Big AI for Small Devices

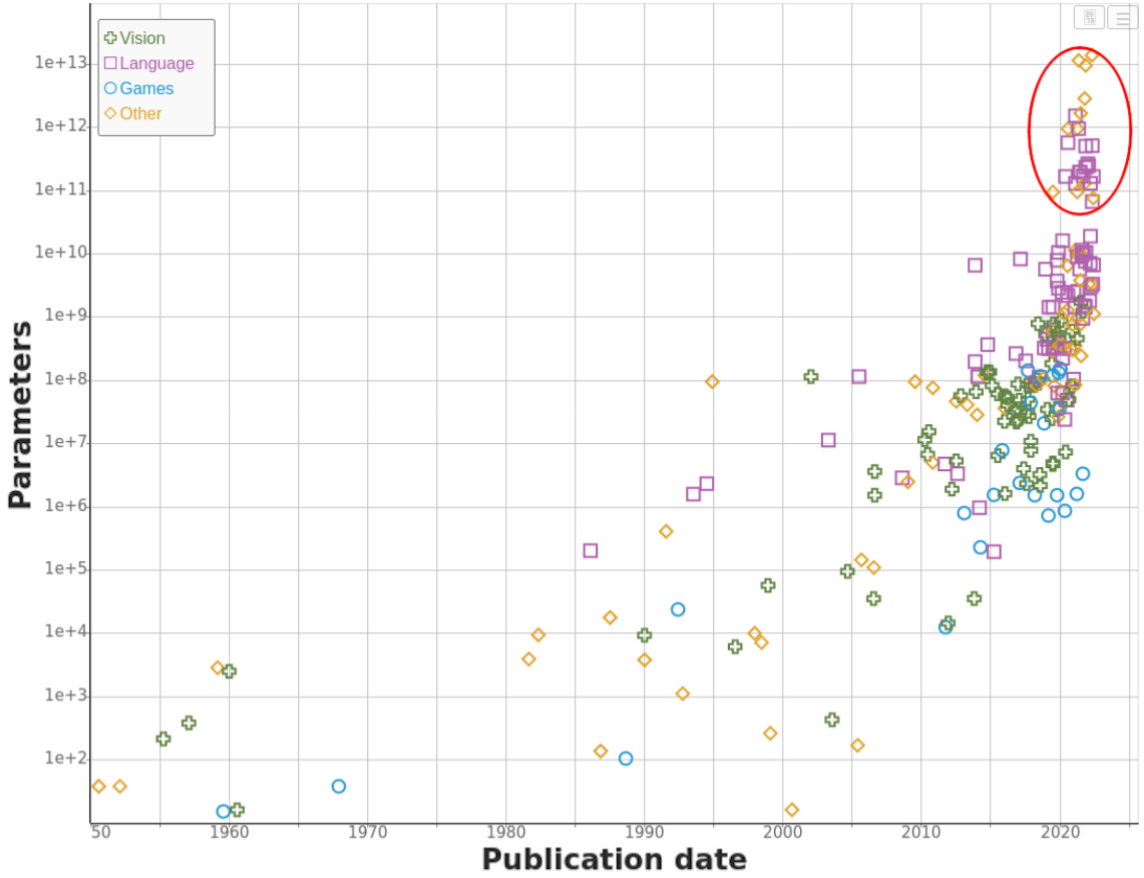
Yiran Chen

Department of Electrical and Computer Engineering, Duke University
Duke University Center for Computational Evolutionary Intelligence (CEI)
NSF IUCRC For Alternative Sustainable and Intelligence Computing (ASIC)
NSF AI Institute for Edge Computing Leveraging Next-generation Networks (Athena)

Duke

Big AI

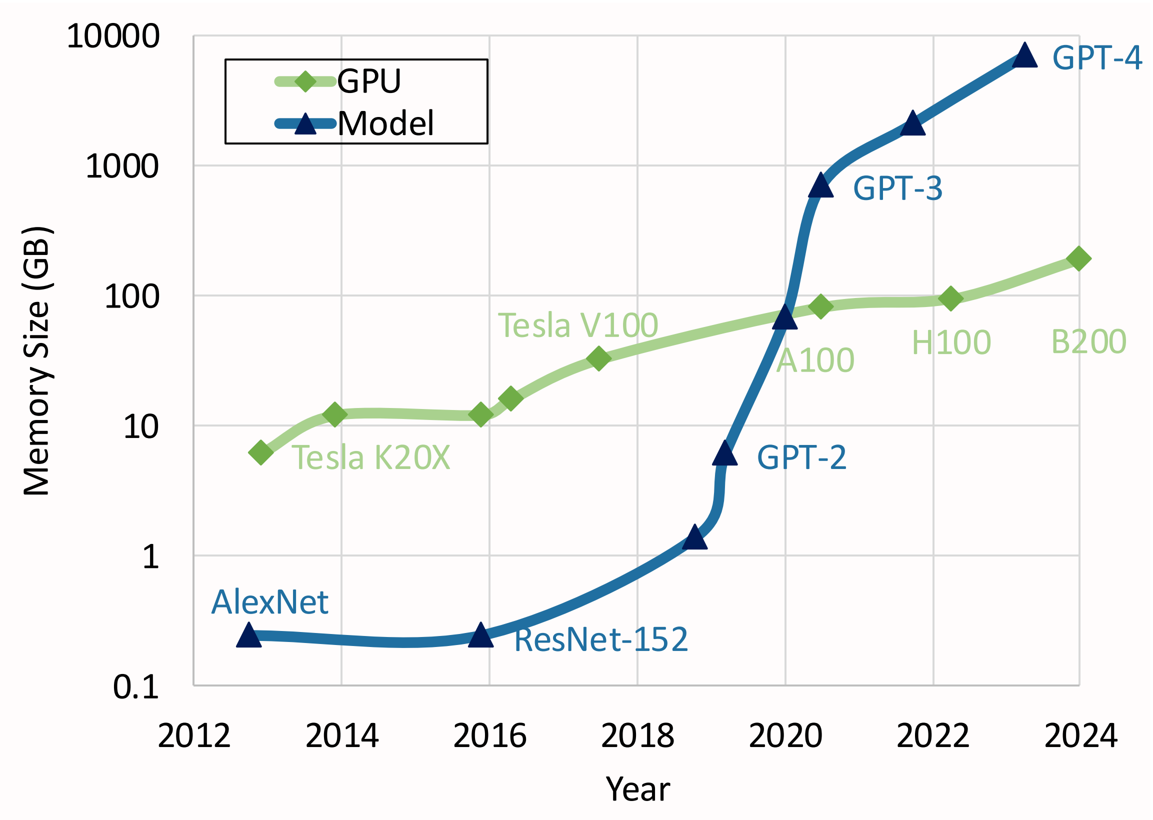
- **We are in the era of Big AI!**
  - **BERT-Base** (110M) may run on your mobile phone.
  - **LLaMA-7B** (7B) can run on your laptop with some optimizations.
  - **Turing NLG** (17B); You will need a powerful workstation.
  - **GPT-3** (175B); You will definitely need a powerful computing server.
  - **DeepSeek-R1** (671B); You will need a data center or cloud supercomputing cluster.

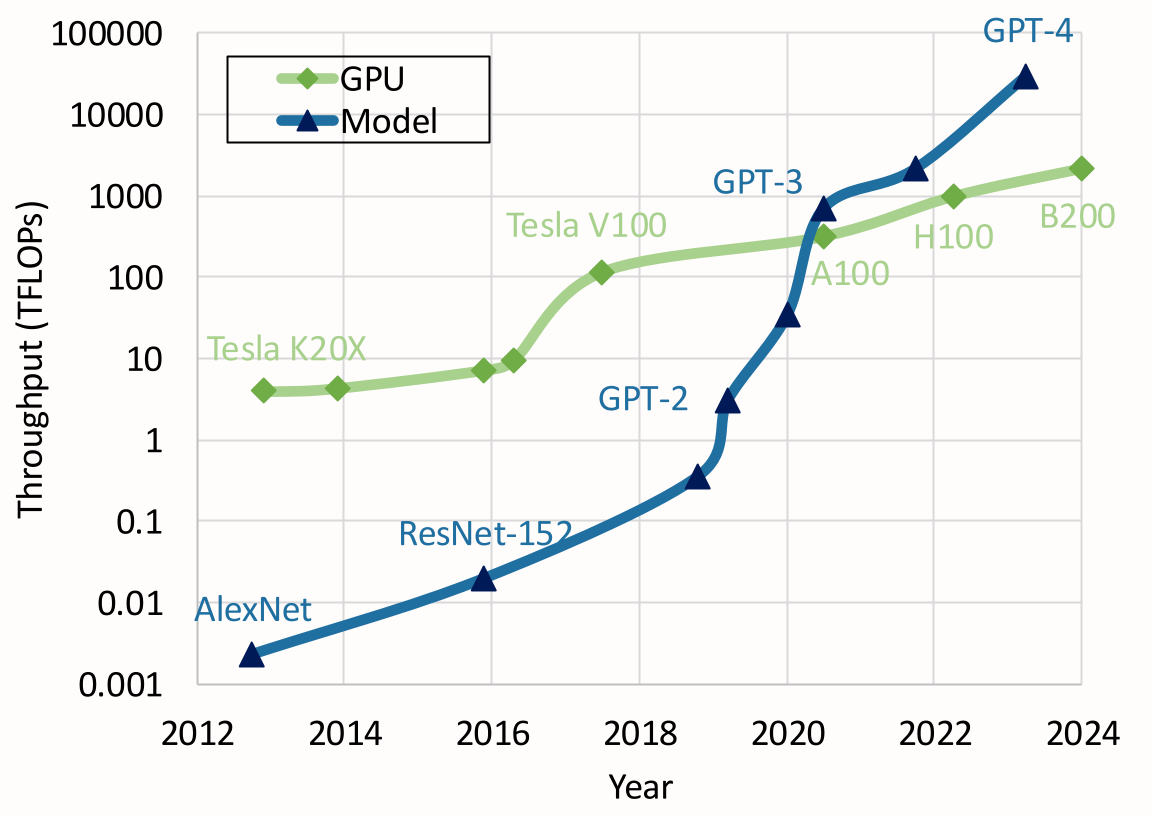


Villalobos, Pablo, et al. "Machine learning model sizes and the parameter gap." *arXiv preprint arXiv:2207.02852* (2022).

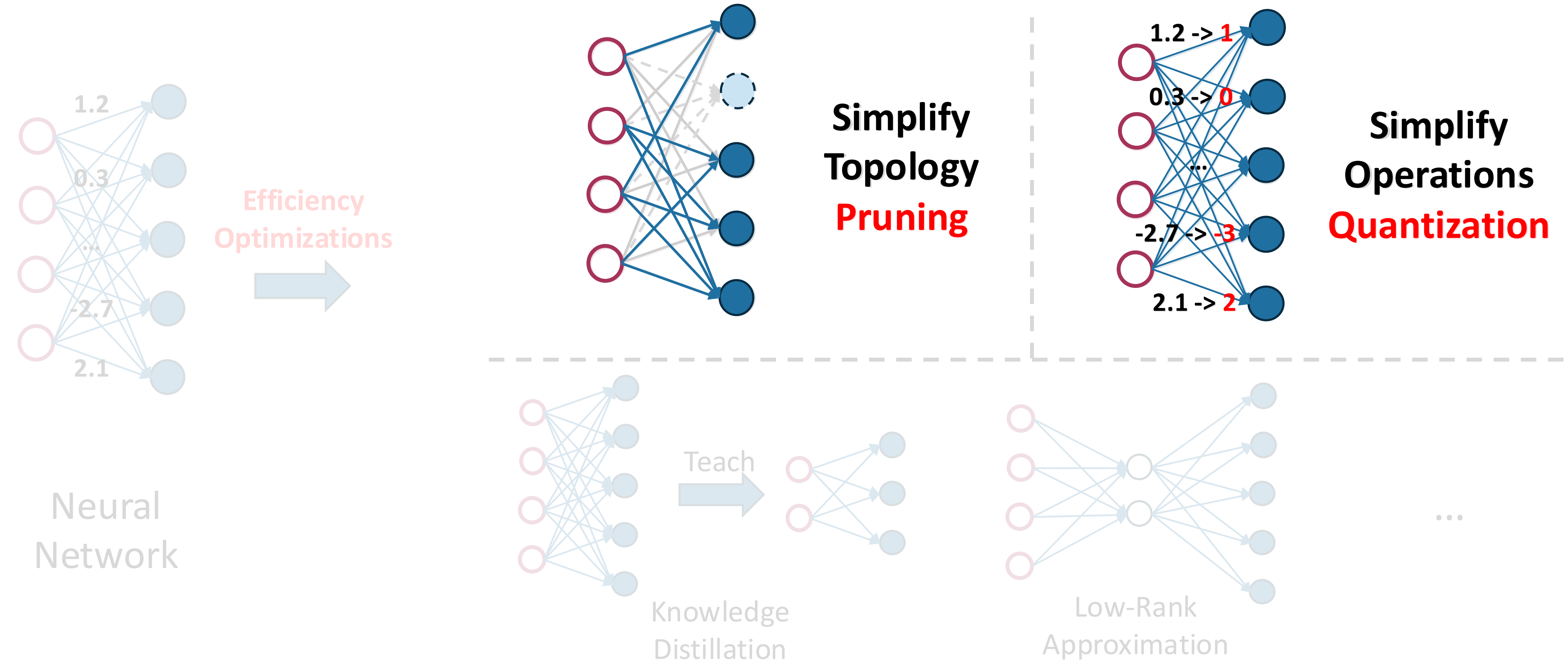
Small Devices

- **But the most powerful GPU failed to catch up with the pace.**





Towards Efficient AI…

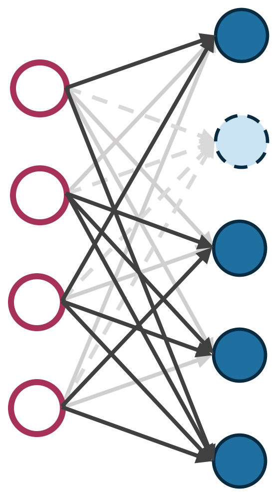


Basic Idea of Pruning

- Redundant parameters in neural networks can be safely removed, i.e., setting them to zero.

$$x \times 0 = 0, \qquad x + 0 = x$$

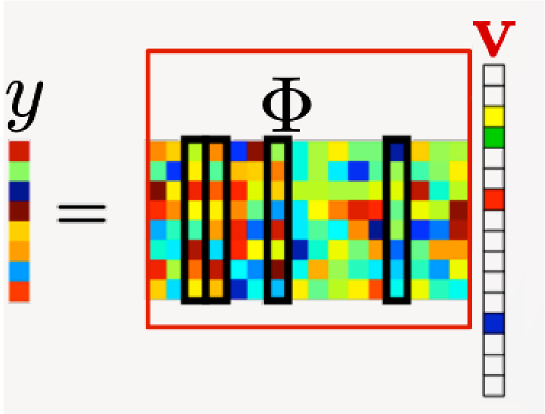
- If an operand is zero, the result is known. No need to perform the corresponding calculation.
  - Skipping the storage of zero values to save **memory**.
  - Skipping the computation with zero operands to reduce **latency** and **energy usage**.



Duke

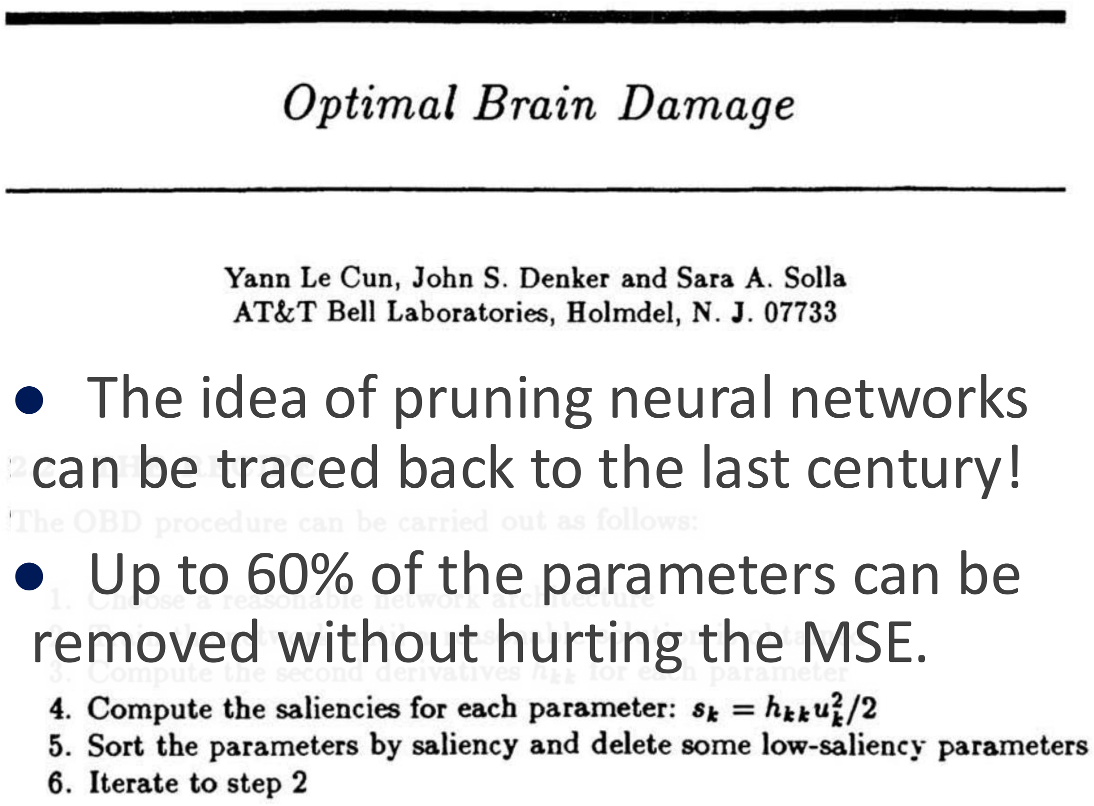
Sparsity and Pruning are Not New Concepts

- Given a signal $y \in \mathbb{R}^n$, a basis $\Phi \in \mathbb{R}^{n \times n}$ and v is the coefficient vector, s.t.,
$$y = \Phi v$$
- Sparsity is defined as k, $\|v\|_0 = k \ll n$.



- Applications:
  - JPEG2000 Compression (2002)
  - Compressed sensing (2007)

Christopoulos, C., et al. (2000).
R. G. Baraniuk (2007).



Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733

4. Compute the saliencies for each parameter: $s_k = h_{kk}u_k^2/2$
5. Sort the parameters by saliency and delete some low-saliency parameters
6. Iterate to step 2

- The idea of pruning neural networks can be traced back to the last century!
- Up to 60% of the parameters can be removed without hurting the MSE.

LeCun, Yann, et al. (1989).

Duke

Obtaining A Sparse Neural Network

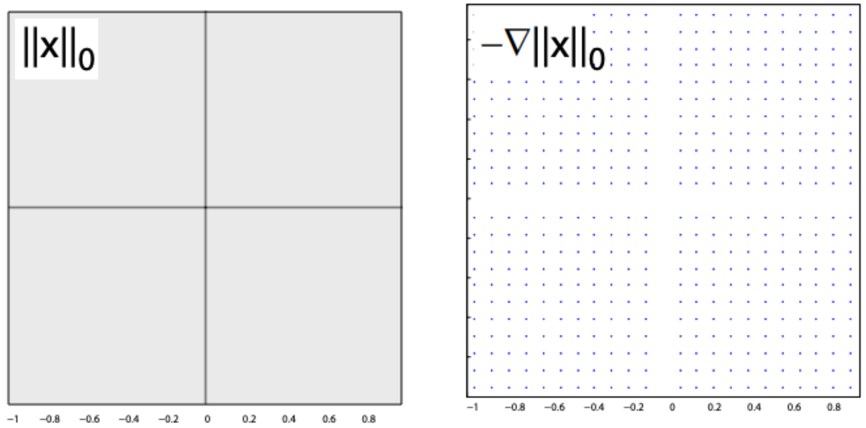
- Objective: Find a set of weights $\hat{\theta} \in \mathbb{R}^N$
  - Can achieve the optimal performance
  - Has a minimal number of nonzero weight elements
- Mathematically, ℓ_0 norm $||\cdot||_0$ is used to represent the number of nonzero elements in a vector or a matrix
- Formal objective:

$$\min_{\theta} ||\theta||_0 \ s.t. \mathcal{L}(\theta) < \epsilon$$

Duke

Dealing with ℓ_0 Minimization

- The ℓ_0 norm is combinatorial, not suitable for gradient-based optimization
  - Not continuous
  - No informative gradients



- Alternative optimization methods
  - Continuous sparsity-inducing regularizer
    - ℓ_1 norm (Lasso), Hoyer, etc.
  - Proximal optimization methods
    - Iterative pruning, ADMM, stochastic approximation, etc.

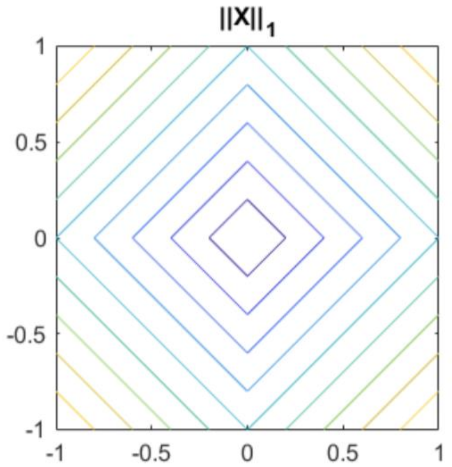
Duke

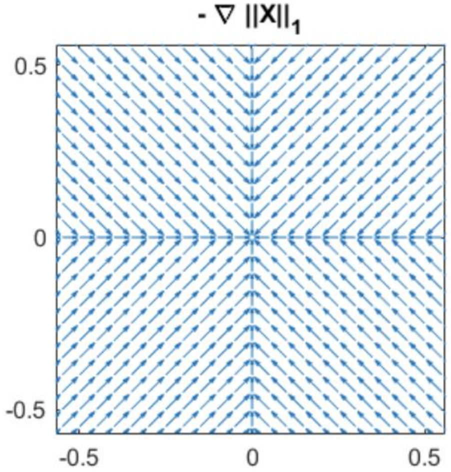
Simple Yet Effective: Lasso

Constant!

$$R(W) = \sum_i |w_i| , \qquad \frac{\partial R(W)}{\partial w_i} = \begin{cases} \boxed{sign(w_i),} & w_i \neq 0 \\ 0, & w_i = 0 \end{cases}$$

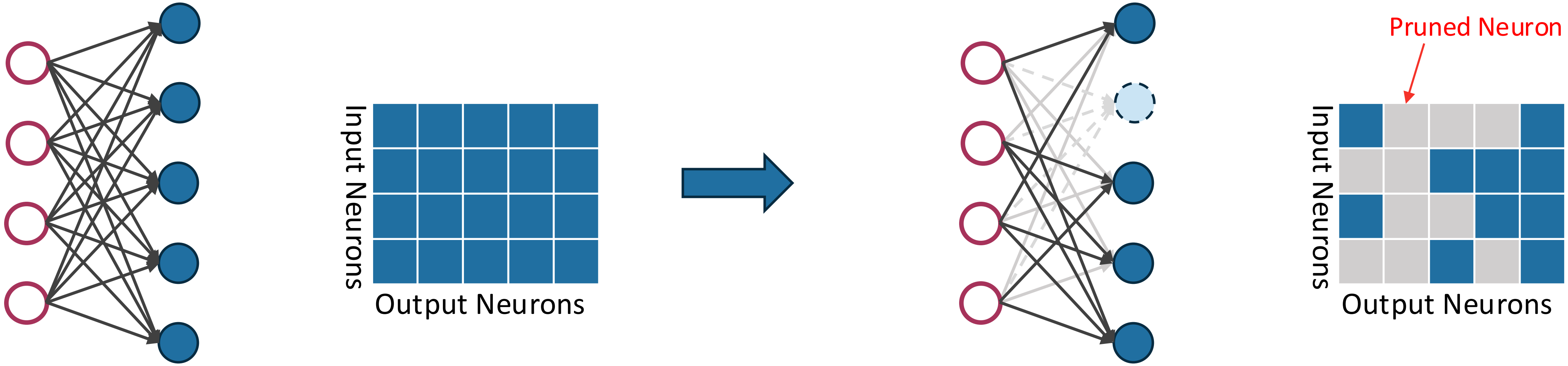
- Always shrink all the nonzero weight elements at a constant speed, until they reach zero
- Least Absolute Shrinkage and Selection Operator





Tibshirani, Robert. "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996): 267-288.

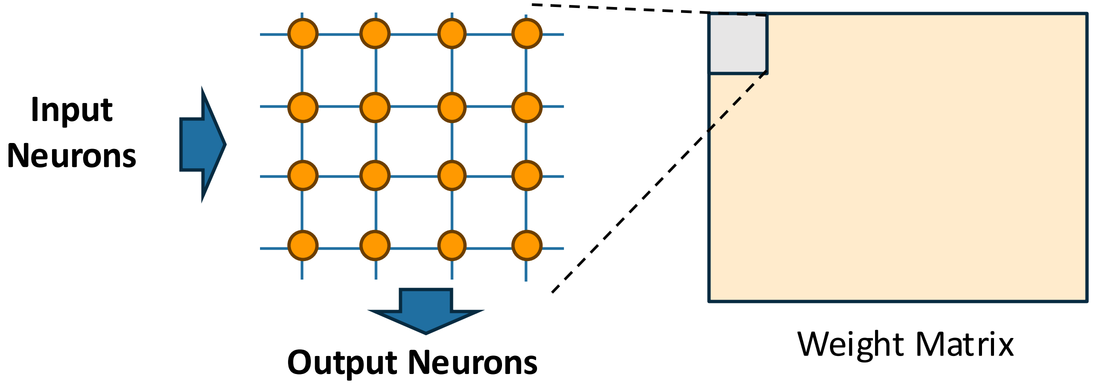
Hardware Representation of Sparsity



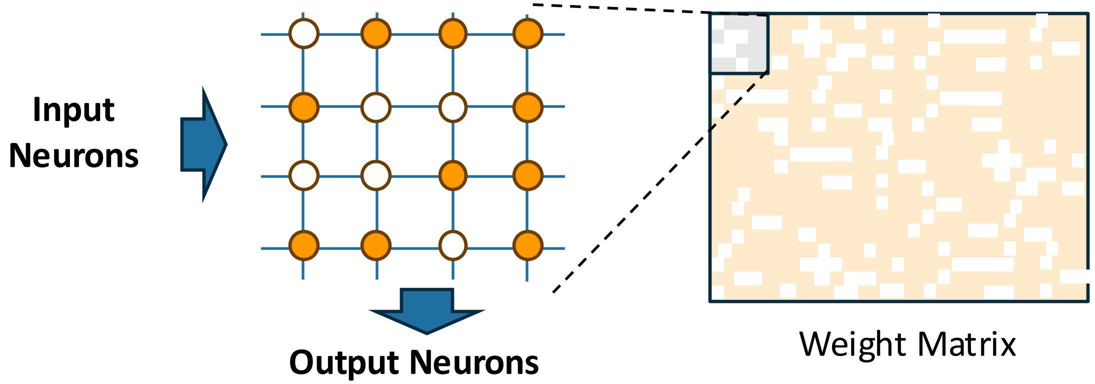
Dense NN | Weight Matrix | Sparse NN | Weight Matrix

Hardware Concerns of Sparsity

- Modern hardware loads and computes the neurons **in parallel**.
- Example: Crossbar-based DNN Accelerator



Dense Neural Network



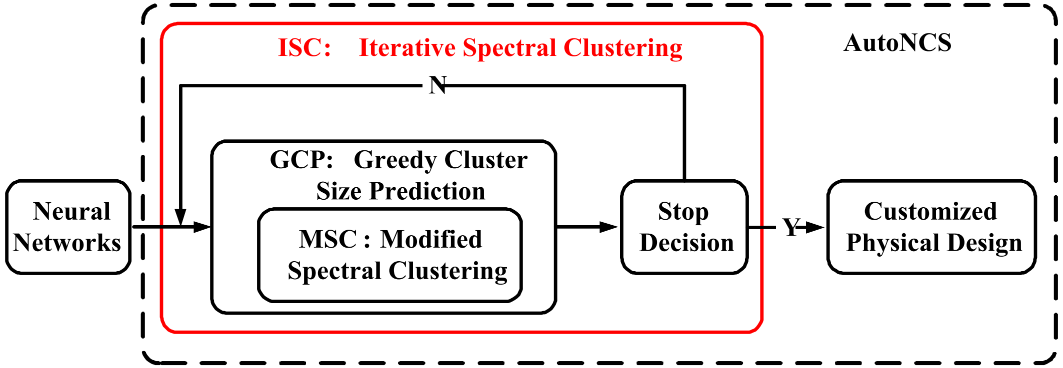
Sparse Neural Network Wastes the Computations

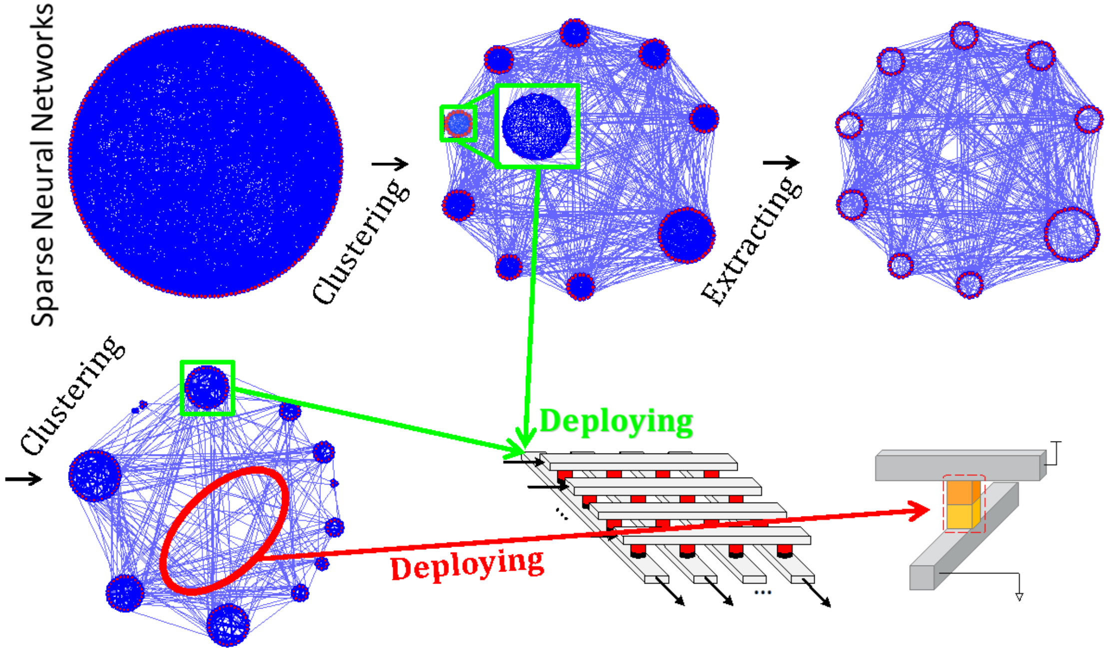
- Solution:
  - Reordering the input/output neurons to cluster dense blocks

Duke

Clustering Sparse Neuron Networks

- **Dilemma of Crossbar-based Implementation**
  - 100% consumption of total available memristors.
  - Maximum allowable size is limited (64x64).
- **Neuron Clustering**

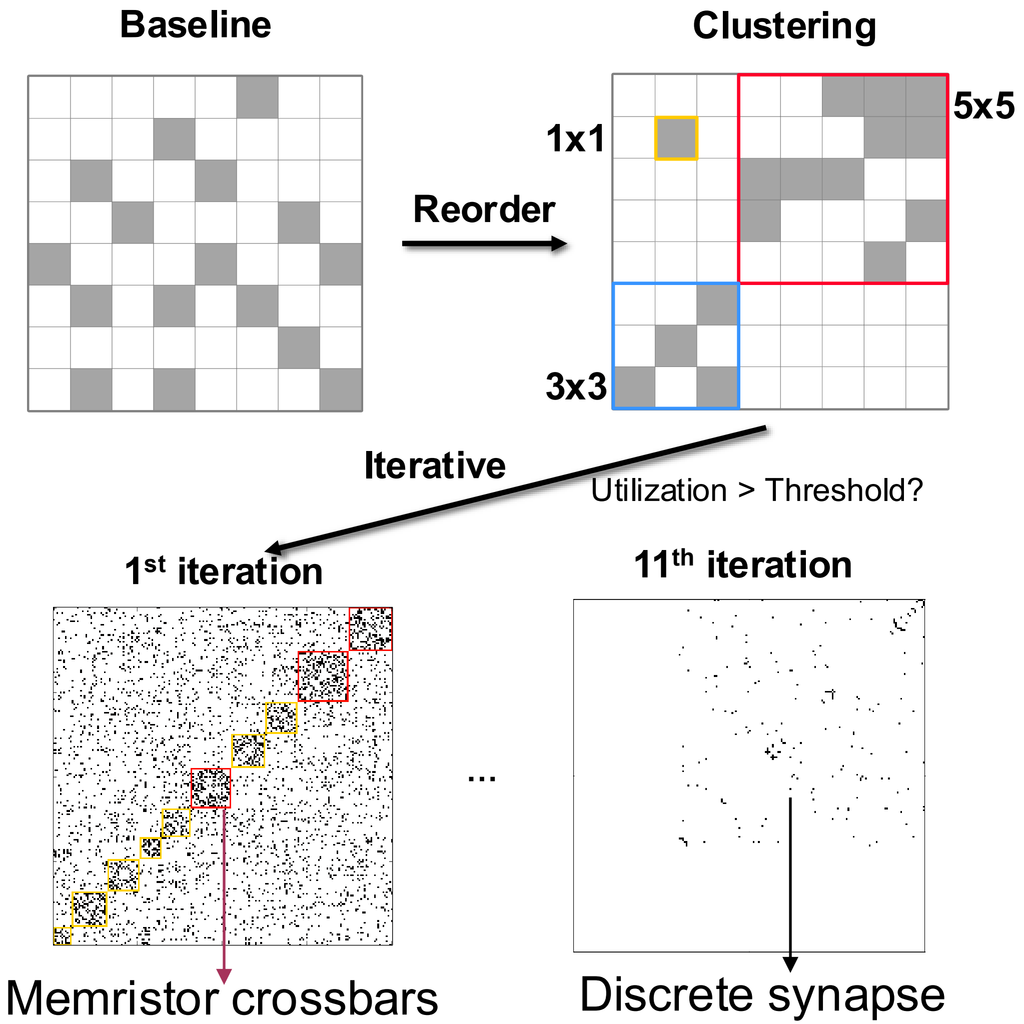


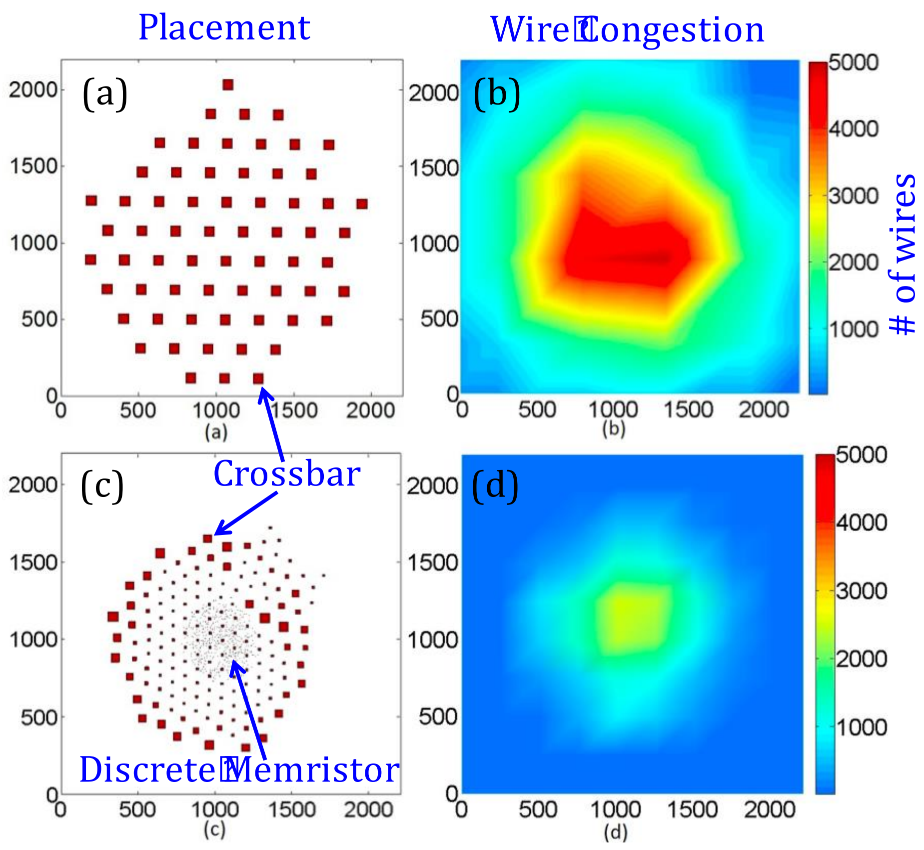


[Best Paper Nomination, DAC15, W. Wen, et al.]

Duke

Clustering Sparse Neuron Networks

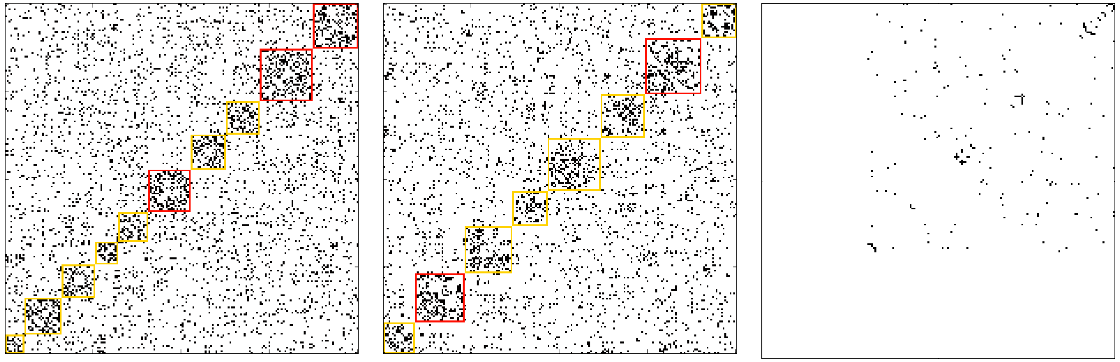




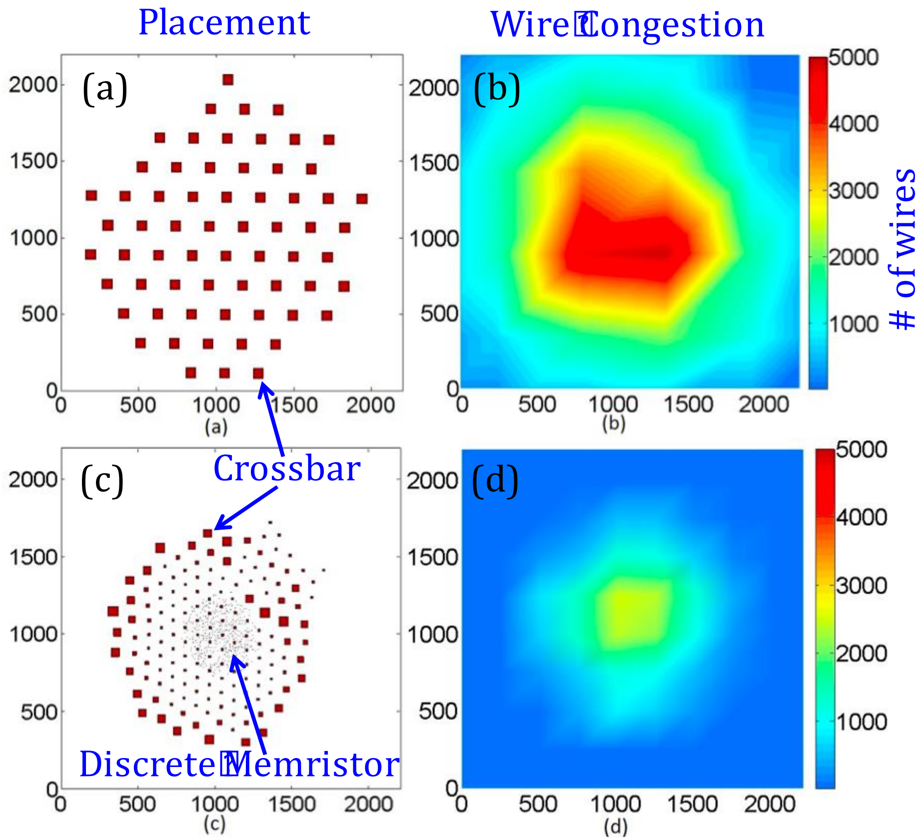
(a)(b): Baseline
(c)(d): AutoNCS

[DAC15, W. Wen, et al.]

Clustering Sparse Neuron Networks



(a) The 1st iteration (b) The 2nd iteration (c) The 11th iteration



Test Bench:

- A Hopfield Network As Associative Memory
- 500 Neurons
- 30 Patterns
- 90%+ Recognition Rate
- 94.39% Sparsity

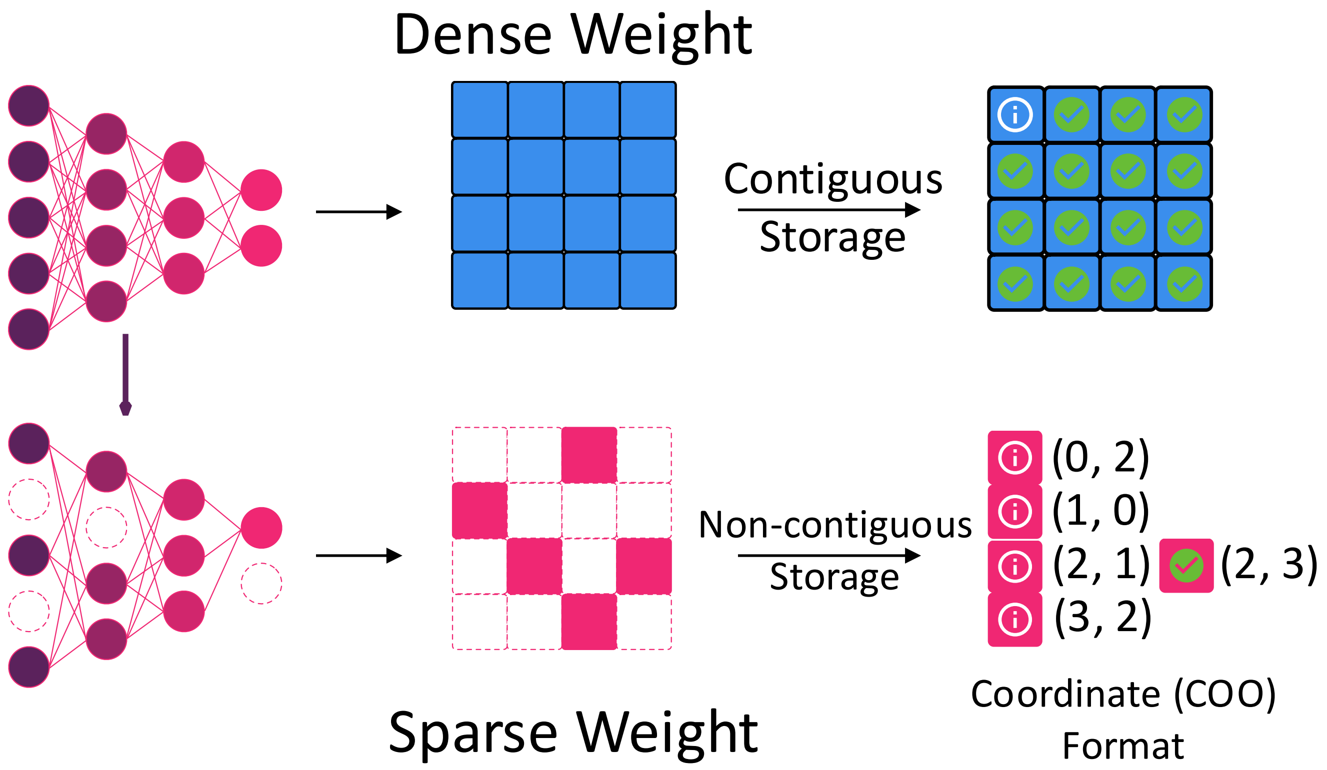
(a)(b): Baseline
(c)(d): AutoNCS

[DAC15, W. Wen, et al.] Duke

Is Clustering Good Enough?

- The clustering process could be time-consuming if the network is large
  - KNN (K-Nearest Neighbors) or similar methods are of $O(n^2)$
- Clustering expects a relatively small block size…
  - Which is not the case for modern GPUs
- The "Clusters" are only **relatively dense**.
- The rest elements are still sparse and irregular.
  - Which encounter dilemma in memory access.

The Sparsity Dilemma: Irregularity in Memory



Dense Memory Access:
Regular Memory Access
1 Cache Miss, 15 Cache Hits

Sparse Memory Access:
Irregular Memory Access
4 Cache Misses, 1 Cache Hit

ⓘ First-time Access, Cache Miss ✅ Follow-up Access, Cache Hit

The Need of Structural Sparsity

- Non-structured sparsity **may not bring much speedup** on traditional platforms like GPUs

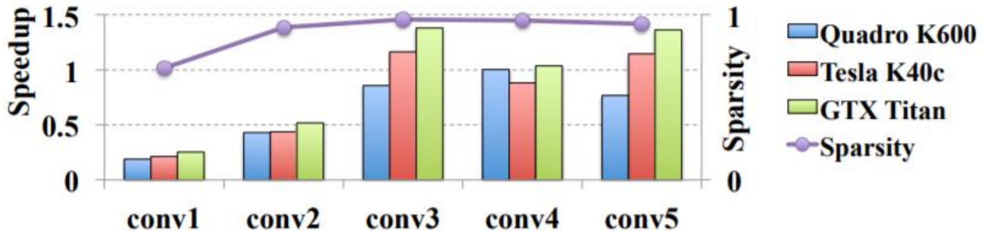
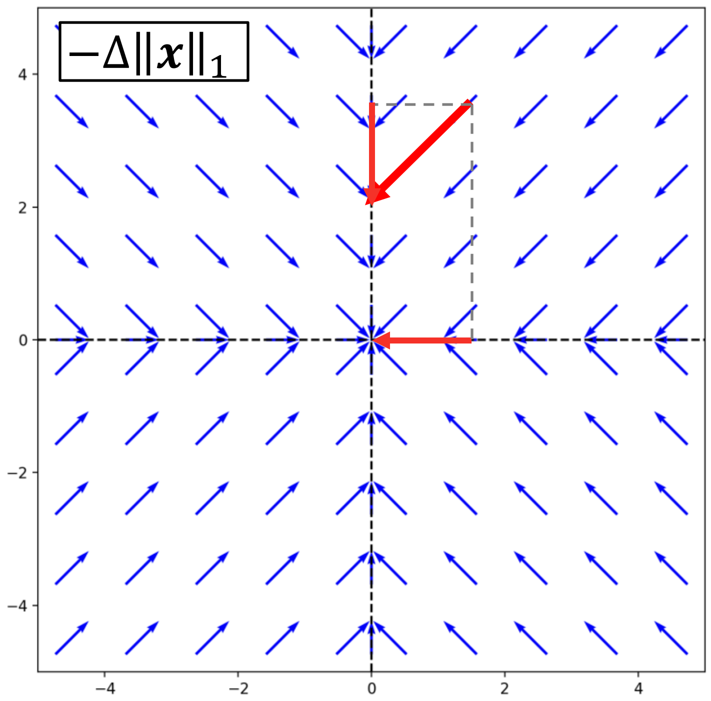


Figure 1: Evaluation speedups of AlexNet on GPU platforms and the sparsity. conv1 refers to convolutional layer 1, and so forth. Baseline is profiled by GEMM of cuBLAS. The sparse matrixes are stored in the format of Compressed Sparse Row (CSR) and accelerated by cuSPARSE.

- Structured sparsity is more hardware-friendly
- Structured sparsity can be achieved by having all the parameters within a structured group become zero or nonzero simultaneously

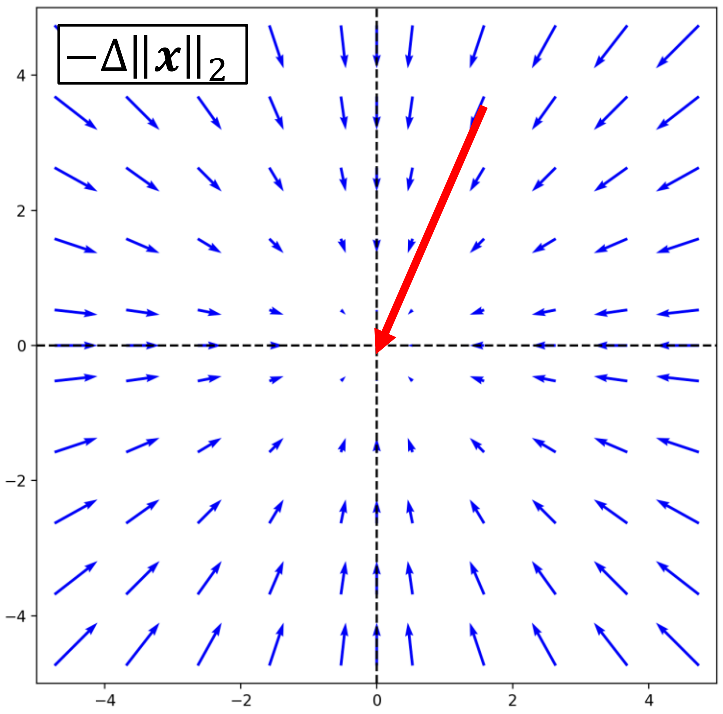
[NeurIPS'16, W. Wen et al.] Duke

Lasso on Structure: Group LASSO



ℓ_1-norm

- Shrink uniformly in each dimension.



ℓ_2-norm

- Shrink toward the origin, where all dimensions are set to 0

ℓ_1 regularizer

$$R(W) = \sum_i |w_i|$$

ℓ_2 regularizer

$$R(W) = \sum_i w_i^2$$

$$R(W) = \sum_i W_i^2, \frac{\partial R(W)}{\partial w_i} = 2W_i$$

Group LASSO: Apply ℓ_1 regularizer to the ℓ_2 norms of the target groups.

$$\sum_{j=1}^{J} \left\|\theta_j\right\|_2$$

Lasso on Structure: Group Lasso

Left: $R(W) = \sqrt{\beta_1^2 + \beta_2^2} + |\beta_3|$

Right: $R(W) = |\beta_1| + |\beta_2| + |\beta_3|$

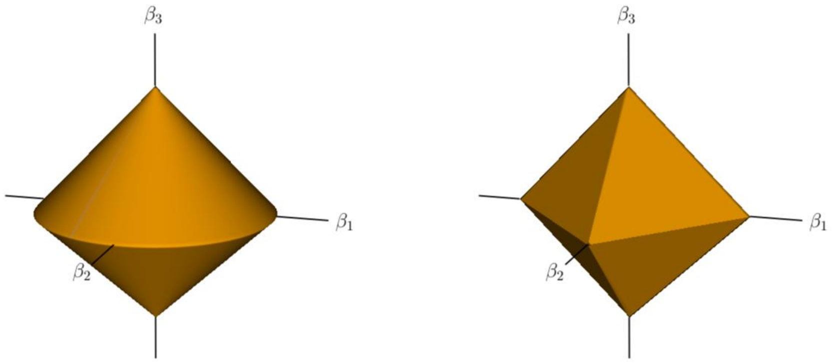


Figure 4.3 *The group lasso ball (left panel) in* $\mathbb{R}^3$*, compared to the* ℓ_1 *ball (right panel). In this case, there are two groups with coefficients* $\theta_1 = (\beta_1, \beta_2) \in \mathbb{R}^2$ *and* $\theta_2 = \beta_3 \in \mathbb{R}^1$.

- The outer Lasso applied on the ℓ_2 norms of each group will encourage some groups' ℓ_2 norms to be 0
- For a ℓ_2 norm to be 0, all elements within the group have to become 0 simultaneously, leading to structured sparsity

Tibshirani, Robert, Martin Wainwright, and Trevor Hastie. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

Structured Sparsity in CNN

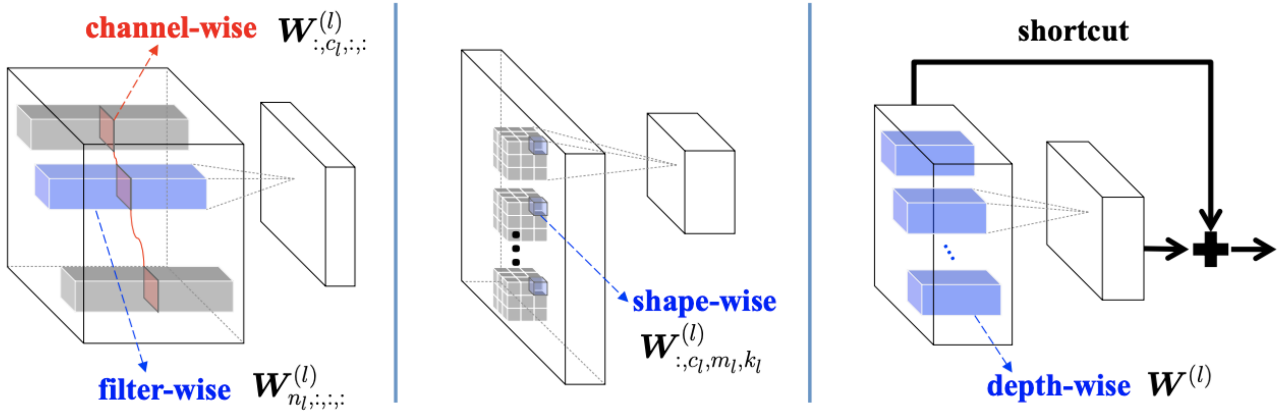
- Removing filters and channels:

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda_n \cdot \sum_{l=1}^{L} \left(\sum_{n_l=1}^{N_l} ||\boldsymbol{W}^{(l)}_{n_l,:,:,:}||_g \right) + \lambda_c \cdot \sum_{l=1}^{L} \left(\sum_{c_l=1}^{C_l} ||\boldsymbol{W}^{(l)}_{:,c_l,:,:}||_g \right)$$

- Modifying filter shape:

$$E(\boldsymbol{W}) = E_D(\boldsymbol{W}) + \lambda_s \cdot \sum_{l=1}^{L} \left(\sum_{c_l=1}^{C_l} \sum_{m_l=1}^{M_l} \sum_{k_l=1}^{K_l} ||\boldsymbol{W}^{(l)}_{:,c_l,m_l,k_l}||_g \right)$$

- Shortcut is added to enable whole layer removal



[NeurIPS'16, W. Wen et al.]

Experiment Results with Group Lasso

Table 4: Sparsity and speedup of *AlexNet* on ILSVRC 2012

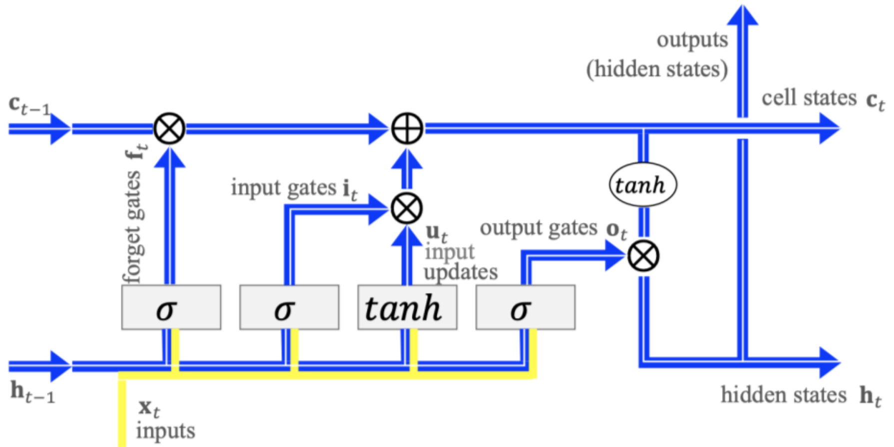
#	Method	Top1 err.	Statistics	conv1	conv2	conv3	conv4	conv5
1	ℓ_1	44.67%	sparsity	67.6%	92.4%	97.2%	96.6%	94.3%
			CPU ×	0.80	2.91	4.84	3.83	2.76
			GPU ×	0.25	0.52	1.38	1.04	1.36
2	SSL	44.66%	column sparsity	0.0%	63.2%	76.9%	84.7%	80.7%
			row sparsity	9.4%	12.9%	40.6%	46.9%	0.0%
			CPU ×	1.05	3.37	6.27	9.73	4.93
			GPU ×	1.00	2.37	4.94	4.03	3.05
3	pruning [7]	42.80%	sparsity	16.0%	62.0%	65.0%	63.0%	63.0%
4	ℓ_1	42.51%	sparsity	14.7%	76.2%	85.3%	81.5%	76.3%
			CPU ×	0.34	0.99	1.30	1.10	0.93
			GPU ×	0.08	0.17	0.42	0.30	0.32
5	SSL	42.53%	column sparsity	0.00%	20.9%	39.7%	39.7%	24.6%
			CPU ×	1.00	1.27	1.64	1.68	1.32
			GPU ×	1.00	1.25	1.63	1.72	1.36

- Less overall sparsity, but higher speedup

[NeurIPS’16, W. Wen et al.]

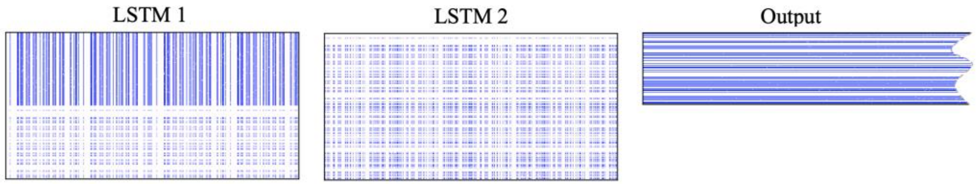
Learning Structural Sparsity in LSTMs

- Learn Intrinsic structural sparsity in LSTMs



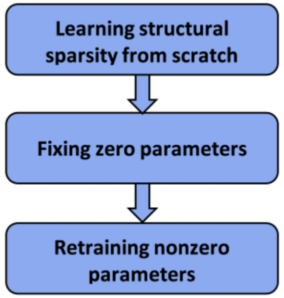
Method	Dropout keep ratio	Perplexity (validate, test)	ISS # in (1st , 2nd) LSTM	Weight #	Total time*	Speedup	Mult-add reduction†
baseline	0.35	(82.57, 78.57)	(1500, 1500)	66.0M	157.0ms	1.00×	1.00×
ISS	0.60	(82.59, 78.65) (80.24, 76.03)	(373, 315) (381, 535)	21.8M 25.2M	14.82ms 22.11ms	10.59× 7.10×	7.48× 5.01×
direct design	0.55	(90.31, 85.66)	(373, 315)	21.8M	14.82ms	10.59×	7.48×

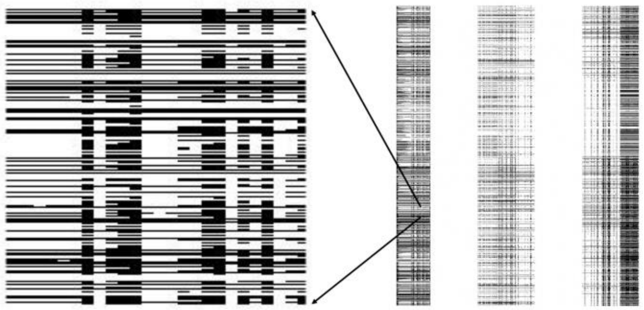
* Measured with 10 batch size and 30 unrolled steps.
† The reduction of multiplication-add operations in matrix multiplication. Defined as (original Mult-add)/(left Mult-add)



[ICLR'18, W. Wen et al.]

- Customized structural sparsity on Gaussian Neural Accelerator (GNA)





Method	λ	Sparsity in LSTM			Sparsity	WER	
group-8		1	2	3	mean	develop	test
baseline	0	0	0	0	0	11.5%	11.4%
ESS	0.15	39.1%	59.6%	36.0%	45.8%	12.0%	11.8%
ESS	0.35	62.2%	82.5%	68.6%	72.5%	12.6%	12.6%
ESS	0.65	74.9%	88.5%	71.7%	78.9%	13.3%	13.3%

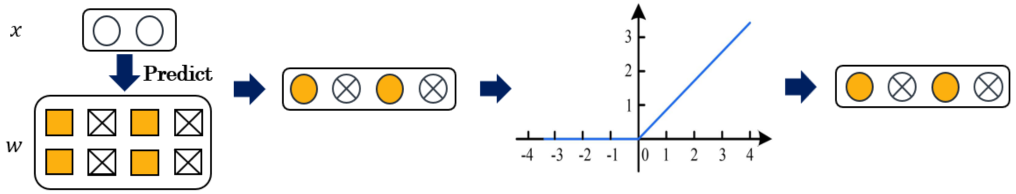
Method	λ	Sparsity in LSTM			Sparsity	WER	
group-16		1	2	3	mean	develop	test
baseline	0	0	0	0	0	11.5%	11.4%
ESS	0.15	36.6%	56.7%	37.2%	44.6%	11.8%	11.7%
ESS	0.35	55.8%	77.0%	63.8%	67.0%	12.5%	12.4%
ESS	0.65	67.9%	84.9%	70.4%	75.4%	13.1%	13.1%

[ICASSP'19, J. Zhang et al.]

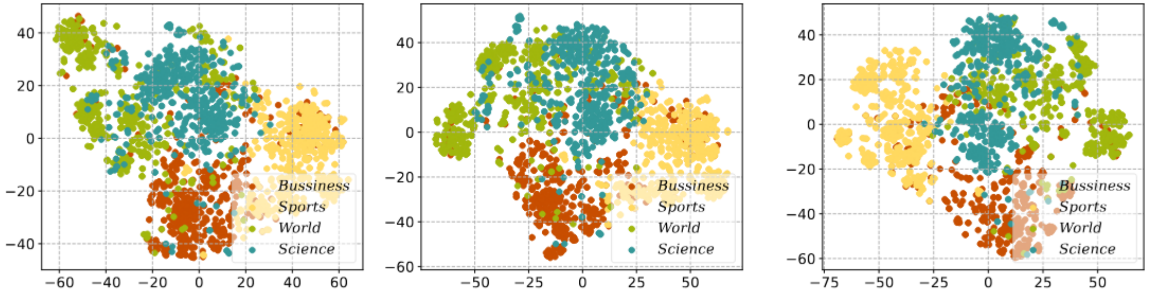
Duke

Activation Sparsity in Foundation Models

- Neurons in LLM are highly sparsely activated.
- Predicting the activation of neurons in advance can save the calculation of non-activated neurons, thereby reducing 70% calculation during inference.



- Activation sparsity is strongly correlated with input semantics in terms of both similarity and stability.

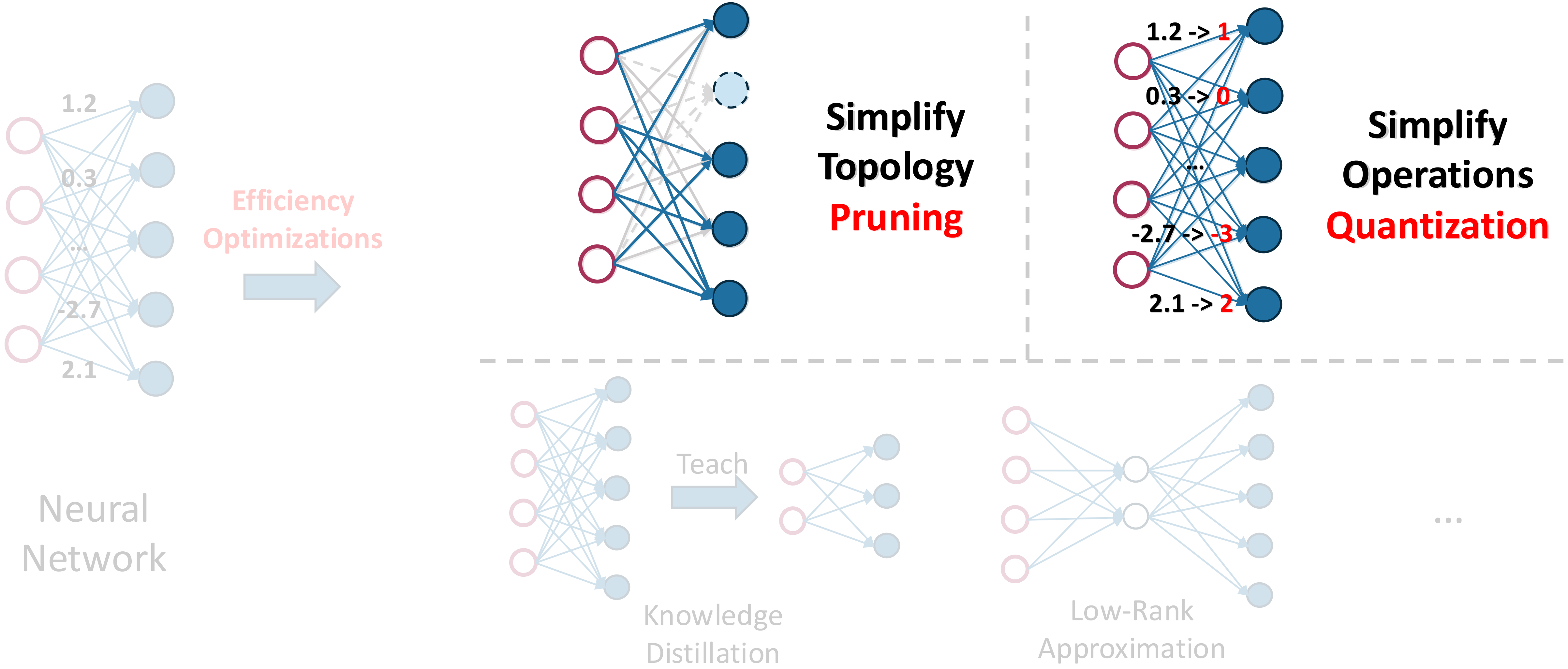


(a) 5-th Layer (b) 15-th Layer (c) 25-th Layer

[arXiv24, Q. Wang et al.]

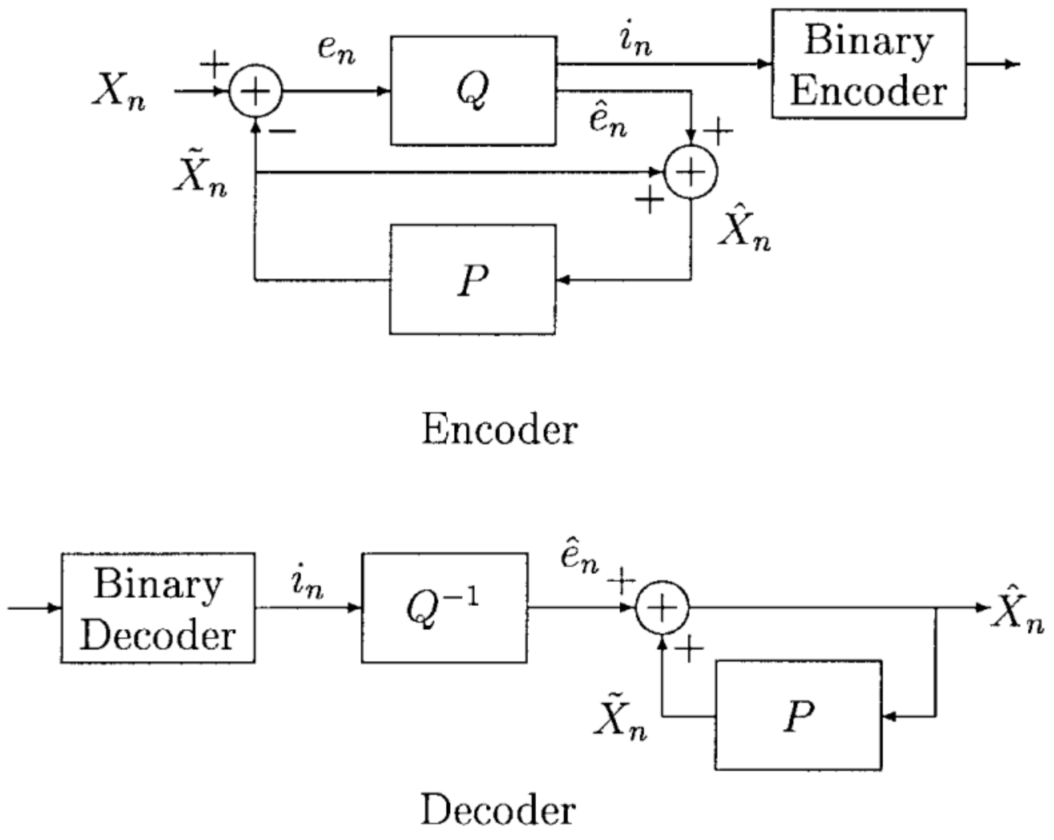
- We propose a framework that activates neurons by input semantic predictions.
- Achieve 10x speedup by leveraging input semantics to predict sparsity patterns.

Towards Efficient AI…



Quantization

- The history of quantization theory and practice dates back to 1948.
- Quantization is a basic technique in digital signal processing (DSP).
- Converting continuous levels into discrete ones (analog-to-digital conversion).



R. M. Gray and D. L. Neuhoff, "Quantization," in *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325-2383, Oct. 1998, doi: 10.1109/18.720541.

Reducing Computational Complexity

Multiplier and adder circuits with different precisions.

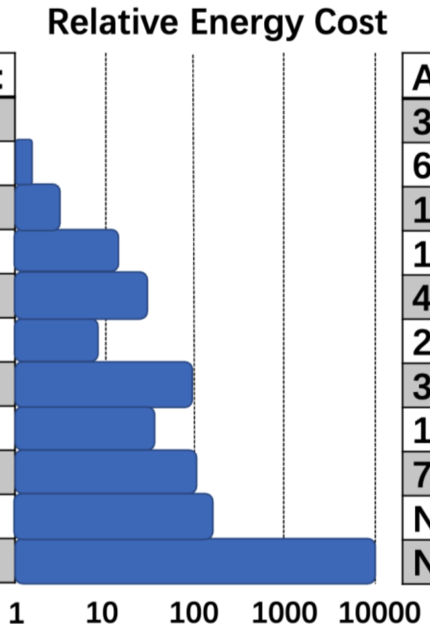
Int4: Area/Power: 1x

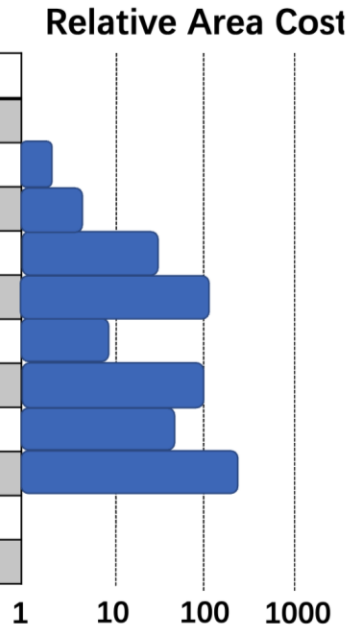
Int8: Area: 3.5×, Power: 3.9×

Int32: Area: 43×, Power: 60×

FP32: Area: 96×, Power: 72×

Operation:	Energy(pJ):	Area(μm^2):
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8kb)	5.0	N/A
32b DRAM Read	640	N/A

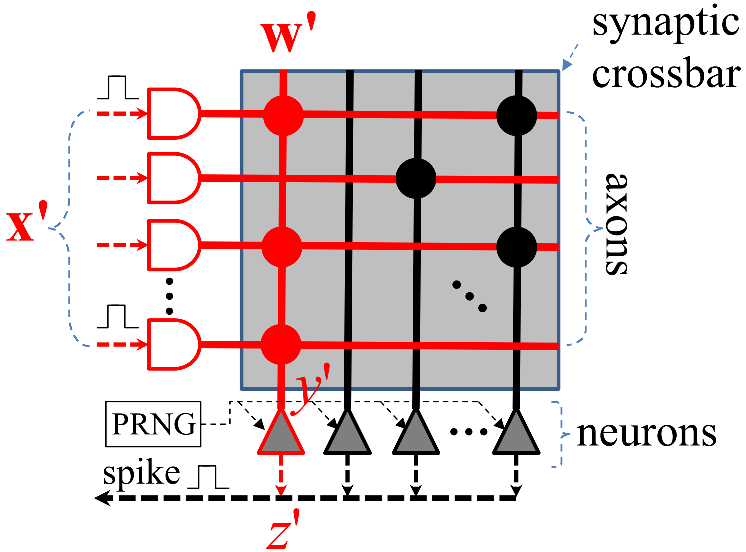




arXiv’23, Sehoon Kim et al.

Duke

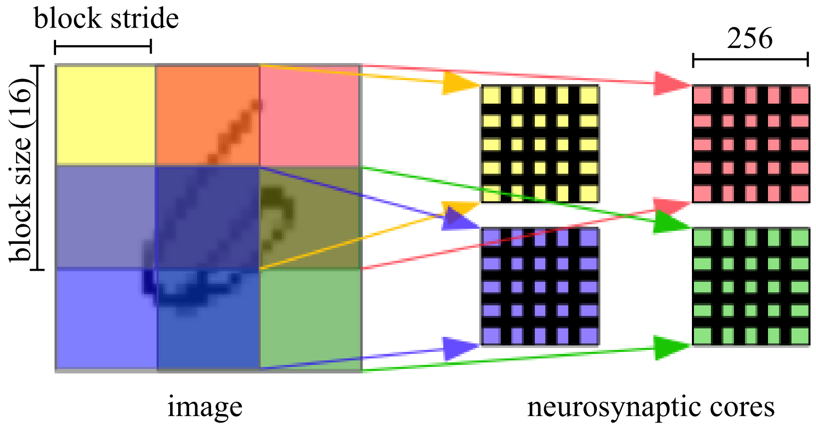
IBM TrueNorth – Binary Neural Networks



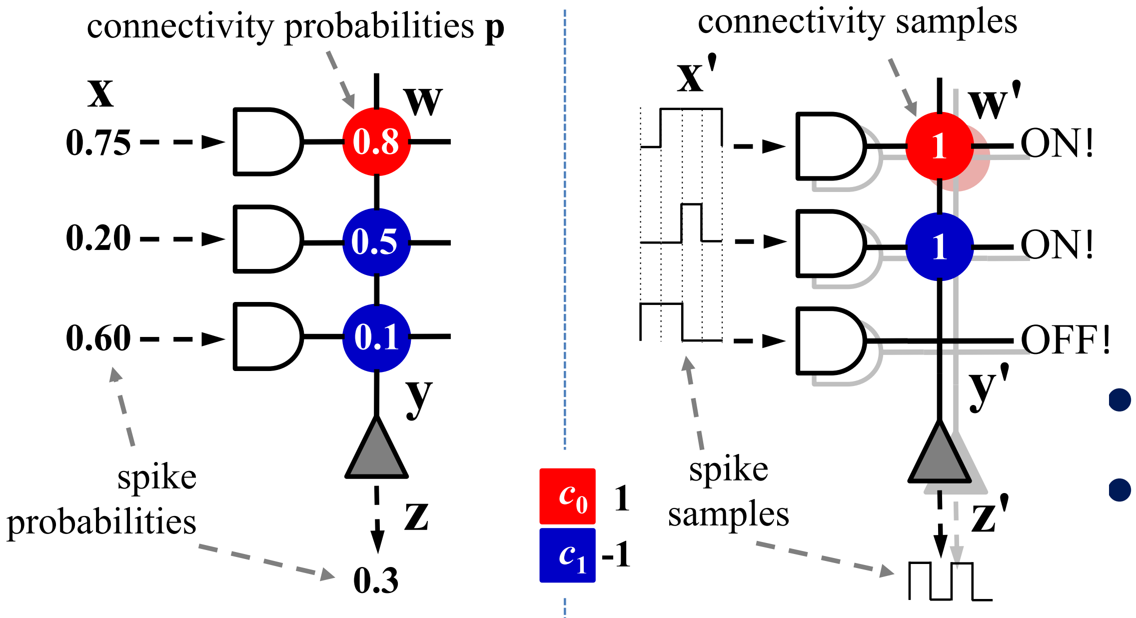
$$y' = \Sigma_i x_i' w_i'$$

(2014)

- **4,096 neurosynaptic cores**
- **1 million neurons**
- **256 million synapses**
- **A 65mW real-time neurosynaptic processor**



Minimizing The Deployment Variance



(a) Tea learning

(b) Tea deploying

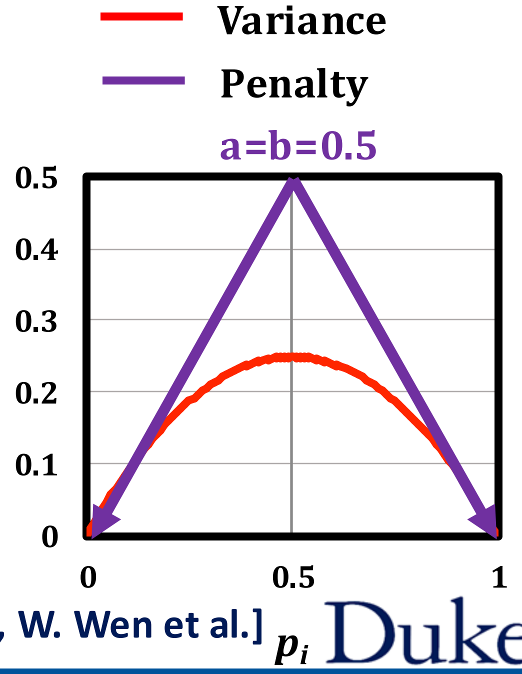
$$\Delta y = y' - y$$

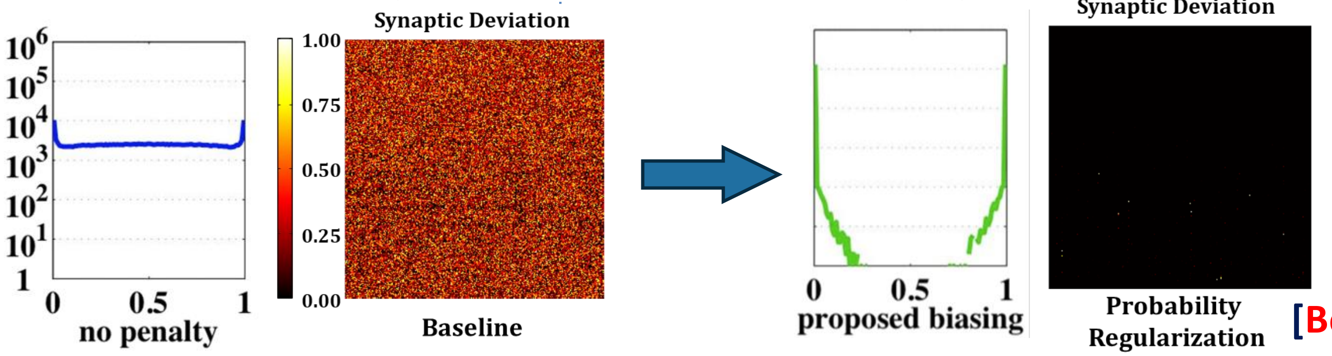
$$\boxed{var\{\Delta y\} = \sum_{i=0}^{n-1} var\{w'_i x'_i\}}$$

$$var\{w'_i\} = E\{(w'_i)^2\} - E\{w'_i\}^2 = p_i(1-p_i)$$

- Target: Minimize the variance.
- Solution: Adding a regularizer.

$$E_W(\boldsymbol{w}) = \||\boldsymbol{w} - a| - b\|$$

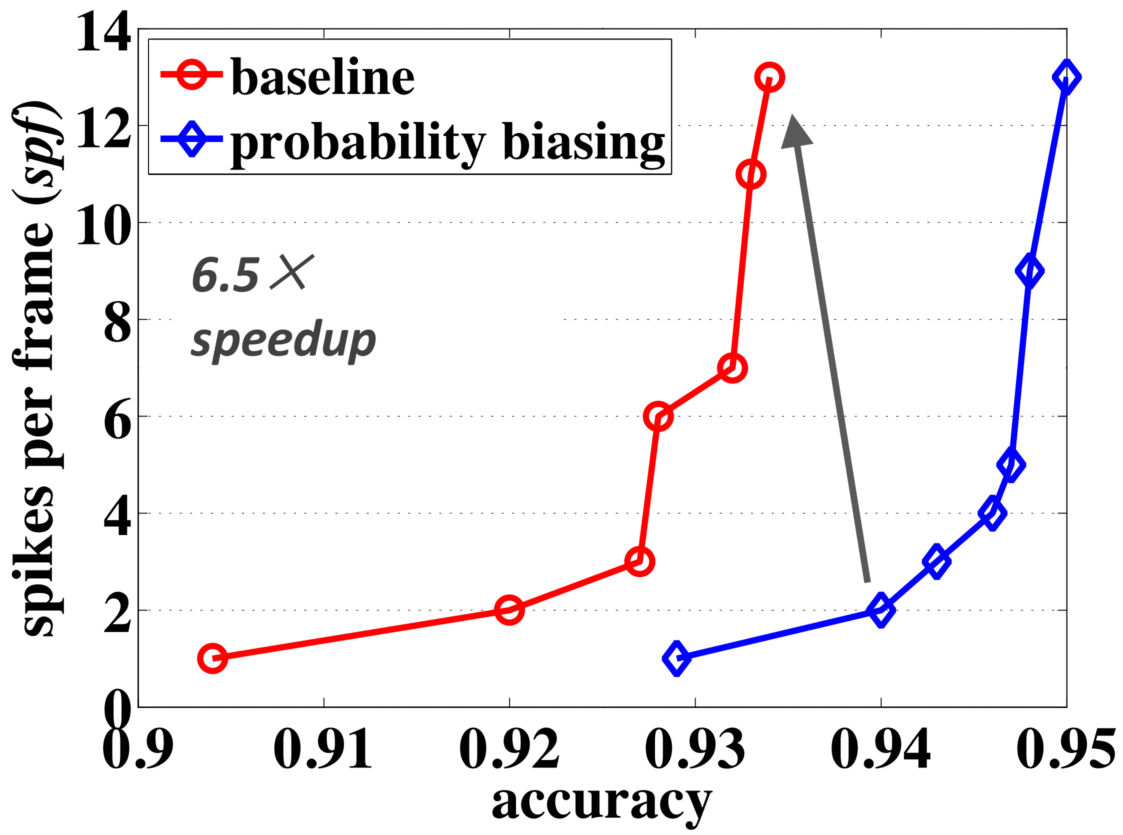


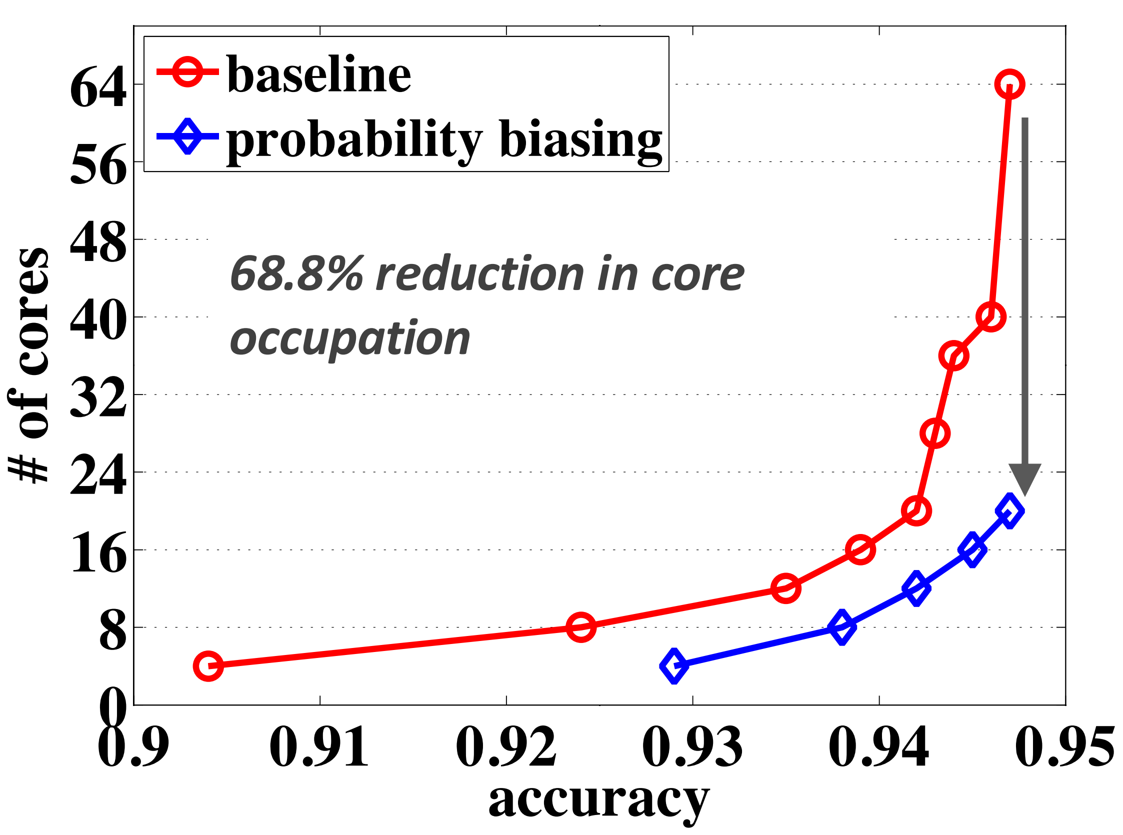


[Best Paper Nomination, DAC'16, W. Wen et al.]

Duke

Experimental Results





[Best Paper Nomination, DAC'16, W. Wen et al.]

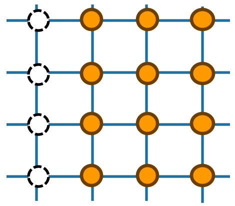
Duke

A Unified View of Pruning and Quantization

- Selecting the optimal precision for each layer introduces a large and discrete design space.
- For a fixed-point quantized matrix, when can its precision be reduced?
  - MSB=0 for all elements: precision can reduce directly

$$\begin{bmatrix}6\\3\end{bmatrix} \equiv \begin{bmatrix}0&1&1&0\\0&0&1&1\end{bmatrix}_2 \equiv \begin{bmatrix}1&1&0\\0&1&1\end{bmatrix}_2$$

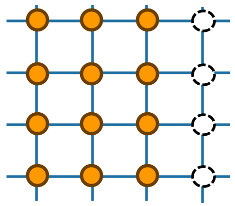
Remove the MSB Column



  - LSB=0 for all elements: precision can be reduced with scaling factor 2

$$\begin{bmatrix}10\\4\end{bmatrix} \equiv \begin{bmatrix}1&0&1&0\\0&1&0&0\end{bmatrix}_2 \equiv 2 \times \begin{bmatrix}1&0&1\\0&1&0\end{bmatrix}_2$$

Remove the LSB Column



- MP quantization scheme can be explored by inducing **structural bit-level sparsity**

[ICLR'21, H. Yang et al.]

Mix-Precision with Bit-Level Sparsity

- We first perform 8-bit quantization.
- The quantized model is converted into a **bit-level representation**
  - Each bit-value is a trainable, floating-point value.
- We apply bit-level group LASSO to the columns.

Structured pruning of the bit representation

↕

Quantization

$$\underset{W}{\begin{bmatrix} 2.0 \\ 1.2 \\ 0.2 \end{bmatrix}} \xrightarrow{\textbf{Quantize}} \underset{W_s}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}$$

Recover weight values from bit representation.

STE forward

$$w_q = \frac{1}{7} R(0.2 \times 4 + 1.3 \times 2 + 0.8 \times 1) = \frac{4}{7}$$

DNN FP

$$\mathcal{L} = \mathcal{L}(sW_q)$$

DNN BP

$$\frac{\partial \mathcal{L}}{\partial w_q} = 0.7$$

STE backward

$$\frac{\partial \mathcal{L}}{\partial w_s^{(2:0)}} = [0.4 \quad 0.2 \quad 0.1]$$

Get gradient for each bit.

[ICLR'21, H. Yang et al.]

Duke

Mix-Precision with Bit-Level Sparsity

Table 2: Quantization results of ResNet-20 models on the CIFAR-10 dataset. BSQ is compared with DoReFa-Net (Zhou et al., 2016), PACT (Choi et al., 2018), LQ-Net (Zhang et al., 2018), DNAS (Wu et al., 2019) and HAWQ (Dong et al., 2019). "MP" denotes mixed-precision quantization.

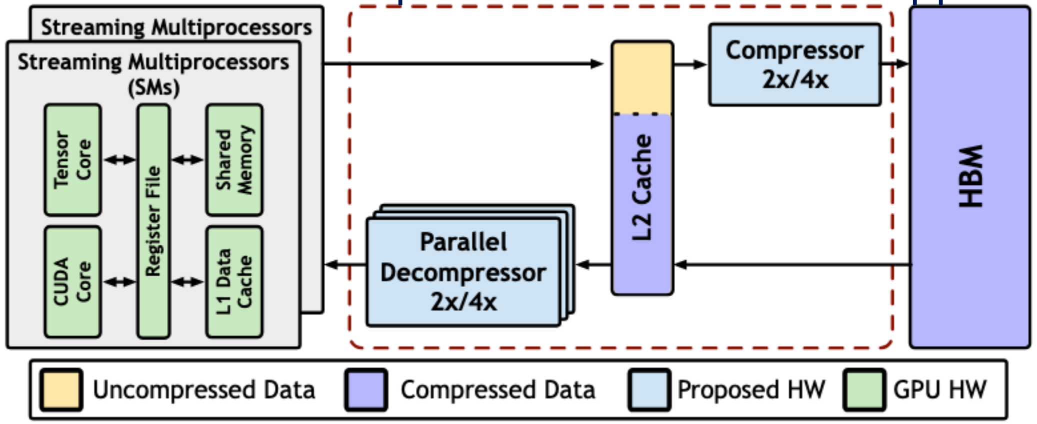
Benchmarks					BSQ		
Act. Prec.	Method	Weight Prec.	Comp ($\times$)	Acc (%)	α	Comp ($\times$)	Acc (%)
32-bit	Baseline	32	1.00	92.62			
	LQ-Nets	3	10.67	92.00	5e-3	14.24	92.77
	DNAS	MP	11.60	92.72	7e-3	19.24	91.87
	LQ-Nets	2	16.00	91.80			
4-bit	HAWQ	MP	13.11	92.22	5e-3	14.24	92.32
3-bit	LQ-Nets	3	10.67	91.60	2e-3	11.04	92.16
	PACT	3	10.67	91.10	5e-3	16.37	91.72
	DoReFa	3	10.67	89.90			
2-bit	LQ-Nets	2	16.00	90.20			
	PACT	2	16.00	89.70	5e-3	18.85	90.19
	DoReFa	2	16.00	88.20			

[ICLR'21, H. Yang et al.]

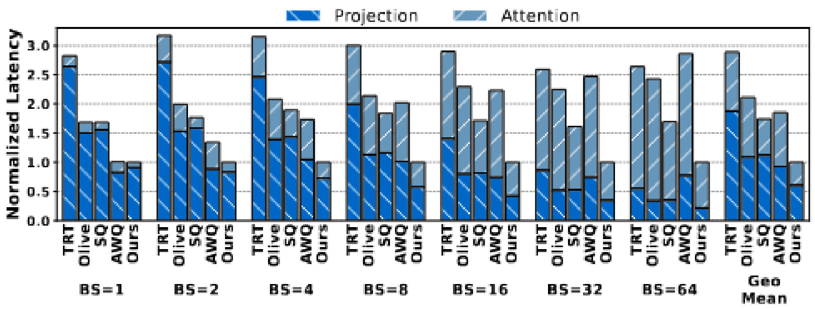
Combining Quantization and Cache Compression for Memory-bound LLMs

- LLMs are memory-intensive due to Keys and Values storage
- Previous quantization methods suffer from high compression overhead

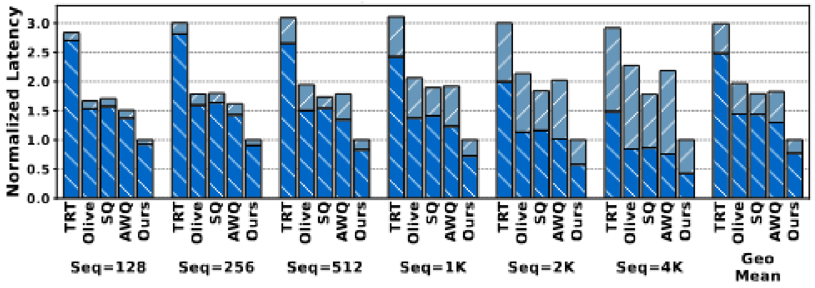
- Integrate compression and decompression in cache-level:
  - Low overhead
  - Effective compression with hardware support



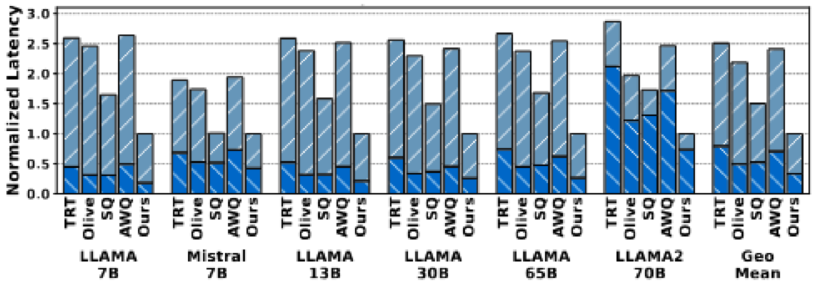
[ISCA'25, C. Feng et al.]



(a) Normalized latency vs. batch sizes on LLaMA-13B.



(b) Normalized latency vs. sequence lengths on LLaMA-13B.



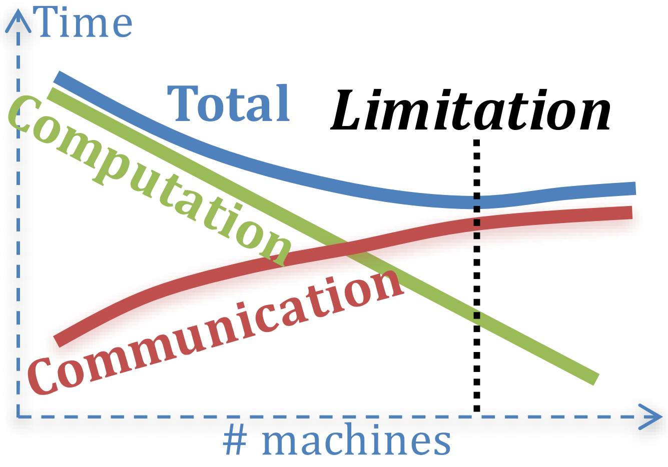
(c) Normalized latency vs. various models.

Duke

Towards Efficient AI…

- One Device -> Computational Complexity for Inference & Training
  - By reducing the complexity of **the topology**
  - By reducing the complexity of **the operation**
- Many devices -> Scale up Training Efficiency with Parallelism

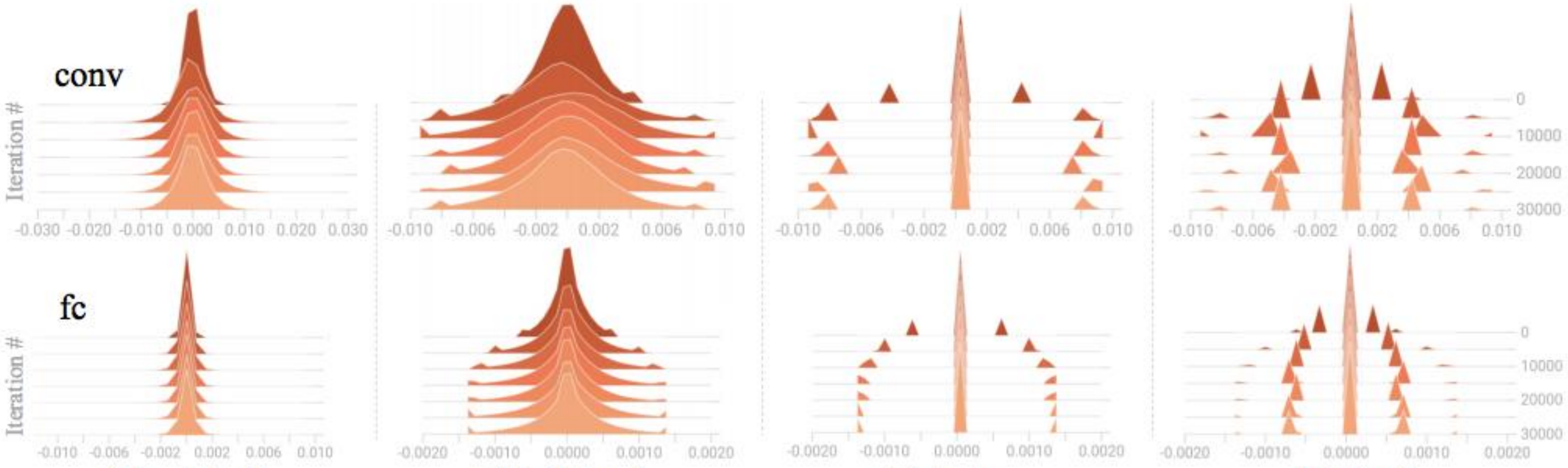
Communication bottleneck



Communication Complexity

Duke

TernGrad – Gradients Histograms



(a) original (b) clipped (c) ternary (d) final

- Transfer the distribution instead of the raw values.
- Use 0, +1, -1 to represent the direction of the gradients.
- Convert 32-bit floating point into 3 levels.

[Oral, NeurIPS'17, W. Wen et al.]

TernGrad – Speedup

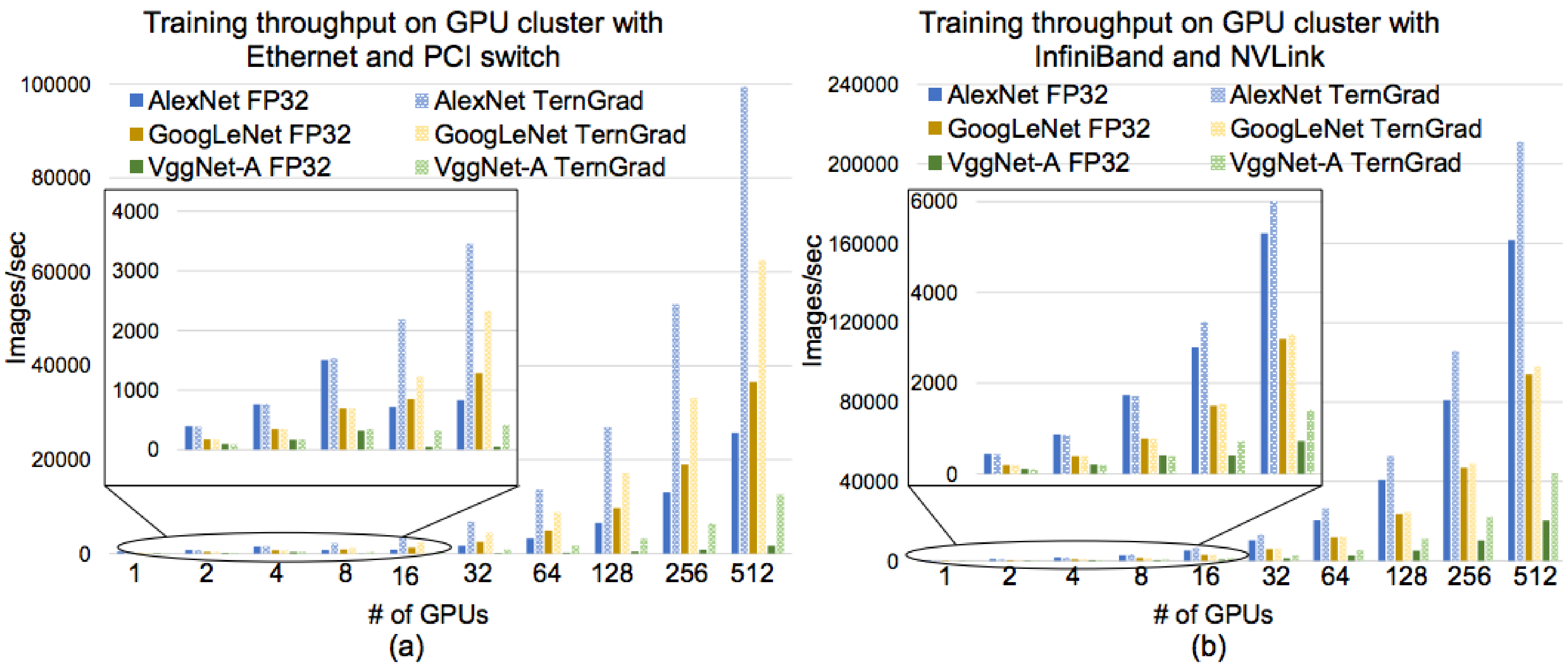
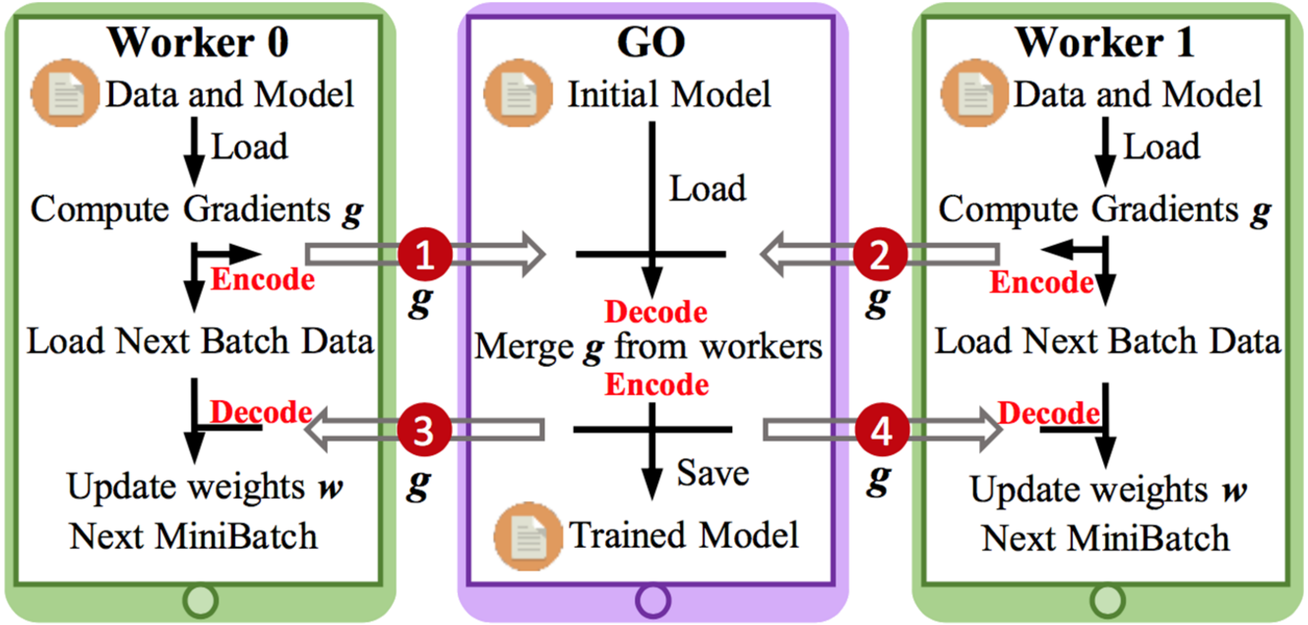


Figure 5: Training throughput on two different GPUs clusters: (a) 128-node GPU cluster with 1Gbps Ethernet, each node has 4 NVIDIA GTX 1080 GPUs and one PCI switch; (b) 128-node GPU cluster with 100 Gbps InfiniBand network connections, each node has 4 NVIDIA Tesla P100 GPUs connected via NVLink. Mini-batch size per GPU of *AlexNet*, *GoogLeNet* and *VggNet-A* is 128, 64 and 32, respectively

[Oral, NeurIPS'17, W. Wen et al.]

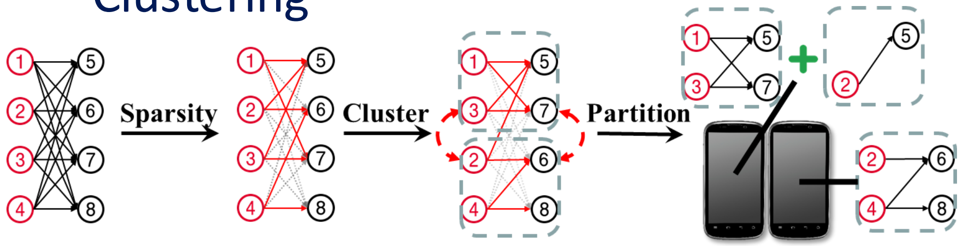
Clustering for Distributed Mobile Training and Testing

Distributed Mobile Training Architecture

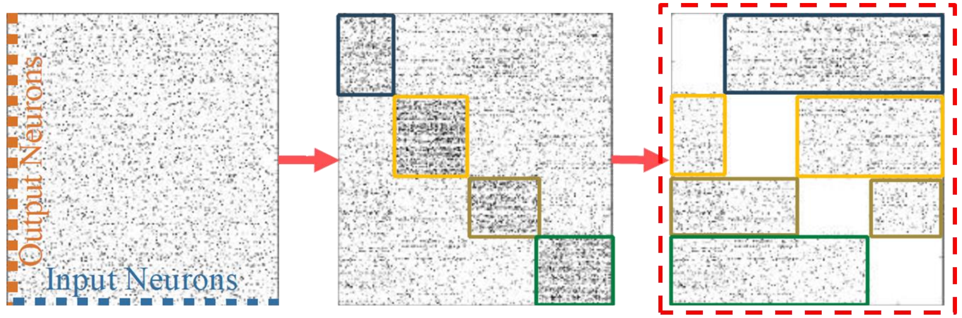


- **Transmission Reduction**

Clustering



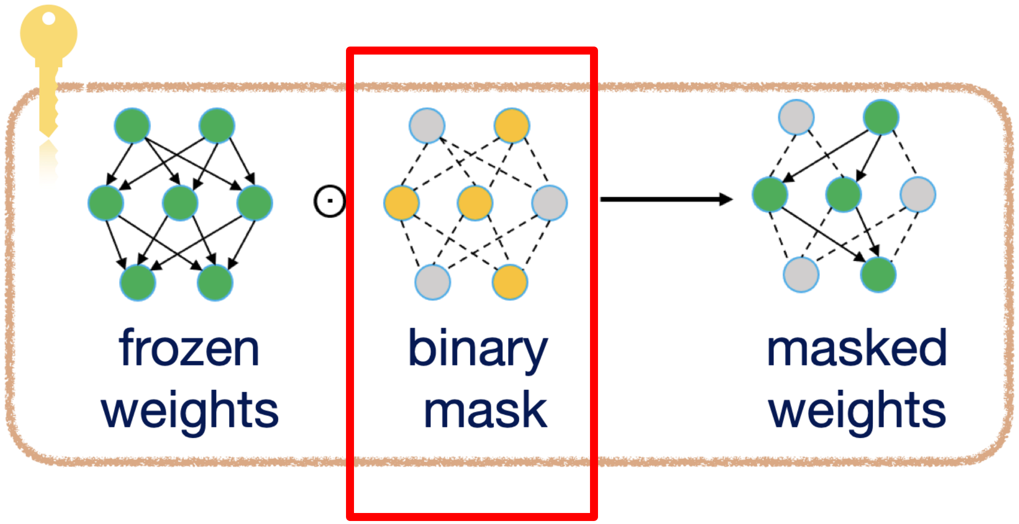
- **Task Mapping**

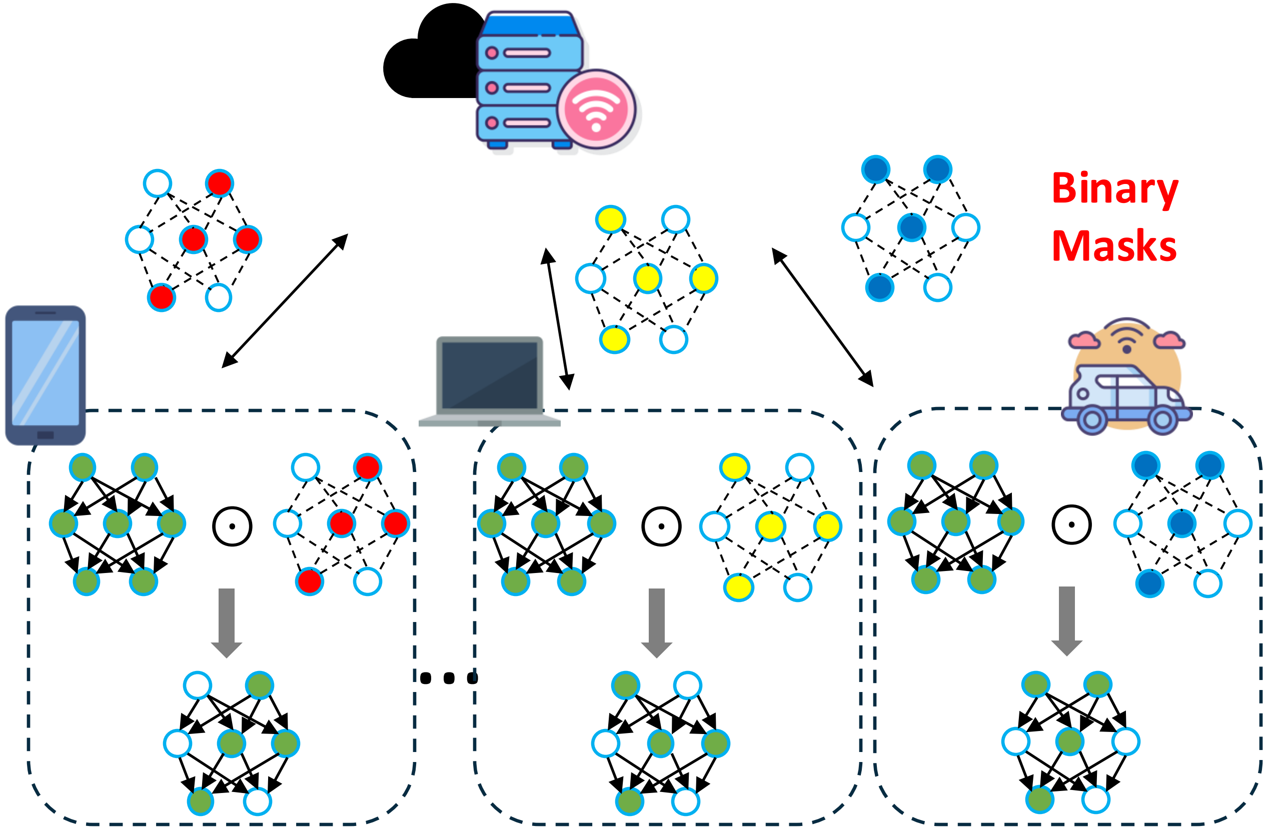


(a) Original (b) Clustered (MSCC) (c) Outliers (FGCP)

[**Best Paper Award**, DATE 2017, ICCAD 2017, J. Mao, et al.]

TernGrad to Binary Mask: FedMask



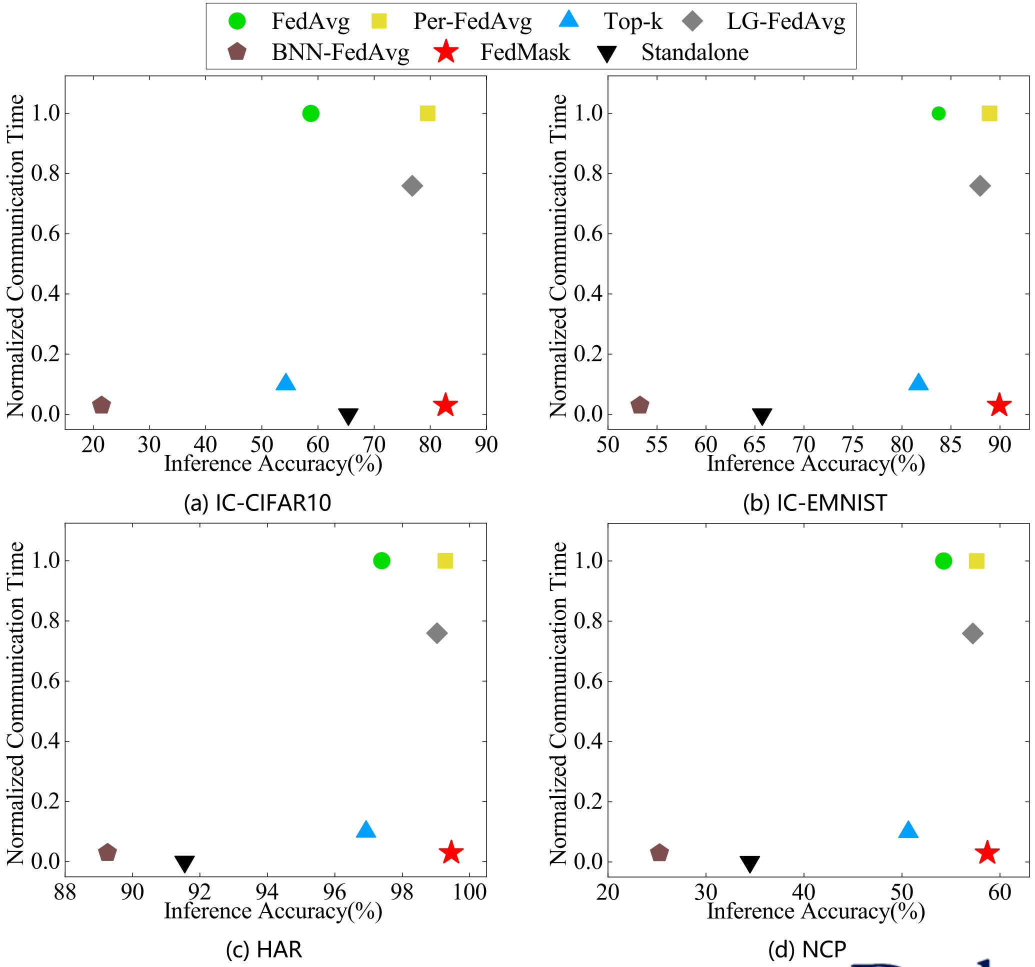


The devices train and transmit 1-bit binary masks rather than the 32-bit weights!

[SenSys'21, A. Li et al.]

Evaluations

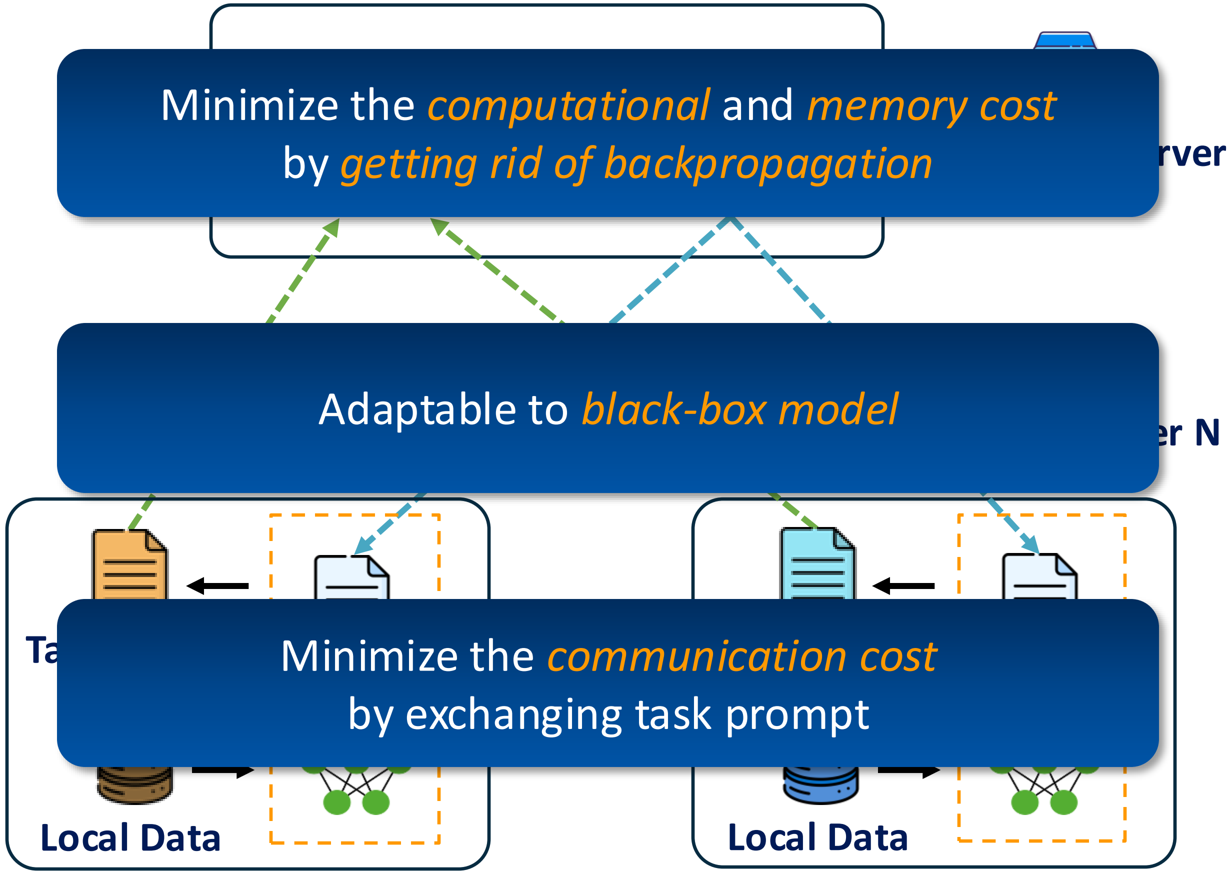
- Dataset
  - EMNIST, CIFAR10, HAR, Shakespeare
- Baselines
  - Standalone
  - FedAvg
  - Top-k (communication efficient)
  - BNN-FedAvg (binary neural network+FedAvg)
  - Per-FedAvg (FedAvg+MAML)
  - LG-FedAvg (personalization+communciation)



[SenSys'21, A. Li et al.]

Duke

FedBPT: Federated Black-box Prompt Tuning

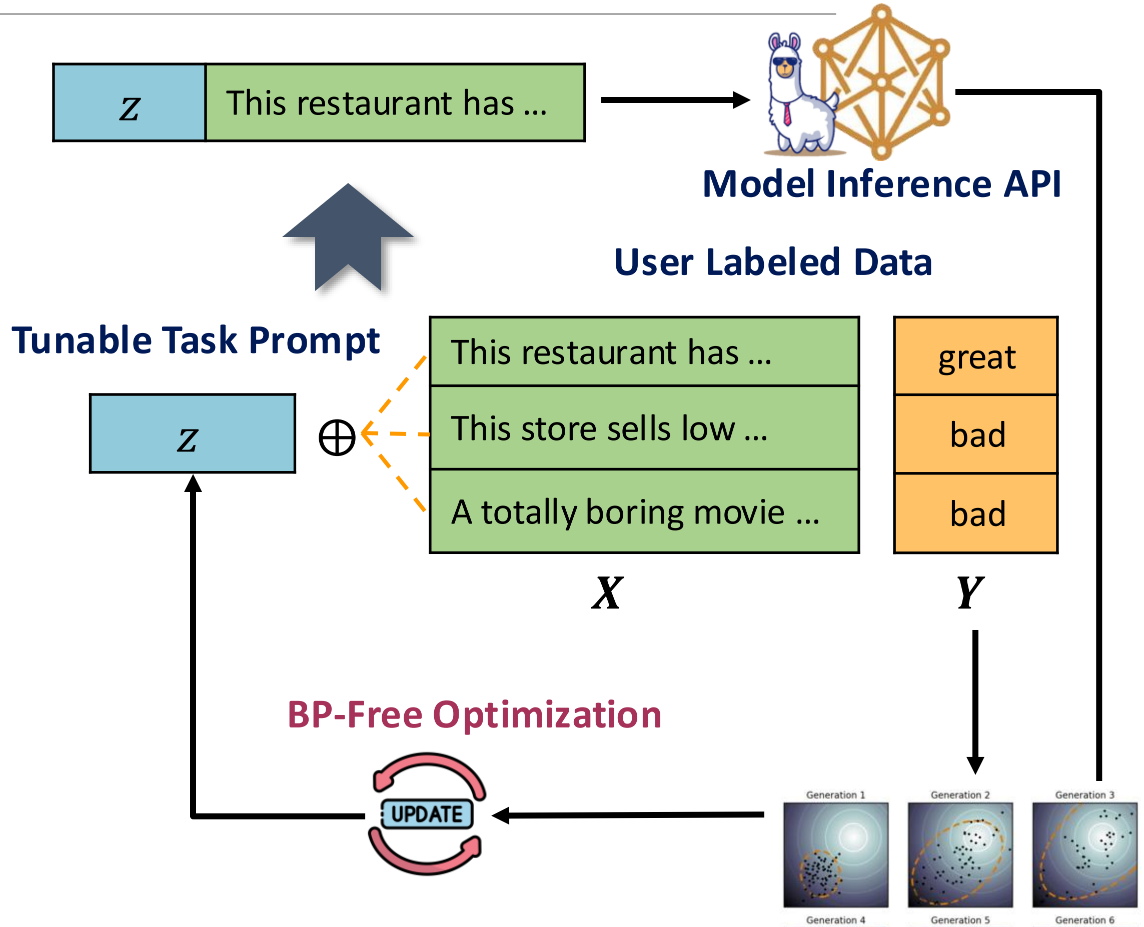


Minimize the *computational* and *memory cost* by *getting rid of backpropagation*

Adaptable to *black-box model*

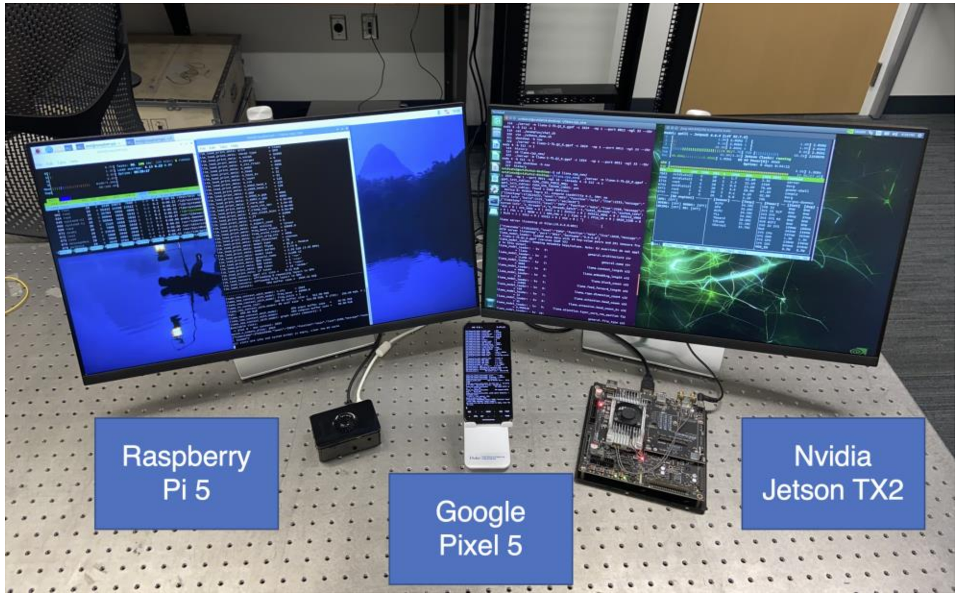
Minimize the *communication cost* by exchanging task prompt

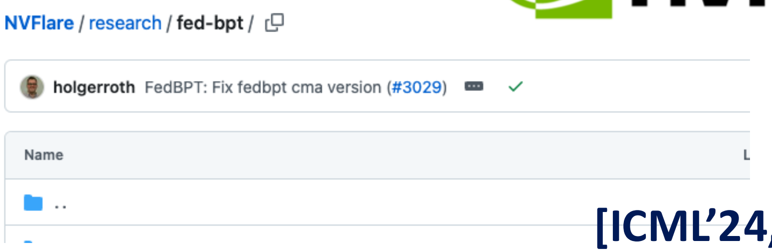
[ICML'24, Jingwei Sun, et al.]

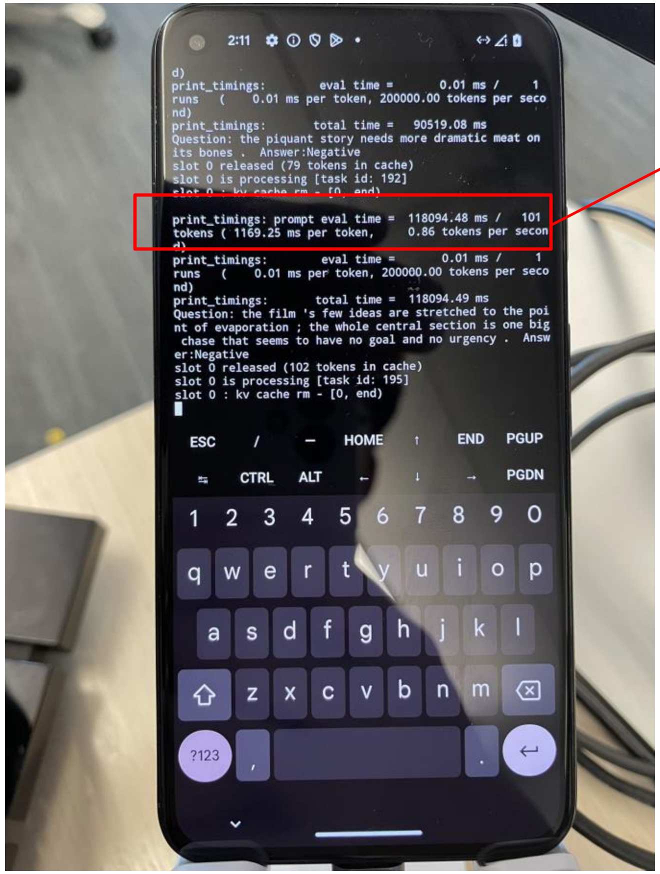


CMA-ES (Covariance Matrix Adaptation Evolutionary Search) algorithm

Demo with Llama2-7B





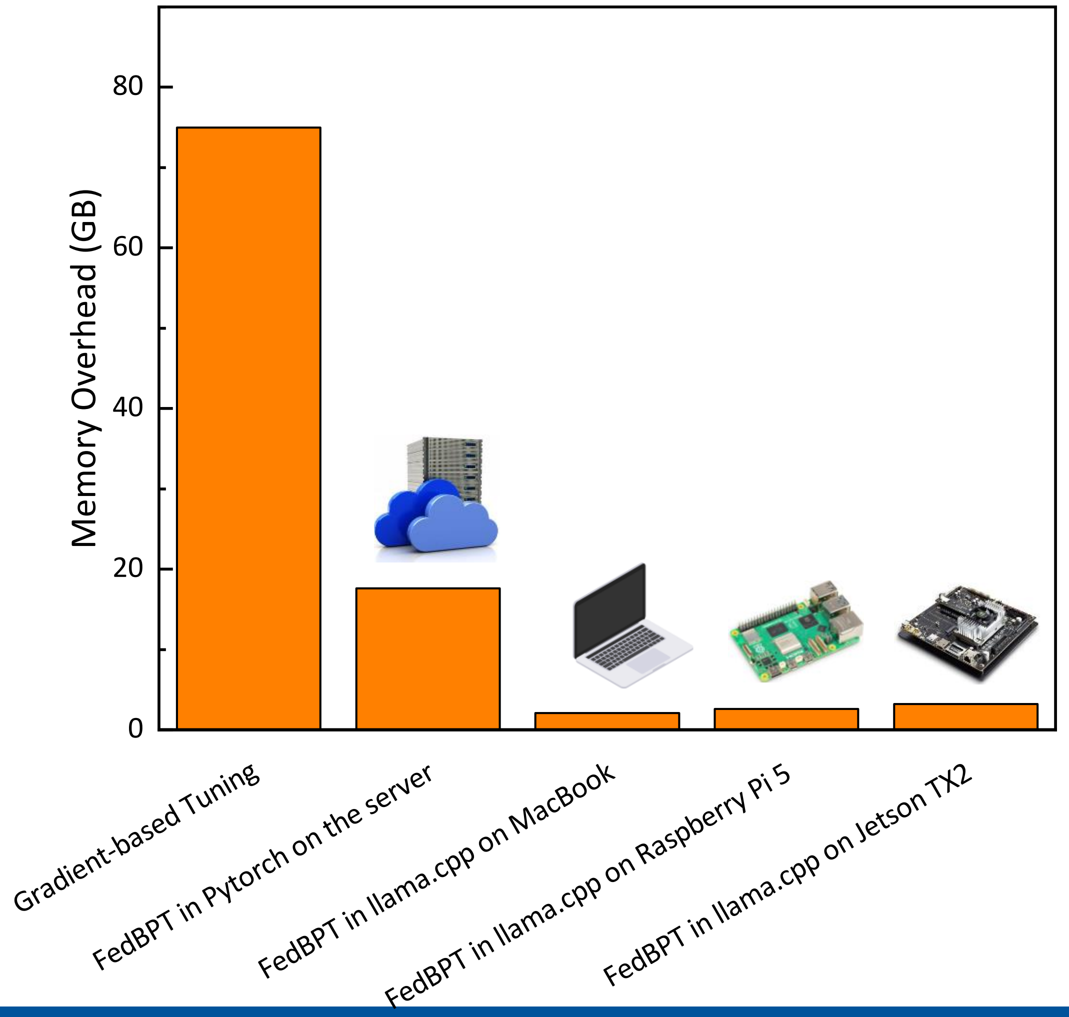


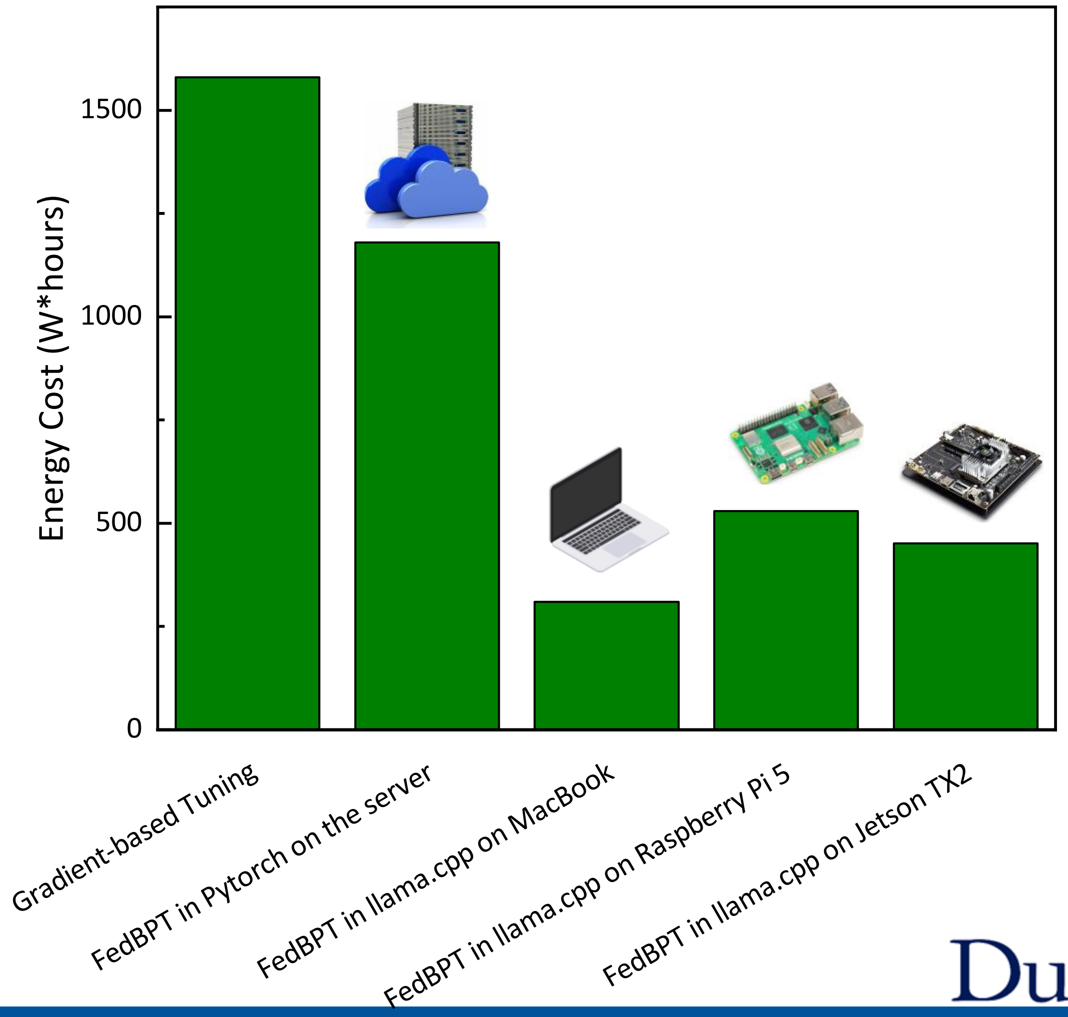
User data

[ICML’24, , Best Paper Award in AAAI Spring Syp. Jingwei Sun, et al.]

Duke

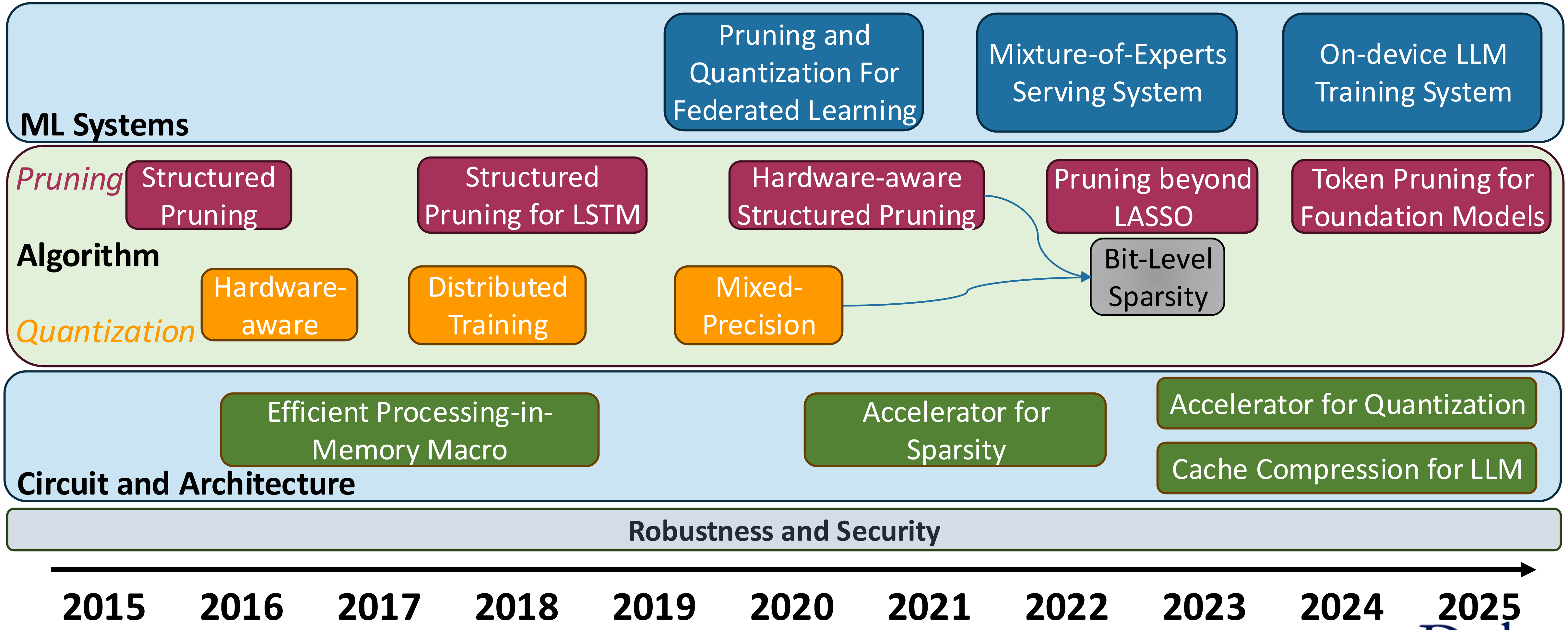
On-device Experimental Results of Llama2-7B





Duke

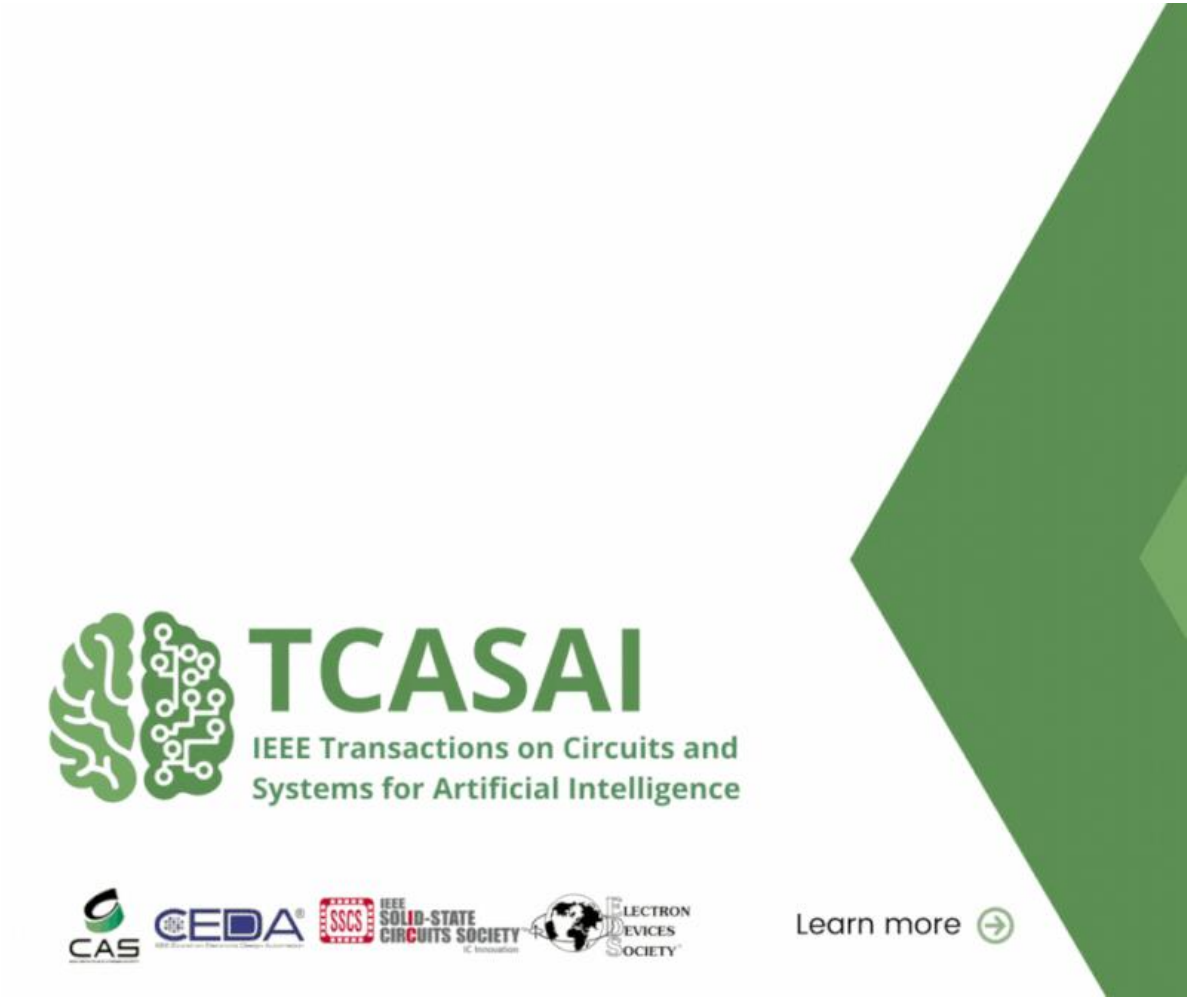
Our Journey Towards Efficient AI



What We Learned from This Journey

- Our goal is to address the cost of **storage**, **computation**, and **communication** when deploying AI models through **software and hardware co-design**.
  - With a unified optimization framework.
- The optimization should consider both software flexibility and hardware constraints.
  - ◦The design could be multi-objective.

TCASAI Call-for-Paper!



The IEEE Transactions on Circuits and Systems for Artificial Intelligence (TCASAI) is financially sponsored by the IEEE CASS, SSCS, and CEDA, and technically sponsored by the IEEE EDS and NANO

Scope

The IEEE Transactions on Circuits and Systems for Artificial Intelligence (TCASAI) publishes contributions related to circuits and systems for artificial intelligence, including circuit and electronic system design, implementation, and demonstration

Submission is now open!

References

- **[DAC 2015, Best Paper Nomination]** Wen, Wei, et al., An EDA Framework for Large Scale Hybrid Neuromorphic Computing Systems.
- **[NeurIPS 2016]** Wen, Wei, et al., Learning Structured Sparsity in Deep Neural Networks.
- **[ICLR 2020]** Yang, Huanrui, et al. "DeepHoyer: Learning Sparser Neural Network with Differentiable Scale-Invariant Sparsity Measures.
- **[ICLR 2018]** Wen, Wei, et al., Learning Intrinsic Sparse Structures within Long Short-Term Memory.
- **[ICASSP 2019]** Zhang, Jingchi, et al., "Learning Efficient Sparse Structures in Speech Recognition .
- **[DAC 2016, Best Paper Nomination]** Wen, Wei, et al., A New Learning Method for Inference Accuracy, Core Occupation, and Performance Co-optimization on TrueNorth Chip.
- **[AICAS 2019, Best Paper Nomination]** Yang, Huanrui, et al., Exploration of Automatic Mixed-Precision Search for Deep Neural Networks.
- **[ICLR 2021]** Yang, Huanrui, et al., BSQ: Exploring Bit-Level Sparsity for Mixed-Precision Neural Network Quantization.
- **[NeurIPS 2017, Oral]** Wen, Wei, et al., Terngrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning.
- **[DATE 2017, Best Paper Nomination]** Mao, Jiachen, et al., Modnn: Local Distributed Mobile Computing System for Deep Neural Network.
- **[ICCAD 2017]** Mao, Jiachen, et al., MeDNN: A Distributed Mobile System with Enhanced Partition and Deployment for Large-scale DNNs.
- **[SenSys 2021]** Li, Ang, et al., FedMask: Joint Computation and Communication-efficient Personalized Federated Learning via Heterogeneous Masking.
- **[MLSys 2024]** Du, Zhixu, et al., SiDA: Sparsity-Inspired Data-Aware Serving for Efficient and Scalable Large Mixture-of-Experts Models.
- **[ICML 2024]** Sun, Jingwei, et al., FedBPT: Efficient Federated Black-box Prompt Tuning for Large Language Models.
- **[ISCA 2025]** Cheng, Feng, et al., Ecco: Improving Memory Bandwidth and Capacity for LLMs via Entropy-aware Cache Compression.
- **[arXiv' 24]** Wang, Qinsi, et al., CoreInfer: Accelerating Large Language Model Inference with Semantics-Inspired Adaptive Sparse Activation

Duke

“Real” Heroes Behind the Scenes

Wei Wen, Meta

Huanrui Yang, U. of Arizona

Jingchi Zhang, Google

Jiachen Mao, Meta

Ang Li, U. of Maryland

Jingwei Sun, U. of Florida

Zhixu Du, Duke

Qinsi Wang, Duke

Feng Cheng, Duke

Acknowledgements

NSF
U.S. Department of Homeland Security
AFRL
AIR FORCE RESEARCH LABORATORY
Army Research Office — Science to Shape the Future of the Army — 1951
Department of Energy — United States of America
SAMSUNG
hp
amazon
Qualcomm
Cadence
Lenovo

Duke

Q&A