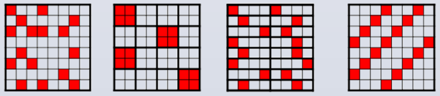
Efficient Sparse NN Processing on Hardware Accelerators: Survey and Insights. Shail Dave and Aviral Shrivastava. [Proceedings of the IEEE 2021]

Sparsity in NN: Sources and Structures

Diverse Sparsity Ranges/Patterns Must be Exploited Well

- **Computer Vision Models:** ~40%, 70%-80% in later layers of deep CNNs
  - Weight sparsity: 50% (MobileNetV2) – 93% (EfficientNets)
    - 80% - 85% in Point-wise Convolutions
    - Can be structured (dimension/block pruning, fixed density in a block)
  - Activation sparsity is low (~20% and can be) - unstructured
- **Language Models**: ~80% -- 93% (baseline Transformers, BERTs, etc.)
  - But, existing accelerators do not exploit 80%+ sparsity well.
  - Very coarse-grain in large language models (e.g., Switch transformers)
    - Easy to exploit; Pruning unimportant tokens and heads (Up to 75%)
- **RNNs:** Up to ~60% activation sparsity; ~80% weight sparsity
  - Usually unstructured, but can be structured with pruning or with factorized operators (batch norm, quantization, activation function)
- Drop-out Layers
- Atrous (dilated convolutions): fine-grained structured in Weights
- **GANs:** ~60% in Activations, in transposed CONV in degenerators.
- **3D Point Clouds:** Up to 80% or more in Activations, Unstructured.
- **Gradient Sparsity in Communication:** >> 95% Unstructured
  - More than 99% for computer vision or language tasks
  - 95% -- 99% for recommendation models
  - Challenging to exploit – both storage-wise and compute-wise
- **Graph Learning:** High (75%--99%) or Hyper (99%+) Unstructured
  - Local features lead to sparse adjacency matrices and sparsity propagates (sparse/dense, dense/dense multiplications)
  - Dense/Sparse compute processed with separate accelerator modules
- **Text Analysis:** ~67% -- 99% Unstructured; Weights (and Activations)



(a) Unstructured (b) Block sparse (Coarse-grain) *block size: 2x2* (c) Density-bounded (k:n) block sparse (fine-grain) *block size: 1x4; k=1* (d) Conditional or Patterned

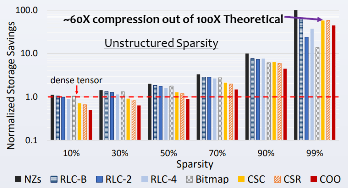
- **Accelerating sparsity is important for many other domains**
  - linear algebra, graph processing, scientific computing, database, genomics, compression (Usually unstructured)

Need Special Hardware or Software Mechanisms

- **<u>Sparsity cannot be leveraged as it is, including on DNN accelerators</u>**
- Need special mechanisms (Even for structured sparsity) for
  - **Encode:** Store only non-zeros with their locations (Encode)
  - **Decode:** Get non-zero values from on/off-chip memory or storage.
  - **Extract:** Find matching non-zeros from two tensors to multiply/add
  - **Load Balance:** Ensure each computing unit has similar work
  - **Communicate:** Both non-zero and its position
  - Do All of Above **Without Much Overhead** (Power, Performance, Area)

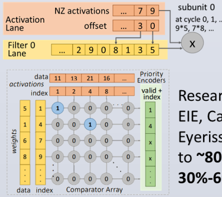
Effectiveness of Special Mechanisms

Works Great for Limited Range in~30%-80%



- **Storage Benefits** With Structured Sparsity:
  - **Same as unstructured,** for density-bounded blocks (2:4 in A100/H100)
  - Benefits **closer to ideal as coarse-grain pruning is higher** (e.g., dimensions)

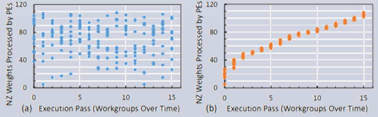
- **Non-Zero Matching** for scalar/vectors:
  - **Lookup:** Use position of a non-zero from first tensor to extract non-zero from second tensor
  - **Intersection:** Compare streams of positions of non-zeros from two tensors
  - **Size of a buffer used for lookup/comparison determines sparsity that can be exploited**
  - 16 non-zeros in 256 positions to feed 16-input vector unit exploits up to 1-1/16 ~93.75% sparsity

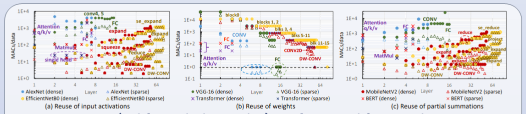


In A100/H100, only 2:4 = 50% sparsity allowed. None in TPU.

Research accelerators like EIE, Cambricon-X/S, EyerissV2 are efficient up to **~80%-85% W-sparsity, 30%-60% IA-sparsity.**

- Structured Sparsity Helps in **Load Balancing:**
  - **Without Balancing, Work among Leading and Trailing Compute Units Differ Notably**(~6.5X)
  - Without balancing the work, idle cycles for computational units can be 40%-50%.

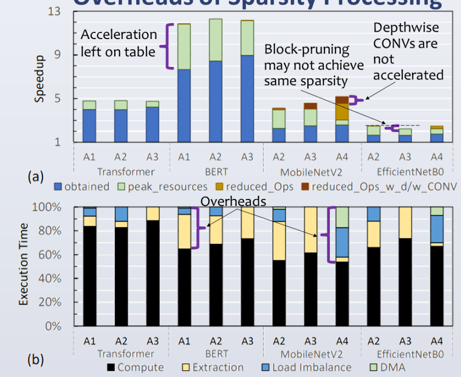




(a) Reuse of input activations (b) Reuse of weights (c) Reuse of partial summations

- **Data Reuse (arithmetic intensity) Reduces With Sparsity**
  - So are opportunities for less on-chip communication (spatial reuse), low memory accesses (temporal reuse)
  - Reuse in CONV2D and batched GEMMs/MLPs still considerable

Overheads of Sparsity Processing



- **Energy reductions relatively quite easier to** attain – lowering memory accesses is the dominating factor.
- **Latency reductions are efficient up to only ~80% sparsity**
  - **Bottlenecks:** Non-zero matching or imbalance–even when weights pruned in structure in tandem with unstructured activations
  - ~8X speedup out of 12X achievable for 92% sparse BERT
  - ~50% of achievable speedup for compact vision models (MobileNets)
  - ~4X lower speedup attained vs. achievable at hyper sparsity
  - Imbalance makes infeasible to feed large vector/SIMD units
  - Hard to achieve similar sparsity with block-pruning (EfficientNets)
- ***Most accelerators poorly process depthwise CONV -- slowdowns***

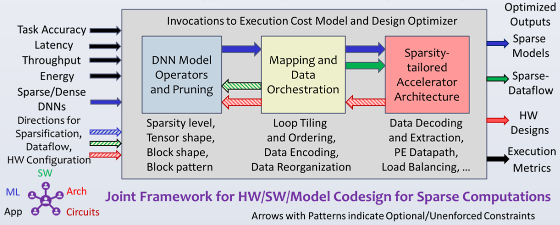
Designing All-Sparsity Exploiting Accelerator

- **<u>Reconfigurable Mechanisms Needed for Flexibility</u>**
  - Sparsity-tailored multiple encodings and non-zero matchings
  - Communication interconnects for multicast, unicast of data and spatial/temporal accumulations
  - Unified Memory for data and metadata
  - Configurable workgroup formulation and asynchronous processing
- Novel mechanisms for high & hyper sparsity (>90%, especially 99%+)

Algorithmic Sparsity Can Be Made System-Aware

- **Apply quantization and sparsity (pruning, operator reformation) <u>based on hardware's capability</u>**
- **<u>Leverage execution models of the system</u> – find out what advantage a sparsity or quantization can offer on your target hardware/compiler**
  - **A model with 80% uniform sparsity across layers may perform better than a model containing layers with 70% and 90% sparsity, with same accuracy.**
- **Maximize improvements jointly with quantization, pruning, and value similarity (<u>interplay on compression, acceleration exploited, and accuracy</u>)**
- **Cross-layer Codesign**

<u>Quantitative Approach for Cross-layer Design</u>



Task Accuracy
Latency
Throughput
Energy
Sparse/Dense DNNs
Directions for Sparsification, Dataflow, HW Configuration
Invocations to Execution Cost Model and Design Optimizer
DNN Model Operators and Pruning
Mapping and Data Orchestration
Sparsity-tailored Accelerator Architecture
Sparsity level, Tensor shape, Block shape, Block pattern
Loop Tiling and Ordering, Data Encoding, Data Reorganization
Data Decoding and Extraction, PE Datapath, Load Balancing, ...
Optimized Outputs
Sparse Models
Sparse-Dataflow
HW Designs
Execution Metrics
SW
ML
Arch
App
Circuits

Joint Framework for HW/SW/Model Codesign for Sparse Computations

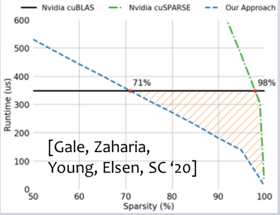
Arrows with Patterns indicate Optional/Unenforced Constraints

AutoML for Model Sparsity and Compression

- For expert-directed or automated search of best compression or codesign, **<u>specify all common hyperparameters for applicability</u>**
  - A variety of pruning options (unstructured, 1D or k:n block-sparsity, bit-widths of tensors/layer, tolerable accuracy, goals: storage/energy/performance)
- **<u>Automated optimization for hyperparameters for generality and efficiency</u>**
  - Pruning ratio for each iteration (epoch); pruning mechanism (which values to prune, e.g., below a certain threshold); pruning pattern (fine-grain, block size); bit-widths of tensors (quantization).
  - This does not need to be manual or explored from a limited pre-designed set.

Tensor Core Accelerators Exploit Sparsity Better

- Sparsity below 99% usually do not lead to practical speedups on CPUs/GPUs without tensor cores
  - Recent algorithmic & software advances strive to reduce gap
- Tensor-cores-based accelerators can provide much higher performance/watt (TPUs, A100)
- **Trends and Roadmap: Flexible hardware/compiler and Cross-layer Codesign for Sparsity**



[Gale, Zaharia, Young, Elsen, SC '20]

sites.google.com/view/shail

Research supported by NSF/Intel Joint CAPA Research Center. Joint work with Prof. Riyadh Baghdadi (MIT CSAIL and NYU), Prof. Tony Nowatzki (UCLA), Dr. Sasikanth Avancha (Intel Labs), Prof. Baoxin Li (ASU).