POLITECNICO DI TORINO
Repository ISTITUZIONALE

Stop Wasting your Cache! Bringing Machine Learning into Cache Computing

Original
Stop Wasting your Cache! Bringing Machine Learning into Cache Computing / Petrolo, V., Guella, F., Caon, M., Masera, G., Martina, M.. - 2:(2025), pp. 86-89. (22nd ACM International Conference on Computing Frontiers 2025, CF 2025 Cagliari (Ita) 28-30 Maggio 2025) [10.1145/3706594.3726983].

Availability:
This version is available at: 11583/3015603 since: 2026-09-16T10:25:48Z

Publisher:
ACM

Published
DOI:10.1145/3706594.3726983

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript, con Copyr. autore

(Article begins on next page)

29 November 2026

Stop Wasting your Cache! Bringing Machine Learning into Cache Computing

Vincenzo Petrolo
Politecnico di Torino
Turin, Italy
vincenzo.petrolo@polito.it

Flavia Guella
Politecnico di Torino
Turin, Italy
flavia.guella@polito.it

Michele Caon
École polytechnique fédérale de Lausanne
Lausanne, Switzerland
michele.caon@epfl.ch

Guido Masera
Politecnico di Torino
Turin, Italy
guido.masera@polito.it

Maurizio Martina
Politecnico di Torino
Turin, Italy
maurizio.martina@polito.it

Abstract

The rapid evolution of Machine Learning (ML) workloads, particularly Deep Neural Networks (DNNs) and Transformer-based models, has intensified demands on computing architectures, highlighting the limitations of traditional von Neumann systems due to the memory bottleneck. To address these challenges, this paper investigates the mapping of fundamental ML operations onto ARCANE, a Near-Memory Computing (NMC)-based architecture that integrates Vector Processing Units (VPUs) directly within the data cache. ARCANE offers a flexible ISA-extension (`xmnmc`) abstracting memory management, effectively reducing data movement and enhancing performance. We specifically explore the acceleration capabilities of ARCANE when executing fundamental DNN and Transformer-based operations. Experimental results show that, with a contained area overhead, ARCANE achieves consistent speedups, delivering up to 150× improvement in 2D convolution, 305× in Linear layer, and over 32× in Fused-Weight Self-Attention (FWSA), compared to conventional CPU approaches. These findings underline ARCANE's significant benefits in supporting efficient deployment of edge-oriented ML workloads.

CCS Concepts

• **Computer systems organization** → **Reduced instruction set computing**; • **Hardware**;

Keywords

In-Cache Computing, Near-memory Computing, Deep Neural Networks

ACM Reference Format:
Vincenzo Petrolo, Flavia Guella, Michele Caon, Guido Masera, and Maurizio Martina. 2025. Stop Wasting your Cache! Bringing Machine Learning into Cache Computing. In *22nd ACM International Conference on Computing Frontiers (CF Companion '25), May 28–30, 2025, Cagliari, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3706594.3726983

V. Petrolo and F. Guella contributed equally to this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
CF Companion '25, Cagliari, Italy
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1393-4/25/05
https://doi.org/10.1145/3706594.3726983

1 INTRODUCTION

The rapid evolution of ML-driven applications has placed unprecedented demands on computing architectures. Workloads such as DNNs inference require massive parallel processing and high memory bandwidth to sustain real-time execution. Traditional von Neumann architectures, while powerful, struggle with the "memory wall", where data movement between the processor and memory becomes a bottleneck in both performance and energy efficiency. As ML models, particularly transformers, continue to scale in size and complexity, these inefficiencies become even more pronounced. To address these challenges, Compute-In-Memory (CIM) architectures have emerged as a promising alternative, reducing data movement by performing computations directly within the memory subsystem. While existing In-Memory Computing (IMC) solutions face trade-offs in flexibility, precision, and scalability due to their reliance on specialized technologies, NMC architectures offer a more scalable and practical alternative by leveraging standard digital design flows while bringing processing elements closer to memory.

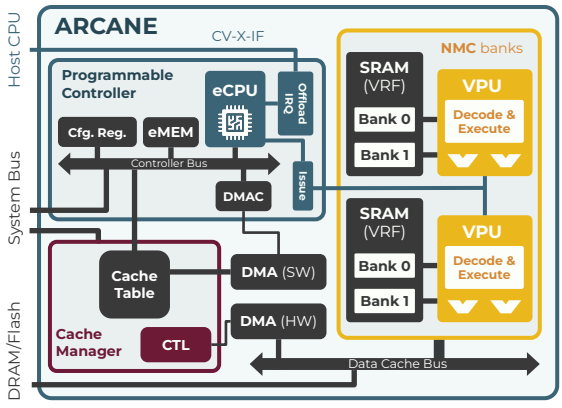


Figure 1: ARCANE block diagram in a two-VPUs, two-lanes configuration

ARCANE [6], an NMC-based cache-integrated computing architecture designed to accelerate massively parallel workloads, aims to bridge the gap between traditional CPU-based processing and the performance benefits of memory-centric computing by embedding VPUs within the data cache.

The architecture enables direct execution of ML primitives within the cache system through its `xmnmc` ISA-extension, avoiding the

need for expensive data transfers between the processor and external accelerators.

In this paper, we demonstrate the performance benefits of offloading ML kernels to an in-cache processing unit as ARCANE obtaining:

- Up to 150× speedup in 2D convolution operations compared to a baseline CPU, with increasing efficiency as input and output channels scale.
- Near 305× sustained speedup in linear layer, significantly outperforming conventional CPU-based Single Instruction Multiple Data (SIMD) execution.
- Over 32× acceleration in Transformer-based FWSA layers for large input sequences, enabling efficient edge deployment of Small Language Models (SLMs).

The remaining sections are organized as follows: Section 2 details ARCANE 's architecture, Section 3 its Application Programming Interface (API), Section 4 presents the logic synthesis results and performance benchmarks, finally Section 5 concludes this work.

ARCANE RTL is available under a permissive license at:

https://github.com/vlsi-lab/ARCANE_OSHW25

2 ARCANE Architecture

The ARCANE in-cache computing paradigm presented in [6] integrates computational functionality within the data cache minimizing high-latency memory operations in data-intensive tasks. The traditional memory subsystem is replaced with a smart data cache interfaced to the system through two slave ports and an external memory connection, as shown in Figure 1. ARCANE architecture employs multiple NM-Carus VPUs [2] managed by an embedded RISC-V core, `cv32e40x`, leveraging a vector-like custom extensions. The data cache maintains a fully associative structure, with cache lines mapped one-by-one to vector registers, optimizing memory coherence. A lightweight Least Recently Used (LRU)-based policy and write-back mechanism handled by the cache controller through a private Direct Memory Access (DMA) enhance performance.

The CV-X-IF interface and software-defined instruction decoding are exploited to offload custom-defined RISC-V macro-instructions to ARCANE, which effectively behaves as a co-processor. During a kernel execution, the region of cache allocated for computation is locked to the host CPU while the remaining portion is accessible for memory operations. Furthermore, a software-driven DMA interacting with the cache controller facilitates operand transfers during kernel execution leveraging optimized 2D transactions, minimizing data movement from the lower-level memory.

3 ML-oriented kernel mapping

The `xmnmc` extension presented in [6] abstracts the complexity of memory management typical in near-memory architectures by introducing custom instructions (`xmr`, `xmk`) to reserve and operate on up to 4-dimensional matrices, a key requirement in ML workloads. It relieves users from manual tiling by providing optimized kernels, two of which—Matrix Multiplication and Conv2D—are presented in this section.

3.0.1 Matrix Multiplication. Algorithm 1 shows the full tiling of a matrix multiplication executed on a single VPU. The matmul between two input tiles is carried out in a row-wise fashion. The output tile dimensions are limited by the physical size of the Vector Register File (VRF). Specifically, the number of columns is constrained by the maximum number of elements a vector can hold, while the number of rows must be split between the second input and the output matrices. The proposed approach prioritizes minimizing memory movement of the output matrix by keeping the output tile within the selected VPU and accumulating partial sums locally. However, this strategy may require reloading an input tile multiple times, as can be observed in Figure 2 (left). This overhead can be mitigated by using multiple VPUs in SIMD fashion and broadcasting shared input tiles.

Algorithm 1 Matrix Multiplication Tiling

```
# C[MxP] = A[MxN] x B[NxP]
for tp in range(0, P, p):        # tiling over C cols
  for tm in range(0, M, m):      # tiling over C rows
    for tn in range(0, N, n):    # tiling over B rows
      DMA_read(A[tm+:m,tn+:n], V31, V31)
      DMA_read(B[tn+:n, tp+:p], V0, V0+n)
      for i in range(tm, min(tm+m, M)):
        for j in range(tn, min(tn+n, N)):
          C[i, tp+:p] += A[i,j] * B[j, tp+:p]
  DMA_write(V15, V15+m, C[tm+:m,tp+:p])
```

3.0.2 2D Convolution. Similarly to the matrix multiplication kernel, the 2D convolution tiling algorithm exploits an output stationary dataflow, minimizing the number of writebacks of the output matrix tile. As can be observed from Figure 2 (right) and Alorithm 2, the additional tiling dimension, compared to the matmul, causes increased computational complexity and memory transfers.

Algorithm 2 Conv2D Tiling

```
# C[Hout,Wout,Cout] = A[Hin,Win,Cin] ⊗ F[Hf,Wf,Cin,Cout]
for tcout in range(0, Cout, 1):
 for tw in range(0, W, w):
  for th in range(0, H, h):
   for tcin in range(0, Cin, 1):
    DMA_read(A[th+:h,tw+:w,tcin], V0, V0+h)
    DMA_read(F[tcin,tcout], V31, V31)
    C[th+:h, tw+:w, tcout] = A[th+:h, tw+:w, tcin] ⊗
                             F[tcin, tcout]
   DMA_write(V20, V20+h, C[th+:h, tw+:w, tcout])
```

4 EXPERIMENTAL RESULTS

This section analyses the results of ARCANE logic synthesis and its performance on a series of edge-orieneted ML benchmarks. Results are obtained encapsulating ARCANE in the eXtendable Heterogeneous Energy-efficient Platform (X-HEEP) [3] system as the L1 data cache. As a comparison, X-HEEP featuring a fully associative 64kB L1-data cache and the `cv32e40px` core [1] with PULP extensions enabled is used.

4.1 Logic Synthesis

The logic synthesis is performed with Synopsys Design Compiler® 2020.09, targeting a 65nm CMOS technology library under worst-case operating conditions. The two systems under analysis feature 4 banks of instruction memory and 2 banks of data cache of 32 kiB each and are synthesized with a clock frequency of 250 MHz. Three configurations are reported for ARCANE and compared to the baseline in Table 1. As expected, the area overhead increases with the number of lanes per VPU. The increase in area is mainly due

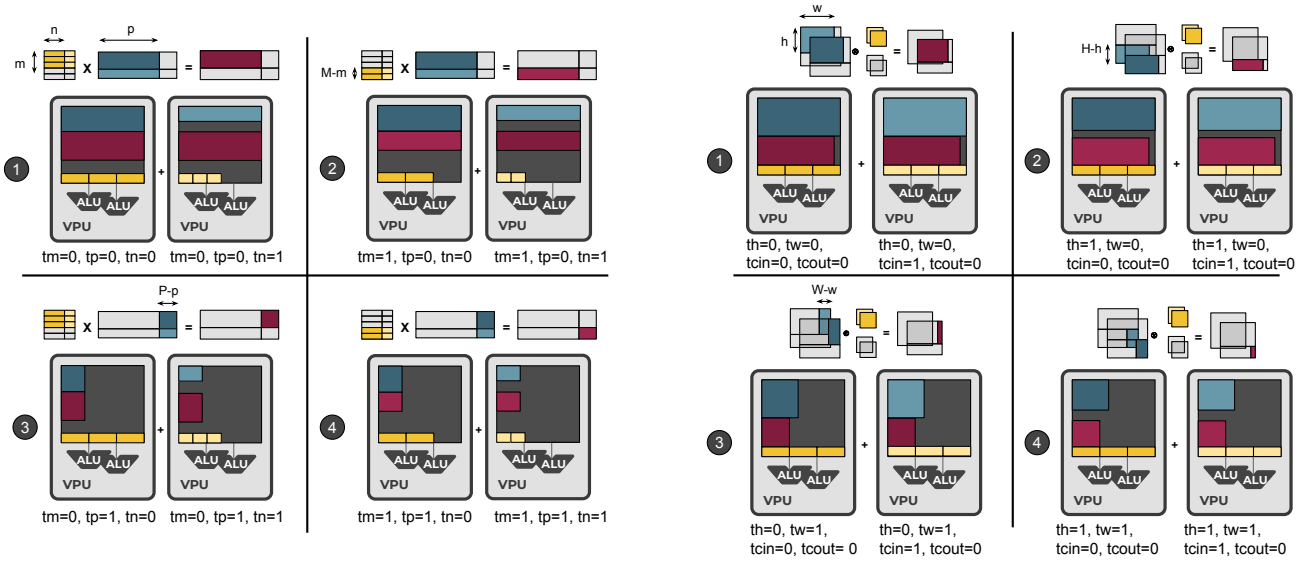


Figure 2: Complete Matmul (left) and single output channel of a Conv2D(right) mapping and tiling on ARCANE VPU.

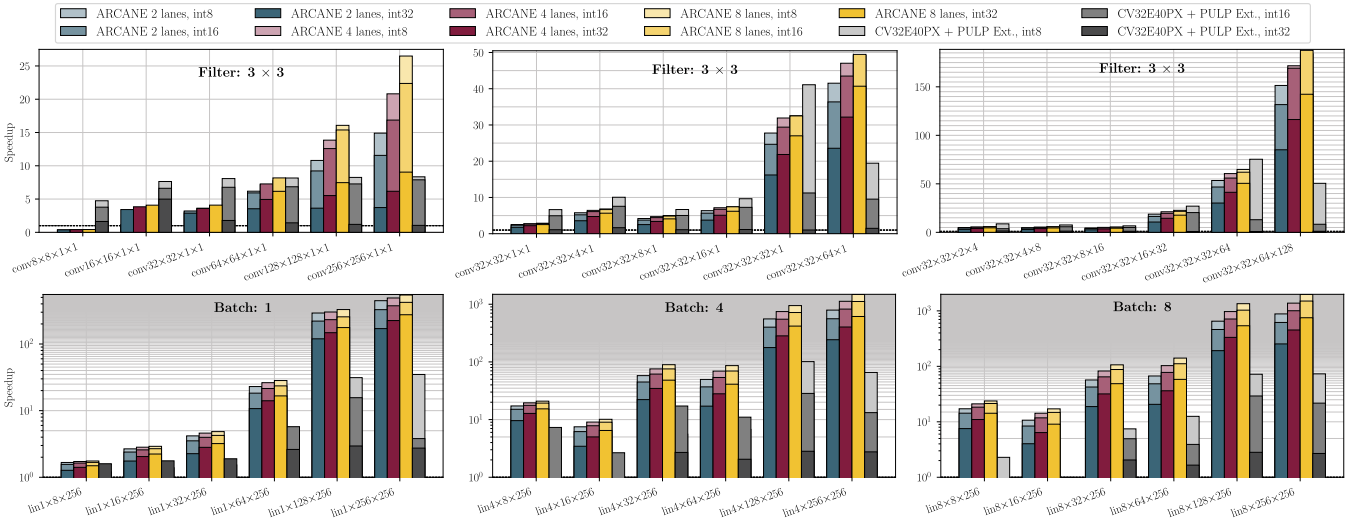


Figure 3: Relative speedup improvements with 2-,4-,8-lane configuration of ARCANE and `cv32e40px` on `int8`, `int16`, `int32` compared to `cv32e40x` when running (Top) 2D Convolution ($convH \times W \times C_{in} \times C_{out}$) with a 3×3 filter and (Bottom) Matrix Multiplication ($linB \times (X_{cols}|W_{rows}) \times W_{cols}$).

to the growing computational and routing complexity, while the contribution of the control logic for concurrent computation and memory operations is negligible.

Table 1: Synthesis results with 64 KiB data cache.

Conf	ARCANE (2 VPUs[a], 2 lanes)	ARCANE (2 VPUs[a], 4 lanes)	ARCANE (2 VPUs[a], 8 lanes)	X-HEEP[3] (2 DMem Banks[a])
Area[μm^2]	2.30×10^6 (+19.9%)	2.38×10^6 (+24.3%)	2.54×10^6 (+32.0%)	1.92×10^6
Area[kGE][b]	1597	1653	1764	1333

[a] 32 KiB each. [b] GE is the 2-input drive strength-one NAND gate equivalent area.

4.2 Benchmarking

We compare ARCANE with the baseline `cv32e40x` core for speedup measurements and the `cv32e40px`, which implements the `xcvpulp` ISA extensions for 8- and 16-bit data, supported by the OpenHW Group toolchain. To evaluate ARCANE 's scalability, we analyze its performance across varying input dimensions.

4.2.1 Convolution. First, we assess the scaling behavior of the 2D convolution kernel (Section 3.0.2). Results (Figure 3 top-left) show the performance of a 3×3 convolution while keeping Input and Output Channel dimensions constant. At smaller input sizes, `cv32e40px` outperforms ARCANE, peaking at a speedup of less than 10× in `int8` data type. Across all ARCANE configurations, the memory allocation overhead initially outweighs the benefits of lane parallelism. However, as input size increases, ARCANE 's speedup grows significantly, surpassing that of `cv32e40px`.

Next, we analyze the impact of input channel scaling while maintaining a fixed 32×32 input size for a fair comparison. As shown

in Figure 3 (top-center), ARCANE consistently achieves a speedup over the baseline core but initially lags behind `cv32e40px`. As the number of input channels increases, `cv32e40px` suffers from frequent cache misses, leading to execution slowdowns and a decline in speedup scaling. In contrast, ARCANE 's optimized data movement sustains a speedup of nearly 50× against the baseline, while `cv32e40px` drops below 20×. Additionally, the limited input size constrains ARCANE 's lane parallelism, resulting in minimal differences between `int16` and `int8` speedup trends.

Finally, we conduct a comprehensive evaluation by scaling both input and output channel dimensions (Figure 3 top-right). In this scenario, the convolution operation requires processing increasingly large data size, with filter complexity $O(C_{in}C_{out}F^2)$. This amplifies cache misses, widening the performance gap between ARCANE and `cv32e40px`. While `cv32e40px` reaches a peak speedup of 75×, ARCANE continues scaling efficiently, exceeding 150×.

4.2.2 Linear Layer. We further analyze the scaling of ARCANE 's speedup when executing a linear layer based on the matrix multiplication described in Section 3.0.1, focusing on the effects of varying batch sizes and input dimensions.

We begin by evaluating the performance of a linear layer using a weight matrix typical of tinyML edge inference scenarios, with a batch size of 1. Results (Figure 3 bottom) indicate that ARCANE outperforms `cv32e40px` as early as the $1 \times 8 \times 8 \times 256$ dimension, demonstrating significant gains across all lane configurations and data types. These gains primarily stem from the efficiency of ARCANE 's DMA transfers and reduced kernel execution time, particularly in the `int8` data type. In contrast, `cv32e40px`'s packed SIMD operations do not yield noticeable benefits until the 1×128×128×256 dimension, where a significant speedup improvement is observed in both `int8` and `int16` formats.

Increasing the batch size (B) further amplifies ARCANE 's speedup, as it processes the matrix multiplication kernel on all the batch as described in Section 3.0.1. Notably, for smaller input dimensions, `cv32e40px` exhibits a negative speedup against the baseline core, likely due to inefficient packed-SIMD execution in this context. At larger input sizes, the expected declining trend in `cv32e40px`'s speedup emerges, driven by increased cache misses that limit its scaling potential.

4.2.3 Fused Weight Attention. The recent rise of SLMs has accelerated efforts to implement Transformer-based architectures on edge devices. In this work, we evaluate ARCANE 's performance on the FWSA layer from the I-BERT Transformer [4]. Specifically, we leverage the FWSA formulation proposed in [5]: $A = \phi(X \cdot W \cdot X^T)$ where A represents the attention matrix, X the input, ϕ the softmax function, and W the fused weights of the Q and K linear projections.

Our implementation employs the matrix multiplication kernel described in Section 3.0.1 and characterized in Section 4.2.2. The $X \cdot W$ is followed by a DMA transfer enabling efficient loading of the transposed input matrix (X^T). A matrix multiplication kernel that processes vector-vector operations on both $(X \cdot W) \cdot X^T$ is then executed. Finally, we integrate an optimized integer Softmax operation from the I-BERT Transformer [4]. We conduct this experiment under worst case conditions (from Figure 3 bottom) using `int32` data type.

To assess scalability, we vary the sequence length (s) from 8 to 256 tokens while proportionally increasing the token embedding dimension (d). Results (Figure 4) indicate that for smaller sequence lengths and embedding dimensions, ARCANE is initially surpassed by `cv32e40px`. However, as both parameters grow, ARCANE 's speedup improves significantly, peaking at over 30×, while `cv32e40px` reaches only 2×. These results highlight ARCANE 's efficiency in handling large-scale matrix operations, making it well-suited for deploying Transformer-based workloads on the edge.

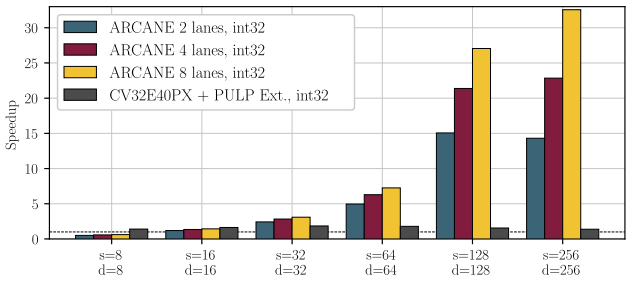


Figure 4: Relative speedup with 2-,4-,8-lane configuration of ARCANE and `cv32e40px` on `int32` compared to `cv32e40x` when running the FWSA layer.

5 CONCLUSIONS

In this work, we demonstrate ARCANE capabilities over ML primitives, showcasing significant performance gains over conventional CPUs in matrix multiplication, convolution, and Fused-Weight Self-Attention (FWSA). ARCANE achieves up to 150× speedup in 2D convolution, 305× in linear layers, and 32× in Transformer-based attention layers while reducing memory movement overhead, optimizing latency and energy efficiency. These results establish ARCANE as a promising solution for real-time inference at the edge. Future work will focus on enhancing SIMD capabilities and instruction set extensions, integrating TinyML frameworks to improve scalability and efficiency while targeting edge ML applications.

References

[1] Michael Gautschi, Pasquale Davide Schiavone, Andreas Traber et al. 2017. Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 10 (2017).

[2] Michele Caon, Clément Choné, Pasquale Davide Schiavone et al. 2024. Scalable and RISC-V Programmable Near-Memory Computing Architectures for Edge Nodes. arXiv:2406.14263 https://arxiv.org/abs/2406.14263

[3] Pasquale Davide Schiavone, Simone Machetti, Miguel Peón-Quirós et al. 2023. X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller. In *Proceedings of the 20th ACM International Conference on Computing Frontiers (CF '23)*. Association for Computing Machinery.

[4] Sehoon Kim, Amir Gholami, Zhewei Yao et al. 2021. I-BERT: Integer-only BERT Quantization. arXiv:2101.01321 https://arxiv.org/abs/2101.01321

[5] Victor Jean-Baptiste Jung, Alessio Burrello, Moritz Scherer et al. 2025. Optimizing the Deployment of Tiny Transformers on Low-Power MCUs. *IEEE Trans. Comput.* 74, 2 (2025).

[6] Vincenzo Petrolo, Flavia Guella, Michele Caon et al. 2025. ARCANE: Adaptive RISC-V Cache Architecture for Near-memory Extensions. https://arxiv.org/abs/2504.02533