CAMformer: Associative Memory is All You Need

Tergel Molom-Ochir[†]*, Benjamin F. Morris[†], Mark Horton[†], Chiyue Wei[†], Cong Guo[†], Brady Taylor[†],
Peter Liu[†], Shan X. Wang[‡], Deliang Fan[§], Hai "Helen" Li[†], and Yiran Chen[†]
[†]Duke University, [‡]Stanford University, [§]Arizona State University
{tergel.molom-ochir, benjamin.morris, hai.li, yiran.chen}@duke.edu

***Abstract*—Transformers face scalability challenges due to the quadratic cost of attention, which involves dense similarity computations between queries and keys. We propose CAMformer, a novel accelerator that reinterprets attention as an associative memory operation and computes attention scores using a voltage-domain Binary Attention Content Addressable Memory (BA-CAM). This enables constant-time similarity search through analog charge sharing, replacing digital arithmetic with physical similarity sensing. CAMformer integrates hierarchical two-stage top-k filtering, pipelined execution, and high-precision contextualization to achieve both algorithmic accuracy and architectural efficiency. Evaluated on BERT and Vision Transformer workloads, CAMformer achieves over 10× energy efficiency, up to 4× higher throughput, and 6–8× lower area compared to state-of-the-art accelerators—while maintaining near-lossless accuracy.**

I. Introduction

Transformer-based models have become foundational in various domains, including natural language processing, computer vision, and speech recognition [1], [2]. Their ability to model long-range dependencies through self-attention mechanisms has led to significant performance improvements across numerous tasks [3]–[9]. However, the computational and memory demands of the attention mechanism, particularly its quadratic complexity with respect to sequence length, pose challenges for deploying Transformers in resource-constrained environments and for processing long sequences efficiently [10]. Traditional hardware accelerators often address this bottleneck by optimizing matrix multiplication (MatMul) operations, such as the $QK^\top$ and AV computations inherent in the attention mechanism [11], [12]. Techniques like low-precision arithmetic, sparsity exploitation, and memory tiling have been employed to mitigate these challenges [13]–[15]. Despite these efforts, the fundamental issue remains: the attention mechanism's reliance on dense matrix operations leads to substantial data movement and energy consumption. An alternative perspective considers attention as a form of content-based memory retrieval, akin to operations performed by content-addressable memory (CAM) systems [16].

Content-Addressable Memory (CAM), also known as associative memory, is a specialized form of memory that enables data retrieval based on content rather than specific addresses [17], [18]. Unlike traditional Random Access Memory (RAM), where data is accessed by providing an explicit address, CAM allows for simultaneous comparison of input data against all stored entries, returning the address of matching data [19]–[21]. This parallel search capability makes CAM exceptionally efficient for applications requiring rapid data matching and retrieval, such as IP look up, forwarding engine, Deep Random Forest acceleration, and DNA classification [22]–[25]. CAM operates by broadcasting the input data across the memory array, where each cell compares the input with its stored content. If a match is found, the corresponding address is returned. This mechanism eliminates the need for sequential searches, significantly reducing search time and enhancing performance in data-intensive tasks. By aligning the computational model of attention with the associative retrieval capabilities of CAMs, we can explore hardware architectures that inherently support efficient attention computations.

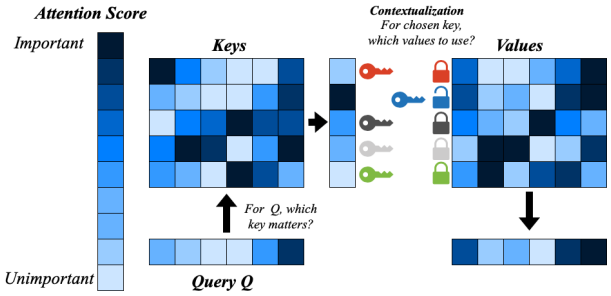


Fig. 1: Attention as a key-lock mechanism. The query vector (Q) is used to determine similarity with stored keys (K), producing an attention score. The resulting soft selection determines which value vectors (V) to aggregate. This metaphor illustrates attention as an associative memory operation, where queries "unlock" relevant stored information.

In this work, we introduce CAMformer, a novel hardware accelerator that leverages CAM structures to perform attention operations. CAMformer reinterprets the attention mechanism as an associative memory process, enabling in-memory computation of attention scores. This approach improved throughput, lowers energy consumption, and offers scalability advantages. Our contributions are as follows:

- **Reconceptualization of Attention**: We present a novel interpretation of the attention mechanism as an associative memory operation, aligning it with CAM functionalities.
- **CAM-based Attention Score**: We introduce a attention-score module that lowers circuit complexity for similarity

*Corresponding Author. This work was supported by the U.S. Department of Energy under Award Nos. DE-SC0026254 and SC0026382, NSF under Award Nos. 2328805 and 2328712, and AFOSR under Award No. FA9550-24-1-0322.

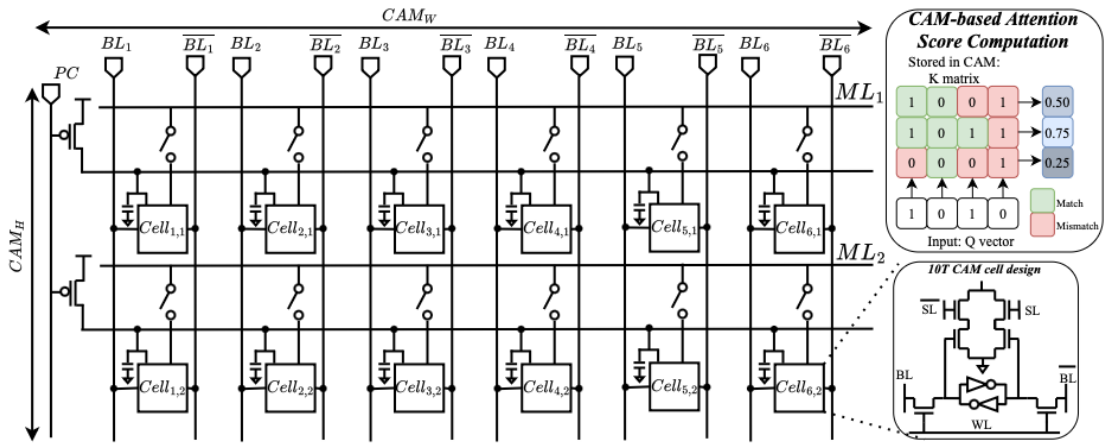


Fig. 2: Array-level architecture of an example 2×6 BA-CAM module's array used for binary attention computation. Each row in the array performs parallel similarity matching against the broadcast query, with charge sharing accumulating match strength on shared matchlines. The inset shows the 10T1C CAM cell structure and an illustrative example of binary attention scoring based on Hamming similarity.

search.

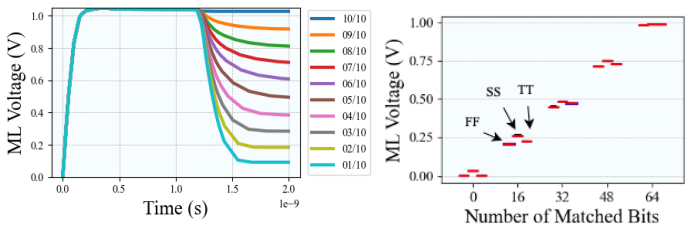
- **CAMformer Architecture**: We design a hardware accelerator that integrates CAM structures to perform attention computations, reducing reliance on traditional MatMul.

The remainder of this paper is organized as follows: Seciton II details the CAMformer circuits and the MatMul operation. Section III details the CAMformer architecture and its operational principles. Section IV presents performance evaluations and comparisons. Finally, Section V concludes the paper and outlines potential directions for future research.

II. Binary Attention CAM

A. Circuit-Level Design

1) Cell design: We implement a 10T1C (10 transistors 1 capacitor) CAM cell tailored for partial-match, and binary vector-matrix multiplication operations. Each cell stores 1-bit data in SRAM logic, and its match results representation in a $22fF$ MIM capacitor, and compares it to the query via XNOR logic. When matched, the precharged capacitor stays high; otherwise, discharged. The charge-sharing mechanism along the matchline enables analog accumulation of Hamming similarity, replacing digital popcount logic [26], [27]. The design operates in four phases—precharge, broadcast, match, and charge share—and avoids destructive readout while supporting pipelined operation.



(a) (b)

Fig. 3: (a) Matchline voltage traces for varying partial matches in 1×10 BA-CAM. (b) PVT analysis across corners for 16×64 array.

2) Array & Matchline Architecture: The BA-CAM array computes binary VMM by broadcasting the query of $Q \in \{0,1\}^{d_k \times 1}$ across columns of $K^\top \in \{0,1\}^{d_k \times N}$ and accumulating bitwise match results as analog voltages in $[0,1]$ on each matchline, resulting in $A \in [0,1]^{1 \times N}$. These voltages, linearly proportional to Hamming similarity, are digitized using shared 6-bit SAR ADCs. Unlike TD-CAM, which uses time-domain delay sensing and requires time-difference amplifiers (TDA), our voltage-based scheme is simpler, faster, and significantly more robust to PVT variations [28]. This linear, delay-free sensing model eliminates timing calibration and scales efficiently to larger arrays and higher throughput without analog complexity. BA-CAM's array is shown in Fig. 2. Compared to the conventional exact matching operation, partial matching via charge sharing allows Hamming distance computation in analog (shown in Fig. 3a).

B. Microarchitecture & VMM Engine

1) BIMV — Binary In-Memory Vector-Matrix Multiplication: We implement binary $QK^\top$ with a *Binary Matrix–Vector (BIMV)* engine built on a *Binary-Attention CAM (BA-CAM)* array. Each row stores a binary key (from $K^\top$); the query is broadcast. Bitwise `XNOR` happens in parallel and matching bits charge-share onto the matchline; the resulting voltage encodes Hamming similarity and is sensed by a simple ADC/comparator, eliminating digital arithmetic and yielding constant-latency compute [26]. CAM performs similarity on the matchline (time/voltage), whereas CiM performs `XNOR`+popcount on bit-lines and digitizes via column-muxed ADCs—adding peripheral/serialization overhead (Table I). We extend capacitive CAM for binary MatMul by linearly scaling matchline outputs

TABLE I: Circuit-Level Comparison of BIMV / Attention-Score *Modules*

Feature	CiM [29]	TD-CAM [28]	BA-CAM (Ours)
Sensing	BL sum (XNOR+Accumulate)	Time ML	Voltage ML
Similarity	No (popcount)	Yes (delay)	Yes (voltage)
Peripherals	Flash ADC (MUX) + Adder Tree	TDA + tune	Shared SAR
Tech	65 nm	65 nm	65 nm
Module area	High (ADC)	Med–High (TDA)	**Low** (shared SAR)
V_{DD}	0.6–1.0 V	1.2 V	1.2 V
Freq	18.5 MHz	200 MHz	**500 MHz**
Overall err.	7% (pred.)	7.76%	**1.12%***
PVT robustness	Moderate (Calibrated ADC)	Low (time-domain)	**High** (voltage-sensed)
Complexity	Very high (ADC+Adder Tree)	High (TDA)	**Low** (no MAC/popcnt)

* simulated at $\sigma = 1.4\%$.

with a 6-bit ADC: $s = 2 \cdot \mathrm{ADC}(v) - CAM_W$, mapping $[0,1] \to [-64, 64]$ while preserving attention-score ordering. Unlike digital BIMV (e.g., XNOR-NE [29], BitFusion [30]) that needs SRAM fetch, external `XNOR`–popcount, and sequential accumulation, CAM unifies compute and memory. TD-CAM [28] removes digital popcount and is row-parallel, but encodes scores in discharge *delay*, requiring TDAs, tight delay matching, and calibration—introducing timing complexity and weakening robustness.

Our BA-CAM encodes similarity by *voltage*, not time: each matching bit adds charge to the matchline; the sensed voltage directly reflects similarity, removing delay-path tuning and TDAs. Unlike TD-CAM's nonlinear delay, BA-CAM yields a linear voltage response, stronger PVT tolerance, and cleaner digital integration (Table I). It scales flexibly and needs no majority-threshold voting. Under PVT, BA-CAM holds matchline deviation within 5.05% with mean error as low as 1.12% across TT/SS/FF (Fig. 3b), outperforming prior TD-CAMs that show delay deviations up to $\leq 7.76\%$. BA-CAM avoids timing-based sensing entirely—computing similarity physically (via voltage) in constant time—eliminating external logic, alignment, and calibration, and delivering energy-efficient, fully in-memory associative compute; digital BIMV *emulates* association, BA-CAM *is* associative.

We illustrate the operation of BIMV in BA-CAM in Fig. 4.

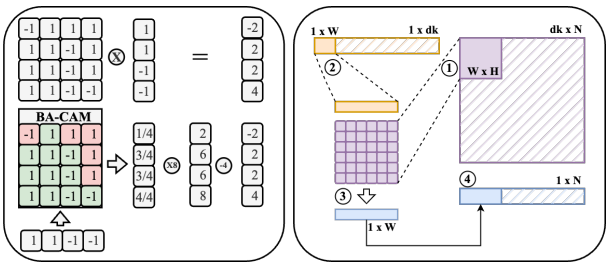


Fig. 4: Illustration of matrix-vector multiplication. Comparison of conventional (left top) versus CAM-based (left bottom). Tiling steps for larger matrix-vector operations (right).

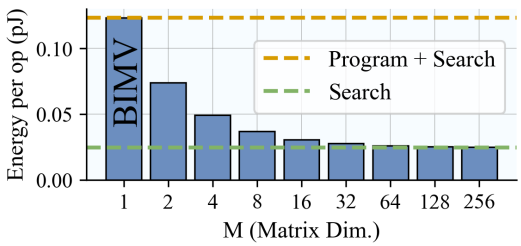


Fig. 5: Per-op energy vs. matrix dimension M in BA-CAM. Larger M reduces energy by amortizing programming cost. Dashed lines show search-only and total energy bounds.

On the top left, a conventional approach to BIMV is shown. Digital components perform multiplication using digital logic. On the bottom left, a BA-CAM array performs the binary multiplication/matching and accumulation all in the analog domain, producing partial results in the range $[0, 1]$. These values are converted to signed values centered around 0 by the fixed functional multiply and subtract units.

On the right, we illustrate how we use tiling to general BIMV for tensors with dimensions larger than the CAM array: $Q \in \{-1, +1\}^{1 \times d_k}$ and $K^\top \in \{-1, +1\}^{d_k \times N}$. We assume that d_k and N are multiples of CAM_W and CAM_H for simplicity, respectively, but padding can be applied otherwise. In step ①, we load a $CAM_W \times CAM_H$ sized tile of $K^\top$ into the BA-CAM array. In step ②, a CAM_W sized segment of the Q vector is loaded into the query register. Then, in step ③, we perform the associative tiled-MAC operation to receive our partial output of size CAM_W. If $N > CAM_W$, we tile horizontally and concatenate partial results to the final result vector in step ④. If $K > CAM_H$, we must tile vertically and accumulate partial results into the same segment of the final result vector (repeat steps ①-④ again).

For higher-precision V, we decompose $K^\top$ entries into binary slices (LSB→MSB) and run per-slice BIMM. Slice outputs are digitally shifted and accumulated, adding precision without changing the CAM path. This supports binary–integer MatMul and quantized $V \in$ `int2`, `int4`, `int8`.

III. CAMformer Architecture

In this section, we develop CAMformer, an attention accelerator built around the BA-CAM circuit introduced in Sec. II. CAMformer attention is shown in Eq. 1. First, we explain the execution model of CAMformer and explain how a larger Deep Learning (DL) system can integrate CAMformer into the attention workflow in Sec. III-A. With this model in mind, we next introduce the high-level CAMFormer architecture in Sec. III-B. Finally, we explore in detail some of the novel optimizations that enable CAMformer to surpass state-of-the-art attention computation performance.

$$\begin{aligned}&\text{CAMformer-Attn}(Q, K, V)\\ &\quad = \text{SoftMax}\big(\text{Top-32}(QK^T)\big) \cdot V\end{aligned} \tag{1}$$

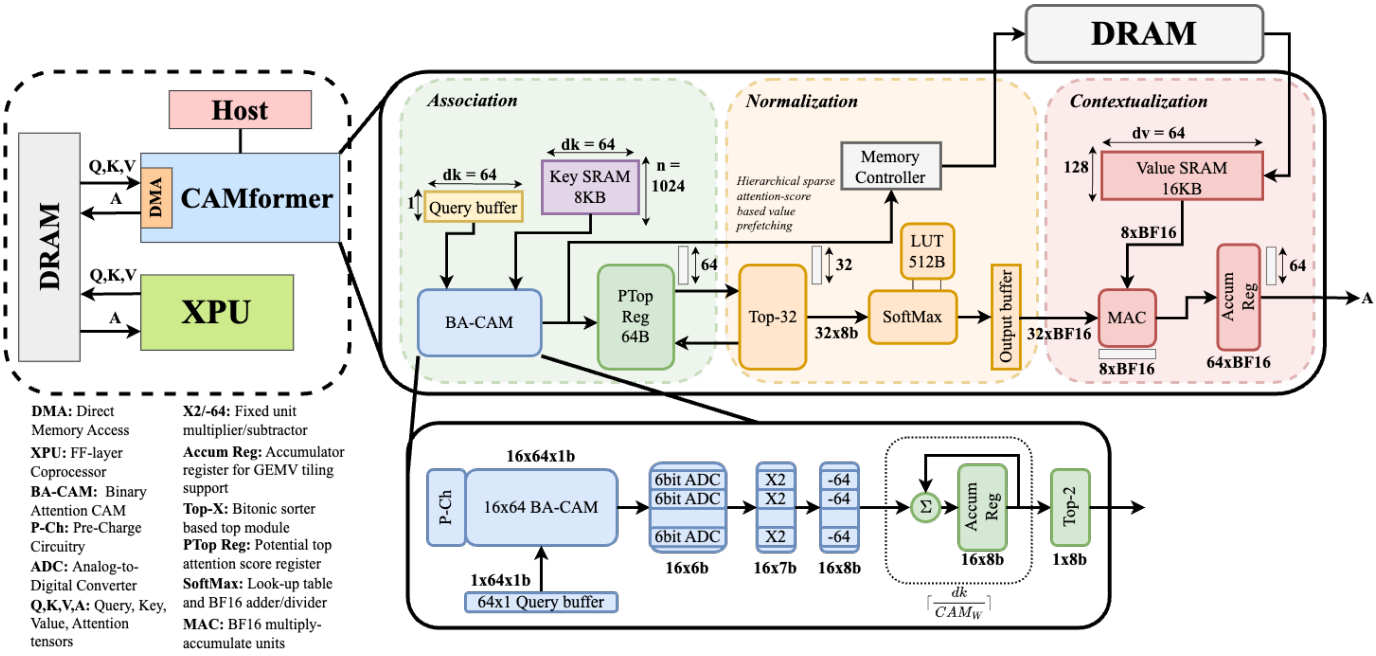


Fig. 6: The design consists of three pipelined stages—association, normalization, and contextualization—centered around a 16×64 BA-CAM array. The BA-CAM computes binary attention scores, which are sparsified and normalized before BF16 attention output is computed using BF16 MACs. Integration with XPUs and DRAM enables efficient end-to-end attention processing.

A. System Integration

CAMformer operates as an attention accelerator for large-scale DL systems. The CAMformer accelerator integrates into existing systems that contain other accelerated processing units, XPUs (GPUs, TPUs, etc.), responsible for the dense matrix multiplications such as FF (feed-forward) layers. We illustrate the integration of CAMformer into a larger system in Fig. 6. We utilize shared memory-based communication with XPUs for the binary Q, K tensors and BF16 V, A tensors. While an external host (CPU) programs and monitors the CAMformer accelerator, CAMformer uses a local Direct Memory Access (DMA) engine and memory controller for fast access to global memory during computation. CAMformer is currently optimized for decoder-style (causal) attention.

B. CAMformer Overview

CAMformer has three pipelined stages (Fig. 6): *Association* computes $QK^\top$ via BA-CAM and launches hierarchical sparse ranking; *Normalization* finalizes ranking and applies softmax, $\hat{A} = \mathrm{SoftMax}(\text{Top-32}(QK^\top))$; *Contextualization* performs high-precision sparse MV with V, yielding $A = \hat{A}V$. Each stage is separable, feed-through, and consumes/produces the prior/next stage's data.

1) Association: We compute attention scores from binary Q, K. The Key SRAM stores the full binarized K and is off the critical path since many queries reuse one K. The Query buffer holds a single query (batch=1). BA-CAM executes the binarized MVM (Sec. II-B1). Although batching can improve $QK^\top$ energy, it inflates downstream hardware, so we use batch=1. The CAM is 16×64: height 16 reduces ADC overhead; width 64 avoids vertical tiling for d_k=64 (Sec. II-B1). For larger d_k, an accumulation register enables vertical tiling. For each tile, we program BA-CAM and run an associative tiled MAC with the query; ADC precision covers the full match range. A bitonic Top-2 picks the highest score per tile; we keep it for sparse attention and drop others. The bitonic sorter also makes sparsity easily configurable. Per tile, Top-2 scores go to a potential-top register, and indices go to the local memory controller to prefetch the corresponding V entries. We use g=16 tiles with Top-2 per tile $\Rightarrow$ Top-32 overall. k fixes the returned indices (V-buffer depth), so we co-design k=32 with V-SRAM capacity: large enough to shrink candidates, small enough to bound accuracy loss; larger k offers diminishing returns (cf. recall bound). If stage-1 scores satisfy $|\hat{s}_i - s_i| \le \varepsilon$, a margin $\Delta_k \equiv s_{(k)} - s_{(k+1)} > 2\varepsilon$ ensures recall@k = 1. For binary similarity (mean of m Bernoulli matches, e.g., $m = d_k$), Hoeffding yields $\Pr[\text{drop any true top-}k] \le k(N-k)\exp(-2m\,\delta_{\min}^2)$, consistent with HAD's binarized Q/K + top-N sparsification. [31], [32]

2) Normalization: We select the top-32 attention scores from the 128 candidates produced in association. A bitonic-sorter Top-32 block provides runtime sparsity flexibility. To reduce area, we use a 64-input module and refine across batches as each 16-tile group yields 32 new top-2 candidates (after the initial 32 tiles). The SoftMax engine uses a 512 B LUT, one BF16 accumulator, and one BF16 divider. For each of the 32 8-bit scores, we compute $\exp(x)/\sqrt{d_k}$ via the LUT, accumulate the denominator on the fly, and normalize once it is complete.

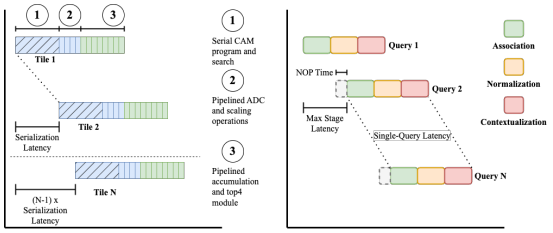


Fig. 7: CAMformer pipelining strategies. Left: Fine-grained pipelining overlaps CAM operations within the association stage. Right: Coarse-grained pipelining enables query-level parallelism across all stages.

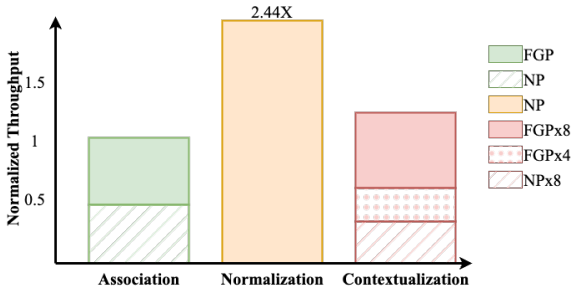


Fig. 9: CAMformer throughput by stage. Fine-grained pipelining and parallelism boost association and contextualization stages, enabling balanced pipeline performance.

The outputs are valid probabilities: each in $[0, 1]$ and summing to 1.

3) Contextualization: In the contextualization stage, we compute the BF16 Attention tensor. The use of higher-precision BF16 has been noted as a requirement for maintaining model accuracy [32]. Our Value SRAM has been loaded with a sufficient number of valid entries to begin the multiply-accumulate (MAC) operations.

C. Optimizations

1) Fully Binarized Attention-Score: Fully binarizing Q and K enables analog $QK^\top$ via the associative BA-CAM (Sec. II) and cuts on-chip storage to 6.25% of BF16 for the Query buffer and Key SRAM. The resulting bounded score range also makes SoftMax cheap: a small LUT handles $\exp(\cdot)$ and normalization.

2) Fine-grained pipelining: We utilize fine-grained pipelining to accelerate the critical path of each stage. The association stage uses fine-grained pipelining strategies so that tiling steps in different parts of the pipeline can operate concurrently. We illustrate the fine-grained pipelining and effect of CAM serialization latency requirements for the association stage in Fig. 7 left. The normalization stage utilizes fine-grained pipelining in the SoftMax module. Because the normalization stage is not on the critical path, we perform the accumulation and division serially. By utilizing a pipelined BF16 divider, the overall latency for the SoftMax operation is greatly reduced from $32t_{\text{div}}$ to $31 + t_{\text{div}}$, where t_{div} is the end-to-end latency of the BF16 divider. In the contextualization stage, we utilize fine-grained pipelining for the MAC operations.

3) Coarse-grained pipelining: In addition to fine-grained pipelining within each stage, we also perform coarse-grained pipelining between stages. This improves hardware utilization by ensuring that each stage remains busy. The overall throughput is determined by the latency of the longest stage. Other stages will incur an amount of stall time equivalent to the difference between their latency and the longest stage latency. The coarse-grained pipelining strategy and corresponding total no-op time are illustrated in Fig. 7 right. In order to maximize hardware utilization, we balance the throughput of the stages in our design space exploration (Sec. IV-B).

4) Hierarchical sparse attention-score ranking: We use a two-stage top-k to cut on-chip score storage and enable V prefetch. Stage-1 keeps top-2 per 16 during each tile; Stage-2 finalizes ranking. Each top-2 triggers the MC/DMA to fetch the corresponding V. V is laid out contiguously in DRAM: rows of 64×16b, so 64 rows fit an 8KB page. With no interleaving, one t_{RC} serves each set of 64 scores; using HBM3 $t_{\text{RC}} = 48$ns [33], the pipeline fully hides DRAM latency. Required bandwidth is ≈ 50,GB/s, which a single HBM3 channel can sustain [34].

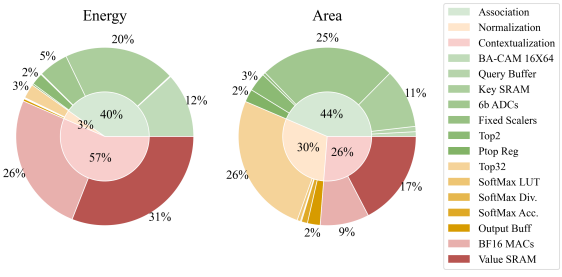


Fig. 8: Breakdown of CAMformer energy and area. Energy is dominated by BF16 MACs and Value SRAM, while area is split across all stages with largest contributions from SRAM and normalization logic.

IV. Results

A. Experimental Methodology

We implement CAMformer's digital blocks in Verilog and synthesize with Synopsys Design Compiler (TSMC 65 nm)

TABLE II: Performance Comparison of CAMformer Variants vs. Existing Architecture Solutions at 1 GHz

Accelerator	Q/K/V bits	Core (#)	Thruput (qry/ms)	Energy Eff. (qry/mJ)	Area (mm^2)	Power (W)
MNNFast [35]	32/32/32	1	28.4	284	–	1.00*
A^3 [36]	8/8/8	1	52.3	636	2.08	0.82
SpAtten$^{1/8}$ [37]	12/12/12	1	85.2	904	1.55	0.94
HARDSEA [38]	8/8/8	12	187†	191†	4.95	0.92
CAMformer	1/1/16	1	191	9045	0.26	0.17
CAMformer$_{\text{MHA}}$	1/1/16	16	3058	9045	4.13	2.69

Notes: MNNFast does not report Q/K/V precision; treated as FP32 (32/32/32). SpAtten uses progressive MSB+LSB quantization (e.g., 6+4), with an up-to-12b on-chip datapath. †Converted from 802.1 GOPS and 821.3 GOPS/W assuming 4.3 GOP/query.

for area/timing/power. Analog CAM arrays are characterized in HSPICE. ADC, BF16 MAC, and BF16 divider costs follow [39]–[41] and are scaled to 45 nm via [42]; DRAM energy is 2.33 nJ/bit [43]. A Python system simulator models performance, energy, and area. Metrics include end-to-end attention latency (with memory), energy (GOP/J), power, and queries/s. CAMformer$_{\text{MHA}}$ spans 16 heads across all 16 HBM channels. We compare against TPU [44], WSE [45], SpAtten [37], and A^3 [36] (Table II).

B. Design Space Exploration

We balance the throughput of CAMformer's three stages through fine-grained pipelining and data parallelism optimizations (Fig. 9). The normalization stage provides sufficient throughput with minimal parallelism due to sparse attention optimization, while the contextualization stage requires 8 parallel MAC units to match the association stage's throughput. As shown in Fig. 8, area is distributed across SRAM (42%), Top-32 module (26%), and processing units, while energy is dominated by the contextualization stage (57%) due to BF16 precision requirements; component-wise, Value/Key SRAM account for 31%/20%, MACs 26%, and BA-CAM 12% of total energy.

C. State-Of-The-Art Comparison

We compare performance of state-of-the-art accelerators on single query attention processing for a BERT-Large [46] model with 16 heads, $d_k = d_v = 64$, and sequence length $n = 1024$. The same CAM top-k mechanism extends to decoders - CAM search over a growing KV cache each step (causal) - and to encoder–decoder models via non-causal search over encoder keys. For longer contexts, CAMformer would scale by provisioning proportionally larger BA-CAM (keys) and V-SRAM (values) sized to the target maximum context (the per-step top-k V-buffer remains fixed by k); note that KV-cache memory grows with sequence length in practice. In Fig. 10, we compare industry products: Cerebras WSE2 [45], Groq TSP [47], Google TPUv4 [44] to the best-performing academic accelerators on this attention workload. We find that the Pareto front of projected academic accelerators (projected node from 45nm to 22nm), defined at the CAMformer point, exceeds that of the industry products, defined at the TPUv4 point.

D. Algorithmic Accuracy

Hamming Attention Distillation (HAD) [32] binarizes Q, K with $< 3\%$ top-1 drop on ImageNet [48] and GLUE [49].

TABLE III: Top-1 accuracy with two-stage HAD on DeiT models. Accuracy remains near baseline for $k \geq 2$ with minimal degradation using group size 16.

first stage k	DeiT-B	DeiT-S	DeiT-T
HAD baseline	79.24	75.60	66.58
k=8	79.27	75.68	66.53
k=4	79.26	75.65	66.48
k=2	79.16	75.29	65.86
k=1	78.11	72.32	61.03

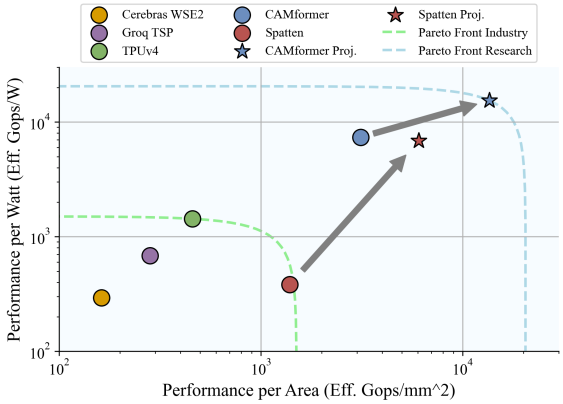


Fig. 10: CAMformer (and projected scaling) lies on the research Pareto frontier, surpassing TPUv4 and WSE2 in performance-per-area and performance-per-watt; points report *effective* GOPS/W at the Table II Q/K/V precisions under fixed accuracy/latency, not peak TOPS.

CAMformer replaces single-stage top-k with two-stage top-k for early filtering and prefetching, reducing storage and hiding DRAM latency. Added loss is negligible: ImageNet $\leq 0.2\%$ even with small first-stage k (Table III); GLUE (group $= 16$, $k \in 2, 4$) $\leq 0.4\%$ vs. single-stage HAD (Table IV). Thus, CAMformer keeps HAD's accuracy while enabling efficient binary-attention inference.

V. Conclusion

We presented CAMformer, a novel accelerator that reinterprets attention as an associative memory operation. Using a voltage-domain Binary Attention CAM (BA-CAM), CAMformer computes similarity scores with constant latency and minimal overhead. Our design achieves over 10× energy efficiency, up to 4× higher throughput, and 6–8× lower area compared to state-of-the-art attention accelerators. while maintaining near-lossless accuracy via two-stage Hamming Attention Distillation.

TABLE IV: GLUE accuracy with two-stage HAD using group size 16. Accuracy remains comparable to the single-stage baseline for $k = 2$ and $k = 4$, with less than 0.4% average degradation.

Metric	HAD baseline	first-stage k=4	first-stage k=2
MNLI	82.45/82.84	82.37/82.98	82.31/82.74
QQP	90.11	90.01	89.87
QNLI	89.68	89.60	89.54
SST-2	91.63	91.42	91.28
CoLA	55.47	55.16	54.90
STS-B	87.46	87.27	87.27
MRPC	83.82	83.87	83.87
RTE	65.70	64.33	64.62
Avg	80.81	80.54	80.48

REFERENCES

[1] S. Khan *et al.*, "Transformers in vision: A survey," *ACM Comput. Surv.*, vol. 54, no. 10s, Sep. 2022.

[2] T. Lin *et al.*, "A survey of transformers," 2021.

[3] S. R. Choi *et al.*, "Transformer architecture and attention mechanisms in genome data analysis: A comprehensive review," *Biology (Basel)*, vol. 12, no. 7, p. 1033, Jul. 2023.

[4] J. Yang *et al.*, "Focal attention for long-range interactions in vision transformers," in *Advances in Neural Information Processing Systems*, M. Ranzato *et al.*, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 30 008–30 022.

[5] A. Hernández *et al.*, "Attention mechanisms and their applications to complex systems," *Entropy*, vol. 23, p. 283, 02 2021.

[6] Q. Wang *et al.*, "Learning deep transformer models for machine translation," 2019.

[7] P. Karpov *et al.*, *A Transformer Model for Retrosynthesis*, 09 2019, pp. 817–830.

[8] K. S. Kalyan *et al.*, "Ammus : A survey of transformer-based pretrained models in natural language processing," 2021.

[9] S. Madan *et al.*, "Transformer models in biomedicine," *BMC Med. Inform. Decis. Mak.*, vol. 24, no. 1, p. 214, Jul. 2024.

[10] F. D. Keles *et al.*, "On the computational complexity of self-attention," 2022.

[11] C. Guo *et al.*, "A survey: Collaborative hardware and software design in the era of large language models," 2024.

[12] S. D. Yamini *et al.*, "Hardware accelerator for transformer based end-to-end automatic speech recognition system," in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2023, pp. 93–100.

[13] A. Inci *et al.*, "Quidam: A framework for quantization-aware dnn accelerator and model co-exploration," 2022.

[14] S. Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," 2016.

[15] H. Wang *et al.*, "Sofa: A compute-memory optimized sparsity accelerator via cross-stage coordinated tiling," 2024.

[16] P.-P. Manea *et al.*, "Gain cell-based analog content addressable memory for dynamic associative tasks in ai," 2024.

[17] K. Pagiamtzis *et al.*, "Content-addressable memory (cam) circuits and architectures: a tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, 2006.

[18] R. Karam *et al.*, "Emerging trends in design and applications of memory-based computing and content-addressable memories," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1311–1330, 2015.

[19] X. Liu *et al.*, "Analog content-addressable memory from complementary fefets," *Device*, vol. 2, no. 2, p. 100218, 2024.

[20] B. L. H. Molom-Ochir, T. Taylor *et al.*, "Advancements in content-addressable memory (cam) circuits: State-of-the-art, applications, and future directions in the ai domain," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2025.

[21] L. Chisvin *et al.*, "Content-addressable and associative memory: alternatives to the ubiquitous ram," *Computer*, vol. 22, no. 7, pp. 51–64, 1989.

[22] S. Veeramani *et al.*, "Efficient IP lookup using hybrid trie-based partitioning of TCAM-based open flow switches," *Photonic Netw. Commun.*, vol. 28, no. 2, pp. 135–145, Oct. 2014.

[23] M. Moradi *et al.*, "Caesar: high-speed and memory-efficient forwarding engine for future internet architecture," in *2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2015, pp. 171–182.

[24] E. Garzón *et al.*, "Hamming distance tolerant content-addressable memory (hd-cam) for dna classification," *IEEE Access*, vol. 10, pp. 28 080–28 093, 2022.

[25] X. Yin *et al.*, "Deep random forest with ferroelectric analog content addressable memory," *Science Advances*, vol. 10, no. 23, p. eadk8471, 2024.

[26] X. Ma *et al.*, "Capcam: A multilevel capacitive content addressable memory for high-accuracy and high-scalability search and compute applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1770–1782, 2022.

[27] H. Zhong *et al.*, "Asmcap: An approximate string matching accelerator for genome sequence analysis based on capacitive content addressable memory," 2023.

[28] W. Choi *et al.*, "Content addressable memory based binarized neural network accelerator using time-domain signal processing," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[29] F. Conti *et al.*, "Xnor neural engine: A hardware accelerator ip for 21.6-fj/op binary neural network inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, p. 2940–2951, Nov. 2018.

[30] H. Sharma *et al.*, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 764–775.

[31] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *JASA*, vol. 58, no. 301, pp. 13–30, 1963.

[32] M. Horton *et al.*, "Hamming attention distillation: Binarizing keys and queries for efficient long-context transformers," 2025.

[33] S. Li *et al.*, "Dramsim3: A cycle-accurate, thermal-capable dram simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.

[34] J. S. S. T. Association, "High bandwidth memory (hbm3) dram standard," JEDEC Solid State Technology Association, Tech. Rep. JESD238B.01, Apr. 2025.

[35] H. Jang *et al.*, "Mnnfast: A fast and scalable system architecture for memory-augmented neural networks," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019, pp. 250–263.

[36] T. J. Ham *et al.*, "A^3: Accelerating attention mechanisms in neural networks with approximation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Los Alamitos, CA, USA: IEEE Computer Society, Feb 2020, pp. 328–341.

[37] H. Wang *et al.*, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, Feb. 2021.

[38] S. Liu *et al.*, "Hardsea: Hybrid analog-reram clustering and digital-sram in-memory computing accelerator for dynamic sparse self-attention in transformer," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 2, pp. 269–282, 2024.

[39] L. Chen *et al.*, "A 0.95-mw 6-b 700-ms/s single-channel loop-unrolled sar adc in 40-nm cmos," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 3, pp. 244–248, 2017.

[40] A. Tiwari *et al.*, "Design of a low power bfloat16 pipelined mac unit for deep neural network applications," in *2021 IEEE Region 10 Symposium (TENSYMP)*, 2021, pp. 1–8.

[41] M. Nagakalyan *et al.*, "Energy efficient approximate bfloat-16 floating point division hardware for image processing," in *2024 International Conference on Smart Electronics and Communication Systems (ISENSE)*, 2024, pp. 1–5.

[42] A. Stillmaker *et al.*, "Scaling equations for the accurate prediction of cmos device performance from 180nm to 7nm," *Integration*, vol. 58, pp. 74–81, 2017.

[43] H. Kawata *et al.*, "Modeling energy consumption of memory systems," in *2015 Third International Symposium on Computing and Networking (CANDAR)*, 2015, pp. 601–603.

[44] N. Jouppi *et al.*, "Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA '23. New York, NY, USA: Association for Computing Machinery, 2023.

[45] S. Lie, "Cerebras architecture deep dive: First look inside the hardware/software co-design for deep learning," *IEEE Micro*, vol. 43, no. 3, pp. 18–30, 2023.

[46] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[47] L. Gwennap, "Groq rocks neural networks: Startup creates new architecture: One core, 1,000 tops," Jan. 6 2020, microprocessor Report, The Linley Group.

[48] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[49] A. Wang *et al.*, "Glue: A multi-task benchmark and analysis platform for natural language understanding," 2019.