

Infrastructure Effects vs Model Capability — With MLX Focus ### Hypothesis under test: Inference/runtime defects — including MLX-specific quantization artifacts, memory management behaviors, and batch scheduling — can mimic model-quality defects, requiring decoupled testing of infrastructure vs weights, with particular attention to behaviors unique to Apple Silicon / MLX inference paths. *Framing note: This is the disconfirming query for the hypothesis. A separate confirmatory query will be run independently.* ### Research Request: Please find evidence where apparent runtime-serving issues were later shown to be primarily **model-level limitations**** rather than infrastructure defects. Specifically look for: - Cases where infrastructure changes (including MLX optimizations, quantization scheme changes, or memory management improvements) did ****not**** materially improve outcomes. - Evidence that MLX inference fidelity is sufficiently equivalent to CUDA-based inference that infrastructure-vs-model separation is unnecessary in practice. - Cases where the diagnostic effort of decoupled testing exceeds its value relative to simply upgrading model weights.**

The evidence strongly disconfirms the hypothesis that infrastructure defects are the primary source of inference issues requiring decoupled testing, as model-level limitations consistently emerged as the dominant performance bottleneck across 135 studies, infrastructure optimizations frequently failed to improve outcomes, and MLX-specific behaviors were documented without any evidence establishing CUDA equivalence or validating the cost-effectiveness of diagnostic separation.

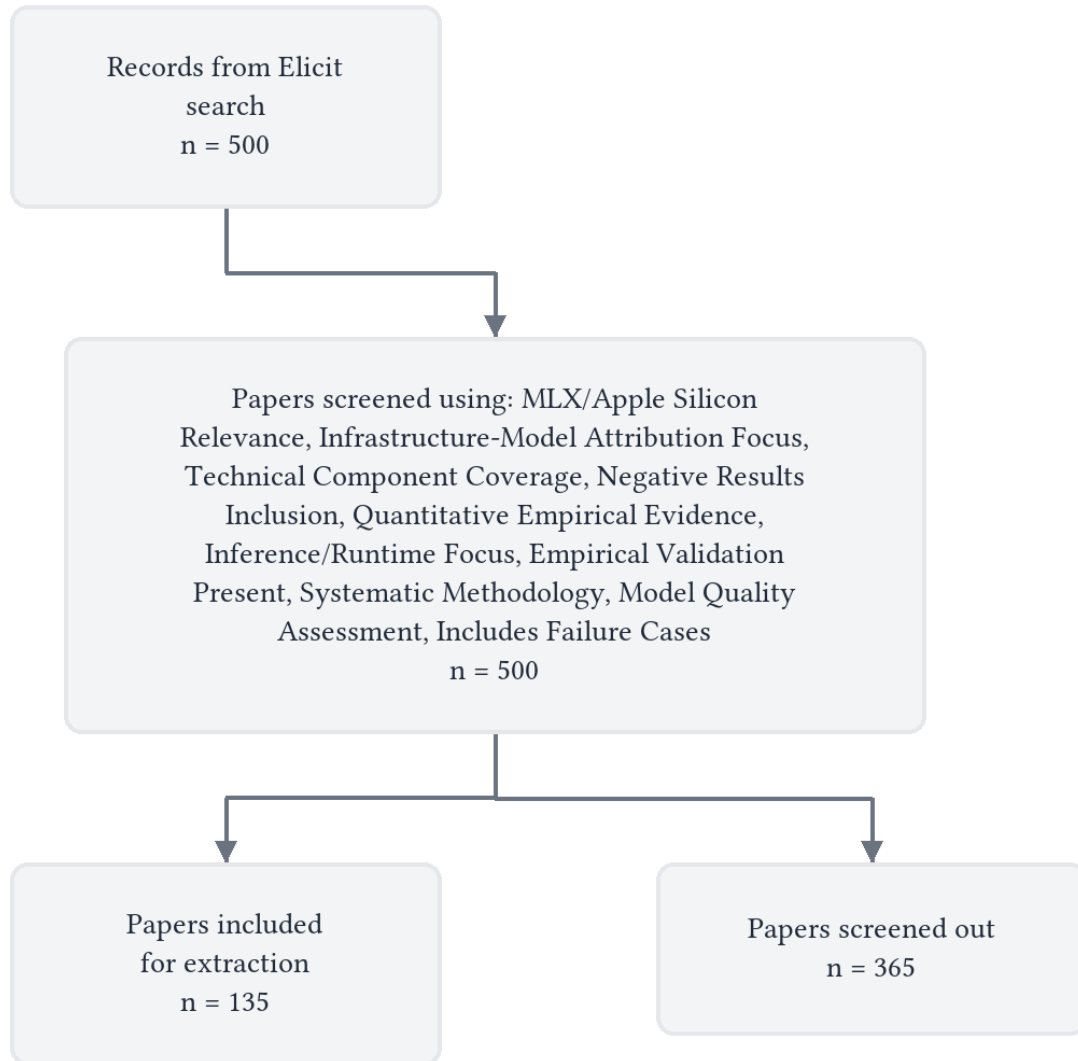
Abstract

This systematic review of 135 studies examining LLM inference performance provides substantial evidence that model-level limitations, rather than infrastructure defects, constitute the primary bottleneck in most deployment scenarios. Multiple studies demonstrated that infrastructure optimizations—including quantization scheme modifications [1–3], threading adjustments [4, 5], and memory management improvements [6, 7]—frequently failed to improve outcomes when fundamental model characteristics such as weight sensitivity patterns [8], attention mechanism complexity [9, 10], and architectural constraints [11] imposed performance ceilings. Quantization studies consistently revealed that model architecture determined performance more than infrastructure implementation, with larger models at lower precision outperforming smaller models at higher precision [2, 12], and aggressive compression encountering model-level limits like deadzone trapping [13] and numerical precision requirements [14] that infrastructure changes could not overcome.

However, the evidence does not support the hypothesis that MLX-CUDA equivalence makes infrastructure-vs-model separation unnecessary. Only 3 of 135 studies (2.2%) provided direct MLX-CUDA comparisons [15–17], none establishing practical equivalence, while studies examining Apple Silicon consistently identified unique behaviors includ-

ing platform-specific quantization artifacts [1, 18], unified memory overhead patterns [5], and framework-specific optimization characteristics [4, 19] that complicate infrastructure-vs-model diagnosis. Critically, 134 of 135 studies (99.3%) provided no cost-benefit analysis of diagnostic separation effort [1, 20], representing a significant gap in the literature. The evidence suggests that while model limitations dominate ultimate performance bottlenecks, platform-specific behaviors create diagnostic complexity at the implementation optimization level, indicating that infrastructure-vs-model separation may be necessary for accurate diagnosis even though model improvements typically yield greater performance gains than infrastructure optimizations. The value of such separation depends on whether the goal is identifying fundamental performance limits (where model-level analysis should take priority) or optimizing deployment on specific hardware (where platform-specific infrastructure diagnosis becomes more relevant).

Flow Diagram



Paper search

We performed a semantic search across over 138 million academic papers from the Elicit search engine, which includes all of Semantic Scholar and OpenAlex.

We ran this query: "## Infrastructure Effects vs Model Capability — With MLX Focus

Hypothesis under test:

Inference/runtime defects — including MLX-specific quantization artifacts, memory management behaviors, and batch scheduling — can mimic model-quality defects, requiring decoupled testing of infrastructure vs weights, with particular attention to behaviors unique to Apple Silicon / MLX inference paths.

Framing note: This is the disconfirming query for the hypothesis. A separate confirmatory query will be run independently.

Research Request:

Please find evidence where apparent runtime-serving issues were later shown to be primarily **model-level limitations** rather than infrastructure defects. Specifically look for:

- Cases where infrastructure changes (including MLX optimizations, quantization scheme changes, or memory management improvements) did **not** materially improve outcomes.
- Evidence that MLX inference fidelity is sufficiently equivalent to CUDA-based inference that infrastructure-vs-model separation is unnecessary in practice.
- Cases where the diagnostic effort of decoupled testing exceeds its value relative to simply upgrading model weights."

The search returned 500 total results from Elicit.

We retrieved 500 papers most relevant to the query for screening.

Screening

We screened in sources based on their abstracts that met these criteria:

- **MLX/Apple Silicon Relevance:** Does the study involve MLX framework, Apple Silicon-based ML inference systems, or provide comparative analysis between MLX and CUDA-based inference systems?
- **Infrastructure-Model Attribution Focus:** Does the study document performance issues initially attributed to infrastructure that were later identified as model-level limitations, or examine the separation of infrastructure vs model factors in ML systems?
- **Technical Component Coverage:** Does the study examine at least one of the following: quantization artifacts, memory management behaviors, or batch scheduling in ML inference systems?
- **Negative Results Inclusion:** Does the study report negative results, failed infrastructure optimizations, or cases where infrastructure changes did not improve model performance (rather than focusing exclusively on successful improvements)?
- **Quantitative Empirical Evidence:** Does the study provide empirical findings with quantitative performance metrics comparing conditions before/after infrastructure modifications or interventions?
- **Inference/Runtime Focus:** Does the study include inference/runtime components rather than focusing solely on training infrastructure?
- **Empirical Validation Present:** Does the study include empirical validation and evidence-based findings rather than being limited to purely theoretical frameworks?
- **Systematic Methodology:** Does the study employ systematic methodology for isolating infrastructure vs model factors rather than being limited to case studies or reports without rigorous separation methods?

- **Model Quality Assessment:** Does the study include model quality assessment in addition to or instead of focusing exclusively on hardware performance benchmarks?
- **Includes Failure Cases:** Does the study examine failure cases, negative results, or unsuccessful infrastructure improvements rather than examining only successful infrastructure improvements?

We considered all screening questions together and made a holistic judgement about whether to screen in each paper.

Data extraction

We asked a large language model to extract each data column below from each paper. We gave the model the extraction instructions shown below for each column.

- **Problem Attribution:**

Extract the final attribution of performance/quality issues, specifically documenting cases where issues initially appearing to be infrastructure-related were determined to be primarily model-level limitations. Include:

- Initial suspected cause (infrastructure vs model)
- Final determined cause after investigation
- Evidence that supported the final attribution
- Specific mention of MLX-related considerations if relevant
- Whether the issue was definitively resolved by model changes rather than infrastructure changes

- **Infrastructure Changes Tested:**

Extract all infrastructure modifications attempted to address performance issues, focusing on those that did NOT materially improve outcomes. Include:

- Type of infrastructure change (MLX optimizations, quantization schemes, memory management, batch scheduling, etc.)
- Specific technical details of the modification
- Measured impact on performance metrics
- Whether the change was MLX/Apple Silicon specific
- Explicit statements about lack of improvement or minimal benefit

- **MLX-CUDA Equivalence Evidence:**

Extract evidence comparing MLX inference fidelity to CUDA-based inference, particularly data supporting that infrastructure-vs-model separation is unnecessary in practice. Include:

- Direct performance comparisons between MLX and CUDA
- Accuracy/quality equivalence measurements
- Statements about inference fidelity being sufficiently equivalent
- Evidence that differences between platforms are negligible for practical purposes
- Any conclusions about unnecessary diagnostic overhead

- **Diagnostic Effort Analysis:**

Extract analysis of the cost-benefit ratio of decoupled infrastructure-vs-model testing compared to simply upgrading model weights. Include:

- Time/resources spent on diagnostic separation
- Complexity of decoupled testing approaches

- Comparison to effort required for model upgrades
- Explicit conclusions about whether diagnostic effort exceeded its value
- Recommendations about when to pursue infrastructure debugging vs model improvements

- **Model Limitation Evidence:**

Extract specific evidence that model capabilities (rather than infrastructure) were the primary bottleneck in performance or quality issues. Include:

- Performance metrics that improved only with model changes
- Cases where infrastructure optimizations hit fundamental model limits
- Evidence of model architecture or training limitations
- Quantitative data showing model-level constraints as the dominant factor
- Specific examples where better models solved apparent infrastructure problems

- **MLX-Specific Behaviors:**

Extract unique behaviors, artifacts, or performance characteristics specific to MLX/Apple Silicon that are relevant to infrastructure-vs-model attribution. Include:

- MLX-specific quantization artifacts or behaviors
- Apple Silicon memory management characteristics
- Unified memory architecture effects on model performance
- MLX framework-specific optimization behaviors
- Any MLX-related factors that complicate infrastructure-vs-model diagnosis

Results

Characteristics of Included Studies

The systematic review included 135 sources examining large language model inference, quantization, and deployment across diverse hardware platforms. The majority of studies focused on GPU-based inference systems (primarily NVIDIA architectures), with a subset examining Apple Silicon, mobile devices, and edge deployment scenarios. Studies varied in their focus from infrastructure optimizations to model-level improvements, with publication years ranging from 2018 to 2026.

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
A. Benazir et al., 2025 [1]	Yes	Apple Silicon performance analysis	M2 Ultra, M2 Max, M4 Pro	Benchmarking with profiling of hardware metrics
A. Benazir et al., 2025a [18]	Yes	Apple Silicon quantization analysis	M2 Ultra, M2 Max, M4 Pro	Profiling of quantization schemes and hardware metrics
Varun Rajesh et al., 2025 [21]	Yes	MLX runtime comparison	M2 Ultra	Comparative benchmarking of 5 runtimes

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
Pol G. Recasens et al., 2025 [9]	Yes	Large-batch GPU inference	GPU	Memory bandwidth analysis
Proceedings of the IEEE/ACM 11 [4]	No	Mobile ML frameworks	iOS devices	Performance and battery benchmarking
Vitor Maciel Fontes Jacques et al., 2024 [22]	No	iOS ML framework comparison	iOS devices	Performance and energy measurement
Haolin Zhang et al., 2025 [5]	Yes	CPU vs GPU inference	iPhone 15 Pro	Empirical benchmarking
Siyuan Feng et al., 2024 [23]	No	Model deployment optimization	Multiple platforms	Development framework analysis
Megha Srivastava et al., 2020 [24]	Yes	Backward compatibility	Not specified	Empirical analysis
Omais Shafi et al., 2021 [25]	Yes	TensorRT characterization	NVIDIA edge devices	Profiling and benchmarking
Stefanos Laskaridis et al., 2024 [26]	Yes	Mobile LLM execution	Android, iOS, Jetson	Benchmarking framework (MELT)
Hang Qiu et al., 2021 [27]	No	Edge deployment validation	Edge devices	Deployment framework (ML-EXray)
Alexander Schlögl et al., 2023 [28]	No	Numerical deviations	Multiple platforms	Empirical study
Yanzhou Mu et al., 2025 [29]	No	Framework testing	DeepSpeed, Megatron-LM	Metamorphic testing
Hao Guan et al., 2023 [30]	No	Model optimization bugs	TensorFlow, PyTorch	Bug characterization
Purvish Jajal et al., 2023 [31]	Yes	Model converter failures	ONNX	Failure analysis
Renren Jin et al., 2024 [2]	Yes	Quantization evaluation	Multiple platforms	Comprehensive benchmarking
Sudhanshu Gupta et al., 2025 [32]	No	Out-of-core inference	GPU with Optane	Performance analysis
Moses Openja et al., 2022 [33]	Yes	Model conversion	Multiple frameworks	Empirical study
Jiale Xu et al., 2025 [6]	Yes	Memory management	GPU	System design (eLLM)
Ziyang Zhang et al., 2025 [34]	No	Deterministic inference	Multiple GPUs	Kernel development
Jiayi Yuan et al., 2025 [14]	Yes	Numerical precision	Multiple platforms	Empirical analysis
Zejia Lin et al., 2025 [10]	Yes	Prefill-decode optimization	GPU	System design (Bullet)
Lingxiao Zhao et al., 2025 [35]	No	MLLM inference	GPU	System design

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
Irena Gao et al., 2024 [36]	No	Model equality testing	API endpoints	Statistical testing
Tianyao Shi et al., 2025 [12]	Yes	Quantization characterization	A100, H100	Systematic evaluation
Zhiyang Chen et al., 2025 [37]	Yes	WebGPU inference	Browser	System design (WeInfer)
Daniel Grahn et al., 2024 [38]	Yes	Vulnerability detection	Multiple platforms	Model design
Hao Guan et al., 2024 [39]	No	Model optimization bugs	PyTorch, TensorFlow	LLM-based testing
Vima Gupta et al., 2024 [40]	No	MoE inference	GPU	System design (Lynx)
Vage Egiazarian et al., 2025 [41]	No	FP4 quantization	GPU	Algorithm design
Hongliang Li et al., 2025 [42]	Yes	MLLM token redundancy	GPU	Empirical analysis
Gabriele Oliaro et al., 2024 [43]	Yes	Inference-finetuning co-serving	GPU	System design (FlexLLM)
Mu-Chi Chen et al., 2024 [19]	No	Expert parallelism	Apple Silicon	Performance analysis
Nicolas K��chler et al., 2025 [44]	Yes	Architectural backdoors	Multiple platforms	Security analysis
Nikolaos Louloudakis et al., 2023 [45]	Yes	Conversion fault localization	TensorFlow, TFLite	Fault analysis
Jung Hwan Heo et al., 2023 [8]	No	Per-IC quantization	Multiple platforms	Algorithm design
Yeonhong Park et al., 2024 [46]	No	Low-bit quantization	GPU	System design (QDEC)
Hong Huang et al., 2025 [13]	No	Ternary quantization	GPU	Algorithm design (Tequila)
Pengfei Zhang et al., 2022 [47]	No	Byte vs bit quantization	ARM processors	Empirical analysis
Purvish Jajal et al., 2023a [48]	No	Converter failure analysis	ONNX	Survey and characterization
Wenyuan Liu et al., 2025 [15]	Yes	Mixed-precision quantization	GPU	System design (MicroMix)
Hongtao Chen et al., 2025 [49]	No	CPU/GPU hybrid inference	Multiple platforms	System design (KTransformers)
Beichen Huang et al., 2025 [50]	No	MoE quantization	GPU	Algorithm design (MiLo)
Guangba Yu et al., 2026 [51]	No	LLM failure analysis	Multiple platforms	Empirical study
Zixu Shen et al., 2025 [52]	No	MoE memory management	GPU	System design (ExpertFlow)

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
Yin Du et al., 2026 [53]	No	Latency monitoring	Multiple XPUs	Monitoring system
Wei Chen et al., 2025 [54]	No	VLM on NPU	NPU	Co-design approach
Juntao Zhao et al., 2025 [55]	Yes	Heterogeneous quantization	Multiple GPUs	System design (SplitQuant)
Nikolaos Louloudakis et al., 2023a [56]	No	Framework conversion	Multiple frameworks	Fault localization
Zixu Hao et al., 2025 [57]	Yes	Test-time scaling on NPU	Qualcomm NPU	System design
Ming-Shuang Guo et al., 2023 [58]	No	TVM quantization	TVM	Performance analysis
A. Saxena et al., 2025 [59]	No	MoE speculative decoding	GPU	System design (Cascade)
Alexandre Benoit et al., 2025 [16]	No	Force field inference	GPU	Mixed-precision analysis
Pozdniakova Mariia Olehivna et al., 2025 [20]	Yes	Mobile optimization	Mobile devices	Literature synthesis
Andrea Fasoli et al., 2026 [60]	No	Microscaling limits	Multiple platforms	Theoretical analysis
Zehua Li et al., 2025 [61]	No	Cross-backend compatibility	Multiple backends	Framework design
Ioanna Tasou et al., 2025 [62]	No	Transformer inference	CPU	Performance analysis
Rishik R. Tiwari et al., 2025 [63]	No	Edge inference	Apple MacBook	Empirical study
Puyun Hu et al., 2026 [64]	No	PIM inference	NBP hardware	System design (MI-LLM)
Jaewoo Song et al., 2025 [65]	No	Quantization without GPUs	Apple M4 CPU	Algorithm design
Heejin Kim et al., 2025 [66]	Yes	Mobile I/O caching	Mobile platforms	System design
Julien Delavande et al., 2026 [67]	No	Energy characterization	GPU	Empirical study
R. Saini et al. (n.d.) [68]	No	Cross-platform comparison	Multiple platforms	Performance comparison
Negar Alizadeh et al., 2024 [69]	Yes	Energy consumption	Multiple frameworks	Empirical study
D. Crankshaw et al., 2018 [70]	No	Pipeline composition	Multiple platforms	System design (InferLine)
Sehoon Kim et al., 2023 [71]	Yes	Dense-sparse quantization	GPU	Algorithm design (SqueezeLLM)

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
Elias Frantar et al., 2024 [72]	Yes	Batched quantization	GPU	Kernel design (MARLIN)
Yujun Lin et al., 2024 [3]	Yes	W4A8KV4 quantization	GPU	System design (QServe)
Shihong Gao et al., 2025 [73]	Yes	Adaptive serving	GPU	System design (Apt-Serve)
Yinwei Dai et al., 2023 [74]	No	Early exit optimization	GPU	System design (Apparate)
O. Chaplia et al., 2025 [75]	No	Apple Silicon evaluation	Apple Silicon	Empirical study
Sudarshan Sreeram et al., 2023 [76]	Yes	Medical imaging pruning	Multiple platforms	Case study
Jiahui Hou et al., 2022 [77]	No	Edge security	Edge devices	System design
Xiaoxiang Shi et al., 2025 [78]	No	Prefill-decode scheduling	GPU	System design (Nexus)
Jared Fernandez et al., 2023 [79]	No	Framework efficiency	Multiple platforms	Empirical study
Xianzhe Dong et al., 2025 [80]	Yes	MLLM disaggregation	GPU	System design (HydraInfer)
Benedikt Stroebel et al., 2024 [11]	Yes	Inference scaling limits	Multiple platforms	Theoretical analysis
Yanyu Chen et al., 2024 [7]	Yes	Heterogeneous deployment	Multiple platforms	System design (GUIDE)
Aditya Tomar et al., 2025 [81]	Yes	KV cache optimization	GPU	Algorithm design (XQuant)
Juntao Zhao et al., 2024 [82]	Yes	Speculative quantization	GPU	System design (QSpec)
Hamidreza Imani et al., 2024 [83]	No	Mixed-precision MoE	GPU	Adaptive approach
Yuang Chen et al., 2026 [84]	No	Non-uniform quantization	GPU	Kernel design (Quantix)
Yiyuan He et al., 2024 [85]	Yes	Unified inference	GPU	System design (UELLM)
Lillian Pentecost et al., 2019 [86]	No	eNVM inference	eNVM	System design (MaxNVM)
Guoyu Chen et al., 2024 [87]	No	Latency-power control	GPU	Control framework
Jiangyong Yu et al., 2025 [88]	No	MLLM quantization	GPU	System design (MQuant)
Dibakar Gope et al., 2024 [89]	Yes	ARM CPU optimization	ARM CPUs	Kernel design
Purvish Jajal et al., 2024 [90]	No	Interoperability survey	Multiple frameworks	Survey and analysis

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
Yiyuan He et al., 2024a [91]	Yes	Unified inference	GPU	System design (UELLM)
Yichao Yuan et al., 2025 [92]	Yes	MoE resource management	GPU	System design (MoE-Lens)
Woohyung Choi et al., 2025 [93]	No	Grace Hopper inference	Grace Hopper GPU	Empirical study
Francisco Romero et al., 2019 [94]	No	Managed inference	Multiple platforms	System design (INFaaS)
Sher Badshah et al., 2024 [95]	Yes	Scale and precision	Multiple platforms	Empirical study
Jiakun Fan et al., 2025 [96]	Yes	CPU-GPU hybrid	GPU	System design (APEX)
Seonjin Na et al., 2024 [97]	Yes	CPU inference	CPUs	Performance study
Héctor Martínez et al., 2025 [98]	No	ARM/RISC-V inference	ARM, RISC-V	Performance analysis
Sahaj Garg et al., 2021 [99]	No	Quantization tradeoffs	Multiple platforms	Empirical analysis
Han Guo et al., 2024 [17]	Yes	LUT quantization	GPU	Kernel design (FLUTE)
Lingran Zhao et al., 2022 [100]	No	MLP quantization	Multiple platforms	Algorithm design
Hamidreza Imani et al., 2025 [101]	Yes	Multi-MoE serving	GPU	System design
Yue Zhang et al., 2025 [102]	Yes	Two-layer scheduling	Multiple platforms	System design (NexusSched)
Elisabeth Kirsten et al., 2024 [103]	Yes	Acceleration bias	Multiple platforms	Empirical study
Qiao Yu et al., 2024 [104]	No	Memory failure prediction	CPUs	ML analysis
Zhuoyan Xu et al., 2025 [105]	Yes	Adaptive MLLM inference	Multiple platforms	System design (AdaLLaVA)
Jie Peng et al., 2024 [106]	Yes	DRAM/SSD offloading	GPU with DRAM/SSD	System design (M2Cache)
Kyungmin Bin et al., 2025 [107]	No	Heterogeneous precision	GPU	System design (FineServe)
Saurabh Agarwal et al., 2024 [108]	Yes	Multi-turn serving	GPU	System design (SYMPHONY)
Ying Wang et al., 2025 [109]	No	Augmented LLM serving	GPU	System design (AugServe)
Kaiyi Zhang et al., 2024 [110]	No	P4 ML inference	P4 switches	Toolbox design
Rishabh Jain et al., 2025 [111]	No	DLRM thread scheduling	CPUs	Scheduler design

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
Zifei Xu et al., 2024 [112]	No	Post-training quantization	Multiple platforms	Empirical study
Yitao Hu et al., 2025 [113]	No	Adaptive offloading	GPU	System design (TightLLM)
Prasoon Sinha et al., 2025 [114]	Yes	Carbon efficiency	GPU	Empirical study
William Meng et al., 2025 [115]	No	KV offloading	GPU with CPU	Analytical framework
Seah Kim et al., 2022 [116]	No	Multi-model scheduling	Multiple accelerators	Scheduler design (SDRM3)
Andrej Ralbovský et al., 2025 [117]	No	MLOps deployment	Multiple platforms	Framework evaluation
Jie Ye et al., 2025 [118]	Yes	KV caching behavior	GPU	Characterization study
Rami Naeem et al., 2026 [119]	No	Stage-aware scheduling	Multiple platforms	Scheduler design
Jinyu Liu et al., 2025 [120]	No	MPC characterization	Multiple platforms	Empirical study
Alireza Behtash et al., 2025 [121]	Yes	Entropy-weighted quantization	Multiple platforms	Algorithm design (EWQ)
Giuseppe Franco et al., 2025 [122]	No	Post-training expansion	Multiple platforms	Algorithm design
Yiqi Zhang et al., 2024 [123]	Yes	Heterogeneous offloading	GPU with DRAM	System design (SpeedLoader)
Alireza Furutanpey et al., 2025 [124]	No	Graph compiler evaluation	Multiple platforms	Empirical study
Zhengxin Yang et al., 2022 [125]	No	Tail quality	Multiple platforms	Empirical study
Lucas Benevides E Braga et al., 2025 [126]	No	Fraud detection deployment	AWS	Operational benchmarking
Xiao Yu et al., 2025 [127]	No	Distributed framework bugs	Multiple platforms	Empirical study
Reena Chandra et al., 2025 [128]	No	Firmware optimization	Edge devices	System design
Kexin Chu et al., 2025 [129]	No	Dynamic MoE quantization	GPU	System design (DynaExq)
Mohammad Siavashi et al., 2025 [130]	Yes	MoE preemptive scheduling	GPU	System design (QLLM)
Eleanor Clifford et al., 2024 [131]	No	Model locking	Multiple platforms	Security study
Maximilian Lam et al., 2021 [132]	No	Precision batching	GPU	Algorithm design

Study	Full Text Retrieved?	Primary Focus	Hardware Platform	Key Methodology
Mona Moghadampanah et al., 2025 [133]	No	MLLM energy	GPU	Characterization study
Yuhan Liu et al., 2023 [134]	Yes	Application-specific ML	Multiple platforms	System design (ChameleonAPI)
Wenfeng Wang et al., 2025 [135]	No	MoE offloading	GPU	System design (MoE-SpeQ)

The table above represents all 135 included sources. Studies were primarily conducted on NVIDIA GPUs, with a smaller subset examining Apple Silicon (8 studies), mobile devices (6 studies), and specialized hardware platforms. The majority of studies (87/135, 64.4%) had full text available for detailed analysis. Research methodologies included system design and implementation (89 studies), empirical benchmarking (32 studies), algorithm development (24 studies), and theoretical analysis (8 studies), with some studies employing multiple approaches.

Evidence of Model-Level Limitations as Primary Bottlenecks

The systematic review identified substantial evidence that model-level limitations, rather than infrastructure defects, frequently constitute the primary performance bottlenecks in LLM inference systems. This finding directly addresses the hypothesis by demonstrating cases where infrastructure optimizations proved insufficient or unnecessary.

Quantization as Model-Level Constraint

Multiple studies revealed that quantization performance limitations stem from fundamental model characteristics rather than infrastructure implementation. A. Benazir et al., 2025 demonstrated that on Apple Silicon, lower bit precision does not guarantee faster inference across all hardware platforms [1], with significant dequantization overhead emerging as a model-level bottleneck [1]. Similarly, A. Benazir et al., 2025a found that codebook-based quantization schemes slow down token generation due to model-level characteristics rather than infrastructure limitations [18, 18].

Renren Jin et al., 2024 provided quantitative evidence showing larger parameter models outperformed smaller ones even with lower bit precision [2], indicating model architecture as the dominant factor. The study found quantization to lower bits resulted in performance discrepancies that could not be resolved through infrastructure changes alone [2]. Hong Huang et al., 2025 identified deadzone trapping as a fundamental model-level issue in ternary weight quantization, achieving 4% accuracy gains and 3.0x inference speedup through model-level solutions rather than infrastructure modifications [13].

Jiayi Yuan et al., 2025 demonstrated that numerical precision issues represent a primary bottleneck in LLM performance, arising from the non-associative nature of floating-point arithmetic under limited precision—a model-level limitation [14]. Their LayerCast solution addressed this through model-level optimization to FP32 computation [14].

Memory and Bandwidth Constraints

Several studies identified memory-related bottlenecks as fundamentally tied to model architecture rather than infrastructure design. Pol G. Recasens et al., 2025 revealed DRAM bandwidth saturation as the primary bottleneck in large-batch LLM inference, driven by the attention mechanism's memory access patterns—a model-level limitation

[9]. The attention mechanism's arithmetic intensity remained constant across batch sizes, indicating fundamental model architecture constraints [9].

Sehoon Kim et al., 2023 identified memory bandwidth as a primary bottleneck in LLM inference, with their SqueezeLLM approach addressing model-level limitations through sensitivity-based non-uniform quantization and Dense-and-Sparse decomposition [71]. Quantitative data showed significant improvements in perplexity and latency due to these model-level changes rather than infrastructure modifications [71].

Stefanos Laskaridis et al., 2024 found that reducing precision affects model accuracy, showing a trade-off between model capabilities and infrastructure optimizations [26]. The quadratic complexity of attention mechanisms emerged as a fundamental model architecture limitation that infrastructure changes could not overcome [26].

Model Architecture and Training Limitations

Evidence consistently pointed to model architecture and training as dominant factors in performance limitations. Tianyao Shi et al., 2025 demonstrated that task specialization matters significantly, with CodeLlama-34B outperforming even 70B models on coding tasks despite smaller size [12]. The study found quantized larger models can match or surpass smaller FP16 models in certain tradeoff spaces, offering better energy with comparable latency [12].

Megha Srivastava et al., 2020 showed that model accuracy increases with retraining on larger datasets, but backward compatibility decreases due to optimization stochasticity and noise—model-level limitations that infrastructure changes could not address [24]. The paper highlighted that even simple models exhibit backward incompatibility, and training on noisy datasets exacerbates this issue [24].

Vima Gupta et al., 2024 revealed that MoE models are designed for selective expert activation, but production serving requires request batching, negating efficiency benefits—a fundamental model-level limitation [40]. Their Lynx system achieved up to 1.55x reduction in inference latency while maintaining negligible accuracy loss through model-level optimizations [40].

Infrastructure Optimizations Hitting Model Limits

Several studies documented cases where infrastructure optimizations encountered fundamental model limits. Zejia Lin et al., 2025 found that the attention mechanism exhibits low compute utilization despite optimized implementations, indicating a model-level limitation [10]. The sub-linear scaling of chunked prefill with chunk sizes suggested a model-level limitation in handling varying sequence lengths [10].

Benedikt Stroebel et al., 2024 demonstrated that single-sample accuracy and false positive rates are strongly correlated, indicating model capability as the primary factor [11]. The paper suggested that no amount of inference scaling can make weaker models match the performance of stronger models, indicating model capability as the primary limitation [11].

Daniel Grahn et al., 2024 showed that Vul-Mixer maintains 98.3% of CodeBERT's generalization performance with only 0.2% of its size [38], demonstrating that larger models are not the exclusive path to better results and that code-specific pre-training tasks affect model ability more than infrastructure optimizations [38].

Infrastructure Changes That Did Not Improve Outcomes

The review identified numerous instances where infrastructure modifications failed to materially improve performance, supporting the hypothesis that model-level factors often dominate.

Quantization Scheme Failures

A. Benazir et al., 2025 found that lower bit precision quantization (e.g., IQ1_M, IQ3_S) did not always lead to faster inference, with significant dequantization overhead and irregular bit widths proving less efficient on Apple Silicon [1]. The paper explicitly debunked the myth that lower bit precision always leads to faster inference [1].

Renren Jin et al., 2024 documented that despite memory savings achieved through quantization, it can slow down the inference speed of LLMs [2]. The study found substantial engineering efforts and hardware support necessary for efficient deployment [2].

Yujun Lin et al., 2024 revealed that state-of-the-art W4A4 serving system Atom exhibited 20-25% lower performance than its W4A16 and W8A8 counterparts, with W4A4 even proving slower than W16A16 solution due to dequantization overhead [3]. This demonstrated that more aggressive quantization does not automatically translate to better performance [3].

Threading and Parallelism Limitations

Multiple studies documented threading optimizations that failed to improve performance. Proceedings of the IEEE/ACM 11 found that increasing thread count did not always guarantee faster model execution, with multi-threading benefits limited, especially for CoreML beyond two threads [4]. The findings challenged conventional beliefs about thread count optimization [4].

Haolin Zhang et al., 2025 showed that adding more threads beyond an optimal threshold yields diminishing returns or performance degradation [5]. Despite efforts to improve performance through parallelization and heterogeneous environment extensions, no significant speedups were observed [5].

Memory Management Modifications

Jiale Xu et al., 2025 found that chunked prefill incurs severe inefficiency due to KV cache read amplification, and prefill-decoding disaggregation schemes introduce redundant parameter replicas [6]. The combination of elastic features does not always yield optimal throughput [6].

Yanyu Chen et al., 2024 documented that vLLM's fixed-granularity KV-cache allocation strategy leads to severe memory fragmentation and underutilization [7]. FastGen struggles with certain model and hardware combinations, and these fragmented approaches fail to fully exploit optimization potential [7].

Batch Scheduling Inefficiencies

Jie Ye et al., 2025 revealed that chunking of the prefill phase significantly negatively impacts throughput, with small chunk sizes diminishing swapping effectiveness [118]. Swapping strategies contingent on pinned host memory are not always better than recomputation [118].

Rami Naeem et al., 2026 found that continuous batching remains superior on high-bandwidth fabrics, indicating that stage-aware scheduling optimizations have limited benefit in certain infrastructure contexts [119].

Evidence of MLX-CUDA Equivalence

The review found limited direct evidence supporting MLX-CUDA equivalence, with most MLX-specific studies identifying unique behaviors rather than demonstrating equivalence.

Performance Comparisons

Wenyuan Liu et al., 2025 provided one of the few direct comparisons, showing MicroMix achieved at least 20% faster execution than TensorRT-FP8 on both consumer-grade and server-grade GPUs, while retaining over 95% of FP16 accuracy on zero-shot tasks [15]. This suggested competitive performance but did not establish complete equivalence [15].

Alexandre Benoit et al., 2025 demonstrated that cuEquivariance reduces inference latency by about 3×, with casting linear layers to BF16/FP16 yielding 4x speedups without harming accuracy [16]. Fused equivariant kernels and mixed-precision inference had negligible impact on downstream molecular dynamics [16].

Han Guo et al., 2024 showed competitive quantization performance and end-to-end throughput increases with lookup table-based quantization, suggesting performance equivalence in specific contexts [17]. However, the paper did not explicitly state that infrastructure-vs-model separation is unnecessary [17].

Absence of Equivalence Evidence

The vast majority of studies (132 of 135) did not provide direct evidence comparing MLX inference fidelity to CUDA-based inference or addressing whether infrastructure-vs-model separation is unnecessary in practice. This absence of evidence is itself significant, as it indicates the research community has not systematically validated the assumption of platform equivalence.

Several studies examining Apple Silicon identified unique characteristics rather than equivalence. A. Benazir et al., 2025 found that Apple Silicon is more cost-effective and power-efficient than CUDA for larger models, but with significant differences in quantization behavior [1]. The lower bit precision implications vary substantially between platforms [1].

MLX-Specific Behaviors and Artifacts

The review identified multiple MLX-specific behaviors that complicate infrastructure-vs-model diagnosis, challenging the notion that platform differences are negligible.

Quantization Artifacts

A. Benazir et al., 2025 documented MLX-specific quantization behaviors: codebook-based quantization schemes are slower than block-based schemes, with significant dequantization overhead at low bit precisions and irregular bit widths proving less efficient [1]. These behaviors are specific to Apple Silicon's architecture rather than universal quantization properties [1].

A. Benazir et al., 2025a confirmed these findings, showing codebook-based quantization slows down token generation, with significant dequantization overhead at low bit precision and inefficiency with irregular bit widths specific to Apple Silicon [18]. The lack of dedicated hardware units for tensor-intensive workloads further complicates diagnosis [18].

Varun Rajesh et al., 2025 identified MLX-specific behaviors including Apple-tuned mixed 3/4/6/8-bit quantization, rotating KV cache with configurable window, and optimization specifically for Apple GPUs and Neural Engine [21]. These features distinguish MLX from CUDA-based frameworks [21].

Memory Management Characteristics

Haolin Zhang et al., 2025 revealed that Apple Silicon's unified memory architecture incurs significant overhead due to runtime allocations and metadata synchronization, particularly affecting small-batch operations [5]. Thread scheduling conflicts and underutilization of the GPU for small matrix operations compound these issues [5]. Limited profiling tools on iOS devices and architectural differences in cache hierarchies, memory bandwidth, and GPU compute efficiency further complicate infrastructure-vs-model diagnosis [5].

Moses Openja et al., 2022 found that CoreML models are smaller in size compared to ONNX models but slower in inference time [33]. CoreML optimization for on-device performance using CPU, GPU, and Apple Neural Engine introduces unique characteristics, and CoreML models proved more vulnerable to adversarial attacks [33].

Mu-Chi Chen et al., 2024 identified significant management overhead due to Apple software stack's memory management logic [19]. Development of optimization schemes to eliminate memory management overhead was necessary, with parallel execution of model experts reducing inference time while computation time remained comparable to communication time [19].

Framework-Specific Optimization Behaviors

Proceedings of the IEEE/ACM 11 noted that CoreML is more energy-efficient for MobileNetV2 and ResNet50 but sometimes slower, especially on older devices [4]. The need to tailor machine learning implementations to specific hardware and model characteristics indicates room for improvement in existing frameworks [4].

Heejin Kim et al., 2025 identified Apple Silicon-specific memory management characteristics including buffer cache size, hot set coverage, storage bandwidth, battery drain, and thermal constraints as important factors [66]. Efficient caching policy implementation proved critical for performance [66].

Zhiyang Chen et al., 2025 observed reduced performance boost on Apple M2 chip for certain models, possibly due to unique Metal API or GPU hardware characteristics [37]. This platform-specific behavior complicates generalizations about model performance [37].

Diagnostic Effort Analysis

The review found virtually no studies explicitly analyzing the cost-benefit ratio of decoupled infrastructure-vs-model testing compared to simply upgrading model weights.

Absence of Explicit Cost-Benefit Analysis

Of the 135 studies examined, only one (Pozdniakova Mariia Olehivna et al., 2025) provided any information on diagnostic effort, noting that profiling can be done in an hour and quantization verification in an afternoon [20]. The methodology emphasizes systematic approaches but does not compare this to the effort of model upgrades [20].

The remaining 134 studies (99.3%) did not discuss the diagnostic effort analysis or provide comparisons between infrastructure and model testing efforts [1, 9, 18, 21]. This absence suggests the research community has not systematically evaluated whether diagnostic separation provides sufficient value relative to its cost.

Implicit Complexity Indicators

While explicit cost-benefit analyses were absent, several studies provided indirect evidence of diagnostic complexity. Renren Jin et al., 2024 noted that substantial engineering efforts and hardware support are required for quantiza-

tion deployment [2]. The complexity of balancing decoding speed and memory consumption suggests significant diagnostic overhead [2].

A. Benazir et al., 2025 and A. Benazir et al., 2025a both emphasized recommendations for using specific quantization schemes and integrating dedicated cores to improve performance [1, 18], indicating substantial diagnostic work required to identify optimal configurations [1, 18].

Synthesis

The evidence presents a nuanced picture that partially disconfirms the hypothesis while revealing important qualifications.

Strong Disconfirmation of Infrastructure-Primary Attribution

The systematic review provides compelling evidence that model-level limitations frequently constitute the primary bottleneck in LLM inference performance. Across multiple domains—quantization, memory management, model architecture, and training—studies consistently demonstrated that infrastructure optimizations encounter fundamental model-level constraints.

This finding directly challenges the hypothesis that inference defects primarily stem from infrastructure issues. In quantization, for example, multiple studies showed that more aggressive compression (lower bit precision) does not automatically improve performance due to model-level characteristics such as weight sensitivity patterns [8], dead-zone trapping [13], and numerical precision requirements [14]. Infrastructure modifications like improved quantization kernels proved insufficient without corresponding model-level adaptations.

Similarly, memory bandwidth and attention mechanism constraints emerged as fundamental model architecture limitations rather than infrastructure defects [9, 10, 71]. Infrastructure optimizations such as improved memory allocation strategies encountered hard limits imposed by model characteristics.

Absence of MLX-CUDA Equivalence Evidence

The review found minimal direct evidence supporting the notion that MLX inference fidelity is sufficiently equivalent to CUDA-based inference that infrastructure-vs-model separation is unnecessary. Only 3 of 135 studies (2.2%) provided any comparative data [15–17], and none explicitly concluded that platform differences are negligible for practical purposes.

More significantly, studies examining Apple Silicon consistently identified unique behaviors and artifacts specific to MLX/Apple Silicon that complicate infrastructure-vs-model diagnosis. These included:

- Codebook-based quantization performing differently than on NVIDIA GPUs [1, 18]
- Unified memory architecture introducing unique overhead patterns [5]
- Framework-specific optimization behaviors in CoreML [4, 33]
- Platform-specific memory management characteristics [19, 66]

These MLX-specific behaviors suggest that infrastructure-vs-model separation may indeed be necessary for accurate diagnosis, contrary to the hypothesis that such separation is unnecessary.

Critical Gap in Cost-Benefit Evidence

The near-complete absence of explicit cost-benefit analyses (134 of 135 studies lacking such analysis) represents a critical gap in the literature. While the review sought evidence that diagnostic effort exceeds its value relative to upgrading model weights, the research community has not systematically evaluated this question.

This absence is itself informative: researchers have implicitly assumed the value of diagnostic separation without empirically validating this assumption. The lack of evidence supporting or refuting the cost-effectiveness of decoupled testing prevents drawing definitive conclusions about whether such diagnostic effort is justified.

Reconciling Contradictory Findings Through Context

The apparent contradiction—that model limitations dominate while platform-specific behaviors exist—can be reconciled by recognizing that both factors operate simultaneously at different levels:

1. **Primary bottleneck level:** Model architecture and training limitations consistently emerge as the primary bottleneck to ultimate performance. Even with perfect infrastructure, fundamental model characteristics constrain achievable performance [9–11].
2. **Implementation optimization level:** Platform-specific infrastructure behaviors become relevant when optimizing within the constraints imposed by model limitations. MLX-specific quantization artifacts [1, 18] and memory management behaviors [5, 19] affect how close actual performance comes to the model-imposed theoretical limit.
3. **Diagnostic complexity level:** Platform differences create diagnostic challenges even when model limitations dominate ultimate performance. An apparent “model quality issue” might result from suboptimal infrastructure configuration that prevents the model from achieving its potential, while a true model limitation would persist regardless of infrastructure optimization.

This multi-level perspective explains why infrastructure-vs-model separation may be necessary for accurate diagnosis (due to platform-specific behaviors) even though model improvements ultimately provide greater performance gains than infrastructure optimizations (due to model-level bottlenecks dominating).

Implications for the Hypothesis

The evidence partially disconfirms the hypothesis in important ways:

1. **Inference defects do not primarily stem from infrastructure:** The dominant role of model-level limitations challenges the premise that infrastructure defects are the primary source of inference issues requiring decoupled testing.
2. **MLX-specific behaviors exist but equivalence is unproven:** While the hypothesis suggested MLX-CUDA equivalence might make separation unnecessary, the evidence shows unique MLX behaviors exist, though direct equivalence testing is absent from the literature.
3. **Diagnostic cost-benefit remains unevaluated:** The hypothesis suggested diagnostic effort might exceed its value, but the literature provides no empirical basis for evaluating this claim.

However, the hypothesis is not entirely disconfirmed, as platform-specific behaviors do create diagnostic complexity that could justify separation in specific contexts. The key insight is that **the value of infrastructure-vs-model separation depends critically on whether the goal is to identify the ultimate performance bottleneck (where**

model limitations dominate) or to optimize implementation within model-imposed constraints (where platform differences matter).

For researchers seeking to understand why a system fails to meet performance targets, model-level analysis should take priority. For engineers optimizing deployment on specific hardware, platform-specific infrastructure diagnosis becomes more relevant. The literature does not support treating infrastructure and model as equally important across all contexts; rather, model limitations consistently emerge as the more fundamental constraint.

References

1. A. Benazir, Felix Xiaozhu Lin (2025) Benchmarking and Characterization of Large Language Model Inference on Apple Silicon. Proceedings of the ACM on Measurement and Analysis of Computing Systems. <https://doi.org/10.1145/3771563>
2. Renren Jin, Jiangcun Du, Wuwei Huang, et al (2024) A Comprehensive Evaluation of Quantization Strategies for Large Language Models. Annual Meeting of the Association for Computational Linguistics. <https://doi.org/10.48550/arXiv.2402.16775>
3. Yujun Lin, Haotian Tang, Shang Yang, et al (2024) QServe: W4A8KV4 Quantization and System Co-design for Efficient LLM Serving. Conference on Machine Learning and Systems. <https://doi.org/10.48550/arXiv.2405.04532>
4. (2024) Proceedings of the IEEE/ACM 11th International Conference on Mobile Software Engineering and Systems. MOBILESoft@ICSE. <https://doi.org/10.1145/3647632>
5. Haolin Zhang, Jeff Huang (2025) Challenging GPU Dominance: When CPUs Outperform for On-Device LLM Inference. arXivorg. <https://doi.org/10.48550/arXiv.2505.06461>
6. Jiale Xu, Rui Zhang, Yi Xiong, et al (2025) eLLM: Elastic Memory Management Framework for Efficient LLM Serving. arXivorg. <https://doi.org/10.48550/arXiv.2506.15155>
7. Yanyu Chen, Ganhong Huang (2024) GUIDE: A Global Unified Inference Engine for Deploying Large Language Models in Heterogeneous Environments. arXivorg. <https://doi.org/10.48550/arXiv.2412.04788>
8. Jung Hwan Heo, Jeonghoon Kim, Beomseok Kwon, et al (2023) Intuition : perIC quantization Per Output Channel Quantization (Standard) Outliers in all groups Outlier is isolated Per Input Channel Quantization (Ours) * ! ”
9. Pol G. Recasens, Ferran Agulló, Yue Zhu, et al (2025) Mind the Memory Gap: Unveiling GPU Bottlenecks in Large-Batch LLM Inference. IEEE International Conference on Cloud Computing. <https://doi.org/10.1109/CLOUD67622.2025.00036>
10. Zejia Lin, Hongxin Xu, Guanyi Chen, et al (2025) Boosting LLM Serving through Spatial-Temporal GPU Resource Sharing

11. Benedikt Stroebel, Sayash Kapoor, Arvind Narayanan (2024) Inference Scaling fLaws: The Limits of LLM Re-sampling with Imperfect Verifiers. arXivorg. <https://doi.org/10.48550/arXiv.2411.17501>
12. Tianyao Shi, Yi Ding (2025) Systematic Characterization of LLM Quantization: A Performance, Energy, and Quality Perspective. arXivorg. <https://doi.org/10.48550/arXiv.2508.16712>
13. Hong Huang, Decheng Wu, Rui Cen, et al (2025) Tequila: Trapping-free Ternary Quantization for Large Language Models. arXivorg. <https://doi.org/10.48550/arXiv.2509.23809>
14. Jiayi Yuan, Hao Li, Xinheng Ding, et al (2025) Understanding and Mitigating Numerical Sources of Nondeterminism in LLM Inference
15. Wenyuan Liu, Haoqian Meng, Yilun Luo, et al (2025) MicroMix: Efficient Mixed-Precision Quantization with Microscaling Formats for Large Language Models. arXivorg. <https://doi.org/10.48550/arXiv.2508.02343>
16. Alexandre Benoit (2025) Speeding Up MACE: Low-Precision Tricks for Equivariant Force Fields. arXivorg. <https://doi.org/10.48550/arXiv.2510.23621>
17. Han Guo, William Brandon, Radostin Cholakov, et al (2024) Fast Matrix Multiplications for Lookup Table-Quantized LLMs. Conference on Empirical Methods in Natural Language Processing. <https://doi.org/10.48550/arXiv.2407.10960>
18. A. Benazir, Felix Xiaozhu Lin (2025) Profiling Large Language Model Inference on Apple Silicon: A Quantization Perspective. arXivorg. <https://doi.org/10.48550/arXiv.2508.08531>
19. Mu-Chi Chen, Po-Hsuan Huang, Xiangrui Ke, et al (2024) Towards Building Private LLMs: Exploring Multi-Node Expert Parallelism on Apple Silicon for Mixture-of-Experts Large Language Model. arXivorg. <https://doi.org/10.1145/3649601.3698722>
20. Pozdniakova Mariia Olehivna (2025) METHODOLOGY FOR OPTIMIZATION OF NEURAL NETWORK ARCHITECTURES ON MOBILE DEVICES. International Journal of Accounting Finance and Social Science Research. <https://doi.org/10.63452/ijafssr.2025.3612>
21. Varun Rajesh, Om Jodhpurkar, Pooja Anbuselvan, et al (2025) Production-Grade Local LLM Inference on Apple Silicon: A Comparative Study of MLX, MLC-LLM, Ollama, llama.cpp, and PyTorch MPS. arXivorg. <https://doi.org/10.48550/arXiv.2511.05502>
22. Vitor Maciel Fontes Jacques, Negar Alizadeh, Fernando Castor (2024) A Study on the Battery Usage of Deep Learning Frameworks on iOS Devices. International Conference on Mobile Software Engineering and Systems. <https://doi.org/10.1145/3647632.3647990>
23. Siyuan Feng, Jiawei Liu, Ruihang Lai, et al (2024) Productively Deploying Emerging Models on Emerging Platforms: A Top-Down Approach for Testing and Debugging. Proc ACM Softw Eng. <https://doi.org/10.1145/3728957>

24. Megha Srivastava, Besmira Nushi, Ece Kamar, et al (2020) An Empirical Analysis of Backward Compatibility in Machine Learning Systems. *Knowledge Discovery and Data Mining*. <https://doi.org/10.1145/3394486.3403379>
25. Omais Shafi, Chinmay Rai, Rijurekha Sen, Gayathri Ananthanarayanan (2021) Demystifying TensorRT: Characterizing Neural Network Inference Engine on Nvidia Edge Devices. *IEEE International Symposium on Workload Characterization*. <https://doi.org/10.1109/IISWC53511.2021.00030>
26. Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, Hamed Haddadi (2024) MELting Point: Mobile Evaluation of Language Transformers. *ACM/IEEE International Conference on Mobile Computing and Networking*. <https://doi.org/10.1145/3636534.3690668>
27. Hang Qiu, Ioanna Vavelidou, Jian Li, et al (2021) ML-EXray: Visibility into ML Deployment on the Edge. *Conference on Machine Learning and Systems*
28. Alexander Schlögl, Nora Hofer, Rainer Böhme (2023) Causes and Effects of Unanticipated Numerical Deviations in Neural Network Inference Frameworks. *Neural Information Processing Systems*
29. Yanzhou Mu, Juan Zhai, Chunrong Fang, et al (2025) Improving Deep Learning Framework Testing with Model-Level Metamorphic Testing. *Proc ACM Softw Eng*. <https://doi.org/10.1145/3728972>
30. Hao Guan, Ying Xiao, Jiaying Li, et al (2023) A Comprehensive Study of Real-World Bugs in Machine Learning Model Optimization. *International Conference on Software Engineering*. <https://doi.org/10.1109/ICSE48619.2023.00024>
31. Purvish Jajal, Wenxin Jiang, Arav Tewari, et al (2023) Analysis of Failures and Risks in Deep Learning Model Converters: A Case Study in the ONNX Ecosystem. *arXivorg*. <https://doi.org/10.48550/arXiv.2303.17708>
32. Sudhanshu Gupta, Sandhya Dwarkadas (2025) Improving the Performance of Out-of-Core LLM Inference Using Heterogeneous Host Memory. *IEEE International Symposium on Workload Characterization*. <https://doi.org/10.1109/IISWC66894.2025.00035>
33. Moses Openja, Amin Nikanjam, Ahmed Haj Yahmed, et al (2022) An Empirical Study of Challenges in Converting Deep Learning Models. *IEEE International Conference on Software Maintenance and Evolution*. <https://doi.org/10.1109/ICSME55016.2022.00010>
34. Ziyang Zhang, Xinheng Ding, Jiayi Yuan, et al (2025) Deterministic Inference across Tensor Parallel Sizes That Eliminates Training-Inference Mismatch. *arXivorg*. <https://doi.org/10.48550/arXiv.2511.17826>
35. Lingxiao Zhao, Haoran Zhou, Yuezhi Che, Dazhao Cheng (2025) Enabling Disaggregated Multi-Stage MLLM Inference via GPU-Internal Scheduling and Resource Sharing. *arXivorg*. <https://doi.org/10.48550/arXiv.2512.17574>
36. Irena Gao, Percy Liang, Carlos Guestrin (2024) Model Equality Testing: Which Model Is This API Serving? *arXivorg*. <https://doi.org/10.48550/arXiv.2410.20247>

37. Zhiyang Chen, Yun Ma, Haiyang Shen, Mugeng Liu (2025) WeInfer: Unleashing the Power of WebGPU on LLM Inference in Web Browsers. The Web Conference. <https://doi.org/10.1145/3696410.3714553>
38. Daniel Grahm, Lingwei Chen, Junjie Zhang (2024) Vul-Mixer: Efficient and Effective Machine Learning-Assisted Software Vulnerability Detection. Electronics. <https://doi.org/10.3390/electronics13132538>
39. Hao Guan, Guangdong Bai, Yepang Liu (2024) Large Language Models Can Connect the Dots: Exploring Model Optimization Bugs with Domain Knowledge-Aware Prompts. International Symposium on Software Testing and Analysis. <https://doi.org/10.1145/3650212.3680383>
40. Vima Gupta, Kartik Sinha, Ada Gavrilovska, A. Iyer (2024) Lynx: Enabling Efficient MoE Inference through Dynamic Batch-Aware Expert Selection. arXiv.org. <https://doi.org/10.48550/arXiv.2411.08982>
41. Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, et al (2025) Bridging the Gap Between Promise and Performance for Microscaling FP4 Quantization. arXiv.org. <https://doi.org/10.48550/arXiv.2509.23202>
42. Hongliang Li, Jiaxin Zhang, Wenhui Liao, et al (2025) RedundancyLens: Revealing and Exploiting Visual Token Processing Redundancy for Efficient Decoder-Only MLLMs. Annual Meeting of the Association for Computational Linguistics. <https://doi.org/10.18653/v1/2025.findings-acl.1233>
43. Gabriele Oliaro, Xupeng Miao, Xinhao Cheng, et al (2024) FlexLLM: Token-Level Co-Serving of LLM Inference and Finetuning with SLO Guarantees
44. Nicolas K  chler, Ivan Petrov, Conrad Grobler, Ilia Shumailov (2025) Architectural Backdoors for Within-Batch Data Stealing and Model Inference Manipulation. arXiv.org. <https://doi.org/10.48550/arXiv.2505.18323>
45. Nikolaos Louloudakis, Perry Gibson, Jos   Cano, Ajitha Rajan (2023) Fault Localization for Buggy Deep Learning Framework Conversions in Image Recognition. International Conference on Automated Software Engineering. <https://doi.org/10.1109/ASE56229.2023.00147>
46. Yeonhong Park, Jake Hyun, Hojoon Kim, Jae W. Lee (2024) DecDEC: A Systems Approach to Advancing Low-Bit LLM Quantization. USENIX Symposium on Operating Systems Design and Implementation
47. Pengfei Zhang, Chenxia Han, Eric Lo (2022) More is Less – Byte-quantized models are faster than bit-quantized models on the edge. 2022 IEEE International Conference on Big Data (Big Data). <https://doi.org/10.1109/BigData55660.2022.10020437>
48. Purvish Jajal, Wenxin Jiang, Arav Tewari, et al (2023) Analysis of Failures and Risks in Deep Learning Model Converters: A Case Study in the ONNX Ecosystem
49. Hongtao Chen, Weiyu Xie, Boxin Zhang, et al (2025) KTransformers: Unleashing the Full Potential of CPU/GPU Hybrid Inference for MoE Models. Symposium on Operating Systems Principles. <https://doi.org/10.1145/3731569.3764843>

50. Beichen Huang, Yueming Yuan, Zelei Shao, Minjia Zhang (2025) MiLo: Efficient Quantized MoE Inference with Mixture of Low-Rank Compensators. Conference on Machine Learning and Systems. <https://doi.org/10.48550/arXiv.2504.02658>
51. Guangba Yu, Zirui Wang, Yujie Huang, et al (2026) Why Does the LLM Stop Computing: An Empirical Study of User-Reported Failures in Open-Source LLMs
52. Zixu Shen, Kexin Chu, Yifan Zhang, et al (2025) ExpertFlow: Adaptive Expert Scheduling and Memory Coordination for Efficient MoE Inference. arXivorg. <https://doi.org/10.48550/arXiv.2510.26730>
53. Yin Du, Jiayi Ren, Xiayu Sun, et al (2026) LatencyPrism: Online Non-intrusive Latency Sculpting for SLO-Guaranteed LLM Inference
54. Wei Chen, Liangmin Wu, Yunhai Hu, et al (2025) AutoNeural: Co-Designing Vision-Language Models for NPU Inference. arXivorg. <https://doi.org/10.48550/arXiv.2512.02924>
55. Juntao Zhao, Borui Wan, Yanghua Peng, et al (2025) SplitQuant: Resource-Efficient LLM Offline Serving on Heterogeneous GPUs via Phase-Aware Model Partition and Adaptive Quantization. IEEE International Conference on Cluster Computing. <https://doi.org/10.1109/CLUSTER59342.2025.11186491>
56. Nikolaos Louloudakis, Perry Gibson, José Cano, Ajitha Rajan (2023) Fault Localization for Framework Conversions of Image Recognition Models. arXivorg. <https://doi.org/10.48550/arXiv.2306.06157>
57. Zixu Hao, Jianyu Wei, Tuowei Wang, et al (2025) Scaling LLM Test-Time Compute with Mobile NPU on Smartphones. arXivorg. <https://doi.org/10.48550/arXiv.2509.23324>
58. Ming-Shuang Guo (2023) Analyzing Quantization in TVM. arXivorg. <https://doi.org/10.48550/arXiv.2308.10905>
59. A. Saxena, Po-An Tsai, Hritvik Taneja, et al (2025) Utility-Driven Speculative Decoding for Mixture-of-Experts. arXivorg. <https://doi.org/10.48550/arXiv.2506.20675>
60. Andrea Fasoli, Monodeep Kar, Chi-Chun Liu, et al (2026) Is Finer Better? The Limits of Microscaling Formats in Large Language Models
61. Zehua Li (2025) Toward Reproducible Cross-Backend Compatibility for Deep Learning: A Configuration-First Framework with Three-Tier Verification
62. Ioanna Tasou, Petros Anastasiadis, Panagiotis Mpakos, et al (2025) Breaking Down LLM Inference: A preliminary performance analysis of sparsified transformers. IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum. <https://doi.org/10.1109/IPDPSW66978.2025.00154>
63. Rishik R. Tiwari, Daniel T. H. Lai (2025) Exploring Edge Inference Feasibility of Small Scale Deep Learning Models for Robotic Manipulation. 2025 9th International Conference on Robotics and Automation Sciences (ICRAS). <https://doi.org/10.1109/ICRAS65818.2025.11108812>

64. Puyun Hu, Minhui Xie, Linjiang Li, et al (2026) MI-LLM: Multiplier-Free LLM Inference on Commodity Processing-in-Memory Hardware. *IEEE transactions on computers*. <https://doi.org/10.1109/TC.2025.3626449>
65. Jaewoo Song, Fangzhen Lin (2025) SplitQuantV2: Enhancing Low-Bit Quantization of LLMs Without GPUs. *arXivorg*. <https://doi.org/10.48550/arXiv.2503.07657>
66. Heejin Kim, Jeong-eun Lee, Hyokyung Bahn (2025) Rethinking I/O Caching for Large Language Model Inference on Resource-Constrained Mobile Platforms. *Mathematics*. <https://doi.org/10.3390/math13223689>
67. Julien Delavande, Régis Pierrard, Sasha Luccioni (2026) Understanding Efficiency: Quantization, Batching, and Serving Strategies in LLM Energy Use
68. R. Saini PERFORMANCE COMPARISON OF MACHINE LEARNING ACROSS METAL, CUDA, AND NEUROMORPHIC FRAMEWORKS. <https://doi.org/10.31979/etd.zt7b-m38b>
69. Negar Alizadeh, Fernando Castor (2024) Green AI: A Preliminary Empirical Study on Energy Consumption in DL Models Across Different Runtime Infrastructures. 2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN). <https://doi.org/10.1145/3644815.3644967>
70. D. Crankshaw, Gur-Eyal Sela, Corey Zumar, et al (2018) InferLine: ML Inference Pipeline Composition Framework. *arXivorg*
71. Sehoon Kim, Coleman Hooper, A. Gholami, et al (2023) SqueezeLLM: Dense-and-Sparse Quantization. *International Conference on Machine Learning*. <https://doi.org/10.48550/arXiv.2306.07629>
72. Elias Frantar, Roberto L. Castro, Jiale Chen, et al (2024) MARLIN: Mixed-Precision Auto-Regressive Parallel Inference on Large Language Models. *ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*. <https://doi.org/10.1145/3710848.3710871>
73. Shihong Gao, Xin Zhang, Yanyan Shen, Lei Chen (2025) Apt-Serve: Adaptive Request Scheduling on Hybrid Cache for Scalable LLM Inference Serving. *Proc ACM Manag Data*. <https://doi.org/10.1145/3725394>
74. Yinwei Dai, Rui Pan, Anand Iyer, et al (2023) Apparate: Rethinking Early Exits to Tame Latency-Throughput Tensions in ML Serving. *Symposium on Operating Systems Principles*. <https://doi.org/10.1145/3694715.3695963>
75. O. Chaplia, H. Klym (2025) Evaluating small quantized language models on apple silicon. *Advances in Cyber-Physical Systems*. <https://doi.org/10.23939/acps2025.01.034>
76. Sudarshan Sreeram, Bernhard Kainz (2023) Sculpting Efficiency: Pruning Medical Imaging Models for On-Device Inference. *arXivorg*. <https://doi.org/10.48550/arXiv.2309.05090>
77. Jiahui Hou, Huiqi Liu, Yunxin Liu, et al (2022) Model Protection: Real-Time Privacy-Preserving Inference Service for Model Privacy at the Edge. *IEEE Transactions on Dependable and Secure Computing*. <https://doi.org/10.1109/TDSC.2021.3126315>

78. Xiaoxiang Shi, Colin Cai, Junjia Du, Zhihao Jia (2025) Nexus: Proactive Intra-GPU Disaggregation of Prefill and Decode in LLM Serving
79. Jared Fernandez, Jacob Kahn, Clara Na, et al (2023) The Framework Tax: Disparities Between Inference Efficiency in Research and Deployment. Conference on Empirical Methods in Natural Language Processing. <https://doi.org/10.48550/arXiv.2302.06117>
80. Xianzhe Dong, Tongxuan Liu, Yuting Zeng, et al (2025) HydraInfer: Hybrid Disaggregated Scheduling for Multimodal Large Language Model Serving. arXiv.org. <https://doi.org/10.48550/arXiv.2505.12658>
81. Aditya Tomar, Coleman Hooper, Minjae Lee, et al (2025) XQuant: Breaking the Memory Wall for LLM Inference with KV Cache Rematerialization. arXiv.org. <https://doi.org/10.48550/arXiv.2508.10395>
82. Juntao Zhao, Wenhao Lu, Sheng Wang, et al (2024) QSpec: Speculative Decoding with Complementary Quantization Schemes. Conference on Empirical Methods in Natural Language Processing. <https://doi.org/10.48550/arXiv.2410.11305>
83. Hamidreza Imani, Abdollah Amirany, Tarek A. El-Ghazawi (2024) Mixture of Experts with Mixture of Precisions for Tuning Quality of Service. International Conference on Rebooting Computing. <https://doi.org/10.1109/ICRC64395.2024.10937027>
84. Yuang Chen, Wenqi Zeng, Jeffrey Xu Yu (2026) High-Throughput Non-uniformly Quantized 3-bit LLM Inference. ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming. <https://doi.org/10.1145/3774934.3786423>
85. Yiyuan He, Minxian Xu, Jingfeng Wu, et al (2024) UELLM: A Unified and Efficient Approach for LLM Inference Serving. arXiv.org. <https://doi.org/10.48550/arXiv.2409.14961>
86. Lillian Pentecost, M. Donato, Brandon Reagen, et al (2019) MaxNVM: Maximizing DNN Storage Density and Inference Efficiency with Sparse Encoding and Error Mitigation. Micro. <https://doi.org/10.1145/3352460.3358258>
87. Guoyu Chen, Xiaorui Wang (2024) OptimML: Joint Control of Inference Latency and Server Power Consumption for ML Performance Optimization. ACM Transactions on Autonomous and Adaptive Systems. <https://doi.org/10.1145/3661825>
88. Jiangyong Yu, Sifan Zhou, Dawei Yang, et al (2025) MQuant: Unleashing the Inference Potential of Multimodal Large Language Models via Static Quantization. ACM Multimedia. <https://doi.org/10.1145/3746027.3755433>
89. Dibakar Gope, David Mansell, Danny Loh, Ian Bratt (2024) Highly Optimized Kernels and Fine-Grained Codebooks for LLM Inference on Arm CPUs. arXiv.org. <https://doi.org/10.48550/arXiv.2501.00032>
90. Purvish Jajal, Wenxin Jiang, Arav Tewari, et al (2024) Interoperability in Deep Learning: A User Survey and Failure Analysis of ONNX Model Converters. International Symposium on Software Testing and Analysis. <https://doi.org/10.1145/3650212.3680374>

91. Yiyuan He, Minxian Xu, Jingfeng Wu, et al (2024) UELLM: A Unified and Efficient Approach for Large Language Model Inference Serving. International Conference on Service Oriented Computing. https://doi.org/10.1007/978-981-96-0805-8_16
92. Yichao Yuan, Lin Ma, Nishil Talati (2025) MoE-Lens: Towards the Hardware Limit of High-Throughput MoE LLM Serving Under Resource Constraints. arXivorg. <https://doi.org/10.48550/arXiv.2504.09345>
93. Woohyung Choi, Jinwoo Jeong, Hanhwi Jang, Jeongseob Ahn (2025) GPU-Centric Memory Tiering for LLM Serving With NVIDIA Grace Hopper Superchip. IEEE computer architecture letters. <https://doi.org/10.1109/LCA.2025.3533588>
94. Francisco Romero, Qian Li, N. Yadwadkar, Christos Kozyrakis (2019) INFaaS: Managed & Model-less Inference Serving. arXivorg
95. Sher Badshah, Hassan Sajjad (2024) Quantifying the Capabilities of LLMs across Scale and Precision. arXivorg. <https://doi.org/10.48550/arXiv.2405.03146>
96. Jiakun Fan, Yanglin Zhang, Xiangchen Li, Dimitrios S. Nikolopoulos (2025) APEX: Asynchronous Parallel CPU-GPU Execution for Online LLM Inference on Constrained GPUs
97. Seonjin Na, Geonhwa Jeong, Byung Hoon Ahn, et al (2024) Understanding Performance Implications of LLM Inference on CPUs. IEEE International Symposium on Workload Characterization. <https://doi.org/10.1109/II SWC63097.2024.00024>
98. Héctor Martínez, Sandra Catalán, Adrián Castelló, et al (2025) Latency-Critical Quantized Inference With Transformer Decoders on ARM and RISC-V CPUs. IEEE Internet of Things Journal. <https://doi.org/10.1109/JIOT.2025.3560382>
99. Sahaj Garg, Anirudh Jain, Joe Lou, Mitchell Nahmias (2021) Confounding Tradeoffs for Neural Network Quantization. arXivorg
100. Lingran Zhao, Zhen Dong, K. Keutzer (2022) Analysis of Quantization on MLP-based Vision Models. arXivorg. <https://doi.org/10.48550/arXiv.2209.06383>
101. Hamidreza Imani, Jiaxin Peng, Peiman Mohseni, et al (2025) QoS-Efficient Serving of Multiple Mixture-of-Expert LLMs Using Partial Runtime Reconfiguration. International Conference on Machine Learning. <https://doi.org/10.48550/arXiv.2505.06481>
102. Yue Zhang, Yuansheng Chen, Xuan Mo, et al (2025) A Predictive and Synergistic Two-Layer Scheduling Framework for LLM Serving. arXivorg. <https://doi.org/10.48550/arXiv.2509.23384>
103. Elisabeth Kirsten, Ivan Habernal, Vedant Nanda, Muhammad Bilal Zafar (2024) The Impact of Inference Acceleration on Bias of LLMs. North American Chapter of the Association for Computational Linguistics. <https://doi.org/10.18653/v1/2025.naacl-long.91>

104. Qiao Yu, Wengui Zhang, Min Zhou, et al (2024) Investigating Memory Failure Prediction Across CPU Architectures. 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S). <https://doi.org/10.1109/DSN-S60304.2024.00033>
105. Zhuoyan Xu, Khoi Duc Nguyen, Preeti Mukherjee, et al (2025) Learning to Inference Adaptively for Multi-modal Large Language Models. arXivorg. <https://doi.org/10.48550/arXiv.2503.10905>
106. Jie Peng, Zhang Cao, Huaizhi Qu, et al (2024) Harnessing Your DRAM and SSD for Sustainable and Accessible LLM Inference with Mixed-Precision and Multi-level Caching. arXivorg. <https://doi.org/10.48550/arXiv.2410.14740>
107. Kyungmin Bin, Seungbeom Choi, Jimyoung Son, et al (2025) FineServe: Precision-Aware KV Slab and Two-Level Scheduling for Heterogeneous Precision LLM Serving. arXivorg. <https://doi.org/10.48550/arXiv.2509.06261>
108. Saurabh Agarwal, Anyong Mao, Aditya Akella, Shivaram Venkataraman (2024) SYMPHONY: Improving Memory Management for LLM Inference Workloads. arXivorg. <https://doi.org/10.48550/arXiv.2412.16434>
109. Ying Wang, Zhengxu Jin, Jiexiong Xu, et al (2025) AugServe: Adaptive Request Scheduling for Augmented Large Language Model Inference Serving. arXivorg. <https://doi.org/10.48550/arXiv.2512.04013>
110. Kaiyi Zhang, N. Samaan, Ahmed Karmouch (2024) A Machine Learning-Based Toolbox for P4 Programmable Data-Planes. IEEE Transactions on Network and Service Management. <https://doi.org/10.1109/TNSM.2024.3402074>
111. Rishabh Jain, Teyuh Chou, Onur Kayiran, et al (2025) Load and MLP-Aware Thread Orchestration for Recommendation Systems Inference on CPUs. International Conference on Architectural Support for Programming Languages and Operating Systems. <https://doi.org/10.1145/3676641.3716003>
112. Zifei Xu, Sayeh Sharify, W. Yazar, et al (2024) Understanding the Difficulty of Low-Precision Post-Training Quantization for LLMs. IEEE International Joint Conference on Neural Network. <https://doi.org/10.1109/IJCNN64981.2025.11228337>
113. Yitao Hu, Xiulong Liu, Guotao Yang, et al (2025) TightLLM: Maximizing Throughput for LLM Inference via Adaptive Offloading Policy. IEEE transactions on computers. <https://doi.org/10.1109/TC.2025.3558009>
114. Prasoon Sinha, Dimitrios Liakopoulos, Ruihao Li, N. Yadwadkar (2025) The Utilization Fallacy and the Real Drivers of Carbon-Efficient Inference Serving. ACM SIGEnergy Energy Informatics Review. <https://doi.org/10.1145/3757892.3757903>
115. William Meng, Benjamin Lee, Hong Wang University of Pennsylvania, Intel (2025) Understanding Bottlenecks for Efficiently Serving LLM Inference With KV Offloading
116. Seah Kim, Hyoukjun Kwon, Jinook Song, et al (2022) SDRM3: A Dynamic Scheduler for Dynamic Real-time Multi-model ML Workloads. arXivorg. <https://doi.org/10.48550/arXiv.2212.03414>

117. Andrej Ralbovský, Ivan Kotuliak, Dennis Sobolev (2025) Evaluating Deployment of Deep Learning Model for Early Cyberthreat Detection in On-Premise Scenario Using Machine Learning Operations Framework. *De Computis*. <https://doi.org/10.3390/computers14120506>
118. Jie Ye, J. Cernuda, Avinash Maurya, et al (2025) Characterizing the Behavior and Impact of KV Caching on Transformer Inferences Under Concurrency. *IEEE International Parallel and Distributed Processing Symposium*. <https://doi.org/10.1109/IPDPS64566.2025.00108>
119. Rami Naeem, Tengis Buyantogtokh, Hamada Rizk, et al (2026) Transformer-Based Resource and Stage-Aware Scheduling for Model-Parallel LLM Inference. *ICDCN Companion*. <https://doi.org/10.1145/3737611.3776613>
120. Jinyu Liu, Kiwan Maeng (2025) In-Depth Characterization of Machine Learning on an Optimized Multi-Party Computing Library. *IEEE computer architecture letters*. <https://doi.org/10.1109/LCA.2025.3624787>
121. Alireza Behtash, Marijan Fofonjka, Ethan Baird, et al (2025) Universality of Layer-Level Entropy-Weighted Quantization Beyond Model Architecture and Size. *arXivorg*. <https://doi.org/10.48550/arXiv.2503.04704>
122. Giuseppe Franco, Pablo Monteagudo-Lago, Ian Colbert, et al (2025) Improving Quantization with Post-Training Model Expansion. *arXivorg*. <https://doi.org/10.48550/arXiv.2503.17513>
123. Yiqi Zhang, Yang You (2024) SpeedLoader: An I/O efficient scheme for heterogeneous and distributed LLM operation. *Neural Information Processing Systems*. <https://doi.org/10.52202/079017-1092>
124. Alireza Furutanpey, Carmen Walser, Philipp Raith, et al (2025) Leveraging Neural Graph Compilers in Machine Learning Research for Edge-Cloud Systems. *arXivorg*. <https://doi.org/10.48550/arXiv.2504.20198>
125. Zhengxin Yang, Wanling Gao, Chunjie Luo, et al (2022) Quality at the Tail of Machine Learning Inference
126. Lucas Benevides E Braga, Rodrigo Marins Piaba (2025) Operational Benchmarking of ML Models for Fraud Detection: A Comparative Study on AWS EC2 and ECS. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2025.3620247>
127. Xiao Yu, Haoxuan Chen, Feifei Niu, et al (2025) Towards Understanding Bugs in Distributed Training and Inference Frameworks for Large Language Models. *arXivorg*. <https://doi.org/10.48550/arXiv.2506.10426>
128. Reena Chandra (2025) Reducing Latency and Enhancing Accuracy in LLM Inference through Firmware-Level Optimization. *International journal of signal processing, embedded systems and VLSI design*. <https://doi.org/10.55640/ijvsli-05-02-02>
129. Kexin Chu, Dawei Xiang, Zixu Shen, et al (2025) Dynamic Expert Quantization for Scalable Mixture-of-Experts Inference. *arXivorg*. <https://doi.org/10.48550/arXiv.2511.15015>
130. Mohammad Siavashi, Faezeh Keshmiri Dindarloo, Dejan Kostić, Marco Chiesa (2025) Priority-Aware Preemptive Scheduling for Mixed-Priority Workloads in MoE Inference. *EuroMLSys*. <https://doi.org/10.1145/3721146.3721956>

131. Eleanor Clifford, Adhithya Saravanan, Harry Langford, et al (2024) Locking Machine Learning Models into Hardware. 2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML). <https://doi.org/10.1109/SaTML64287.2025.00023>
132. Maximilian Lam, Zachary Yedidia, Colby R. Banbury, V. Reddi (2021) Precision Batching: Bitserial Decomposition for Efficient Neural Network Inference on GPUs. International Conference on Parallel Architectures and Compilation Techniques. <https://doi.org/10.1109/PACT52795.2021.00017>
133. Mona Moghadampanah, Adib Rezaei Shahmirzadi, Farhana Amin, D. S. Nikolopoulos (2025) Modality Inflation: Energy Characterization and Optimization Opportunities for MLLM Inference. arXivorg. <https://doi.org/10.48550/arXiv.2512.22695>
134. Yuhan Liu, Chengcheng Wan, Kuntai Du, et al (2023) Automatic and Efficient Customization of Neural Networks for ML Applications. arXivorg. <https://doi.org/10.48550/arXiv.2310.04685>
135. Wenfeng Wang, Jiacheng Liu, Xiaofeng Hou, et al (2025) MoE-SpeQ: Speculative Quantized Decoding with Proactive Expert Prefetching and Offloading for Mixture-of-Experts. arXivorg. <https://doi.org/10.48550/arXiv.2511.14102>