MOFFETT AI

Hardware Support Requirement of Sparse ML Inference

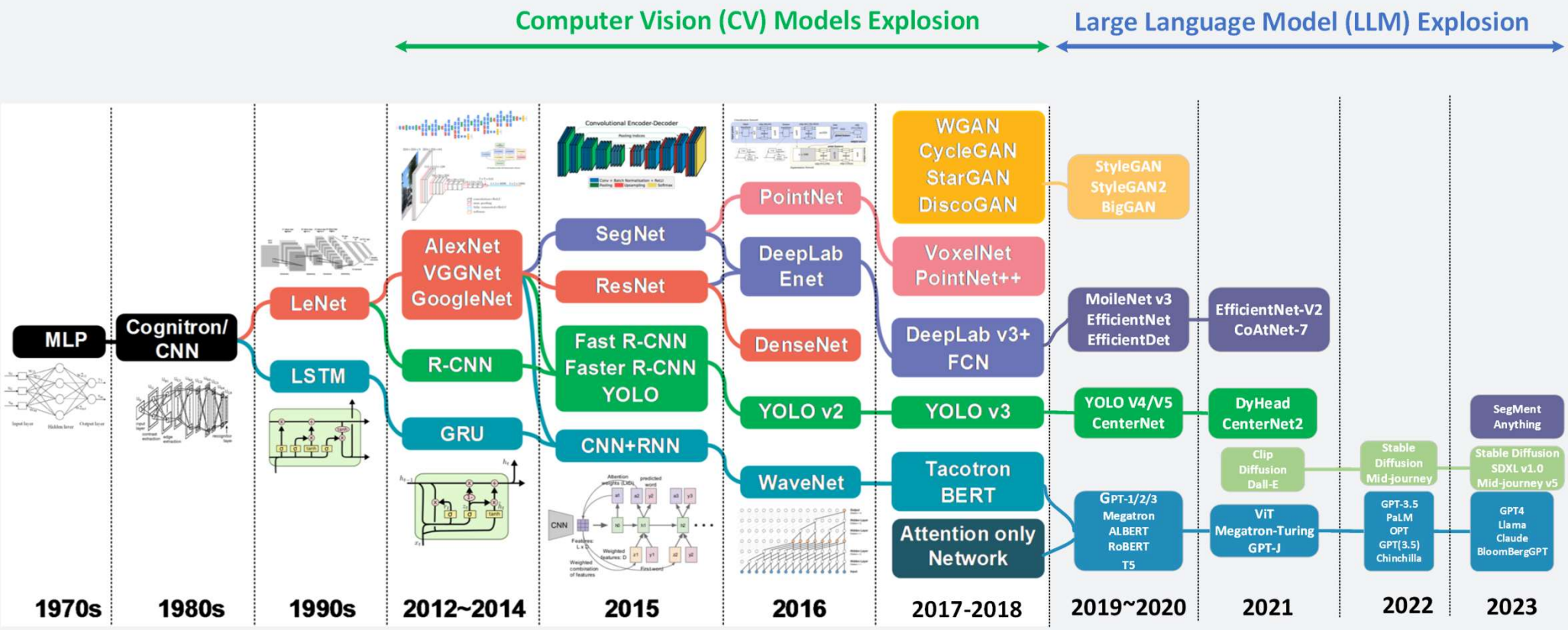
Dr. Zhibin Xiao
Chief Architect and Co-founder
Moffett AI

Outline

- **Introduction to ML Inference**
- Sparsity in ML Inference
- Hardware Software Co-design for Sparsity
- Case Studies: Sparsity Support in CPU, GPU and AI Chips
- Summary

A Brief History of AI Models

MOFFETT AI



*Source: Partly from **Hoi-Jun Yoo**, ISSCC 2019

A Brief History of LLMs (2019 – 2023)

MOFFETT AI

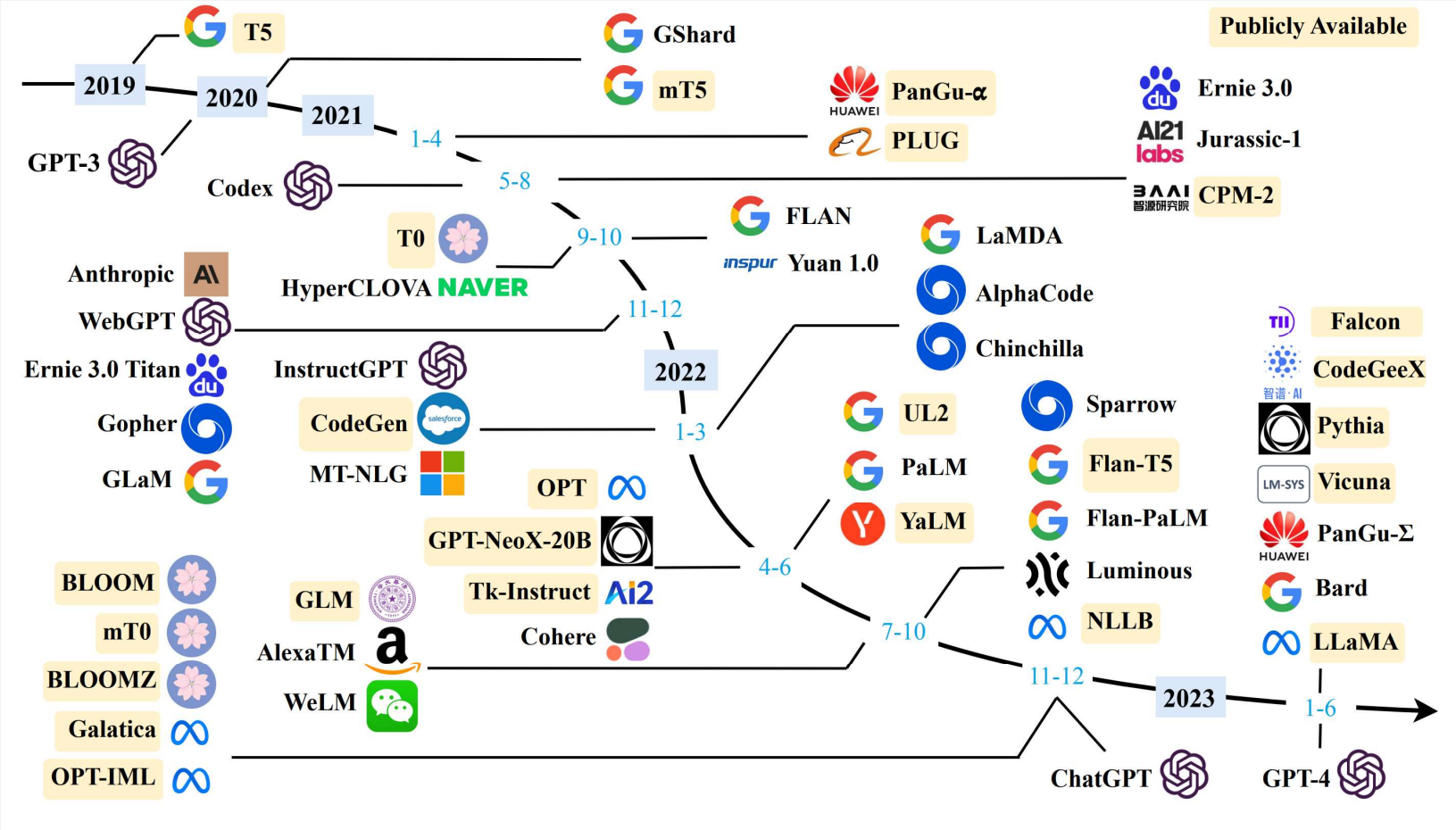
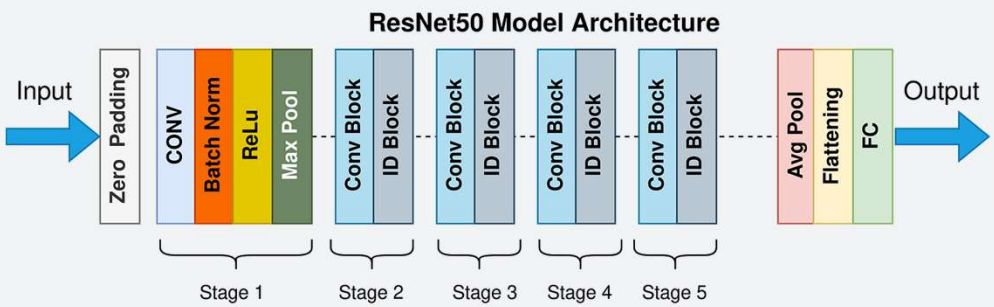


*image credit: Wayne Xin Zhao, et.al, “A Survey of Large Language Models”

Introduction to ML Inference

MOFFETT AI

- ML Model Operations Converges to a small subset of operators
  - ONNX v1.15.0 (192 Operators)
  - Key operators:
    - >90% of Number of Parameters and Computation FLOPS
    - Convolution, Matrix Multiplication, Inner Product, Element-wise Addition, Mean, Reshape, etc.



ResNet50: Conv, Matrix Multiplication, Pooling, ReLU

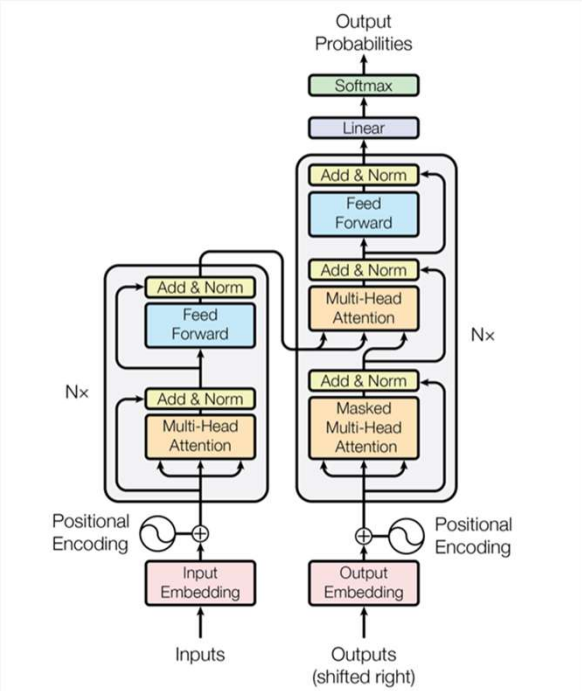


Figure 1: The Transformer - model architecture.

Transformer: Matrix Multiplication, Element-wise Operations, GELU, Softmax, Embedding Lookup, etc.

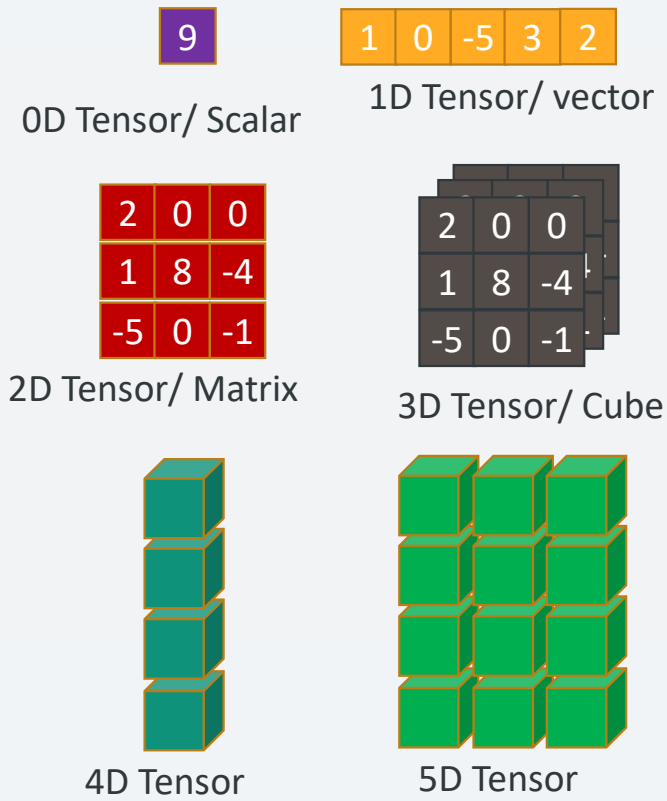
Outline

- Introduction to ML Inference
- **Sparsity in ML Inference**
- Hardware Software Co-design for Sparsity
- Case Studies: Sparsity Support in CPU, GPU and AI Chips
- Summary

Sparsity ML Inference

MOFFETT AI

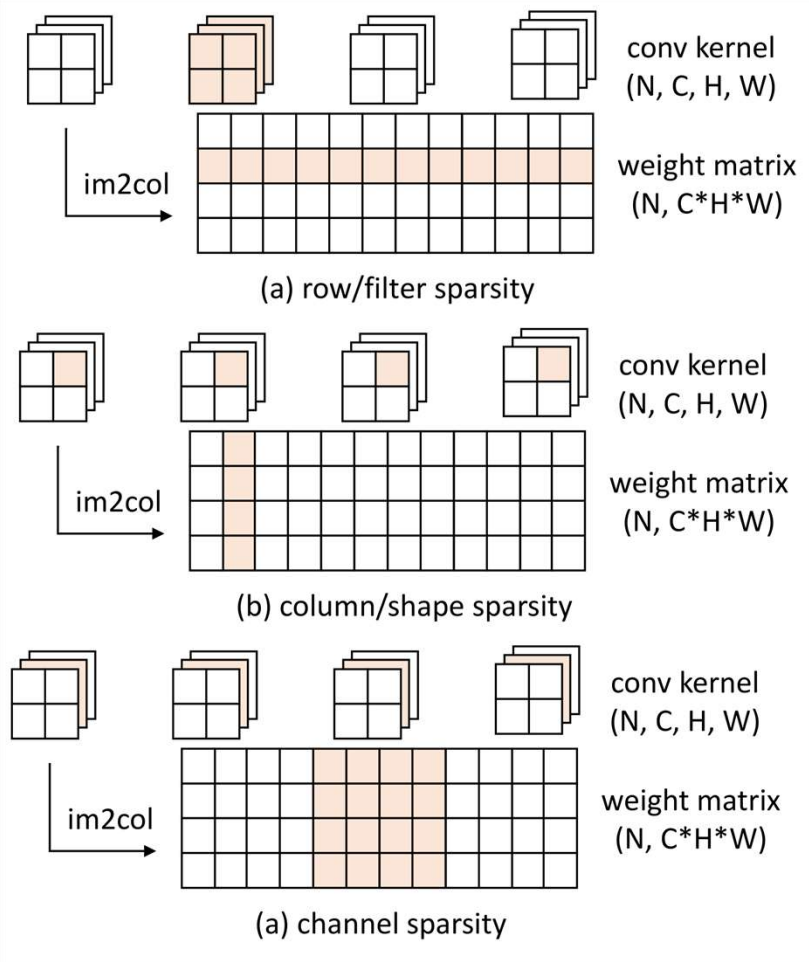
- The core of ML inference is **Tensor Algebra**
  - Tensor format, E.g., a typical 4D tensor (NHWC) in image processing
- Sparsity in ML Inference
  - Zero naturally exist or can be induced in Tensors
  - No need to store zero or compute zero in a tensor
    - ☺ Save storage, computation time, memory bandwidth, reduce power
    - ☹ Extra HW cost for compression, decompression, schedule (limit the throughput and power/area overhead) **"Sparsity Tax"**



Sparsity on Convolution Kernels

MOFFETT AI

- Convolution Kernels can be converted to weight matrix
  - Filter sparsity
  - Shape sparsity
  - Channel sparsity



Sparsity is an Active Algorithm Research Area

MOFFETT AI

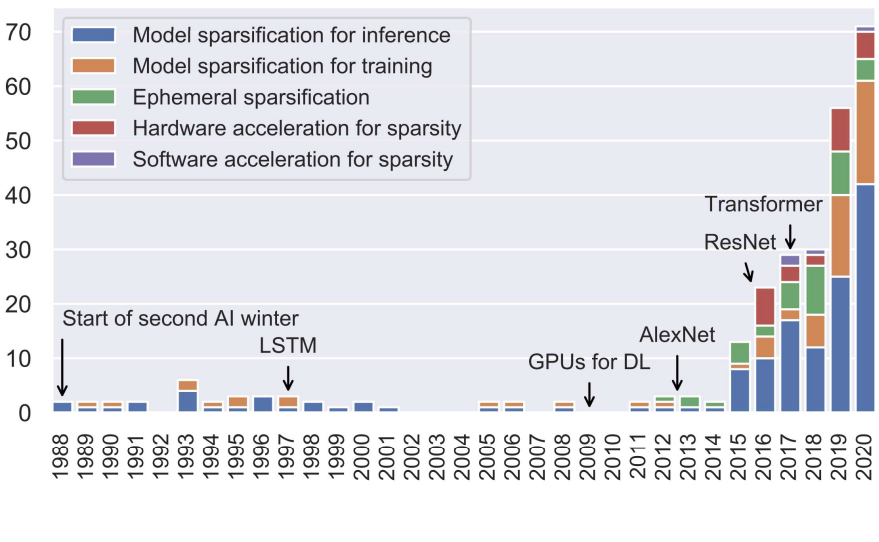
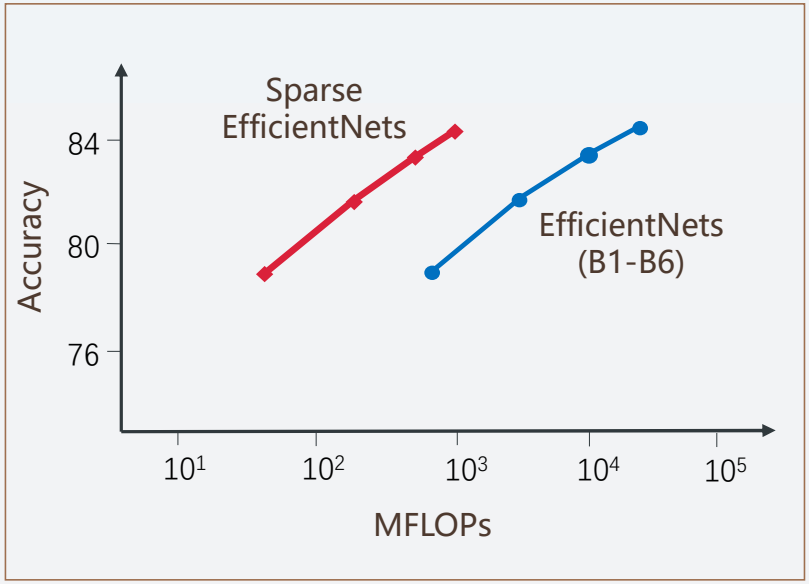


Fig. 1. Literature development over the years.



Google & Deepmind paper, "Fast Sparse ConvNets"

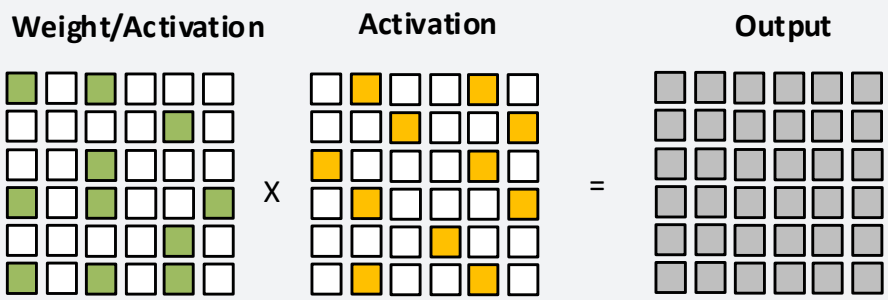
- The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks (MIT) – ICLR 2019 Best Paper

  → Any dense neural network contains one sparse neural network

Image Credit: Torsten Hoefler et.al, Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks

Type of Sparsity in ML Inference

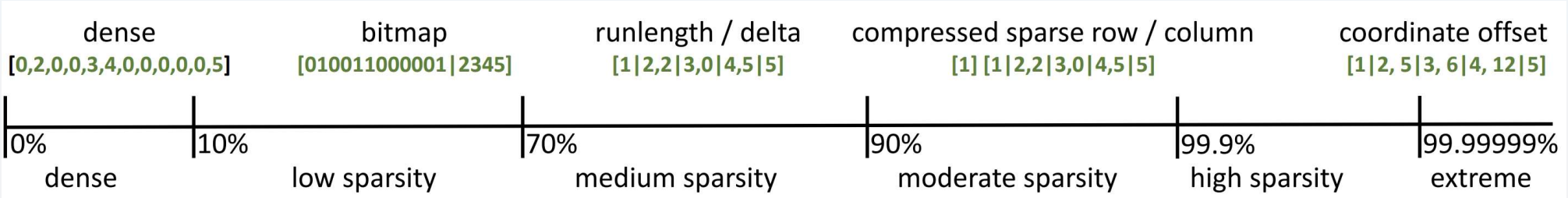
MOFFETT AI



- **Static and Dynamic Sparsity**
  - Weight Sparsity
    - Static Weight Sparsity (Pruning)
    - Dynamic Weight Sparsity (Conditional)
  - Activation Sparsity
    - Contextual/Attention Sparsity (LLM)
    - Feature Sparsity (CV)
- **Sparsity Granularity**
  - Coarse-granularity Sparsity
  - Fine-grained Sparsity
- **Sparsity Patten**
  - Structured sparsity
  - Unstructured sparsity

Sparse Matrix Storage Format

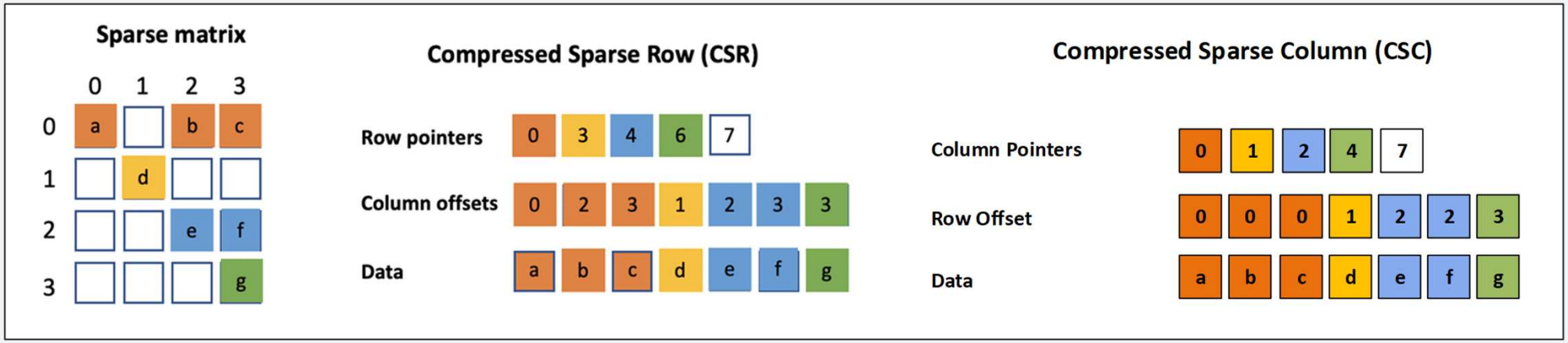
MOFFETT AI



- Bitmap
- Run-length /delta
- Compressed Sparse Row / Column (CSR/CSC)
- Coordinate Offset (index, value)
- Hierarchical Hybrid Sparse Format

Image Credit: Torsten Hoefler et.al, Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks

Sparse Matrix Format: CSR and CSC Format



CSR Format

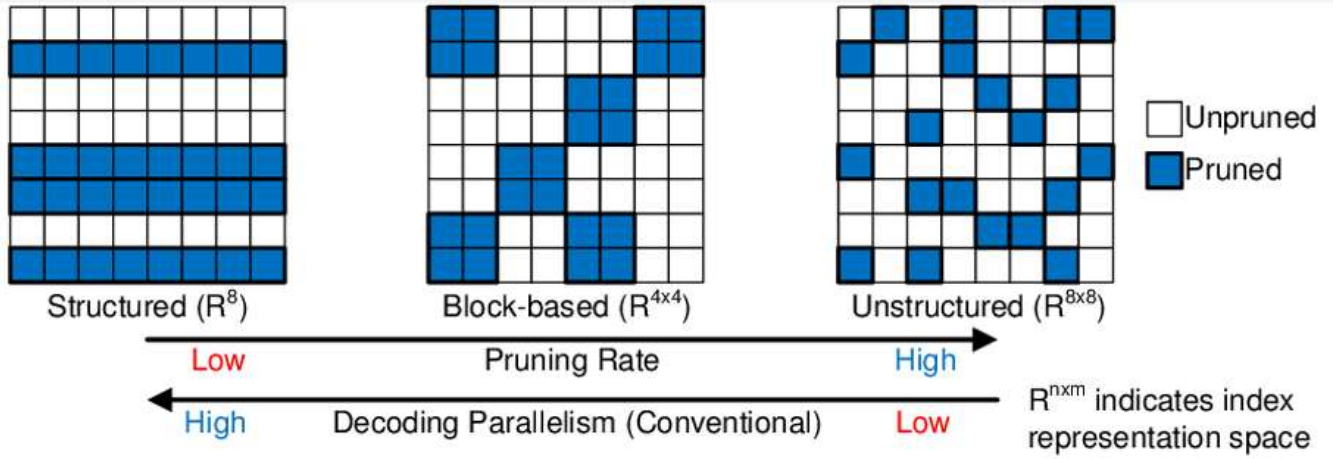
- Data: an array for all non-zero values
- Column_offsets[i]: records the actual column index of the data[i]
- **Row_pointers[i]: records the number of non-zero of of all (i-1) rows**

CSC Format

- Data: an array for all non-zero values
- Row_offsets[i]: records the actual row index of the data[i]
- **Column_pointers[i]: records the number of non-zero of of all (i-1) columns**

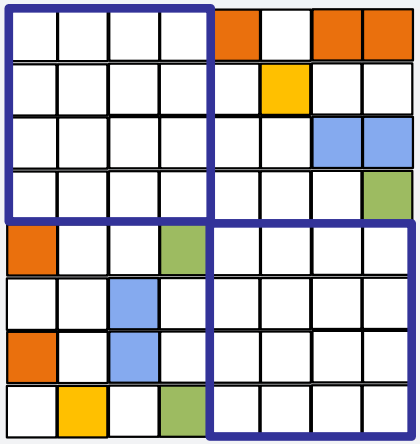
Sparse Matrix Format: Coordinate Index and Hierarchical Hybrid Format

MOFFETT AI



Coordinate Index

Structured and Unstructured Sparsity



Hierarchical Hybrid Format

Top-level: bit-vector format: (0, 1, 1, 0)

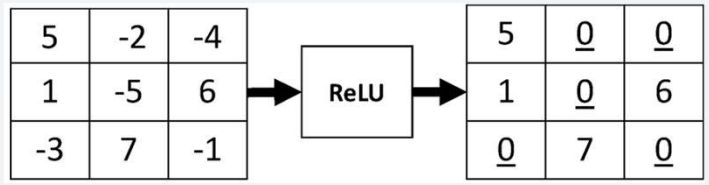
Block-level: CSR/CSC/Coordinate Offset

Image Credit: Se Jung Kwon et. al, Structured Compression by Weight Encryption for Unstructured Pruning and Quantization

Activation Sparsity and Conditional Sparsity

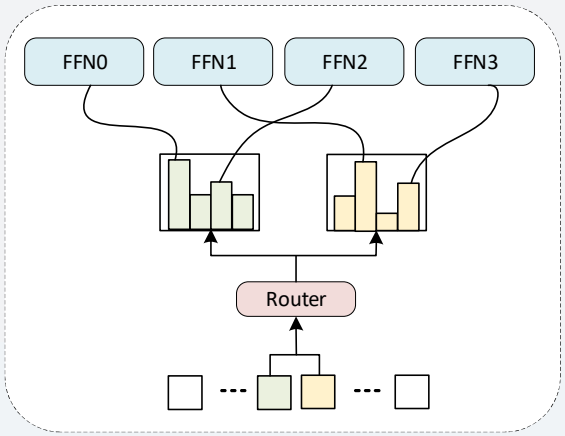
MOFFETT AI

+ **Activation Sparsity**
  - Sparse Input data or induced activation matrix based on activation functions (ReLU/Softmax)
  - Dynamic Sparsity (run time)

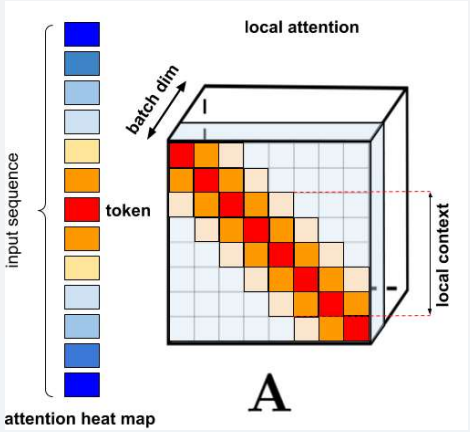


Activation Sparsity

+ **Conditional Sparsity**
  - Conditions are normally calculated at runtime based on input
  - Use the condition to decide the weight matrix patterns
    - block level
    - sub-model level (MoE)
  - Use the condition to decide token correlation
    - Sparse Attention in LLM



Mixture of Experts (MoE)



Sparse Attention

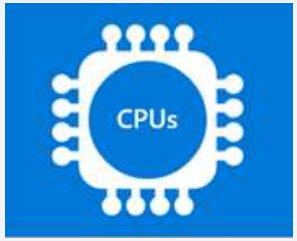
Outline

- Introduction to ML Inference
- Sparsity in ML Inference
- **Hardware Software Co-design for Sparsity**
- Case Studies: Sparsity Support in CPU, GPU and AI Chips
- Summary

Sparsity Tax and HW/SW Scenario

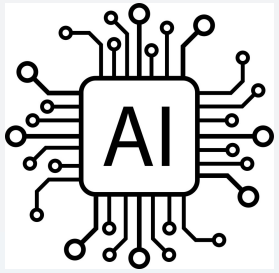
MOFFETT AI

+ **Sparsity Tax**
  - Extra storage overhead
  - Extra decompression/compression overhead
  - Model accuracy loss
  - No wall-clock speedup or even slower without special sparse accelerators
+ **Sparsity Support on devices**
  - CPU
  - GPU
  - AI Accelerators



Highly-sparse Matrix/Vector
HPC field

Coarse-grained sparsity
Fine-grained 2:4
Structure Sparsity

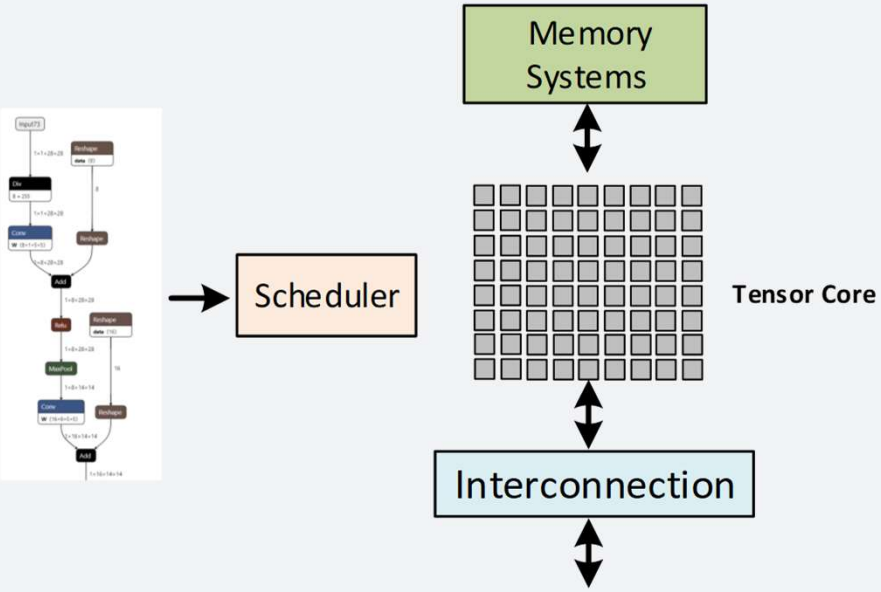


All sparsity type
(Dynamic, Static, Structured, non-structured, fine-grained, coarse-grained, conditional execution)

Sparse AI Accelerator Architecture Design Consideration

MOFFETT AI

+ **Sparsity Support Impact on AI Accelerator Design**
  - Programming Model
  - Scheduler (Data Flow)
  - **Memory Systems**
  - **Tensor Core Processing Datapath**

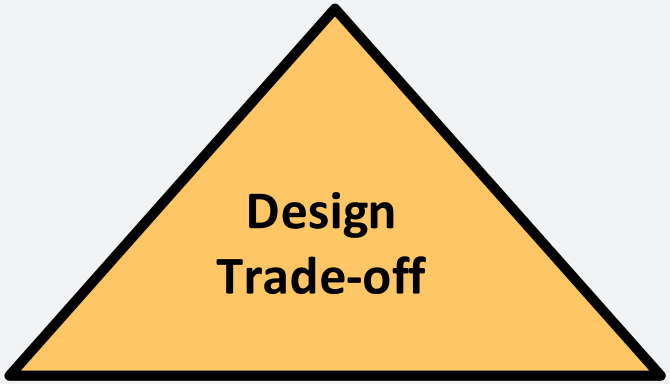


Challenges in Designing Sparse Accelerators

MOFFETT AI

TCO Saving
Wall-clock speedup
Power saving
Area Saving (memory, die size)

Benefits



Sparsity Tax

Decompression/compression
Workload Imbalance
Area and power overhead

Model Accuracy

Architecture/Algorithm Co-design
Application Requirements
Cost to Retrain
Time-to-Market

Outline

- Introduction to ML Inference
- Sparsity in ML Inference
- Hardware Software Co-design for Sparsity
- **Case Studies: Sparsity Support in CPU, GPU and AI Chips**
- Summary

An Overview of Mainstream AI Accelerator Architecture

MOFFETT AI

Popular AI Accelerators

- CPU (X86, RISC-V): Vector/Matrix Instruction Extension
- Nvidia Tensor Core: 4x4 GEMM
- Huawei Ascend: 16x16 GEMM + VPU
- Google TPU: Systolic Array + VPU
- Graphcore: Massively Parallel BSP Cores
- SambaNova: Dataflow RDU
- Cerebras: Wafer-scale many-core architecture
- Habala Labs/Intel Spring Hill: DSP Array + GEMM
- Cambricon/Hanguang 800/NVDLA/Tesla FSD: DSA Accelerators

Special Technology AI Accelerators

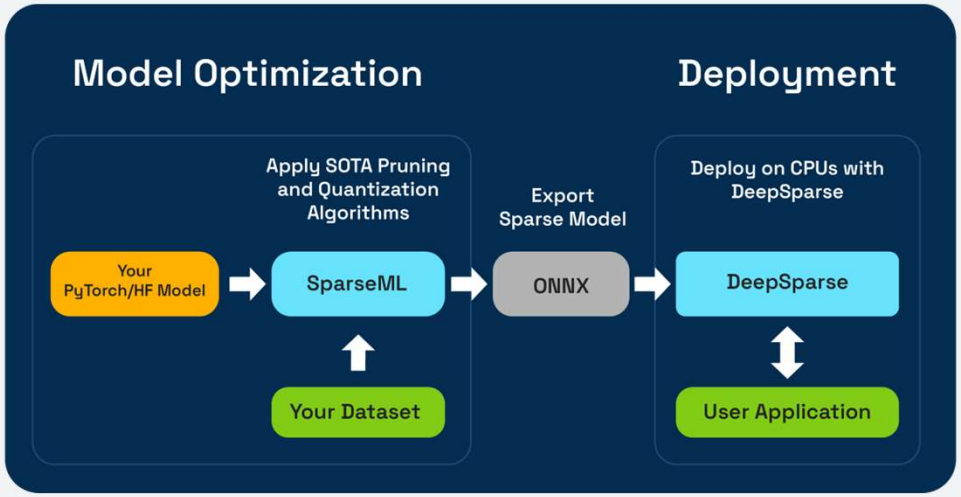
- Spiking Neural Nets and Neuromorphic Architectures
- Resistor/Memristor matrices and Analog Computing
- Optical and Spintronics Implementations

- **Key buzz words:** Systolic Array, Tensor Core, Vector Core, Many-core, DSA, Dataflow
- **Special Technology AI Accelerators:** Very efficient for specific applications, limited operator support

Sparsity Support CPUs

MOFFETT AI

- **CPU offers thread-level parallelism and dense vector/matrix extension**
  - Limited by low peak MAC performance of CPUs
  - Limited by SW for sparse matrix compress and decompress
    - Limit speedup for sparse matrix
    - Available Intel Sparse BLAS support



NeuralMagic's DeepSparse Inference Runtime on CPU

Image Source: https://github.com/neuralmagic/deepsparse

Sparsity Support on GPU Tensor Core – Micro-52 (2019)

MOFFETT AI

- **Weight Sparsity**
  - Structured
  - Vector-wise Balanced Sparse Pattern
- **Minimal change to Volta GPU Tensor core**
  - 75% sparsity with 1.49x speedup vs dense tensor core

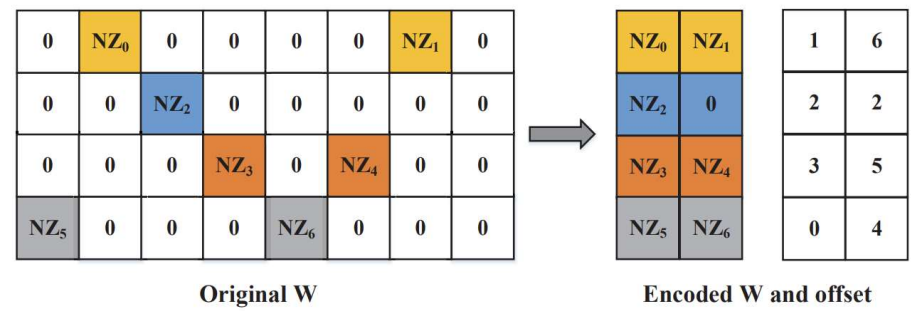


Figure 5: An example of vector-wise sparse matrix encoding with L=8 and K=2. Two non-zero elements in a row vector are compressed into one compact vector associated with their indices. All row vectors are encoded to the same length. If a row vector has less non-zero elements than the compact vector length K, the empty entries are padded with zeros.

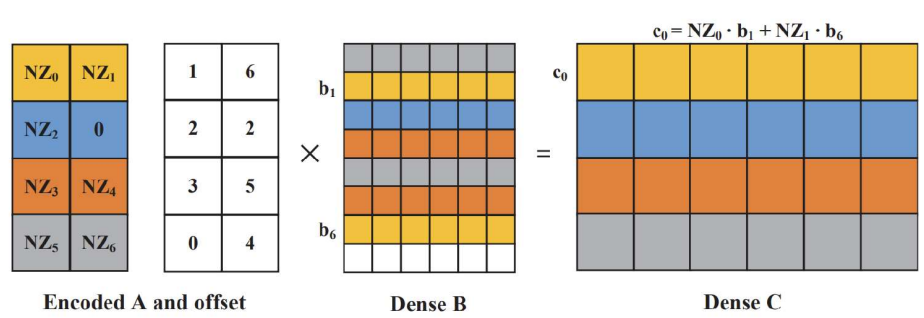


Image Source: Maohua Zhu et. al, Sparse Tensor Core: Algorithm and Hardware Co-Design for Vector-wise Sparse Neural Networks on Modern GPUs

Nvidia Ampere/Hopper Sparse Tensor Core

MOFFETT AI

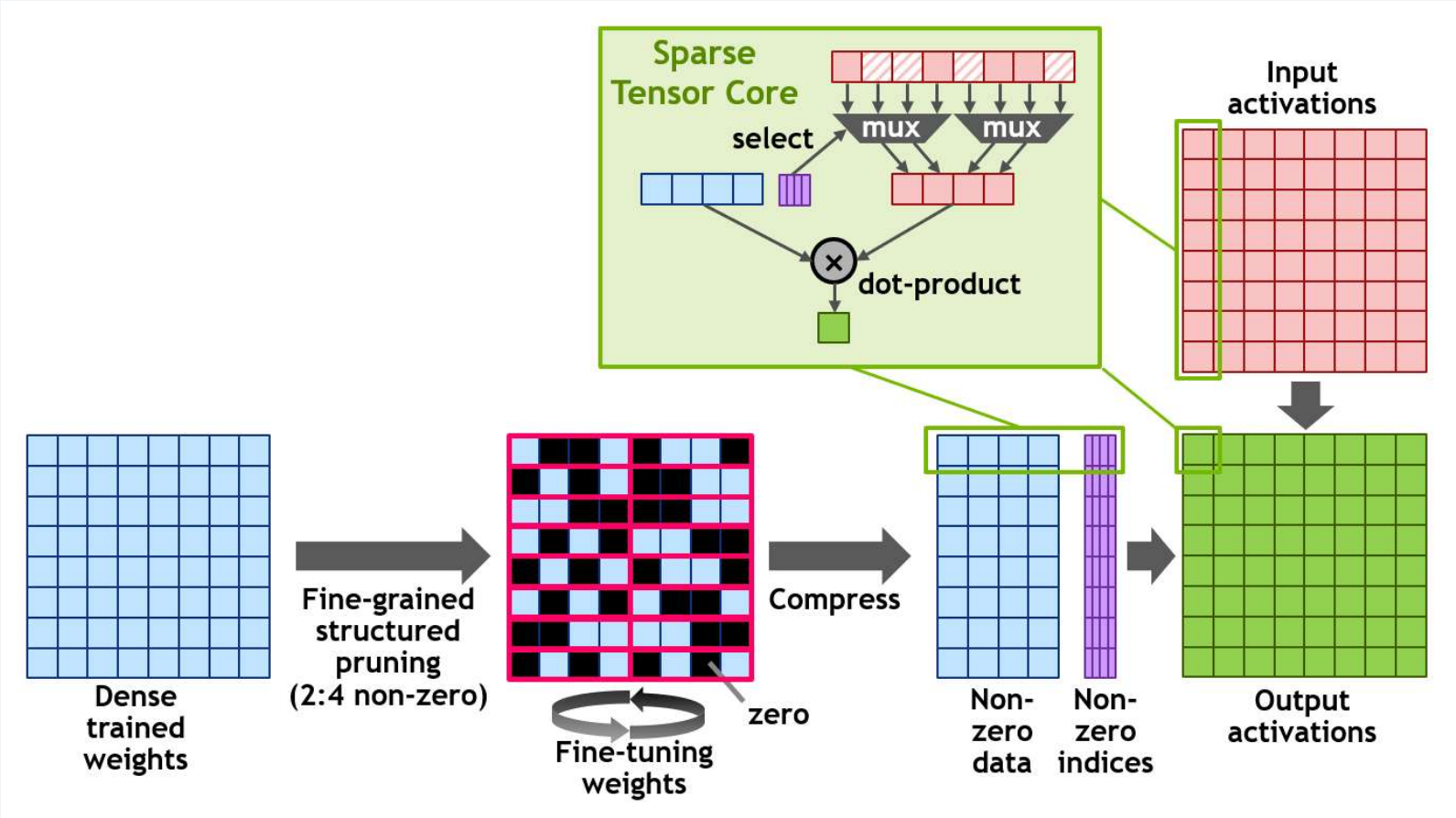


Image Source: Nvidia website

Nvidia Ampere/Hopper Sparse Core Performance

MOFFETT AI

Speedup on Matrix-Multiplication

M	N	K	Speedup
1024	8192	1024	1.44x
1024	16384	1024	1.73x
4096	8192	1024	1.53x
4096	16384	1024	1.78x

Speedup on Convolution

N	C	K	H,W	R,S	Speedup
32	1024	2048	14	1	1.52x
32	2048	1024	14	1	1.77x
32	2048	4096	7	1	1.64x
32	4096	2048	7	1	1.75x
256	256	512	7	3	1.85x

NETWORK	DATA TYPE	SCENARIO	PERFORMANCE
BERT-Large	INT8	BS=256, SeqLen=128	6200 seq/s
		BS=1-256, SeqLen=128	1.3X-1.5X
ResNeXt-101_32x16d	FP16	BS=256	2700 images/second
		BS=1-256	Up to 1.3X
	INT8	BS=256	4400 images/second
		BS=1-256	Up to 1.3X

End to End Inference Speedup

Data Source: Nvidia website

Sparsity Support on TPUs – (ICS 2020)

MOFFETT AI

- **TPU is difficult to support sparsity**
  - Static and fixed data-flow
- Research on SparseTPU
  - Packing technique to condense sparse matrices
  - Still calculate some zeros and no reduce of memory footprint
  - Performance gain:
    - 16.08x and 4.39x and 19.79x lower energy (INT8/FP32)
  - Sparsity tax:
    - 12.93% area overhead
    - 4.14% energy overhead (FP32 TPU)

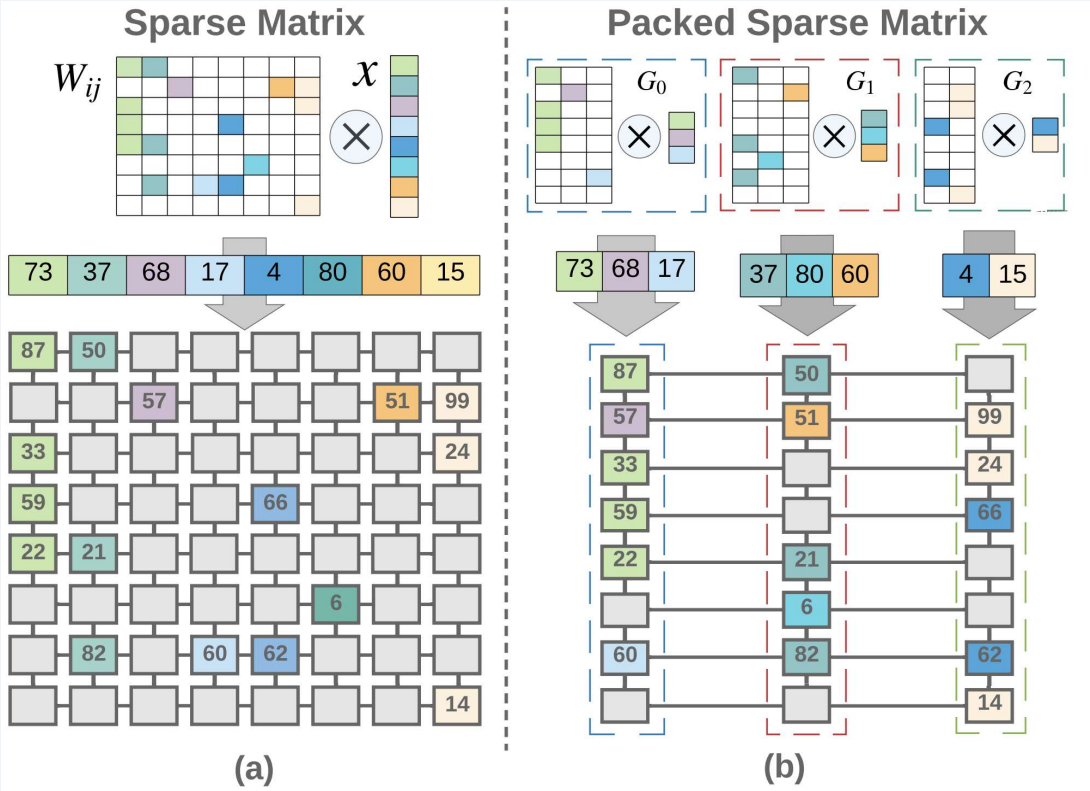


Image Source: Xin He, Sparse-TPU: Adapting Systolic Arrays for Sparse Matrices, ICS 2020

Sparsity Support on TPUs: – (ICS 2020)

MOFFETT AI

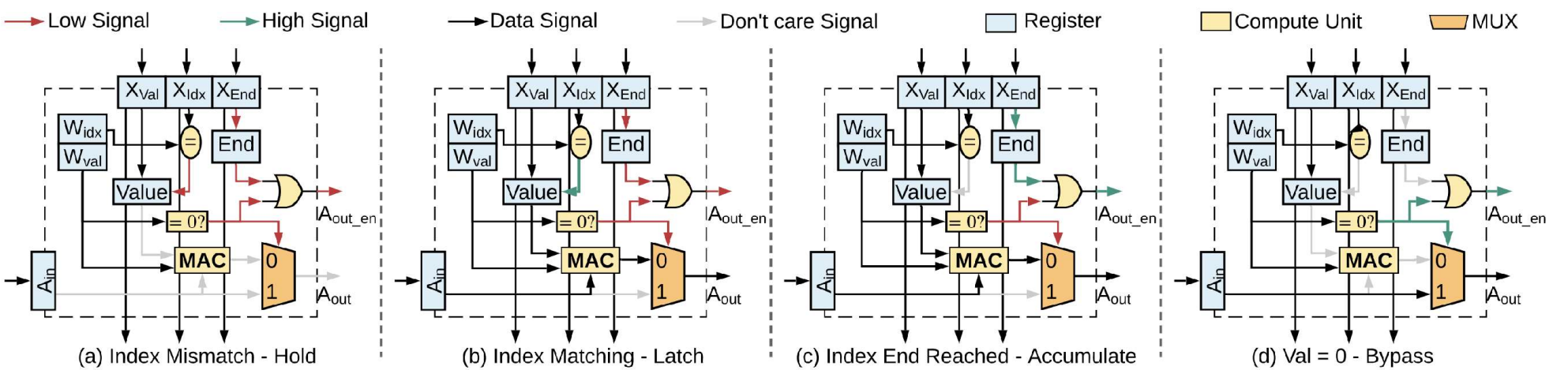


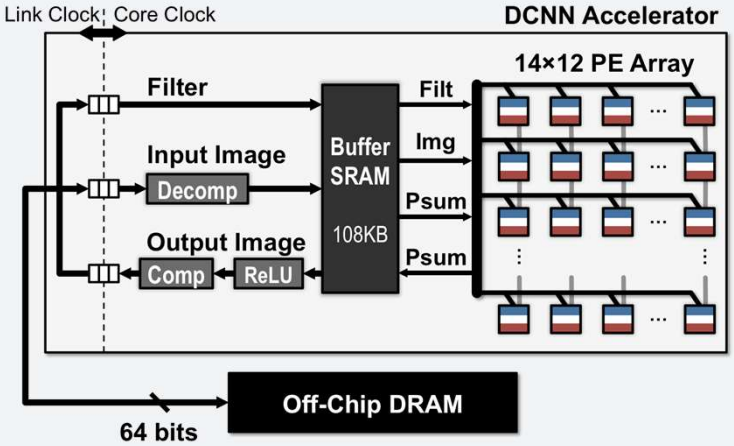
Figure 7: Microarchitecture and dataflow within a PE in different modes of operation. (a) Hold: The PE holds the accumulated result when the index mismatches. (b) Latch: The PE latches the input vector element when the input index matches (c) Accumulate: When the end of a vector group arrives, the PE calculates the MAC result and updates the Accu register of the rightward PE. (d) Bypass: If the matrix value held in the PE is 0, the data coming from leftward is bypassed rightward.

Image Source: Xin He, Sparse-TPU: Adapting Systolic Arrays for Sparse Matrices, ICS 2020)

MIT Eyeriss Project – Eyeriss v1 (2016 ISSCC)

MOFFETT AI

+ **One of the earliest AI Accelerator chip**
  - A Spatial Multi-PE architecture
  - Support Weight Sparsity by reducing memory footprint and bandwidth
  - Saving power by clock gating PE for zero operands
  - No wall-clock speedup



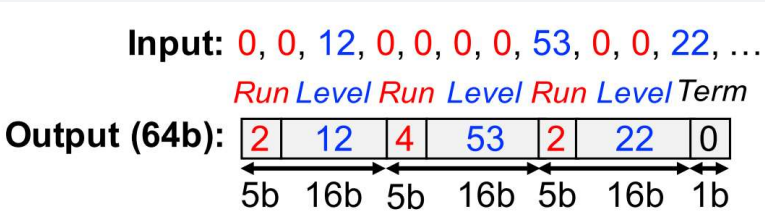


Fig. 8. Encoding of the RLC.

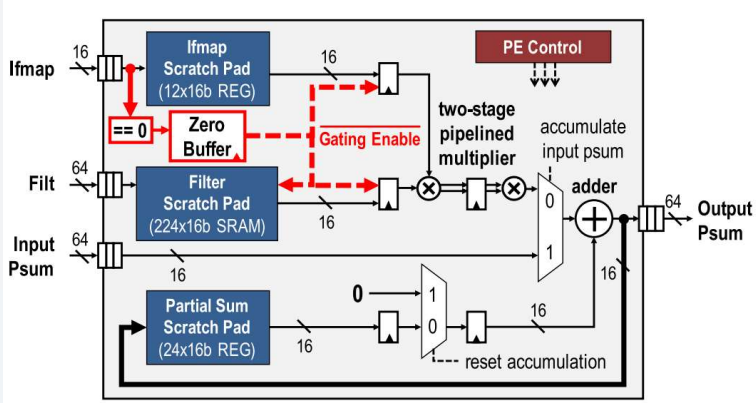


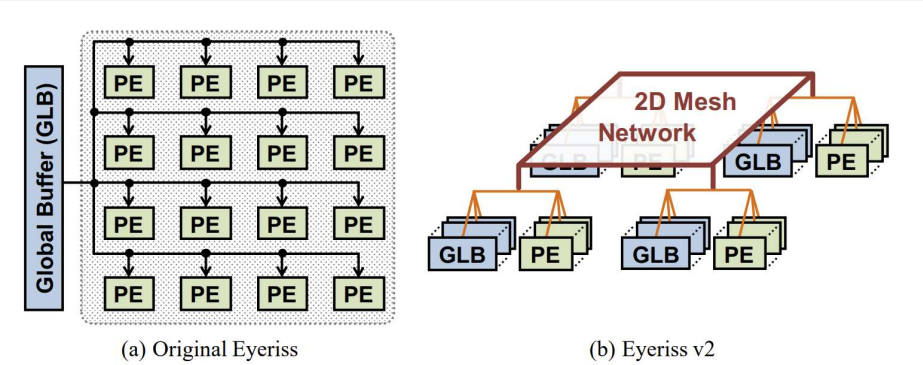
Fig. 12. PE architecture. The datapaths in red show the data gating logic to skip the processing of zero ifmap data.

Image Source: Yu-Hsin Chen et. al Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks

MIT Eyeriss Project – Eyeriss v2 (2018)

MOFFETT AI

+ **Compared to Eyeriss v1**
  - A Scalable Architecture
  - Change of matrix compressed format
  - Dual-sparsity Support
  - Wall-clock speedup



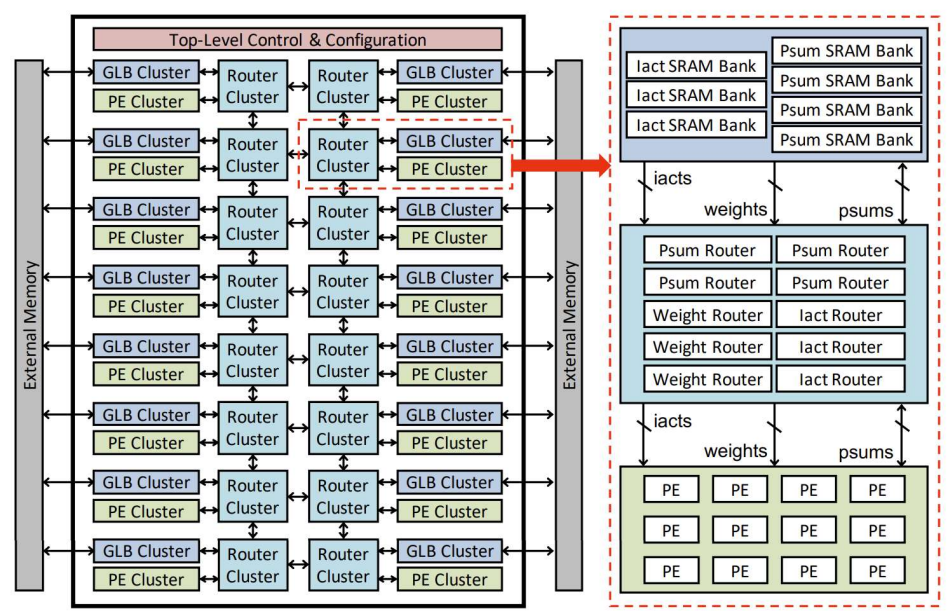
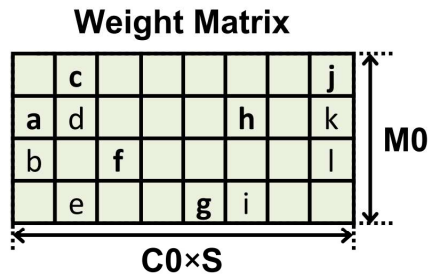


Image Source: Yu-Hsin Chen et. al Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices

MIT Eyeriss Project – Eyeriss v2 (2018)

MOFFETT AI



CSC Compressed Data:

data vector: {**a**, b, **c**, d, e, **f**, **g**, **h**, i, **j**, k, l}
count vector: {1, 0, 0, 0, 1, 2, 3, 1, 1, 0, 0, 0}
address vector: {0, 2, 5, 6, 6, 7, 9, 9, 12}

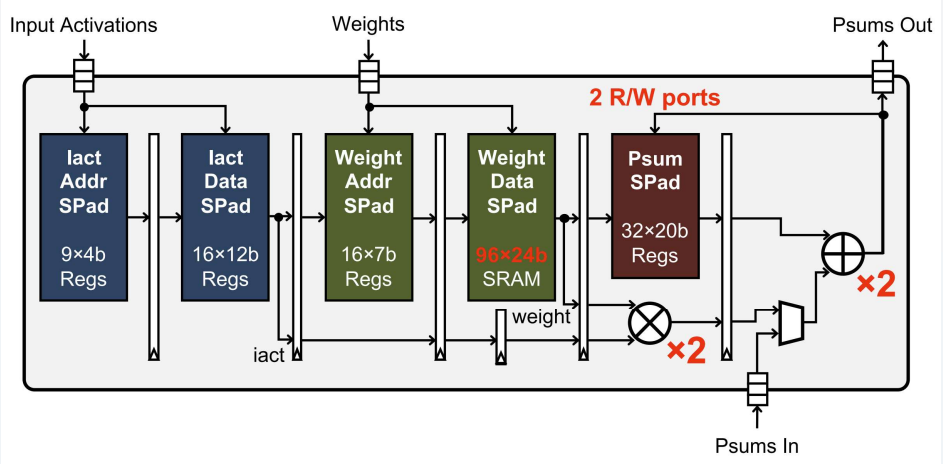


Image Source: Yu-Hsin Chen et. al Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices

EIE: Efficient Inference Engine on Compressed Deep Neural Network (2016)

MOFFETT AI

+ **One of the earliest AI Accelerator research**
  - A Spatial Multi-PE architecture
  - Support dual sparsity by reducing memory footprint and bandwidth and save wall-clock speedup
  - Weight matrices: CSC format
  - Proposed an activation buffer before different PEs for workload balance
  - Use activation to lookup compressed weight

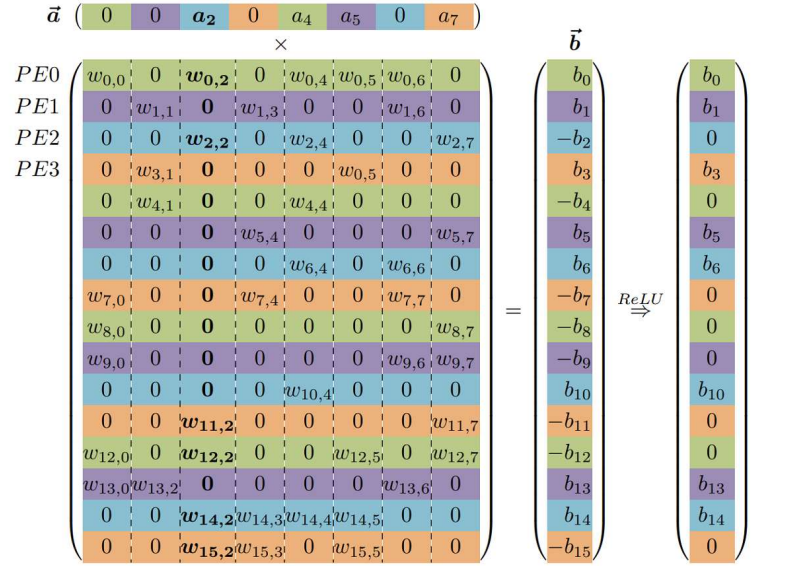


Figure 2. Matrix W and vectors a and b are interleaved over 4 PEs. Elements of the same color are stored in the same PE.

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

Figure 3. Memory layout for the relative indexed, indirect weighted and interleaved CSC format, corresponding to PE_0 in Figure 2.

Image Source: Song Han et. al, EIE: Efficient Inference Engine on Compressed Deep Neural Network

EIE uArch

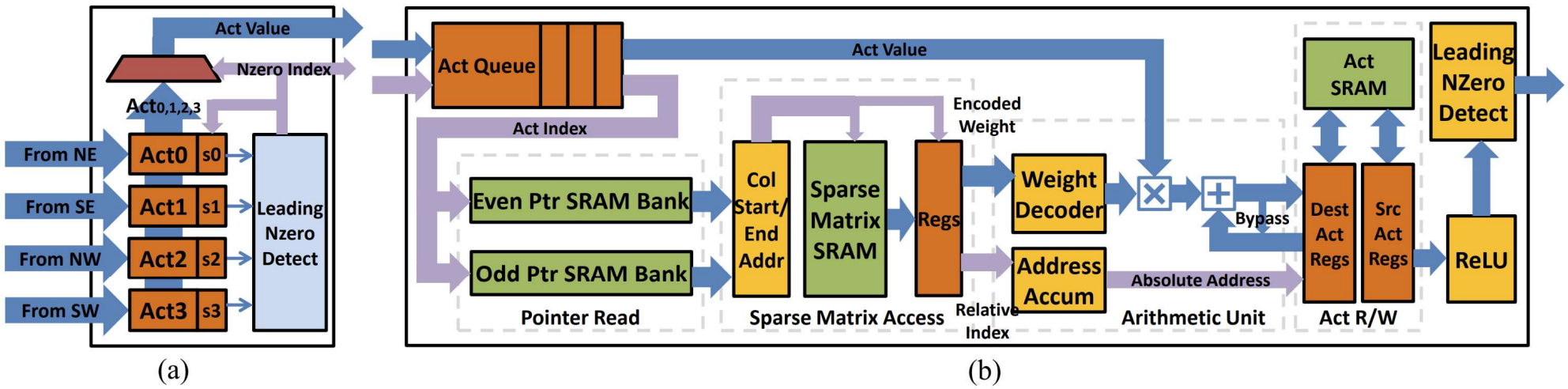


Figure 4. (a) The architecture of Leading Non-zero Detection Node. (b) The architecture of Processing Element.

Image Source: Song Han et. al, EIE: Efficient Inference Engine on Compressed Deep Neural Network

Alibaba Hanguang-800 Sparsity Engine (2020)

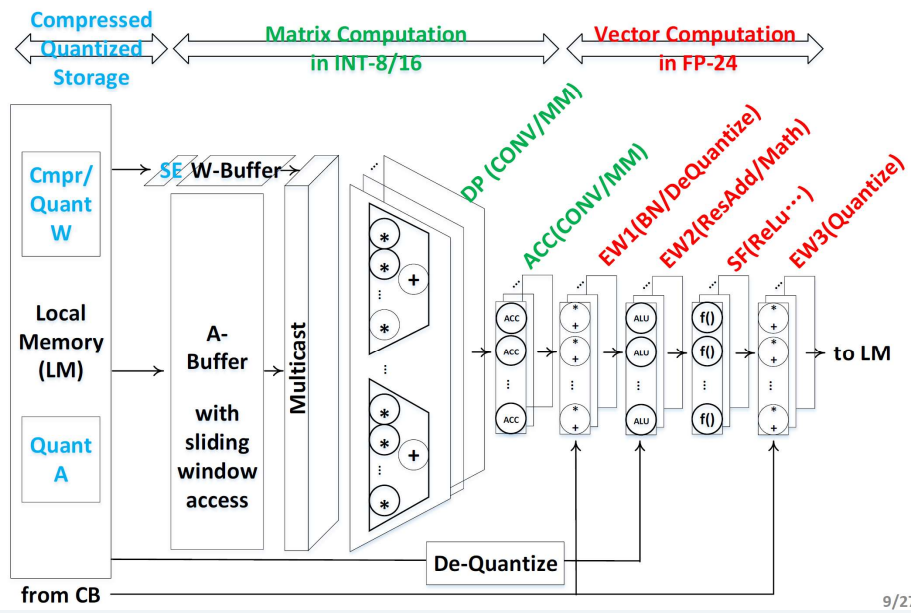
MOFFETT AI

+ **A High-performance Commercial Data-center Inference Chip**
  - DSA architecture
  - Support weight compression in memory to reduce memory footprint
  - No external DDR and all-on-chip Memory
  - Weight matrices: bit-vector representation for low to medium sparsity
  - No wall-clock speedup

Alibaba Group 阿里巴巴集团

Compressed and Quantized Storage/Processing

- **Compressed** Model
  - Sparsity Engine to unpack with bit-masks
  - Pruning is optional though
- **Quantized** computation and storage
- Vector Unit w/ **FP-24**
  - 1sign.8exp.15man



9/27

Source: Hanguang 800 NPU – The Ultimate AI Inference Solution for Data Centers, Hotchips 2020

SambaNova RDU Sparsity Support

MOFFETT AI

+ **A Reconfigurable Dataflow tiled Architecture (RDU series)**
  - Scalable design with on-chip switch connect array of RDUs and memory units
  - Scale-out support
  - Support CSR-like matrix compression
  - Wall clock-time speedup

Sparse Matrix Multiply on RDU

Logical

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -2 \\ 0 & -3 & -4 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} -3 & -4 \\ -10 & -12 \\ -29 & -36 \end{bmatrix}$$

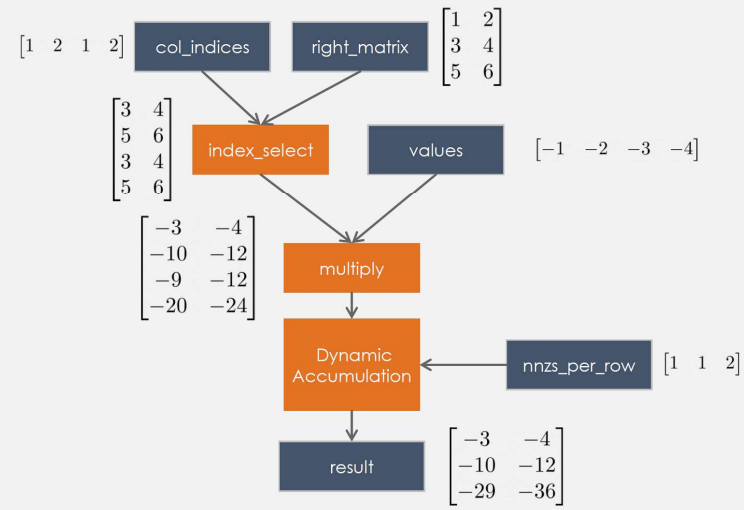
CSR-like Representation

$$col\_indices = \begin{bmatrix} 1 & 2 & 1 & 2 \end{bmatrix}$$

$$values = \begin{bmatrix} -1 & -2 & -3 & -4 \end{bmatrix}$$

$$nnzs\_per\_row = \begin{bmatrix} 1 & 1 & 2 \end{bmatrix}$$

Spatial



* Source: SambaNova SN10 RDU: Accelerating Software 2.0 with Dataflow, Hotchips 2021

Cerebras Sparsity Support

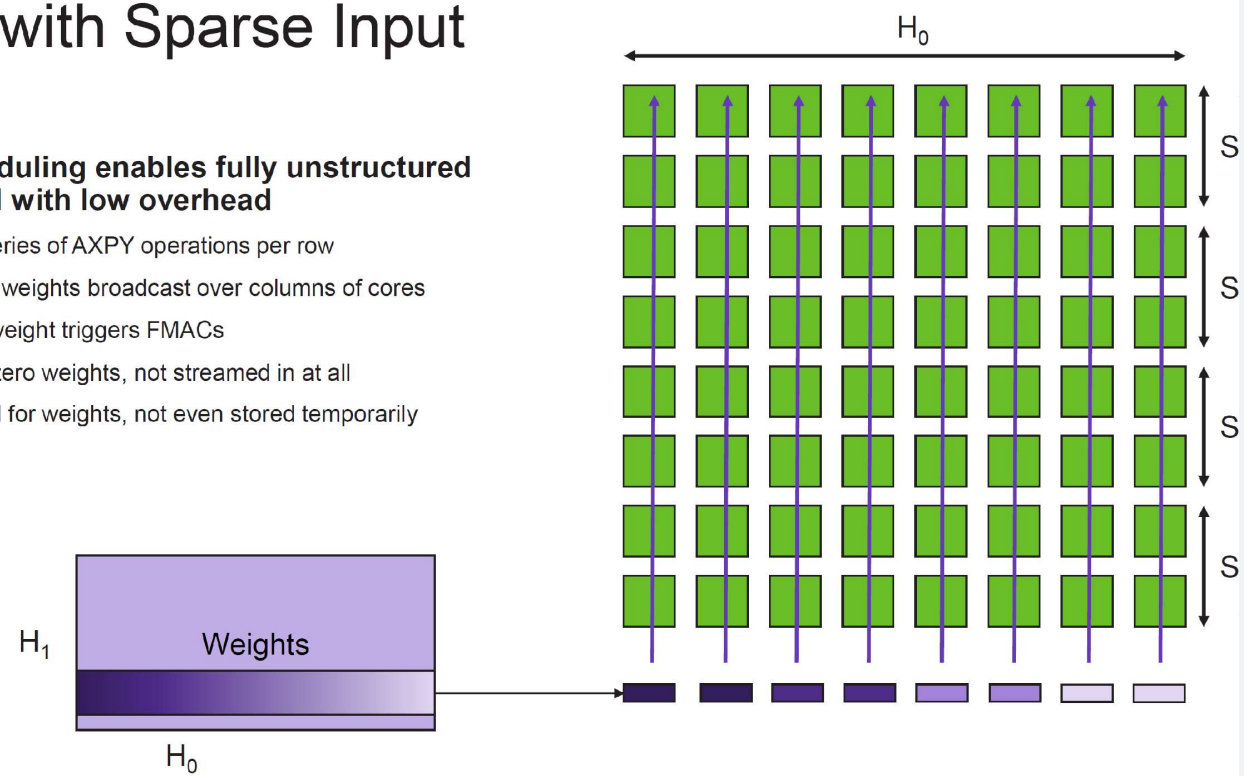
MOFFETT AI

- A commercial data-flow wafer-scale spatial architecture
- Fine-granularity fully unstructured sparse MatMul
- 10x sparse utilization vs. GPU
- Not clear on the weight sparse storage format

GEMM with Sparse Input

Dataflow scheduling enables fully unstructured sparse MatMul with low overhead

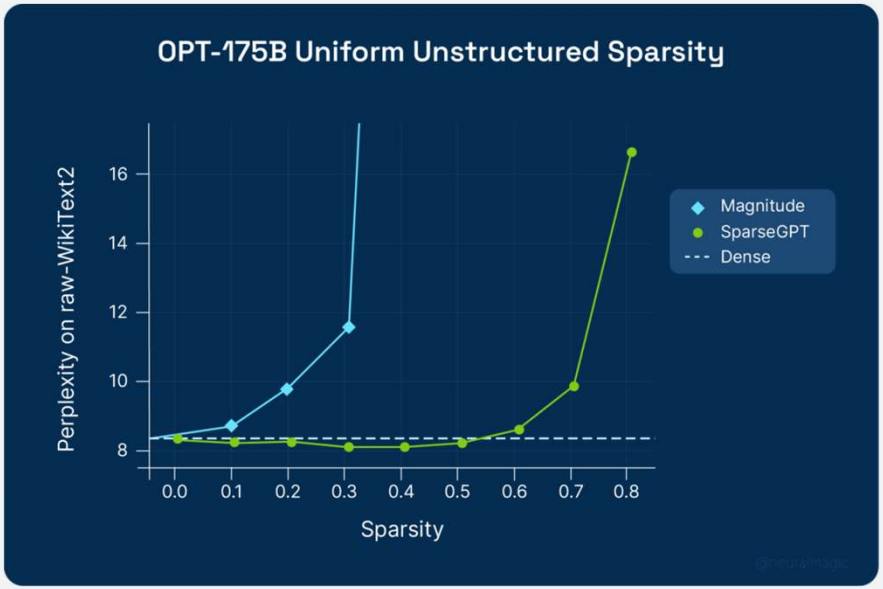
- Executed as a series of AXPY operations per row
- Row of non-zero weights broadcast over columns of cores
- Each individual weight triggers FMACs
- No compute for zero weights, not streamed in at all
- No memory used for weights, not even stored temporarily



Recent Industry Trends on Sparse Transformers

MOFFETT AI

- A One-shot post-training sparse methods
- 50% sparsity to deploy on A100/H100 GPU



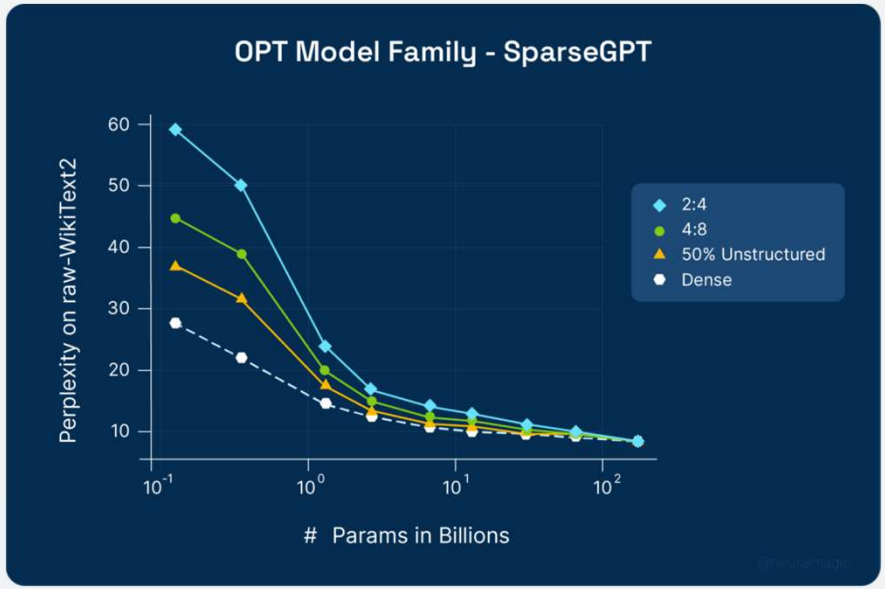


Image credit: https://neuralmagic.com/blog/sparsegpt-remove-100-billion-parameters-for-free/

Recent Industry Trends on Sparse Transformers

MOFFETT AI

- Company: Cerebras
- GPT-3 XL 1.3B parameter model with 84% sparsity
  - 3x fewer inference FLOPS
  - 4.3x fewer parameters
  - No loss in accuracy

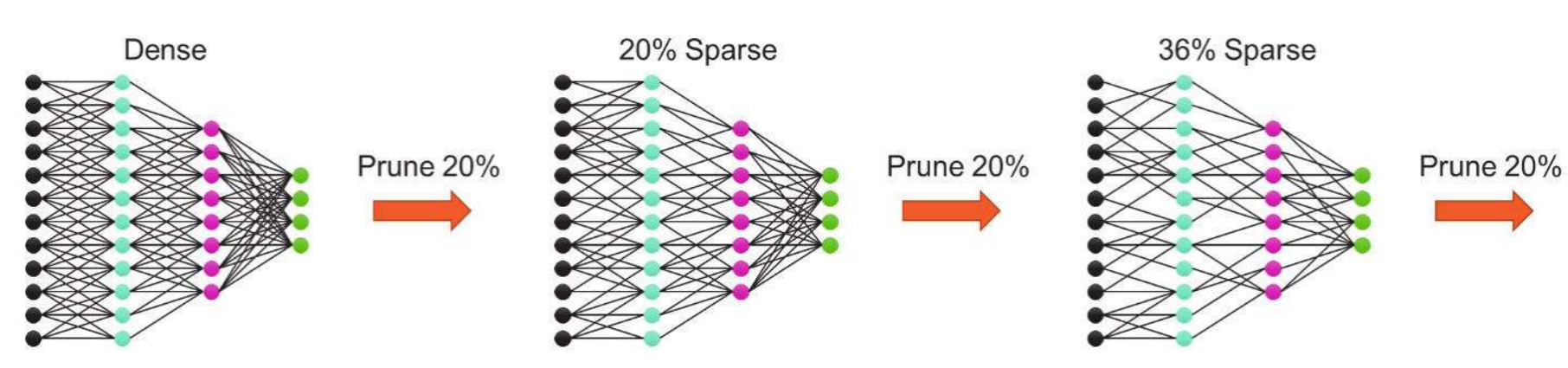


Image credit: Cerebras website

Summary

MOFFETT AI

- **Sparsity is an active research area**
  - Promising direction for both Vision and LLM
  - Save computation, memory bandwidth/capacity and power
  - Reduce TCO
- The memory storage format is the key for hardware support of sparsity
  - Affected by algorithm (sparsity ratio, accuracy)
  - Impact on:
    - Memory system design
    - Datapath design
    - Scheduler design
- Sparse AI Accelerator needs trade off on more dimension
  - Model Accuracy
  - Sparsity overhead
  - Sparsity benefits
- Research and commercial AI accelerators are embracing sparsity support

MOFFETT AI

Moffett Deep-Sparse AI Inference chip will be presented at Tuesday Afternoon Session

Thank you and Questions?

HOT CHIPS