A Survey of High-Level Modeling and Simulation Methods for Modern Machine Learning Workloads

MICRO 2026 Submission – Confidential Draft – Do NOT Distribute!!

Anonymous Author(s)
Under Review
Anonymous

Abstract

As machine learning workloads grow in scale and complexity—spanning training and inference for CNNs, transformers, mixture-of-experts models, and LLMs—architects and system designers need fast, accurate methods to predict their performance across diverse hardware platforms. This survey provides a comprehensive analysis of the tools and methods available for modeling and simulating the performance of ML workloads, covering analytical models, cycle-accurate simulators, trace-driven approaches, and ML-augmented hybrid techniques. We survey approximately 25 tools drawn from 53 papers across architecture venues (MICRO, ISCA, HPCA, ASPLOS) and systems venues (MLSys, OSDI, NSDI) published between 2016–2026, spanning DNN accelerator modeling (Timeloop, MAESTRO, Sparseloop), GPU simulation (GPGPU-Sim, Accel-Sim, NeuSight), distributed training simulation (ASTRA-sim, Lumos, SimAI), and LLM inference serving (VIDUR, Frontier, AMALI). We organize the literature along three dimensions—methodology type (analytical, simulation, ML-augmented, hybrid), target platform (accelerators, GPUs, distributed systems, edge devices), and abstraction level (kernel, model, system)—while additionally characterizing tools by workload coverage, revealing a pervasive CNN-validation bias. Our analysis reveals that hybrid approaches combining analytical structure with learned components achieve the best accuracy-speed trade-offs, while pure analytical models offer superior interpretability for design space exploration. We conduct hands-on reproducibility evaluations of five representative tools, finding that reproducibility varies dramatically: Docker-first tools score 8.5+/10 on our rubric while tools relying on serialized ML models risk becoming unusable. We identify key open challenges including cross-workload generalization beyond CNNs, composition of kernel-level predictions to end-to-end accuracy, and support for emerging architectures. This survey provides practitioners guidance for selecting appropriate modeling tools and researchers a roadmap for advancing the field of ML workload performance prediction.

Keywords

ML workload performance prediction, DNN accelerator modeling, GPU simulation, distributed training simulation, LLM inference serving, design space exploration, survey

1 Introduction

Machine learning workloads—spanning training and inference for CNNs, transformers, mixture-of-experts models, and graph neural networks—have become the dominant consumers of compute across datacenters and edge devices. The shift toward domain-specific architectures [21], from Google's TPU [29, 30] to custom training accelerators, has created a heterogeneous hardware landscape where architects and system designers need fast, accurate performance predictions to navigate vast design spaces, select parallelization strategies, provision serving infrastructure, and optimize hardware-software co-design. Yet ML workloads pose unique modeling challenges: they exhibit diverse computational patterns (dense matrix operations in attention layers, sparse accesses in GNNs, communication-bound collective operations in distributed training) across this increasingly heterogeneous landscape of GPUs, TPUs, custom accelerators, and multi-device clusters.

A rich ecosystem of tools has emerged, spanning analytical models (Timeloop [47], MAESTRO [37]: 5–10% error at microsecond speed), cycle-accurate simulators (GPGPU-Sim [4], Accel-Sim [33]: detailed but hours per workload), trace-driven simulators (ASTRA-sim [67], VIDUR [3]: system-scale training and serving), and ML-augmented hybrid approaches (NeuSight [41]: 2.3% error on GPU kernels). Each methodology occupies a distinct point in the accuracy-speed-generality trade-off space.

Despite this rich tool landscape, no comprehensive survey organizes these methods from the perspective of the ML workload practitioner—the architect or engineer who needs to select a modeling tool for a specific design or deployment task. Existing surveys focus on ML *techniques* for performance modeling [61] or on specific hardware targets [47], leaving practitioners without guidance on which tools suit their needs across the full modeling spectrum. This survey fills that gap by providing a methodology-centric view of the tools and methods available for predicting ML workload performance.

We make the following contributions:

- A **methodology-centric taxonomy** organizing tools along three dimensions: methodology type (analytical, simulation, ML-augmented, hybrid), target platform (DNN accelerators, GPUs, distributed systems, edge devices), and abstraction level (kernel, model, system), with a quantitative coverage matrix identifying research gaps and a workload coverage analysis exposing the CNN-validation bias in the literature.
- A **systematic survey** of over 30 modeling tools drawn from 53 papers across architecture venues (MICRO, ISCA, HPCA, ASPLOS) and systems venues (MLSys, OSDI, NSDI) published between 2016–2026, using documented selection criteria.
- A **comparative analysis** examining trade-offs between accuracy, speed, generalization, and interpretability, with careful qualification of paper-reported accuracy claims and

MICRO 2026, Austin, TX, USA
2026.

identification of cases where reported numbers are unverifiable.
- **Hands-on reproducibility evaluations** of representative tools with a 10-point rubric, and identification of **open challenges** including the CNN-to-transformer generalization gap, kernel-to-end-to-end error composition, and emerging accelerator support.

The paper proceeds as follows: Section 2 describes our methodology; Section 3 provides background; Section 4 presents the taxonomy; Section 5 surveys tools by platform; Section 6 compares accuracy and provides tool selection guidance; Section 7 presents reproducibility evaluations; Section 8 discusses open challenges; and Section 9 concludes.

The field has evolved from early analytical frameworks (Eyeriss [11], Paleo [51]) through systematic accelerator modeling (Timeloop [47], MAESTRO [37]) and distributed training simulation (ASTRA-sim [54]), to modern hybrid analytical+learned approaches (NeuSight [41]) and LLM-specific modeling (VIDUR [3], Frontier [18]); Table 3 provides a comprehensive chronological overview.

2 Survey Methodology

We searched ACM Digital Library, IEEE Xplore, Semantic Scholar, and arXiv using terms related to ML performance modeling, with backward/forward citation tracking from seminal works. Target venues include architecture (MICRO, ISCA, HPCA, ASPLOS), systems (MLSys, OSDI, SOSP, NSDI), and related (NeurIPS, MobiSys, DAC, ISPASS). Papers must propose or evaluate a tool for predicting ML workload performance with quantitative evaluation; we exclude non-performance tasks and general-purpose workloads. From 287 initial candidates, title/abstract screening yielded 118 papers; full-text review reduced the set to 53 that met all criteria, supplemented by 12 foundational works for context. We cover 2016–2026 and classify each paper by *methodology type* (analytical, simulation, trace-driven, ML-augmented, hybrid), *target platform*, and *abstraction level* (kernel, model, system).

2.1 Related Surveys

Prior surveys address adjacent topics: Rakhshanfar and Zarandi [53] survey ML for processor DSE; Sze et al. [62] treat DNN hardware design (the foundation for Timeloop/MAESTRO); GPGPU-Sim [4] and gem5 [6] have extensive evaluation literature; and MLPerf [44, 56] standardizes *measurement* rather than *prediction*. This survey differs by spanning the full methodology spectrum across all major platforms with hands-on reproducibility evaluations. The closest prior work, Dudziak et al. [15], compares edge device predictors for NAS; we broaden to the full landscape.

3 Background

3.1 ML Workload Characteristics

ML workloads, defined as computation graphs in frameworks like PyTorch [49] and TensorFlow [1], present distinct modeling challenges. Their operators (convolutions, matrix multiplications, attention) have statically known shapes amenable to analytical modeling, though mixture-of-experts and dynamic inference introduce input-dependent control flow. Performance is highly sensitive to how tensors map onto specialized memory hierarchies (dataflow, tiling, loop ordering), and for LLM inference, KV cache management dominates memory behavior [38]. At scale, training distributes across thousands of GPUs via data, tensor, pipeline, and expert parallelism [13], requiring system-level modeling of compute–memory–network interactions. LLM inference further splits into compute-bound prefill and memory-bound decode phases [50], both of which must be modeled under batched serving [2, 70].

3.2 Modeling Methodologies

We classify approaches into four categories forming our taxonomy's primary axis. **Analytical models** express performance as closed-form functions—e.g., the roofline model [66] bounds throughput by $P = \min(\pi, \beta \cdot I)$, while Timeloop [47] computes data movement costs for DNN accelerator mappings. They offer microsecond evaluation but require per-architecture derivation. **Cycle-accurate simulators** (gem5 [6], GPGPU-Sim [4], Accel-Sim [33]) achieve high fidelity but at 1000–10000× slowdown; sampling techniques [57, 69] help but were not designed for ML workloads. **Trace-driven simulators** like ASTRA-sim [67] (distributed training via Chakra traces [59]) and VIDUR [3] (LLM serving) trade some fidelity for orders-of-magnitude speedup. **ML-augmented approaches** learn performance functions from profiling data, ranging from random forests (nn-Meter [73]) and XGBoost (TVM [10]) to deep learning (NeuSight [41]) and meta-learning (HELP [40]); they capture non-linear relationships but may not generalize beyond their training distribution.

3.3 Problem Formulation

Performance modeling maps workload $\mathcal{W}$ and hardware $\mathcal{H}$ to a metric y: $\hat{y} = f(\mathcal{W}, \mathcal{H}; \theta)$, with workloads represented at operator, graph, IR, or trace level, and hardware characterized by specifications, counters, or learned embeddings. Prediction targets include latency, throughput, energy, and memory footprint. Accuracy metrics—MAPE, RMSE, and rank correlation (Kendall's τ)—vary across the literature, and differences in benchmarks, hardware targets, and evaluation protocols limit direct comparison (Section 6).

4 Taxonomy

We organize the literature along three dimensions. The *primary axis* is methodology type—how a tool predicts performance—because methodology determines the fundamental trade-offs between accuracy, speed, interpretability, and data requirements. The *secondary axes* are target platform and abstraction level, which together determine the scope and applicability of each tool. We additionally characterize tools by workload coverage, exposing a pervasive CNN-validation bias in the literature.

Table 1 provides a unified view combining the coverage matrix (number of surveyed tools per methodology–platform cell) with trade-off profiles, with empty cells highlighting research gaps. The dominant pairings are: analytical models for accelerators, cycle-accurate simulation for GPUs/CPUs, trace-driven simulation for distributed systems, and ML-augmented approaches for edge devices.

Table 1: Methodology taxonomy: coverage matrix and trade-off profile. Platform columns show the number of surveyed tools per cell; 0 indicates an explicit research gap. Speed, data requirements, and interpretability determine practical applicability; the failure mode column identifies the primary condition under which each methodology breaks down.

Methodology	DNN Accel.	GPU	Distrib. Systems	Edge/ Mobile	CPU	Eval. Speed	Data Req.	Interp.	Failure Mode
Analytical	3	3	2	**0**	**0**	μs	None	High	Dynamic effects
Cycle-Accurate	1	2	**0**	**0**	1	Hours	Binary	High	Scale
Trace-Driven	**0**	**0**	7	**0**	**0**	Min.	Traces	Med.	Trace fidelity
ML-Augmented	**0**	3	**0**	3	1	ms	Profiling	Low	Distrib. shift
Hybrid	1	2	**0**	**0**	1	ms	Mixed	Med.	Training domain

Table 1 reveals three structural observations. First, trace-driven simulation is exclusively used for distributed systems—no surveyed tool applies trace-driven methods to single-device GPU or accelerator modeling, despite the potential for trace-driven approaches to avoid the slowdown of cycle-accurate simulation while retaining more fidelity than analytical models. Second, edge/mobile devices are served exclusively by ML-augmented approaches; the absence of analytical or hybrid models for edge devices reflects the hardware diversity problem but also represents a research gap, since hybrid approaches could combine the interpretability of analytical models with the adaptability of learned components. Third, no ML-augmented or hybrid tool specifically targets distributed system modeling—tools like VIDUR use ML internally for kernel prediction but are architecturally trace-driven simulators. The trade-off columns further show that methodologies cluster into two speed regimes: sub-millisecond (analytical, ML-augmented, hybrid) suitable for design space exploration, and minutes-to-hours (simulation, trace-driven) suitable for detailed validation.

4.1 Primary Axis: Methodology Type

The choice of methodology determines fundamental trade-offs (Table 1); Section 5 provides detailed per-tool analysis. **Analytical models** express performance as closed-form functions, offering microsecond evaluation and full interpretability but requiring per-architecture derivation (e.g., Timeloop [47]: 5–10% error at 2000× speedup). **Cycle-accurate simulators** (GPGPU-Sim [4], Accel-Sim [33]) achieve high fidelity at 1000–10000× slowdown, impractical for DSE. **Trace-driven simulators** replay execution traces for system-level modeling (ASTRA-sim [67]: 5–15% error; VIDUR [3]: <5%), with some using ML internally. **ML-augmented models** learn from profiling data (nn-Meter [73], TVM [10]), but suffer from *silent distribution shift*. **Hybrid** approaches combine analytical structure with learned residuals (NeuSight [41]: 2.3% MAPE), achieving the best accuracy–speed trade-offs; transfer learning provides 22.5% average improvement [15].

4.2 Secondary Axes: Platform and Abstraction Level

The target platform constrains applicable methodologies: **DNN accelerators** [29, 30] are best served by analytical models (Timeloop, MAESTRO, Sparseloop) due to regular memory hierarchies; **GPUs** span the full methodology spectrum reflecting SIMT complexity; **distributed systems** require trace-driven simulation (ASTRA-sim, VIDUR, SimAI) for collective communication and pipeline parallelism; **edge/mobile devices** are dominated by ML-augmented approaches (nn-Meter, LitePred, HELP) due to hardware diversity; and **CPUs** are less studied for ML workloads (Concorde [46], GRANITE [61]).

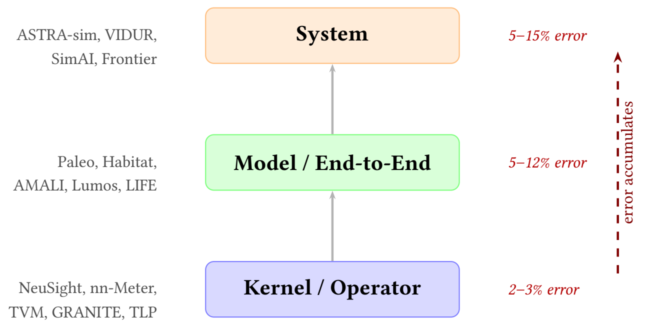
ASTRA-sim, VIDUR, SimAI, Frontier
System
5–15% error
Paleo, Habitat, AMALI, Lumos, LIFE
Model / End-to-End
5–12% error
NeuSight, nn-Meter, TVM, GRANITE, TLP
Kernel / Operator
2–3% error
error accumulates

Figure 1: Abstraction level hierarchy and the composition problem. Tools operate at one of three levels; composing predictions across levels accumulates error. Error ranges are representative values from surveyed papers.

Abstraction level determines where composition errors arise. **Kernel-level** tools (NeuSight, nn-Meter, TVM) achieve 2–3% error but composing predictions into end-to-end latency introduces errors from memory allocation, kernel launch overhead, and inter-operator data movement. **Model-level** tools (Paleo, Habitat, AMALI) account for graph-level effects at 5–12% error. **System-level** tools (ASTRA-sim, VIDUR, SimAI) capture communication and scheduling at 5–15% error, with kernel-level errors propagating through the composition chain. Figure 1 illustrates this hierarchy.

4.3 Workload Coverage

Table 2 characterizes the workload types on which each tool has been validated, exposing a pervasive CNN-validation bias.

The table reveals that **no surveyed tool has been validated on diffusion models or dynamic inference workloads** [35], only Frontier [18] has validated MoE support, and no single tool offers validated transformer prediction across the full kernel-to-system stack. Practitioners working with non-CNN workloads must accept unvalidated predictions, collect their own validation data, or fall back to measurement.

Table 2: Workload validation coverage. ✓ = validated in the original paper; ○ = partial or indirect validation; — = no validation. Nearly all tools report accuracy on CNN workloads; transformer and MoE coverage is sparse. Empty columns (diffusion, dynamic inference) represent workload types with *no* validated performance modeling tools.

Tool	CNN	Trans-former	LLM Train	MoE	Diff.
Timeloop	✓	○	—	—	—
MAESTRO	✓	—	—	—	—
NeuSight	✓	✓	—	—	—
Habitat	✓	—	—	—	—
AMALI	—	✓	—	—	—
ASTRA-sim	✓	○	✓	—	—
VIDUR	—	✓	—	—	—
SimAI	—	—	✓	—	—
Lumos	—	—	✓	—	—
Frontier	—	✓	—	✓	—
nn-Meter	✓	—	—	—	—
LitePred	✓	—	—	—	—
HELP	✓	—	—	—	—
TVM/Ansor	✓	○	—	—	—

5 Survey of Approaches

This section surveys performance modeling tools for ML workloads, organized by target platform, examining modeling challenges, available tools, and their strengths and limitations. Table 3 provides a comprehensive comparison.

5.1 DNN Accelerator Modeling

DNN accelerators employ specialized dataflows and memory hierarchies optimized for tensor operations [62], and their computational regularity makes this domain particularly amenable to analytical modeling.

Analytical frameworks dominate. Timeloop [47] computes data reuse, latency, and energy from loop-nest representations at 5–10% error versus RTL simulation with 2000× speedup, and provides deterministic reference outputs for standard designs (Eyeriss [11], Simba). MAESTRO [37] simplifies specification via data-centric directives but is less precise for energy modeling. Sparseloop [68] extends Timeloop to sparse tensors by modeling sparsity patterns, compression formats (CSR, bitmap), and hardware intersection units—critical for transformer inference but limited to static, known sparsity distributions.

Simulation and ML-augmented approaches. PyTorchSim [34] integrates PyTorch 2 with cycle-accurate NPU simulation, directly consuming computation graphs to eliminate manual workload translation, but does not report real-hardware accuracy and inherits simulation speed limitations. ArchGym [36] connects ML surrogates to analytical simulators for design space exploration; its 0.61% RMSE measures surrogate-vs-simulator fidelity, not real-hardware accuracy. **Synthesis.** Accelerator modeling is the most mature subdomain, with Timeloop as the de facto DSE standard. The key gap is silicon validation—no surveyed accelerator tool validates against manufactured hardware, leaving all anchored to RTL comparisons. Emerging PIM modeling tools [22, 27, 39, 48] lack real hardware validation (Section 8).

5.2 GPU Performance Modeling

GPUs dominate ML training and inference, requiring models that account for SIMT execution, warp scheduling, memory coalescing, and occupancy effects.

Cycle-accurate simulation. GPGPU-Sim [4] and Accel-Sim [33] achieve 0.90–0.97 IPC correlation; recent reverse-engineering of modern GPU cores [26] improved Accel-Sim to 13.98% MAPE. However, 1000–10000× slowdown makes these tools impractical at production scale.

Analytical models. The roofline model [66] provides upper bounds but misses dynamic effects; Roofline-LLM [28] extends it to LLM inference. AMALI [9] reduces LLM inference MAPE from 127% to 23.6% via memory hierarchy modeling—the residual error reflects the fundamental difficulty of analytically capturing GPU dynamic behavior (warp scheduling, cache contention).

Hybrid learned models. NeuSight [41] achieves 2.3% MAPE on GPT-3 inference (H100/A100/V100) via tile-based prediction mirroring CUDA execution. Habitat [71] uses wave scaling to decompose compute and memory components, achieving 11.8% cross-GPU transfer error (e.g., V100→A100), but requires source GPU profiling and assumes occupancy patterns remain similar across generations. Direct comparison requires caution: NeuSight targets 2023–2025 hardware with LLMs, while Habitat was designed for earlier GPUs with CNNs.

LLM-specific modeling. LLM execution exhibits distinct prefill (compute-bound) and decode (memory-bound) phases [50, 75]. VIDUR [3] simulates serving with scheduling strategies (Orca [70], Sarathi [2]) at <5% error; LIFE [17] and HERMES [5] target analytical and disaggregated inference, respectively.

Compiler cost models. TVM [10] and Ansor [74] use ML cost models for autotuning at ~15% MAPE; TLP [72] uses transformer-based modeling at <10% MAPE. SynPerf [65] and SwizzlePerf [63] prioritize ranking accuracy over absolute error.

Synthesis. GPU modeling spans the widest methodological range (2%–24% error), reflecting the tension between microarchitectural complexity and rapid hardware evolution. NeuSight offers the best accuracy–speed trade-off for LLM workloads, while AMALI and LIFE fill the pre-silicon gap despite higher error.

5.3 Distributed Training and LLM Serving

Distributed systems introduce communication overhead, synchronization barriers, and parallelism strategy choices across tensor [58], pipeline [25], and data parallelism [52].

Training simulation. ASTRA-sim [67] provides end-to-end simulation via Chakra traces [59] at 5–15% error versus HGX-H100 clusters. SimAI [64] achieves 1.9% MAPE at Alibaba Cloud scale; Lumos [43] achieves 3.3% on H100 training; TrioSim [42] and Echo [7] offer additional approaches. PRISM [19] produces prediction intervals at 10K+ GPU scale.

Scaling and parallelism. Paleo [51] pioneered analytical training-time estimation; MAD Max [24] extends this per parallelism dimension; Sailor [60] addresses heterogeneous clusters. The Llama 3

Table 3: Summary of surveyed performance modeling tools for ML workloads, organized by target platform. Methodology: A=Analytical, S=Simulation, T=Trace-driven, M=ML-augmented, H=Hybrid. *Accuracy measures surrogate-vs-simulator fidelity, not real hardware error. †Reported accuracy unverifiable due to reproducibility issues. ‡No accuracy baseline against real hardware reported.

Tool	Platform	Method	Target	Accuracy	Speed	Key Capability
DNN Accelerator Modeling						
Timeloop [47]	NPU	A	Latency/Energy	5–10%	μs	Loop-nest DSE
MAESTRO [37]	NPU	A	Latency/Energy	5–15%	μs	Data-centric directives
Sparseloop [68]	NPU	A	Sparse tensors	5–10%	μs	Compression modeling
PyTorchSim [34]	NPU	S	Cycle-accurate	N/A‡	Hours	PyTorch 2 integration
ArchGym [36]	Multi	H	Multi-objective	0.61%*	ms	ML-aided DSE
GPU Performance Modeling						
Accel-Sim [33]	GPU	S	Cycle-accurate	10–20%	Hours	SASS trace-driven
GPGPU-Sim [4]	GPU	S	Cycle-accurate	10–20%	Hours	CUDA workloads
AMALI [9]	GPU	A	LLM inference	23.6%	ms	Memory hierarchy
NeuSight [41]	GPU	H	Kernel/E2E latency	2.3%	ms	Tile-based prediction
Habitat [71]	GPU	H	Training time	11.8%	Per-kernel	Wave scaling
Distributed Training and LLM Serving						
ASTRA-sim [67]	Distributed	T	Training time	5–15%	Minutes	Collective modeling
SimAI [64]	Distributed	T	Training time	1.9%	Minutes	Full-stack simulation
Lumos [43]	Distributed	T	LLM training	3.3%	Minutes	H100 training
VIDUR [3]	GPU cluster	T	LLM serving	<5%	Seconds	Prefill/decode phases
Frontier [18]	Distributed	T	MoE inference	—	Minutes	Stage-centric sim.
TrioSim [42]	Multi-GPU	T	DNN training	N/A‡	Minutes	Lightweight multi-GPU
Edge Device Modeling						
nn-Meter [73]	Edge	M	Latency	<1%†	ms	Kernel detection
LitePred [16]	Edge	M	Latency	0.7%	ms	85-platform transfer
HELP [40]	Multi	M	Latency	1.9%	ms	10-sample adaptation
Compiler Cost Models						
TVM [10]	GPU	M	Schedule perf.	~15%	ms	Autotuning guidance
Ansor [74]	GPU	M	Schedule perf.	~15%	ms	Program sampling
TLP [72]	GPU	M	Tensor program	<10%	ms	Transformer cost model

study [13] documents 4D parallelism at 16K H100 GPUs as validation ground truth.

Inference serving. VIDUR [3] extends to distributed serving with vLLM [38] scheduling; Frontier [18] targets MoE and disaggregated inference; ThrottLL'eM [31] models GPU power effects. KV cache management remains the key bottleneck [38], and algorithmic innovations (speculative decoding [8]) create a moving-target challenge for simulators.

Synthesis. Distributed modeling is the fastest-growing subdomain, bifurcating into *trace-driven fidelity* (ASTRA-sim, SimAI) and *analytical decomposition* (Paleo, MAD Max), with PRISM's probabilistic approach as an emerging third path.

5.4 Edge Device Modeling

Edge devices impose strict power, memory, and latency constraints, and their hardware diversity (mobile CPUs, GPUs, NPUs, DSPs) makes per-device analytical modeling impractical, driving ML-augmented approaches.

nn-Meter [73] reports <1% MAPE but scores 3/10 in our reproducibility evaluation (Section 7) due to unpinned dependencies. LitePred [16] achieves 0.7% MAPE across 85 (predominantly ARM) platforms; HELP [40] achieves 1.9% via 10-sample meta-learning adaptation. ESM [45] finds well-tuned random forests match deep learning surrogates, and transfer learning provides 22.5% average improvement [15].

Synthesis. Edge modeling is dominated by ML-augmented approaches with data quality and tool longevity mattering more than model sophistication—nn-Meter's reproducibility failure (Section 7) underscores this lesson.

5.5 Cross-Cutting Themes

Structural decomposition mirroring hardware execution outperforms black-box approaches (Timeloop's loop nests, NeuSight's tiles, VIDUR's prefill/decode), while *verifiable moderate accuracy* matters more than unverifiable high accuracy (Docker-first tools: 8.5+/10 vs. nn-Meter: 3/10). A persistent **accuracy–generality–speed trilemma** explains why no single methodology dominates: each maximizes a different vertex at the expense of the others.

6 Comparison and Analysis

We analyze trade-offs across methodology types along accuracy and speed dimensions (see Table 3 for per-tool details); generalization and interpretability challenges are deferred to Section 8.

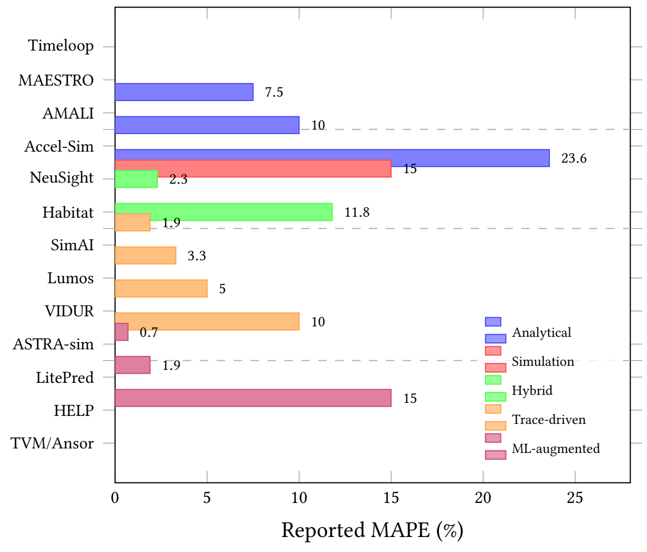

Figure 2: Reported accuracy (MAPE) of surveyed tools, grouped by methodology type. Range midpoints used where ranges are reported. Cross-tool comparison is approximate due to differing benchmarks, workloads, and hardware targets.

6.1 Accuracy by Problem Difficulty

We organize accuracy results by inherent problem difficulty rather than comparing across incompatible benchmarks (Figure 2). Accelerator dataflow modeling is most tractable (Timeloop: 5–10%); single-GPU kernel prediction achieves 2–12% via hybrid methods (NeuSight, Habitat); distributed systems reach 2–15% (SimAI 1.9%, ASTRA-sim 5–15%); cross-platform edge prediction achieves 0.7–2% but requires per-device profiling; and GPU analytical modeling remains hardest (AMALI: 23.6%). Setup costs vary dramatically: analytical models require only architecture specifications, ML-augmented approaches need 10–10K profiling samples per device, and cycle-accurate simulators require hardware-specific binaries or traces.

6.2 Practitioner Tool Selection

Tool selection follows platform and use case. For *accelerator DSE*, use Timeloop or MAESTRO for microsecond-speed exhaustive search with interpretable bottleneck feedback. For *GPU evaluation*, NeuSight offers the best accuracy–speed balance for LLMs; use Accel-Sim when microarchitectural detail is needed. For *distributed systems*, use VIDUR for LLM serving configuration and ASTRA-sim or SimAI for training parallelism at scale. For *edge devices*, LitePred offers the broadest platform coverage, while HELP excels with minimal profiling data.

7 Experimental Evaluation

We conducted hands-on evaluations of five tools spanning methodology types: Timeloop (analytical), ASTRA-sim (trace-driven, distributed), VIDUR (trace-driven, LLM serving), nn-Meter (ML-augmented, edge), and NeuSight (hybrid, GPU).

Table 4: Reproducibility evaluation scores (10-point rubric). Tools are ranked by total score. †Timeloop CLI works but Python bindings fail.

Tool	Setup	Reprod.	Usability	Total
VIDUR	2.5	3.5	3	9/10
Timeloop†	3	4	2	9/10
ASTRA-sim	2.5	3	3	8.5/10
NeuSight	2	3	2.5	7.5/10
nn-Meter	2	0	1	3/10

Table 5: VIDUR simulation results for Llama-2-7B inference serving on a simulated A100 GPU. All metrics from our own experiments.

Metric	vLLM	Sarathi
Requests	200	50
Avg E2E latency (s)	0.177	0.158
P99 E2E latency (s)	0.320	0.270
Avg TTFT (s)	0.027	0.025
Avg TPOT (s)	0.0093	0.0090
Requests preempted	53	0

Environment and rubric. All evaluations ran on Apple M2 Ultra (aarch64, 192 GB RAM) using Docker containers where provided—no GPU hardware was available, so we cannot validate absolute accuracy claims. We score each tool on a 10-point rubric: *Setup* (3 pts), *Reproducibility* (4 pts), *Usability* (3 pts). Table 4 summarizes results.

7.1 Per-Tool Results

VIDUR (9/10)—the highest-scoring tool. We simulated Llama-2-7B inference on a simulated A100 using vLLM and Sarathi scheduling (Table 5). VIDUR correctly captures scheduling trade-offs: Sarathi achieves 12.2% lower latency than vLLM (0.158 s vs. 0.177 s) via chunked prefill [2], TPOT differs by only 3.5% (confirming hardware-bound decode), and vLLM preempted 26.5% of requests while Sarathi preempted zero—matching the algorithmic difference in KV-cache management [38]. VIDUR's Docker-pinned dependencies avoid the serialization failures seen in nn-Meter.

Timeloop (9/10). The Docker image provides CLI tools that work correctly for Eyeriss-like configurations with fully deterministic, bit-identical outputs—a significant reproducibility strength. Reference outputs for standard designs (Eyeriss, Simba) enable verification without hardware. However, Python bindings fail (`ImportError: libbarvinok.so.23`), preventing programmatic use.

ASTRA-sim (8.5/10). We executed 8-NPU collective microbenchmarks and ResNet-50 data-parallel training at 2–8 GPUs on HGX-H100 (Table 6). Collective operation ratios are physically plausible: Reduce-Scatter takes half the time of All-Reduce (consistent with half the data), while All-to-All takes ~2× All-Reduce. Communication overhead scales from 0.05% (2 GPUs) to 0.30% (8 GPUs)—a 5.76× increase for 4× more GPUs, consistent with ring All-Reduce scaling. Scale coverage is limited to 8 NPUs by included topology files.

Table 6: ASTRA-sim quantitative results from our experiments on the HGX-H100 configuration. Top: collective microbenchmarks (8 NPUs, 1 MB). Bottom: ResNet-50 data-parallel training scaling.

Collective Microbenchmarks (8 NPUs, 1 MB)		
Collective	**Cycles**	**Ratio vs. AR**
All-Reduce	57,426	1.000
All-Gather	44,058	0.767
Reduce-Scatter	28,950	0.504
All-to-All	114,000	1.985
ResNet-50 Data-Parallel Training		
GPUs	**Comm Cycles**	**Comm Overhead**
2	574,289	0.05%
4	1,454,270	0.13%
8	3,307,886	0.30%

NeuSight (7.5/10). The tile-based decomposition correctly mirrors CUDA tiling for standard dense operations, but testing on irregular workloads was limited by missing examples, suggesting the tool is best validated for regular LLM workloads.

nn-Meter (3/10)—the lowest-scoring tool. After four installation attempts (>4 hours), we could not execute *any* predictions due to a chain of unpinned dependencies: pickle-serialized predictors (scikit-learn 0.23.1) are incompatible with current versions, and onnx-simplifier fails on aarch64. The claimed <1% MAPE is **unverifiable on any current software stack**; the tool has received no updates since 2022.

7.2 Lessons and Threats to Validity

Five lessons emerge: (1) **Docker-first deployment** is the strongest reproducibility predictor (Docker tools: 8.5+/10; nn-Meter without Docker: 3/10). (2) **ML model serialization is fragile**—nn-Meter's pickle-based predictors became unusable within two years. (3) **Reference outputs enable trust without hardware**—Timeloop and ASTRA-sim include verifiable baselines. (4) **Scale-limited evaluation understates system tools**—our 2–8 GPU tests show only 0.30% communication overhead, far below production scales [13]. (5) **Reproducible accuracy claims should be weighted higher** than unreproducible ones.

Threats. Our venue-focused search may under-represent industry and non-English publications; we exclude proprietary tools (Nsight Compute, internal TPU models); and accuracy metrics vary across papers (MAPE, RMSE, Kendall's τ), limiting direct comparison.

8 Open Challenges and Future Directions

Generalization gaps. Three dimensions of generalization remain largely unsolved. *Workload generalization*: nearly all accuracy numbers are measured on CNNs, with CNN→transformer transfer largely unvalidated (NeuSight is a notable exception); MoE, diffusion models, and dynamic inference [35] have almost no validated prediction tools. Neural scaling laws [12, 20, 23, 32] predict training loss but not hardware-specific latency. Figure 3 shows the shift

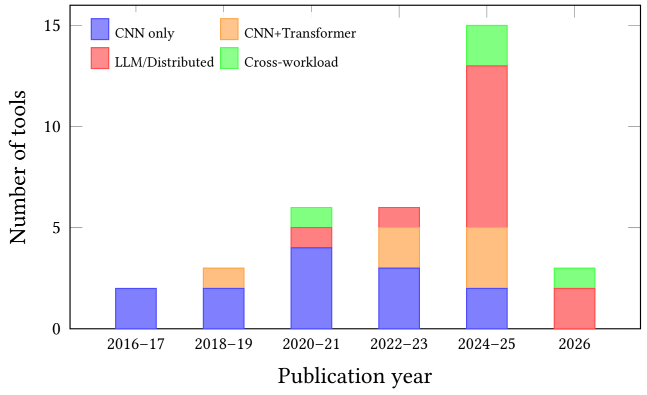

Figure 3: Workload coverage of surveyed tools by publication period. The shift toward transformer and LLM workloads accelerates from 2023, but MoE and diffusion models remain largely uncharacterized.

from CNN-only validation toward LLM workloads since 2023, but MoE and diffusion remain uncharacterized. *Hardware generalization*: meta-learning (HELP: 10-sample adaptation), feature-based transfer (LitePred: 85 devices), and analytical decomposition (Habitat) show promise, but cross-family transfer (GPU→TPU→PIM) remains unsolved. *Temporal generalization*—software stack evolution silently invalidating trained models—is addressed by no surveyed tool.

The composition problem. Composing kernel-level predictions into end-to-end estimates is unsolved: NeuSight's 2.3% kernel MAPE yields ~10× higher variance at model level ($\sigma_{\text{model}} \approx \sigma_{\text{kernel}} \cdot \sqrt{N}$), and correlated errors can compound linearly. VIDUR sidesteps this by profiling entire prefill/decode phases.

Emerging hardware and reproducibility. PIM architectures [22, 27, 39, 48], chiplets, and disaggregated designs blur conventional memory hierarchy assumptions; hardware-aware optimizations (FlashAttention [14]) change the performance landscape faster than models can be retrained. On the reproducibility front, no equivalent of MLPerf [44, 56] exists for performance *prediction*—standardized prediction benchmarks would significantly advance the field. Analytical models remain the most interpretable: Timeloop identifies data movement bottlenecks, MAESTRO reveals suboptimal dataflows, and VIDUR exposes scheduling inefficiencies, while ML-augmented approaches offer limited causal understanding.

Future directions. Five priorities: (1) transformer/MoE-aware tools with validated non-CNN accuracy; (2) validated composition methods with bounded end-to-end error; (3) unified energy-latency-memory prediction [55]; (4) temporal robustness benchmarks for software stack evolution; (5) unified tooling with Docker-first deployment, portable formats (ONNX), and standard workload representations (Chakra [59]).

9 Conclusion

This survey analyzed approximately 25 tools for predicting ML workload performance, organized by methodology type, target platform, and abstraction level. Key findings: (1) *Methodology determines* *trade-offs, not quality*—analytical models offer microsecond interpretable evaluation, trace-driven simulators provide 2–15% system-level error, and hybrid approaches achieve the best accuracy–speed balance (NeuSight: 2.3% MAPE). (2) *LLM workloads demand specialized modeling*—prefill/decode distinctions, KV cache management, and dynamic batching require purpose-built tools (VIDUR, Frontier) rather than CNN-era extensions. (3) *Reproducibility is a practical bottleneck*—Docker-first tools score 8.5+/10 while tools relying on serialized ML models have become unusable. (4) *Accuracy claims require scrutiny* due to varying benchmarks and metrics.

The most pressing gaps are CNN-to-transformer generalization, kernel-to-end-to-end composition, emerging hardware support (PIM, chiplets), and reproducibility failures. As ML workloads grow in scale and diversity, this survey provides practitioners guidance for tool selection and researchers a roadmap for advancing the field.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 265–283.

[2] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramachandran. 2024. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 117–134.

[3] Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramachandran. 2024. VIDUR: A Large-Scale Simulation Framework for LLM Inference. In *Proceedings of Machine Learning and Systems (MLSys)*. 1–15.

[4] Ali Bakhoda, George L. Yuan, Wilson W. L. Fung, Henry Wong, and Tor M. Aamodt. 2009. Analyzing CUDA Workloads Using a Detailed GPU Simulator. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 163–174. https://doi.org/10.1109/ISPASS.2009.4919648

[5] Abhimanyu Rajeshkumar Bambhaniya et al. 2025. HERMES: Understanding and Optimizing Multi-Stage AI Inference Pipelines. *arXiv preprint arXiv:2504.09775* (2025). Heterogeneous multi-stage LLM inference simulator with analytical modeling.

[6] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 Simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7. https://doi.org/10.1145/2024716.2024718

[7] Kai Cai, Wei Miao, Junyu Zhu, Jiaxu Chen, Hao Shan, Huanyu Li, and Chi Zhang. 2024. Echo: Simulating Distributed Training At Scale. *arXiv preprint arXiv:2412.12487* (2024).

[8] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. MEDUSA: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. 1–15.

[9] Zheng Cao et al. 2025. AMALI: An Analytical Model for Accurately Modeling LLM Inference on Modern GPUs. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–14. https://doi.org/10.1145/3695053.3731064 Reduces GPU LLM inference MAPE from 127.56% to 23.59% vs GCoM baseline.

[10] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. 2018. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 578–594.

[11] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*. 367–379. https://doi.org/10.1109/ISCA.2016.40

[12] Leshem Choshen, Yang Zhang, and Jacob Andreas. 2025. A Hitchhiker's Guide to Scaling Law Estimation. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*. 1–25. Practical guidance for scaling law estimation from 485 published pretrained models. IBM/MIT..

[13] Weiwei Chu, Xinfeng Xie, Jiecao Yu, Jie Wang, Pavan Balaji, Ching-Hsiang Chu, Jongsoo Park, et al. 2025. Scaling Llama 3 Training with Efficient Parallelism Strategies. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–15. 4D parallelism for Llama 3 405B on 16K H100 GPUs. Achieves 400 TFLOPs/GPU. Meta..

[14] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35. 16344–16359.

[15] Lukasz Dudziak, Thomas Chau, Mohamed S. Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. 2024. Latency Predictors for Neural Architecture Search. In *Proceedings of Machine Learning and Systems (MLSys)*. 1–14.

[16] Yang Feng, Zhehao Li, Jiacheng Yang, and Yunxin Liu. 2024. LitePred: Transferable and Scalable Latency Prediction for Hardware-Aware Neural Architecture Search. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 1–18.

[17] Paraskevas Gavriilidis et al. 2025. LIFE: Forecasting LLM Inference Performance via Hardware-Agnostic Analytical Modeling. *arXiv preprint arXiv:2508.00904* (2025). Hardware-agnostic analytical model for LLM inference performance forecasting.

[18] Siddharth Ghosh et al. 2025. Frontier: Simulating the Next Generation of LLM Inference Systems. *arXiv preprint arXiv:2508.03148* (2025). Stage-centric simulator for MoE and disaggregated LLM inference, models expert parallelism and cross-cluster routing.

[19] Alicia Golden et al. 2025. PRISM: Probabilistic Runtime Insights and Scalable Performance Modeling for Large-Scale Distributed Training. *arXiv preprint arXiv:2510.15596* (2025). Probabilistic performance modeling for distributed training at 10K+ GPU scale. Meta..

[20] Alexander Hagele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. 2024. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 37. Spotlight. Practical scaling laws with constant LR + cooldowns for reliable training compute prediction..

[21] John L. Hennessy and David A. Patterson. 2019. A New Golden Age for Computer Architecture. *Commun. ACM* 62, 2 (2019), 48–60. https://doi.org/10.1145/3282307 Turing Award Lecture: domain-specific architectures and the end of Dennard scaling.

[22] Guseul Heo, Sangyeop Lee, Jaehong Cho, Hyunmin Choi, Sanghyeon Lee, Hyungkyu Ham, Gwangsun Kim, Divya Mahajan, and Jongse Park. 2024. NeuPIMs: NPU-PIM Heterogeneous Acceleration for Batched LLM Inferencing. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–17. NPU-PIM heterogeneous architecture for LLM inference with performance modeling. KAIST/Georgia Tech..

[23] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556* (2022). Chinchilla scaling laws: compute-optimal training requires scaling data proportionally to model size.

[24] Samuel Hsia, Kartik Chandra, and Kunle Olukotun. 2024. MAD Max Beyond Single-Node: Enabling Large Machine Learning Model Acceleration on Distributed Systems. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*. 753–766. https://doi.org/10.1109/ISCA59077.2024.00064

[25] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32. 103–112.

[26] Rodrigo Huerta, Mojtaba Abaie Shoushtary, Jose-Lorenzo Cruz, and Antonio Gonzalez. 2025. Dissecting and Modeling the Architecture of Modern GPU Cores. In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 369–384. Reverse-engineers modern NVIDIA GPU cores, improves Accel-Sim to 13.98% MAPE. UPC Barcelona..

[27] Bongjoon Hyun, Taehun Kim, Dongjae Lee, and Minsoo Rhu. 2024. Pathfinding Future PIM Architectures by Demystifying a Commercial PIM Technology. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–15. uPIMulator: cycle-accurate PIM simulation framework for UPMEM. KAIST..

[28] Ryota Imai, Kentaro Harada, Ryo Sato, and Toshio Nakaike. 2024. Roofline-Driven Machine Learning for Large Language Model Performance Prediction. *NeurIPS Workshop on Machine Learning for Systems* (2024).

[29] Norman P. Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)* (2023), 1–14. https://doi.org/10.1145/3579371.3589350 4096-chip pods with 3D optical interconnect; up to 1.7x/2.1x faster than TPU v3.

[30] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borber, et al. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*. 1–12. https://doi.org/10.1145/3079856.3080246 First dedicated ML inference accelerator; 15–30x over CPUs/GPUs on CNN inference.

[31] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Petros Vavaroutsos, Sotirios Xydis, and Dimitrios Soudris. 2025. throttLL'eM: Predictive GPU Throttling for Energy Efficient LLM Inference Serving. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. Achieves up to 43.8% lower energy consumption for LLM inference.

[32] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020). Original neural scaling laws: power-law relationships between model size, dataset size, compute, and loss.

[33] Mahmoud Khairy, Zhesheng Shen, Tor M. Aamodt, and Timothy G. Rogers. 2020. Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling. In *Proceedings of the 47th International Symposium on Computer Architecture (ISCA)*. 473–486. https://doi.org/10.1109/ISCA45697.2020.00047

[34] Jungho Kim et al. 2025. PyTorchSim: A Comprehensive, Fast, and Accurate NPU Simulation Framework. In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–14. https://doi.org/10.1145/3725843.3756045 PyTorch 2-integrated NPU simulator with custom RISC-V ISA and Tile-Level Simulation.

[35] Jiin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. 2026. The Cost of Dynamic Reasoning: Demystifying AI Agents and Test-Time Scaling from an AI Infrastructure Perspective. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. HPCA 2026 (Jan 31–Feb 4, 2026, Las Vegas). First comprehensive system-level analysis of AI agents; quantifies resource usage, latency, and datacenter power consumption.

[36] Srivatsan Krishnan, Amir Yazdanbakhsh, Shvetank Prakash, Norman P. Jouppi, Jignesh Parmar, Hyoukjun Kim, James Laudon, and Chandrakant Narayanaswami. 2023. ArchGym: An Open-Source Gymnasium for Machine Learning Assisted Architecture Design. In *Proceedings of the 50th International Symposium on Computer Architecture (ISCA)*. 1–16. https://doi.org/10.1145/3579371.3589049

[37] Hyoukjun Kwon, Prasanth Chatarasi, Michael Sarber, Michael Pellauer, Angshuman Parashar, and Tushar Krishna. 2019. MAESTRO: A Data-Centric Approach to Understand Reuse, Performance, and Hardware Cost of DNN Mappings. In *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–14. https://doi.org/10.1145/3352460.3358292

[38] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*. 611–626. https://doi.org/10.1145/3600006.3613165

[39] Hyojung Lee, Daehyeon Baek, Jiyoung Son, Jieun Choi, Kihyo Moon, and Minsung Jang. 2025. PAISE: PIM-Accelerated Inference Scheduling Engine for Transformer-based LLM. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. PIM-based LLM inference scheduling. 48.3% speedup, 11.5% power reduction. Samsung..

[40] Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. 2021. HELP: Hardware-Adaptive Efficient Latency Prediction for NAS via Meta-Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 27016–27028.

[41] Seunghyun Lee, Amar Phanishayee, and Divya Mahajan. 2025. NeuSight: GPU Performance Forecasting via Tile-Based Execution Analysis. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–15.

[42] Jianbo Li et al. 2025. TrioSim: A Lightweight Simulator for Large-Scale DNN Workloads on Multi-GPU Systems. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–13. Multi-GPU DNN simulation with lightweight approach for distributed training analysis.

[43] Wenxuan Liang et al. 2025. Lumos: Efficient Performance Modeling and Estimation for Large-scale LLM Training. In *Proceedings of Machine Learning and Systems (MLSys)*. 1–16. Trace-driven performance modeling achieving 3.3% error on H100 GPUs for LLM training.

[44] Peter Mattson, Christine Cheng, Cody Coleman, Greg Diamos, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, et al. 2020. MLPerf Training Benchmark. In *Proceedings of Machine Learning and Systems (MLSys)*. 336–349. Standard ML training benchmark suite covering image classification, object detection, NLP, recommendation, reinforcement learning.

[45] Azaz-Ur-Rehman Nasir, Samroz Ahmad Shoaib, Muhammad Abdullah Hanif, and Muhammad Shafique. 2025. ESM: A Framework for Building Effective Surrogate Models for Hardware-Aware Neural Architecture Search. In *Proceedings of the 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–6. 97.6% accuracy surrogate model framework for HW-aware NAS.

[46] Amir Nasr-Esfahany et al. 2025. Concorde: Fast and Accurate CPU Performance Modeling with Compositional Analytical-ML Fusion. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–15. Hybrid analytical-ML approach achieving 2% CPI error at 5 orders of magnitude faster than gem5.

[47] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. 2019. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 304–315. https://doi.org/10.1109/ISPASS.2019.00042

[48] Jaehyun Park, Jaewan Choi, Kwanhee Kyung, Michael Jaemin Kim, Yongsuk Kwon, Nam Sung Kim, and Jung Ho Ahn. 2024. AttAcc! Unleashing the Power of PIM for Batched Transformer-based Generative Model Inference. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–16. PIM-based accelerator for batched transformer attention. Seoul National University/UIUC..

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32. 8024–8035.

[50] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aakanksha Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*. 118–132. https://doi.org/10.1109/ISCA59077.2024.00019 Best Paper Award.

[51] Hang Qi, Evan R. Sparks, and Ameet Talwalkar. 2017. Paleo: A Performance Model for Deep Neural Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=SyVVJ85lg

[52] Samyam Rajbhandari, Jeff Rasley, Olatunji Rber, and Yuxiong He. 2020. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*. 1–16. https://doi.org/10.1109/SC41405.2020.00024 DeepSpeed ZeRO optimizer partitioning for memory-efficient distributed training.

[53] Mehdi Rakhshanfar and Aliakbar Zarandi. 2021. A Survey on Machine Learning-based Design Space Exploration for Processor Architectures. *Journal of Systems Architecture* 121 (2021), 102339. https://doi.org/10.1016/j.sysarc.2021.102339

[54] Saeed Rashidi, Srinivas Srinivasan, Kazem Hamedani, and Tushar Krishna. 2020. ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 81–92. https://doi.org/10.1109/ISPASS48437.2020.00018

[55] Vijay Janapa Reddi et al. 2025. MLPerf Power: Benchmarking the Energy Efficiency of Machine Learning Inference. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. Energy efficiency benchmarking for ML inference workloads.

[56] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maxim Breeshekov, Mark Duber, et al. 2020. MLPerf Inference Benchmark. In *Proceedings of the 47th International Symposium on Computer Architecture (ISCA)*. 446–459. https://doi.org/10.1109/ISCA45697.2020.00045 Standard ML inference benchmark suite with server and offline scenarios.

[57] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. 2002. Automatically Characterizing Large Scale Program Behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 45–57. https://doi.org/10.1145/605397.605403 Introduces SimPoint: automatic selection of representative simulation points using k-means clustering.

[58] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. In *arXiv preprint arXiv:1909.08053*. Intra-layer tensor parallelism for large language model training.

[59] Srinivas Sridharan, Taekyung Heo, Jinwoo Choi, Garyfallia Yu, Saeed Rashidi, William Won, Zhaodong Meng, and Tushar Krishna. 2023. Chakra: Advancing Performance Benchmarking and Co-design using Standardized Execution Traces. *arXiv preprint arXiv:2305.14516* (2023).

[60] Foteini Strati, Zhendong Zhang, George Manos, Ixeia Sanchez Periz, Qinghao Hu, Tiancheng Chen, Berk Buzcu, Song Han, Pamela Delgado, and Ana Klimovic. 2025. Sailor: Automating Distributed Training over Dynamic, Heterogeneous, and Geo-distributed Clusters. In *Proceedings of the 30th ACM Symposium on Operating Systems Principles (SOSP)*. 1–18. Automated distributed training with runtime/memory simulation over heterogeneous resources. ETH Zurich/MIT..

[61] Ondrej Sykora, Alexis Rucker, Charith Mendis, Rajkishore Barik, Phitchaya Mangpo Phothilimthana, and Saman Amarasinghe. 2022. GRANITE:

A Graph Neural Network Model for Basic Block Throughput Estimation. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*. 1–13. https://doi.org/10.1109/IISWC55918.2022.00014

[62] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. In *Proceedings of the IEEE*, Vol. 105. 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740 Canonical DNN accelerator taxonomy covering dataflows, data reuse, and energy efficiency.

[63] Adrian Tschand, Mohamed Awad, et al. 2025. SwizzlePerf: Hardware-Aware LLMs for GPU Kernel Performance Optimization. *arXiv preprint arXiv:2508.20258* (2025). LLM-based spatial optimization for GPU kernels, up to 2.06x speedup via swizzling.

[64] Xizheng Wang, Qingxu Li, Yichi Xu, Gang Lu, Heyang Zhou, Sen Zhang, Yikai Zhu, Yang Liu, Pengcheng Zhang, Kun Qian, et al. 2025. SimAI: Unifying Architecture Design and Performance Tuning for Large-Scale LLM Training with Scalability and Precision. In *Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 1–18. Full-stack LLM training simulator achieving 98.1% alignment with real-world results. Alibaba Cloud/Tsinghua..

[65] Zixian Wang et al. 2025. SynPerf: Synthesizing High-Performance GPU Kernels via Pipeline Decomposition. *arXiv preprint* (2025). Under review.

[66] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (2009), 65–76. https://doi.org/10.1145/1498765.1498785

[67] William Won, Taekyung Heo, Saeed Rashidi, Saeed Talati, Srinivas Srinivasan, and Tushar Krishna. 2023. ASTRA-sim2.0: Modeling Hierarchical Networks and Disaggregated Systems for Large-Model Training at Scale. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 283–294. https://doi.org/10.1109/ISPASS57527.2023.00035

[68] Yannan Nellie Wu, Joel Emer, and Vivienne Sze. 2022. Sparseloop: An Analytical Approach to Sparse Tensor Accelerator Modeling. In *Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–15. https://doi.org/10.1109/MICRO56248.2022.00078

[69] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. 2003. SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. In *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA)*. 84–97. https://doi.org/10.1109/ISCA.2003.1206991 Statistical sampling achieving 0.64% CPI error with 35x speedup over detailed simulation.

[70] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. ORCA: A Distributed Serving System for Transformer-Based Generative Models. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 521–538.

[71] Geoffrey X. Yu, Yubo Gao, Pavel Golber, and Asaf Cidon. 2021. Habitat: A Runtime-Based Computational Performance Predictor for Deep Neural Network Training. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. 503–521.

[72] Yi Zhai, Yu Cheng Wang, Peng Jiang, and Congming Kang. 2023. TLP: A Deep Learning-based Cost Model for Tensor Program Tuning. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 833–845. https://doi.org/10.1145/3575693.3575736

[73] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. 2021. nn-Meter: Towards Accurate Latency Prediction of Deep-Learning Model Inference on Diverse Edge Devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 81–93. https://doi.org/10.1145/3458864.3467882 Best Paper Award.

[74] Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Wu, Cody Hao Yu, Ameer Haj-Ali, Yida Wang, Jun Yang, Danyang Zhuo, Koushik Sen, Joseph E. Gonzalez, and Ion Stoica. 2020. Ansor: Generating High-Performance Tensor Programs for Deep Learning. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 863–879.

[75] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianyu Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 1–18.