# Memory hierarchy in AI systems: dual contexts, estimation modelling, and optimisation strategies

**Manmohan Brahma[1,*] and G. K. Patra[1,2]**
[1]Academy of Scientific and Innovative Research, Ghaziabad 201 002, India
[2]Council of Scientific and Industrial Research-National Institute of Data Science and AI, Bengaluru 560 037, India

**The rapid growth of large-scale artificial intelligence (AI) models has placed memory management at the centre of both hardware and software design challenges. The present study investigates and introduces the Memory Capacity Estimation Model (MCEM): a parametric, device-agnostic, and precision-aware analytical framework for predicting peak memory utilisation across central processing units, graphics processing units, and neural processing units. MCEM formalises a dual-context memory taxonomy that distinguishes training (bandwidth-bound, write-dominated) from inference (latency-sensitive, key-value cache-dominated) based on dataflow characteristics, temporal write/read ratios, and reuse distance. Validated against open benchmarks from MLPerf, Hugging Face, and advanced micro devices Instinct MI300X hardware reports, MCEM achieves estimation errors of 0.73%–2.7% across five diverse model–device configurations. The framework is shown to reduce memory overprovisioning by 30%–40% relative to naive sizing, with implications for sustainable, cost-effective AI deployment at cloud, high-performance computing, and edge scales.**

**Keywords:** Distributed training optimisation, edge AI and NPU workloads, high-bandwidth memory, key–value cache management, large language model deployment.

bandwidth memory on the package (about 80 to 150 cycles), off-chip DRAM (200 to 400 cycles), and finally, NVMe storage that is thousands of times slower. For large AI models, especially transformer architectures with hundreds of billions of parameters, these latency penalties accumulate rapidly. Each time one trains or runs inference, large volumes of activations, gradients, and state data are continuously transferred, placing significant demands on the memory subsystem[2].

AI workloads are not the same every time. For example, the training operations usually require more memory than available. For instance, the training of a large language model such as the 70B parameter LLaMA can consume 3 terabytes per second of bandwidth just for updating gradients, optimiser states, and gigantic activation tensors[3].

Inference, however, only reads from memory. Inference is sequentially generating tokens, accessing key–value (KV) caches, and trying to meet tight latency budgets (less than 50 ms per token), especially when context windows increase from 128,000 to a million tokens[4]. Speculative decoding[5] analyses the memory overhead of maintaining a draft model alongside the target model during inference; demonstrates 2–3 × latency reduction with a bounded additional memory cost. The memory demand here is huge, and it differs substantially from that in the training process. Training memory is heavily dominated by writing and extensive synchronisation, whereas inference is reading-intensive and limited by the size and speed of the KV cache. It is futile to attempt