3D Analog In-Memory Computing for Efficient Mixture of Experts Large Language Model Inference (Extended Abstract)

Athanasios Vasilopoulos
IBM Research Europe, 8803 Rüschlikon, Switzerland
E-mail: atv@zurich.ibm.com

Full paper: *Büchel, J., Vasilopoulos, A. et al. Efficient scaling of large language models with mixture of experts and 3D analog in-memory computing. Nat Comput Sci 5, 13–26 (2025). https://doi.org/10.1038/s43588-024-00753-x*

Transformer-based Large Language Models (LLMs) have achieved state-of-the-art results in diverse applications, from natural language generation to code synthesis and reasoning. The performance of these models scales predictably with the number of parameters, prompting the development of models with hundreds of billions of parameters. A crucial step towards the democratization of this technology across various services and products involves reducing the hardware cost and the inference cost in terms of time and energy. The ongoing trend of ever-increasing model sizes, however, contradicts that objective since the inflated parameter count incurs an increase in memory footprint, memory transactions and compute latency. Moreover, the primary hardware used for serving such models, notably Graphics Processing Units (GPUs), are power-intensive and not designed for LLM inference. This raises a key challenge: how to reduce the energy demands of LLM inference, both from a hardware and a model architecture perspective.

An algorithmic approach to curb the ever increasing inference cost of LLM inference is embracing the paradigm of conditional computing. In this paradigm, the trajectory of an input through a neural network is dynamically determined, and only a fraction of the parameters is utilized to process the input. These disjoint sets of parameters are referred to as "experts", and models based on this concept are often called Mixture of Experts (MoE) (Fig. 1a). MoEs have recently demonstrated superior performance over dense models, offering a scalable way to increase model capacity without proportionally increasing Floating Point Operations (FLOPs). However, while MoEs reduce FLOP growth, they do not fully address other bottlenecks of LLM inference on conventional Artificial Intelligence (AI) accelerators, such as the need for large memory capacity and frequent data movement between memory and compute units—limitations stemming from their von Neumann architecture (Fig. 1b).

A promising approach to overcome the hardware inefficiencies associated with neural network inference is the paradigm of in-memory computing using non-volatile memory devices, commonly referred as Analog in-Memory Computing (AIMC). AIMC accelerators leverage device characteristics and circuit laws to perform Matrix-Vector Multiplication (MVM) operations within the memory arrays storing the weights, thus eliminating the necessity for weight shuffling, requiring only movement and buffering of activations. Moreover, the operation is performed in the analog domain, resulting in energy-efficient computation with $\mathcal{O}(1)$ time complexity. However, for AIMC to be practical, the weights of the entire LLM must reside on-chip—a requirement unmet by current chip prototypes. The advancement from 2D planar memories to 3D non-volatile memories offers a path forward, potentially allowing entire large LLMs to fit on a single chip (Fig. 1c). Yet, the use of 3D memories introduces new limitations, namely that due to constraints on memory technology or peripheral circuits only a single 2D slice (tier) of the 3D stack can be used at a time, a constraint referred to here as One-Tier-at-a-Time (OTT) (Fig. 1d).

In our study (detailed in the full paper), we combine the conditional computing mechanism of MoEs with the characteristics of 3D AIMC. We show that conditional computing is particularly well-suited to 3D AIMC, as it allows only a subset of the stored parameters to be accessed per inference. This inherently mitigates the OTT constraint by aligning computation with the capacity and selective activation characteristics of 3D AIMC (Fig. 1e). We simulate the execution of dense and MoE-based LLMs in an abstract 3D AIMC system and compare their scaling behavior. Furthermore, we study the impact of noise introduced by the analog hardware on a language modelling task and demonstrate floating point arithmetic-equivalent accuracy (iso-performance) at noise levels comparable to those observed in previously published AIMC hardware.

Our results demonstrate a significant advantage for MoE-based LLM inference on 3D AIMC hardware compared to modern GPUs like the A100. We observe up to a 6× increase in throughput (tokens/s), a 20× improvement in area efficiency (tokens/s/mm^2), and up to a 1000× gain in energy efficiency (tokens/s/W), primarily due to the elimination of weight shuffling. These findings position 3D AIMC as a highly promising post-von Neumann computing paradigm for AI workloads and we aim to motivate further architectural exploration in this direction based on the learnings presented in this study.

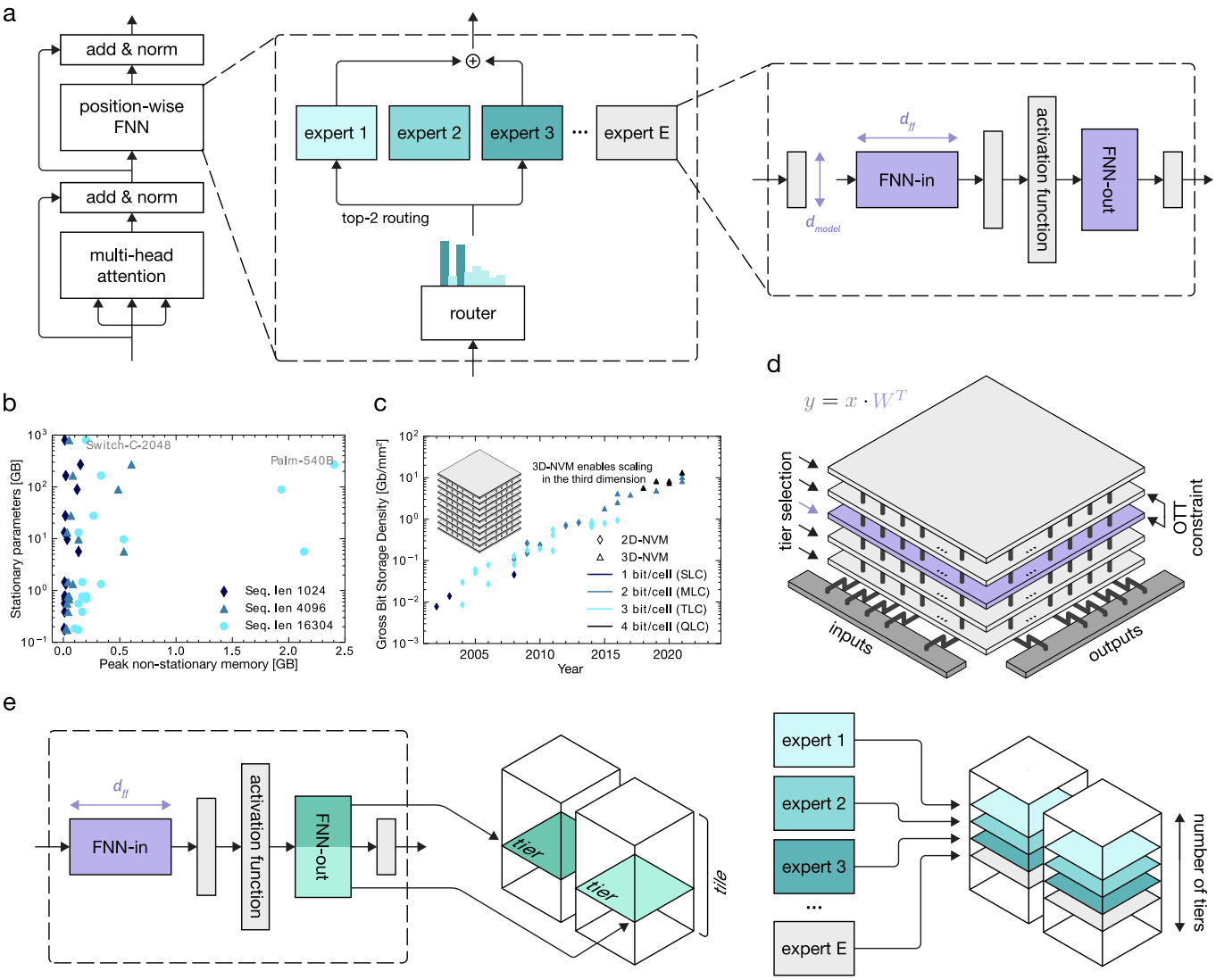

FIG. 1. **3D AIMC-based implementation of MoEs. a.** MoEs typically replace the FFN blocks in conventional Transformers with multiple expert FFNs. In the forward pass, the router first determines for each token the experts that will be used for processing this token. Here, top-2 routing is illustrated, meaning that the router identifies 2 experts that process the token. **b.** As the model sizes of LLMs steadily increase, the memory requirements for stationary parameters (weights) dominate the non-stationary memory requirements (activations). **c.** Illustration of the increase in storage density over the years as a result of stacking dense Non-Volatile Memory (NVM) arrays vertically. **d.** Illustration of an abstract 3D AIMC tile. The dark rectangles depict input and output circuitry. The stacked square boxes indicate the vertically stacked tiers each comprising an analog crossbar. The purple box indicates the activated tier. The black strings illustrate vertical connections. When computing the MVM $y = x \cdot W^T$, the tier holding the weights W is selected and input vectors, x, are fed into the array. The resulting output vectors, y are read out. Performing MVMs in parallel across multiple tiers is often not feasible and we refer to this as the OTT constraint. **e.** The experts, which are typically a two-layer FFN, are stacked on top of each other. In the case where the weight matrix of one of the layers of the FFN exceeds the tier dimension, it must be split into chunks which are distributed across multiple tiles. Note that these chunks can also be stacked on top of each other.