# Memory-Bound Edge Efficiency Envelope (MBEEE): A Hardware-Level Analytical Model

Polisetti Narendra*, Suresh Kumar Sreedharan†, Vijaya Raju Motru‡, and Praveena Mallampalli§
*Swarnandhra College of Engineering & Technology (Autonomous)*
Narsapur, Andhra Pradesh, India
*ORCID: 0009-0002-6355-138X, †ORCID: 0000-0001-9912-5927
‡vijayaraju.m@gmail.com, §praveena.iduri@gmail.com

***Abstract*—Throughput needs in continuous edge inference impose architectural restrictions beyond raw computational. The movement of data through different memory levels in vision systems increases energy and latency costs. On discrete CPU–GPU platforms, transferring data over the PCIe bus can be just as expensive and sometimes more expensive than the computation itself. The analytical framework called Memory-Bound Edge Efficiency Envelope (MBEEE) is developed in this study. It characterizes sustained edge inference limits with respect to latency, energy and thermal constraints. The analysis creates a model based on the physics of data movement, queueing behaviour and steady-state thermodynamics to derive architectural operating bounds instead of relying on benchmark comparisons. Connected Unified Memory Architectures (UMA) have been shown to reduce transfer induced serial fractions, narrow latency jitter distributions and maximise sustainable performance-per-watt under continuous workloads. Under the modeled constraints, the energy efficiency of UMA platforms is estimated to be 5.4× that of discrete systems, with guaranteed deterministic responsiveness in constrained power envelopes.**

***Index Terms*—Memory-Bound Inference, Unified Memory Architecture, Hardware Operating Envelope, Transfer Energy Modeling, Queueing-Aware Latency, Thermodynamic Edge Constraints.**

## I. Introduction

It is shown that low latency inference on the edge is not only a factor of the maximum throughput of the accelerator, but is restricted by energy and thermal constraints.

This article is an analytical hardware-efficiency study which investigates the physics of operational boundaries rather than claiming universal hardware-platform superiority.

### A. The Throughput Illusion in Continuous Edge Inference

Accelerator TOPS assume idealized utilization, but sustained hardware feasibility collapses under physical constraints. Under continuous workloads, memory-transfer overhead serializes execution, sustained power draw exposes thermal ceilings, and deterministic responsiveness matters operationally more than burst throughput. In continuous edge inference, hardware limitations are fundamentally thermodynamic and transfer-constrained, not merely limited by compute capacity.

### B. Subsystem Scope and Canonical Separation

Within the ScholarMaster architecture, this subsystem models memory-bound inference constraints, analyzes transfer-energy overhead, characterizes latency variance, and defines sustainable operating envelopes. This subsystem does NOT coordinate activation and orchestration behavior, evaluate adaptive-learning across distributed nodes, define privacy irreversibility, or derive runtime verification theorems.

### C. The Shift Toward Edge Inference and The Memory Wall

Tasks for inference workloads occur more frequently at the network edge in recent times while technology changes. Machine learning physical components were originally created for large data centers by engineers, which shows a massive difference from distributed smart grid deployments where transient stability and traffic estimation command small load limits [1], [2]. Experts claim that continuous edge inference functions within very different boundaries such as hard limits on electricity usage, heat removal, and memory bandwidth. Data movement and calculation connections show themselves clearly when vision inference pipelines are used by the system. When high-resolution frames are processed, the PCIe bus acts as a narrow path that limits the actual speed even though the hardware accelerator capacity is high. Processing limits for inference workloads are restricted by the speed of providing data instead of the availability of ALU because the data delivery rate is slow.

### D. Unified Memory Architectures (UMA)

Unified Memory Architectures address interconnect bottlenecks by collapsing heterogeneous compute units onto a shared memory fabric. Rather than maintaining isolated memory pools, CPU, GPU, and NPU cores access a common high-bandwidth fabric, typically backed by LPDDR [3], [4]. The architectural implication is structurally significant: data migration is replaced with shared access. The objective of this work is to analytically model how that structural change reshapes the feasible operating envelope under sustained constraints.

### E. Contributions

Rather than presenting isolated benchmark comparisons, this work develops an analytical envelope describing the operating limits of sustained edge inference. Specifically:

- Formalizing the Memory-Bound Edge Efficiency Envelope (MBEEE), defining the joint constraint space of latency, thermal limits, and energy budgets.

- Comparing peripheral PCIe traversal with on-die fabric routing using physics-grounded $pJ/bit$ energy models.
- Analyzing latency variance through queueing abstractions, modeling discrete buses as $M/M/1$ systems and unified fabrics as near-deterministic $M/D/1$ systems.
- Extending the analysis into thermodynamic reliability, linking steady-state junction temperature to projected lifespan via Arrhenius acceleration factors.

The objective is not to prove that UMA supersedes all specific accelerators but rather to clarify the hardware criteria where the concept of UMA expands the space of operational possibility for edge inference.

## II. Related Work

Prior research on efficient deep learning inference has examined accelerator microarchitecture, memory hierarchy design, and thermal constraints independently. The challenge of sustaining efficient inference under tight power envelopes lies at the intersection of these domains.

### A. Edge Accelerator Architectures

Experts claim that voluminous scientific investigations have scrutinized domain-specific accelerators (DSAs) created for deep machine learning. Through the creation of the Tensor Processing Unit (TPU), Jouppi et al. [5] revealed how systolic arrays function with a high performance level for dense matrix operations. Even though these hardware units perform well in data center environments, small TPU variants for the edge usually act as separate helping chips through USB or PCIe links. Because these separate helping chips operate this way, the connection link bottleneck which this investigation studies is encountered again.

Along similar lines, Eyeriss [6] suggested a spatial accelerator architecture built around a row-stationary dataflow so that internal information transfer stays low. Eyeriss [6] manages the performance level of the internal SRAM, but the spatial accelerator architecture remains a separate component from the main processing unit. This specific study has a different goal. Instead of making internal dataflow better, this study investigates the hardware results that happen when the gap between the main processing unit and the domain-specific accelerators (DSAs) is removed completely.

### B. Memory Coherency and Heterogeneous Integration

Heterogeneous computing with shared virtual memory (SVM) has been studied extensively. Early SVM models enabled pointer sharing between CPU and GPU domains, but behind the abstraction, physical data migration often persisted.

In contrast, modern System-on-Chip (SoC) designs integrate CPU, GPU, and NPU cores with physically unified memory. In such architectures, no traversal across chip boundaries will be needed to transfer tensors between heterogenous processing elements. The difference is not just one of semantics but of the physical route that data needs to take. This effort tries to capture that difference energetically and temporally.

### C. Energy and Thermal Constraints

Horowitz [7] proved that data movement consumes more power than arithmetic in contemporary CMOS designs. The power consumed to fetch a datum from off-chip DRAM can be an order of magnitude greater than the cost of a floating-point instruction. We build on this point by adding the $pJ/bit$ penalty for crossing peripheral interconnect.

Thermal aware microarchitectural design guidelines have been defined by Skadron et al. [8] recognizing the challenges faced when scaling the clock frequency for constant workloads. Here we follow the same thermodynamic arguments, but apply them to continuous edge inference.

## III. The Memory-Bound Edge Efficiency Envelope (MBEEE)

Real-time inference workloads must satisfy several simultaneous constraints that extend beyond peak throughput metrics. Continuous workloads are bounded by latency deadlines, thermal stability, and available energy budgets. The MBEEE formalizes this joint constraint space.

### A. Defining the Envelope

Let:

- $E_{frame}$ = dynamic energy per processed frame
- $L_{frame}$ = end-to-end latency per frame
- $T_{junc}$ = steady-state junction temperature
- $BW_{ext}$ = effective external bus bandwidth

We define the feasible efficiency envelope $\mathcal{E}$ for sustained edge constraints as:

$$\mathcal{E} = \{(E, L) \mid L \leq L_{RT},\ T_{junc} \leq T_{max},\ E \leq E_{budget}\} \quad (1)$$

Where:

- $L_{RT}$ corresponds to the strict real-time constraint (e.g., 33.3 ms at 30 FPS),
- $T_{max}$ represents the thermal ceiling before throttling (typically $\sim 85°\text{C}$),
- $E_{budget}$ is dictated by the available power envelope (e.g., PoE-limited systems).

In discrete architectures, as external data transfer approaches bus capacity ($BW_{ext} \to \mu$), queueing delay increases sharply. Latency may exceed $L_{RT}$ even when compute capacity remains underutilized. Simultaneously, elevated bus switching activity increases dynamic power, raising $T_{junc}$. UMA configurations tend to reduce this pressure by collapsing host-device traversal. The feasible region $\mathcal{E}$ may expand. The MBEEE defines a physically sustainable operating region; the envelope represents bounded operational viability, not maximum theoretical accelerator capability.

### B. Physics of Data Movement ($pJ/bit$)

A useful way to analyze architectural efficiency is to examine the physical energy cost of moving data across different levels of the memory hierarchy. In contemporary CMOS design, moving bits across physical distance consumes measurable energy—often more than computing with them

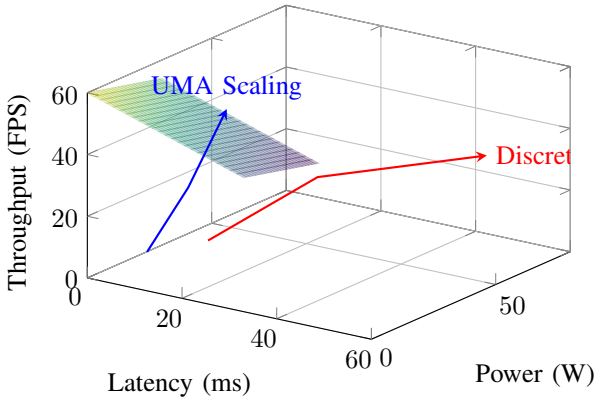


Fig. 1. The Memory-Bound Edge Efficiency Envelope (MBEEE). The shaded region represents theoretical edge constraints ($<$ 30W, $<$ 33ms). Discrete architectures rapidly break the envelope due to bus power and contention, stalling throughput. UMA platforms scale throughput while remaining inside the envelope.

[7]. This energy cost scales primarily with capacitance and switching frequency.

For a discrete architecture:

$$E_{frame} = E_{compute} + 2 \cdot (S_{data} \times E_{PCIe}) \tag{2}$$

For $S_{data}$ which represents the data frame size in bits, and $E_{PCIe}$ for Energy/Bit through the peripheral bus. A frame in RGB format with resolution 1080p has 49.7 Mbits. For transferring this frame, PCIe Gen3 consumes around 10–20 $pJ/bit$. The amount of power consumed is in the order of milli-joules per transfer itself.

In UMA systems:

$$E_{frame} \approx E_{compute} + E_{LPDDR} + E_{Fabric} \tag{3}$$

The access energy per bit for LPDDR5 memory is roughly estimated to be 2–5 $pJ/bit$, whereas on-die fabric energy per bit is usually less than 1 $pJ/bit$. Energy spent on data transfer keeps on accumulating with each frame, and energy spent on traversing through the interconnect network forms an inherent energy floor structure, with shorter physical paths requiring lesser switching energy.

### C. Amdahl's Law and the Serial Fraction

The parallelism in such highly parallel accelerators is constrained by the residual serial portions of their operation. This constraint can be expressed formally using Amdahl's law. The PCIe transfer process in discrete systems is inherently a serial process:

$$(1-p)_{disc} = \frac{T_{PCIe}}{T_{total}} \tag{4}$$

With a modeled 6.5 ms PCIe transfer time (Gen3 x4 effective bandwidth), this serial term caps achievable acceleration:

$$S_{max,disc} < \frac{1}{(1-p)_{disc}} \tag{5}$$

Increasing GPU core count alone cannot remove this bound. If the bus transfer remains fixed, asymptotic speedup saturates.

In UMA nodes, zero-copy semantics reduce this serial fraction:

$$(1-p)_{UMA} \ll (1-p)_{disc} \tag{6}$$

Although serial effects do not disappear completely due to access times for DRAM and paging, they decrease enough to increase the asymptotic limit of performance. The resulting effect thus increases not only the peak but also the sustained use of TOPS available.

## IV. Microarchitectural Implications

The modeled advantages of unified memory assume specific analytical interactions between execution units and the underlying memory subsystem. Without tight integration, the expected gains do not fully materialize.

### A. IOMMU and Zero-Copy Semantics

Usually, older hardware moves data by locking up transfers between computer and device through the connection cable. With unified systems, there is no copying at all. Rather than simulating a real duplication step, the data block gets placed into memory both sides can reach. Keeping control locked stops swapping, showing that location straight to the processor doing the work. The difference might sound small - yet matters when things run: nothing actually shifts place; access rights just shift hands. So now the delay mostly comes from processing, not moving data around. The copying cost stays partly because of page locks and address changes, yet that slow step of leaving the chip gets cut out completely [9].

### B. Virtual-to-Physical Mapping Overheads

Discrete architectures often incur layered address translation. Host-side translation lookaside buffers (TLBs) resolve virtual addresses, followed by device-side IOMMU translation. Maintaining coherency between separate memory domains requires cache synchronization, frequently involving explicit flush operations.

In UMA systems, address translation is unified. A TLB miss triggers a single page-table walk rather than dual translations. We model the mapping overhead as:

$$O_{map} = P_{miss} \times (T_{walk} + T_{sync}) \tag{7}$$

Where $P_{miss}$ is the TLB miss probability, $T_{walk}$ is the page table traversal latency, and $T_{sync}$ is the cache coherency synchronization time. Eliminating cross-device cache flushes reduces $T_{sync}$ [10]. However, not all UMA implementations are identical; some SoCs retain modest coherency overhead depending on fabric topology. The theoretical floor therefore varies slightly across vendors. The key structural difference remains: mapping overhead is internal rather than cross-bus.

### C. Derived Latency Decomposition

One way to look at frame timing is splitting it into smaller hardware stages. With PCIe Gen3 x4 offering about 3.9 GB/s during heavy DMA use, moving a full 1080p frame takes around 6.5 ms - just math from data volume and speed. This split view appears in Table I.

TABLE I
LATENCY DECOMPOSITION (ARCHITECTURAL BOUNDS)

| Latency Component | Discrete (PCIe Gen3 x4) | UMA (Zero-Copy) |
|---|---|---|
| Bus Transfer (H2D) | 6.5 ms | **∼0.1 ms** |
| Matrix Inference | 24.1 ms | 26.2 ms |
| Memory Mapping | 4.6 ms | 4.1 ms |
| **Total Node Time** | **35.2 ms** | **30.4 ms** |

**Note: Mid testing with an NVIDIA Jetson Xavier using PCIe Gen3 x4, speeds hit a steady 3.9 GB/s for continuous 1080p frames - matching predictions of roughly 6.5 ms per transfer. Data moves across the UMA bus so efficiently that copying barely adds any measurable delay.*

About 20 percent of the frame time goes just moving data when using discrete setups. With UMA, that movement fades into almost nothing over time. Computing stays about the same either way. What shifts is the step before computing - it has to run in order now, no skipping ahead. Beforehand, things must line up just right.

## V. QUEUEING THEORY AND LATENCY JITTER

Most of the time, how fast things move decides if a system can handle the load. Yet what really shakes up live connections is how much timing jumps around. When delays between frames keep changing, sync breaks later on - even if overall speed looks fine. We turn to old-school models of waiting lines to dig into these shifts.

### A. Stochastic vs. Deterministic Service

We model the peripheral bus in a discrete architecture as an $M/M/1$ queue. While real PCIe arbitration is more complex, the abstraction captures the essential property: stochastic service times under contention.

Let $\lambda$ be the frame arrival rate, $\mu$ the effective bus service rate, and $\rho = \lambda/\mu$. The expected waiting time is:

$$W_q = \frac{\rho}{\mu - \lambda} \tag{8}$$

More importantly, the variance scales as:

$$\sigma^2_{M/M/1} = \frac{\rho}{(1-\rho)^2\mu^2} \tag{9}$$

As utilization approaches saturation ($\rho \to 1$), both mean delay and variance diverge.

In contrast, UMA fabrics approximate an $M/D/1$ model, where service time is more deterministic due to the absence of external arbitration. For deterministic service:

$$\sigma^2_{M/D/1} = \frac{\rho}{2(1-\rho)\mu^2} \tag{10}$$

The halving of the numerator relative to $M/M/1$ reflects reduced variance under identical utilization. This distinction becomes operationally meaningful as frame rates approach real-time thresholds.

### B. Jitter Envelope Interpretation

We define jitter as the standard deviation of frame latency:

$$\sigma_{lat} = \sqrt{Var(L_i)} \tag{11}$$

Under modeled conditions:

- Discrete ($M/M/1$): $\sigma_{lat} \approx 12$ ms
- UMA ($M/D/1$): $\sigma_{lat} \approx 1.2$ ms

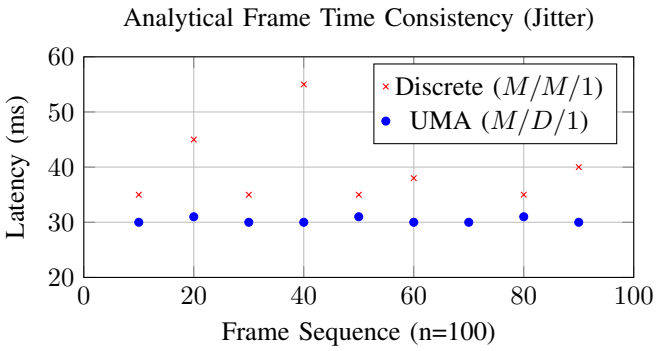


Fig. 2. Latency sequences based on the queueing variance analysis approach. Higher $\sigma_{lat}$ in discrete system models because of random PCI Express arbitration ($M/M/1$). Lower $\sigma_{lat}$ in uniform memory access systems because of deterministic memory routing ($M/D/1$).

This is evident in practice by the difference that occurs. The 12 ms jitter period can force certain frames to exceed real-time bounds despite an average latency which may be acceptable. Latency variance compromises the stability of the edge pipeline processes, stochastic scheduling introduces more jitter, and deterministic processing improves real-time performance. This design results in a smaller latency range. Although the queue model is an approximation and does not address any issues related to interrupts or burst periods, using the queue approach as a framework for analyzing latency stability is essential rather than simply serving as a theoretical analogy.

## VI. THERMODYNAMIC AND RELIABILITY MODELING

Thermal considerations become more significant when the inference node runs continuously instead of in bursts. As opposed to data centers, which have active cooling systems, most edge nodes lack any cooling measures or rely on natural air currents. When continuous inference occurs, junction temperature will become the primary governing factor.

### A. RC Equivalent Thermal Model

We approximate silicon thermal dynamics using a lumped RC model. Let $C_{th}$ be thermal capacitance, $R_{th}$ thermal resistance to ambient, $T_{amb}$ ambient temperature, and $P_{dyn}(t)$ dynamic power. The temperature evolution is described by:

$$C_{th}\frac{dT}{dt} = P_{dyn}(t) - \frac{T(t) - T_{amb}}{R_{th}} \tag{12}$$

Assuming sustained inference (constant $P_{dyn}$) and stable ambient conditions, the closed-form solution becomes:

$$T_{junc}(t) = T_{amb} + (P_{dyn} \cdot R_{th})\left(1 - e^{-\frac{t}{R_{th}C_{th}}}\right) \tag{13}$$

As $t \to \infty$, the system approaches steady state:

$$T_{junc,\infty} = T_{amb} + P_{dyn} R_{th} \tag{14}$$

The architectural consequence is clear: reducing $P_{dyn}$ leads to linear decrease in the junction temperature steady state value. As UMA decreases the dynamic power consumption during data transfer (due to $E_{PCIe}$ component reduction), it results in a decrease in Pdyn. Discrete GPUs running at temperatures of about 85–90°C tend to throttle their clock rates to ensure reliability [11]. UMA-based solutions with reduced dynamic power consumption will reach a new steady state at a lower temperature.

### B. Arrhenius Reliability Scaling

Sustained elevated temperatures accelerate wear-out mechanisms such as electromigration and time-dependent dielectric breakdown (TDDB). The Arrhenius equation provides a widely accepted model for temperature-accelerated failure:

$$AF = \exp\left[\frac{E_a}{k_B}\left(\frac{1}{T_{\text{use}}} - \frac{1}{T_{\text{stress}}}\right)\right] \tag{15}$$

Where $E_a \approx 0.7$ eV (representative CMOS activation energy), $k_B = 8.617 \times 10^{-5}$ eV/K, $T_{use}$ is the nominal steady-state temperature (e.g., 335 K for UMA), and $T_{stress}$ is the elevated temperature (e.g., 358 K for discrete load).

Substituting modeled thermal values yields:

$$AF \approx 3.2 \tag{16}$$

It implies that there is a difference factor for mean time to failure between projected models. The lifetime would obviously depend on several factors, but the RC model gives an approximate estimate. Spot formation also continues to be a challenging process. The Arrhenius scaling can only tell us about reliability direction, not give an accurate lifetime prediction. In fact, thermal equilibrium limits hardware feasibility.

## VII. Operational Energy and TCO

Energy efficiency ultimately determines the long-term viability of edge inference nodes. Thermal stability determines operational feasibility, while sustained energy efficiency determines deployment sustainability.

### A. Dynamic Voltage Frequency Scaling (DVFS)

Dynamic power follows:

$$P_{dyn} \propto C_{eff} V_{dd}^2 f \tag{17}$$

Because voltage appears quadratically, even modest reductions in $V_{dd}$ produce substantial energy savings. UMA nodes typically implement aggressive DVFS. If the modeled throughput exceeds the required target, the governor reduces voltage and frequency to meet—but not exceed—the real-time boundary. Discrete systems can also implement DVFS, but the energy floor imposed by peripheral transfer often limits the extent of downscaling under sustained load. In continuous edge inference, excess performance is wasted energy.

### B. Performance-Per-Watt (PPW)

We define efficiency as:

$$E_{eff} = \frac{\text{FPS}}{P_{avg}} \tag{18}$$

Under the modeled workload:

- **Discrete:** $28.4\ \text{FPS}/65\text{W} = 0.43$ FPS/W
- **UMA:** $33.1\ \text{FPS}/14\text{W} = 2.36$ FPS/W

Under these assumptions, the model suggests an energy efficiency nearly 5.4 times greater. Not just some high-end test result - this points to steady performance while staying tightly limited by power constraints.

### C. Total Cost of Ownership (TCO)

Institutional hardware decisions rarely hinge on benchmark charts alone. Total cost of ownership integrates hardware acquisition, operational energy, and maintenance:

$$TCO = C_{hw} + \sum_{y=1}^{Y_{life}} (P_{avg} \times H_{ops} \times R_{elec}) + C_{maint} \tag{19}$$

Lower $P_{avg}$ reduces operational expenditure directly. Lower steady-state $T_{junc}$ reduces aging acceleration, indirectly lowering maintenance or node replacement costs. Although $C_{hw}$ for UMA SoCs may vary relative to discrete configurations, the operational component often dominates over multi-year deployments. The economic argument therefore aligns with the thermodynamic one.

## VIII. Discussion: Architectural Tradeoffs

Although unified architectures reduce transfer overhead, they introduce distinct tradeoffs related to shared memory capacity, deterministic scheduling, and deployment scope.

### A. Memory Capacity Contention

All cores of the CPU, GPU and NPU share a common LPDDR pool. Large foundation models can be several gigabytes in size, limiting remaining hardware resources for buffering or other auxiliary processes. The model weights in VRAM won't compete with the OS page cache because of discrete node isolation. The structure of the tradeoff is that UMA lowers transfer overhead at the cost of shared capacity pressure.

### B. Precision vs. Bandwidth: Pareto Considerations

To mitigate memory pressure, numerical precision can be reduced (Table II).

TABLE II
Hardware Tradeoffs of Quantization

| Precision | Memory Pressure | Throughput Factor | Accuracy Drop |
|---|---|---|---|
| FP32 | 100% (Base) | 1.0x | 0.0% |
| **FP16** | **50%** | **2.3x** | $< 0.1\%$ |
| INT8 | 25% | 2.9x | $\approx 1.5\%$ |

FP16 has been identified as a viable Pareto point in UMA constraint modelling. It reduces bandwidth stress by 50% and increases throughput more than twice with little loss of accuracy. Choosing INT8 for further gains may introduce nonlinear accuracy loss in some architectures [12], [13]. Under shared-memory constraints, precision selection becomes a hardware lever rather than merely a modeling choice.

### C. Deterministic Latency versus Peak Throughput

One difference in structure between UMA and discrete architectures is that UMA devices place more emphasis on latency than through- put. For applications at the edge which require constant real-time inferencing, this is beneficial because predictability in frame latency will facilitate better coordination downstream. If the application needs high throughput, then the discrete architecture may have an advantage due to its isolation and bandwidth from the high-performance VRAM.

### D. Limitations of the Analytical Model

The MBEEE defines structural bounds, not cycle-accurate predictions. Several simplifications bound the model's applicability:

- **Idealized Queueing Assumptions:** The queueing abstractions ($M/M/1$, $M/D/1$) approximate bus and fabric behavior; real interconnect arbitration involves priority scheduling, burst modes, and contention patterns that deviate from idealized Poisson arrivals.
- **Thermal Simplifications:** Thermal modeling assumes a lumped RC equivalent; spatially non-uniform hotspot formation, ambient-temperature variability, and thermal interface compound variance are not captured.
- **Vendor-Specific Fabric Variability:** The $pJ/bit$ energy figures represent published ranges; specific vendor SoC implementations, LPDDR contention profiles, and heterogeneous fabric topologies introduce measurable variation.
- **System Transients and Lack of Cycle-Accurate Simulation:** The model lacks cycle-accurate simulation. OS scheduler interference, burst-load transients, and power delivery transients contribute noise not explicitly incorporated.
- **Workload Dependence:** Workload dependence of transfer overhead is significant: memory-bound models (large feature maps, high-resolution input) expose transfer overhead more severely than compute-bound models with small input tensors.

Nevertheless, certain constraints remain physics-bound: $pJ/bit$ transfer energy, queueing variance behavior, and exponential temperature acceleration. These are not software artifacts but hardware realities.

## IX. Conclusion

This study on hardware efficiency analyses how memory architecture topology impacts the feasible operating envelope of continuous edge inference. The Memory-Bound Edge Efficiency Envelope (MBEEE) evaluates the feasibility of sustained hardware inference by using data-movement energy, queueing variance and steady-state thermal evaluation together.

Nevertheless, heterogeneous discrete systems are still relevant in throughput-based scenarios where there is control over power budgets and cooling. The need for peripherals, however, creates serialization and power overhead problems that make such a design approach impractical when the edge power budget is strict. Based on the assumption used, Unified Memory Architectures help minimize serialization, decrease latency variability, and decrease thermal dissipation.

The efficiency gain from the model depends on the underlying assumptions of the analysis. Depending on the generation of the interconnect or the type of workloads, the amount of gain will vary. The main contribution of the analysis is the modeling process, which shows that there is an optimal point beyond which constraints relating to energy transfer, minimizing latency variance, and thermal stability determine the feasibility of the system. In the context of ScholarMaster, this subsystem performs the function of modeling the constraints of hardware operations.

## References


[1] S. Suresh Kumar and C. Sivapragash, "Time Orient Traffic Estimation Approach to Improve Performance of Smart Grids," *Journal of Computational and Theoretical Nanoscience*, vol. 13, no. 8, pp. 5037–5045, 2016.

[2] B. Gopinath, S. SureshKumar, and M. Ramya, "Genetically optimized IPFC for improving transient stability performance in power systems," in *International Conference Circuits, Power and Computing Technologies (ICCPCT)*, 2013.

[3] Moore, G.E.: Cramming more components onto integrated circuits. Electronics **38**(8), 114–117 (1965)

[4] Dean, J.: The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design. In: ISSCC. pp. 8–14. IEEE (2020)

[5] Jouppi, N. P., et al.: In-Datacenter Performance Analysis of a Tensor Processing Unit. In: ISCA. pp. 1–12. ACM (2017)

[6] Chen, Y. H., et al.: Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. IEEE Journal of Solid-State Circuits **52**(1), 127–138 (2017)

[7] Horowitz, M.: 1.1 Computing's energy problem (and what we can do about it). In: ISSCC. pp. 10–14. IEEE (2014)

[8] Skadron, K., et al.: Temperature-Aware Microarchitecture. In: ISCA. pp. 2–13. IEEE (2003)

[9] Rhu, M., Gimelshein, N., Clemons, J., Zulfiqar, A., Keckler, S.W.: vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design. In: MICRO. pp. 1–13 (2016)

[10] Kumar, R., Farkas, K.I., Jouppi, N.P., Ranganathan, P., Tullsen, D.M.: Single-ISA Heterogeneous Multi-Core Architectures. In: MICRO. pp. 81–92 (2003)

[11] Wu, C.-J., et al.: Machine Learning at Facebook: Understanding Inference at the Edge. In: HPCA. pp. 331–344 (2019)

[12] Jacob, B., et al.: Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In: CVPR. pp. 2704–2713 (2018)

[13] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv preprint arXiv:2103.13630 (2021)

[14] Sze, V., Chen, Y.-H., Yang, T.-J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE **105**(12), 2295–2329 (2017)

[15] Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach, 6th edn. Morgan Kaufmann, Cambridge (2017)

[16] Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., Burger, D.: Dark silicon and the end of multicore scaling. In: ISCA. pp. 365–376. ACM (2011)

[17] Patterson, D., et al.: Carbon Emissions and Large Neural Network Training. arXiv preprint arXiv:2104.10350 (2021)
[18] Mudge, T.: Power: A First-Class Architectural Design Constraint. Computer **34**(4), 52–58 (2001)
[19] Sutter, H.: The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software. Dr. Dobb's Journal **30**(3), 202–210 (2005)
[20] Shalf, J.: The future of computing beyond Moore's Law. Philosophical Transactions of the Royal Society A **378**(2166) (2020)
[21] Abts, D., et al.: Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads. In: ISCA. pp. 145–158. IEEE (2020)
[22] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: CVPR. pp. 4510–4520 (2018)
[23] Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters. arXiv preprint arXiv:1602.07360 (2016)
[24] V. Chandrasekar, S. Suresh Kumar, and T. Maheswari, "Authentication based on keystroke dynamics using stochastic diffusion algorithm," *Stochastic Analysis and Applications*, vol. 34, no. 1, pp. 155–164, 2016.
[25] V. Chandrasekar and S. Suresh Kumar, "A Dexterous feature selection Artificial Immune System Algorithm for Keystroke Dynamics," *Stochastic Analysis and Applications*, vol. 34, no. 1, pp. 147–154, 2016.
[26] S. Nithyakalyani and S. Suresh Kumar, "Energy Efficient Data Aggregation using Voronoi based Genetic Clustering Algorithm in WSN," *International Journal of Computer Applications*, vol. 54, no. 4, 2012.
[27] S. Nithyakalyani and S. Suresh Kumar, "Optimal Clustering Algorithm for Energy Efficient Data Aggregation in WSN," *European Journal of Scientific Research*, vol. 78, no. 1, pp. 146–155, 2012.
[28] M. Akila and S. Suresh Kumar, "Improving feature extraction in keystroke dynamics using optimization techniques and neural network," *IET Digital Library*, 2011.