
**Sparsity in NN: Sources and Structures**

- **Diverse Sparsity Ranges/Patterns Must be Exploited Well**
  - **Computer Vision Models:** ~60%, 70%~80% in later layers of deep CNNs
  - Weight sparsity: >90% (MobileNetV2) - >95% (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%–95% (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 40% 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
  - **GATs:** ~60% in Activations, in transposed CONV in de-encoders.
  - **3D Point Clouds:** Up to 85% or more in Activations, Unstructured.
  - **Gradient Sparsity in Communication:** >> 90% Unstructured
  - More than 95% for computer vision or language tasks
  - 95% – 99% for recommendation models
  - Challenging to exploit – both storage-wise and compute-wise
  - **Graph Learning:** High (75%–90%) or Hyper (99%) Unstructured
  - Steps to sparse adjacency matrices and sparsity propagates (sparse/dense, block/dense multiplications)
  - Dense/Sparse compute processed with separate accelerator modules
  - **Text Analysis:** ~60%–90% Unstructured: Weights (and Activations)
   - A 100x/K100, only 24% = 50% sparsity allowed.
   - None in TPU.
  - **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**

- **Sparsity cannot be leveraged as it is.** Including on DNN accelerators
  - Need special mechanisms (Even for structured sparsity) for
    - Encode: Store only non-zero with their locations (Encode)
    - Decode: Get non-zero values from off-chip memory or storage.
    - Extract: Find matching non-zero 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%

**Algorithmic Sparsity Can Be Made System-Aware**

- **Apply quantization and sparsity (pruning, operator reformation) based on hardware’s capability**
- Leverage execution models of the system – find 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 (Interplay on compression, acceleration exploited, and accuracy)

**AutoML for Model Sparsity and Compression**

- For expert-directed or automated search of best compression or codex, specify all common hyperparameters for applicability
  - A variety of pruning options (unstructured, 1D or kn block-sparsity, bit-widths of tensors/layer, tolerable accuracy, goals: storage/energy/performance)
- **Automated optimization for hyperparameters for generalization and efficiency**
  - pruning ratio for each iteration (epoch); pruning mechanism (which value 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 5% usually does not lead to practical speedups on CPUs/CUDA without tensor cores
  - Recent algorithmic & software advances strive to reduce gap
  - Tensor-cores based accelerators can provide much higher performance/efficiency (TPUs, A100)
- **Trends and Roadmap:**
  - Flexible hardware/compiler and Cross-layer Design for Sparsity
  - New mechanisms for block & hyper-sparsity (90%, especially 99%)
  - Configurable workgroup formulation and asynchronous processing

---

**Sparse-Optimized Accelerator Architecture**

**Optimized Outputs**

- Sparse-Optimized Accelerator Architecture

**AutoML Framework for HW/SW/Coding Sparsify for Sparse Computations**

- Joint Framework for HW/SW/Model Codex for Sparse Computations

- **AutoML Parameters/Tradeoffs**
  - Optimized Outputs
  - Sparse-Optimized Accelerator Architecture
  - AutoML Framework for HW/SW/Coding

---

**References**

- Gale, Zaharia, Young, Elsen, SC’20
- Prof. Byadgadhi (MIT CSAIL and NYU), Prof. Tony Nowatzki (UCLA), Dr. Saksirath Ananda (Intel Labs), Prof. Baoxin Li (ASU).