Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network

Song Han[1,3], Xingyu Liu[4], Huizi Mao[3], Jing Pu[5], Ardavan Pedram[2,6], Mark A. Horowitz[2], William J. Dally[2,3],
[1]MIT [2]Stanford [3] NVIDIA [4] CMU [5] Google [6] Samsung

***Abstract*—EIE proposed to accelerate pruned and compressed neural networks, exploiting weight sparsity, activation sparsity, and 4-bit weight-sharing in neural network accelerators. Since published in ISCA'16, it opened a new design space to accelerate pruned and sparse neural networks and spawned many algorithm-hardware co-designs for model compression and acceleration, both in academia and commercial AI chips. In retrospect, we review the background of this project, summarize the pros and cons, and discuss new opportunities where pruning, sparsity, and low-precision can accelerate emerging deep learning workloads.**

I. What we did well

We started this project as deep learning accelerators are bottlenecked by the memory footprint. Computation is cheap and memory is expensive. Existing algorithm and hardware stack accelerate the inference of a neural network "as is." We asked, can we compress the model first? and we developed the "Deep Compression" [1], [2] technique that can compress the weights of a neural network by an order of magnitude by pruning and quantization. Since pruned weights become zero, and zero multiplied by anything is still zero, we can potentially save the computation and memory. However, the resulting neural network is sparse and irregular, which conflicts with massively parallel computing, and runs inefficiently on general-purpose hardware.

EIE demonstrated that special-purpose hardware can make it cost-effective to do sparse operations with matrices that are upto 50% dense - while in software, density must be much less than 1% to overcome the overhead of the sparse package.

EIE exploits both weight sparsity and activation sparsity. It stores the weights in compressed sparse column format, parallelizes the computation by interleaving matrix rows over the processing elements, and detects the leading non-zero in activations. It not only saves energy by skipping zero weights but also saves the cycle by not computing it. EIE supports fine-grained sparsity, and allows pruning to achieve a higher pruning ratio.

EIE adopted aggressive weight quantization (4bit) to save memory footprint. To maintain accuracy, EIE decodes the weight to 16bit and uses 16bit arithmetic. This W4A16 approach (4-bit weight, 16-bit activation) is different from the conventional W8A8 approach. Such a design has been reborn in large language models (LLM). The single batch text generation of these models is dominated by matrix-vector multiplication — same as EIE. It is memory-bounded, and the weight memory is the bottleneck, not the activation — 4bit weight and 16bit activation become attractive to save memory and maintain accuracy at the same time, as adopted by many software LLM inference engines.[1] However, these software solutions use linear integer weights, rather than a Kmeans codebook to make the weight decoding simpler and the arithmetic cheaper.

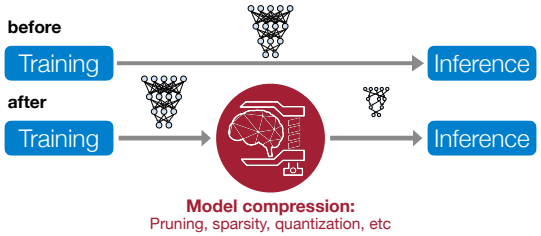

Fig. 1. EIE opened a new opportunity to build hardware accelerator for sparse and compressed neural networks.

EIE demonstrates the opportunity for accelerator and neural network co-design. There's plenty of room at the top to compress the neural network before accelerating it (Figure 1). Deep Compression and EIE show the benefit of refactoring the design stack.

II. Later work

EIE generated a new wave of AI accelerator design by opening a new dimension: sparsity. Cambricon-X [3] proposes a prefix-sum-based indexing module and supports sparse CNNs. SCNN [4] utilizes outer product and scatter-add to process sparse CNN while maximizing the input data reuse. Pragmatic [5] skips bit-level zeros and eliminates ineffectual computations. UCNN [6] generalizes the sparsity problem to the repetition of weights with any value instead of zero. Eyeriss V2 [7] proposes a flexible interconnect and PE architecture to accelerate sparse CNN. ExTensor [8] hierarchically eliminates the computation in sparse tensor computations using an efficient intersection architecture. SIGMA [9] proposes flexible interconnect to perform the distribution/reduction of sparse data for DNN training. The Sparse Abstract Machine [10] targets sparse tensor algebra to reconfigurable and fixed-function spatial dataflow accelerators.

EIE had substantial impacts on commercial AI chip design, leveraging pruning and sparsity for higher efficiency. NVDLA [11] gates the pruned weights to save energy. NVIDIA Sparse Tensor Core [12] adopt structured 2:4 sparsity to speed up pruned models. Samsung NPU [13] uses a priority-based search algorithm to skip zeros in activations. Ambarella CV22 [14] supports both structured and unstructured weight sparsity.

[1]4bit LLM projects such as: GPTQ, AWQ, llama.cpp, MLC LLM

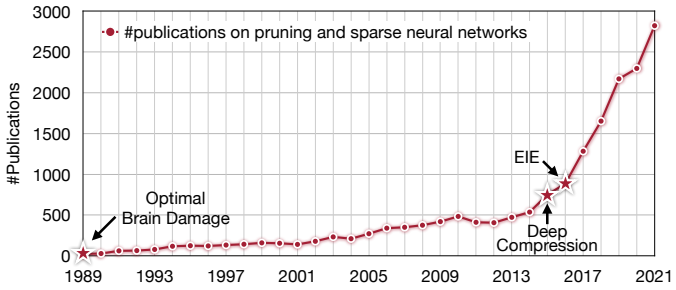

Fig. 2. The number of publications on neural network pruning and sparsity quickly increased since 2015, including both algorithms and systems. Source.

III. Lessons

Notwithstanding that EIE started sparse acceleration, this technique isn't as easily applied to arrays of vector processors. There are several improved designs that solved the issue, including Sparse Tensor Core [12], [15] that adopted structured sparsity (N:M sparsity), where one PE becomes more effective PEs in a regular manner. Another improvement is load-balance-aware pruning [16] to avoid PE starvation.

While EIE's special-purpose hardware is orders of magnitude more efficient than a software implementation of sparse M $\times$ V, the overhead of traversing the CSC structure is non-zero. One PE performs only one MAC, but is associated with many overhead structures, including pointer read, sparse matrix access, leading non-zero detector, etc. In EIE, the weight and index are both 4bit giving a 50% storage overhead. Other designs use structured sparsity or coarse-grained block sparsity to reduce storage and control overhead.

EIE only accelerates fully connected layers. Later, SCNN [4], Cambricon-X [3] and Eyeriss-V2 [7] can also accelerate sparse convolution layers. EIE stores all the weights in SRAM. Commercially, Cerebras tried this path to put everything in SRAM. This setting is perfect for vision models, but not easy for LLM: the number of parameters of recent LLMs ranges from 10 billion to 100 billion, making it difficult to fit SRAM.

IV. New opportunities

DNN architecture has witnessed rapid change. After EIE, we developed hardware-aware neural architecture search (NAS) techniques, ProxylessNAS [17] and Once-for-all [18] that design small and fast models before model compression.

The first principle of efficient AI computing is to be lazy: avoid redundant computation, quickly reject the work, or delay the work. We show a few more examples.

After compressing the weights, the activation becomes the bottleneck. Therefore, we developed the MCUNet family [19], [20] that aggressively shrinks the activation for TinyML. MCUNet performs not only ImageNet classification but also detection with only 256KB SRAM and 1MB Flash on a microcontroller. By sparse update and low precision, we can even do on-device training under 256KB memory [21].

Generative AI: spatial sparsity persists in image editing or image in-painting; users don't edit the whole image. So rather than generating the full image, sparsely generating where is edited [22] can speed up inference.

Transformer is a major neural architecture after EIE, and FC layer is back again. The attention layer has no weights to prune. However, not all tokens are useful: SpAtten [23] proposes cascade token-pruning and gradually removes redundant tokens with the smallest attention score. It exploits "progressive quantization" that lazily fetches MSBs only, run inference; if the confidence is low, it fetches LSBs.

Temporal sparsity exists in videos. Adjacent frames are similar. Rather than using expensive 3D convolution, temporal shift [24] can efficiently exploit temporal redundancy with zero FLOPs. Point cloud is spatially sparse. TorchSparse [25] adaptively groups sparse matrices to trade computation for regularity. PointAcc [26] employs a sorting array to perform sparse input-output mapping and avoid zero computation.

We envision future AI models will be sparse at various granularity and structures. Co-designed with specialized accelerators, sparse models will become more efficient and accessible.

Acknowledgements

We thank Zhekai Zhang and Yujun Lin for the discussions and collecting data for the figure.

References

[1] Han *et al.*, "Learning both weights and connections for efficient neural network," *NIPS*, 2015.
[2] Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *ICLR*, 2016.
[3] Zhang *et al.*, "Cambricon-x: An accelerator for sparse neural networks," *MICRO*, 2016.
[4] Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," *ISCA*, 2017.
[5] Albericio *et al.*, "Bit-pragmatic deep neural network computing," *MICRO*, 2017.
[6] Hegde *et al.*, "Ucnn: Exploiting computational reuse in deep neural networks via weight repetition," *ISCA*, 2018.
[7] Chen *et al.*, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *JETCAS*, 2019.
[8] Hegde *et al.*, "Extensor: an accelerator for sparse tensor algebra," *MICRO*, 2019.
[9] Qin *et al.*, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for DNN training," *HPCA*, 2020.
[10] Hsu *et al.*, "The sparse abstract machine," *ASPLOS*, 2023.
[11] Zhou *et al.*, "Research on nvidia deep learning accelerator," *ASID*, 2018.
[12] Mishra *et al.*, "Accelerating sparse deep neural networks," *arXiv*, 2021.
[13] Jang *et al.*, "Sparsity-aware and re-configurable npu architecture for samsung flagship mobile soc," *ISCA*, 2021.
[14] Ambarella, "CV22S Computer Vision SoC for IP Cameras," 2022.
[15] Zhu *et al.*, "Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern gpus," *MICRO*, 2019.
[16] Han *et al.*, "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," *FPGA*, 2017.
[17] Cai *et al.*, "ProxylessNAS: Direct neural architecture search on target task and hardware," *ICLR*, 2019.
[18] Cai *et al.*, "Once for all: Train one network and specialize it for efficient deployment," *ICLR*, 2020.
[19] Lin *et al.*, "MCUNet: tiny deep learning on IoT devices," *NeurIPS*, 2020.
[20] Lin *et al.*, "MCUNet-V2: Memory-efficient patch-based inference for tiny deep learning," *NeurIPS*, 2021.
[21] Lin *et al.*, "On-device training under 256kb memory," *NeurIPS*, 2022.
[22] Li *et al.*, "Efficient spatially sparse inference for conditional GANs and diffusion models," *NeurIPS*, 2022.
[23] Wang *et al.*, "SpAtten: Efficient sparse attention architecture with cascade token and head pruning," *HPCA*, 2021.
[24] Lin *et al.*, "TSM: Temporal shift module for efficient video understanding," *ICCV*, 2019.
[25] Tang *et al.*, "TorchSparse: Efficient point cloud inference engine," *MLSys*, 2022.
[26] Lin *et al.*, "PointAcc: Efficient point cloud accelerator," *MICRO*, 2021.