Sparsity-Aware Hardware: From Overheads to Performance Benefits

Shi, Man ; Kneip, A.; Chauvaux, N.; Sun, Jiacong ; Frenkel, C.; Verhelst, Marian

DOI
10.1109/MSSC.2025.3549709

Publication date
2025

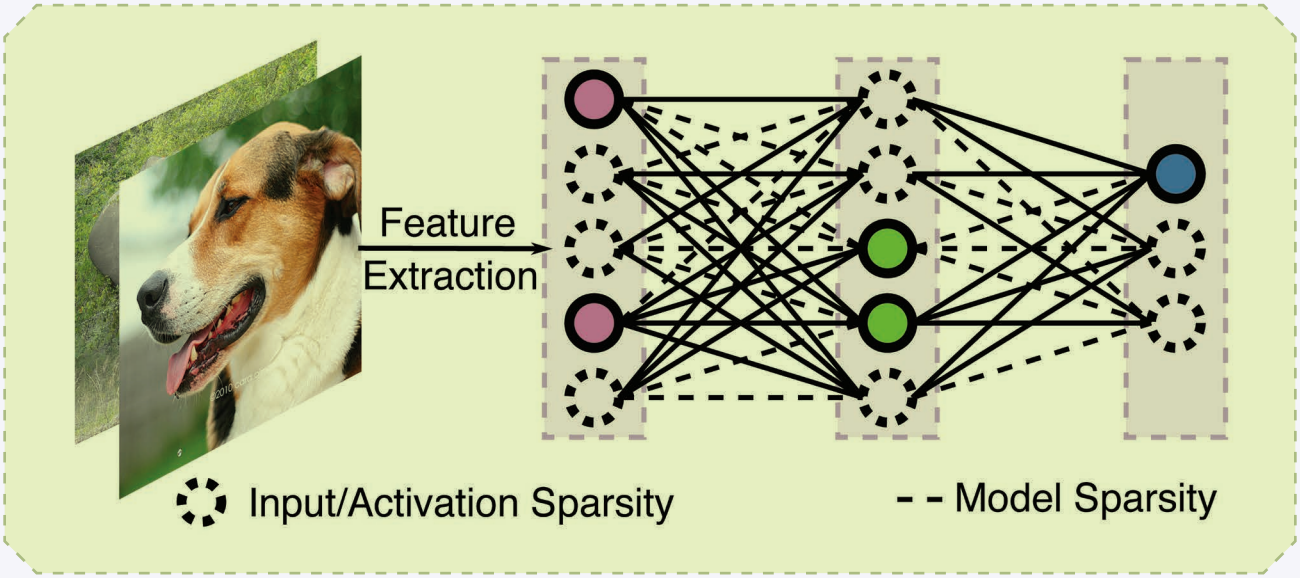
Document Version
Final published version

Published in
IEEE Solid-State Circuits Magazine

Man Shi, Adrian Kneip, Nicolas Chauvaux, Jiacong Sun, Charlotte Frenkel, and Marian Verhelst

Sparsity-Aware Hardware

From Overheads to Performance Benefits



OPENER ART CREDIT: IMAGENET

As artificial intelligence (AI) continues to transform multiple sectors, its exponential growth in computational demands presents significant challenges for hardware infrastructure. This article examines sparsity, the prevalence of zeros in AI workloads, as a promising approach to address these challenges. While sparsity offers potential efficiency gains, its practical implementation requires careful consideration of hardware constraints and computational overheads. Therefore, this article cooperates with a virtual performance roofline model to analyze various sparsity techniques and their associated tradeoffs, aiming to bridge the gap between theoretical potential and practical implementation in AI accelerator design.

Digital Object Identifier 10.1109/MSSC.2025.3549709
Date of current version: 20 June 2025

Introduction: Sparsity in AI Workloads

AI systems exert a transformative impact across numerous fields, reshaping our modern world [1], [2], [3], [4]. However, this rapid growth comes at a significant cost: The computational demands of AI infrastructure are doubling approximately every 100 days [5]. This skyrocketing trend poses performance concerns for the supporting AI computing hardware, which struggles to scale at a comparable pace. This challenge has prompted hardware architecture designers to seek solutions off the beaten path.

One particularly promising strategy leverages an intriguing characteristic of AI workloads: their sparsity [6]. Sparsity represents the fraction of operands in AI workloads that consist of zero values (whether weights or activations), and thus do not contribute any useful information to the output However, leveraging sparsity in AI accelerators is not trivial and requires a detailed understanding of the

This tight codesign process requires a good understanding of the various existing types of algorithmic sparsity that can be exploited at the hardware level.

underlying hardware constraints and computational overheads. To bring sparsity from a theoretical concept into a practical efficiency technique, the key lies in strategically balancing computational performance gains and the hardware overheads induced by sparsity handling.

In this article, we revisit the hardware roofline model to take sparsity into performance consideration, thereby sketching the landscape of sparsity techniques, their opportunities, as well as their related overheads.

Hardware Implications of Sparsity

Whether sparse or not, AI workloads, such as deep neural networks (DNNs) usually consist of simple yet numerous tensor operations, mainly matrix–vector or matrix–matrix multiplications (denoted MVMs and MMs, respectively), which dominate the workload's latency and energy per inference. These workloads are thus typically executed on specific tensor accelerators optimized for dense MMs/MVMs, with large computational parallelism, excellent data reuse, and a resulting 100 to 1,000× lower energy per operation than conventional CPUs or GPUs (without tensor cores) [7]. These hardware accelerators [8], [9] typically consist of three parts [Figure 1(a)]: a *datapath* formed by an array of processing elements (PEs) that compute parallel multiply-accumulate (MAC) operations to yield the MM/MVM result, a local *memory* connected to the datapath through a high-bandwidth interface to store the DNN's weights and input/output activations, and a *controller* orchestrating the accelerator's dataflow.

To minimize the latency and energy footprint of DNN inference, one aims to utilize 100% of the accelerator's computing resources at all times. Under such peak utilization, each PE performs a MAC operation at each clock cycle, and the memory bandwidth utilization matches the processing rate of the datapath. These conditions leverage the peak computational efficiency of the accelerator, commonly described by two performance metrics: its throughput (GOPS) and energy efficiency (TOPS/s/W). Nonetheless, the actual performance of the accelerator is in practice affected by the arithmetic intensity of the DNN workload, as described by its *roofline model* (see "Visualizing Computational Performance: The Virtual Roofline Model").

Yet, dense DNN accelerator cores are optimized for dense DNN workloads: They process MACs regardless of the presence of zero operands. These zero-valued MACs do not contribute to the final result, and therefore result in wasted computing time and energy. As such, sparsity effectively leads to an underutilization of either the datapath, memory bandwidth,

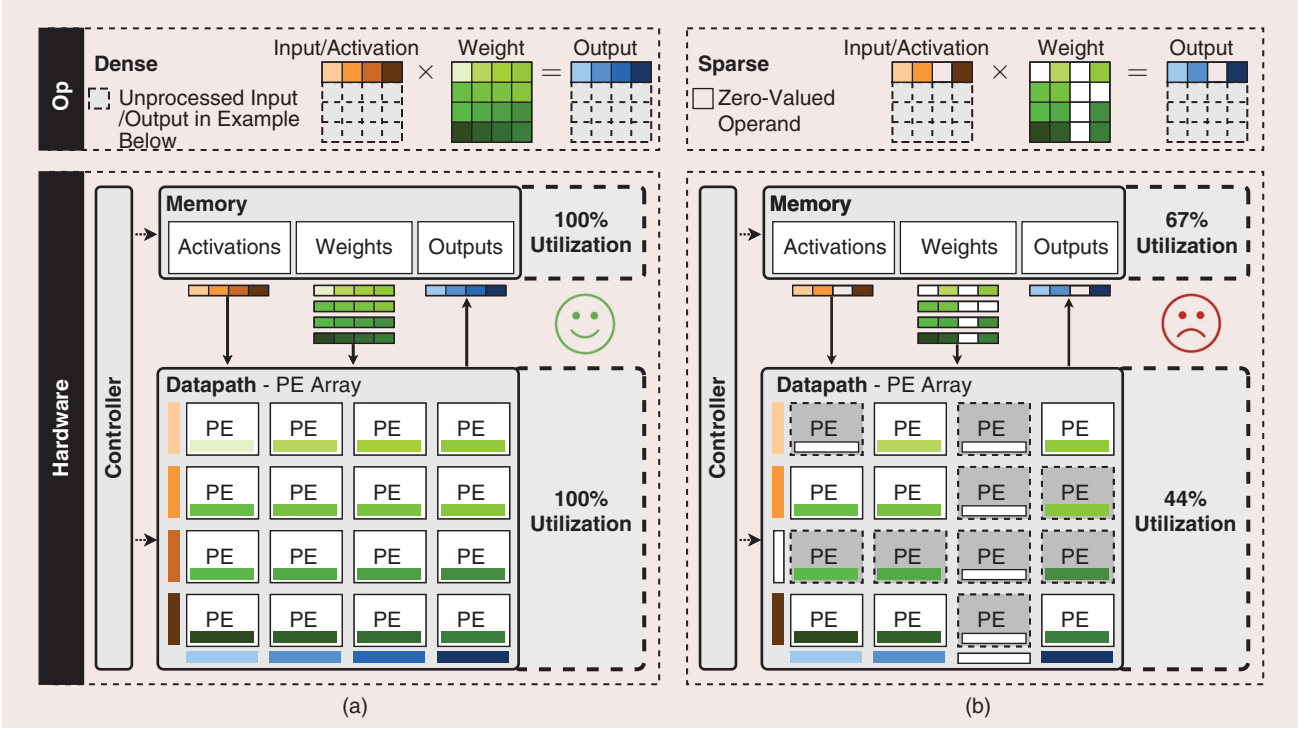


FIGURE 1: Hardware architecture of a dense DNN accelerator, running (a) a dense and (b) a sparse matrix-matrix multiplication, typically consisting of several MAC operations. Reported hardware utilization is illustrative.

and/or memory content, as shown in Figure 1(b). To overcome these, sparsity-aware techniques have been investigated to account for sparse operands directly at the hardware level. Their impact on performance can be captured in a virtual roofline model (see "Visualizing Computational Performance: The Virtual Roofline Model") reflected by a change in the roofline boundary as a function of the density level (= 1 – sparsity). At low sparsity, sparsity-aware accelerator cores generally suffer from a drop in virtual peak throughput compared to dense cores due to the introduced hardware overheads to handle sparsity. As the sparsity level rises, benefits from sparsity increase the virtual throughput: The gap between dense and sparsity-aware performance narrows down, eventually disappearing at the sparsity turning point, beyond which virtual throughput gains start to be leveraged. Yet, doing so requires matching the supported sparsity technique(s) with the nature of the sparsity present in the workload at hand. This tight codesign process requires a good understanding of the various existing types of algorithmic sparsity that can be exploited at the hardware level.

The Many Faces of Sparsity

Sparsity in DNNs can be classified into different categories (Figure 2) based on the organization of zeros in operands (i.e., whether structured or not), as well as on the underlying granularity of the zero elements (i.e., from individual bits to full values). While sparsity can be inherently present in DNN operands, it often needs to be induced in the software pipeline to reach significant levels [10]. It usually relies on adjusting or retraining a model prior to deployment [11], or on setting activations to zero based on known statistics [12]. This often leads to an accuracy penalty, which will typically be more severe for structured than for unstructured sparsity [13].

Along the first categorization axis, *unstructured sparsity* corresponds to the absence of clear patterns in an operand's sparsity, with the zero-valued elements irregularly scattered across a tensor. Unstructured sparsity inherently arises at the bit level in binary data representations. Yet, it can also occur at the value level as a result of weight pruning or regularization

VISUALIZING COMPUTATIONAL PERFORMANCE: THE VIRTUAL ROOFLINE MODEL

The virtual roofline model is derived from the traditional roofline model, commonly used to benchmark the performance landscape of CPU or GPU architectures [S1], [S2], [S3]. It allows assessing the maximum achievable throughput of a processor with a given memory bandwidth and peak performance, as a function of a workload's arithmetic intensity. The *virtual* roofline extends it to include the first-order dependence of a core's performance on the sparsity-aware characteristics (Figure S1):

- The *arithmetic intensity* ❶ is the workload-dependent number of (zero and nonzero) operations performed per virtually transferred (zero and nonzero) byte of data (OPs/byte).
- The *virtual throughput* ❷ is the ratio between the total number of (zero and nonzero) operations and their total effective execution time (GOPS).
- In the *memory-bound regime* ❸ at low arithmetic intensity, *peak throughput* is limited by the *virtual memory bandwidth* ❹. In the *compute-bound regime* ❺ at high arithmetic intensity, it is limited by the datapath's parallelism. The resulting roofline curve corresponds to performance with a 100% utilization of the core's resources. The region above it is unreachable, while that below means under-utilization.
- While dense-optimized processors have a fixed roofline (black line in Figure S1), that of sparsity-aware processors changes with the sparsity level. At 0% sparsity (i.e. 100% density), the roofline drops below the dense processor performance (red area in Figure S1) due to hardware overheads for handling sparsity. At high sparsity levels, gains from virtual memory and compute throughput increases kick in (green area in Figure S1).
- The *sparsity turning point* ❻ is the sparsity level at which hardware overheads brought by sparsity-aware hardware techniques are overcome by virtual throughput gains. The memory- and compute-bound regimes can have different turning points depending on the supported sparsity-aware hardware techniques.

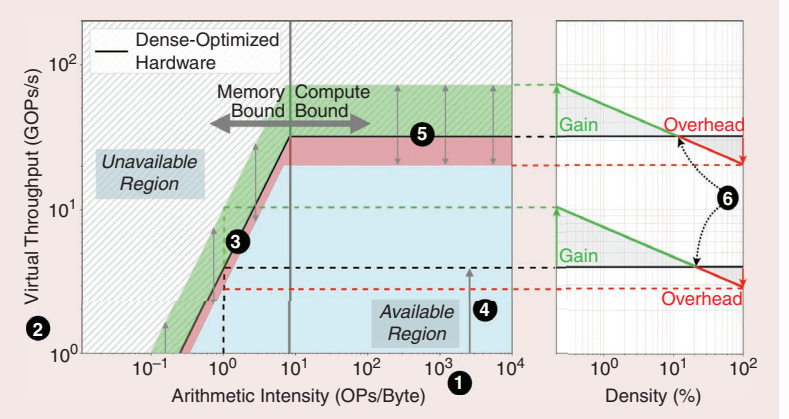


FIGURE S1: (a) The virtual roofline model and (b) corresponding tradeoffs in the memory- and compute-bound regimes as a function of density (= 1 – sparsity).

References

[S1] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009, doi: 10.1145/1498765.1498785.

[S2] G. Ofenbeck, R. Steinmann, V. Caparros, D. G. Spampinato, and M. Püschel, "Applying the roofline model," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2014, pp. 76–85, doi: 10.1109/ISPASS.2014.6844463.

[S3] N. Ding and S. Williams, "An instruction roofline model for GPUs," in *Proc. IEEE/ACM Perform. Model., Benchmarking Simul. High Perform. Comput. Syst. (PMBS)*, 2019, pp. 7–18, doi: 10.1109/PMBS49563.2019.00007.

> ***Altogether, the structured nature of these sparsity types results in a much higher regularity and determinism in the processing of the sparse workload.***

techniques during training [14], or due to thresholding functions during inference which zero out operands over ranges of input values [e.g., the conventional rectified linear unit activation function] [15]. However, the irregularity of the zero element locations entails fine-grained hardware and control overheads.

In contrast to unstructured sparsity, *structured sparsity* is characterized by a clear pattern of the zero elements within operand tensors. These patterns can also appear at a bit-level granularity, or at a value-level one. Moreover, at both granularities, the structure can present itself in different ways. First, in block sparse tensors, zeros are clustered together into subtiles of the tensor. At the bit level granularity, zero bits are packed together within one word or even across adjacent words [10]. At value-level granularity, such a tile can be a complete tensor row, column, 2D or 3D subset of the tensor [16], [17]. Alternatively, zero-valued operands can be structured based on *density bounding*, in which case every block of M tensor elements can have at most N nonzero elements, also denoted as N:M sparsity [18]. This form of structured sparsity is also possible at the value or at the bit level, as done in bit-balance [19], where the latter limits the number of "1" bits in each element. Quantization [20], [21] represents a specialized form of structured bit-level sparsity. Unlike previously discussed approaches that permit distributed nonzero bits, quantization typically mandates consecutive bit placement. This design choice eliminates index storage overhead and simplifies hardware implementation, although it imposes a more severe tradeoff between training effort and bit reduction while maintaining model accuracy [10], [20].

Altogether, the structured nature of these sparsity types results in a much higher regularity and determinism in the processing of the sparse workload. This in turn simplifies encoding/decoding schemes, run-time scheduling, or workload balancing complexity. The benefits this brings in terms of reducing hardware overheads can, however, be overshadowed by the inferior tradeoff between sparsity levels and task accuracy compared to unstructured sparsity.

Exploiting Unstructured Sparsity in Hardware: High Cost, High Reward?

To harness the potential benefits of sparsity, various accelerators and techniques have been developed in recent years, with some examples provided in Table 1. Among these techniques, those exploiting unstructured sparsity were the earliest to emerge [11], [24]. Unstructured sparsity can be leveraged in two distinct ways: From the memory perspective, it offers compression opportunities, while from the datapath perspective, it allows gating or skipping zero-valued operations, as visualized in Figure 3.

Compression techniques, on the one hand, are commonly used to compact sparse activations/weights into a dense representation, primarily aiming for a reduced memory footprint. Yet, these methods are typically penalized with an index overhead associated with indicating nonzero elements and ensuring accurate data decoding during the computational processes. The critical challenge lies in identifying an optimal sparsity turning point that strategically balances compression efficiency with indexing overhead, as discussed in [25]. Consequently, researchers have proposed diverse compression techniques to optimize

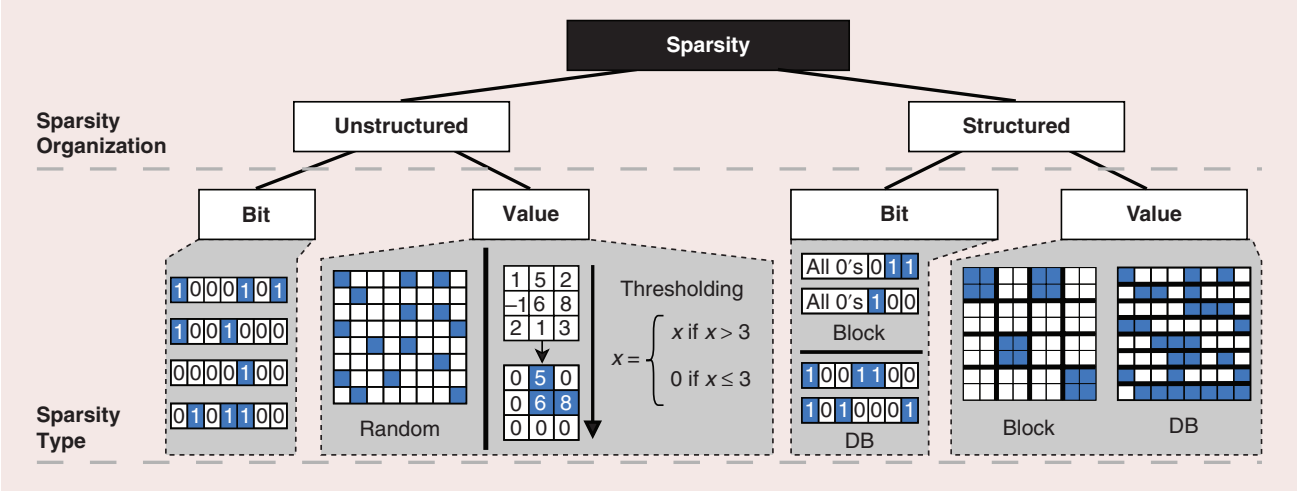


FIGURE 2: Classification of sparsity types appearing in DNN operands. DB: density bounded.

sparse data representations. Bit-masking (BM)-based compression [Figure 3(a)], for example, used in SparTen [26], employs "1" to indicate the locations of nonzero values within tensors, which has proven to be particularly advantageous for scenarios with lower sparsity (<30% sparsity [27]). In contrast, the compressed sparse row or column (CSR/CSC) representation emerges as a prominent technique for high-sparsity scenarios (>70% sparsity [27]), and is widely adopted in existing work [11], [23]. As illustrated in Figure 3(b), this method encodes nonzero element locations into coordinates, which are then stored consecutively in memory. In addition, recent advances [28], [29] leverage the abundant bit-level sparsity present in AI workloads to conduct data compression. For instance, as shown in Figure 3(c), the compact canonical signed digit is able to compress an N-bit data element into N/2 bits by encoding data in common 2's complement format with dedicated encoder/decoder support [28].

Sparsity-aware datapaths, on the other hand, typically involve MAC gating and/or skipping techniques to save computational energy and increase the throughput of the datapath. As illustrated in Figure 3(f), MAC gating introduces an additional signal and/or clock gating cell to each MAC, reducing computational energy consumption by freezing the inputs and disabling the MAC whenever a zero is detected. Envision [24] applies this gating technique at the value level for both activations and weights, leading to a standalone 1.6× energy saving for 30%–60% sparsity. Such gating techniques bring energy benefits and only come with small hardware overheads, yet they fail to provide any improvement in system throughput.

Alternatively, the sparsity-driven MAC skipping techniques skip the computation whenever a zero is detected at the MAC inputs [34]. As shown in Figure 3(g), online value-level activation-sparsity detectors are introduced to squeeze out zero activations, hence streaming only nonzero activations into the PE array and increasing the temporal utilization. Similarly, the technique can also be applied at the bit level, by skipping the ineffective bit-level operations to enhance the datapath efficiency in a more fine-grained manner. Bit-pragmatic [35] skips the zero-bit–associated computation by only processing nonzero activation bits in a bit-serial manner, as shown in Figure 3(h). In contrast to MAC gating, MAC skipping brings both energy savings as well as throughput benefits, making it an attractive target. However, a naive implementation of MAC skipping entails a risk for compute resources underutilization, due to uneven workload allocations to the different MACs, as shown in Figure 3 (g) and (h). State-of-the-art hardware implementations further incorporate advanced runtime load-balancing techniques to improve utilization, yet at the expense of increased complexity and hardware overhead. These additional hardware costs can be attributed to control logic overhead to dynamically parse the sparse data elements, manage nonzero value routing, and balance computational resources online to maintain efficient processing. Practical implementations underline the significant challenges: Both sparse convolutional neural networks (SCNN) [23] and SIGMA [27] dedicated more than 30% of their hardware area to their online data scheduler. These routing and compilation overheads can potentially negate the intended performance benefits, worsening the sparsity turning point for such accelerators.

Adapting the formalism of the virtual roofline model (see "Visualizing Computational Performance: The Virtual Roofline Model") to the above hardware techniques allows estimating this overhead-performance tradeoff. The virtual roofline of Figure 4(a) illustrates the typical sparsity turning point of accelerator cores concurrently supporting CSR compression and operand skipping in the presence of unstructured activation sparsity.[1] Such sparsity is notably exploited by spatial approaches, which leverage activation sparsity within individual tensor frames. In the compute-bound regime, initial overheads arise from the complex cycle-skipping control circuitry and the distribution of nonsparse operands across PEs, resulting in a turning point around a density of 60% for cycle skipping to become effective [23]. In the memory-bound regime, the virtual bandwidth utilization is originally degraded by the additional transfer of the compression metadata (i.e., encoding indices) compared to the dense case. Figure 4(a) highlights that in the typical range of activation

[1]The rooflines are drawn by assuming 128 parallel PEs in the datapath running at a 125-MHz clock, with a 4-Gb/s maximum memory bandwidth (1 MAC = 2 OPs) for each accelerator core.

TABLE 1. STATE-OF-THE-ART SPARSITY-AWARE AI ACCELERATORS.

ARCHI.	SCNN [23]	STICKER [11]	S2TA [30]	E-LLM [31]	BITLET [32]	BSAA [28]	SIBIA [33]	BitWAVE [10]
SPARSITY TYPE	UNSTRUCTURED VALUE SPARSITY		STRUCTURED VALUE SPARSITY		UNSTRUCTURED BIT SPARSITY		STRUCTURED BIT SPARSITY	
Gating	✗	✓	✗	✗	✗	✗	✗	✗
Skipping	✓	✓	✓	✓	✓	✓	✓	✓
Compression	✓	✓	✓	✓	✗	✓	✓	✓

BSAA: Bit-sparsity aware acceleration; LLM: large language model; SCNN: sparse convolutional neural networks.

sparsity harnessed in mainstream spatial DNN models, the dense-optimized core always outperforms its sparsity-aware counterpart under the given assumptions. Therefore, the turning point lies in the unavailable region of the sparsity-aware core. Altogether, the gains of unstructured sparsity seem limited, unless very high sparsity levels can be achieved. The graph neural networks (GNNs) can be a concrete example, as GNN sparse hardware still yields substantial gains due to extreme sparsity [36], [37] despite requiring complex memory controllers and schedulers to handle irregular connectivity patterns [38], [39].

Increasing input sparsity is also pursued by spatiotemporal approaches, which exploit activation

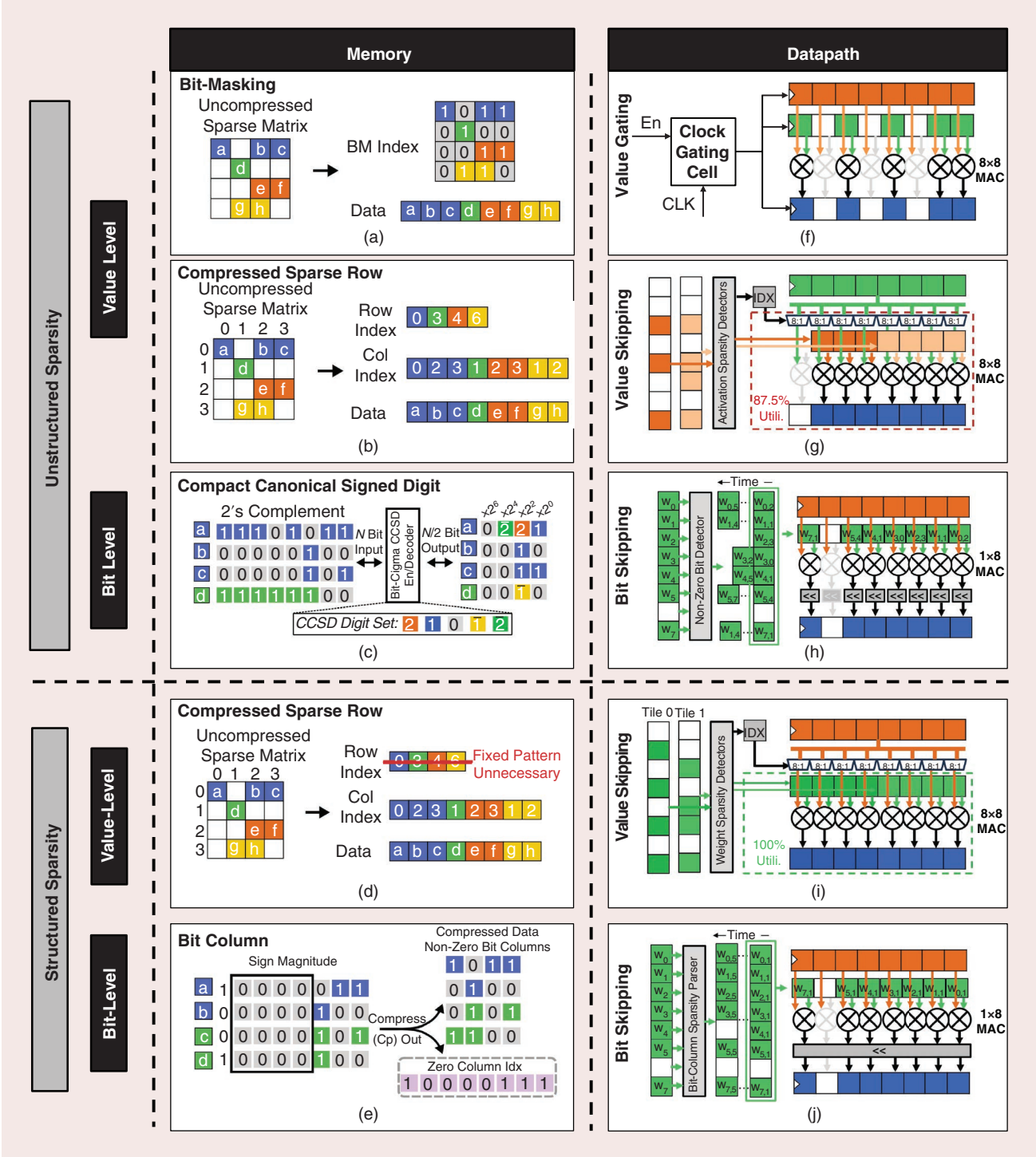


FIGURE 3: (a)–(j) Examples of hardware techniques suiting different sparsity types, from the memory and datapath perspectives. BM: bit masking.

sparsity along the time dimension. For example, deltaCNN and deltaRNN skip the processing of static elements between successive frames [41], [42], [43], enabling up to 98% unstructured activation sparsity. However, they both require computing the frame-to-frame difference on the fly, involving a latency penalty as well as a significant memory overhead to store the previous frame. Instead, event-based approaches can leverage the inherent sparsity of event cameras to bypass the latency bottleneck, processing events either in time-windowed batches using accumulated bin histograms [44], or in a fully event-driven fashion [22], [37], [45]. As the level of sparsity is purely dependent on the activity in the observed scene, extreme sparsity levels can be leveraged when moving from frame-based to pure event-driven processing. Nonetheless, leveraging temporal activation sparsity implies a memory overhead to store additional neuron states, needed to keep track of the evolution of information in time. These extra states also translate into increased data transfers that further pressure the available memory bandwidth.

The virtual roofline of such a spatiotemporal processing of unstructured sparse inputs and activations is compared to the previously discussed roofline in Figure 4(b). In the compute-bound regime, overheads from cycle-skipping are similar and the turning point likewise lies around a 60% density [23]. In contrast, in the memory-bound region, the increased data transfer overhead stemming from storing and communicating the neuron state information [22] brings the sparsity turning point down below the 10% density mark [Figure 5(b)]. As a result, the transition between the compute- and memory-bound regimes moves toward a higher arithmetic intensity. Despite this, the high sparsity levels unlocked by the spatiotemporal approaches outline order-of-magnitude throughput gains, especially when striving toward event-driven processing in the compute-bound regime.

The high sparsity levels unlocked by the spatiotemporal approaches outline order-of-magnitude throughput gains, especially when striving toward event-driven processing in the compute-bound regime.

Leveraging Predeployment Model Knowledge for Structured Sparsity Hardware

Previous discussions showed that, to exploit unstructured sparsity, a complex online scheduler or encoders/decoders are required to retain a high utilization of the PE array or memory bandwidth. Such burdens can be alleviated if the sparsity patterns are regular and known beforehand, which are exactly the properties of structured sparsity in a DNN model [46], [47]. For instance, NVIDIA's support for 2:4 density-bounded structured sparsity in the Ampere architecture offers a more regular workload mapping with fewer overhead losses compared to unstructured sparsity [47]. The impact on processor performance can again be assessed from a memory and from a datapath perspective.

From the memory side, when exploiting the value-level sparsity as shown in Figure 3(d), the indexing cost of the CSR format under structured sparsity can be reduced by removing the row (or tile) indices, and only indicating the nonzero positions within each tile [30]. The compressed data can moreover be organized in a regular pattern for efficient storage and access. Both of these are attributed to the known number of nonzero elements per tile. In contrast, as shown in Figure 3(e), the bit-level

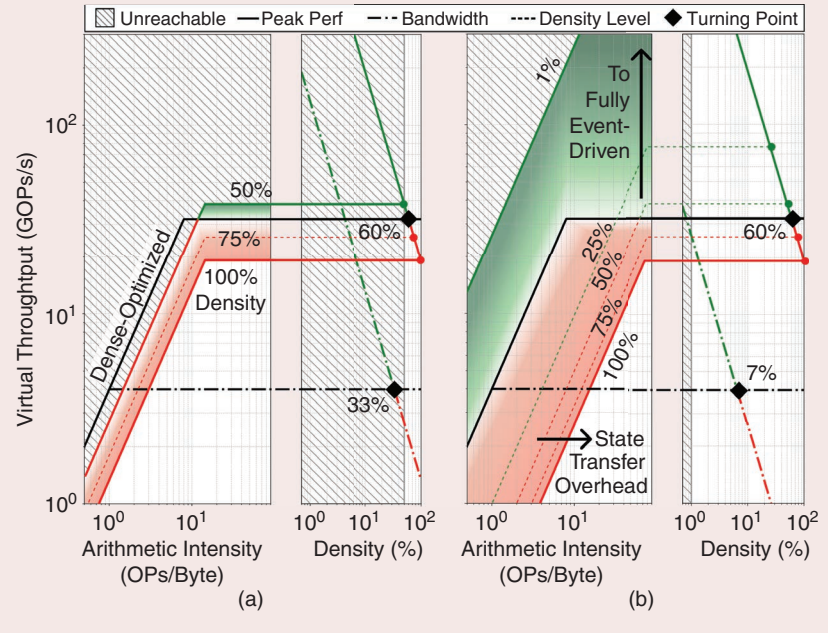


FIGURE 4: Virtual roofline representation of two different accelerator cores, respectively exploiting (a) spatial-only and (b) spatiotemporal unstructured activation sparsity. The 33% memory-bound turning point in (a) results from the compression of 8-b values into a CSR format, bringing a 200% overhead over a dense representation [18]. The 7% memory-bound turning point in (b) relates to the additional neuron state information [22]. The compute-bound turning points derived from hardware overheads in [23].

Nevertheless, the conclusion remains valid: At a fixed level of sparsity, DNN cores exploiting structured sparsity outperform those supporting unstructured sparsity for a same array size.

sparsity can be exploited by reorganizing weights into structured bit vectors that share zero-bit positions in each bit column [10].

From the datapath side, when applying the skipping technique on the value level, structured sparsity patterns allow for a full utilization of the MAC arrays with little control overhead, as the nonzero positions and MAC availability follow strict structured patterns. Specifically, comparing the skipping techniques for unstructured and structured value sparsity, as in Figure 3 (g) and (i), a 1-to-8 MUX is required for each MAC in both cases. Yet, in contrast to unstructured sparsity, structured sparsity can easily maintain high utilization by combining multiple processing tiles with fixed sparsity levels, without the need for intricate load-balancing techniques. Similarly, as shown in Figure 3(j), bit-level structured sparsity enables simpler hardware implementation than those for unstructured sparsity, as bit-column sparsity parsers can skip zero bits vector-wise.

The performance impact of the above techniques can again be reflected in the virtual throughput roofline, and compared to that of cores optimized for unstructured sparsity (Figure 5). The baseline shown in Figure 5(a) represents the roofline for unstructured weight sparsity processed on an accelerator optimized for this sparsity type. Compared to the unstructured activation sparsity roofline in Figure 4(a), the predeployment knowledge of the model's sparsity enables a lower compute overhead. Indeed, the load-balancing challenge in the compute-bound regime can be attenuated by extracting the sequence of weights to skip directly at compile time. Moreover, higher sparsity levels can be attained in some classical models through weight pruning techniques without incurring a significant accuracy degradation (e.g., down to a 10% density in VGG-16's or ResNet50's last layers [13]). Nonetheless, the initial overhead on virtual throughput remains significant, especially in the memory-bound regime.

Structure-optimized DNN processors leverage the better regularity of structured sparsity to mitigate the initial overhead of their unstructured counterpart, as shown in Figure 5(b). In the memory-bound regime, the use of *N:M* compression techniques pull the sparsity turning point significantly up at constant bandwidth. In the compute-bound regime, improvements are also observed at constant PE array size. Although lesser here, their absolute value depends on the underlying implementation of the zero-skipping and load-balancing techniques (Figure 3). Nevertheless, the conclusion remains valid: At a fixed level of sparsity, DNN cores exploiting structured sparsity outperform those supporting unstructured sparsity for a same array size. Moreover, the much lower design complexity of structured approaches allows the ability to drastically scale up the size of their MAC array without dramatically raising the hardware complexity. This makes structured-optimized processors especially appealing to handle low-to-moderate sparsity levels with very high parallelism [18], [30] . While unstructured-optimized processors remain a suitable choice to leverage high model sparsity in models and layers that show this property, one must realize that the high-throughput benefits of a few sparse layers can end up being masked by the significant overheads of denser layers. Altogether, these points suggest that structured-optimized processors make a more complete and reliable compromise.

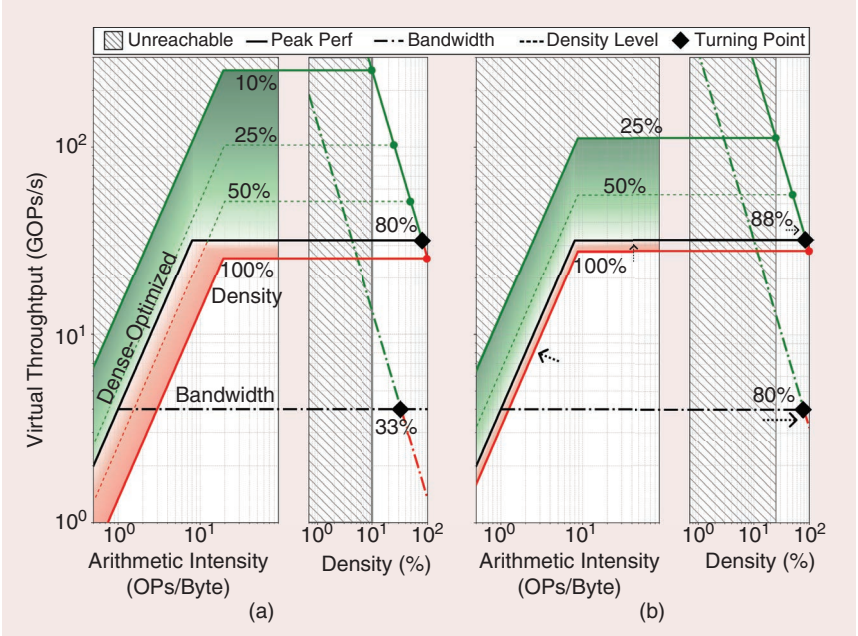


FIGURE 5: Virtual roofline representation of two different accelerator cores, respectively exploiting (a) unstructured spatial and (b) structured spatial model sparsity. The 33% memory-bound turning point in (a) is identical to that in Figure 4(a), as the same CSR compression is assumed. The 80% memory-bound turning point in (b) results from the 25% overhead of using a 2:4 sparse storage format over a dense representation [18]. The compute-bound turning points are derived from hardware overheads in [40].

WHAT ABOUT TRANSFORMER MODELS?

Hardware implications of sparsity in transformers:

- *Unstructured sparsity:* In contrast to DNNs, transformer-based models, among which large language models (LLMs), initially demonstrated minimal activation sparsity, primarily due to the adoption of Gaussian error linear units [S4] or Sigmoid-weighted linear units [S5] activation functions. Yet, recent research [12] has unveiled significant potential for sparsity by reintroducing rectified linear units in the transformers architectures. Remarkably, more than 90% activation values exhibit sparse characteristics in the feed-forward layer. As illustrated in Figure S2(a), once the activation (A) is zero, it can effectively skip the associated computation and transmission of the entire weight (W) chunk (in grey), directly translating to computational efficiency. Additionally, transformer-based LLMs showcase weight sparsity through unstructured parameter pruning techniques. Methods like Wanda [S6] employ sophisticated algorithms to estimate model weight importance, strategically removing noncritical weights to enhance random value-level sparsity up to 50% with minimal accuracy drop.
- *Structured sparsity:* Contemporary research extends beyond unstructured approaches, exploring structured sparsity to optimize transformers deployment efficiency [S7]. This involves implementing sparsity-aware training methodologies. Notably, the 2:4 density-bound sparsity can be introduced for LLMs as well [S8], achieving a 50% reduction in computational complexity while preserving model performance. Recent literature has also investigated structured bit-level sparsity to further minimize memory and datapath costs during transformers deployment. For instance, Anda [S9] leverages activation redundancy to bypass more than 50% ineffectual bit-block computations and memory footprints.

The exploration of unstructured/structured activation and weight sparsity in transformers, whether at the value or bit level, presents a promising avenue for optimization. These techniques can be synergistically combined with various hardware approaches, illustrated in Figure 3 and Table 1, to significantly enhance transformer performance and computational efficiency.

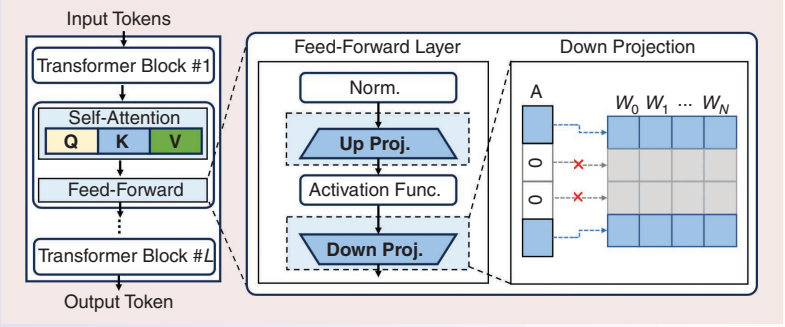


FIGURE S2: (a) Transformer-based architecture and (b) activation sparsity in down projection.

References

[S4] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[S5] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, Nov. 2018, doi: 10.1016/j.neunet.2017.12.012.

[S6] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," 2024, *arXiv:2306.11695*.

[S7] H. Zheng et al., "Learn to be efficient: Build structured sparsity in large language models," 2024, *arXiv:2402.06126*.

[S8] E. Kurtić, A. Marques, M. Kurtz, D. Alistarh, and S. Pandit, "2:4 sparse Llama: Smaller models for efficient GPU inference," *Neural Magic*, Nov. 25, 2024. [Online]. Available: https://neuralmagic.com/blog/24-sparse-llama-smaller-models-for-efficient-gpu-inference/

[S9] C. Fang, M. Shi, R. Geens, A. Symons, Z. Wang, and M. Verhelst, "Anda: Unlocking efficient LLM inference with a variable-length grouped activation data format," 2024, *arXiv:2411.15982*.

While the discussion in this article focused mainly on vanilla multilayer perceptrons and DNNs, the same classes of sparsity techniques can also be exploited in modern transformer-based AI workloads (see "What About Transformer Models?"), such as large language models.

Conclusion and Perspectives

Sparsity is a powerful and intrinsic characteristic of AI workloads that offers potential for enhancing computational efficiency through specialized hardware architectures. While the promise of performance gains is compelling, careful assessment is still crucial to understand the actual benefits under a given sparsity level and associated hardware overheads. The introduced virtual roofline model enables such a holistic view toward an effective hardware–software codesign. In addition, an emerging research direction focuses on sparsity-aware hardware modeling and design space exploration [48], [49], aiming to analyze the complex tradeoffs in more detail and develop new computational strategies. Going forward, we observe a growing trend of increasingly combining multiple types of structured and unstructured weight/activation sparsity within flexible architectures, aiming to maximize utilization in large MAC arrays to sustain the scaling laws of modern AI workloads.

Acknowledgment

This work was cofunded by the European Research Council under Grant Agreement 101088865, the Flanders AI Research Program, KU Leuven, Prophesee, and by the Dutch government as an High-Tech Systems and Materials-TKI project. Man Shi, Adrian Kneip, Nicolas Chauvaux, and Jiacong Sun are co-first authors, with equal contributions. Man Shi is the corresponding author.

References

[1] O. Hennigh et al., "NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework," in *Proc. Int. Conf. Comput. Sci.*, M. Paszynski, D. Kranzlmüller, V. V. Krzhizhanovskaya, J. J. Dongarra, and

P. M. Sloot, Eds., Cham, Switzerland: Springer-Verlag, 2021, pp. 447–461.
[2] Z. J. Baum, X. Yu, P. Y. Ayala, Y. Zhao, S. P. Watkins, and Q. Zhou, "Artificial intelligence in chemistry: Current trends and future directions," *J. Chem. Inf. Model.*, vol. 61, no. 7, pp. 3197–3212, 2021, doi: 10.1021/acs.jcim.1c00619.
[3] W. L. Neto, M. Austin, S. Temple, L. Amaru, X. Tang, and P.-E. Gaillardon, "LSOracle: A logic synthesis framework driven by artificial intelligence: Invited paper," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2019, pp. 1–6, doi: 10.1109/ICCAD45719.2019.8942145.
[4] D. Vrontis, M. Christofi, V. Pereira, S. Tarba, A. Makrides, and E. Trichina, "Artificial intelligence, robotics, advanced technologies and human resource management: A systematic review," in *Artificial Intelligence and International HRM*, A. Malik and P. Budhwar, Eds., London, U.K.: Routledge, 2023, pp. 172–201.
[5] M. Buchanan, "The laws of inflating the AI bubble," *Nature Phys.*, vol. 20, no. 9, Sep. 2024, Art. no. 1362, doi: 10.1038/s41567-024-02627-5.
[6] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *J. Mach. Learn. Res.*, vol. 22, no. 241, pp. 1–124, 2021.
[7] K. Guo et al., "Neural network accelerator comparison." [Online]. Available: https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator.html
[8] P. Houshmand et al., "DIANA: An end-to-end hybrid digital and ANAlog neural network SoC for the edge," *IEEE J. Solid-State Circuits*, vol. 58, no. 1, pp. 203–215, Jan. 2023, doi: 10.1109/JSSC.2022.3214064.
[9] M. Verhelst, M. Shi, and L. Mei, "Ml processors are going multi-core: A performance dream or a scheduling nightmare?" *IEEE Solid State Circuits Mag.*, vol. 14, no. 4, pp. 18–27, Fall 2022, doi: 10.1109/MSSC.2022.3201783.
[10] M. Shi, V. Jain, A. Joseph, M. Meijer, and M. Verhelst, "BitWave: Exploiting column-based bit-level sparsity for deep learning acceleration," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2024, pp. 732–746, doi: 10.1109/HPCA57654.2024.00062.
[11] Z.Yuanetal.,"STICKER:Anenergy-efficient multi-sparsity compatible accelerator for convolutional neural networks in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 55, no. 2, pp. 465–477, Feb. 2020, doi: 10.1109/JSSC.2019.2946771.
[12] I. Mirzadeh et al., "ReLU strikes back: Exploiting activation sparsity in large language models," 2023, *arXiv:2310.04564*.
[13] M. Zhu, T. Zhang, Z. Gu, and Y. Xie, "Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern GPUs," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA: ACM, 2019, pp. 359–371, doi: 10.1145/3352460.3358269.
[14] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2017, *arXiv:1611.06440*.
[15] A. Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375*.
[16] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2082–2090.
[17] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "SBNet: Sparse blocks network for fast inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8711–8720, doi: 10.1109/CVPR.2018.00908.
[18] A. Mishra et al., "Accelerating sparse deep neural networks," 2021, *arXiv:2104.08378*.
[19] W. Sun, Z. Zou, D. Liu, W. Sun, S. Chen, and Y. Kang, "Bit-balance: Model-hardware codesign for accelerating NNs by exploiting bit-level sparsity," *IEEE Trans. Comput.*, vol. 73, no. 1, pp. 152–163, Jan. 2024, doi: 10.1109/TC.2023.3324477.
[20] S. Dai, R. Venkatesan, H. Ren, B. Zimmer, W. J. Dally, and B. Khailany, "VS-Quant: Per-vector scaled quantization for accurate low-precision neural network inference," 2021, *arXiv:2102.04503*.
[21] J. Lin et al., "AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration," in *Proc. Mach. Learn. Syst.*, 2024, vol. 6, pp. 87–100.
[22] N. Chauvaux, A. Kneip, C. Posch, K. Makinwa, and C. Frenkel, "An event-based digital compute-in-memory accelerator with flexible operand resolution and layer-wise weight/output stationarity," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2025, pp. 1–4.
[23] A. Parashar et al., "SCNN: An accelerator for compressed-sparse convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 27–40, Jun. 2017, doi: 10.1145/3140659.3080254.
[24] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Piscataway, NJ, USA: IEEE Press, 2017, pp. 246–247, doi: 10.1109/ISSCC.2017.7870353.
[25] E. Qin et al., "Extending sparse tensor accelerators to support multiple compression formats," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 1014–1024, doi: 10.1109/IPDPS49936.2021.00110.
[26] A. Gondimalla, N. Chesnut, M. Thottethodi, and T. N. Vijaykumar, "SparTen: A sparse tensor accelerator for convolutional neural networks," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit., (MICRO)*, New York, NY, USA: ACM, 2019, pp. 151–165, doi: 10.1145/3352460.3358291.
[27] E. Qin et al., "SIGMA: A sparse and irregular GEMM accelerator with flexible interconnects for DNN training," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2020, pp. 58–70, doi: 10.1109/HPCA47549.2020.00015.
[28] Z. Zhu, X. Zhou, C. Wang, L. Tian, and Y. Zhu, "Bit-sparsity aware acceleration with compact csd code on generic matrix multiplication," *IEEE Trans. Comput.*, vol. 74, no. 2, pp. 1–12, Feb. 2024, doi: 10.1109/TC.2024.3483632.
[29] V. Sakellariou, V. Paliouras, I. Kouretas, H. Saleh, and T. Stouraitis, "A multiplier-free RNS-based CNN accelerator exploiting bit-level sparsity," *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 2, pp. 667–683, Apr./Jun. 2024, doi: 10.1109/TETC.2023.3301590.
[30] Z.-G. Liu, P. N. Whatmough, Y. Zhu, and M. Mattina, "S2TA: Exploiting structured sparsity for energy-efficient mobile CNN acceleration," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2022, pp. 573–586, doi: 10.1109/HPCA53966.2022.00049.
[31] M. Huang, A. Shen, K. Li, H. Peng, B. Li, and H. Yu, "EdgeLLM: A highly efficient CPU-FPGA heterogeneous edge accelerator for large language models," 2024, *arXiv:2407.21325*.
[32] H. Lu et al., "Distilling bit-level sparsity parallelism for general purpose deep learning acceleration," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, 2021, pp. 963–976, doi: 10.1145/3466752.3480123.
[33] D. Im, G. Park, Z. Li, J. Ryu, and H.-J. Yoo, "Sibia: Signed bit-slice architecture for dense DNN acceleration with slice-level sparsity exploitation," in *Proc. IEEE Int. Symp. High-Perform. Comput. Architect. (HPCA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 69–80, doi: 10.1109/HPCA56546.2023.10071031.
[34] C. Zhang et al., "DSTC: Dual-side sparsity tensor core for DNNs acceleration on modern GPU architectures," *IEEE Trans. Comput.*, vol. 74, no. 2, pp. 341–355, Feb. 2025, doi: 10.1109/TC.2024.3475814.
[35] J. Albericio et al., "Bit-pragmatic deep neural network computing," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*. New York, NY, USA: ACM, 2017, pp. 382–394, doi: 10.1145/3123939.3123982.
[36] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.
[37] Y. Yang, A. Kneip, and C. Frenkel, "EvGNN: An event-driven graph neural network accelerator for edge vision," 2024, *arXiv:2404.19489*.
[38] Y. Wang et al., "A GNN computing-in-memory macro and accelerator with analog-digital hybrid transformation and CAMenabled search-reduce," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, 2023, pp. 1–2, doi: 10.1109/CICC57935.2023.10121238.
[39] Z. Zhu et al., "Mega: A memory-efficient GNN accelerator exploiting degree-aware mixed-precision quantization," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2024, pp. 124–138, doi: 10.1109/HPCA57654.2024.00020.
[40] G. Jeong, P.-A. Tsai, A. R. Bambhaniya, S. W. Keckler, and T. Krishna, "Abstracting sparse DNN acceleration via structured sparse tensor decomposition," 2024, *arXiv:2403.07953*.
[41] M. Parger, C. Tang, C. D. Twigg, C. Keskin, R. Wang, and M. Steinberger, "DeltaCNN: End-to-end CNN inference of sparse frame differences in videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 12,497–12,506.
[42] C. Gao, D. Neil, E. Ceolini, S.-C. Liu, and T. Delbruck, "DeltaRNN: A power-efficient recurrent neural network accelerator," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA)*, New York, NY, USA: ACM, 2018, pp. 21–30, doi: 10.1145/3174243.3174261.
[43] C. Gao, T. Delbruck, and S.-C. Liu, "Spartus: A 9.4 TOp/s FPGA-based LSTM accelerator exploiting spatio-temporal sparsity," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 1098–1112, Jan. 2024, doi: 10.1109/TNNLS.2022.3180209.
[44] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of averaged time surfaces for robust event-based object classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 1731–1740, doi: 10.1109/CVPR.2018.00186.
[45] C. Frenkel and G. Indiveri, "ReckOn: A 28nm Sub-mm2 task-agnostic spiking

recurrent neural network processor enabling on-chip learning over second-long timescales," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2022, vol. 65, pp. 1–3, doi: 10.1109/ISSCC42614.2022.9731734.

[46] Y. N. Wu, P.-A. Tsai, S. Muralidharan, A. Parashar, V. Sze, and J. Emer, "HighLight: Efficient and flexible DNN acceleration with hierarchical structured sparsity," in *Proc. 56th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA: ACM, 2023, pp. 1106–1120, doi: 10.1145/3613424.3623786.

[47] "NVIDIA ampere architecture." NVIDIA. Accessed: Dec. 20, 2024. [Online]. Available: https://www.nvidia.com/en-us/data-center/ampere-architecture/

[48] Y. N. Wu, P.-A. Tsai, A. Parashar, V. Sze, and J. S. Emer, "Sparseloop: An analytical approach to sparse tensor accelerator modeling," in *Proc. 55th IEEE/ACM Int. Symp. Microarchit. (MICRO)*, 2022, pp. 1377–1395, doi: 10.1109/MICRO56248.2022.00096.

[49] N. Nayak, T. O. Odemuyiwa, S. Ugare, C. Fletcher, M. Pellauer, and J. Emer, "TeAAL: A declarative framework for modeling sparse tensor accelerators," in *Proc. 56th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA: ACM, 2023, pp. 1255–1270, doi: 10.1145/3613424.3623791.

About the Authors

Man Shi (man.shi@kuleuven.be) received her B.Sc. degree from the School of Information Science and Engineering from Shandong University China, in 2017, and her M.Sc. degree from the Institute of Microelectronics, Tsinghua University, China, in 2020. She is with the Department of Electrical Engineering, KU Leuven, 3000 Leuven, Belgium, where she is currently pursuing a Ph.D. degree in accelerators architecture for deep neural network with the MICAS Laboratories, KU Leuven, 3000 Leuven, Belgium. Her current research interests include low-power deep neural network hardware accelerator design, algorithm-hardware codesign, and reconfigured computation. She is a Student Member of IEEE.

Adrian Kneip (adrian.kneip@kuleuven.be) received his M.Sc. degree in electrical engineering from the Université Catholique de Louvain (Belgium), and his Ph.D. degree in 2024. He is now a postdoctoral research fellow with the Microelectronics Department (EEMCS Faculty), at Delft University of Technology, 2628 CD Delft, The Netherlands, and is also with the Department of Electrical Engineering, KU Leuven, 3000 Leuven, Belgium. His research interests include the design of ultralow-power digital ICs, including analog/mixed-signal in-memory computing accelerators for edge artificial intelligence. He is the author of several research papers in IEEE conferences and journals, receiving the Best Student Paper Award at the 2022's European Conference on Solid-State Circuits/European Conference on Solid-State Device Research conference. He serves as a reviewer for various IEEE journals. From 2020 to 2023, he also was a student representative in the IEEE Benelux Section. He is a Member of IEEE.

Nicolas Chauvaux (n.chauvaux@tudelft.nl) received his B.S. and M.S. degrees in electrical engineering from the Catholic University of Louvain-la-Neuve, Louvain-la-Neuve, Belgium, in 2019 and 2021, respectively. He is with the Microelectronics Department (EEMCS Faculty), Delft University of Technology, 2628 CD Delft, The Netherlands. He is currently pursuing a Ph.D. degree at the Delft University of Technology, 2628 CD Delft, The Netherlands. His research interests include flexible digital computing-in-memory, low-power and low-latency system-level design optimization for spiking neural network accelerators, and design space exploration tools. He is a Graduate Student Member of IEEE.

Jiacong Sun (jiacong.sun@kuleuven.be) received his M.S. in microelectronics and solid-state electronics from Peking University in 2020, where he focused on low-power SRAM compilation research. He then worked as a hardware engineer at ZTE Sanechips Technology Corporation until 2021. Subsequently, he returned to Peking University as a research assistant, contributing to in-memory computing hardware development projects. Since 2022, Jiacong has been pursuing research at the MICAS Laboratory, KU Leuven's Department of Electrical Engineering, 3000 Leuven, Belgium. His research interests include artificial intelligence accelerator modeling to bridge the gap between software and hardware development. He is a Graduate Student Member of IEEE.

Charlotte Frenkel (c.frenkel@tudelft.nl) received her Ph.D. from Université Catholique de Louvain in 2020 and was a postdoctoral researcher at the Institute of Neuroinformatics, University of Zürich, and ETH Zürich, Switzerland. She is an assistant professor at Delft University of Technology, 2628 CD Delft, The Netherlands. Her research aims at bridging the bottom-up (bio-inspired) and top-down (engineering-driven) design approaches toward neuromorphic intelligence, with a focus on digital neuromorphic processor design, embedded machine learning, and brain-inspired on-device learning. She coleads NeuroBench and serves as an associate editor for the *IEEE Transactions on Biomedical Circuits and Systems*. She is a Member of IEEE.

Marian Verhelst (marian.verhelst@kuleuven.be) received her Ph.D. degree from KU Leuven in 2008, and worked as a research scientist at Intel Labs from 2008 till 2010. She is the Department of Electrical Engineering, KU Leuven, 3000 Leuven, Belgium and is a professor at the MICAS Laboratory, KU Leuven, 3000 Leuven, Belgium, and a research director at imec, 3001 Leuven, Belgium. Her research interests include embedded machine learning, hardware accelerators, and low-power edge processing. She is active in the Technical Programming Committees of the International Solid-State Circuits Conference, European Conference on Solid-State Circuits, DATE, and International Symposium on Computer Architecture. She served in the board of directors of tinyML, as an associate editor for *IEEE Transactions on Very Large Scale Integration Systems*, *Transactions on Circuits and Systems-II*, and the *IEEE Journal of Solid-State Circuits*, and as a member of the STEM advisory committee to the Flemish Government, and is a member of the Belgian Royal Academy of Science and Arts. She is a Fellow of IEEE.