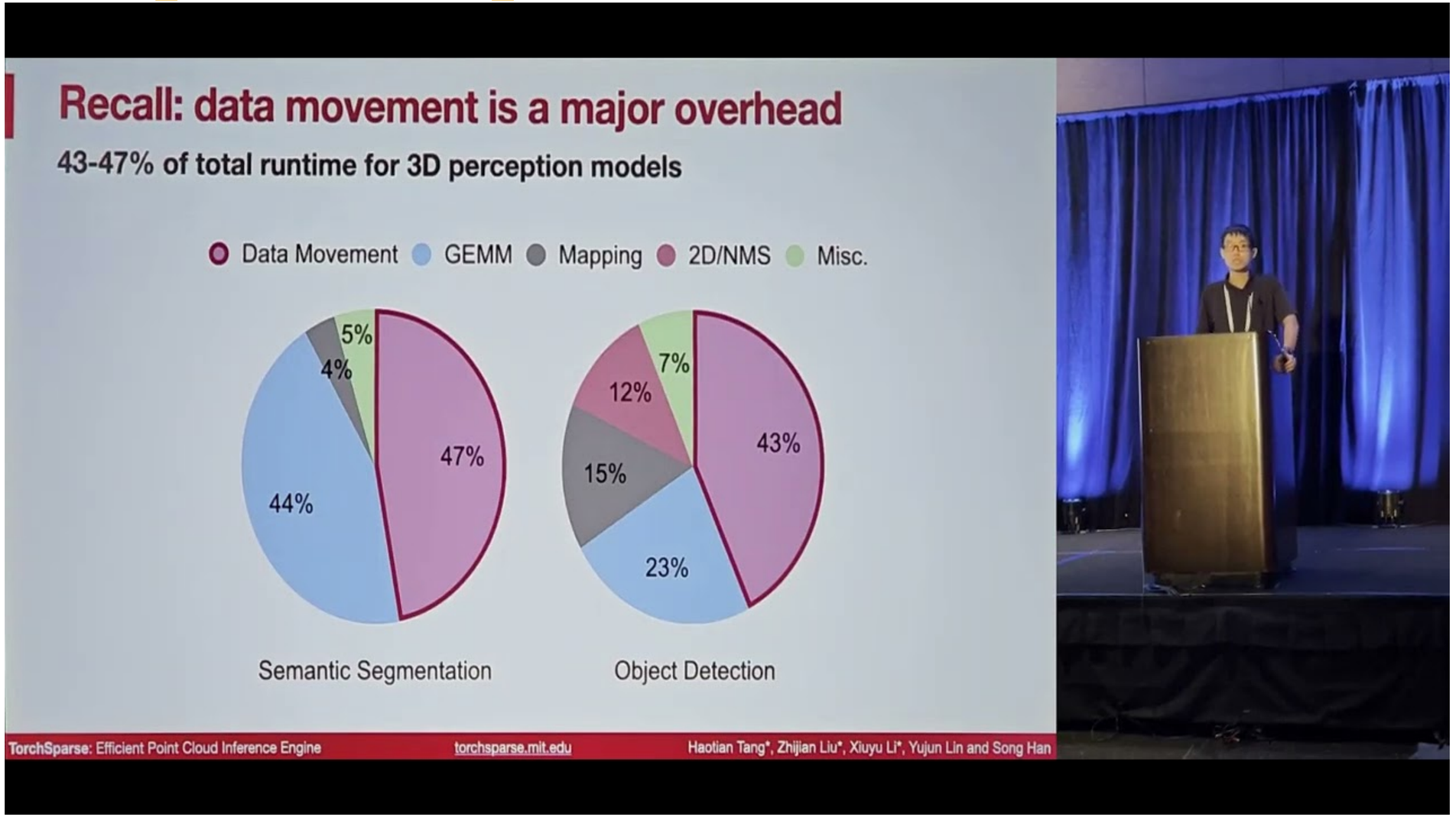
$(P_0, Q_0, W_{1,1})$	No compute
$(P_0, Q_1, W_{1,0})$	No compute
$(P_0, Q_2, W_{1,-1})$	No compute
$(P_0, Q_3, W_{0,1})$	No compute
$(P_0, Q_4, W_{0,0})$	$(P_0, Q_0, W_{0,0})$
$(P_0, Q_5, W_{0,-1})$	No compute
$(P_0, Q_8, W_{-1,1})$	No compute
$(P_0, Q_9, W_{-1,0})$	No compute
$(P_0, Q_{10}, W_{-1,-1})$	$(P_0, Q_1, W_{-1,-1})$

9 matrix multiplications

2 matrix multiplications

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

TorchSparse: Sparse Convolution Library



MLSys'22 TorchSparse: Efficient Point Cloud Inference Engine

Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.

PointAcc: Hardware Accelerator for Sparse Convolution

MIT HAN LAB

PointAcc

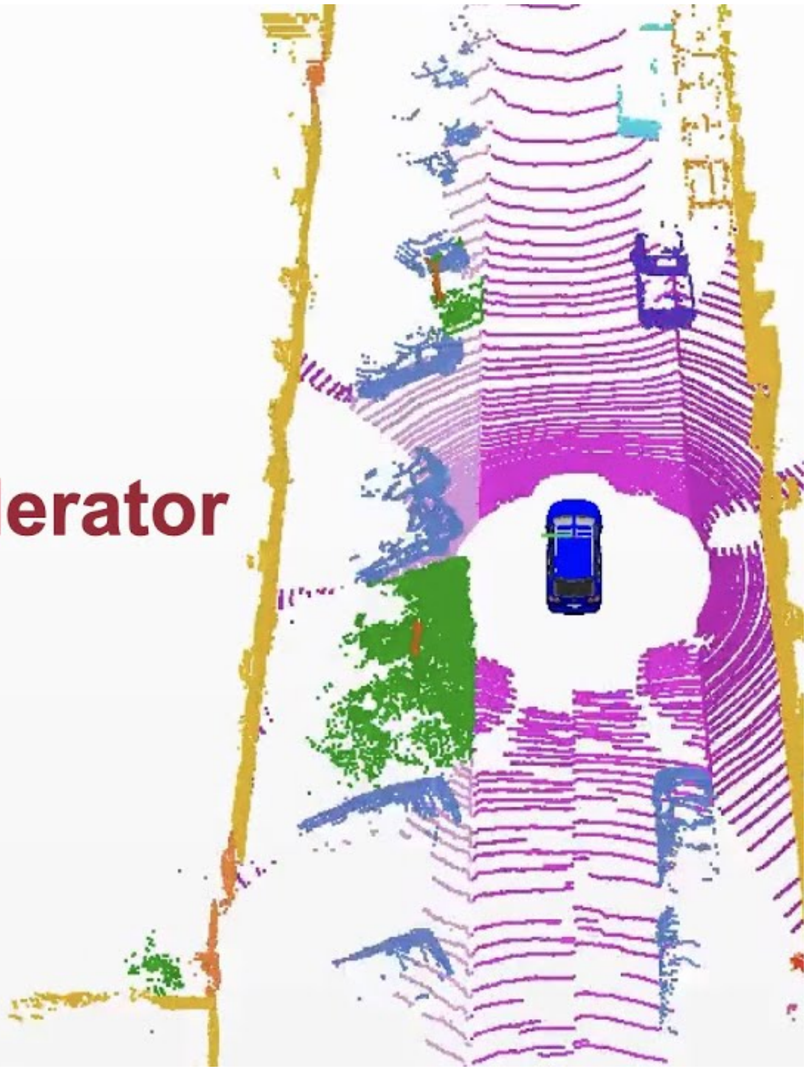
Efficient Point Cloud Accelerator

Yujun Lin, Zhekai Zhang, Haotian Tang,

Hanrui Wang, and Song Han

https://hanlab.mit.edu/projects/pointacc

video source: http://www.semantic-kitti.org/



Micro'21 PointAcc: Efficient Point Cloud Accelerator

Lin, Y., Zhang, Z., Tang, H., Wang, H., & Han, S. (2021, October). Pointacc: Efficient point cloud accelerator. In MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 449-461).

Lab 2 is out

Reference

- He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). AMC: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).
- Yang, T. J., Howard, A., Chen, B., Zhang, X., Go, A., Sandler, M., ... & Adam, H. (2018). Netadapt: Platform-aware neural network adaptation for mobile applications. In Proceedings of the European conference on computer vision (ECCV) (pp. 285-300).
- Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.
- Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE international conference on computer vision (pp. 2736-2744).
- Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016). EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3), 243-254.
- Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2023). Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network. arXiv preprint arXiv:2306.09552.
- Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., ... & Micikevicius, P. (2021). Accelerating sparse deep neural networks. arXiv preprint arXiv:2104.08378.
- Tang, H., Liu, Z., Li, X., Lin, Y., & Han, S. (2022). Torchsparse: Efficient point cloud inference engine. Proceedings of Machine Learning and Systems, 4, 302-315.
- Lin, Y., Zhang, Z., Tang, H., Wang, H., & Han, S. (2021, October). Pointacc: Efficient point cloud accelerator. In MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 449-461).
- Graham, B., & Van der Maaten, L. (2017). Submanifold sparse convolutional networks. arXiv preprint arXiv:1706.01307.
- Pruning and Sparsity [MIT 6.5940]