Design of Efficient and Privacy Preserving Machine Learning

Dr. Caiwen Ding

Assistant Professor

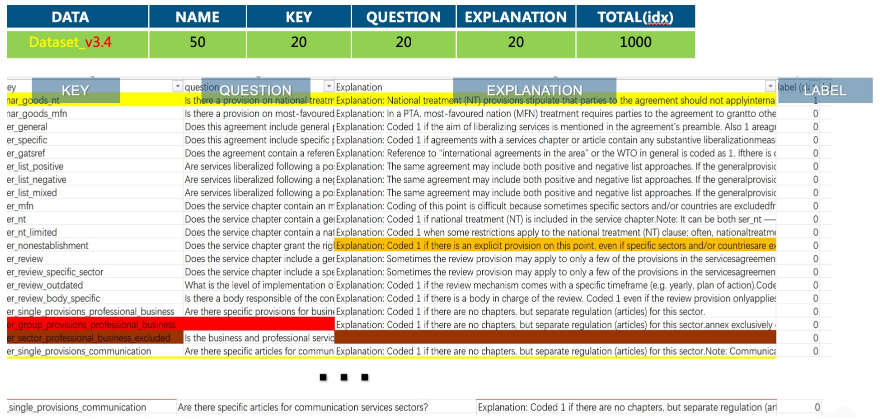
Department of Computer Science & Engineering

University of Connecticut

Email: caiwen.ding@uconn.edu

https://caiwending.cse.uconn.edu/

Machine Learning is Everywhere

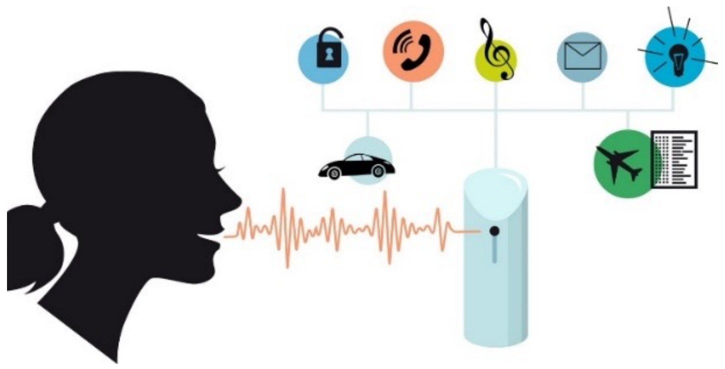


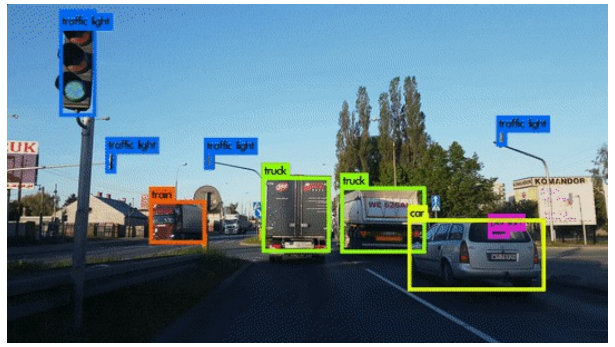
DATA	NAME	KEY	QUESTION	EXPLANATION	TOTAL(idx)
Dataset_v3.4	50	20	20	20	1000

ey KEY	question QUESTION	Explanation EXPLANATION	label (cl LABEL
nar_goods_nt	Is there a provision on national treatm	Explanation: National treatment (NT) provisions stipulate that parties to the agreement should not applyinterna	1
nar_goods_mfn	Is there a provision on most-favoured	Explanation: In a PTA, most-favoured nation (MFN) treatment requires parties to the agreement to grantto othe	0
er_general	Does this agreement include general p	Explanation: Coded 1 if the aim of liberalizing services is mentioned in the agreement's preamble. Also 1 areagr	0
er_specific	Does this agreement include specific p	Explanation: Coded 1 if agreements with a services chapter or article contain any substantive liberalizationmeas	0
er_gatsref	Does the agreement contain a referen	Explanation: Reference to "international agreements in the area" or the WTO in general is coded as 1. Ifthere is c	0
er_list_positive	Are services liberalized following a pos	Explanation: The same agreement may include both positive and negative list approaches. If the generalprovisio	0
er_list_negative	Are services liberalized following a neg	Explanation: The same agreement may include both positive and negative list approaches. If the generalprovisio	0
er_list_mixed	Are services liberalized following a pos	Explanation: The same agreement may include both positive and negative list approaches. If the generalprovisio	0
er_mfn	Does the service chapter contain an m	Explanation: Coding of this point is difficult because sometimes specific sectors and/or countries are excludedfr	0
er_nt	Does the service chapter contain a ger	Explanation: Coded 1 if national treatment (NT) is included in the service chapter.Note: It can be both ser_nt —	0
er_nt_limited	Does the service chapter contain a nat	Explanation: Coded 1 when some restrictions apply to the national treatment (NT) clause; often, nationaltreatme	0
er_nonestablishment	Does the service chapter grant the rigl	Explanation: Coded 1 if there is an explicit provision on this point, even if specific sectors and/or countriesare ex	0
er_review	Does the service chapter include a ger	Explanation: Sometimes the review provision may apply to only a few of the provisions in the servicesagreemen	0
er_review_specific_sector	Does the service chapter include a spe	Explanation: Sometimes the review provision may apply to only a few of the provisions in the servicesagreemen	0
er_review_outdated	What is the level of implementation o	Explanation: Coded 1 if the review mechanism comes with a specific timeframe (e.g. yearly, plan of action).Code	0
er_review_body_specific	Is there a body responsible of the con	Explanation: Coded 1 if there is a body in charge of the review. Coded 1 even if the review provision onlyapplies	0
er_single_provisions_professional_business	Are there specific provisions for busine	Explanation: Coded 1 if there are no chapters, but separate regulation (articles) for this sector.	0
er_group_provisions_professional_business		Explanation: Coded 1 if there are no chapters, but separate regulation (articles) for this sector.annex exclusively	0
er_sector_professional_business_excluded	Is the business and professional servic		0
er_single_provisions_communication	Are there specific articles for commun	Explanation: Coded 1 if there are no chapters, but separate regulation (articles) for this sector.Note: Communica	0

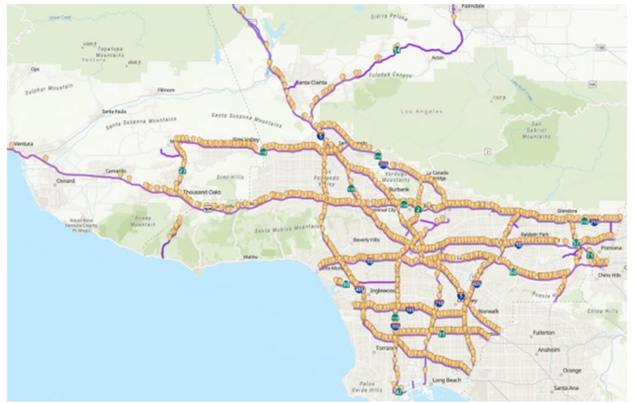
■ ■ ■

_single_provisions_communication	Are there specific articles for communication services sectors?	Explanation: Coded 1 if there are no chapters, but separate regulation (ar	0





https://gfycat.com/gifs/tag/object+detection



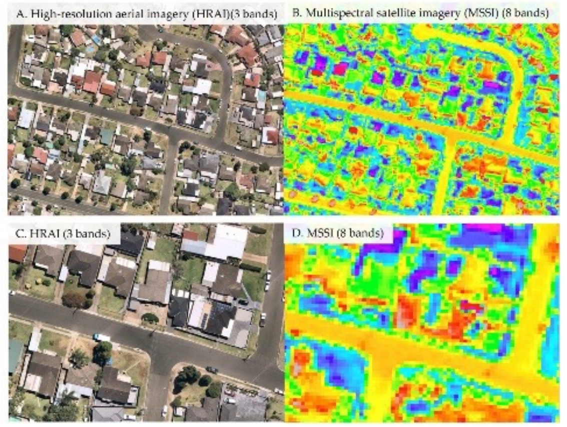
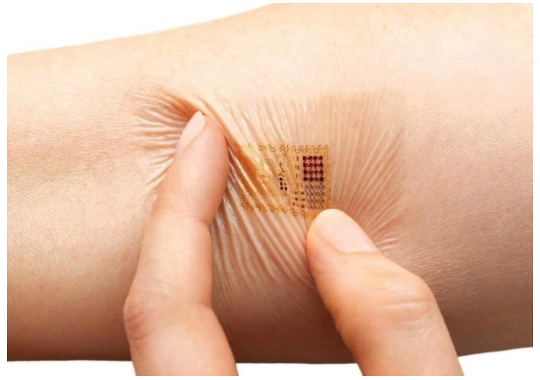
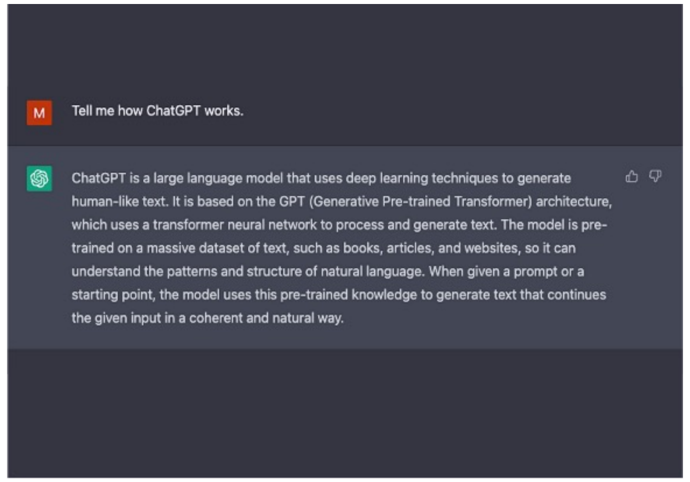

A. High-resolution aerial imagery (HRAI)(3 bands)
B. Multispectral satellite imagery (MSSI) (8 bands)
C. HRAI (3 bands)
D. MSSI (8 bands)

Image credits to Hikuwai et al

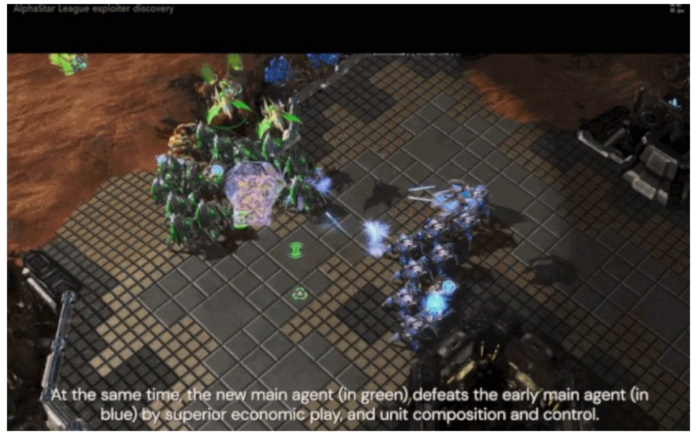




M Tell me how ChatGPT works.

ChatGPT is a large language model that uses deep learning techniques to generate human-like text. It is based on the GPT (Generative Pre-trained Transformer) architecture, which uses a transformer neural network to process and generate text. The model is pre-trained on a massive dataset of text, such as books, articles, and websites, so it can understand the patterns and structure of natural language. When given a prompt or a starting point, the model uses this pre-trained knowledge to generate text that continues the given input in a coherent and natural way.

Image credits
to Brother UK.



At the same time, the new main agent (in green) defeats the early main agent (in blue) by superior economic play, and unit composition and control.

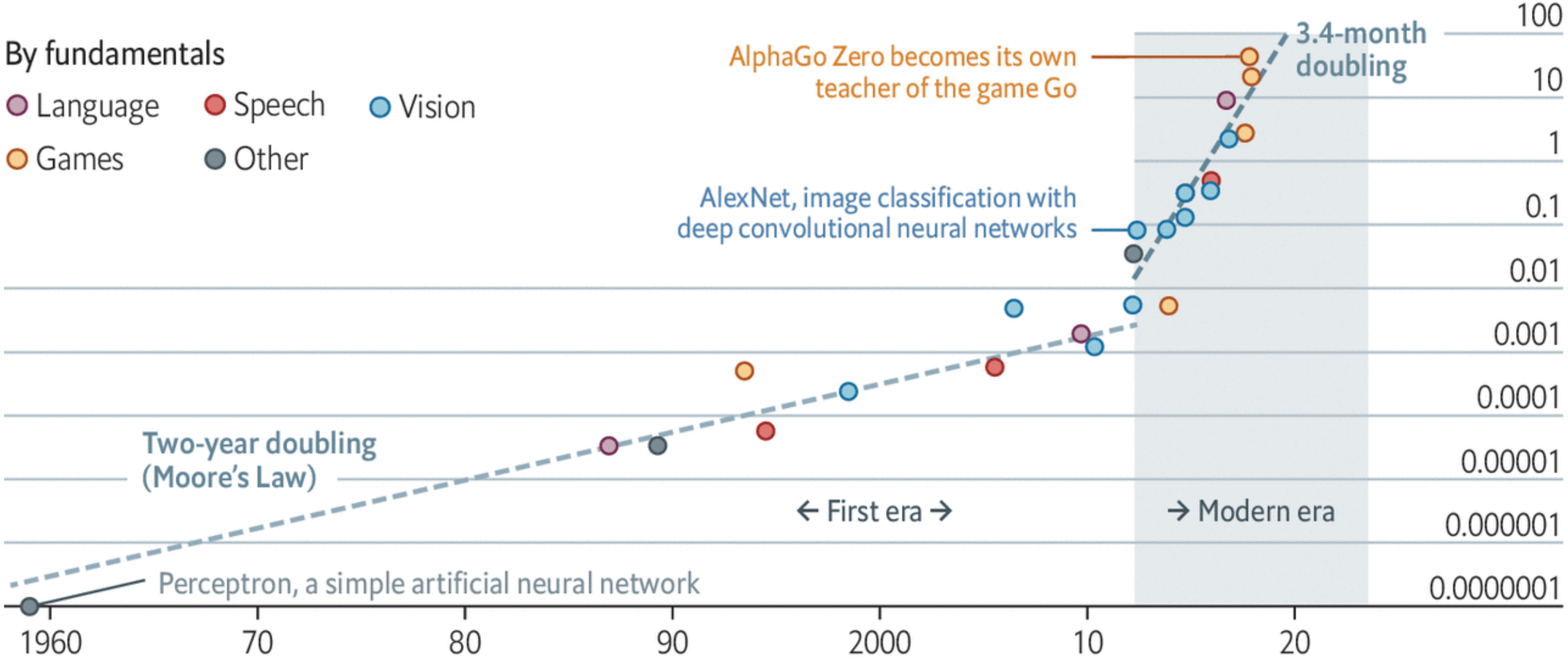
Source: DeepMind

Machine Learning is Expensive

Deep and steep

Computing power used in training AI systems

Days spent calculating at one petaflop per second*, log scale

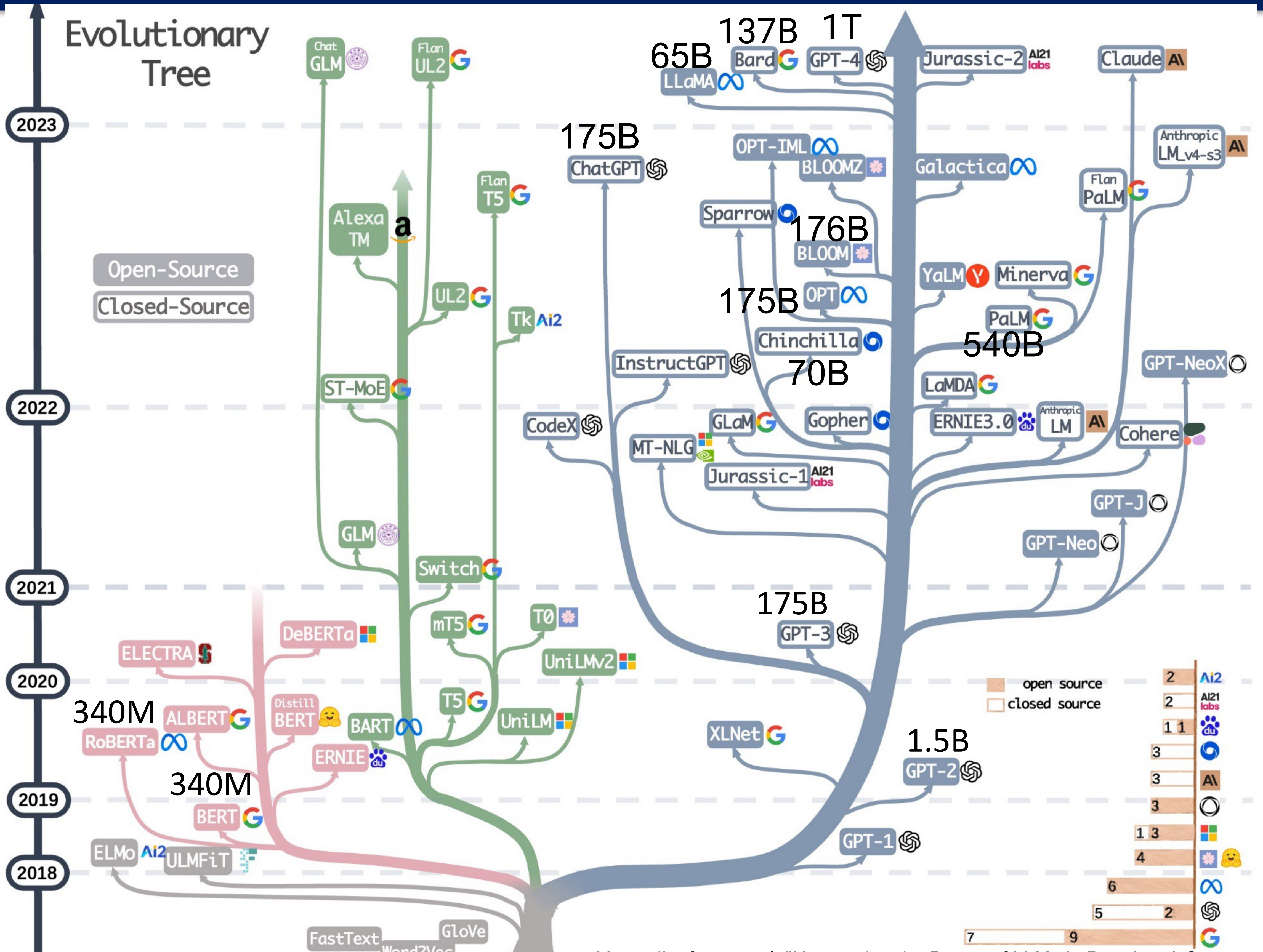


Source: OpenAI

*1 petaflop=10^{15} calculations

The Economist

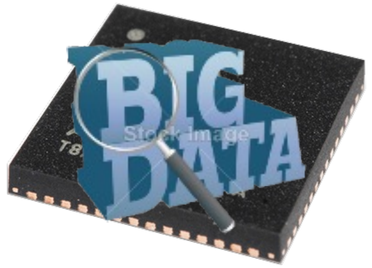
Language Models are Much Bigger (Billions of Parameters)

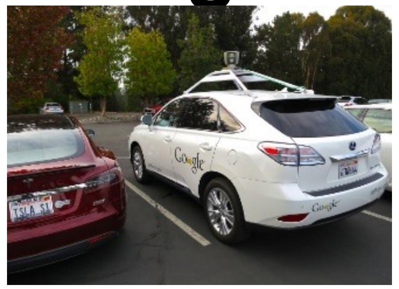


Yang, Jingfeng, et al. "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond."

Machine Learning Challenges

Challenges: Model Storage

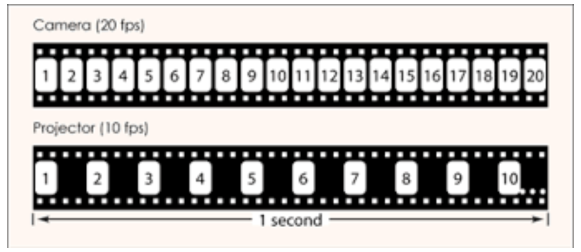




- Large model sizes → store large-scale, multi-modal deep learning systems.

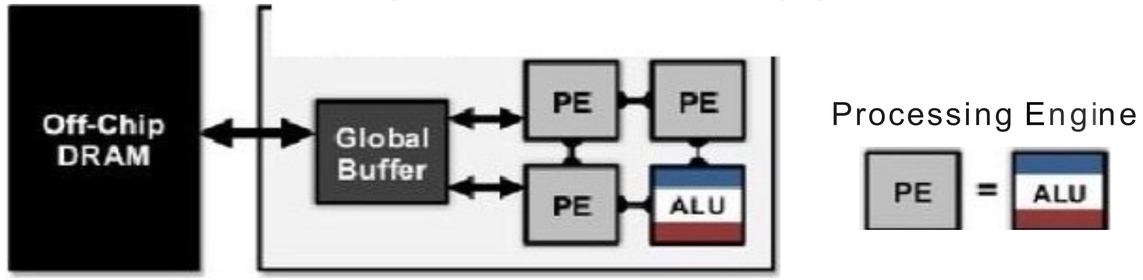
Challenges: Computation

- ➢ Increase computations, therefore increase training time.
- ➢ Increases throughput/latency
  - Frame rate, delay

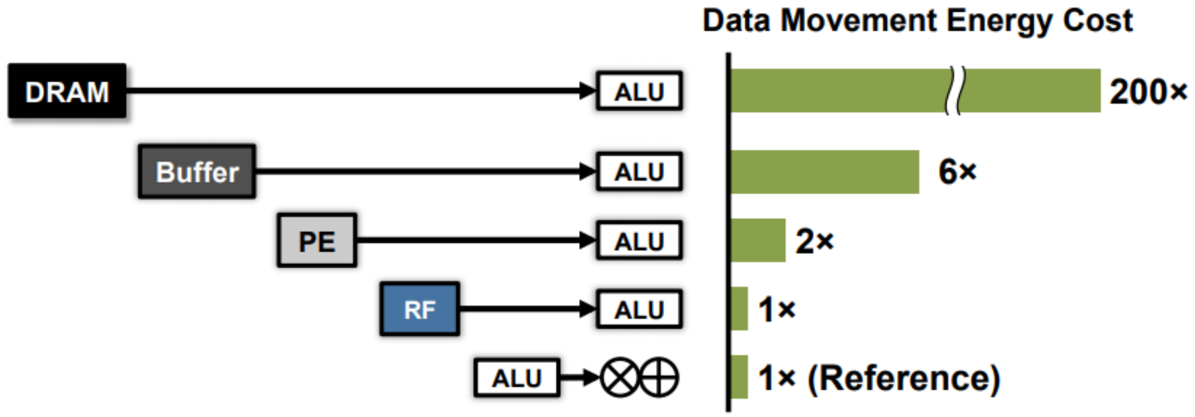


He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

Challenges: Energy



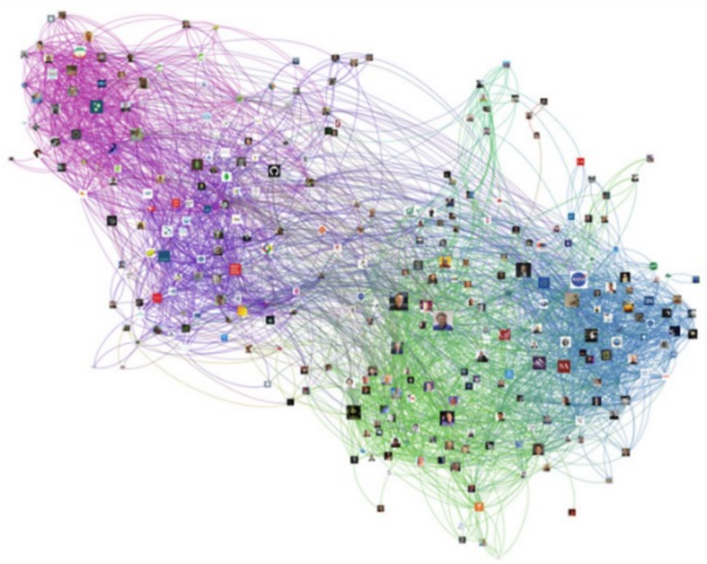
Von Neumann architecture

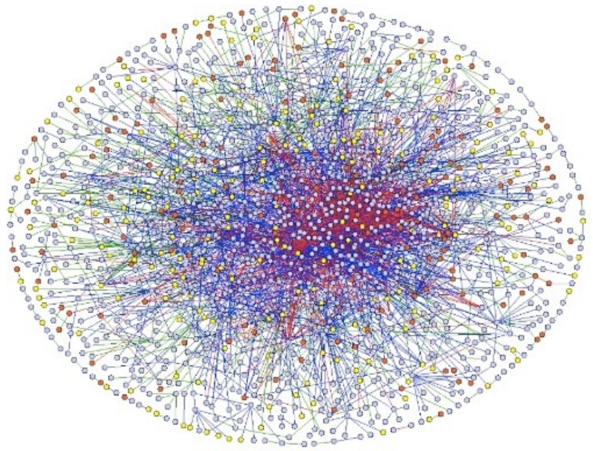


Slide courtesy: V. Sze, et.al., "Hardware for Machine Learning: Challenges and Opportunities", CICC' 17

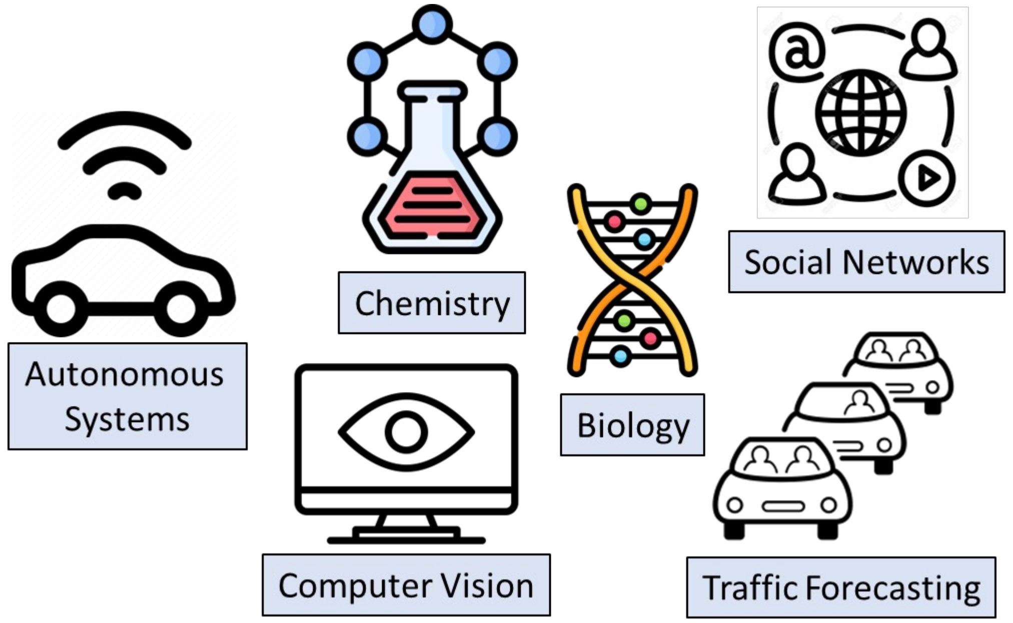
Challenges with Large and Sparse Graph Input

Growing massive real-world graphs = Larger GNNs

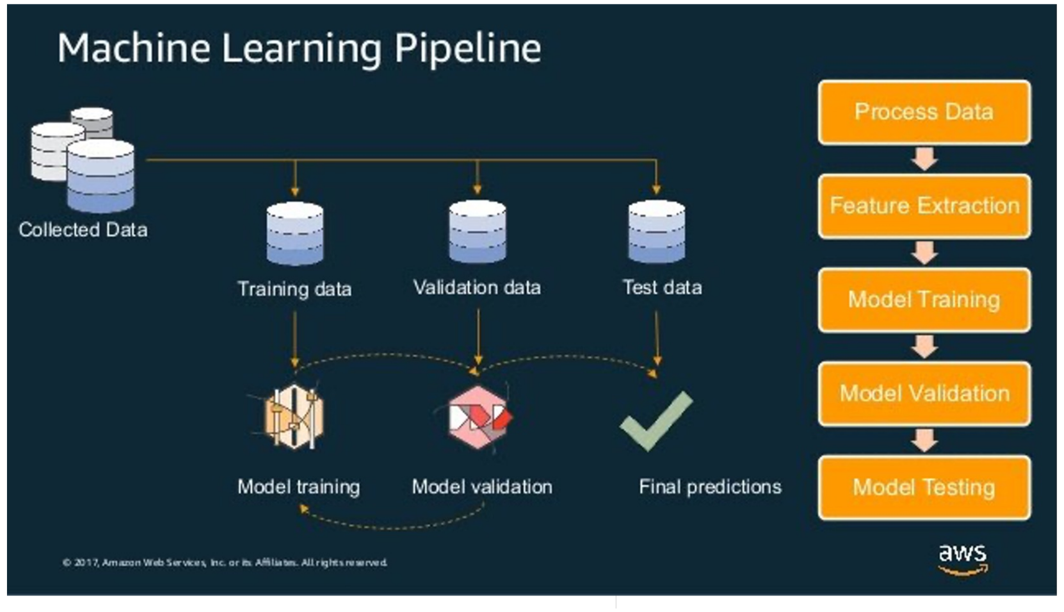




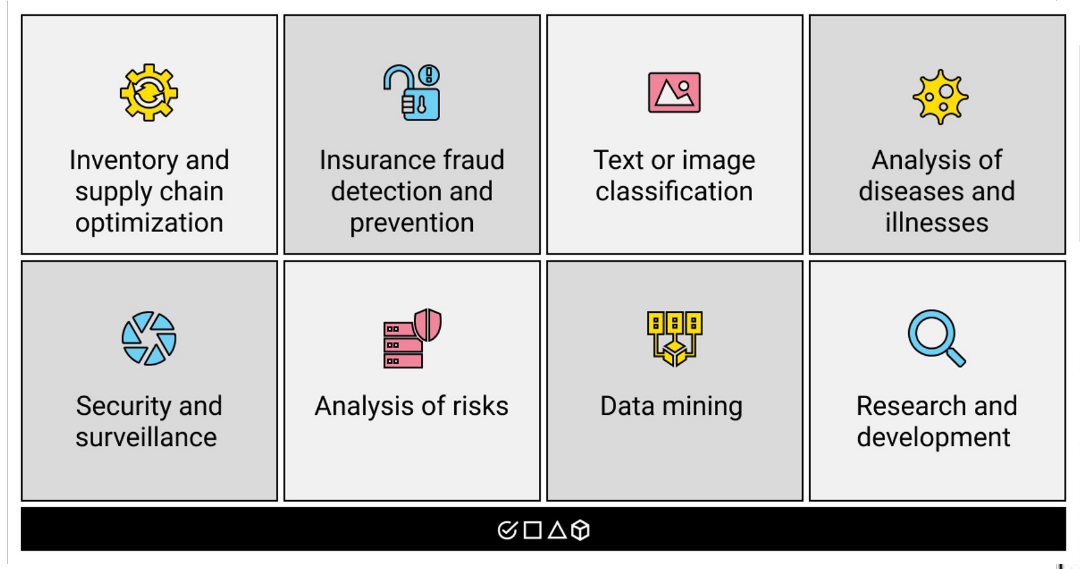
- Example: **492 million** nodes, **6.8 billion** edges in Alibaba’s AliGraph
- **Problem:** Increasing computational and memory cost, need for more high-end servers with expensive GPUs, increased training and inference time...



Machine Learning as a Service (MLaaS)

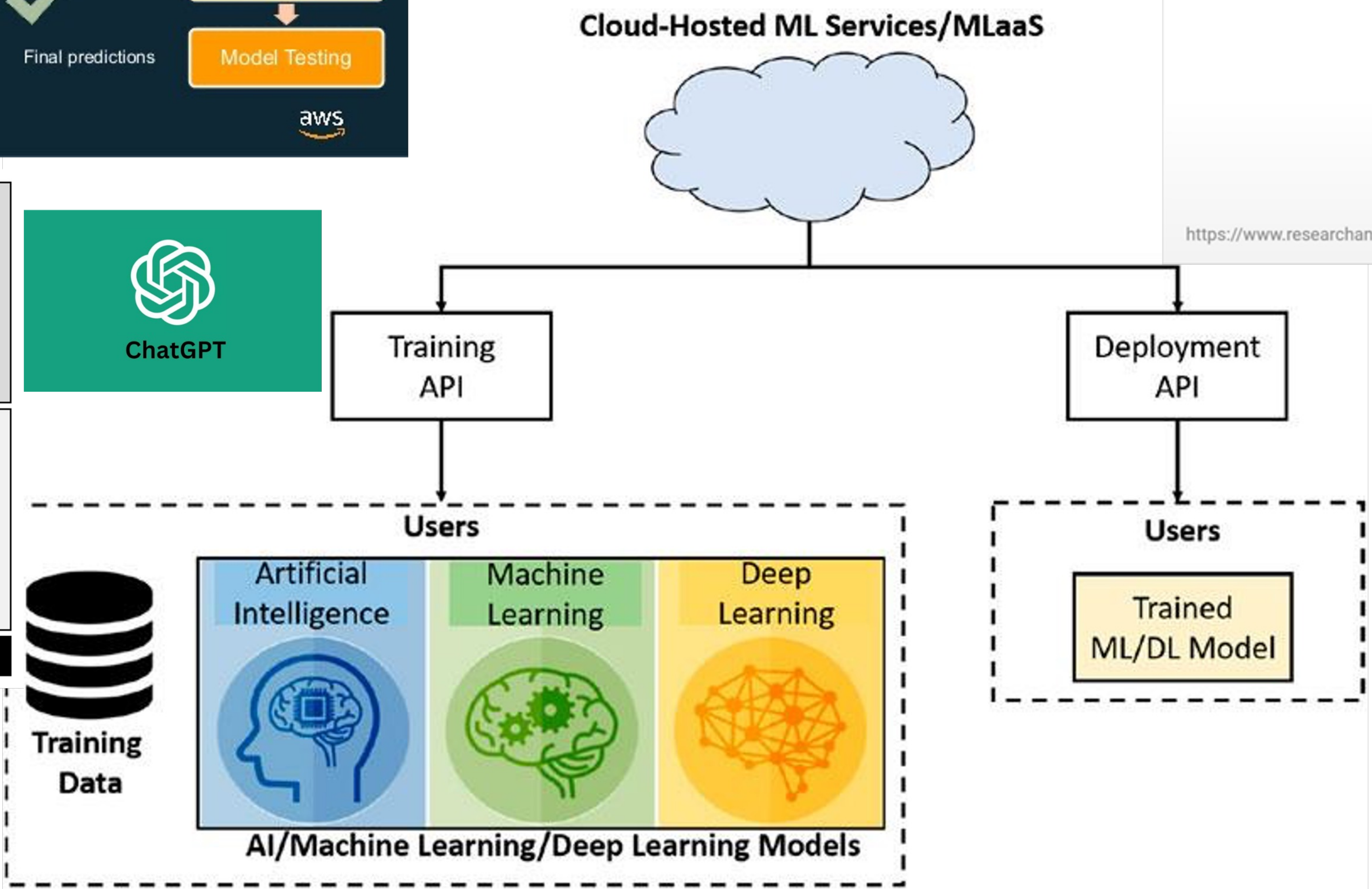

Machine Learning Pipeline

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

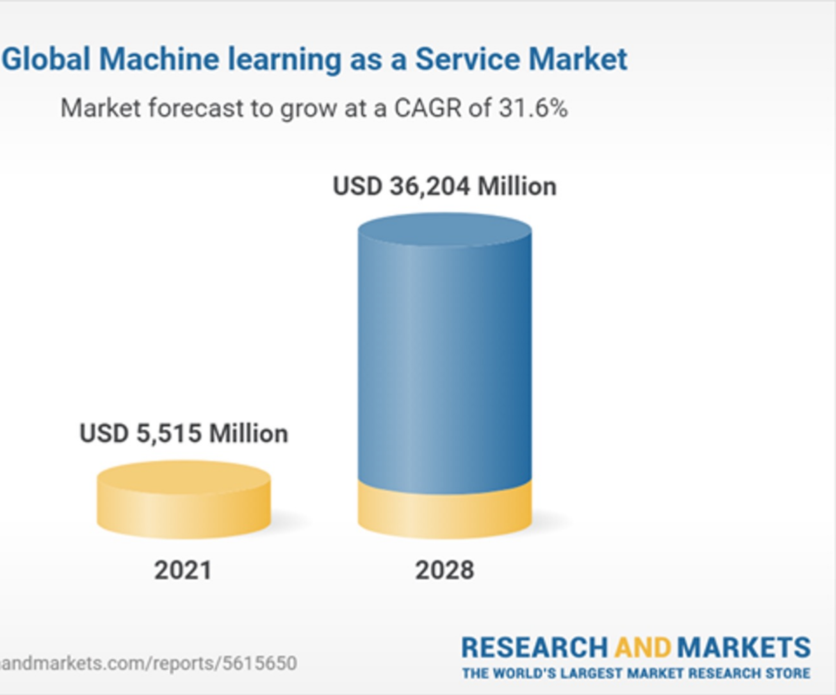


Inventory and supply chain optimization	Insurance fraud detection and prevention	Text or image classification	Analysis of diseases and illnesses
Security and surveillance	Analysis of risks	Data mining	Research and development

ChatGPT

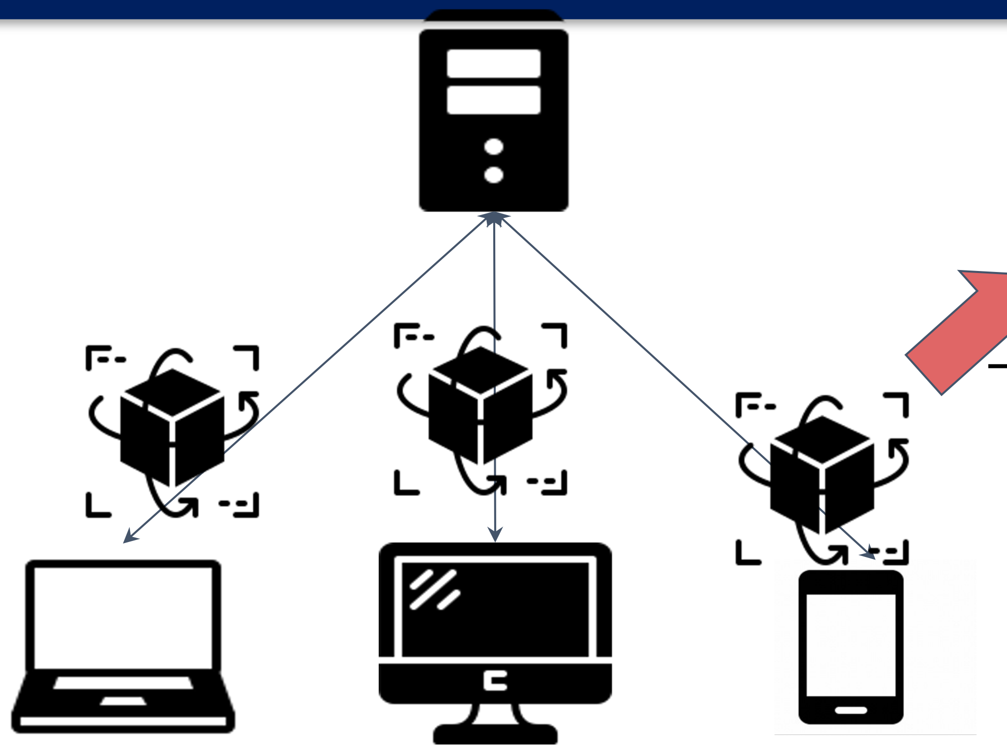


Qayyum et al. "Securing machine learning in the cloud: A systematic review of cloud machine learning security." *Frontiers in big Data* 3 (2020): 587139.

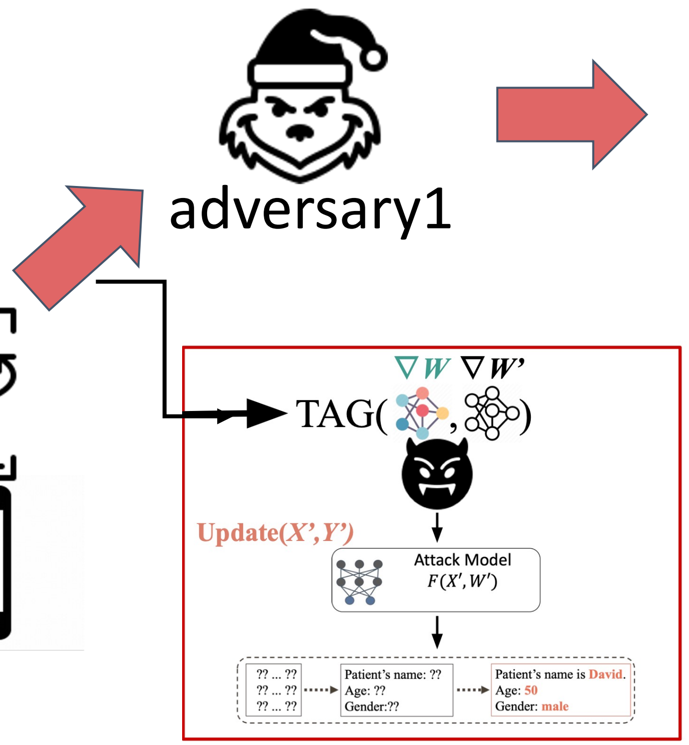


https://www.researchandmarkets.com/reports/5615650

More Challenges with Privacy Leakage



MLaaS (sensitive data such as medical and billing info).



TAG Adversary

Ground Truth: **[CLS] the sailors rode the breeze clear of the rocks .**

Dummy: ufo ↄ ##ub 999 12 hostages strictly ##ouse cool writing nonstop

(a). 5 iterations (Dummy data contains 0 tokens in Ground Truth)

Dummy: **rocks** . hydroelectric ari jamie cornerstone greenfield herrera **rocks** . cares **the**

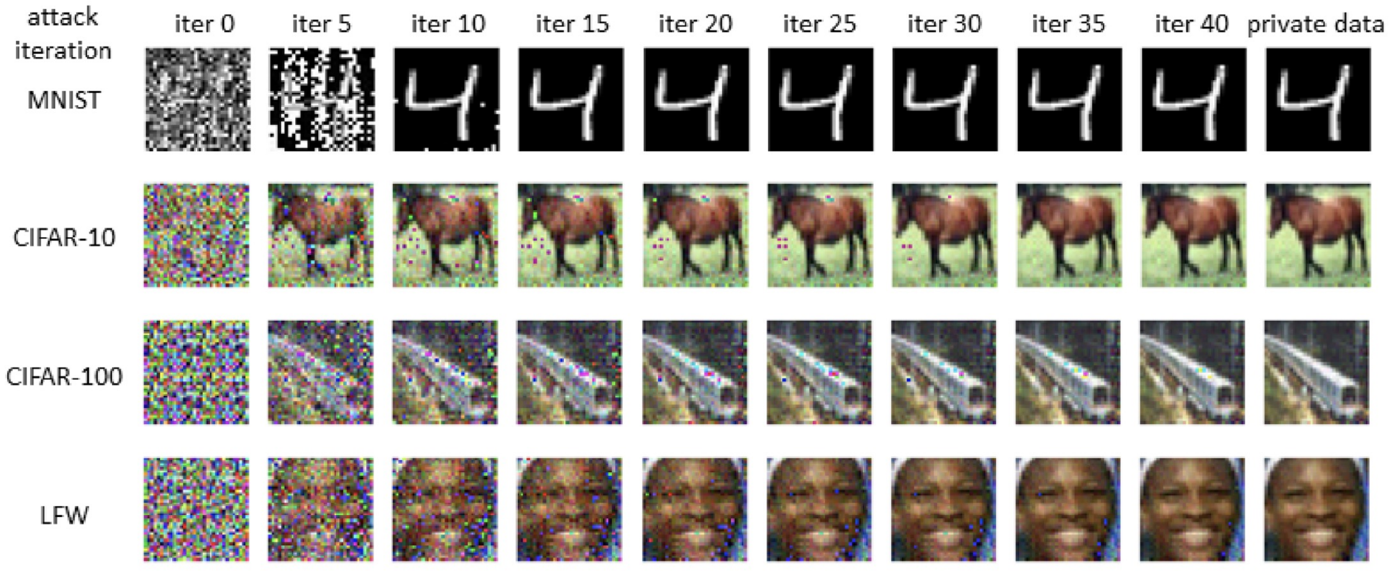
(b). 20 iterations (Dummy data contains 2 out of 9 tokens in Ground Truth)

Dummy: **rocks [CLS] . . . the rode breeze the . clear the**

(c). 50 iterations (Dummy data contains 7 out of 9 tokens in Ground Truth)

Dummy: **rocks [CLS] sailors . . the rode breeze the . clear the**

(d). 200 iterations (Dummy data contains 8 out of 9 tokens in Ground Truth)



Gradient Attack on Federated Learning[1,2, 3, 4]

- Data are distributed across devices
- Decentralized collaborative machine learning
- Prevent direct access to private data

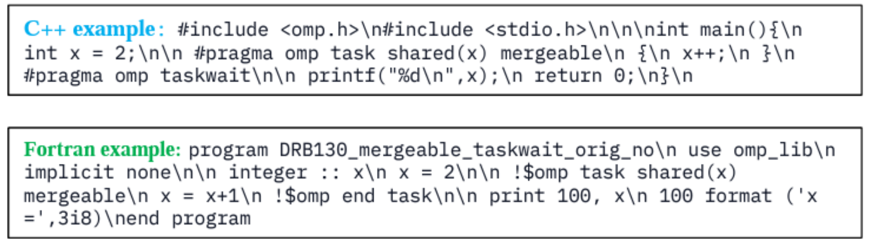
Algorithm 1 TAG

```
Input: $\nabla \mathbf{W}$: ground truth gradient; $\mathcal{F}(\mathbf{X}, \mathbf{W}')$: NLP model; $\eta$: learning rate; $\mathbf{W}'$: parameter weights
Initial: $\mathbf{X}' \sim \mathcal{N}(0, 1)$, $\mathbf{Y}' \sim \mathcal{N}(0, 1)$
for the i-th iteration do
    $\nabla \mathbf{W}'_i \leftarrow \partial \ell(\mathcal{F}(\mathbf{X}', \mathbf{W}')/\partial \mathbf{W}')$ //get dummy gradient by TAG
    $\mathcal{D}(\nabla \mathbf{W}, \nabla \mathbf{W}'_i) \leftarrow \|\nabla \mathbf{W}'_i - \nabla \mathbf{W}\|_2 + \alpha(\nabla \mathbf{W})\|\nabla \mathbf{W}'_i - \nabla \mathbf{W}\|$
    update $(\mathbf{X}', \mathbf{Y}')$:
    $\mathbf{X}' \leftarrow \mathbf{X}' - \eta \frac{\partial \mathcal{D}(\nabla \mathbf{W}, \nabla \mathbf{W}'_i)}{\partial \nabla \mathbf{X}'}$,
    $\mathbf{Y}' \leftarrow \mathbf{Y}' - \eta \frac{\partial \mathcal{D}(\nabla \mathbf{W}, \nabla \mathbf{W}'_i)}{\partial \nabla \mathbf{Y}'}$
end for
Output: Recovered Data $\mathbf{X}^*$,$\mathbf{Y}^*$
```

[1] J. Deng et al. EMNLP 2021
[2] W. Wei et al. ESORICS 2020
[3] L. Zhu et al. NeurIPS 2019
[4] Y. Wang et al. IJCNN 2022

Research Overview

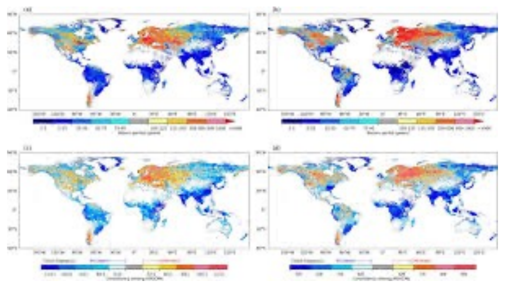
Images



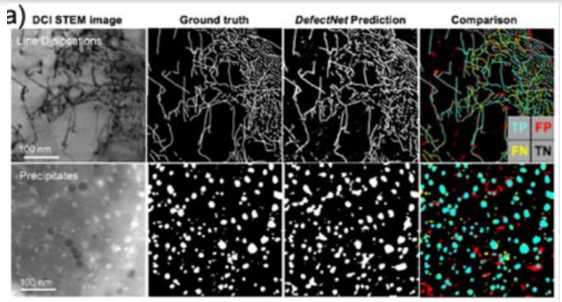
```
C++ example: #include <omp.h>\n#include <stdio.h>\n\n\nint main(){\n
int x = 2;\n\n #pragma omp task shared(x) mergeable\n {\n x++;\n }\n
#pragma omp taskwait\n\n printf("%d\n",x);\n return 0;\n}\n
```

```
Fortran example: program DRB130_mergeable_taskwait_orig_no\n use omp_lib\n
implicit none\n\n integer :: x\n x = 2\n\n !$omp task shared(x)
mergeable\n x = x+1\n !$omp end task\n\n print 100, x\n 100 format ('x
=',3i8)\nend program
```

HPC

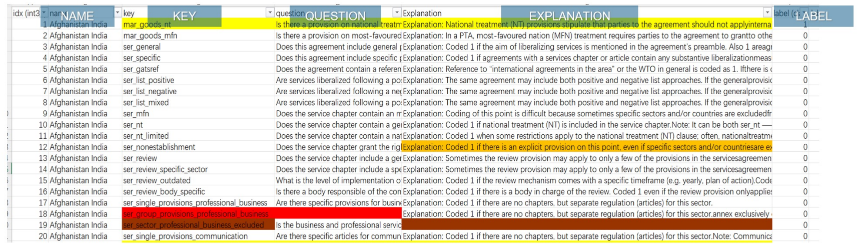


climate



Material science

…


NAME
KEY
QUESTION
EXPLANATION
LABEL

idx (int3	name	key	question	Explanation	label (c
1	Afghanistan India	mar_goods_nt	Is there a provision on national treatm	Explanation: National treatment (NT) provisions stipulate that parties to the agreement should not applyinterna	1
2	Afghanistan India	mar_goods_mfn	Is there a provision on most-favoured	Explanation: In a PTA, most-favoured nation (MFN) treatment requires parties to the agreement to grantto othe	0
3	Afghanistan India	ser_general	Does this agreement include general	Explanation: Coded 1 if the aim of liberalizing services is mentioned in the agreement's preamble. Also 1 areagı	0
4	Afghanistan India	ser_specific	Does this agreement include specific	Explanation: Coded 1 if agreements with a services chapter or article contain any substantive liberalizationmeas	0
5	Afghanistan India	ser_gatsref	Does the agreement contain a referen	Explanation: Reference to "international agreements in the area" or the WTO in general is coded as 1. Ifthere is	0
6	Afghanistan India	ser_list_positive	Are services liberalized following a po	Explanation: The same agreement may include both positive and negative list approaches. If the generalprovisic	0
7	Afghanistan India	ser_list_negative	Are services liberalized following a ne	Explanation: The same agreement may include both positive and negative list approaches. If the generalprovisic	0
8	Afghanistan India	ser_list_mixed	Are services liberalized following a po	Explanation: The same agreement may include both positive and negative list approaches. If the generalprovisic	0
9	Afghanistan India	ser_mfn	Does the service chapter contain an m	Explanation: Coding of this point is difficult because sometimes specific sectors and/or countries are excludedfr	0
10	Afghanistan India	ser_nt	Does the service chapter contain a ge	Explanation: Coded 1 if national treatment (NT) is included in the service chapter.Note: It can be both ser_nt —	0
11	Afghanistan India	ser_nt_limited	Does the service chapter contain a nat	Explanation: Coded 1 when some restrictions apply to the national treatment (NT) clause; often, nationaltreatme	0
12	Afghanistan India	ser_nonestablishment	Does the service chapter grant the rig	Explanation: Coded 1 if there is an explicit provision on this point, even if specific sectors and/or countriesare ex	0
13	Afghanistan India	ser_review	Does the service chapter include a ge	Explanation: Sometimes the review provision may apply to only a few of the provisions in the servicesagreemen	0
14	Afghanistan India	ser_review_specific_sector	Does the service chapter include a spe	Explanation: Sometimes the review provision may apply to only a few of the provisions in the servicesagreemen	0
15	Afghanistan India	ser_review_outdated	What is the level of implementation o	Explanation: Coded 1 if the review mechanism comes with a specific timeframe (e.g. yearly, plan of action).Code	0
16	Afghanistan India	ser_review_body_specific	Is there a body responsible of the con	Explanation: Coded 1 if there is a body in charge of the review. Coded 1 even if the review provision onlyapplies	0
17	Afghanistan India	ser_single_provisions_professional_business	Are there specific provisions for busin	Explanation: Coded 1 if there are no chapters, but separate regulation (articles) for this sector.	0
18	Afghanistan India	ser_group_provisions_professional_business		Explanation: Coded 1 if there are no chapters, but separate regulation (articles) for this sector.annex exclusively	0
19	Afghanistan India	ser_sector_professional_business_excluded	Is the business and professional servic		0
20	Afghanistan India	ser_single_provisions_communication	Are there specific articles for commun	Explanation: Coded 1 if there are no chapters, but separate regulation (articles) for this sector.Note: Communica	0

…

economy

Efficient Computing for ML system [Training and Inference]

Algorithm

- Hardware-aware pruning (SC-21, EMNLP-20, DAC-21, ISLPED-20, IJCAI-21, ACL-22)
- Knowledge Distillation (ACL-22, IJCAI-23)
- Fine-tuning free (IJCAI-21)
- Quantization (SC-23)
- Sparse training (ICCD-22, **DAC-23**)

System

- GPU kernel design (**ICCAD-23**)
- Run-time reconfigurable Inference (ICCAD-22, DAC-21, DATE-22, DAC-22)
- Platform (ReRAM, FPGA) (**ISCA-21**, DATE-21 (BPN), DAC-20)

Technology

- Optical Neural Network (DAC-23)

Efficient Computing for privacy preserving protocols

Algorithm

- Gradient attack (**EMNLP-21, IJCNN-22**)
- Membership Inference Attack (IJCAI-21)
- Secure federated learning (EMNLP-21 (*Oral*), Oakland-23)
- FHE-based PPML (ICML-23)
- MPC-based PPML

System

- MPC-based PPML (**DAC-23**, MICRO-23, ICCV-23)

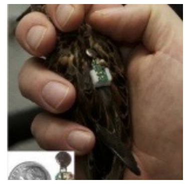
Hardware

GPU

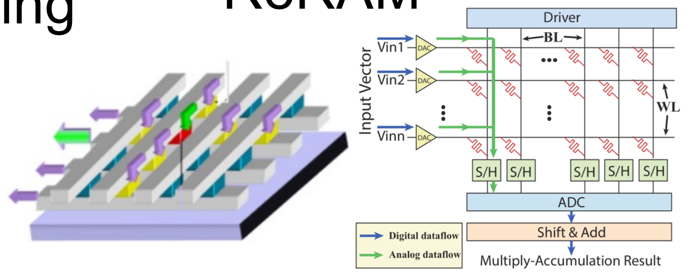
FPGA

Mobile Device

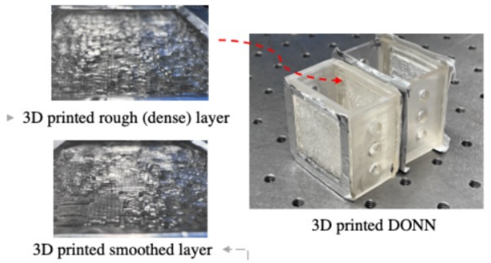
Energy Harvesting devices



ReRAM



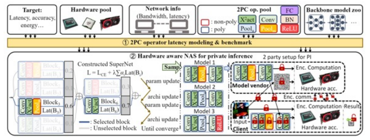
Optics



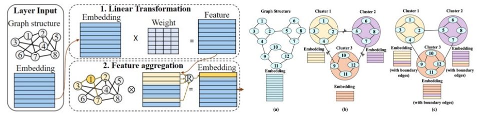
Source: Xu et al, HPCA 2015

Major Sponsors

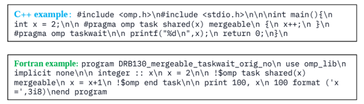
PI, Collaborative Research: SaTC: CORE: Medium: Accelerating Privacy-Preserving Machine Learning as a Service: From Algorithm to Hardware, NSF



Co-PI, GNN-based Hardware Trojan Detection for Large Complex Third-Party Ips, NSF IUCRC CHEST



PI, Feasibility of Transformer-based Code Migration for HPC, DOE/LLNL

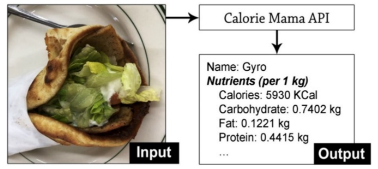


```
C++ example: #include <omp.h>\n#include <stdio.h>\n\n\nint main(){\n
int x = 2;\n\n #pragma omp task shared(x) mergeable\n {\n x++;\n }\n
#pragma omp taskwait\n\n printf("%d\n",x);\n return 0;\n}\n
```

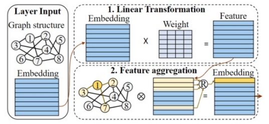
```
Fortran example: program DRB130_mergeable_taskwait_orig_no\n use omp_lib\n
implicit none\n\n integer :: x\n x = 2\n\n !$omp task shared(x)
mergeable\n x = x+1\n !$omp end task\n\n print 100, x\n 100 format ('x
=',3i8)\nend program
```

PI, Evaluating the Impact of Preferential Trade Agreements on Agricultural and Food Trade: New Insights from Natural Language Processing and Machine Learning, USDA NIFA

Co-PI, Developing a food image recognition technique to evaluate the nutrition information of restaurant foods and community food environment, USDA NIFA Hatch

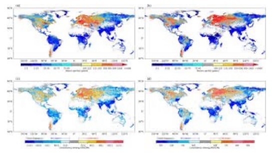


PI, Exploring Extreme Sparsity for GNNs to Achieve High Energy Efficiency in Large Core-Count Machines



PI, CLIMB: Connecticut's Low-carbon, Innovative, and Modernized electric grid for Better resilience

PI, Optigrid: Planning & Optimizing the Power Grid During the Low Carbon Transition in Connecticut



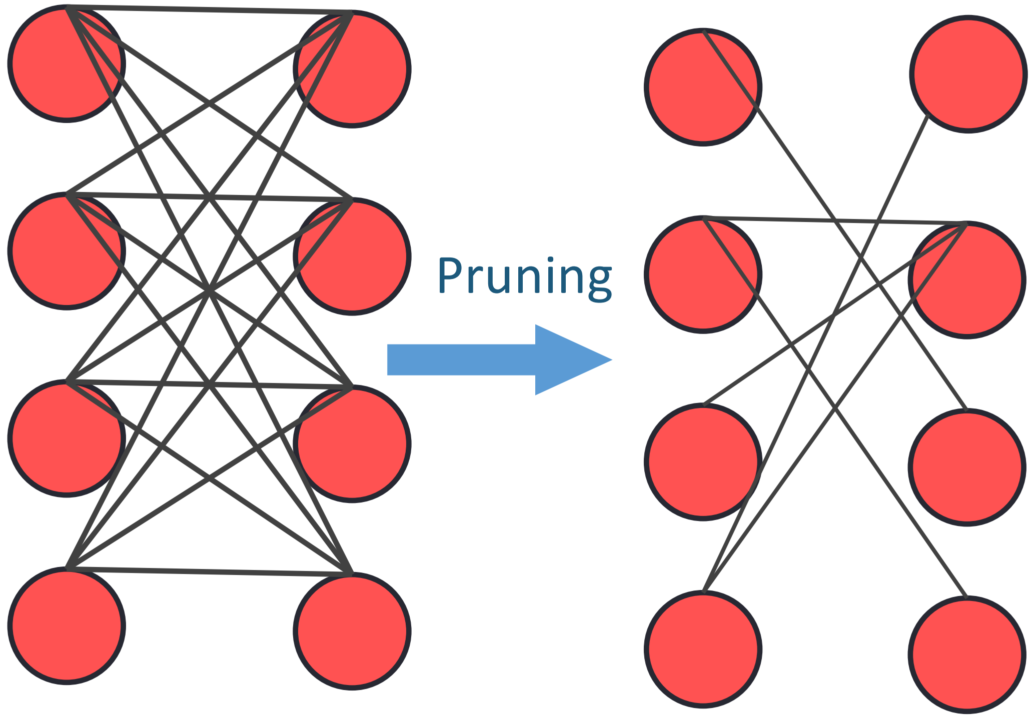
Co-PI, Change and Damage Detection from Aerial Images

Co-PI, Creating Insurance-Specific Transformers for Representation Learning from Large-scale Unstructured Claim Text, Travelers

Outstanding Student Paper Award @ 2023 HPEC. Bin Lei, Caiwen Ding (UConn), Le Chen, Pei-Hung Lin, Chunhua Liao (LLNL). *Creating a Dataset for High-Performance Computing Code Translation using LLMs: A Bridge Between OpenMP Fortran and C++.*

Sparsity types

Sparsity (e.g., pruning) makes the machine learning model small & fast



Dense

4	3	6	9
8	7	6	2
4	8	3	2
5	2	4	9

Pruning

Sparse

0	0	6	0
0	7	0	2
0	8	0	0
5	2	0	0

Irregular

0	0	6	0
0	7	0	2
0	8	0	0
5	2	0	0

Row

4	3	6	9
0	0	0	0
4	8	3	2
0	0	0	0

Column pruning

4	0	6	0
8	0	6	0
4	0	3	0
5	0	4	0

Tensor-tile Pruning

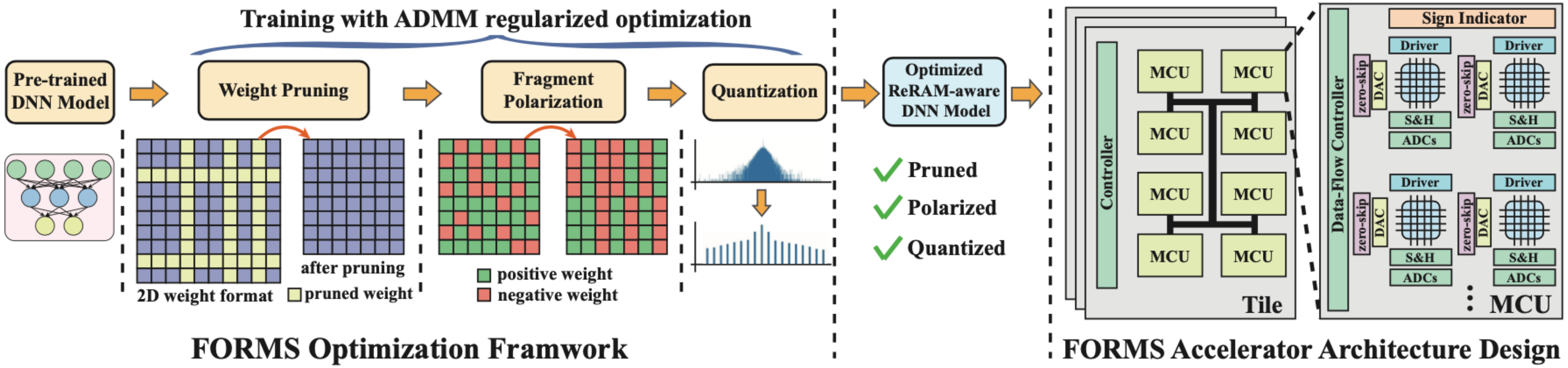
0	0	6	9
0	0	6	2
4	8	0	0
5	2	0	0

Or the combinations

FORMS: Fine-grained Polarized ReRAM-based In-situ Computation for Mixed-signal DNN Accelerator

Geng Yuan, Payman Behnam, Zhengang Li, Ali Shafiee, Xiaolong Ma, Hang Liu, Xuehai Qian, Mahdi Bojnordi, Yanzhi Wang, **Caiwen Ding**

(ISCA'21) The 48th International Symposium on Computer Architecture, 2021



FORMS: Fine-grained Polarized ReRAM-based DNN Accelerator

Challenges

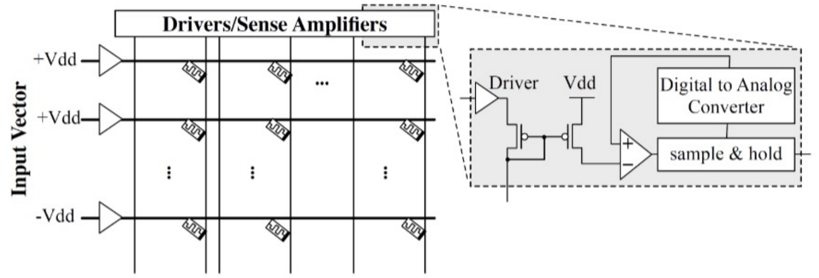


Figure 1: Illustrative example of the circuit for the RRAM crossbar arrays.

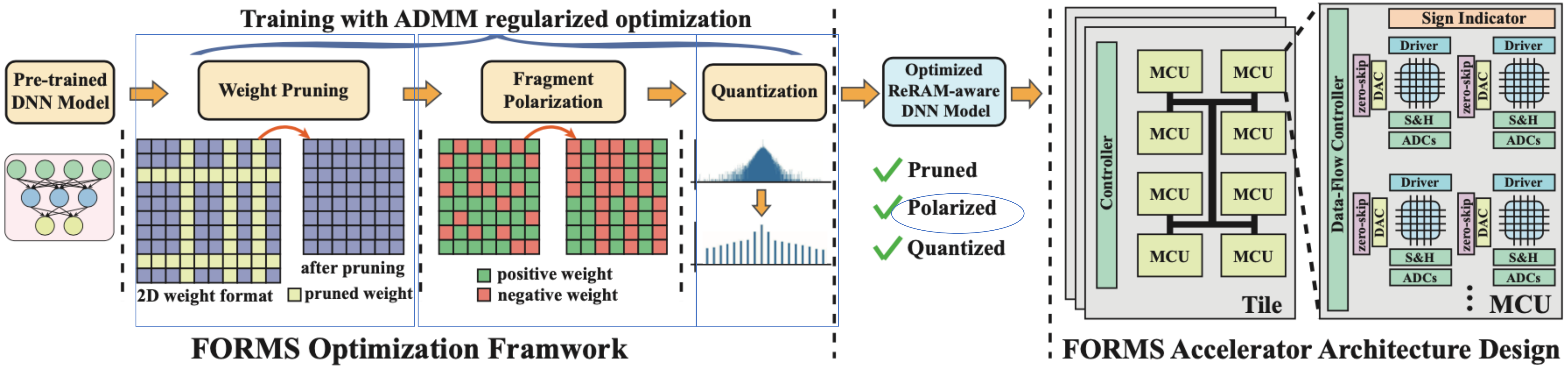
- Mapping signed weights
  - Decompose crossbars to positive and a negative ones (PRIME [1])
    - Significant crossbar overhead
  - Add an offset to the original negative weight values (ISAAC [2])
    - A bias must be subtracted from the results.
    - Count all 1s in MSB position (negative values) for all inputs and perform subtractions for each of 1s.
    - Peripheral circuit overhead
- Software optimization agnostic (mapping, pruning, quantization)
  - Not explore intrinsic sparsity of current DNNs
    - Hardware managed with remarkable overheads such as row indexing, routing controls, word-line controls, etc. (ReCom: DATE’18, SRE: ISCA’19)
    - highly model-dependent: (SNRram: DAC’18)

[1] PRIME, Chi et al. 2016 ISCA

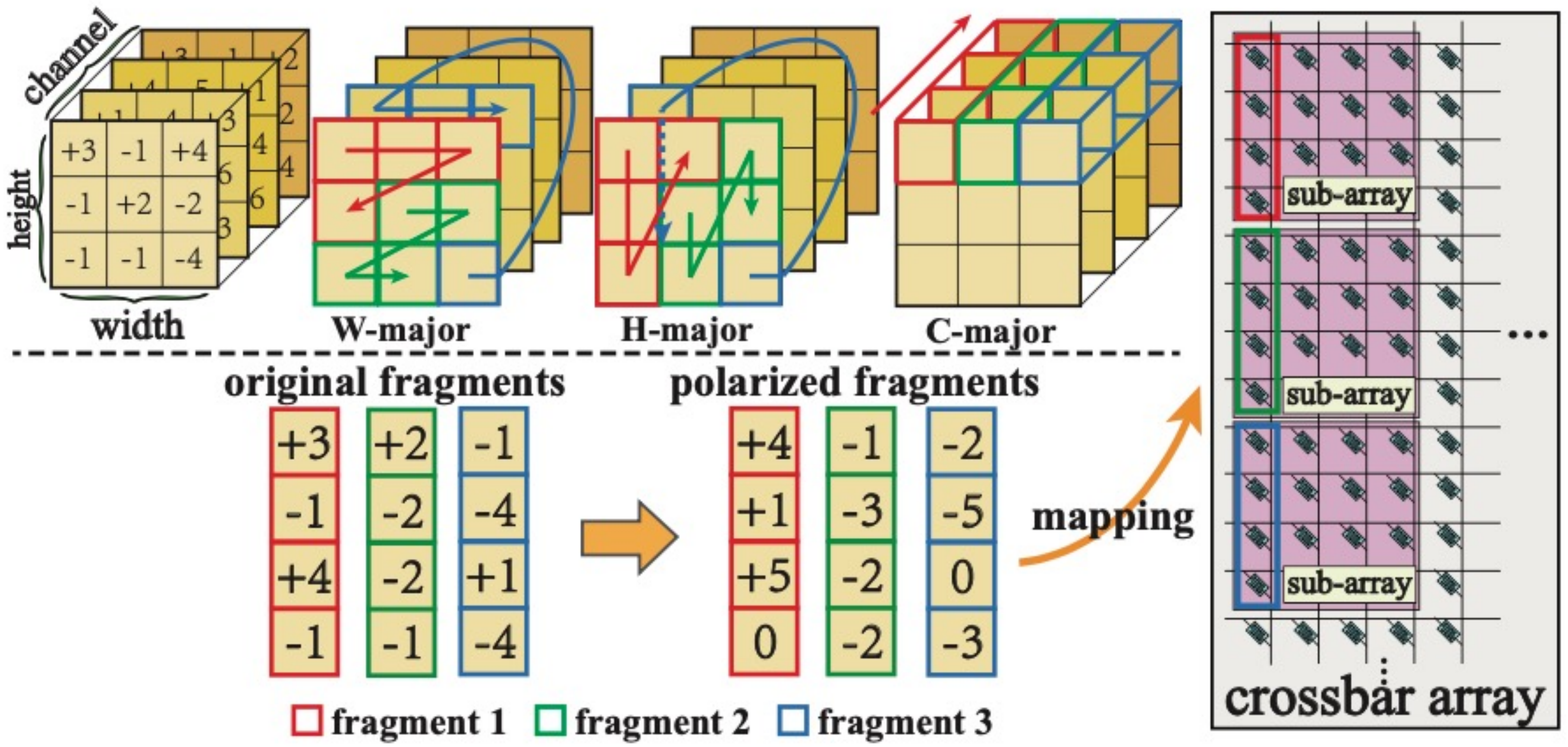
[2] ISAAC, Shafiee et al. 2016 ISCA

FORMS: Fine-grained Polarized ReRAM-based DNN Accelerator

- Our solution: FORMS -- algorithm/hardware co-design



- Two types of structured pruning methods
  - Filter pruning
  - Filter-shape pruning



- Weights in a fragment are polarized with the same sign.
  - Either positive or negative.
  - Sign bit saving.
- New Challenges:
  - How to determine the sign of weights in a fragment
  - The mapping policy of weights to sub-array columns

FORMS: Fine-grained Polarized ReRAM-based DNN Accelerator

ADMM-Regularized Optimization

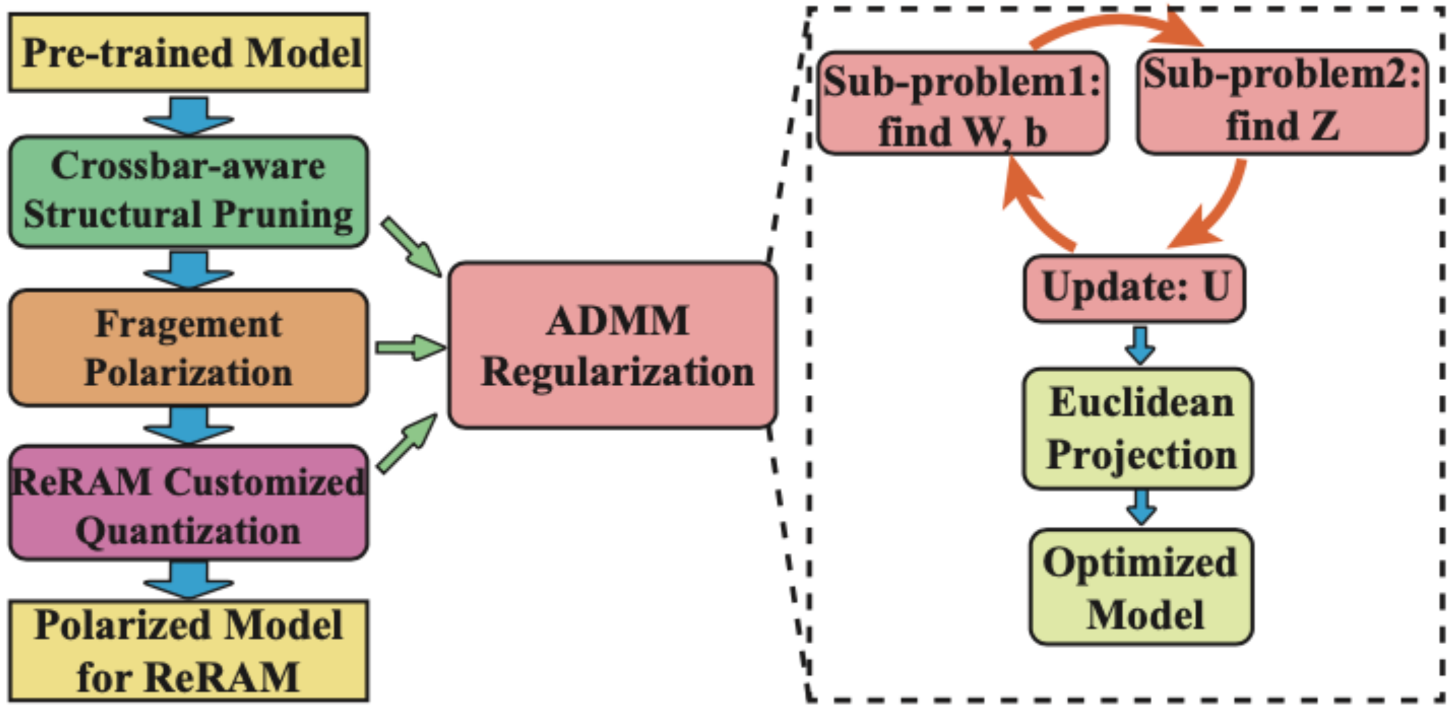
- To achieve optimized weight pruning, polarization, and quantization, FORMS utilizes Alternating Direction Method of Multipliers (ADMM) into the training process.
- ADMM regularization can reforge and separate the problem, then solve them iteratively.

$$\begin{aligned} \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad & \mathcal{L}(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N), \\ \text{subject to} \quad & \mathbf{W}_i \in \mathbf{S}_i,\ \mathbf{W}_i \in \mathbf{P}_i,\ \mathbf{W}_i \in \mathbf{Q}_i,\ i = 1, \dots, N, \end{aligned} \qquad (1)$$

$\mathbf{W}_i \in \mathbf{S}_i := \{\mathbf{H} \mid$ the percentage of nonzero *filters* and *filter-shapes* in $\mathbf{H}$ is less than or equal to α_i and β_i, where α_i and β_i are predefined hyperparameters. For example, suppose we want a 43% filter sparsity and 62% shape sparsity in i_{th} layer, then we set $\alpha_i = 0.57$ and $\beta_i = 0.38$.

P_i={the weights on each fragment(a column of a crossbar sub-array)

$$Sign_{\mathbf{f}} = \begin{cases} + & \text{if } \sum_{i=1}^{m}(W_i) \geq 0, \\ - & \text{otherwise}, \end{cases}$$



Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3(1) (2011) 1–122

FORMS: Fine-grained Polarized ReRAM-based DNN Accelerator

Formulation:

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f(\{\mathbf{W}_i\},\{\mathbf{b}_i\}) + \sum_{i=1}^{N} g_i(\mathbf{W}_i),$$

$$g_i(\mathbf{W}_i) = \begin{cases} 0 & \text{if } \mathbf{W}_i \in \mathbf{S}_i \ or \ \mathbf{P}_i \ or \ \mathbf{Q}_i, \\ +\infty & \text{otherwise.} \end{cases}$$

The original compression problem is not differentiable, thus not applicable through backpropagation

$\downarrow$

ADMM formulation

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f(\{\mathbf{W}_i\}_{i=1}^N,\{\mathbf{b}_i\}_{i=1}^N) + \sum_{i=1}^{N} g_i(\mathbf{Z}_i),$$
$$\text{subject to} \quad \mathbf{W}_i = \mathbf{Z}_i,\ i = 1,\ldots,N,$$

$\downarrow$

Augmented Lagrangian of ADMM formulation

Solving 2 sub-problems iteratively

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f(\{\mathbf{W}_i\}_{i=1}^N,\{\mathbf{b}_i\}_{i=1}^N)+\sum_{i=1}^{N}\frac{\rho_i}{2}\|\mathbf{W}_i-\mathbf{Z}_i^t+\mathbf{U}_i^t\|_F^2,$$

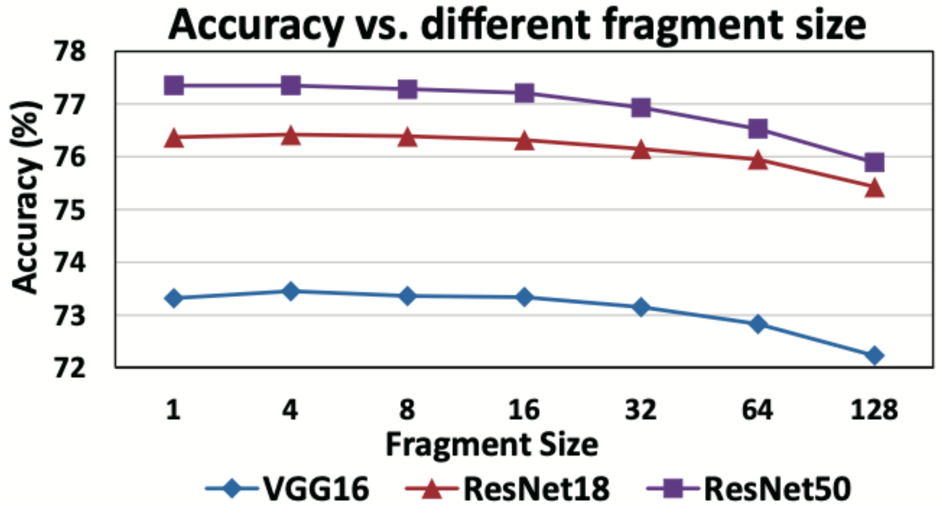
Stochastic Gradient Decent.

$$\underset{\{\mathbf{Z}_i\}}{\text{minimize}} \quad \sum_{i=1}^{N} g_i(\mathbf{Z}_i) + \sum_{i=1}^{N}\frac{\rho_i}{2}\|\mathbf{W}_i^{t+1} - \mathbf{Z}_i + \mathbf{U}_i^t\|_F^2,$$

Euclidean projection: $\mathbf{Z}_i^{t+1} = \prod_{\mathbf{X}_i}(\mathbf{W}_i^{t+1} + \mathbf{U}_i^t).$

Fragment size

- There is no accuracy drop with appropriate fragment size (e.g., 8, 16).
- Minor accuracy drop when fragment size is 32.

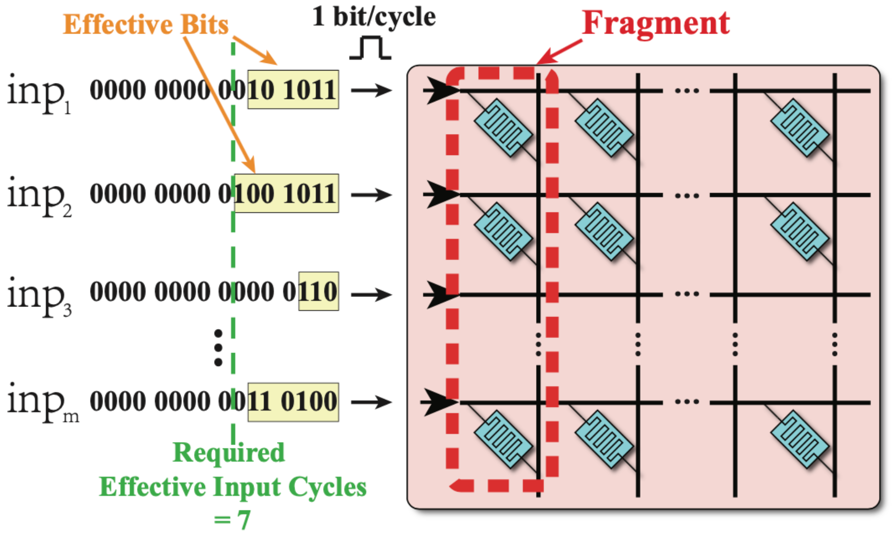


Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3(1) (2011) 1–122

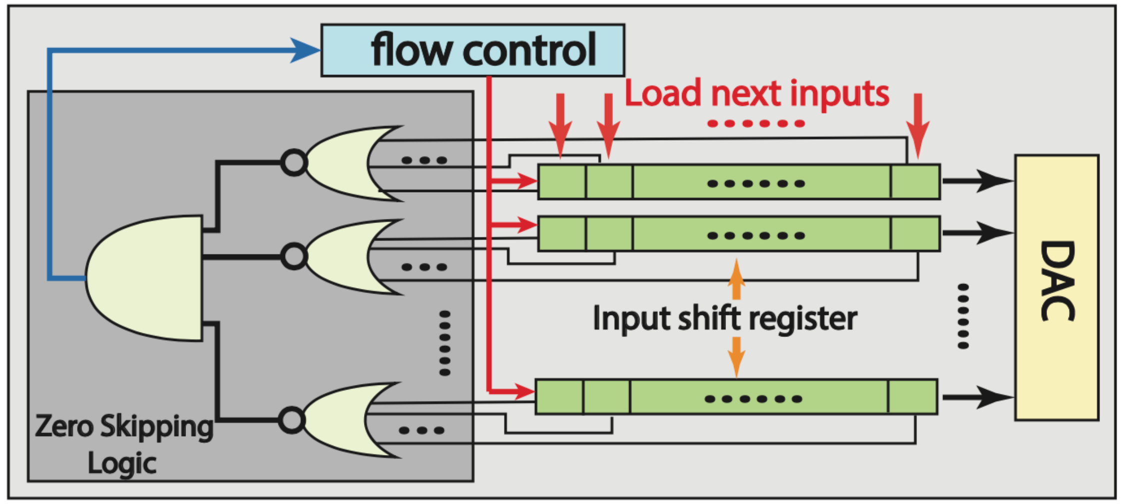
FORMS: Fine-grained Polarized ReRAM-based DNN Accelerator

- Larger fragments (e.g., ISAAC) have a higher probability to contain at least one input that its significant bits are not zero.
- Zero-skipping logic with negligible overhead in corporation with small fragment size can catch the intrinsic sparsity of DNN models.

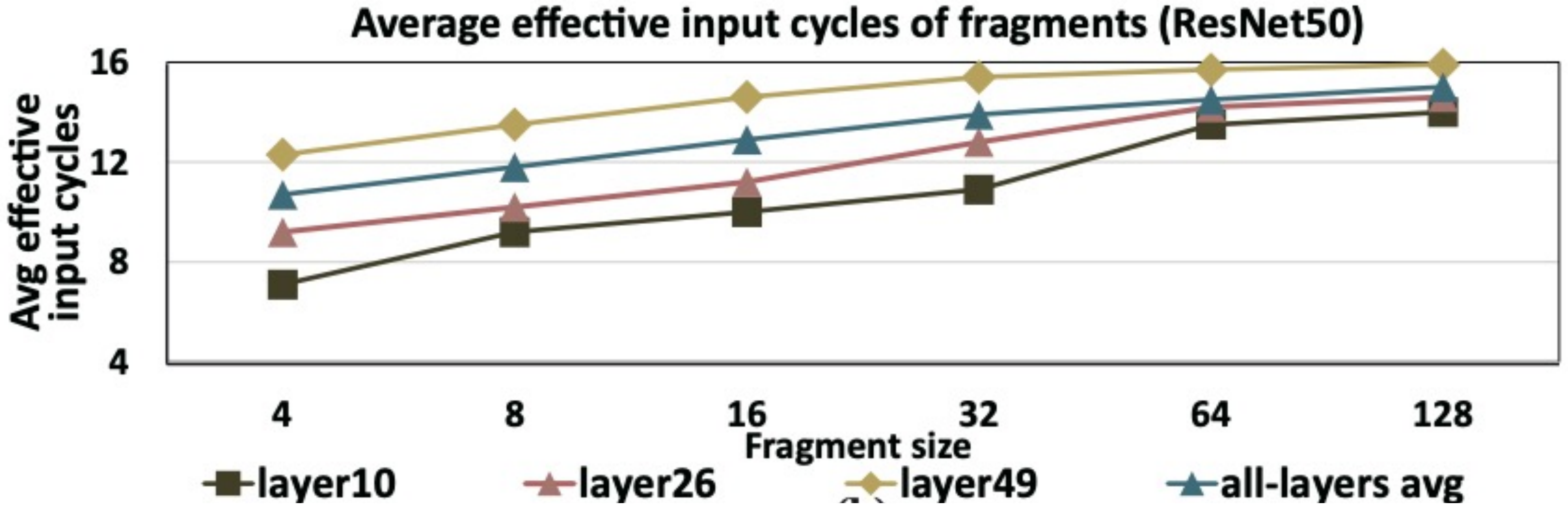
Zero-skipping Logic



Effective Bits
1 bit/cycle
Fragment
inp_1 0000 0000 0010 1011
inp_2 0000 0000 0100 1011
inp_3 0000 0000 0000 0110
inp_m 0000 0000 0011 0100
Required
Effective Input Cycles
= 7

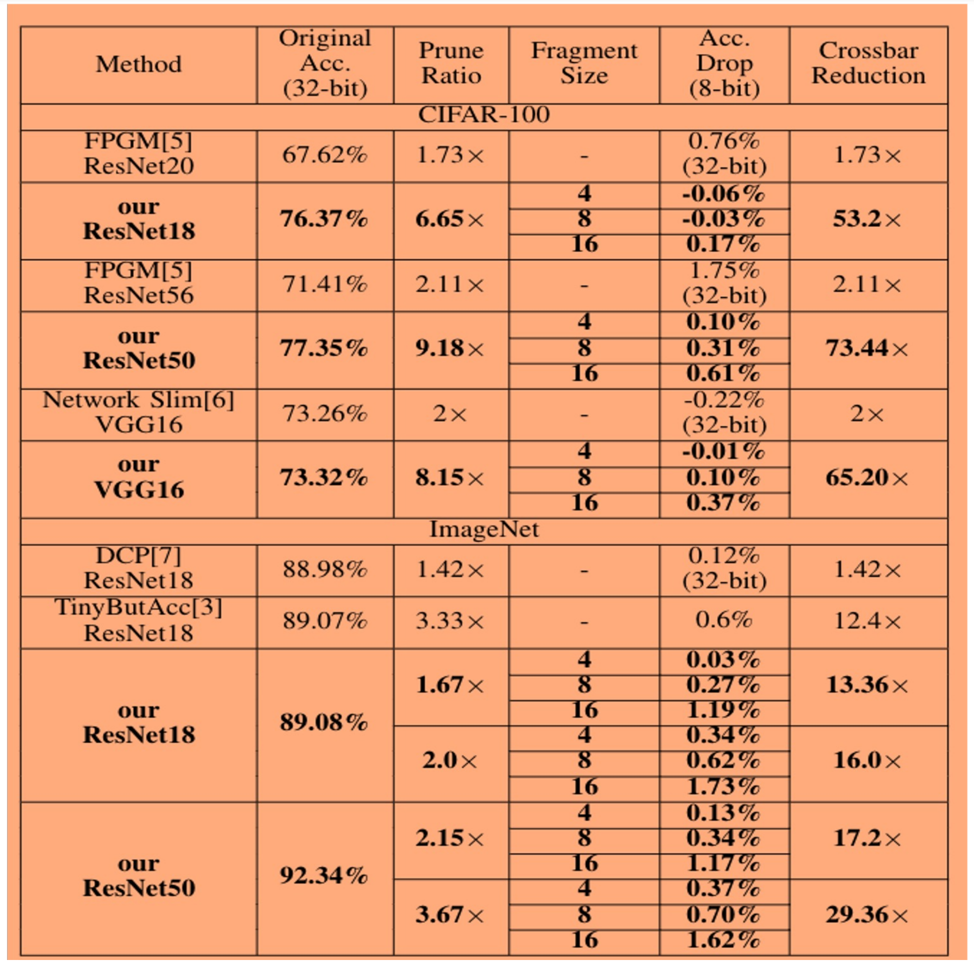


- Zero skipping saves requited feeding input cycles.
- Average effective input cycles for various fragment sizes of different layers of ResNet50 (CIFAR100).



FORMS: Fine-grained Polarized ReRAM-based DNN Accelerator

- **Training:** 8× NVIDIA Quadro RTX 6000 GPU by PyTorch API
- **Data Sets:** MNIST, CIFAR10, CIFAR100, ImageNet
- **DNN Models:** LeNet5, VGG16, ResNet18, ResNet50
- **Software Baselines:**
    - NeurIPS'15, ICCV'17, DAC'17, ECCV'18, NeurIPS'18, CVPR'19, ASPDAC'20
- **Design Space Exploration:** In-house tool while its back-end utilizes unified CACTI 7.0, NVSIM, and NVSIM-CAM, with multi-banking support
- **Hardware Simulator:** In-house tool while SW results are back annotated
- **Hardware Baselines:**
    - Digital: DaDianNao, TPU, WAX, SIMBA
    - Mixed-Signal: ISAAC, PUMA



Method	Original Acc. (32-bit)	Prune Ratio	Fragment Size	Acc. Drop (8-bit)	Crossbar Reduction
CIFAR-100					
FPGM[5] ResNet20	67.62%	1.73×	-	0.76% (32-bit)	1.73×
our ResNet18	**76.37%**	**6.65×**	**4**	**-0.06%**	**53.2×**
			8	**-0.03%**	
			16	**0.17%**	
FPGM[5] ResNet56	71.41%	2.11×	-	1.75% (32-bit)	2.11×
our ResNet50	**77.35%**	**9.18×**	**4**	**0.10%**	**73.44×**
			8	**0.31%**	
			16	**0.61%**	
Network Slim[6] VGG16	73.26%	2×	-	-0.22% (32-bit)	2×
our VGG16	**73.32%**	**8.15×**	**4**	**-0.01%**	**65.20×**
			8	**0.10%**	
			16	**0.37%**	
ImageNet					
DCP[7] ResNet18	88.98%	1.42×	-	0.12% (32-bit)	1.42×
TinyButAcc[3] ResNet18	89.07%	3.33×	-	0.6%	12.4×
our ResNet18	**89.08%**	**1.67×**	**4**	**0.03%**	**13.36×**
			8	**0.27%**	
			16	**1.19%**	
		2.0×	**4**	**0.34%**	**16.0×**
			8	**0.62%**	
			16	**1.73%**	
our ResNet50	**92.34%**	**2.15×**	**4**	**0.13%**	**17.2×**
			8	**0.34%**	
			16	**1.17%**	
		3.67×	**4**	**0.37%**	**29.36×**
			8	**0.70%**	
			16	**1.62%**	

CIFAR100: 53×, 73×, and 65× crossbar reduction on ResNet-18, ResNet-50, and VGG-16, respectively.

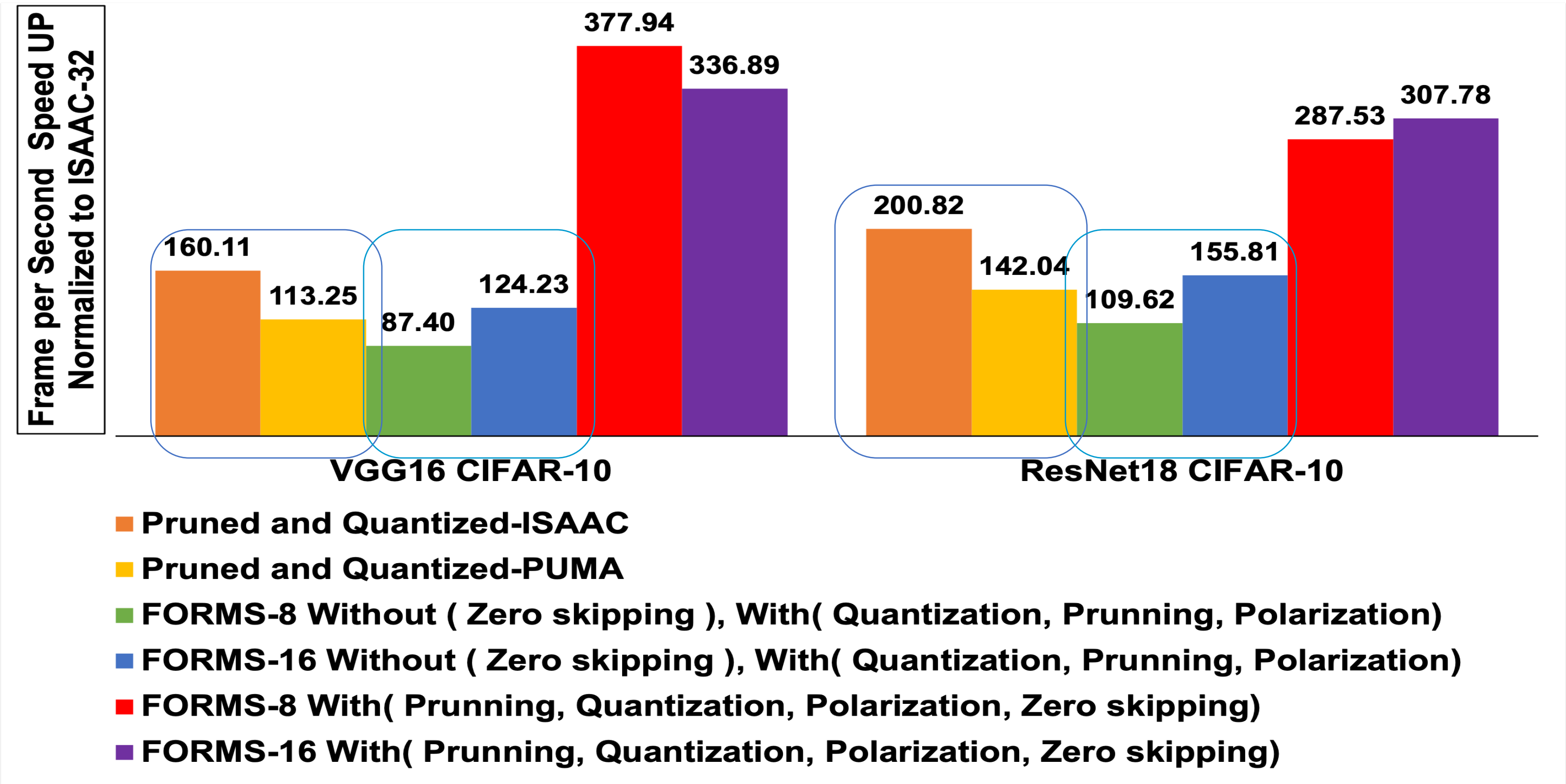
ImageNet: higher crossbar reduction (13× and 16×) with higher or similar accuracy when fragment size is 4 and 8.

[1]Y. Wang et al., Group scissor, DAC'17
[2] S. Han et al., IterativePrune, NeurIPS'15
[3] X. MA et al., TinyBytAcc, ASPDAC'20
[4] Y. He et al., AMC, ECCV'18
[8] S. Sayyaparaju et al., GLVLSI'17

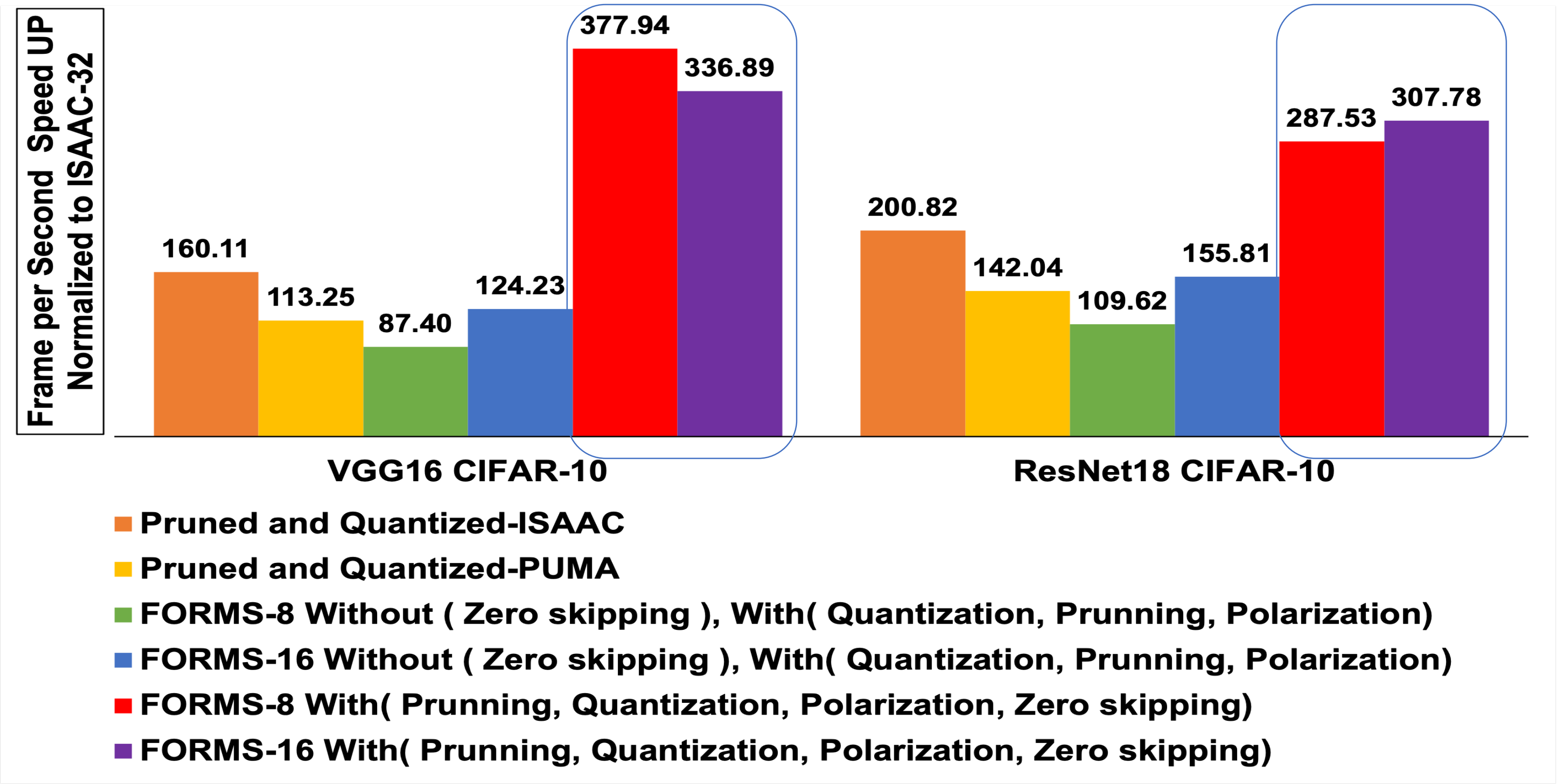
[5]Y. He et al., FPGM, CVPR'19
[6] Z. Liu et al., NetworkSlim, ICCV'17
[7] Z. Zhuang et al., DCP, NeurIPS'18

FORMS: Fine-grained Polarized ReRAM-based DNN Accelerator

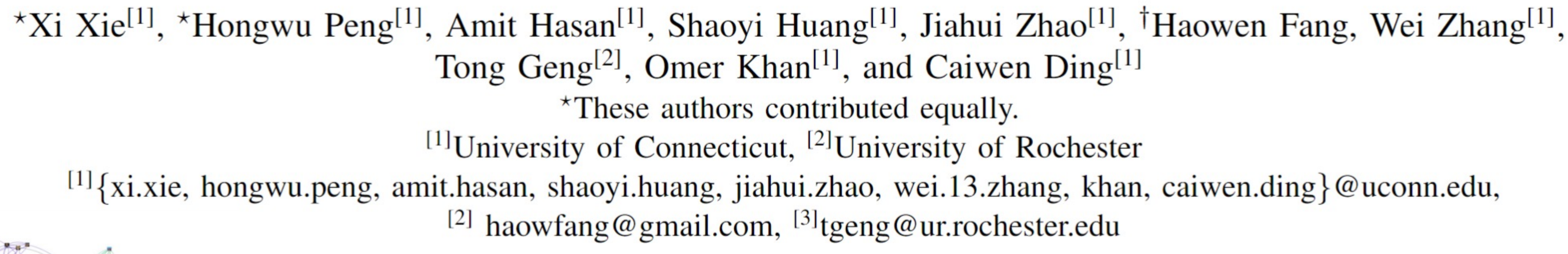
- Applying pruning and quantization will speed up the frame processing rate of ISAAC by 160× and 200× for VGG16 and ResNet18.
- Applying pruning and quantization increases FORMS speed up, up to 109× and155× when the fragment sizes are 8 and16, respectively.

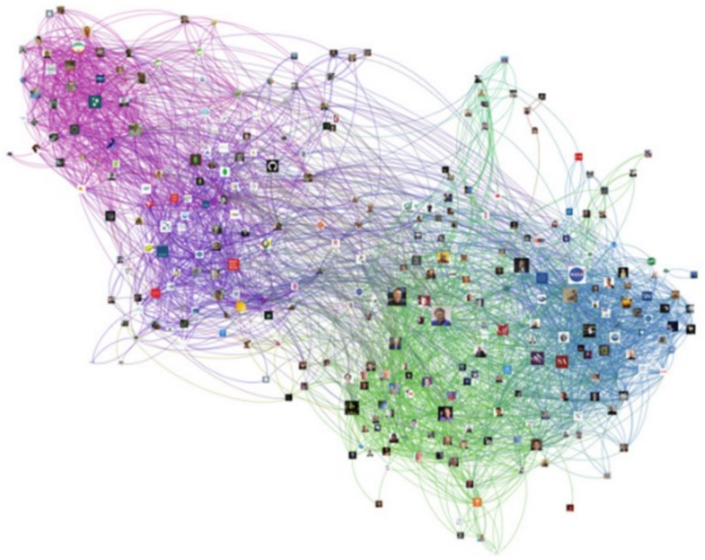


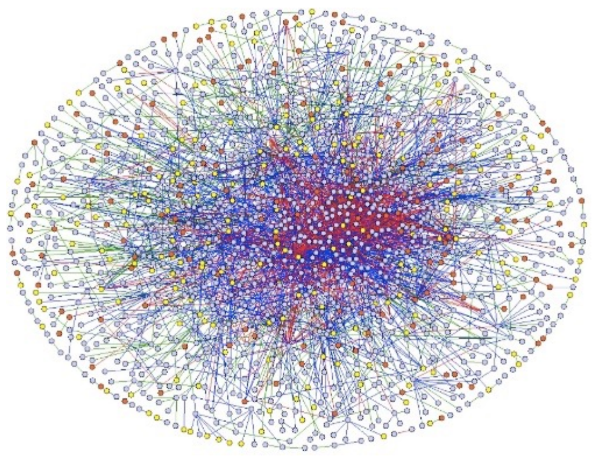
- By applying zero-skipping on top of model optimizations, the speed-up of FORMS goes up to 377 × and 366 × when fragment sizes are 8 and 16, respectively.

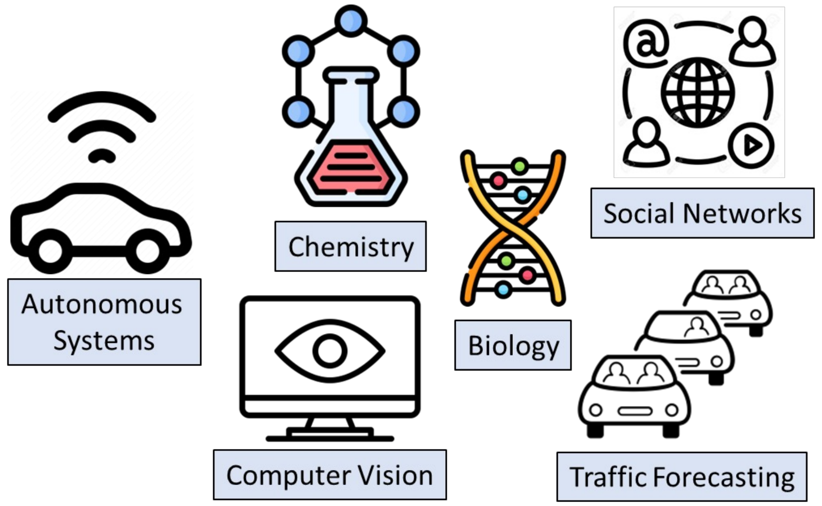


Accel-GCN: High-Performance GPU Accelerator Design for Graph Convolution Networks


*Xi Xie[1], *Hongwu Peng[1], Amit Hasan[1], Shaoyi Huang[1], Jiahui Zhao[1], †Haowen Fang, Wei Zhang[1],
Tong Geng[2], Omer Khan[1], and Caiwen Ding[1]
*These authors contributed equally.
[1]University of Connecticut, [2]University of Rochester
[1]{xi.xie, hongwu.peng, amit.hasan, shaoyi.huang, jiahui.zhao, wei.13.zhang, khan, caiwen.ding}@uconn.edu,
[2] haowfang@gmail.com, [3]tgeng@ur.rochester.edu







UCONN
UNIVERSITY OF CONNECTICUT

UNIVERSITY of ROCHESTER

Graph Datasets Features

Accel-GCN's Benchmark Graph Datasets Details

Graph Name	# Nodes	#Edges	Sparsity (%)	Graph Name	# Nodes	#Edges	Sparsity (%)	Graph Name	# Nodes	#Edges	Sparsity (%)
am	881,680	5,668,682	99.9993	amazon0601	403,394	5,478,357	99.9966	Artist	50,515	1,638,396	99.9358
Arxiv	169,343	1,166,243	99.9959	Citation	2,927,963	30,387,995	99.9996	Collab	235,868	2,358,104	99.9958
com-amazon	334,863	1,851,744	99.9983	OVCAR-8H	1,889,542	3,946,402	99.9999	PRODUCTS	2,449,029	123,718,280	99.9979
Pubmed	19,717	99,203	99.9745	PPA	576,289	42,463,862	99.9872	Reddit	232,965	114,615,891	99.7888
SW-620H	1,888,584	3,944,206	99.9999	TWITTER-Partial	580,768	1,435,116	99.9996	wikikg2	2,500,604	16,109,182	99.9997
Yelp	716,847	13,954,819	99.9973	Yeast	1,710,902	3,636,546	99.9999	youtube	1,138,499	5,980,886	99.9995

- **High Memory and Computation Costs:**
  - Large graphs demand substantial memory storage and computational resources
  - Optimizing algorithms and utilizing parallel processing can help manage the increased overheads
- **High Sparsity:**
  - The high sparsity; many nodes have few connections
  - Techniques tailored for sparse graphs can be beneficial in this regard.

SPMM Background

- Basic SPMM approaches[1]:

(a) Inner product (b) Outer product (c) Row-wise product (d) Column-wise product

- Row-wise SPMM is well-suited for GPUs for several reasons:
  - Coalesced Memory Access
  - Optimized Data Locality and Cache Usage
  - Balanced Workload
  - Streamlined Control Flow

[1] Srivastava, N., Jin, H., Liu, J., Albonesi, D., and Zhang, Z. (2020). "Matraptor: A sparse-sparse matrix multiplication accelerator based on row-wise product," 2020 MICRO.

State of the Arts

- **PyTorch-Geometric (PyG)[1]** utilizes the torch-scatter library, which is built on CUDA, to perform graph aggregation operations central to its function. This library aids in the support of node embedding propagation, following the essential principles of graph-processing systems.
- **Deep Graph Library (DGL)[2]** incorporates a ready-to-use Sparse-Matrix Multiplication (SpMM) solution, specifically utilizing the functionalities in cuSPARSE, for straightforward sum-reduced aggregation. Furthermore, it leverages its own CUDA kernel for the implementation of more sophisticated aggregation schemes that factor in edge attributes.
- **GNNAdvisor[3]** is an adaptive and efficient runtime system developed to address the limitations of existing Graph Neural Networks (GNNs) frameworks. It is designed to foster the acceleration of various GNN workloads on GPU platforms through several strategic interventions, such as performance-relevant features identification, workload management, GPU memory hierarchy utilization and lightweight analytical model.
- **CuSPARSE[4]** is a library in NVIDIA's CUDA toolkit, facilitating optimized sparse matrix operations through GPU-acceleration. It is essential in high-performance computing applications, helping developers manage sparse matrices more efficiently in CUDA-accelerated environments.

[1] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds (ICLR), 2019.

[2] MinjieWang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

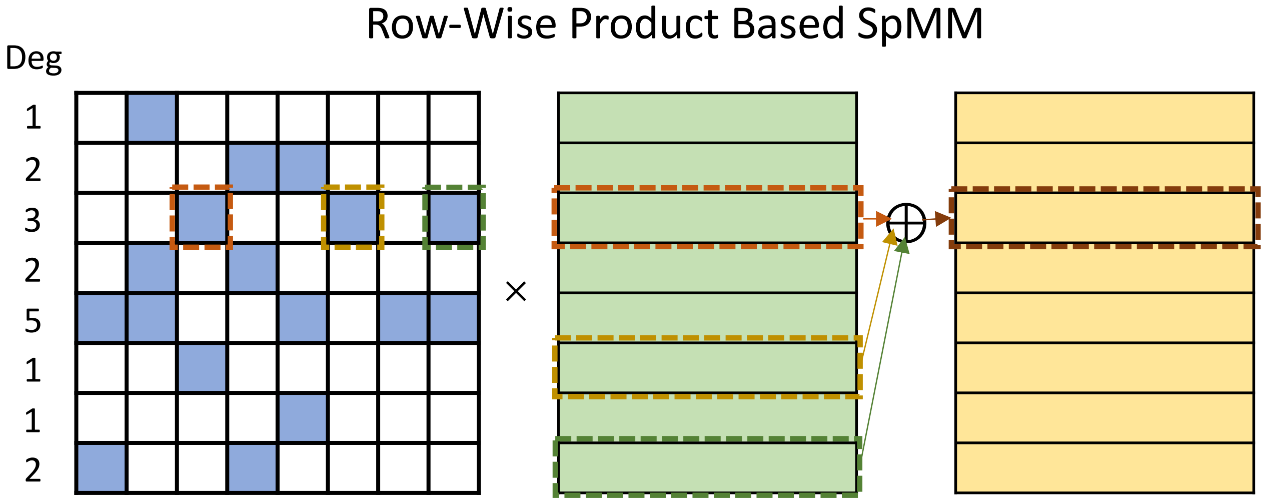
[3] Yuke Wang, Boyuan Feng, Gushu Li, Shuangchen Li, Lei Deng, Yuan Xie, and Yufei Ding. Gnnadvisor: An efficient runtime system for gnn acceleration on gpus. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI'21), 2021.

[4] M. Naumov, L. Chien, P. Vandermersch, and U. Kapasi. Cusparse library. GPU Technology Conference (GTC), 2010.

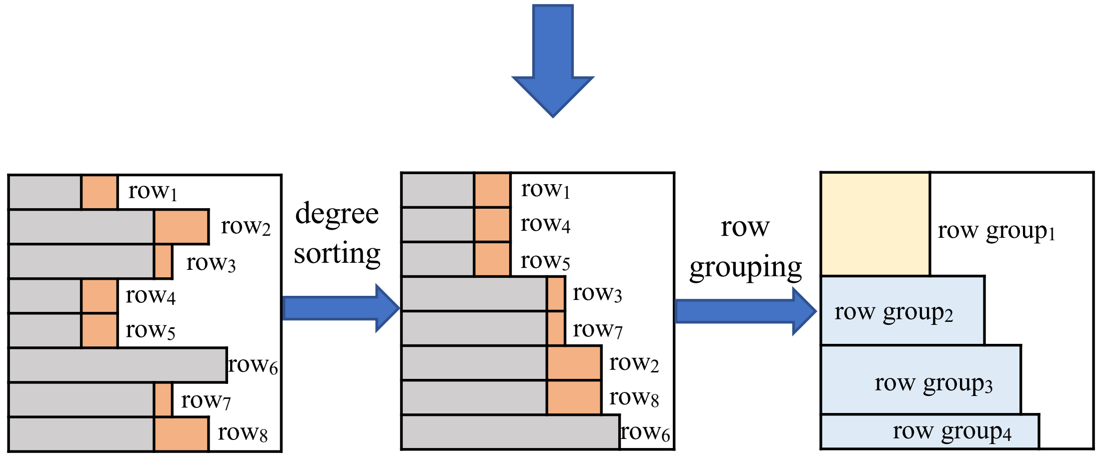
Research Gaps

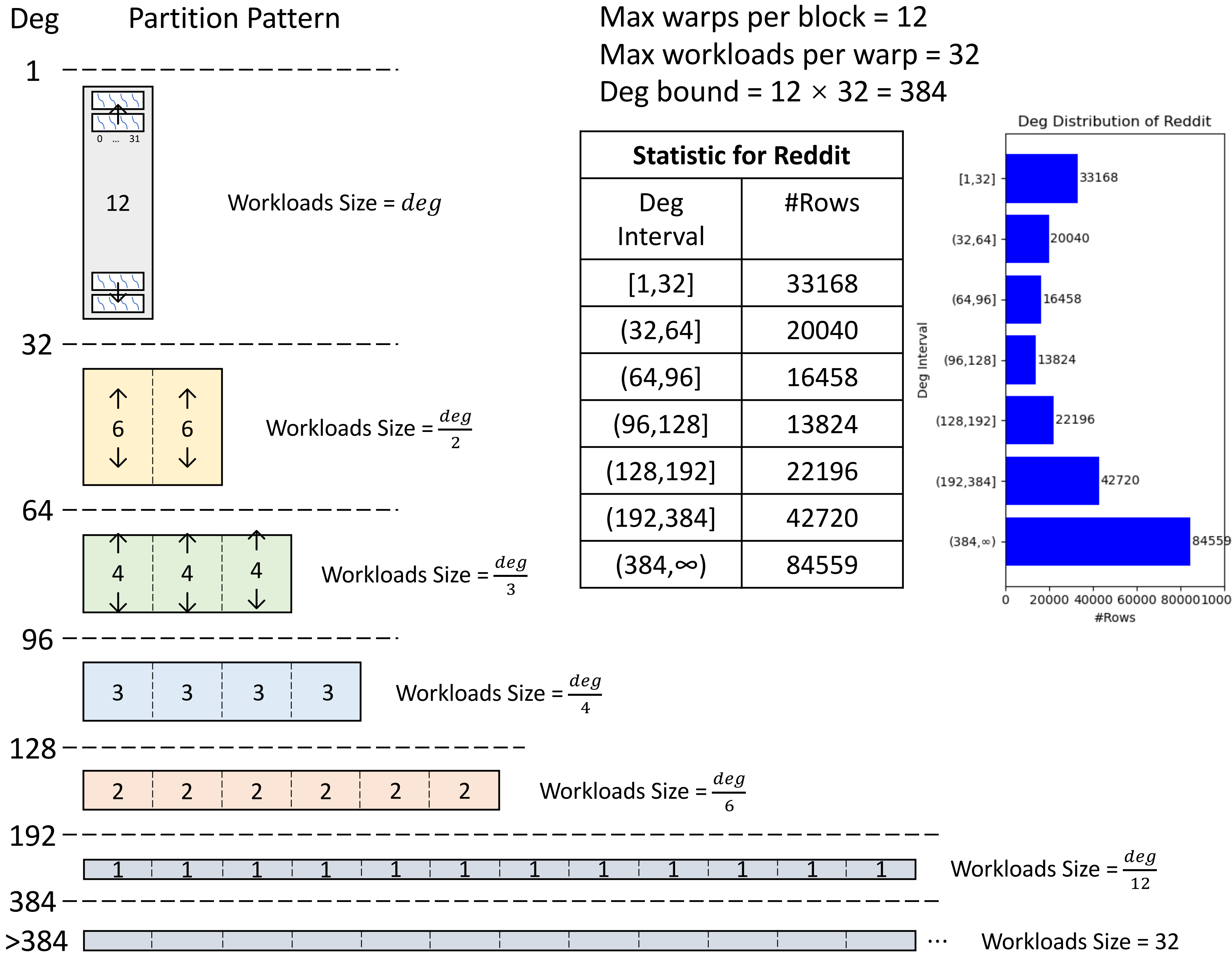
- Limitations:
  - Both PyG and DGL face low scalability when dealing with large sparse graphs with high-dimensional node embeddings.
    - PyG: due to high-overhead atomic operations.
    - DGL: despite utilizing SpMM strategies and its CUDA kernel for more complex aggregations.
  - GNNAdvisor leverages Non-zero groups (NG) to improve workload mapping.
    - Warp-level workload imbalance and resource underutilization when dealing with graphs exhibiting power-law non-zero distribution.
  - CuSPARSE is a popular baseline for SpMM kernels, but restricts further insight due to its closed-source nature.
- Hilight of our **Accel-GCN** design:
  - Lightweight **degree sorting** stage to group nodes with similar degreeBlock-level partition.
  - **Block-level partition** strategy that dynamically adjusts warp workload sizes, enhancing shared memory locality and workload balance, and reducing metadata overhead.
  - **Combined warp** strategy that improves memory coalescing and computational parallelism in the column dimension of dense matrices.

Degree Sorting and Block-level Partition



Workload unit: One non-zero multiplies with one right-hand row 🟦 × 🟩 = 🟨





Deg
Partition Pattern
1
0 ... 31
12
Workloads Size = deg
32
6
6
Workloads Size = $\frac{deg}{2}$
64
4
4
4
Workloads Size = $\frac{deg}{3}$
96
3
3
3
3
Workloads Size = $\frac{deg}{4}$
128
2
2
2
2
2
2
Workloads Size = $\frac{deg}{6}$
192
1
1
1
1
1
1
1
1
1
1
1
1
Workloads Size = $\frac{deg}{12}$
384
>384
···
Workloads Size = 32

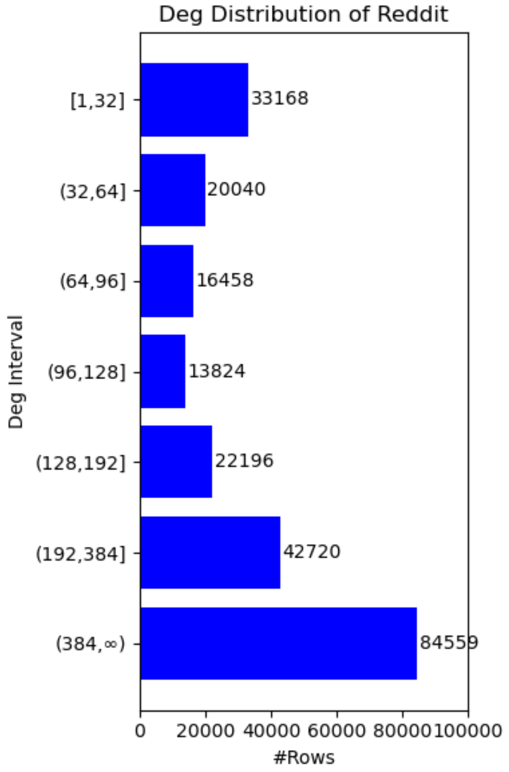
Max warps per block = 12

Max workloads per warp = 32

Deg bound = 12 × 32 = 384

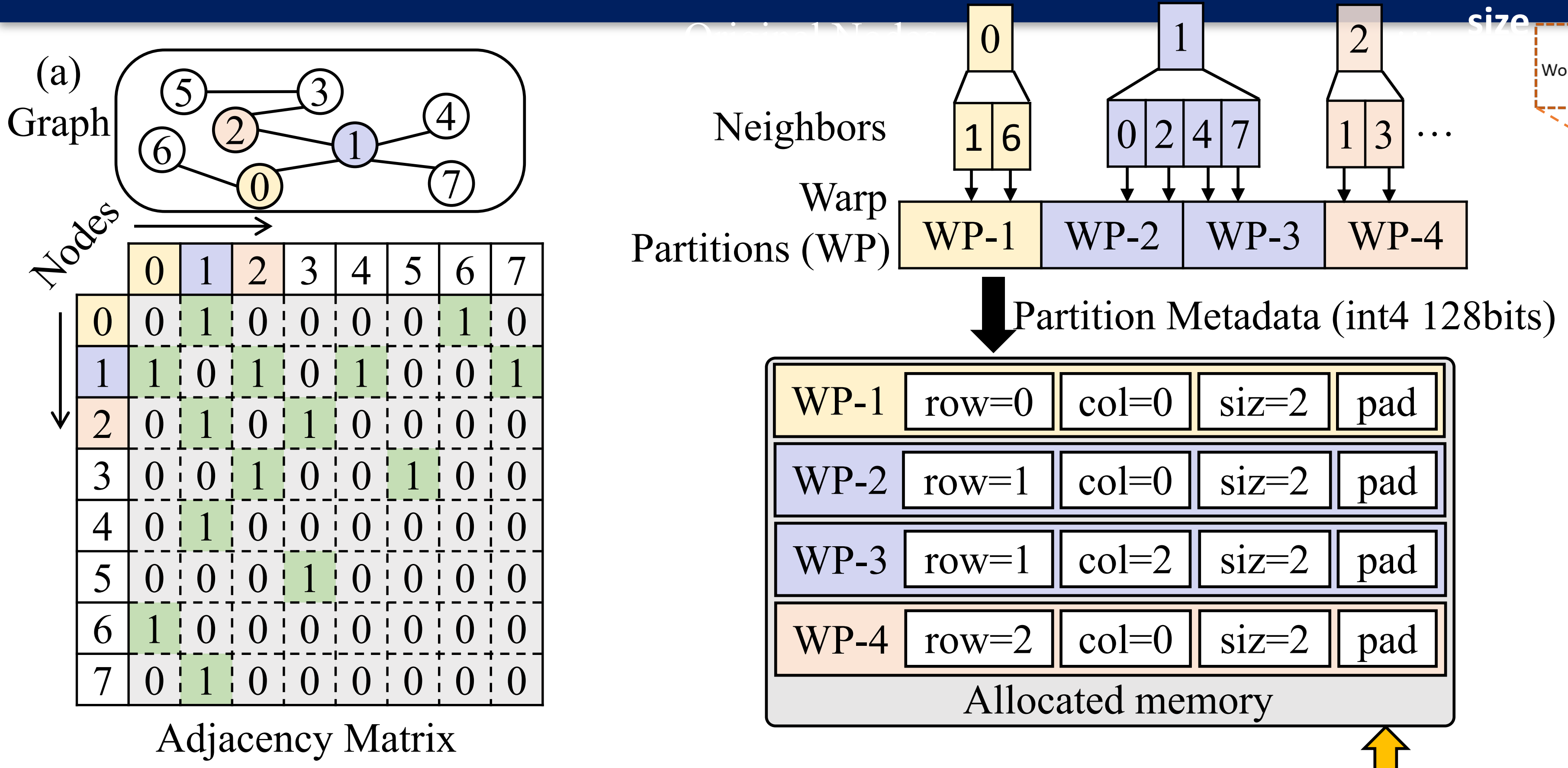
Statistic for Reddit

Deg Interval	#Rows
[1,32]	33168
(32,64]	20040
(64,96]	16458
(96,128]	13824
(128,192]	22196
(192,384]	42720
(384,∞)	84559



Meta-data

Workload size = 2

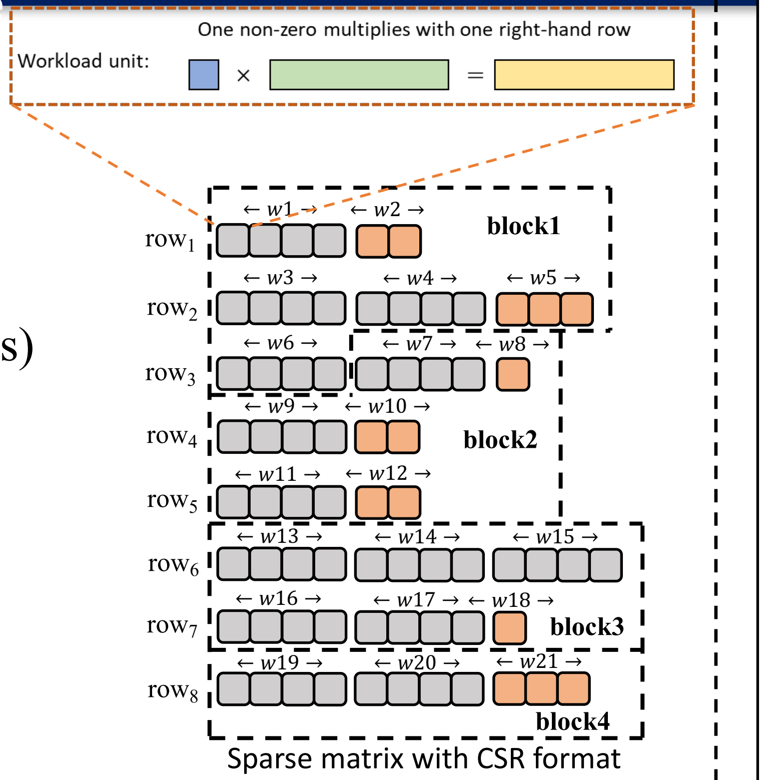


	0	1	2	3	4	5	6	7
0	0	1	0	0	0	0	1	0
1	1	0	1	0	1	0	0	1
2	0	1	0	1	0	0	0	0
3	0	0	1	0	0	1	0	0
4	0	1	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0
6	1	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	0

Adjacency Matrix

WP-1	row=0	col=0	siz=2	pad
WP-2	row=1	col=0	siz=2	pad
WP-3	row=1	col=2	siz=2	pad
WP-4	row=2	col=0	siz=2	pad

GNNAdvisor: fixed neighbor group (or workload) size



Sparse matrix with CSR format

Workload size = avg deg

Block-level partition meta-data

Decoding procedure

$$block\_rows = \begin{cases} lower\ 16bits\ of\ info, & deg \le \mathrm{deg}\_bound \\ 1, & else \end{cases}$$

$$warp\_nzs = \begin{cases} higher\ 16bits\ of\ info, & deg \le \mathrm{deg}\_bound \\ max\ workloads\ per\ warp\ , & else \end{cases}$$

$$row\_nzs = \begin{cases} deg, & deg \le \mathrm{deg}\_bound \\ info, & else \end{cases}$$

$$warps\_per\_row = \left\lceil \frac{row\_nzs}{warp\_nzs} \right\rceil$$

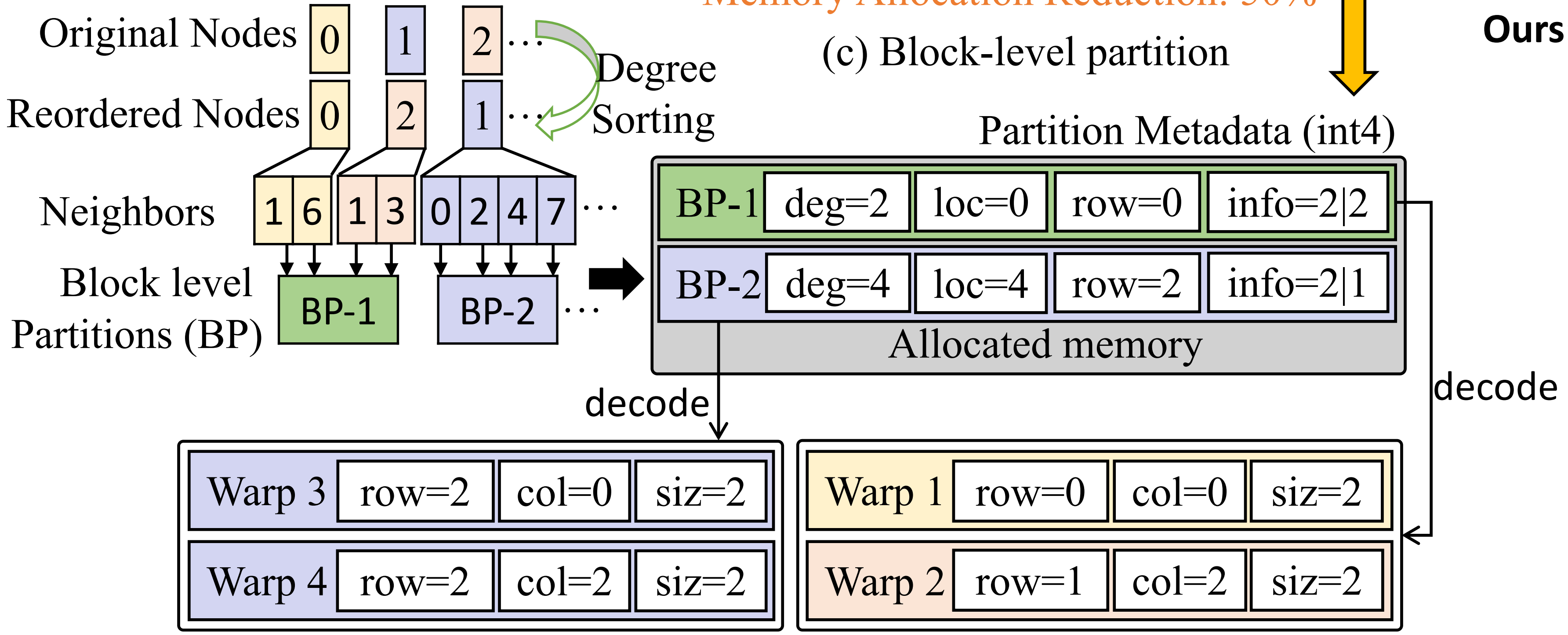
$$row = \left\lfloor \frac{warp\_id}{warps\_per\_row} \right\rfloor$$

$$col = (warp\_id\ mod\ warps\_per\_row)\ \cdot warp\_nzs$$

Memory requirement

$$memory\ for\ BP \approx \frac{memory\ for\ WP}{avg.warps\ per\ block}$$

Memory Allocation Reduction: 50%

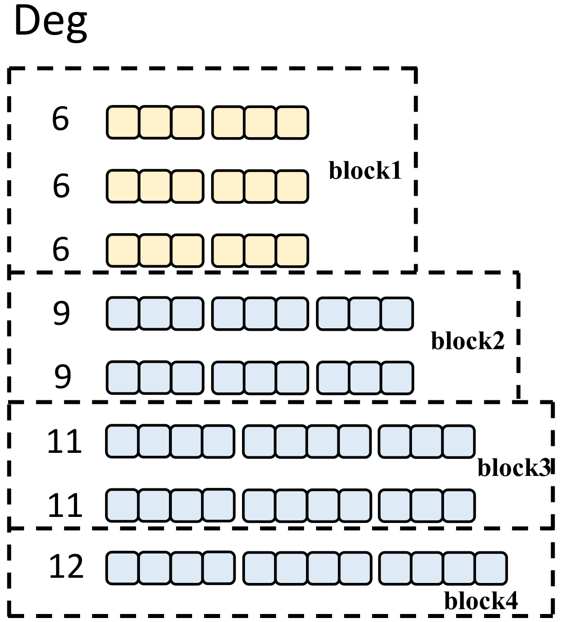


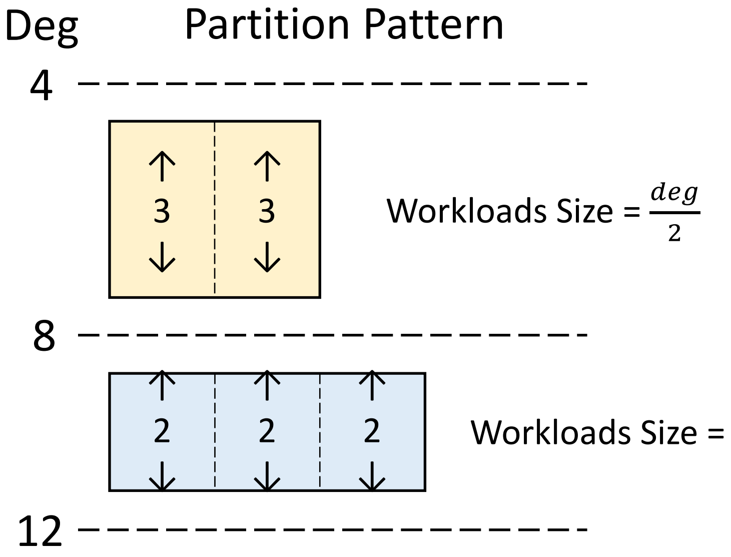
BP-1	deg=2	loc=0	row=0	info=2|2
BP-2	deg=4	loc=4	row=2	info=2|1

Warp 3	row=2	col=0	siz=2
Warp 4	row=2	col=2	siz=2

Warp 1	row=0	col=0	siz=2
Warp 2	row=1	col=2	siz=2

Ours: dynamic workload size




Deg
Partition Pattern
4
8
12
3
3
2
2
2
Workloads Size = $\frac{deg}{2}$
Workloads Size = $\frac{deg}{3}$

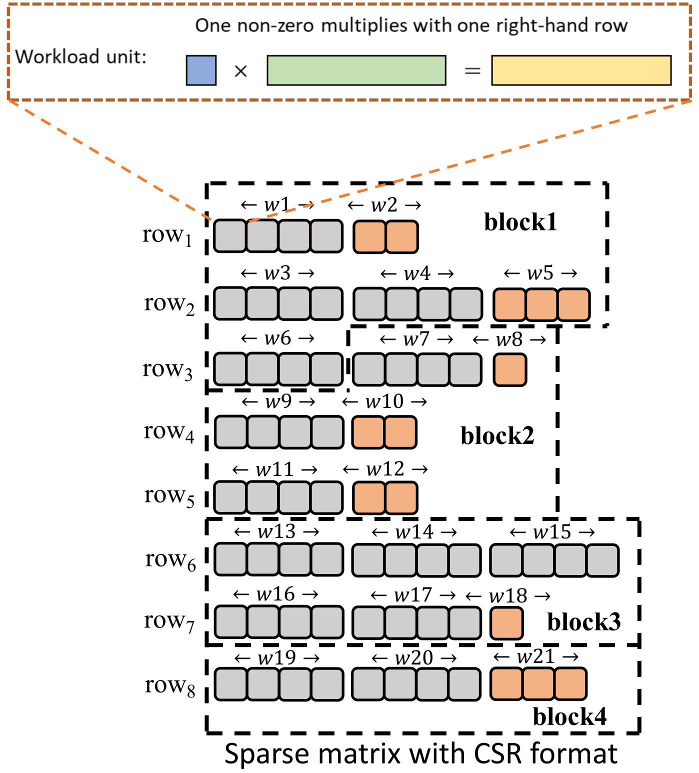
Example for

Max warps per block = 6

Max workloads per warp = 4

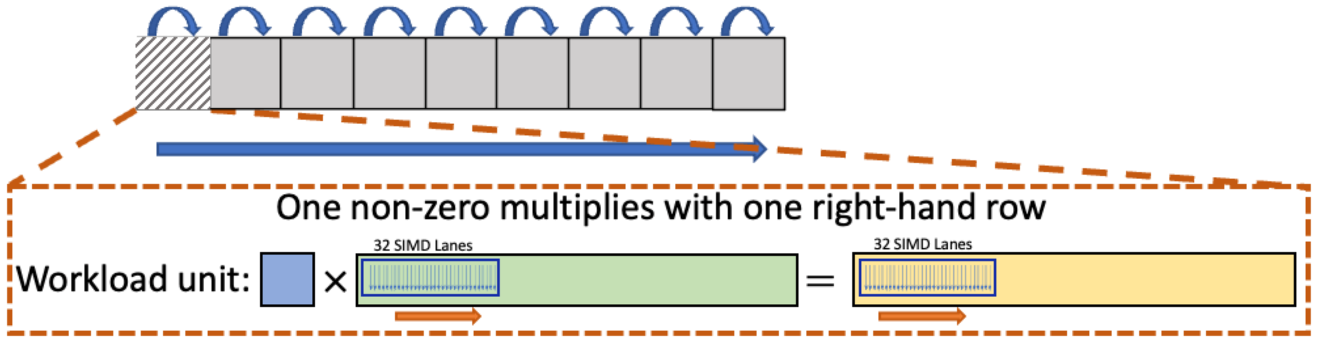
Fixed NG (workload) Size

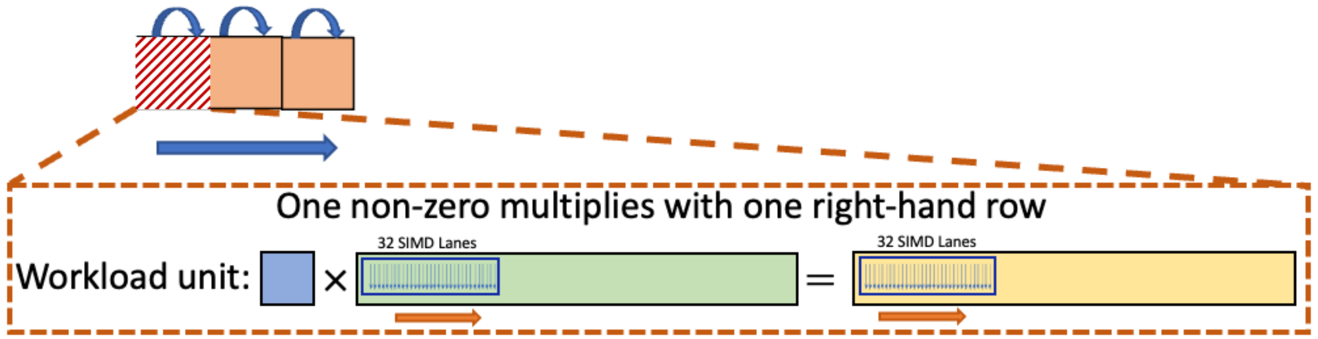
(imbalanced workload within blocks)



Sparse matrix with CSR format

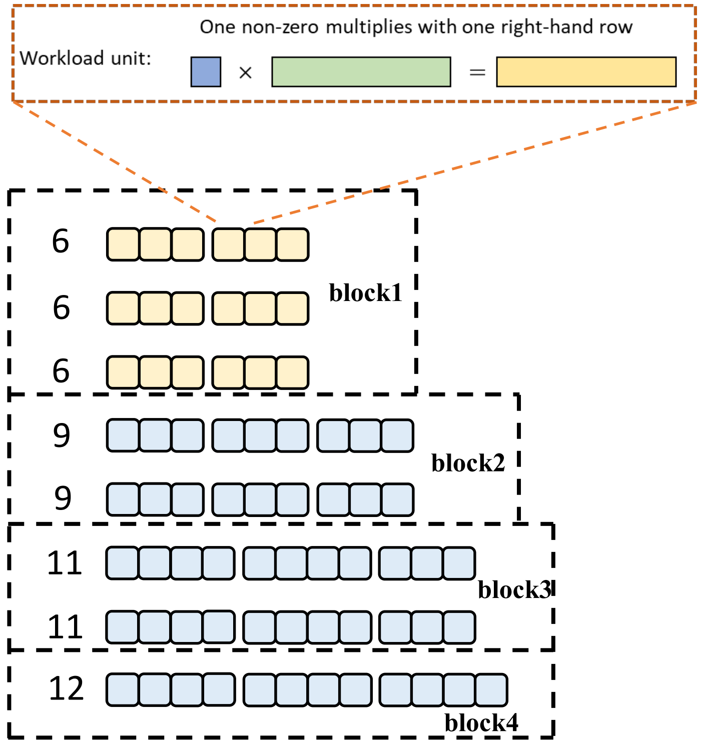
Workload size = avg deg

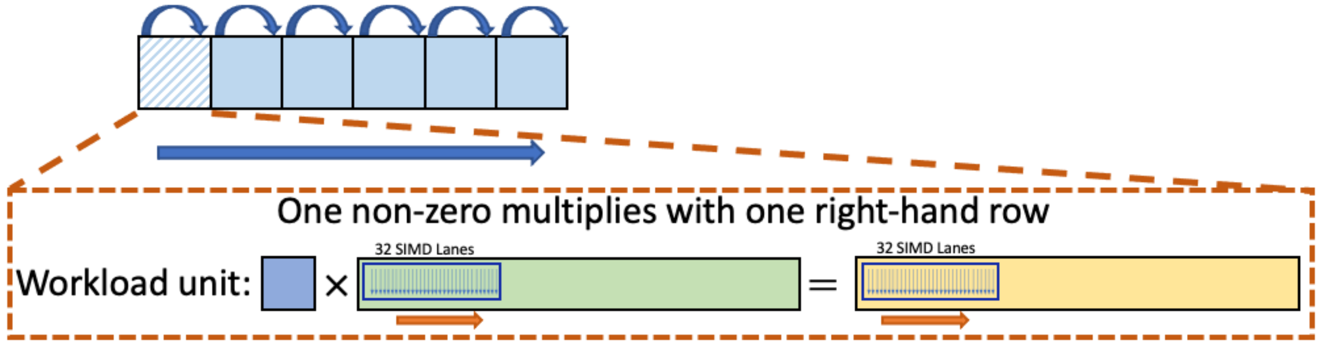


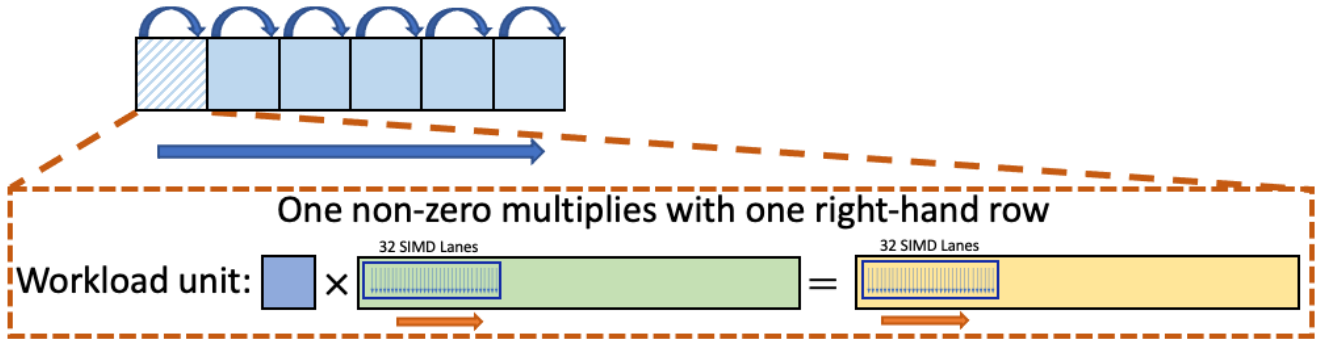


Block-level Partition (ours)

(balanced workload within blocks)

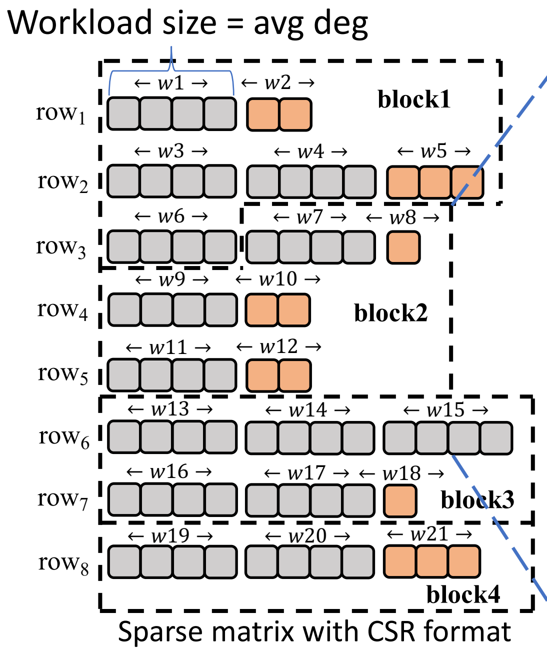


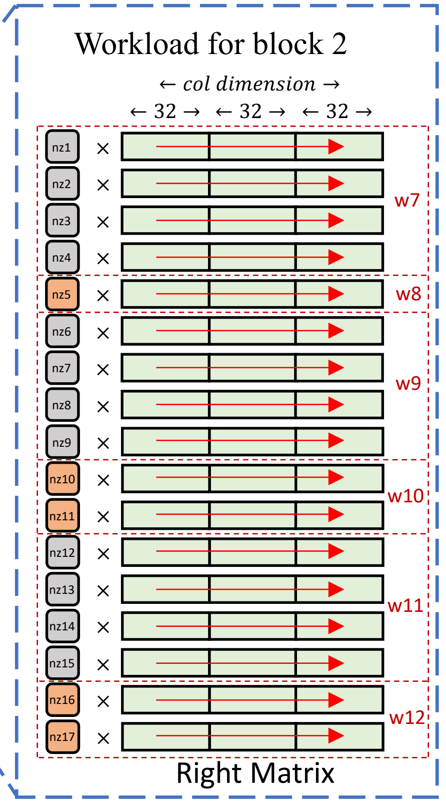




Combined Warp Strategy

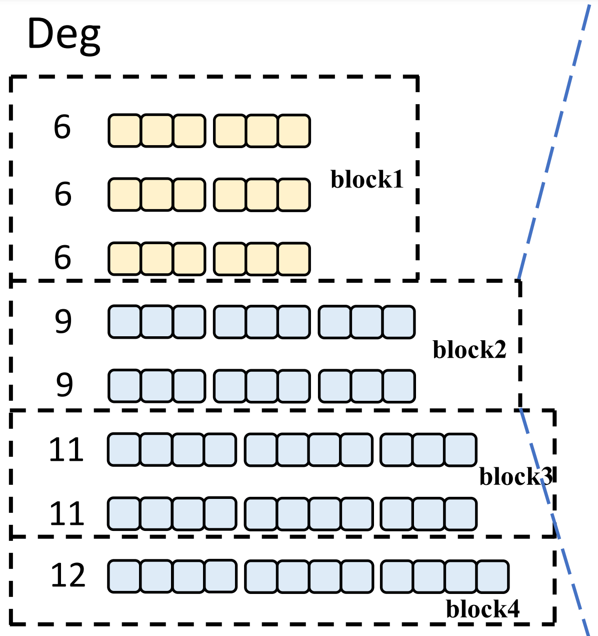
Single warp looping manner



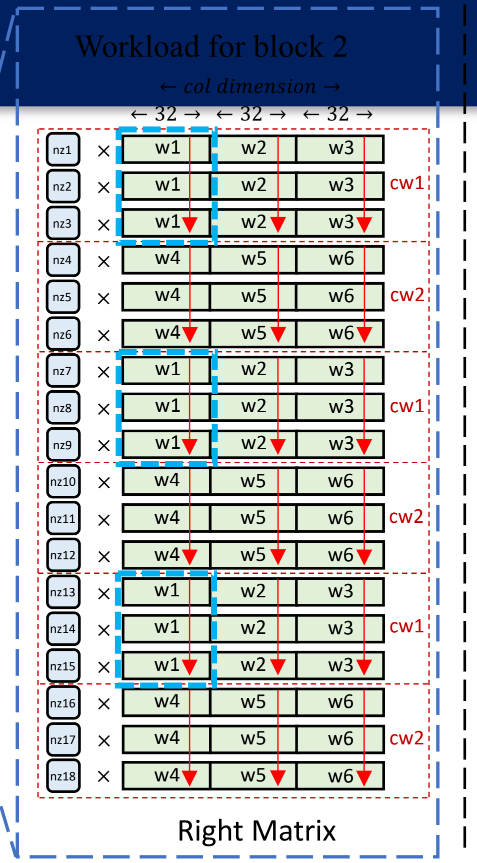


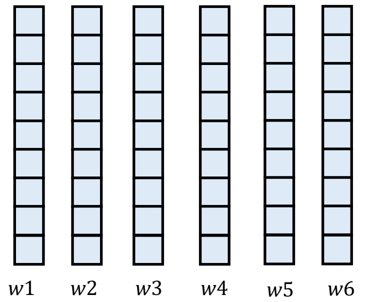
(a) warp-level workload partition

Combined warp manner

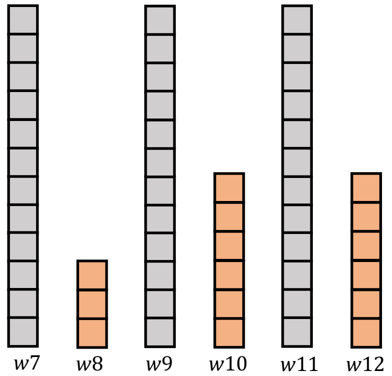


(b) block-level workload partition





Block partition workload (column dimension = 96)



Warp-level partition workload (column dimension = 96)

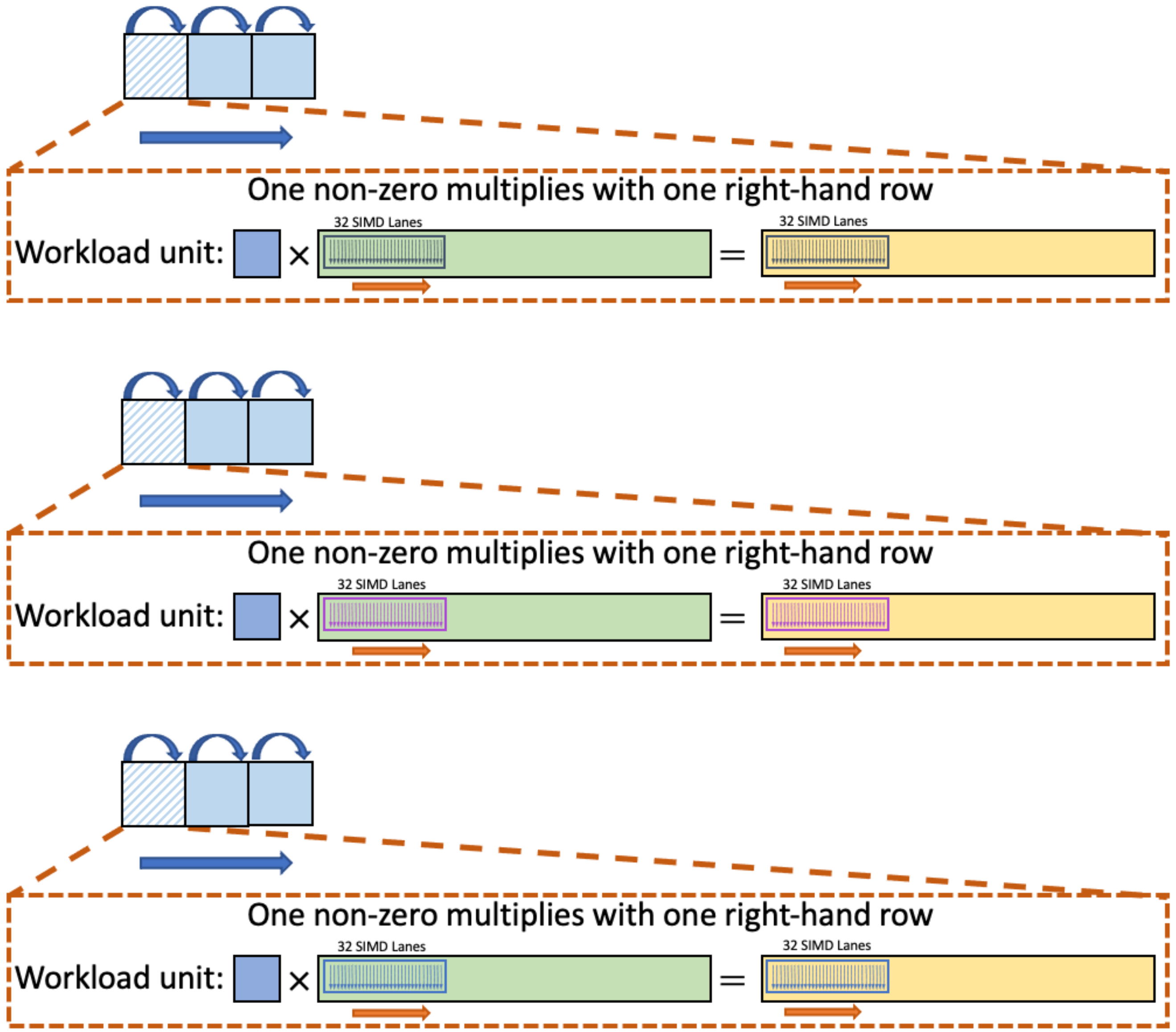
Single warp looping manner:

```
for each warp:
    for all nzs handled by this warp:
        get the location of the workload;
        for j = 1, … , dim / 32:
            execute the workload with parallelism of 32;
        …
    …
```

Combined warp manner:

```
round_dim = ((dim + 31) / 32) * 32;
for i = 1, … , dim / 32 :
    form combined warps consist of round_dim threads;
    for each combined warp:
        for all nzs handled by this combined warp in i-th interval :
            get the location of the workload;
            execute the workload with parallelism of round_dim;
        …
    …
```

Single Warp Looping Manner
(less parallelism in execution)



One non-zero multiplies with one right-hand row

Workload unit: × 32 SIMD Lanes = 32 SIMD Lanes

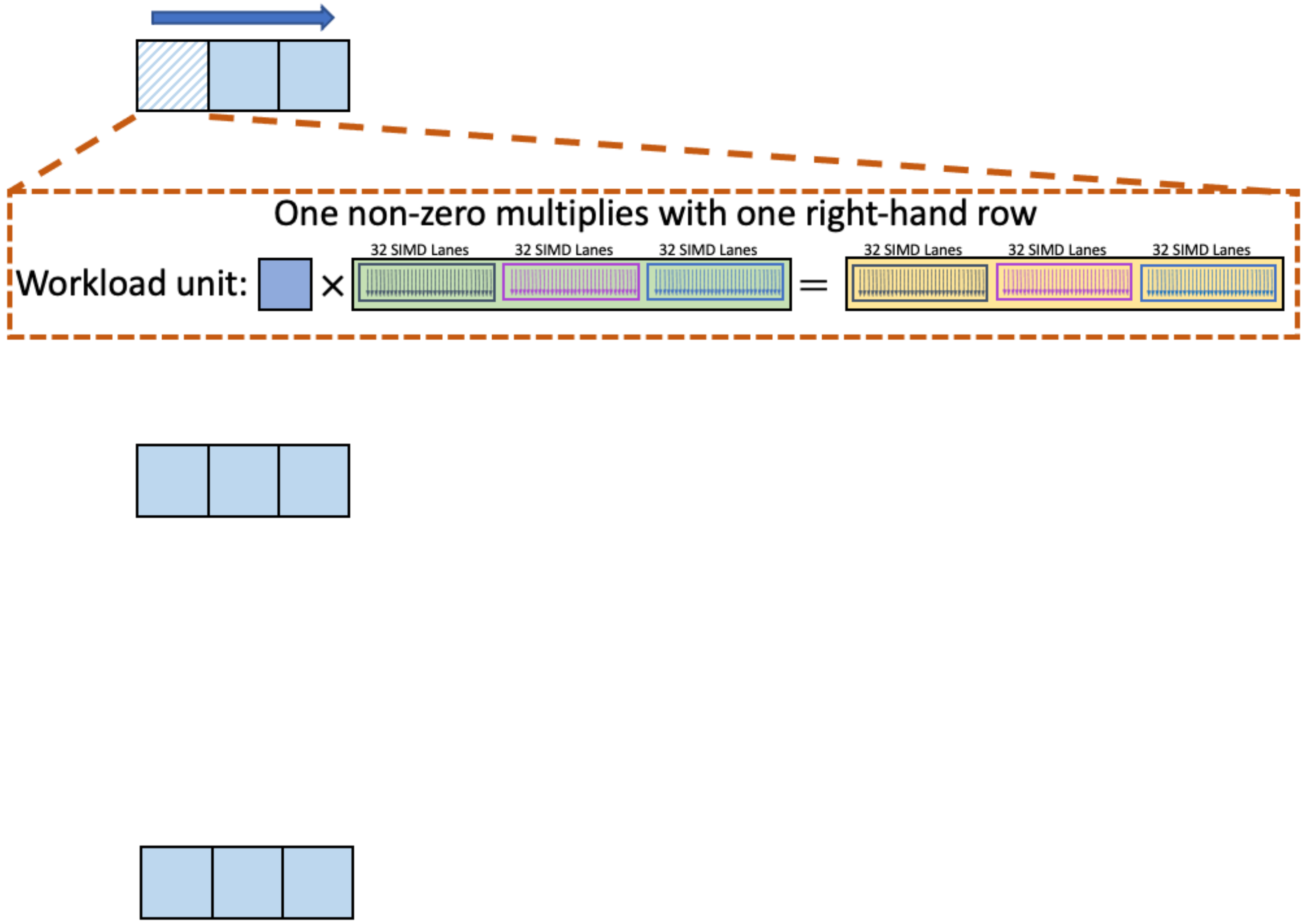
One non-zero multiplies with one right-hand row

Workload unit: × 32 SIMD Lanes = 32 SIMD Lanes

One non-zero multiplies with one right-hand row

Workload unit: × 32 SIMD Lanes = 32 SIMD Lanes

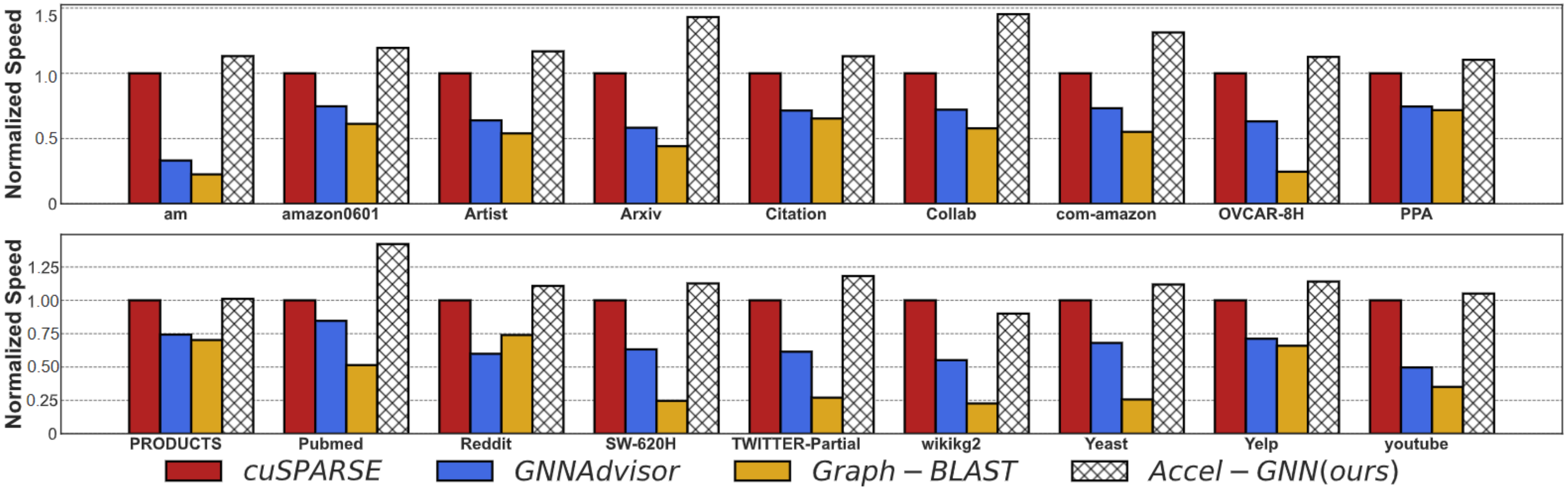
Combined Warp Manner (ours)
(improved parallelism in execution)



One non-zero multiplies with one right-hand row

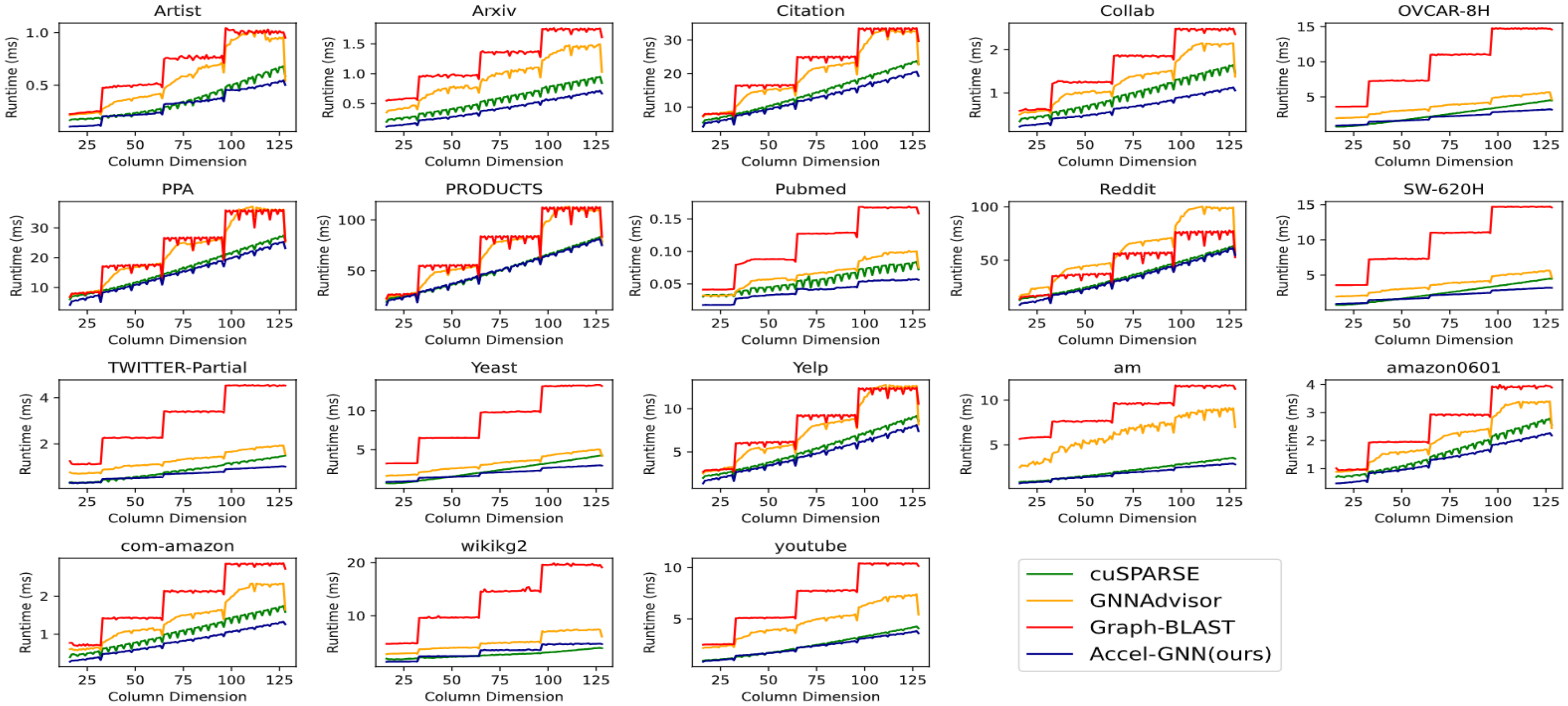
Workload unit: × 32 SIMD Lanes 32 SIMD Lanes 32 SIMD Lanes = 32 SIMD Lanes 32 SIMD Lanes 32 SIMD Lanes

Overall Performance



- Average improvement of 1.17× over cuSPARSE; up to 1.45× improvement.
- More improvement than GNNAdvisor and graph-BLAST across all benchmark graphs,
  - Average speedup of 1.86× and 2.94×, and a maximum speedup of 3.41× and 5.02×, respectively.

Kernel Execution Time vs. Column Dimension



- A gradual and smooth l increase in runtime as the column dimension increases.
  - Benefiting from memory coalescing and automatic alignment of intermediate results proffered by the combined warp strategy.

Block-level Partition vs. Warp-level Partition

Profiling results on 18 graphs; each with 18 dimensions; total **2016** graph tests.

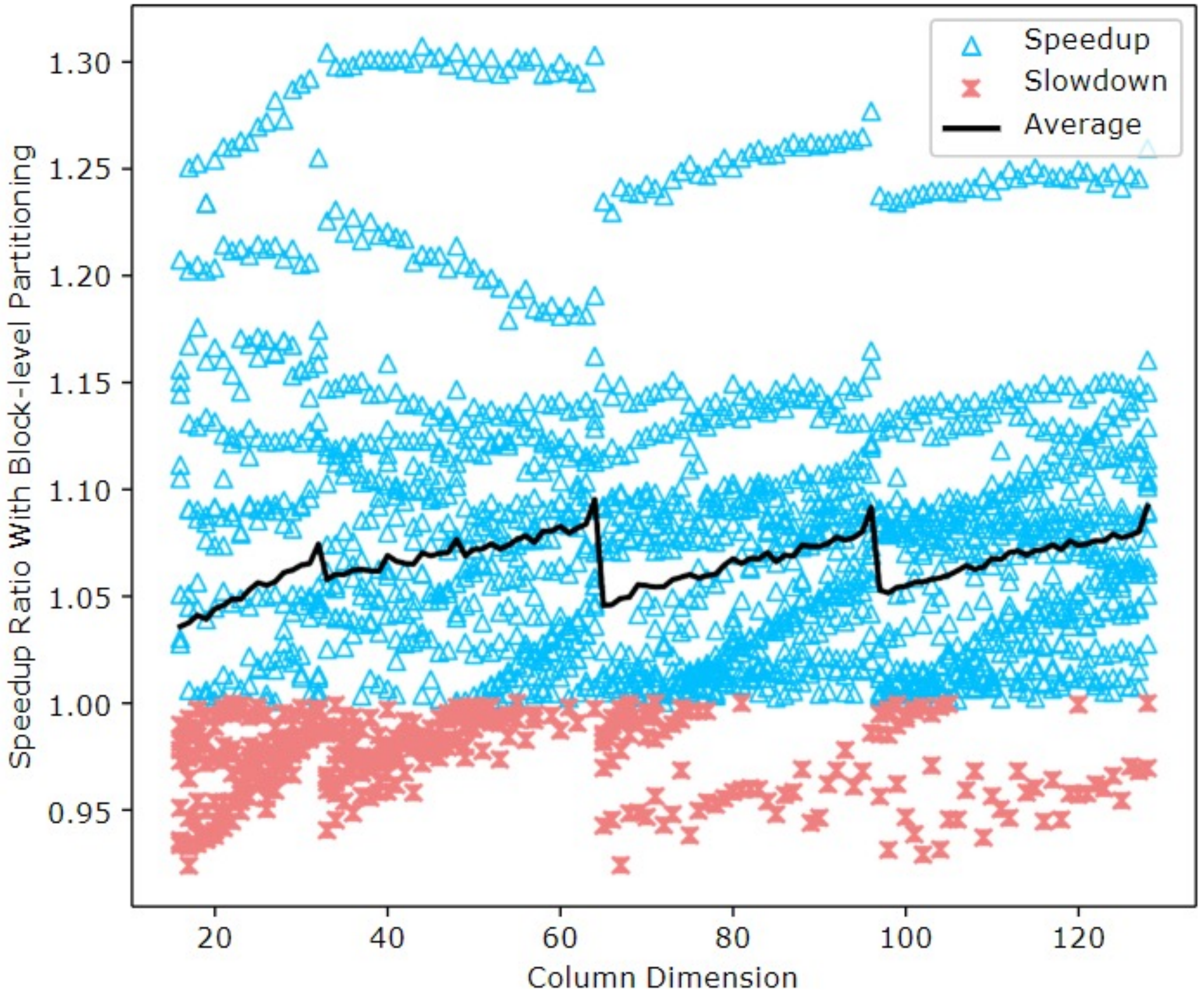


Fig. 7. Speedup of (i). degree sorting & block-level partition over (ii). warp-level partition. Both integrated with combined-warp strategy.

- Block-level partition has realized an average speedup ranging from 1.05× to 1.07× across disparate column dimension intervals, culminating in a peak improvement of 1.31×; and a least effective case of 0.92×.

Combined Warp vs. Single warp looping

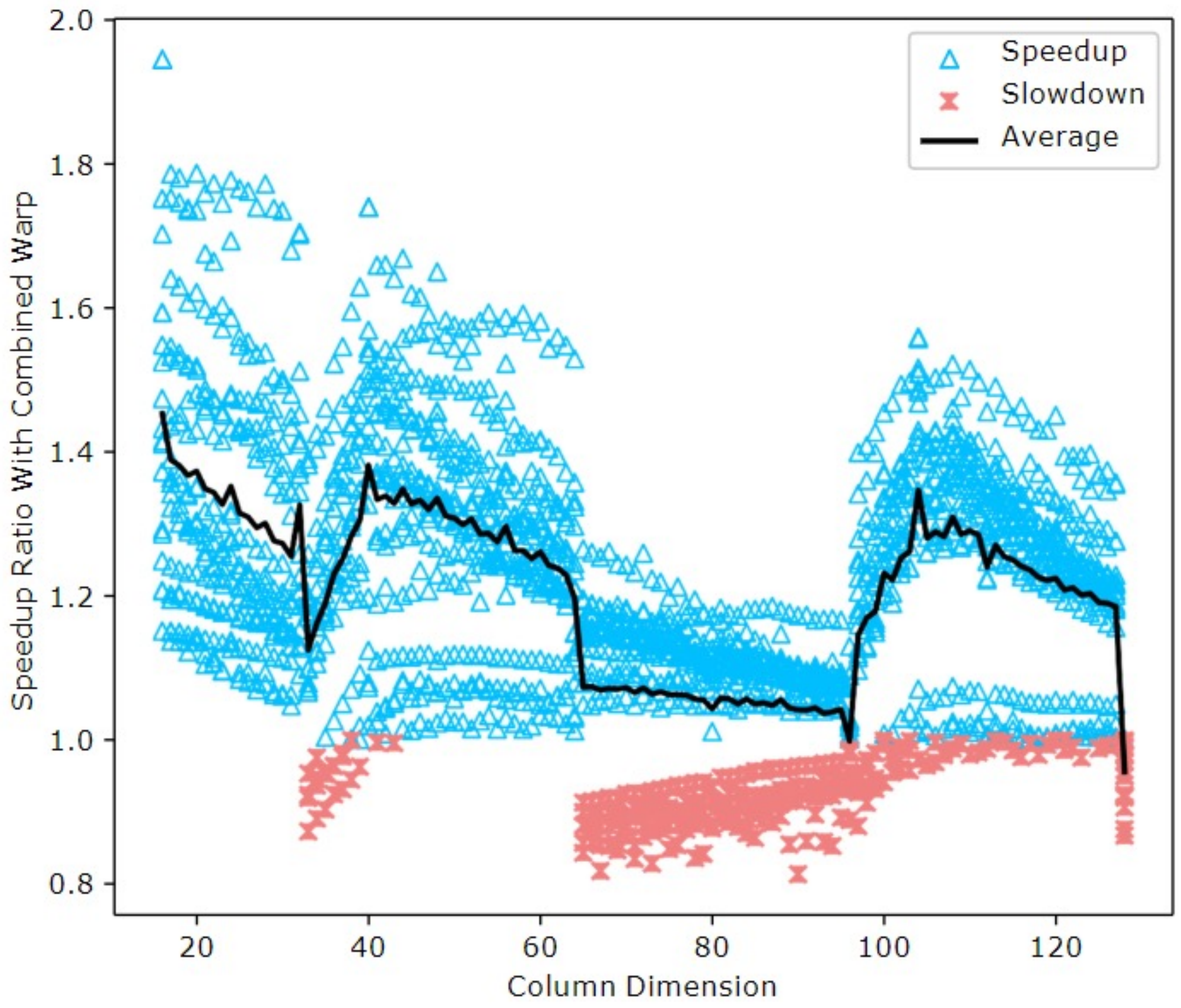
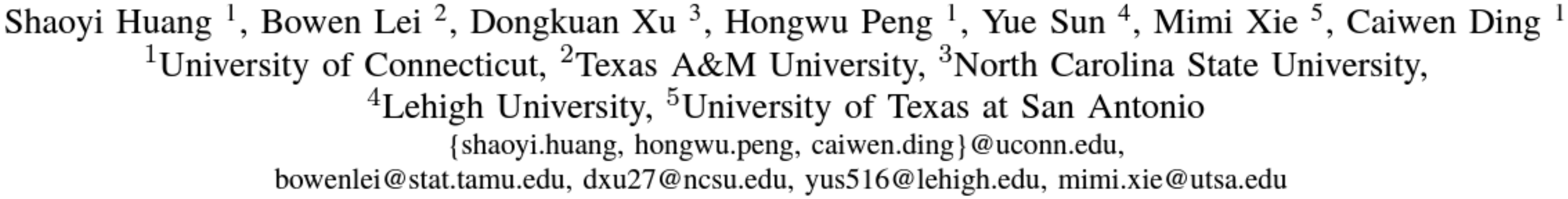


Fig. 8. Speedup of degree sorting & block-level partition (i). with combined-warp strategy over (ii). without combined-warp strategy.

- Combined warp strategy leads to performance improvement specifically within the column dimension intervals [0, 32], [32, 64], and [96, 128], with an average speed gain recorded between 1.23× and 1.33×.
- This enhancement is somewhat diminished within the column dimension range [64, 96], a divergence potentially ascribable to unaligned cache line size in the prevailing GPU architecture.

Open sourced at: https://github.com/xiexi1990/ICCAD-Accel-GCN in mid August 2023, +14 stars

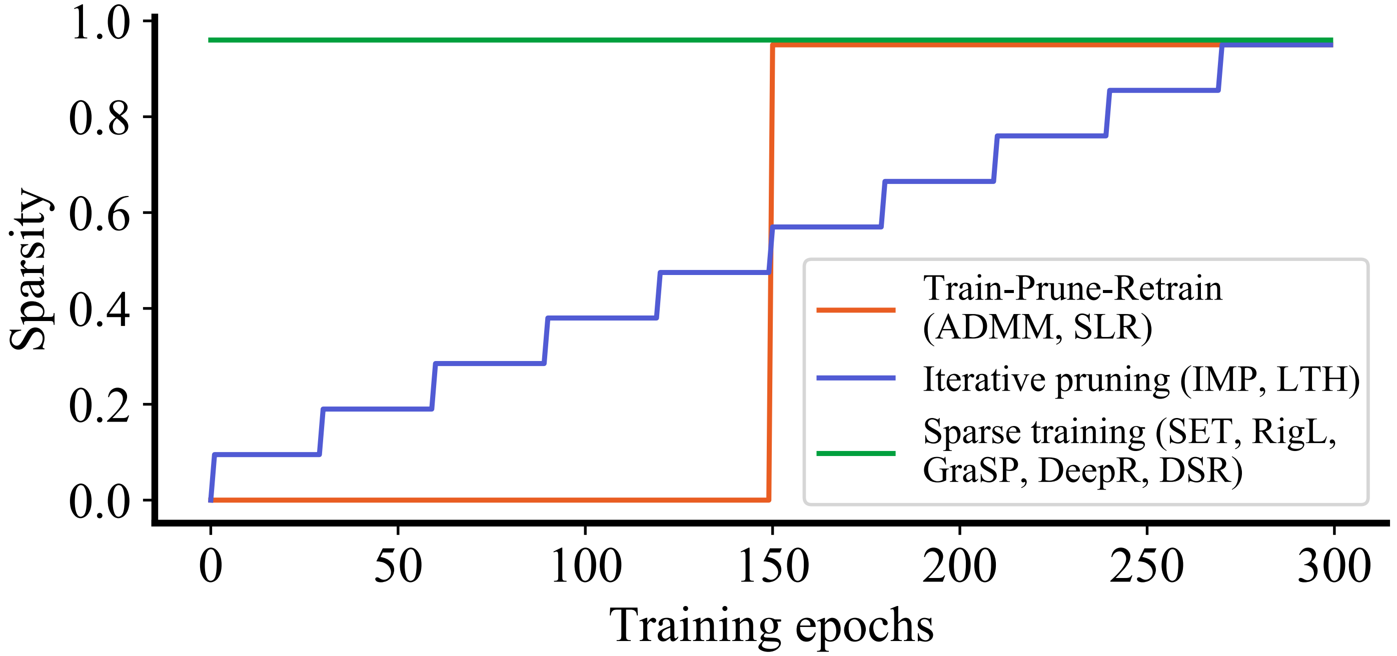
Dynamic Sparse Training via Balancing the Exploration-Exploitation Trade-off



Shaoyi Huang [1], Bowen Lei [2], Dongkuan Xu [3], Hongwu Peng [1], Yue Sun [4], Mimi Xie [5], Caiwen Ding [1]
[1]University of Connecticut, [2]Texas A&M University, [3]North Carolina State University,
[4]Lehigh University, [5]University of Texas at San Antonio
{shaoyi.huang, hongwu.peng, caiwen.ding}@uconn.edu,
bowenlei@stat.tamu.edu, dxu27@ncsu.edu, yus516@lehigh.edu, mimi.xie@utsa.edu

Sparse Training

Weight pruning vs. sparse training



Comparison of different sparsification methods

	Train-prune-retrain	**Iterative pruning**	**Sparse training**
Sparsity	Low	Medium	**High**
Accuracy	Low	Medium	**High**
Training cost	High	Medium	**Low**

Weight pruning: ADMM [1], Lottery Ticket Hypothesis (LTH) [2];

- ➢ **Training is NOT efficient**; Efficient Inference.

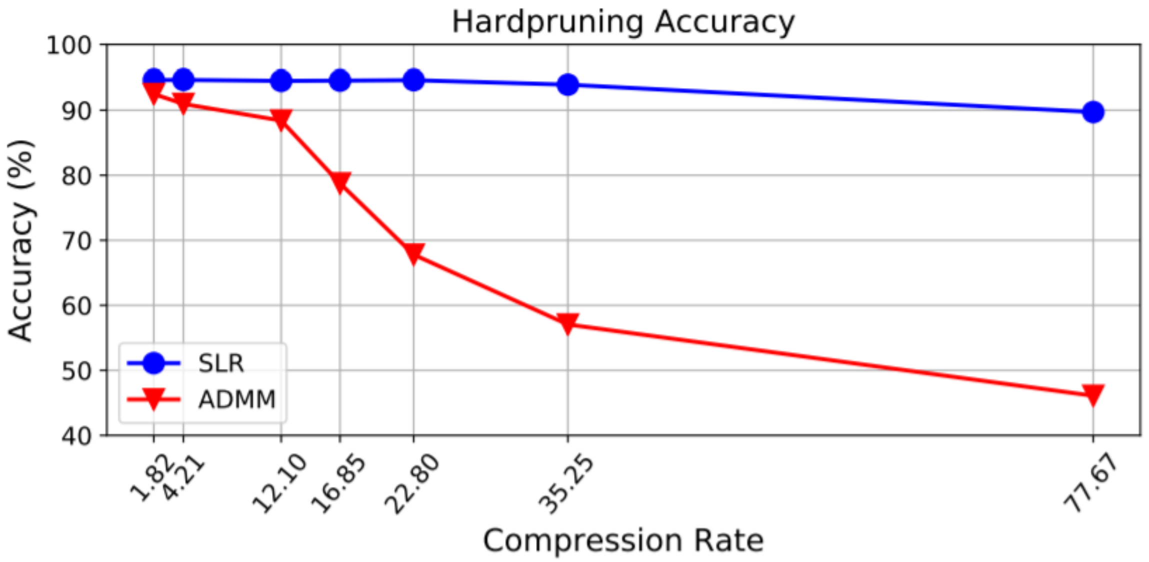
Sparse training: SET[3], RigL[4], GraSP[5], DeepR[6], DSR[7];

- ➢ both efficient **training** and inference
- ➢ Sparsity is introduced from the beginning of training → less memory footprint due to a smaller number of parameters
- ➢ Can achieve same accuracy compared to dense training using same training epochs

[1] Zhang, Tianyun, et al. “A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers.” ECCV (2018).
[2] Jonathan F., and Carbin M., "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks." ICLR, 2019
[3] Mocanu, D., et al., "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science." Nature communications 9.1 (2018): 2383.
[4] Evci, U., et al., "Rigging the lottery: Making all tickets winners." ICML. PMLR, 2020.
[5] Wang, C., et al., "Picking Winning Tickets Before Training by Preserving Gradient Flow." ICLR, 2019.
[6] Bellec, G., et al., Deep rewiring: Training very sparse deep networks. ICLR, 2018.
[7] Mostafa H. and Wang X., Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. ICML, 2019.

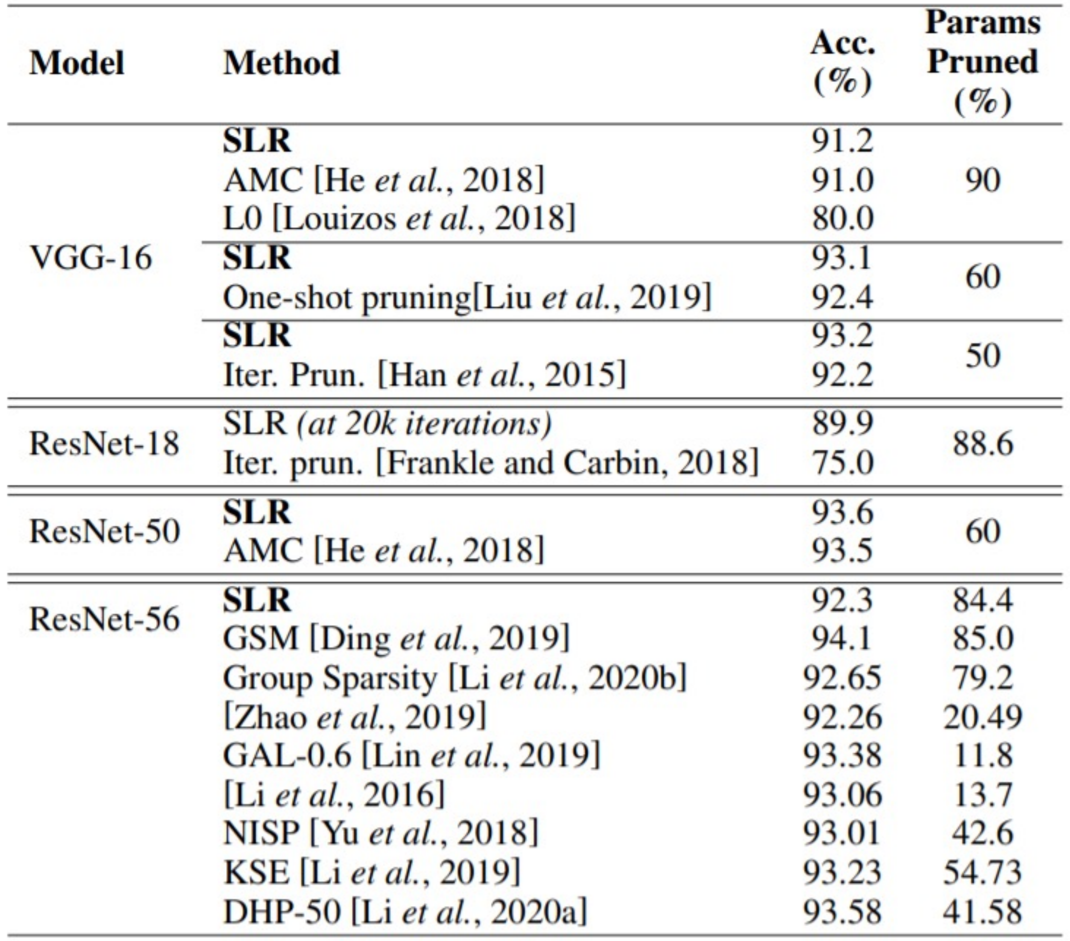
Weight pruning

- One of the most common model compression methods.
- Many works: ADMM [1], AMC [2], GSM [3], LTH...
- Use *Surrogate Lagrangian Relaxation (SLR)-based pruning as an example[4]* **(our work).**



Faster convergence than ADMM

(Image Classification)

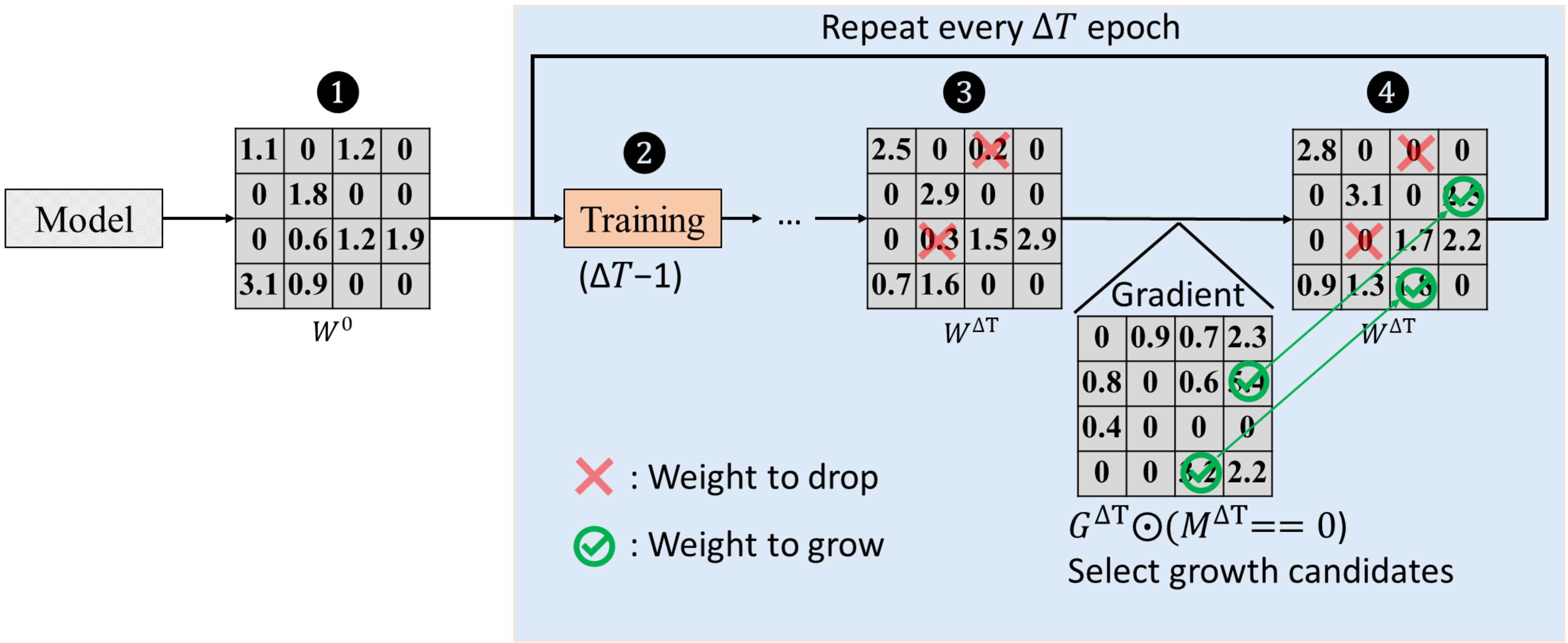


Model	Method	Acc. (%)	Params Pruned (%)
VGG-16	**SLR**	91.2	90
	AMC [He *et al.*, 2018]	91.0	
	L0 [Louizos *et al.*, 2018]	80.0	
	SLR	93.1	60
	One-shot pruning[Liu *et al.*, 2019]	92.4	
	SLR	93.2	50
	Iter. Prun. [Han *et al.*, 2015]	92.2	
ResNet-18	SLR *(at 20k iterations)*	89.9	88.6
	Iter. prun. [Frankle and Carbin, 2018]	75.0	
ResNet-50	**SLR**	93.6	60
	AMC [He *et al.*, 2018]	93.5	
ResNet-56	**SLR**	92.3	84.4
	GSM [Ding *et al.*, 2019]	94.1	85.0
	Group Sparsity [Li *et al.*, 2020b]	92.65	79.2
	[Zhao *et al.*, 2019]	92.26	20.49
	GAL-0.6 [Lin *et al.*, 2019]	93.38	11.8
	[Li *et al.*, 2016]	93.06	13.7
	NISP [Yu *et al.*, 2018]	93.01	42.6
	KSE [Li *et al.*, 2019]	93.23	54.73
	DHP-50 [Li *et al.*, 2020a]	93.58	41.58

Better accuracy

[1] Li, Zhe, et al. "E-RNN: Design optimization for efficient recurrent neural networks in FPGAs." In 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 69-80. IEEE, 2019. **(our work).**
[2] He, Yihui, et al. "AMC: AutoML for Model Compression and Acceleration on Mobile Devices." ECCV (2018).
[3] Ding, Xiaohan, et al. "Global Sparse Momentum SGD for Pruning Very Deep Neural Networks." NeurIPS (2019).
[4] Gurevin, Deniz, et al. "Enabling Retrain-free Deep Neural Network Pruning Using Surrogate Lagrangian Relaxation." IJCAI (2021). **(our work).**

State-of-the-arts on sparse training



1.1	0	1.2	0
0	1.8	0	0
0	0.6	1.2	1.9
3.1	0.9	0	0

W^0

2.5	0	0.2 ✗	0
0	2.9	0	0
0	0.3 ✗	1.5	2.9
0.7	1.6	0	0

$W^{\Delta T}$

0	0.9	0.7	2.3
0.8	0	0.6	5.4 ✓
0.4	0	0	0
0	0	3.2 ✓	2.2

$G^{\Delta T} \odot (M^{\Delta T} == 0)$

Select growth candidates

2.8	0	0 ✗	0
0	3.1	0	2.5 ✓
0	0 ✗	1.7	2.2
0.9	1.3	1.8 ✓	0

$W^{\Delta T}$

Step 1: Randomly initialize weight tensor with a fixed sparsity.

Step 2: Train the sparsified weight for ΔT -1 epochs, where ΔT is the drop-grow frequency.

Step 3: During forward propagation, drop *k* weights with the least absolute magnitude (*k*=2 in the example).

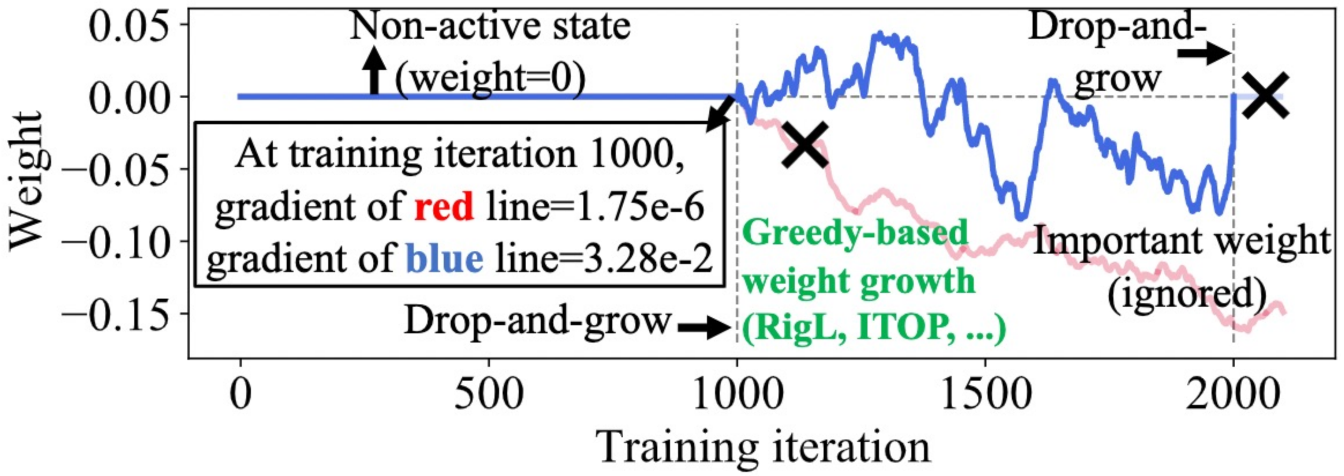
Step 4: During backpropagation, grow the weights with the highest k absolute gradients back to non-zero.

Repeat steps 2-4 till the end of the training.

Evci, Utku, et al "Rigging the lottery: Making all tickets winners." ICML, 2020.

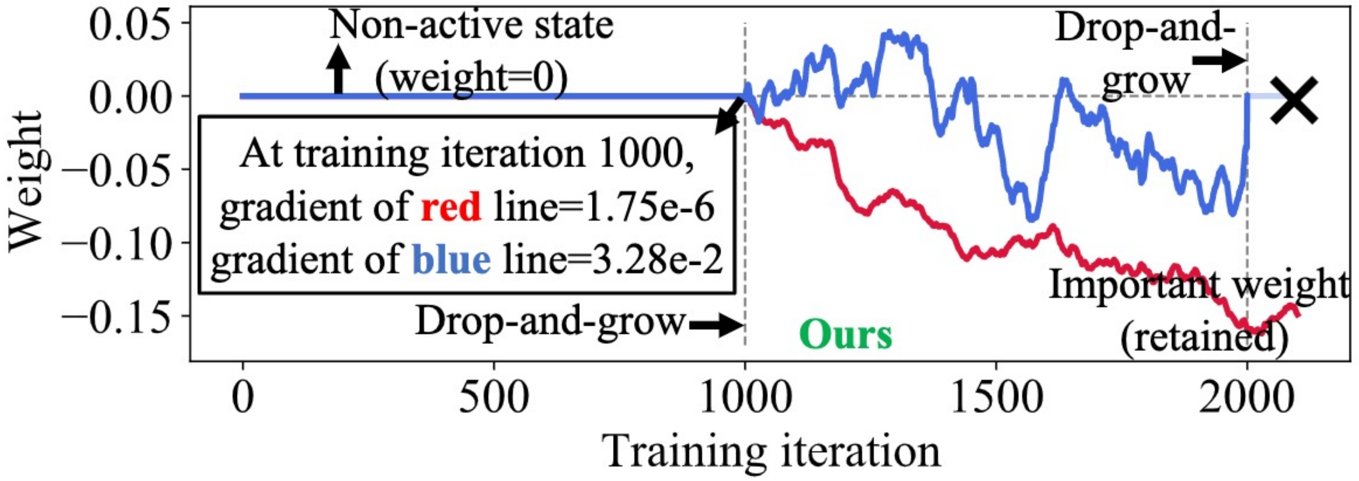
How to achieve Extreme Sparsity?

Motivation



(a) Non-active weights with small initial gradients are ignored in greedy-based weight growth methods (i.e., RigL, ITOP, ...)

Highlight of our solution



(b) Non-active weights with small initial gradients could be retained and grown in proposed method.

- Greedy-based grow methods (e.g., RigL, ITOP [1], GraSP, DeepR, DSR;): search for sparse masks with a local minimal -> limited weights coverage-> limited accuracy or sparsity
- Random-based grow methods (SET, ITOP [1]): lower accuracy

[1] Liu, Shiwei, et al. "Do we actually need dense over-parameterization? in-time over-parameterization in sparse training." ICML, 2021.

How to achieve Extreme Sparsity?

Our solution – Balancing the Exploration-Exploitation Trade-off

$$\mathbf{S}_i^t = |\frac{\partial l(\mathbf{W}_i^t, \mathcal{X})}{\partial \mathbf{W}_i^t}| + c\frac{\ln t}{\mathbf{N}_i^t + \epsilon}, \quad t = q\Delta T, \quad i = 1, 2, ..., L$$

Exploitation (magnitude)

Exploration (prefer unexplored regions)

i: Layer number

t: Training iteration

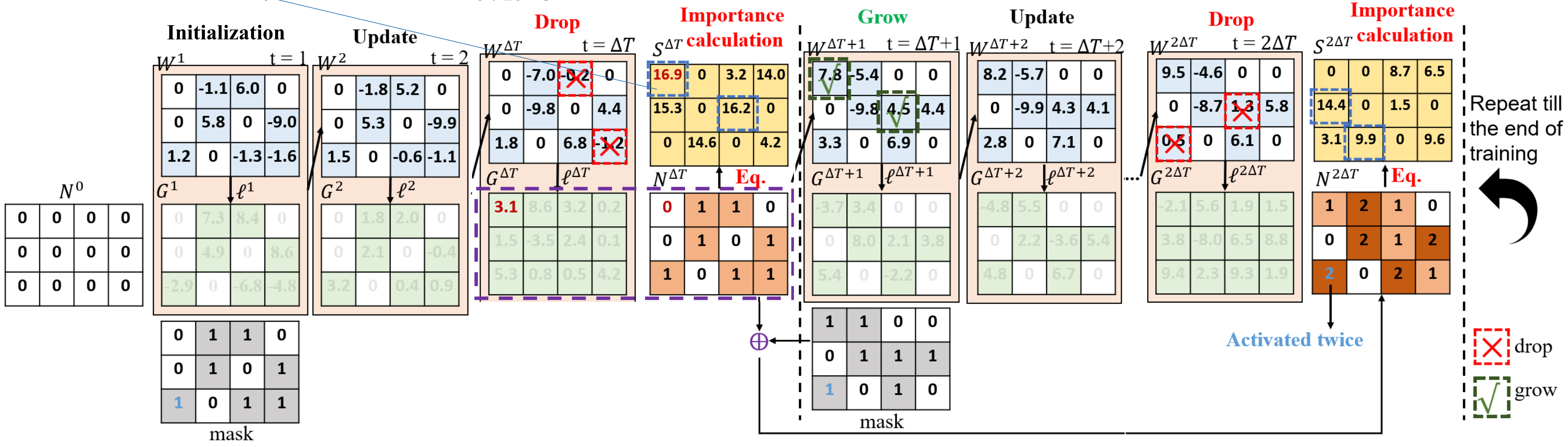
$\mathcal{X}$: Input training data

S_i^t: Importance score tensor in $q-th$ mask update round

$|\frac{\partial l(\mathbf{W}_i^t, \mathcal{X})}{\partial \mathbf{W}_i^t}|$: absolute gradient tensor of $i-th$ layer at $t-th$ iteration

N_i^t: Counter tensor that collects the activated frequency for each weight element

Example: $S_1^{1000} = 16.9 = 3.1 + 2e^{(-5)} \times \frac{ln1000}{0+1e-5}$



N^0

0	0	0	0
0	0	0	0
0	0	0	0

W^1, $t=1$

0	-1.1	6.0	0
0	5.8	0	-9.0
1.2	0	-1.3	-1.6

G^1

0	7.3	8.4	0
0	4.9	0	8.6
-2.9	0	-6.8	-4.8

mask

0	1	1	0
0	1	0	1
1	0	1	1

W^2, $t=2$

0	-1.8	5.2	0
0	5.3	0	-9.9
1.5	0	-0.6	-1.1

G^2

0	1.8	2.0	0
0	2.1	0	-0.4
3.2	0	0.4	0.9

$W^{\Delta T}$, $t=\Delta T$

0	-7.0	☒ -0.2	0
0	-9.8	0	4.4
1.8	0	6.8	☒ -1.2

$G^{\Delta T}$

3.1	8.6	3.2	0.2
1.5	-3.5	2.4	0.1
5.3	0.8	0.5	4.2

$S^{\Delta T}$

16.9	0	3.2	14.0
15.3	0	16.2	0
0	14.6	0	4.2

$N^{\Delta T}$

0	1	1	0
0	1	0	1
1	0	1	1

$W^{\Delta T+1}$, $t=\Delta T+1$

✓ 7.8	-5.4	0	0
0	-9.8	✓ 4.5	4.4
3.3	0	6.9	0

$G^{\Delta T+1}$

-3.7	3.4	0	0
0	8.0	2.1	3.8
5.4	0	-2.2	0

mask

1	1	0	0
0	1	1	1
1	0	1	0

$W^{\Delta T+2}$, $t=\Delta T+2$

8.2	-5.7	0	0
0	-9.9	4.3	4.1
2.8	0	7.1	0

$G^{\Delta T+2}$

-4.8	5.5	0	0
0	2.2	-3.6	5.4
4.8	0	6.7	0

$W^{2\Delta T}$, $t=2\Delta T$

9.5	-4.6	0	0
0	-8.7	☒ 1.3	5.8
☒ 0.5	0	6.1	0

$G^{2\Delta T}$

-2.1	5.6	1.9	1.5
3.8	-8.0	6.5	8.8
9.4	2.3	9.3	1.9

$S^{2\Delta T}$

0	0	8.7	6.5
14.4	0	1.5	0
3.1	9.9	0	9.6

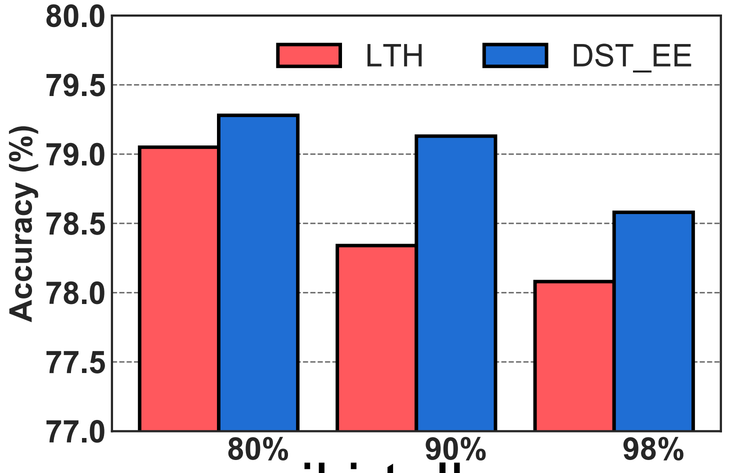
$N^{2\Delta T}$

1	2	1	0
0	2	1	2
2	0	2	1

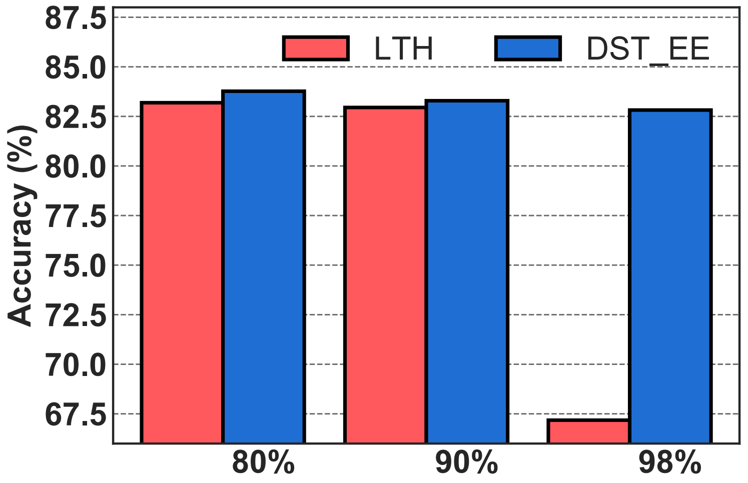
Dynamic Sparsity Training via Balancing the Exploration-Exploitation Trade-off (DST-EE) (ours)

Extreme Sparsity

GNN link prediction

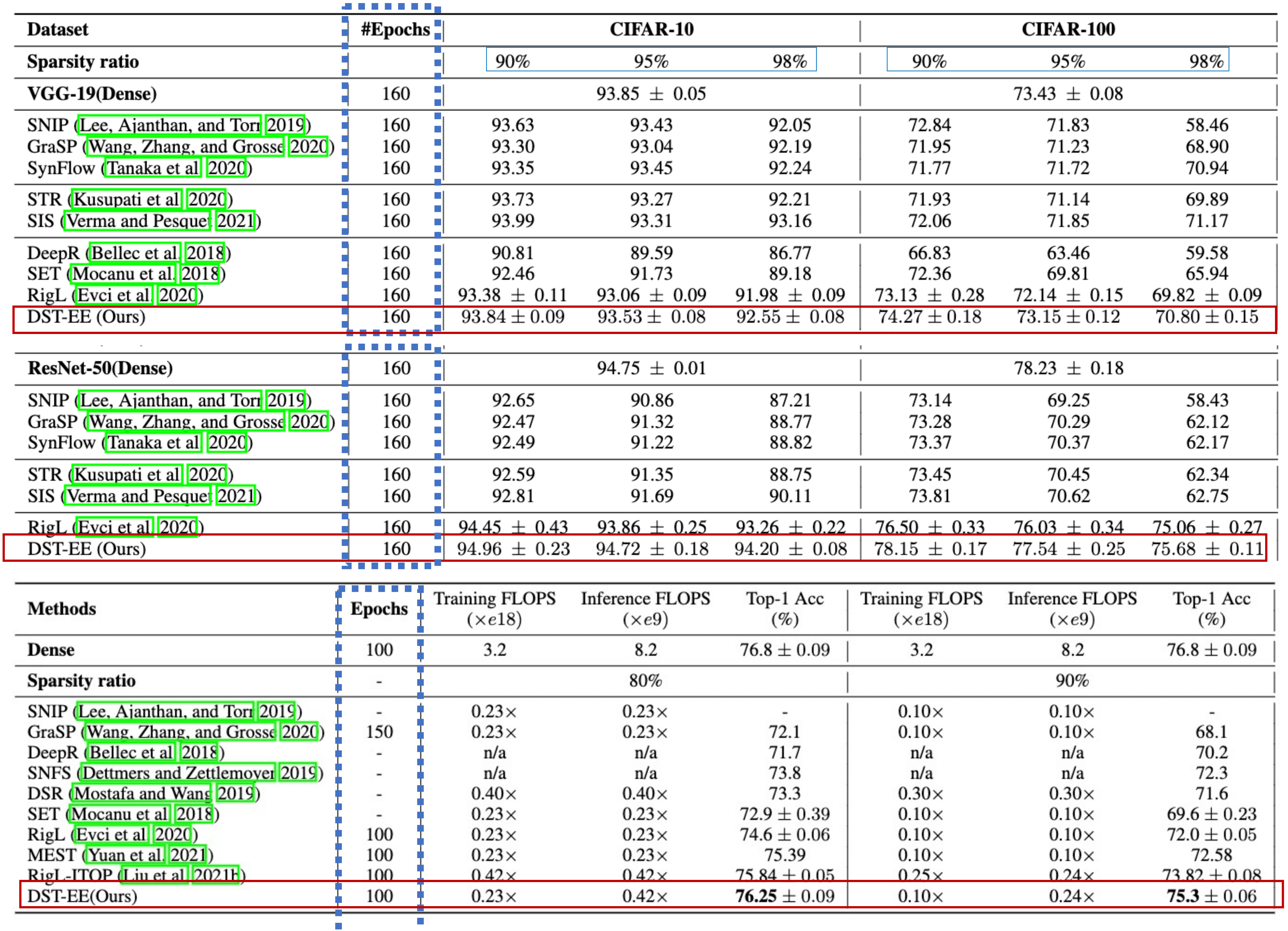


wiki-talk



ia-email

Convolutional neural network (CNN)



Dataset	#Epochs	CIFAR-10			CIFAR-100		
Sparsity ratio		90%	95%	98%	90%	95%	98%
VGG-19(Dense)	160	93.85 ± 0.05			73.43 ± 0.08		
SNIP (Lee, Ajanthan, and Torr 2019)	160	93.63	93.43	92.05	72.84	71.83	58.46
GraSP (Wang, Zhang, and Grosse 2020)	160	93.30	93.04	92.19	71.95	71.23	68.90
SynFlow (Tanaka et al. 2020)	160	93.35	93.45	92.24	71.77	71.72	70.94
STR (Kusupati et al. 2020)	160	93.73	93.27	92.21	71.93	71.14	69.89
SIS (Verma and Pesquet 2021)	160	93.99	93.31	93.16	72.06	71.85	71.17
DeepR (Bellec et al. 2018)	160	90.81	89.59	86.77	66.83	63.46	59.58
SET (Mocanu et al. 2018)	160	92.46	91.73	89.18	72.36	69.81	65.94
RigL (Evci et al. 2020)	160	93.38 ± 0.11	93.06 ± 0.09	91.98 ± 0.09	73.13 ± 0.28	72.14 ± 0.15	69.82 ± 0.09
DST-EE (Ours)	160	93.84 ± 0.09	93.53 ± 0.08	92.55 ± 0.08	74.27 ± 0.18	73.15 ± 0.12	70.80 ± 0.15
ResNet-50(Dense)	160	94.75 ± 0.01			78.23 ± 0.18		
SNIP (Lee, Ajanthan, and Torr 2019)	160	92.65	90.86	87.21	73.14	69.25	58.43
GraSP (Wang, Zhang, and Grosse 2020)	160	92.47	91.32	88.77	73.28	70.29	62.12
SynFlow (Tanaka et al. 2020)	160	92.49	91.22	88.82	73.37	70.37	62.17
STR (Kusupati et al. 2020)	160	92.59	91.35	88.75	73.45	70.45	62.34
SIS (Verma and Pesquet 2021)	160	92.81	91.69	90.11	73.81	70.62	62.75
RigL (Evci et al. 2020)	160	94.45 ± 0.43	93.86 ± 0.25	93.26 ± 0.22	76.50 ± 0.33	76.03 ± 0.34	75.06 ± 0.27
DST-EE (Ours)	160	94.96 ± 0.23	94.72 ± 0.18	94.20 ± 0.08	78.15 ± 0.17	77.54 ± 0.25	75.68 ± 0.11

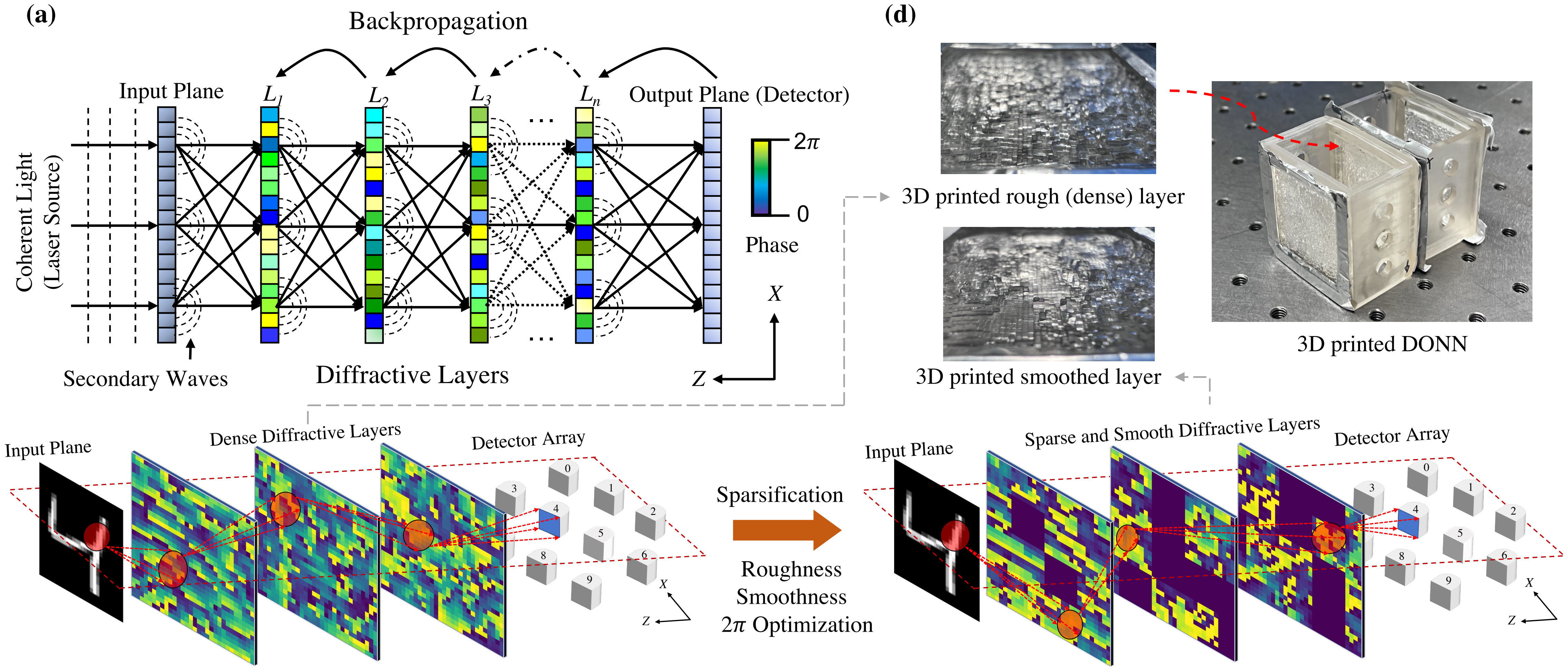
Methods	Epochs	Training FLOPS ($\times e18$)	Inference FLOPS ($\times e9$)	Top-1 Acc (%)	Training FLOPS ($\times e18$)	Inference FLOPS ($\times e9$)	Top-1 Acc (%)
Dense	100	3.2	8.2	76.8 ± 0.09	3.2	8.2	76.8 ± 0.09
Sparsity ratio	-		80%			90%	
SNIP (Lee, Ajanthan, and Torr 2019)	-	0.23×	0.23×	-	0.10×	0.10×	-
GraSP (Wang, Zhang, and Grosse 2020)	150	0.23×	0.23×	72.1	0.10×	0.10×	68.1
DeepR (Bellec et al. 2018)	-	n/a	n/a	71.7	n/a	n/a	70.2
SNFS (Dettmers and Zettlemoyer 2019)	-	n/a	n/a	73.8	n/a	n/a	72.3
DSR (Mostafa and Wang 2019)	-	0.40×	0.40×	73.3	0.30×	0.30×	71.6
SET (Mocanu et al. 2018)	-	0.23×	0.23×	72.9 ± 0.39	0.10×	0.10×	69.6 ± 0.23
RigL (Evci et al. 2020)	100	0.23×	0.23×	74.6 ± 0.06	0.10×	0.10×	72.0 ± 0.05
MEST (Yuan et al. 2021)	100	0.23×	0.23×	75.39	0.10×	0.10×	72.58
RigL-ITOP (Liu et al. 2021b)	100	0.42×	0.42×	75.84 ± 0.05	0.25×	0.24×	73.82 ± 0.08
DST-EE(Ours)	100	0.23×	0.42×	**76.25** ± 0.09	0.10×	0.24×	**75.3** ± 0.06

Table 2: Performance of sparse ResNet-50 models on ImageNet dataset. The results reported with (mean ± std) are run with three different random seeds.

Efficient DONN

- Challenges posed by high computation and memory storage of CMOS-based DNN systems.
- Diffractive optical neural networks (DONNs) as an ultra-efficient DNN accelerator
  - Mismatch between numerical modeling and physical deployment of current DONN systems

- Physics-aware roughness optimization process for DONN system
  - Improves accuracy awareness, reduces interpixel crosstalk impacts
  - Three steps: roughness regularization, block sparsification, 2π periodic phase modulation



(b) Train Baseline Model

(c) Train Sparse Model with Roughness and Smoothness

[23'DAC] Shanglin Zhou*, Yingjie Li*, Minhan Lou, Weilu Gao, Zhijie Shi, Cunxi Yu, **Caiwen Ding**, Physics-aware Roughness Optimization for Diffractive Optical Neural Networks, In Proceedings of ACM/EDAC/IEEE Design Automation Conference (DAC).

Efficient DONN

- Ours-A: roughness-aware trained model
- Ours-B: the model trained with sparsity
- Ours-C: the model trained with sparsity and roughness
- Ours-D: the model trained with sparsity, plus roughness and intra-block smoothness

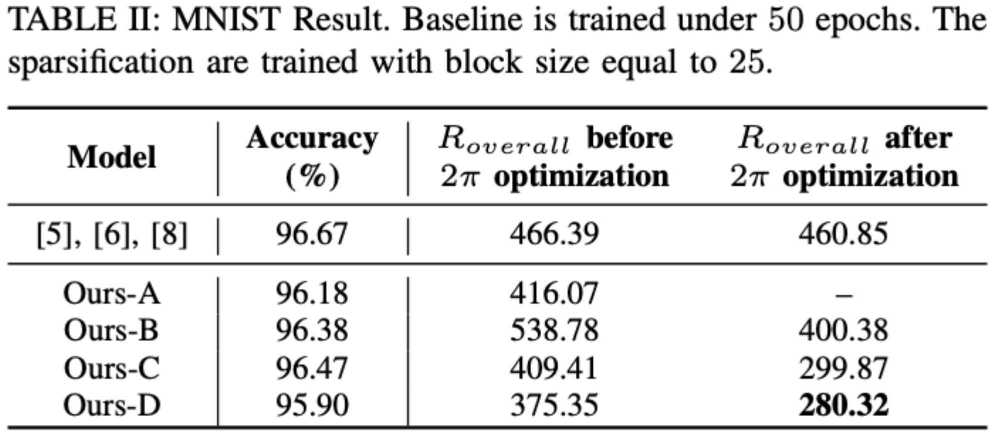


TABLE II: MNIST Result. Baseline is trained under 50 epochs. The sparsification are trained with block size equal to 25.

Model	Accuracy (%)	$R_{overall}$ before 2π optimization	$R_{overall}$ after 2π optimization
[5], [6], [8]	96.67	466.39	460.85
Ours-A	96.18	416.07	–
Ours-B	96.38	538.78	400.38
Ours-C	96.47	409.41	299.87
Ours-D	95.90	375.35	**280.32**

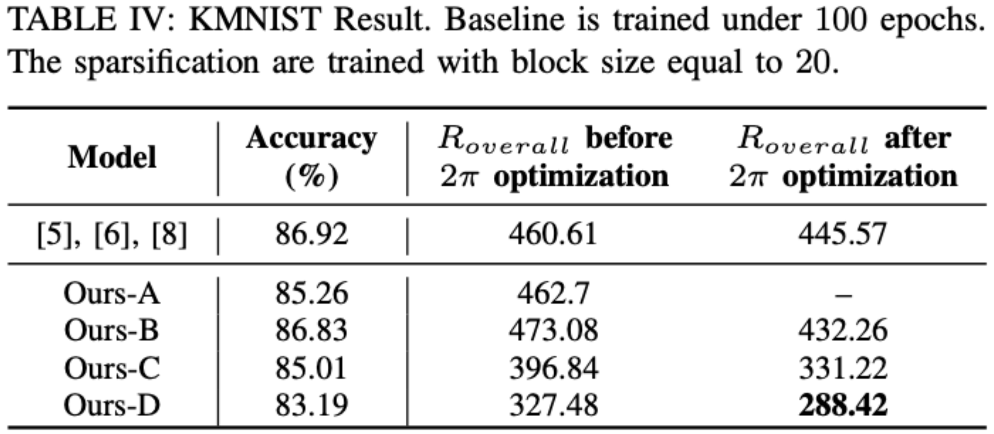


TABLE IV: KMNIST Result. Baseline is trained under 100 epochs. The sparsification are trained with block size equal to 20.

Model	Accuracy (%)	$R_{overall}$ before 2π optimization	$R_{overall}$ after 2π optimization
[5], [6], [8]	86.92	460.61	445.57
Ours-A	85.26	462.7	–
Ours-B	86.83	473.08	432.26
Ours-C	85.01	396.84	331.22
Ours-D	83.19	327.48	**288.42**

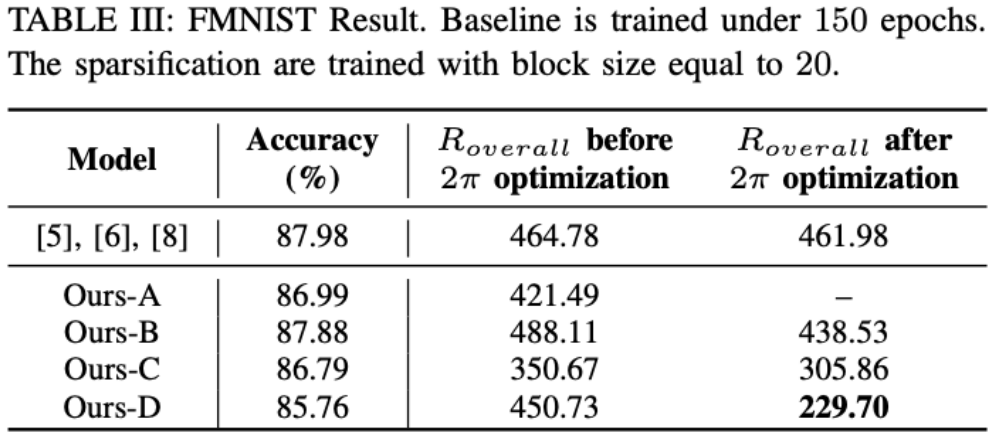


TABLE III: FMNIST Result. Baseline is trained under 150 epochs. The sparsification are trained with block size equal to 20.

Model	Accuracy (%)	$R_{overall}$ before 2π optimization	$R_{overall}$ after 2π optimization
[5], [6], [8]	87.98	464.78	461.98
Ours-A	86.99	421.49	–
Ours-B	87.88	488.11	438.53
Ours-C	86.79	350.67	305.86
Ours-D	85.76	450.73	**229.70**

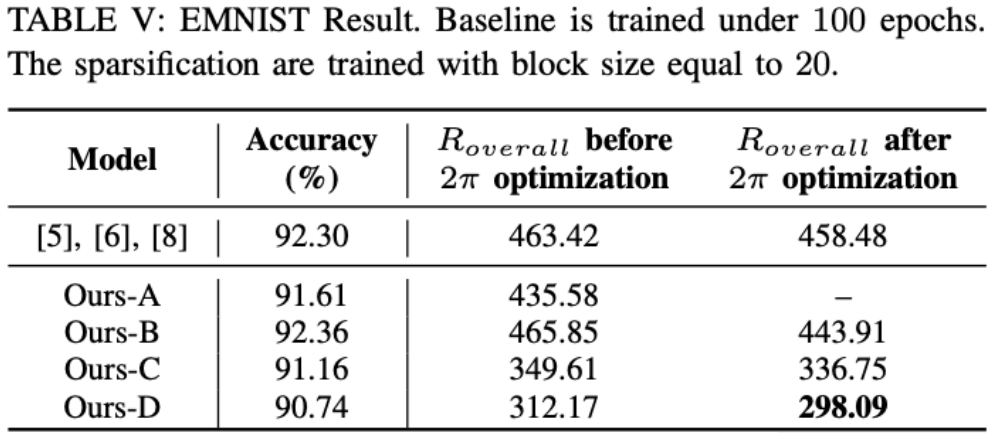


TABLE V: EMNIST Result. Baseline is trained under 100 epochs. The sparsification are trained with block size equal to 20.

Model	Accuracy (%)	$R_{overall}$ before 2π optimization	$R_{overall}$ after 2π optimization
[5], [6], [8]	92.30	463.42	458.48
Ours-A	91.61	435.58	–
Ours-B	92.36	465.85	443.91
Ours-C	91.16	349.61	336.75
Ours-D	90.74	312.17	**298.09**

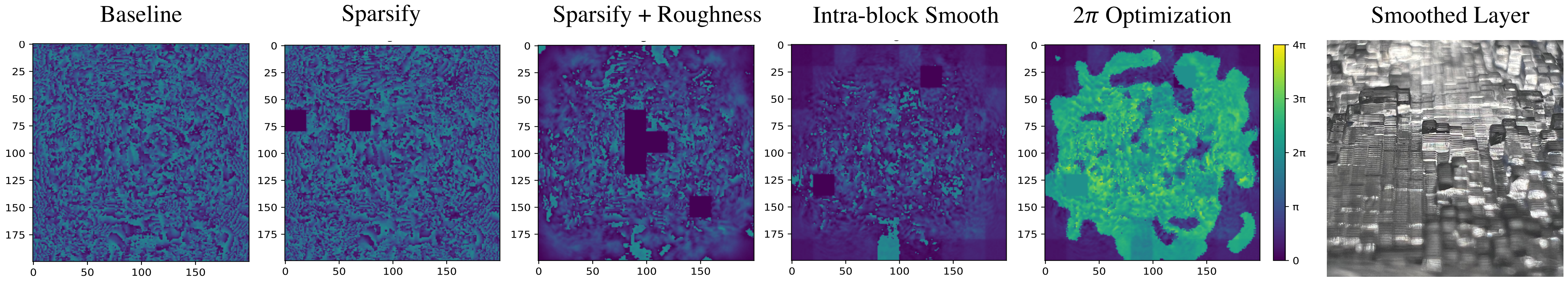
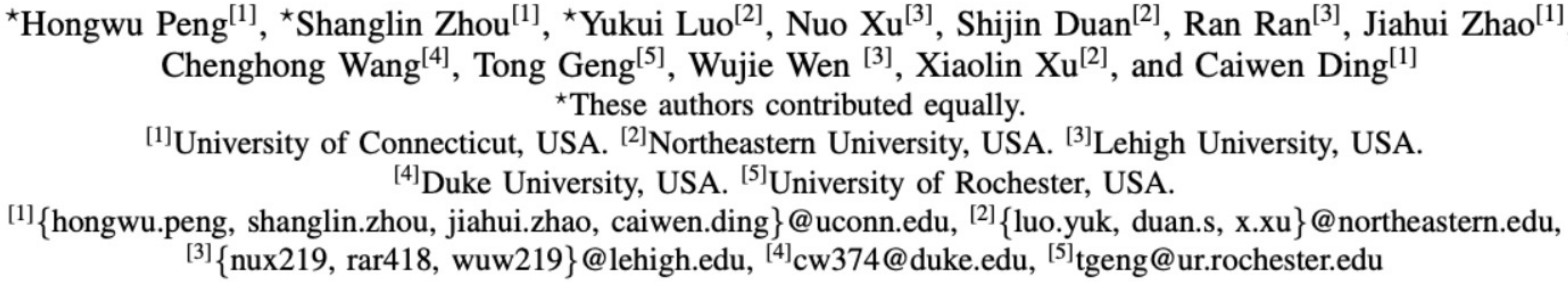


Fig: Comparison of the phase mask of the second diffractive layer under EMNIST dataset. Black blocks mean weights inside are sparsified and set to zero. The fifth is 2π optimization of phase mask that trained with sparsification, roughness and intra-block smoothness. The last is 3D printed smoothed diffractive layer.

PASNet: Polynomial Architecture Search Framework for Two-party Computation-based Secure Neural Network Deployment



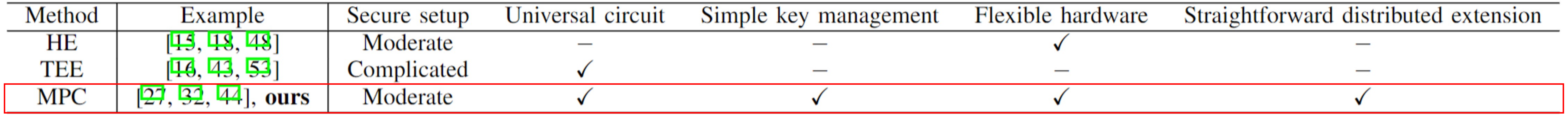
*Hongwu Peng[1], *Shanglin Zhou[1], *Yukui Luo[2], Nuo Xu[3], Shijin Duan[2], Ran Ran[3], Jiahui Zhao[1], Chenghong Wang[4], Tong Geng[5], Wujie Wen [3], Xiaolin Xu[2], and Caiwen Ding[1]
*These authors contributed equally.
[1]University of Connecticut, USA. [2]Northeastern University, USA. [3]Lehigh University, USA.
[4]Duke University, USA. [5]University of Rochester, USA.
[1]{hongwu.peng, shanglin.zhou, jiahui.zhao, caiwen.ding}@uconn.edu, [2]{luo.yuk, duan.s, x.xu}@northeastern.edu,
[3]{nux219, rar418, wuw219}@lehigh.edu, [4]cw374@duke.edu, [5]tgeng@ur.rochester.edu

Background

Trusted Execution Environment (TEE)

Multi-Party Computation (MPC)

Homomorphic Encryption (HE)

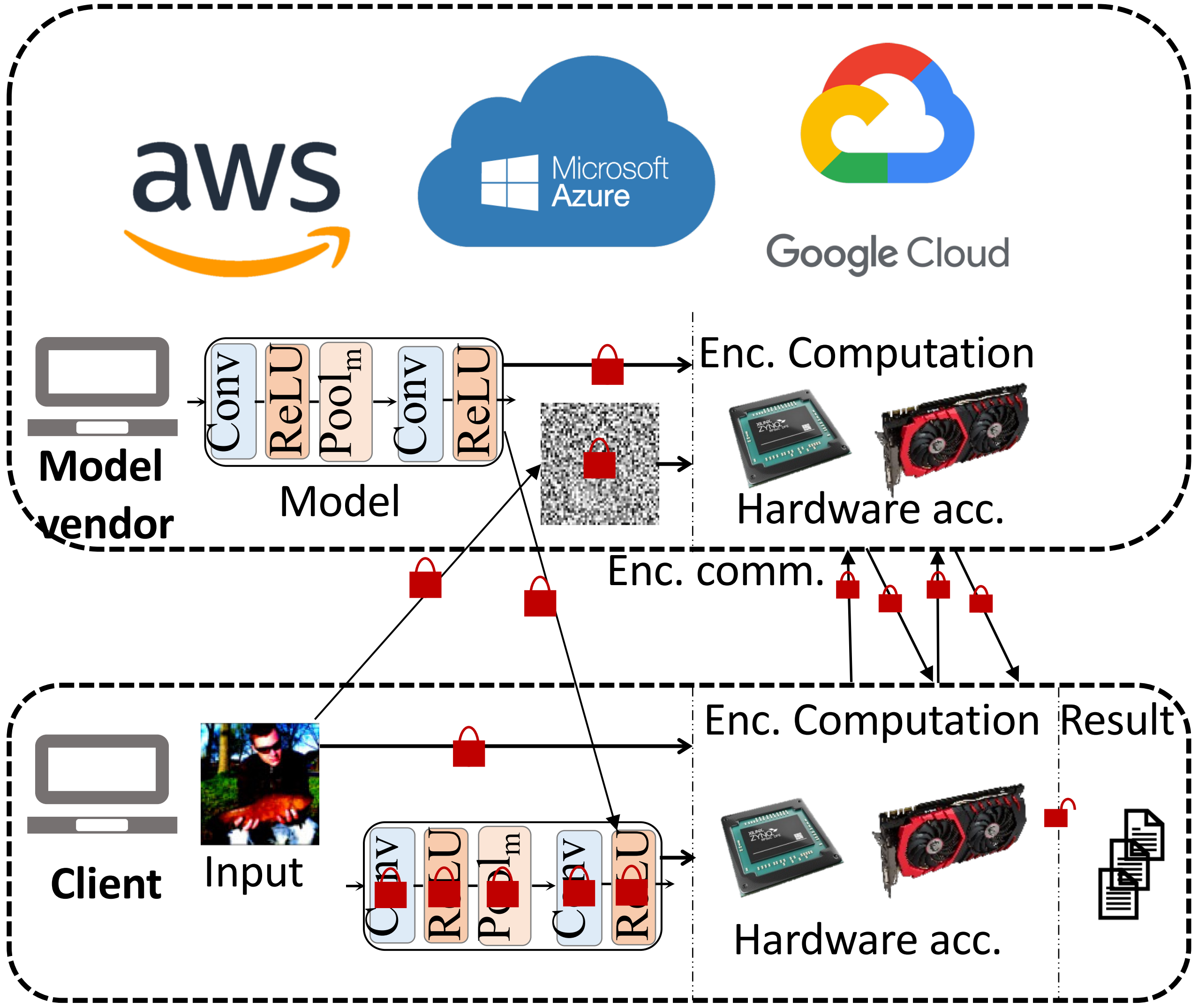


Method	Example	Secure setup	Universal circuit	Simple key management	Flexible hardware	Straightforward distributed extension
HE	[15, 18, 48]	Moderate	–	–	✓	–
TEE	[16, 43, 53]	Complicated	✓	–	–	–
MPC	[27, 32, 44], **ours**	Moderate	✓	✓	✓	✓

- HE applies operations on ciphertext while the result still can be recovered to plaintext;
- TEE constructs an environment that allow content inside works with confidentiality and integrity;
- MPC manages sensitive operations and communications on multiple parties while maintaining the security among each party

[15] Cryptonets [18] Cryptodl
[16] Mlcapsule [43] Visor [53] Slalom
[27] Mp-spdz [32] Cryptflow [44] Cryptflow2

Hardware Solution – GPU vs. CPU



Secure Machine-Learning-As-A-Service (MLaaS), MPC setting

HE:

- High cost multiplicative gates for DNN inference.
- >100s per image for a 10-layer SqueezeNet on CIFAR10 [1]. Intel Xeon E5- 2667v3@3.2GHz with 224 GB of memory

MPC: CryptGPU [2]

- achieves takes ~2s (VGG-16, CIFAR10) on Tesla V100 GPU with 16 GB memory
- 2.3× faster than the CRYPTFLOW [3] (CPU). (ResNet-152, ImageNet)
- Two CryptGPU with power budget of 315 × 2 Watt
- Still expensive for non-linear operators such as ReLU

FPGA advantages:

Power efficiency: computational efficient compared to CPUs and GPUs

Design flexibility: Rich reconfigurable gate-level hardware resources

[1] Dathathri et al, 2019 ASPLOS.
[2] Tan et al, 2021 Oakland
[3] Kumar et al, 2020 Oakland

Privacy Preserving Machine Learning - Research Gap

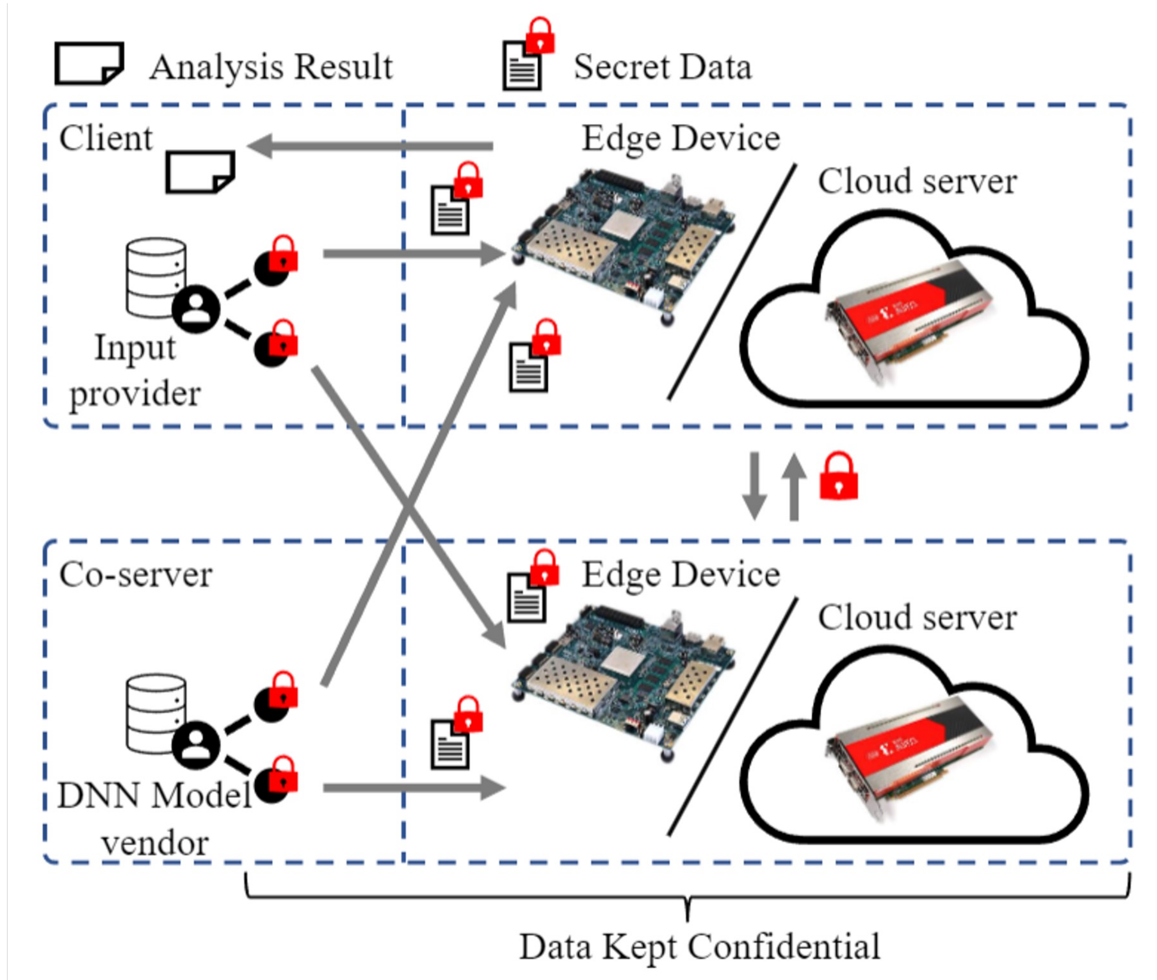
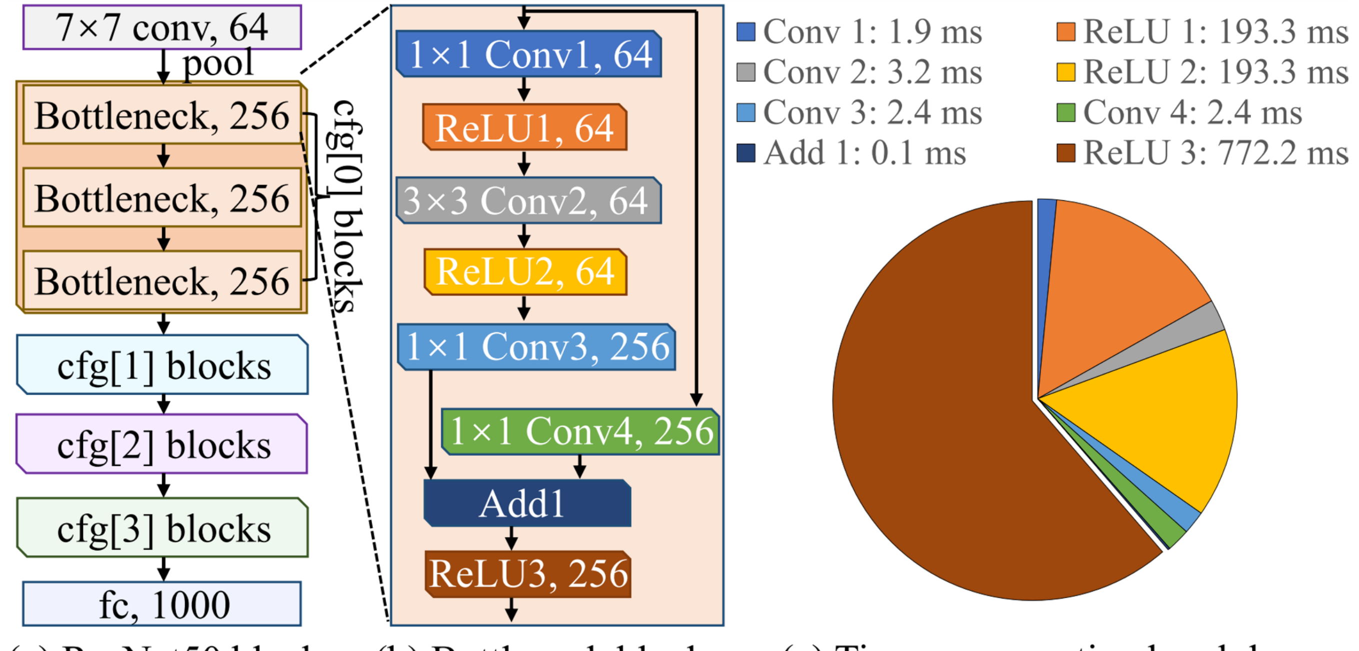


Fig. 1: Two-party secure computing (2PC) deployment.

A generic framework that combines the aforementioned FPGA advantages when both DNN inference and MPC are required, has not yet been widely investigated.

(1) ultra-high computation and communication overhead;
- ➢ vast amount of data communication between the edge devices and cloud,
- ➢ limited resources (e.g., memory size) on the edge

(2) Adding cryptographic operations introduces more overhead.

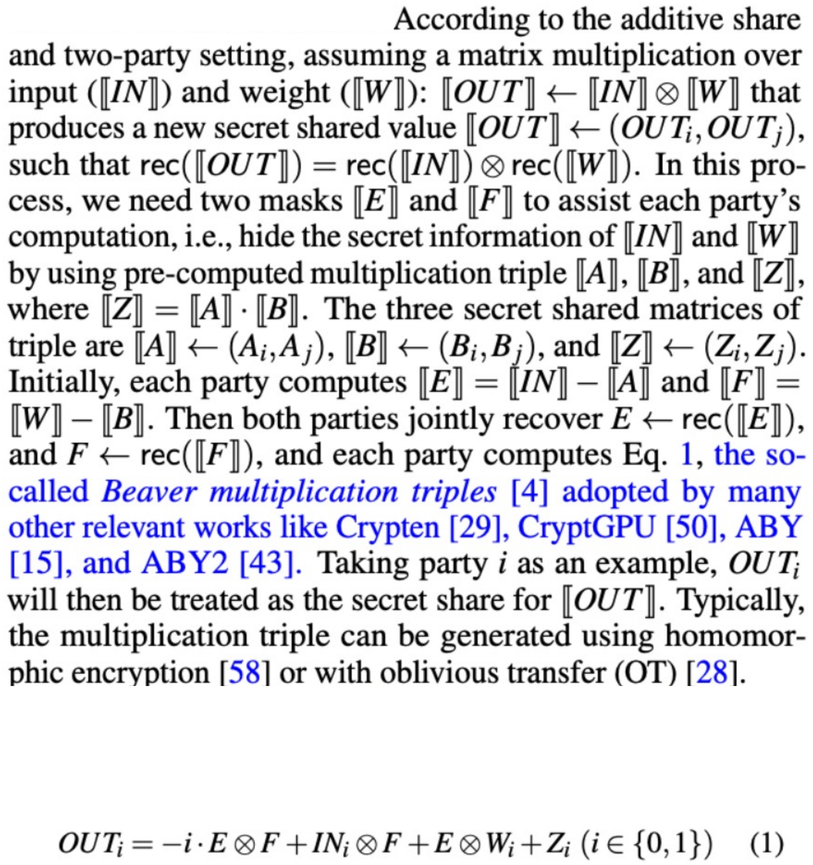


(a) ResNet50 blocks (b) Bottleneck block (c) Time consumption breakdown

Lantecy breakdown under 2PC PI setup. Network banwidth: 1 GB/s. Device: ZCU104. ResNet50 on ImageNet

- ❖ ReLU operator has much longer latency than Conv
- ❖ Hardware & software techniques are required for secure privarate inference (PI) acceleration.

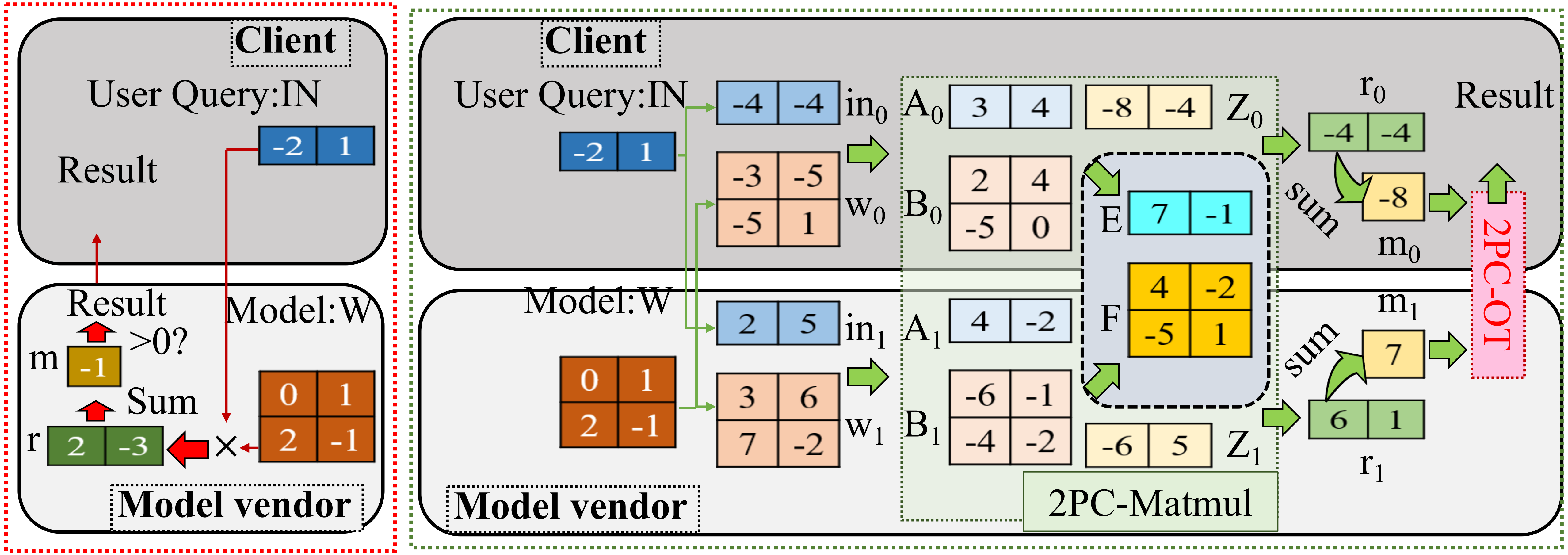
Background: MPC Basics



According to the additive share and two-party setting, assuming a matrix multiplication over input ($[\![IN]\!]$) and weight ($[\![W]\!]$): $[\![OUT]\!] \leftarrow [\![IN]\!] \otimes [\![W]\!]$ that produces a new secret shared value $[\![OUT]\!] \leftarrow (OUT_i, OUT_j)$, such that $\text{rec}([\![OUT]\!]) = \text{rec}([\![IN]\!]) \otimes \text{rec}([\![W]\!])$. In this process, we need two masks $[\![E]\!]$ and $[\![F]\!]$ to assist each party's computation, i.e., hide the secret information of $[\![IN]\!]$ and $[\![W]\!]$ by using pre-computed multiplication triple $[\![A]\!]$, $[\![B]\!]$, and $[\![Z]\!]$, where $[\![Z]\!] = [\![A]\!] \cdot [\![B]\!]$. The three secret shared matrices of triple are $[\![A]\!] \leftarrow (A_i, A_j)$, $[\![B]\!] \leftarrow (B_i, B_j)$, and $[\![Z]\!] \leftarrow (Z_i, Z_j)$. Initially, each party computes $[\![E]\!] = [\![IN]\!] - [\![A]\!]$ and $[\![F]\!] = [\![W]\!] - [\![B]\!]$. Then both parties jointly recover $E \leftarrow \text{rec}([\![E]\!])$, and $F \leftarrow \text{rec}([\![F]\!])$, and each party computes Eq. 1, the so-called *Beaver multiplication triples* [4] adopted by many other relevant works like Crypten [29], CryptGPU [50], ABY [15], and ABY2 [43]. Taking party i as an example, OUT_i will then be treated as the secret share for $[\![OUT]\!]$. Typically, the multiplication triple can be generated using homomorphic encryption [58] or with oblivious transfer (OT) [28].

$$OUT_i = -i \cdot E \otimes F + IN_i \otimes F + E \otimes W_i + Z_i \ (i \in \{0,1\}) \quad (1)$$

An example of 4-bit plaintext vs. ciphertext evaluation.

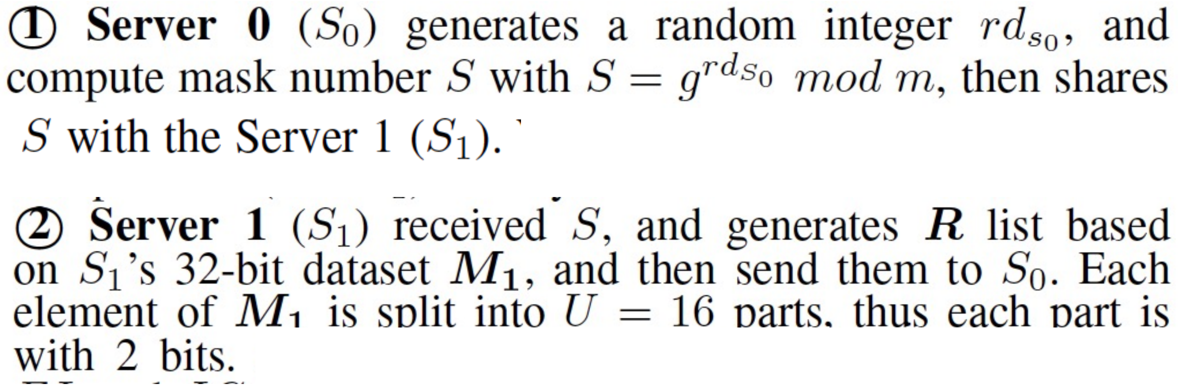


Plaintext Evaluation, user query revealed to vendor ✗ Not safe!

Ciphertext Evaluation, protection for both user query & vendor's model ✓ Safe!

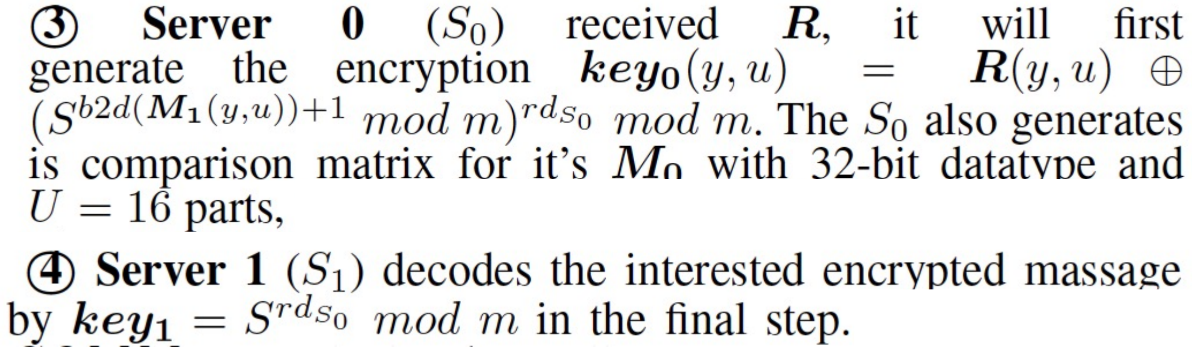
- 2PC-Oblivious Transfer (OT) -- foundation of secure comparison (ReLU)
- Multiple rounds of communication
- Reducing ReLU is essential

2PC-OT Processing Flow:



① **Server 0** (S_0) generates a random integer rd_{s0}, and compute mask number S with $S = g^{rd_{s_0}}\ mod\ m$, then shares S with the Server 1 (S_1).

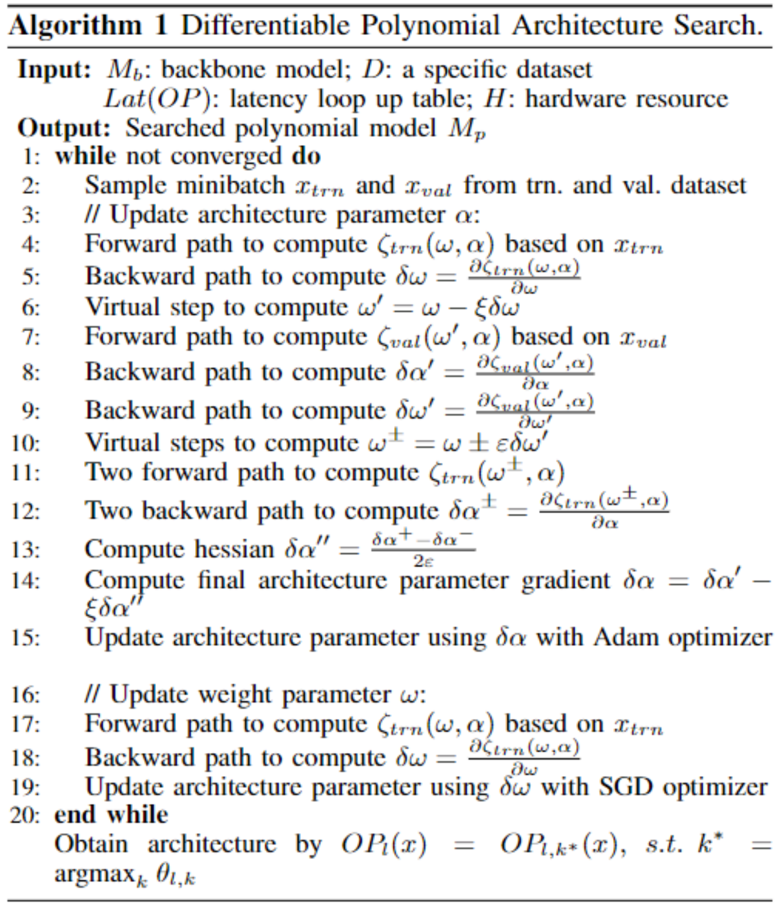
② **Server 1** (S_1) received S, and generates $\boldsymbol{R}$ list based on S_1's 32-bit dataset $\boldsymbol{M_1}$, and then send them to S_0. Each element of $\boldsymbol{M_1}$ is split into $U = 16$ parts, thus each part is with 2 bits.



③ **Server 0** (S_0) received $\boldsymbol{R}$, it will first generate the encryption $\boldsymbol{key_0}(y,u) = \boldsymbol{R}(y,u) \oplus (S^{b2d(\boldsymbol{M_1}(y,u))+1}\ mod\ m)^{rd_{s_0}}\ mod\ m$. The S_0 also generates is comparison matrix for it's $\boldsymbol{M_0}$ with 32-bit datatype and $U = 16$ parts,

④ **Server 1** (S_1) decodes the interested encrypted massage by $\boldsymbol{key_1} = S^{rd_{s_0}}\ mod\ m$ in the final step.

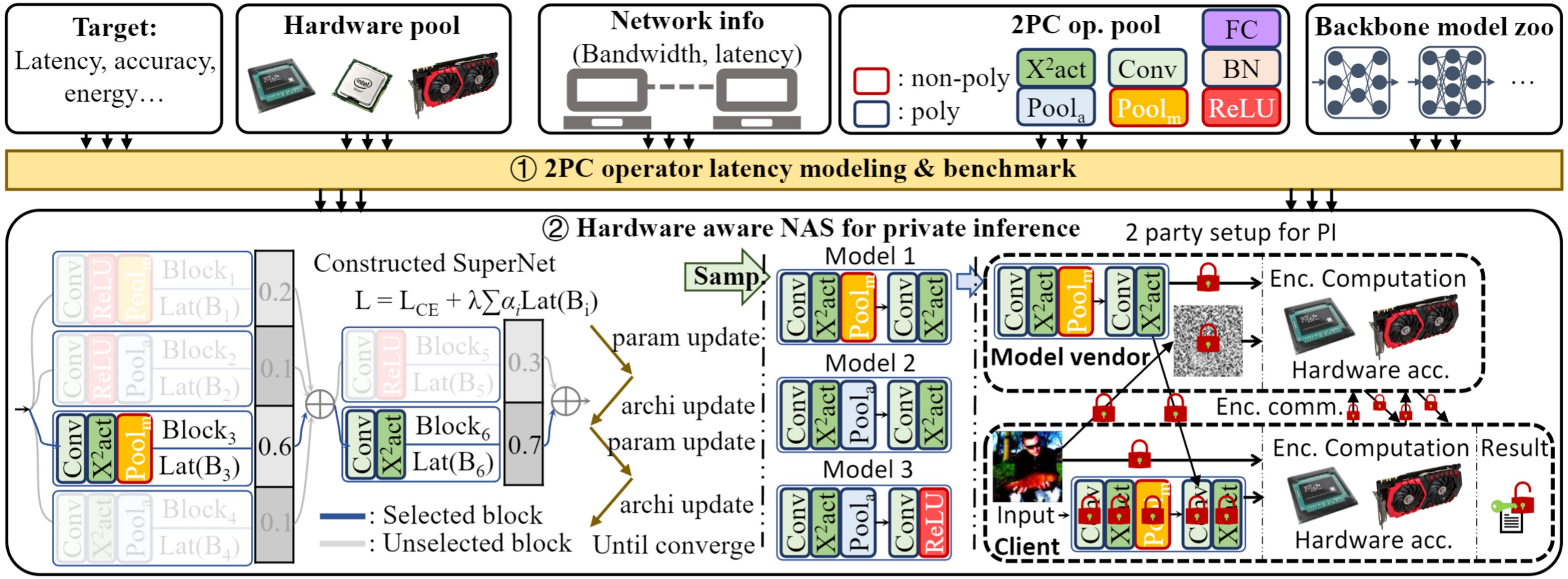
Crypto-Friendly Neural Architecture Co-search



Algorithm 1 Differentiable Polynomial Architecture Search.

```
Input: $M_b$: backbone model; $D$: a specific dataset
       $Lat(OP)$: latency loop up table; $H$: hardware resource
Output: Searched polynomial model $M_p$
while not converged do
    Sample minibatch $x_{trn}$ and $x_{val}$ from trn. and val. dataset
    // Update architecture parameter $\alpha$:
    Forward path to compute $\zeta_{trn}(\omega, \alpha)$ based on $x_{trn}$
    Backward path to compute $\delta\omega = \frac{\partial \zeta_{trn}(\omega,\alpha)}{\partial \omega}$
    Virtual step to compute $\omega' = \omega - \xi\delta\omega$
    Forward path to compute $\zeta_{val}(\omega', \alpha)$ based on $x_{val}$
    Backward path to compute $\delta\alpha' = \frac{\partial \zeta_{val}(\omega',\alpha)}{\partial \alpha}$
    Backward path to compute $\delta\omega' = \frac{\partial \zeta_{val}(\omega',\alpha)}{\partial \omega'}$
    Virtual steps to compute $\omega^{\pm} = \omega \pm \varepsilon\delta\omega'$
    Two forward path to compute $\zeta_{trn}(\omega^{\pm}, \alpha)$
    Two backward path to compute $\delta\alpha^{\pm} = \frac{\partial \zeta_{trn}(\omega^{\pm},\alpha)}{\partial \alpha}$
    Compute hessian $\delta\alpha'' = \frac{\delta\alpha^{+} - \delta\alpha^{-}}{2\varepsilon}$
    Compute final architecture parameter gradient $\delta\alpha = \delta\alpha' - \xi\delta\alpha''$
    Update architecture parameter using $\delta\alpha$ with Adam optimizer

    // Update weight parameter $\omega$:
    Forward path to compute $\zeta_{trn}(\omega, \alpha)$ based on $x_{trn}$
    Backward path to compute $\delta\omega = \frac{\partial \zeta_{trn}(\omega,\alpha)}{\partial \omega}$
    Update architecture parameter using $\delta\omega$ with SGD optimizer
end while
Obtain architecture by $OP_l(x) = OP_{l,k^*}(x)$, s.t. $k^* = \text{argmax}_k\, \theta_{l,k}$
```



- ❖ The search algorithm uses hardware latency modeling as input for loss function
- ❖ Trainable X2act Non-linear Function
- ❖ Operator Modeling and Latency Analysis

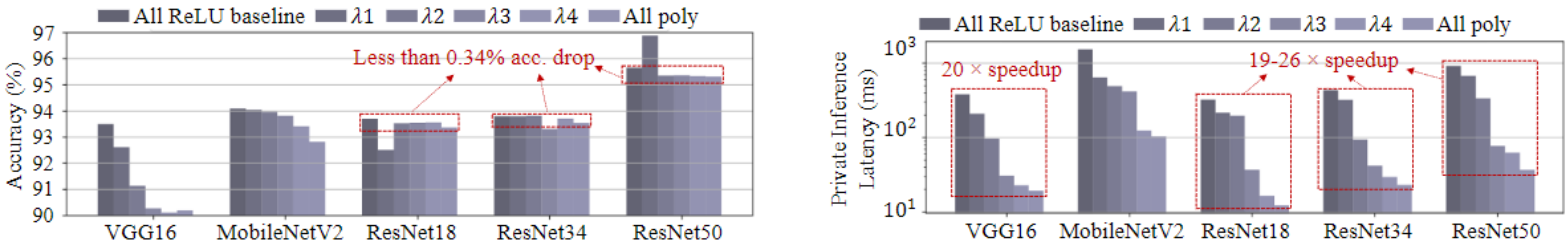
2PC-MaxPool Operator

2PC-X2act Operator

2PC-AvgPool Operator

2PC-Conv Operator

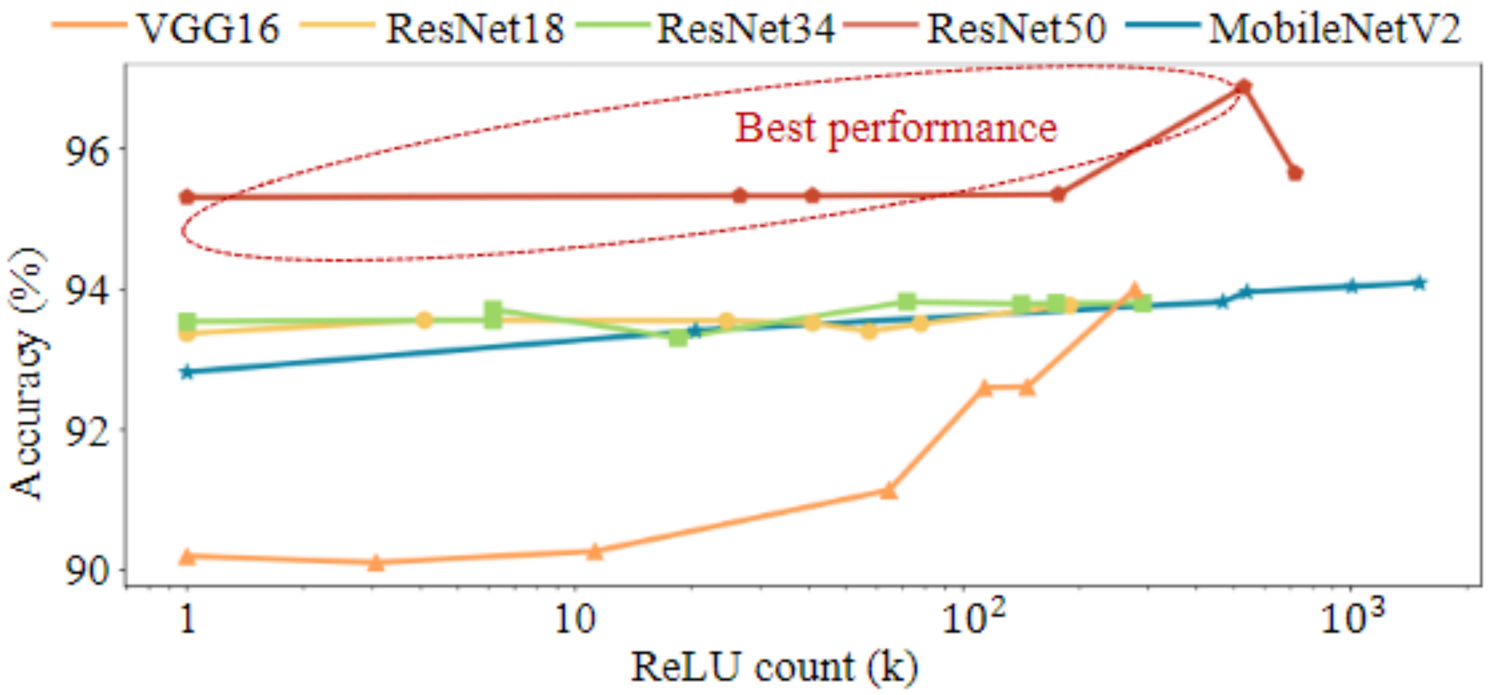
Results



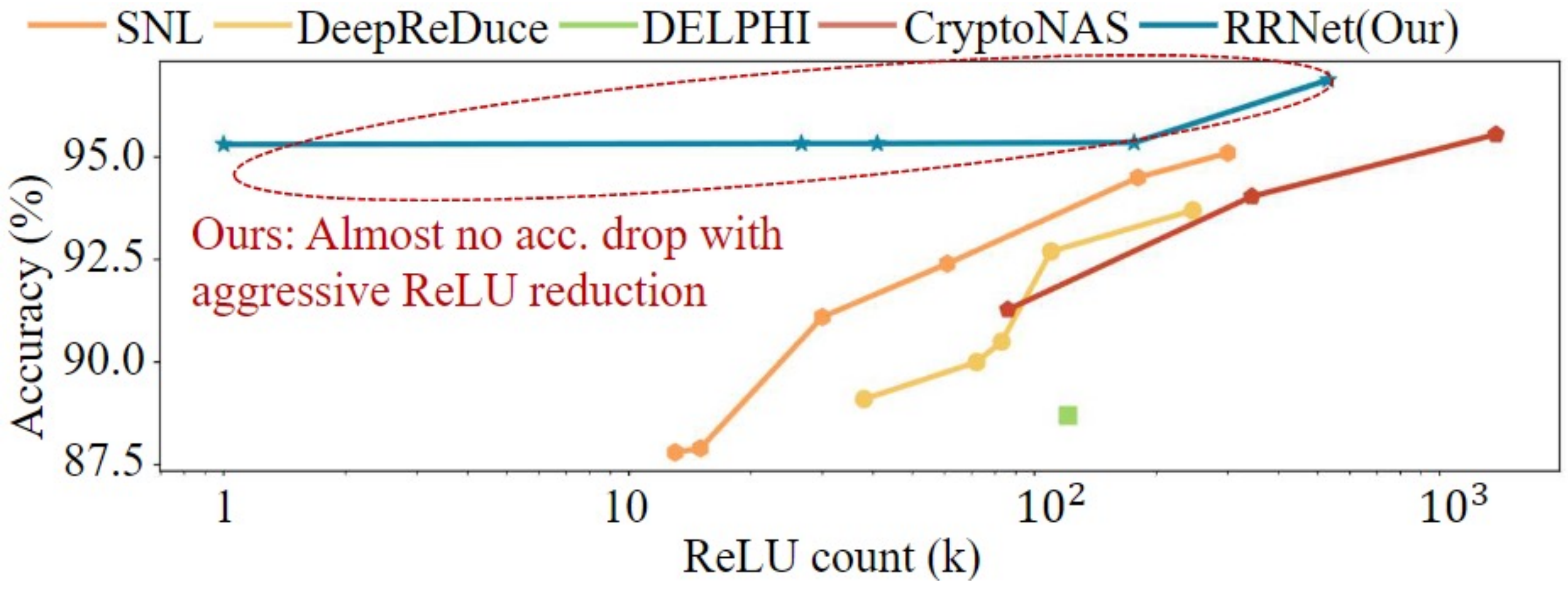
(a) Searched model accuracy comparison

(b) Searched model private inference latency comparison

PASNet on CIFAR-10 under 2PC PI setup. Network banwidth: 1 GB/s. Device: ZCU104.

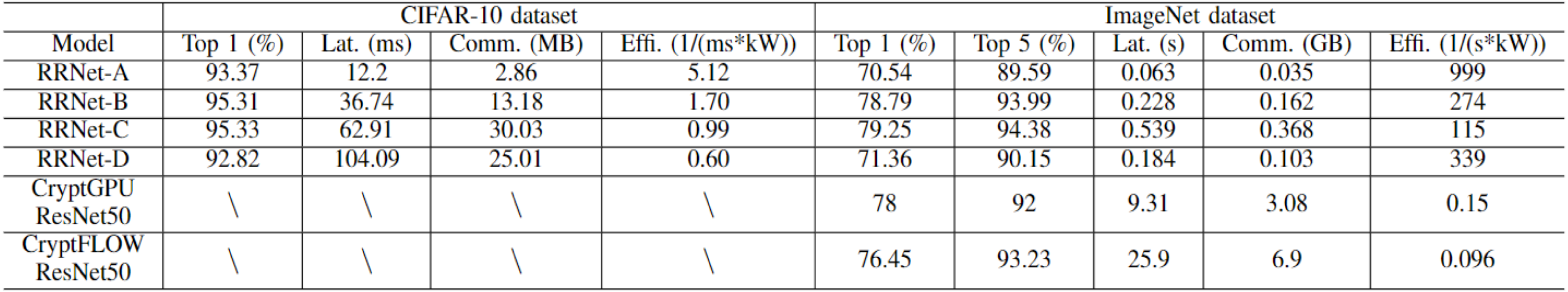


Accuracy-ReLU count trade-off on CIFAR-10.



ReLU reduction comparison with SOTA works on CIFAR-10.

Results



	CIFAR-10 dataset				ImageNet dataset				
Model	Top 1 (%)	Lat. (ms)	Comm. (MB)	Effi. (1/(ms*kW))	Top 1 (%)	Top 5 (%)	Lat. (s)	Comm. (GB)	Effi. (1/(s*kW))
RRNet-A	93.37	12.2	2.86	5.12	70.54	89.59	0.063	0.035	999
RRNet-B	95.31	36.74	13.18	1.70	78.79	93.99	0.228	0.162	274
RRNet-C	95.33	62.91	30.03	0.99	79.25	94.38	0.539	0.368	115
RRNet-D	92.82	104.09	25.01	0.60	71.36	90.15	0.184	0.103	339
CryptGPU ResNet50	\	\	\	\	78	92	9.31	3.08	0.15
CryptFLOW ResNet50	\	\	\	\	76.45	93.23	25.9	6.9	0.096

PASNet evaluation & cross-work comparison with CryptGPU and CryptFLOW. Batch size = 1, Network bandwidth: 1 GB/s. Device: ZCU104.

CryptGPU: Tan et al, 2021 Oakland
CryptFLOW: Kumar et al, 2020 Oakland

Acknowledgement: Team Members

Ph.D. Students

Bingbing Li

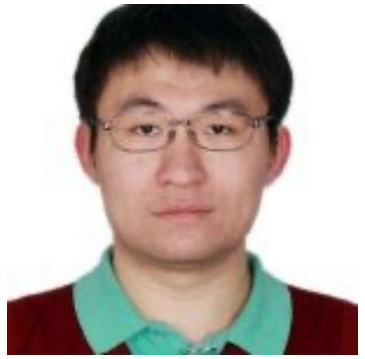
Shanglin Zhou

Shaoyi Huang

Hongwu Peng

Amit Hasan

Ronuk Sahu



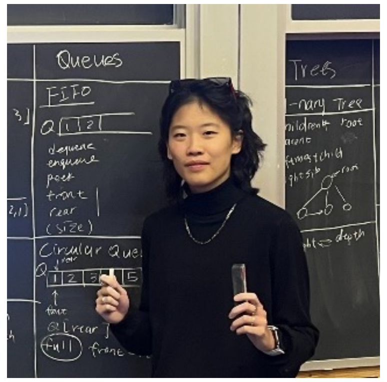
Xi Xie

Kiran Thorat

Jiahui Zhao

Bin Lei

Yuebo Luo

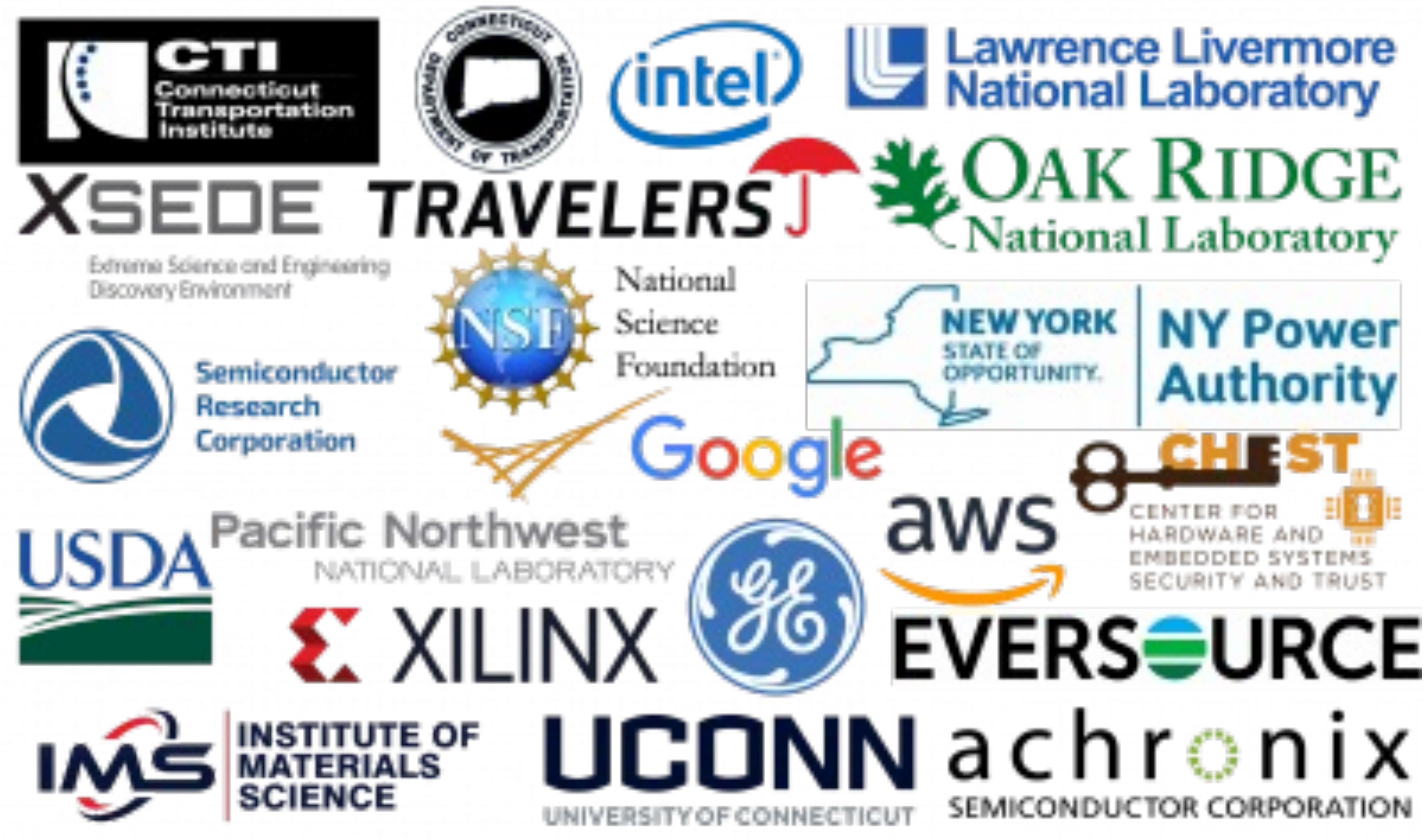


Nicole Meng

Master Students

Ya-Sine Agrignan

Acknowledgement


CTI Connecticut Transportation Institute
Connecticut Department of Transportation
intel
Lawrence Livermore National Laboratory
XSEDE Extreme Science and Engineering Discovery Environment
TRAVELERS
OAK RIDGE National Laboratory
NSF National Science Foundation
Semiconductor Research Corporation
NEW YORK STATE OF OPPORTUNITY.
NY Power Authority
Google
CHEST CENTER FOR HARDWARE AND EMBEDDED SYSTEMS SECURITY AND TRUST
USDA
Pacific Northwest NATIONAL LABORATORY
aws
XILINX
EVERSOURCE
IMS INSTITUTE OF MATERIALS SCIENCE
UCONN UNIVERSITY OF CONNECTICUT
achronix SEMICONDUCTOR CORPORATION

Thank you!

caiwen.ding@uconn.edu