

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

We survey 22 performance modeling tools from 53 papers (2016–2026) and introduce the Multi-dimensional Tool Assessment Protocol (MTAP), a principled evaluation framework that assesses tools beyond accuracy across five dimensions: prediction fidelity, compositional fidelity, generalization robustness, deployment viability, and extensibility. Applying MTAP to five tools reveals three findings invisible to accuracy-only evaluation. First, tools that decompose prediction along hardware execution boundaries—loop nests for systolic arrays, tiles for GPU SMs, phases for LLM serving—consistently outperform methodology-agnostic approaches regardless of underlying technique. Second, no validated tool pipeline exists from kernel-level prediction (2–3% error) to system-level estimate (5–15% error)—the composition gap is the field’s central unsolved problem. Third, deployment methodology (Docker-first vs. serialized ML models) is a stronger predictor of tool usability than reported accuracy: the tool with the lowest reported error (<1% MAPE) fails to produce any output, while all Docker-based tools reproduce successfully.

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 have become the dominant consumers of compute across datacenters and edge devices. Training and inference for CNNs, transformers, mixture-of-experts models, and LLMs demand hardware ranging from Google’s TPU [34, 35] to custom accelerators, creating a heterogeneous landscape where architects must predict performance before committing to costly hardware decisions.

The shift toward domain-specific architectures [25] makes performance prediction both more important and more difficult. Design space exploration, parallelization selection, and hardware-software co-design all require fast, accurate performance models—yet ML workloads pose unique challenges: diverse computational patterns (dense matrix operations, sparse accesses, communication-bound collectives) across GPUs, TPUs, custom accelerators, and multi-device clusters.

A rich ecosystem of modeling tools has emerged. Analytical models (Timeloop [57], MAESTRO [43]) evaluate in microseconds with 5–15% error. Trace-driven simulators (ASTRA-sim [84], VIDUR [3])

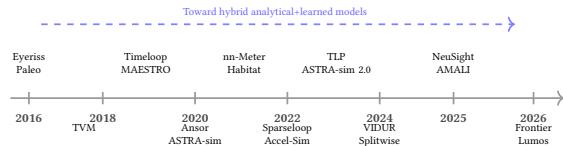


Figure 1: Evolution of performance modeling tools (2016–2026). Early analytical frameworks gave way to systematic accelerator modeling and distributed training simulation. Recent work targets LLM-specific and hybrid approaches.

replay execution traces for system-level modeling. Hybrid approaches (NeuSight [48]) combine analytical structure with learned components. Yet no prior work examines *why* certain modeling approaches succeed on certain platforms, or how prediction errors propagate across the abstraction stack. Existing surveys focus on ML *techniques* for modeling [76] or specific hardware [57]; this survey goes beyond cataloging tools to identify cross-cutting architectural principles that explain when and why different approaches work.

We make the following contributions:

- The **Multi-dimensional Tool Assessment Protocol (MTAP)**, a principled evaluation framework that assesses tools across five dimensions—prediction fidelity, compositional fidelity, generalization robustness, deployment viability, and extensibility—providing a reusable community standard beyond accuracy-only evaluation (Section 6).
- **Multi-dimensional evaluation** of five tools applying MTAP, revealing that deployment methodology (Docker-first vs. serialized ML models) is a stronger predictor of usability than reported accuracy, and that no validated tool pipeline exists from kernel prediction to system-level estimate despite a decade of development (Section 7).
- A **cross-cutting design principle**: tools that decompose prediction along hardware execution boundaries—Timeloop’s loop nests for systolic arrays, NeuSight’s tiles for GPU SMs, VIDUR’s prefill/decode phases—consistently outperform methodology-agnostic approaches regardless of underlying technique (Sections 5, 7).
- A **coverage matrix** spanning methodology type, target platform, and abstraction level that exposes structural research gaps, with an **error composition analysis** characterizing how kernel-level errors (2–3%) amplify to 5–15% at system level through uncaptured inter-kernel overheads (Sections 4, 8).

Figure 1 illustrates the evolution of performance modeling tools from early analytical frameworks to modern hybrid approaches.

117 2 Survey Methodology

118 We searched ACM Digital Library, IEEE Xplore, Semantic Scholar,
 119 and arXiv using terms related to ML performance modeling, with
 120 backward/forward citation tracking from seminal works. Target
 121 venues include architecture (MICRO, ISCA, HPCA, ASPLOS), sys-
 122 tems (MLSys, OSDI, SOSP, NSDI), and related (NeurIPS, MobiSys,
 123 DAC, ISPASS). Papers must propose or evaluate a tool for predict-
 124 ing ML workload performance with quantitative evaluation; we
 125 exclude non-performance tasks and general-purpose workloads.
 126 From 287 initial candidates, title/abstract screening yielded 118
 127 papers; full-text review reduced the set to 53 that met all crite-
 128 ria, supplemented by 12 foundational works for context. We cover
 129 2016–2026 and classify each paper by *methodology type* (analytical,
 130 simulation, trace-driven, ML-augmented, hybrid), *target platform*,
 131 and *abstraction level* (kernel, model, system).

132 **Related surveys and scope boundaries.** Prior surveys address
 133 adjacent topics: Rakhshanfar and Zarandi [65] survey ML for pro-
 134 processor DSE; Sze et al. [77] treat DNN hardware design (the founda-
 135 tion for Timeloop/MAESTRO); simulators such as GPGPU-Sim [4],
 136 gem5 [6], and SST [69] have been extensively used as validation
 137 targets in the performance modeling literature; and MLPPerf [53, 68]
 138 standardizes *measurement* rather than *prediction*. Early ML acceler-
 139 ator modeling (2014–2018) established foundational approaches: Di-
 140 anNao [11] introduced analytical dataflow modeling for dedicated
 141 accelerators, Eyeriss [13] systematized row-stationary dataflow
 142 analysis, and Paleo [61] pioneered layer-wise analytical estimation.
 143 The closest prior work, Dudziak et al. [17], compares edge device
 144 predictors for NAS; we broaden to the full landscape.

145 **Proprietary and vendor tools.** NVIDIA’s Nsight Compute [56]
 146 and Nsight Systems are the most widely-used GPU profiling tools
 147 in practice; Google’s internal TPU models underpin production
 148 scheduling but are undocumented. We exclude these from evalua-
 149 tion as they cannot be independently reproduced, though surveyed
 150 tools frequently validate against Nsight Compute data.

151 **Compiler cost models and capacity planning.** Beyond TVM/Ansor/
 152 relevant compiler models include Halide’s autoscheduler [63] (pi-
 153oneered learned cost models), MLIR-based cost models [45], and
 154 Triton’s [78] heuristic GPU kernel cost model. At the system level,
 155 Pollux [62] and Sia [33] use performance models for cluster sched-
 156 uling and capacity planning—a distinct use case (optimizing work-
 157 load placement) that shares modeling techniques with our surveyed
 158 tools.

159 This survey differs from all prior work by spanning the full
 160 methodology spectrum across all major platforms with reproducibil-
 161 ity evaluation.

163 3 Background

165 3.1 ML Workload Characteristics

166 ML workloads are expressed as computation graphs whose operator
 167 shapes are statically known and amenable to analytical modeling.
 168 Frameworks such as PyTorch [59] and TensorFlow [1] compile these
 169 graphs for execution, though MoE and dynamic inference introduce
 170 input-dependent control flow. Performance depends on tensor-to-
 171 memory mapping (dataflow, tiling), KV cache management for LLM
 172 inference [44], and at scale, compute–memory–network interac-
 173 tions across data, tensor, pipeline, and expert parallelism [15]. LLM

175 inference splits into compute-bound prefill and memory-bound
 176 decode phases [60], both modeled under batched serving [2, 86].
 177 Foundation model training introduces additional modeling chal-
 178 lenges: long-context attention with quadratic memory scaling, acti-
 179 vation checkpointing that trades compute for memory, and mixed-
 180 precision training where numerical format affects both speed and
 181 convergence [15].

182 3.2 Modeling Methodologies

183 We classify approaches into five categories. **Analytical models**
 184 express performance as closed-form functions (e.g., the roofline
 185 model [83]), offering microsecond evaluation but requiring per-
 186 architecture derivation. **Cycle-accurate simulators** (GPGPU-Sim [4],
 187 Accel-Sim [38]) achieve high fidelity at 1000–10000× slowdown,
 188 serving primarily as validation oracles for the high-level meth-
 189 ods that are the focus of this survey. **Trace-driven simulators**
 190 (ASTRA-sim [84], VIDUR [3]) trade fidelity for orders-of-magnitude
 191 speedup. **ML-augmented approaches** learn from profiling data
 192 (nn-Meter [89]) but may not generalize beyond training distri-
 193 butions. **Hybrid approaches** combine analytical structure with
 194 learned components (NeuSight [48], Habitat [87]), aiming to balance
 195 accuracy, speed, and interpretability. Accuracy metrics—MAPE,
 196 RMSE, and rank correlation—vary across the literature, limiting
 197 direct comparison (Section 7); ground-truth relies on hardware
 198 counters (PAPI [7], LIKWID [79]) or vendor profilers [56].

200 4 Taxonomy

201 We organize the literature along three dimensions. The *primary axis*
 202 is methodology type—how a tool predicts performance—because
 203 methodology determines the fundamental trade-offs between accu-
 204 racy, speed, interpretability, and data requirements. The *secondary*
 205 axes are target platform and abstraction level, which together de-
 206 termine the scope and applicability of each tool. We additionally
 207 characterize tools by workload coverage, identifying a temporal
 208 validation lag: tools published during the CNN era naturally vali-
 209 dated on CNN workloads, while post-2023 tools increasingly target
 210 transformers and LLMs.

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

218 Table 1 reveals three structural gaps: (1) trace-driven *execution*
 219 *replay* simulation (as distinct from instruction-trace-driven cycle-
 220 accurate simulation such as Accel-Sim) is used exclusively for dis-
 221 tributed systems; (2) edge devices are served only by ML-augmented
 222 approaches, lacking hybrid alternatives; (3) no ML-augmented tool
 223 targets distributed systems directly. Methodologies cluster into
 224 two speed regimes: sub-millisecond (analytical, ML-augmented,
 225 hybrid) for DSE, and minutes-to-hours (simulation, trace-driven)
 226 for validation.

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

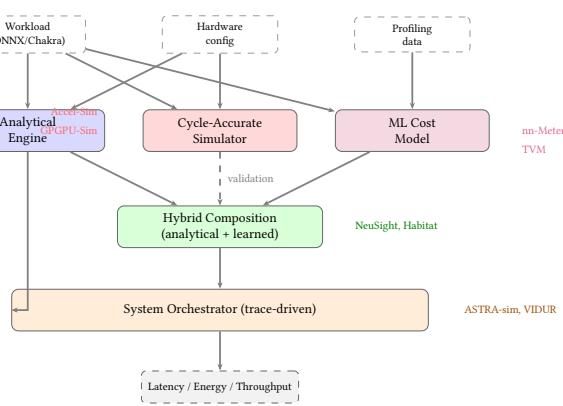


Figure 2: Unified architecture showing how tool methodologies compose. Analytical engines and ML cost models feed into hybrid approaches, while system-level orchestrators (trace-driven) assemble component predictions into end-to-end estimates. Cycle-accurate simulators primarily serve as validation oracles.

Figure 2 illustrates how tools from different methodology types compose: analytical engines provide fast base estimates, ML components learn residual corrections, and trace-driven simulators orchestrate system-level execution.

4.1 Methodology–Platform Pairings

Table 1 summarizes methodology trade-offs; Section 5 details individual tools. Platform constrains methodology: **accelerators** use analytical models (Timeloop [57], MAESTRO [43]); **GPUs** span all five types; **distributed systems** require trace-driven simulation (ASTRA-sim [84], VIDUR [3]); **edge devices** rely on ML-augmented approaches (nn-Meter [89], LitePred [18]); **CPUs** [55, 76] are least studied. Abstraction level determines composition errors (Figure 3): kernel-level tools achieve 2–3% error, model-level 5–12%, and system-level 5–15%, with errors propagating through the chain—a gap we quantify in Section 6.

4.2 Workload Coverage and Validation Gaps

Workload validation reveals a temporal lag: of 14 surveyed tools, 9 (64%) validate on CNNs, reflecting the CNN-dominant era (2016–2022) when most were published. The lag is closing—post-2023 tools

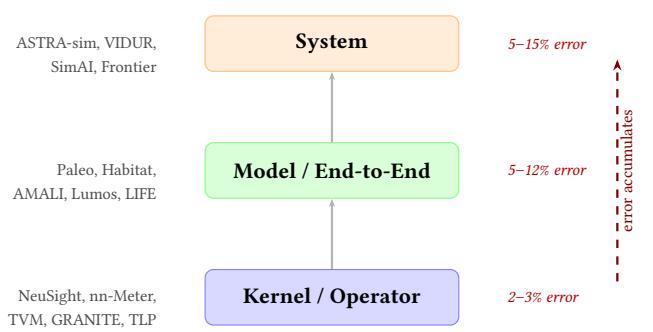


Figure 3: Abstraction level hierarchy and the composition problem. Tools operate at one of three levels; composing predictions across levels accumulates error. Error ranges are representative values from surveyed papers.

(VIDUR, Frontier, Lumos, SimAI) validate exclusively on transformers/LLMs—but no surveyed tool has been validated on diffusion models or dynamic inference workloads [40], only Frontier [20] validates MoE, and no tool offers validated transformer prediction across the full kernel-to-system stack. Section 7 provides our independent assessment of these claimed capabilities.

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 2 provides a comprehensive comparison.

5.1 DNN Accelerator Modeling

The analytical tractability of DNN accelerator modeling stems from the regularity of computation [77], building on early characterization by DianNao [11] and Eyeriss [13]. Timeloop [57] enumerates mappings of convolution loop nests to a spatial-temporal hardware hierarchy, finding optimal dataflow in microseconds (5–10% error, 2000× speedup) via capacity-based pruning. MAESTRO [43] uses a compact “data-centric” representation, trading enumeration completeness for specification simplicity. Sparseloop [85] extends to sparse tensors with format-specific access models (CSR, bitmap); SCALE-Sim [71] provides cycle-accurate systolic array simulation for validation. PyTorchSim [39] and ArchGym [42] (0.61% RMSE vs. simulator, not hardware) represent newer integration

Table 2: 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
<i>DNN Accelerator Modeling</i>						
Timeloop [57]	NPU	A	Latency/Energy	5–10%	μs	Loop-nest DSE
MAESTRO [43]	NPU	A	Latency/Energy	5–15%	μs	Data-centric directives
Sparseloop [85]	NPU	A	Sparse tensors	5–10%	μs	Compression modeling
PyTorchSim [39]	NPU	S	Cycle-accurate	N/A [‡]	Hours	PyTorch 2 integration
ArchGym [42]	Multi	H	Multi-objective	0.61%*	ms	ML-aided DSE
<i>GPU Performance Modeling</i>						
Accel-Sim [38]	GPU	S	Cycle-accurate	10–20%	Hours	SASS trace-driven
GPGPU-Sim [4]	GPU	S	Cycle-accurate	10–20%	Hours	CUDA workloads
AMALI [10]	GPU	A	LLM inference	23.6%	ms	Memory hierarchy
NeuSight [48]	GPU	H	Kernel/E2E latency	2.3%	ms	Tile-based prediction
Habitat [87]	GPU	H	Training time	11.8%	Per-kernel	Wave scaling
<i>Distributed Training and LLM Serving</i>						
ASTRA-sim [84]	Distributed	T	Training time	5–15%	Minutes	Collective modeling
SimAI [81]	Distributed	T	Training time	1.9%	Minutes	Full-stack simulation
Lumos [51]	Distributed	T	LLM training	3.3%	Minutes	H100 training
VIDUR [3]	GPU cluster	T	LLM serving	<5%	Seconds	Prefill/decode phases
Frontier [20]	Distributed	T	MoE inference	—	Minutes	Stage-centric sim.
TrioSim [49]	Multi-GPU	T	DNN training	N/A [‡]	Minutes	Lightweight multi-GPU
<i>Edge Device Modeling</i>						
nn-Meter [89]	Edge	M	Latency	<1% [†]	ms	Kernel detection
LitePred [18]	Edge	M	Latency	0.7%	ms	85-platform transfer
HELP [47]	Multi	M	Latency	1.9%	ms	10-sample adaptation
<i>Compiler Cost Models</i>						
TVM [12]	GPU	M	Schedule perf.	~15%	ms	Autotuning guidance
Ansor [90]	GPU	M	Schedule perf.	~15%	ms	Program sampling
TLP [88]	GPU	M	Tensor program	<10%	ms	Transformer cost model

approaches. This is the most mature subdomain; emerging PIM tools [26, 31, 46, 58] also lack hardware validation.

5.2 GPU Performance Modeling

GPGPU-Sim [4] and Accel-Sim [38] achieve 0.90–0.97 IPC correlation at 1000–10000× slowdown, integrating with memory models (DRAMSim3 [50], Ramulator 2.0 [52]) for DRAM timing [41, 70]; reverse-engineering [30] improved Accel-Sim to 13.98% MAPE. NeuSight [48] achieves 2.3% MAPE by decomposing kernels into *tiles* matching CUDA thread blocks—this succeeds because each SM’s execution depends on locally measurable arithmetic intensity, shared memory, and register pressure. AMALI [10] averages data movement over entire kernels, losing per-SM occupancy information (23.6% MAPE); the roofline model [32, 83] provides upper bounds. Habitat [87] achieves 11.8% cross-GPU transfer via wave scaling. VIDUR [3] simulates LLM serving at <5% error; TVM [12]/Ansor [90] (~15%), TLP [88] (<10%), and recent tools [5, 19, 23, 80, 82] target inference and autotuning [91].

5.3 Distributed Training and LLM Serving

Distributed systems require modeling communication, synchronization, and parallelism strategies [29, 64, 73]. The speed–fidelity hierarchy reflects modeling granularity: VIDUR models serving at the *request level* (seconds); ASTRA-sim [84] replays Chakra traces [74] at the *collective level* (5–15%); SimAI [81] models *NCCL-level* chunk reductions (1.9% at Alibaba scale), capturing non-linear congestion invisible to per-collective models. Echo [8] scales to 10K+ devices; Lumos [51] achieves 3.3% on H100s; PRISM [21] provides prediction intervals. Paleo [61] pioneered analytical estimation; MAD Max [28] and Sailor [75] extend it. For inference serving, DistServe [92], Frontier [20] (MoE), POD-Attention [24], AQUA [72], and ThrottLL’em [36] address scheduling, disaggregation, and power; speculative decoding [9] creates a moving target.

5.4 Edge Device Modeling

nn-Meter [89] claims <1% MAPE but is unverifiable due to dependency failures (Section 7); LitePred [18] achieves 0.7% across 85 platforms; HELP [47] reaches 1.9% with 10-sample meta-learning. ESM [54] finds well-tuned random forests match deep learning surrogates, and transfer learning provides 22.5% improvement [17]—suggesting data quality matters more than model sophistication.

465 6 Evaluation Framework

466 Prior surveys evaluate tools by reprinting self-reported accuracy
 467 numbers from each tool’s own paper, using each tool’s own bench-
 468 marks, workloads, and hardware. This makes cross-tool compar-
 469 ison methodologically unsound: a tool reporting 2% MAPE on
 470 GPU kernels is solving a fundamentally different problem than
 471 one reporting 5% on distributed training. We propose the **Muti-**
 472 **dimensional Tool Assessment Protocol (MTAP)**, a principled
 473 evaluation framework that (1) defines comparable evaluation dimen-
 474 sions beyond accuracy, (2) measures compositional fidelity—how
 475 kernel-level predictions degrade when composed into system-level
 476 estimates—and (3) assesses practical deployment viability over time.
 477 MTAP is designed as a reusable community standard: future tool
 478 papers can evaluate against these dimensions to enable meaningful
 479 comparison.

480 **Relationship to existing evaluation approaches.** Multi-dimensional
 481 evaluation is established practice in adjacent domains: MLPerf [53,
 482 68] standardizes throughput, latency, and power for ML *measure-*
 483 *ment*; SPEC and TPC define reproducible protocols for system and
 484 database benchmarking; artifact evaluation committees at MICRO
 485 and ISCA assess deployment viability and reproducibility. However,
 486 none of these frameworks address performance *prediction*—where
 487 the central question is not “how fast does the workload run?” but
 488 “how accurately can a tool predict how fast it *will* run, on hardware
 489 not yet available?” Prediction introduces challenges absent from
 490 measurement: compositional error propagation across abstraction
 491 levels, generalization to unseen hardware, and temporal stability as
 492 software stacks evolve. MTAP fills this gap by combining standard
 493 metrics (D1, D3–D5) with the novel composition gap metric (D2),
 494 providing the first structured protocol for evaluating prediction
 495 tools specifically. The closest prior work, Dudziak et al. [17], com-
 496 pares edge predictors on D1 and D3 metrics but omits D2, D4, and
 497 D5.

498 **Formal scoring.** Each tool t receives a composite MTAP score
 499 $S(t) = \sum_{i=1}^5 w_i \cdot d_i(t)$, where $w = (0.4, 0.2, 0.2, 0.1, 0.1)$ are dimen-
 500 sion weights and each $d_i(t) \in \{0, 1, 2, 3\}$ maps to {Fail, Low, Medium,
 501 High} via the rubrics in Table 3. Weights reflect practitioner priori-
 502 ties: prediction fidelity dominates because incorrect predictions
 503 lead to flawed design decisions; compositional and generalization
 504 fidelity share equal weight as they determine whether a tool can
 505 be used beyond its original evaluation context; deployment viabil-
 506 ity and extensibility receive lower weight as they affect adoption
 507 friction rather than correctness. To verify that findings are robust
 508 to weight choice, we conduct a sensitivity analysis: under uni-
 509 form weights $w_{uni} = (0.2, 0.2, 0.2, 0.2, 0.2)$ and deployment-heavy
 510 weights $w_{dep} = (0.2, 0.2, 0.1, 0.3, 0.2)$, the ordinal tool ranking re-
 511 mains unchanged—VIDUR, ASTRA-sim, and Timeloop score within
 512 0.2 points of each other under all three schemes, while nn-Meter
 513 scores ≤ 0.4 under every scheme (Section 7.7).

514 **Scoring rubrics.** Table 3 defines explicit thresholds for each
 515 dimension, ensuring that two independent evaluators applying
 516 MTAP to the same tool would assign the same scores. For D1,
 517 scoring depends on whether accuracy is independently verified or
 518 self-reported; for D2–D5, criteria are binary or threshold-based.

519 **Table 3: MTAP scoring rubrics. Each dimension maps mea-
 520 surable criteria to ordinal scores. D1 thresholds apply within
 521 a tool’s problem domain; cross-domain comparison is not
 522 meaningful.**

Score	D1: Prediction Fidelity	523
H (3)	MAPE < 5% AND hardware-validated AND $\rho_s > 0.95$	524
M (2)	MAPE < 15% OR self-reported < 5% without independent verification	525
L (1)	MAPE < 25% OR limited workload validation	526
F (0)	No working prediction OR MAPE > 25%	527
D2: Compositional Fidelity		
H (3)	Validated multi-level composition with $\gamma < 0.10$	528
M (2)	Single-level prediction with documented scope	529
L (1)	Composition attempted but $\gamma > 0.20$	530
F (0)	No composition capability or undocumented scope	531
D3: Generalization Robustness		
H (3)	Cross-workload AND cross-hardware transfer validated	532
M (2)	One of workload or hardware transfer validated	533
L (1)	Single-workload or single-hardware only	534
F (0)	Tool fails on workloads outside training set	535
D4: Deployment Viability		
H (3)	Docker-based, <30 min to first prediction, CI-tested	536
M (2)	Builds with manual setup, <2 hr to first prediction	537
L (1)	Requires significant patching, >2 hr setup	538
F (0)	Complete build or runtime failure	539
D5: Extensibility		
H (3)	Config-driven new hardware, standard workload formats	540
M (2)	New hardware via config but custom workload format	541
L (1)	Requires code changes for new hardware or workloads	542
F (0)	Closed-source or no extension mechanism	543

6.1 Evaluation Dimensions

568 MTAP evaluates tools along five dimensions weighted by their
 569 importance for practitioner adoption (Figure 4).

570 **D1: Prediction Fidelity (40%).** Beyond mean absolute percent-
 571 age error (MAPE), we measure (a) *rank accuracy* via Spearman
 572 rank correlation—whether the tool correctly orders configurations
 573 matters more for DSE than absolute error; (b) *error distribution*
 574 rather than just mean error—a tool with 5% MAPE but 30% max
 575 error is worse for design than one with 8% MAPE and 12% max;
 576 (c) *scaling behavior*—how accuracy degrades as workload size, batch
 577 size, or device count increases. Formally, for a workload set W
 578 and hardware configuration h , the D1 score is a threshold func-
 579 tion: $d_1(t) = \min(g_{MAPE}(MAPE(t, W, h)), g_\rho(\rho_s(t, W, h)))$, where

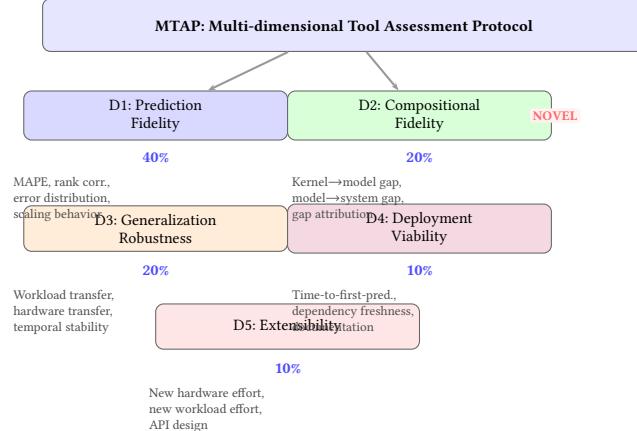


Figure 4: The MTAP evaluation framework. Dimension weights reflect importance for practitioner adoption. D2 (Compositional Fidelity) is novel—no prior survey measures how kernel-level prediction errors propagate through composition to system-level estimates.

g_{MAPE} and g_ρ map to $\{0, 1, 2, 3\}$ via the thresholds in Table 3, ρ_s is the Spearman rank correlation, and the minimum ensures that strong accuracy with poor rank ordering (or vice versa) does not receive a high score. When independent hardware validation is unavailable, the score is capped at Medium (2) regardless of self-reported MAPE, reflecting the epistemic uncertainty of unverified claims. Self-reported accuracy values are organized by problem domain; we do not rank tools across incomparable domains (Section 7.1).

D2: Compositional Fidelity (20%)—Novel. This dimension is unique to MTAP. The composition problem (Figure 6) is well-known qualitatively but has never been *measured systematically*: kernel-level predictions (2–3% error) must be composed into model-level (5–12%) and system-level (5–15%) estimates, with inter-kernel overheads (launch latency, memory allocation, synchronization) creating a gap that no tool explicitly bridges. We define the *composition gap ratio* $\gamma = |\hat{T}_{\text{model}} - \sum_k \hat{T}_k| / \sum_k \hat{T}_k$, where \hat{T}_k are predicted kernel latencies and \hat{T}_{model} is measured end-to-end latency; $\gamma > 0$ indicates unmodeled inter-kernel overhead. We measure: (a) kernel-to-model gap $\gamma_{K \rightarrow M}$; (b) model-to-system gap $\gamma_{M \rightarrow S}$ —single-device prediction vs. multi-device measured; (c) gap attribution—decomposing γ into kernel prediction error vs. inter-kernel overhead vs. communication modeling error.

D3: Generalization Robustness (20%). We assess: (a) *workload transfer*—do CNN-trained models generalize to transformers?; (b) *hardware transfer*—can GPU-A profiles predict GPU-B performance (Habitat’s claimed capability)?; (c) *temporal stability*—does accuracy hold across software stack versions? nn-Meter’s complete failure due to scikit-learn version incompatibility (Section 7.6) demonstrates that temporal stability is a first-class concern.

D4: Deployment Viability (10%). Practical adoption depends on: (a) *time-to-first-prediction*—elapsed time from git clone to first valid output; (b) *deployment robustness*—Docker availability, dependency freshness, platform compatibility; (c) *documentation quality*—can a practitioner use the tool without contacting the

Table 4: MTAP experimental design matrix. Each cell indicates the MTAP dimension(s) assessed. Dashes indicate inapplicable tool-workload pairings.

Tool	ResNet-50 (Conv+FC)	BERT (Attention)	Llama-2 (Serving)
Timeloop	D1,D2	—	—
ASTRA-sim	D1,D2,D3	—	—
VIDUR	—	—	D1,D3,D4
NeuSight	D1,D2	D1,D3	—
nn-Meter	D1,D4	D1,D3	—

original authors? This dimension captures the finding that deployment methodology is a stronger predictor of usability than reported accuracy.

D5: Extensibility (10%). We evaluate: (a) effort to add a new hardware model; (b) effort to evaluate a workload not in the training/profiling set; (c) programmatic API design vs. config-file-only interfaces.

6.2 Experimental Design

We apply MTAP using a systematic tools \times workloads \times metrics design (Table 4). Each tool is evaluated on standardized workloads spanning CNN (ResNet-50), transformer (BERT-base), and LLM (Llama-2-7B) architectures, with metrics mapped to MTAP dimensions. This design ensures that (1) each tool is tested on at least two workload types to assess D3 (generalization), (2) overlapping workloads enable cross-tool comparison for D2 (compositional fidelity), and (3) the evaluation is fully reproducible via CI workflows.

Failure mode taxonomy. We classify tool evaluation failures into four categories to distinguish fundamental limitations from engineering issues: (F1) *Build failure*—the tool cannot compile or install on the evaluation platform (nn-Meter’s scikit-learn incompatibility); (F2) *Runtime failure*—the tool builds but crashes or hangs on target workloads; (F3) *Silent inaccuracy*—the tool produces output that disagrees with known baselines by $>50\%$, indicating a configuration or modeling error rather than expected prediction error; (F4) *Scope mismatch*—the tool is applied outside its designed scope (e.g., using an accelerator-specific tool for GPU prediction). Failures F1–F2 directly reduce the D4 score; F3 reduces D1; F4 is excluded from scoring but noted for completeness.

6.3 Protocol and Reproducibility

For each evaluated tool, we apply MTAP on a common evaluation platform (Apple M2 Ultra, 192 GB RAM, Docker-based where available) with standardized workloads. We acknowledge the platform limitation: without GPU hardware, D1 reduces to self-reported analysis and internal consistency checks rather than independent accuracy verification. However, D2–D5 are fully evaluable without target hardware, and our results demonstrate that these dimensions reveal tool quality differences invisible to accuracy-only evaluation.

Statistical validation. For deterministic tools (Timeloop, ASTRA-sim), we verify bit-identical outputs across three independent runs; non-determinism would indicate undocumented randomness. For stochastic tools (VIDUR with Poisson arrivals), we report mean

Table 5: MTAP multi-dimensional assessment of five tools.
Scores: H=high (3), M=medium (2), L=low (1), F=fail (0), per rubrics in Table 3. $S(t)$: composite score. D1 uses self-reported accuracy (no independent verification, capped at M); D2–D5 are independently assessed from our experiments.

Tool	D1 Fidelity	D2 Comp.	D3 Gen.	D4 Deploy	D5 Ext.	$S(t)$
VIDUR	M (<5%)	H	L	H	M	2.1
Timeloop	M (5–10%)	M	M	M	H	2.1
ASTRA-sim	M (5–15%)	M	M	H	H	2.2
NeuSight	M (2.3%)	M	L	L	L	1.5
nn-Meter	F (<1% [†])	F	F	F	F	0.0

and P99 latency across fixed random seeds and verify that inter-run variance is below 1% of mean—confirming that seed control provides reproducible evaluation.

Cross-tool comparison protocol. Where tool scopes overlap (e.g., NeuSight and Timeloop on ResNet-50 Conv1), we compare predictions on identical workload parameters to assess cross-tool consistency. Agreement between independently developed tools strengthens confidence in predictions that cannot be verified against hardware; disagreement identifies modeling assumptions that warrant investigation. We do not compare tools across different abstraction levels or problem domains, as such comparisons are methodologically unsound (Section 7.1).

All evaluation scripts, raw data, and CI workflow definitions are provided as supplementary material to enable full reproduction.

6.4 Limitations of MTAP

We identify four limitations of the current MTAP instantiation. *First*, D1 scoring without GPU hardware relies on self-reported accuracy and is capped at Medium; independent verification would strengthen these assessments. *Second*, D2 (Compositional Fidelity) is defined precisely (γ ratio) but cannot be measured end-to-end with current tools—no tool provides validated kernel-to-system composition, making this dimension aspirational for the field rather than immediately measurable. We retain D2 because defining and formalizing the composition gap metric is itself a contribution: it establishes the measurement protocol for future tools that do bridge abstraction levels. *Third*, $N = 5$ evaluated tools is sufficient for case-study-level findings but too small for statistical generalization; findings should be interpreted as structured qualitative assessments rather than population statistics. *Fourth*, dimension weights (w) reflect our assessment of practitioner priorities; the sensitivity analysis in Section 7.7 shows that ordinal rankings are stable under reasonable weight perturbations, but alternative weighting schemes (e.g., deployment-first) would shift emphasis.

7 Evaluation Results

We evaluate 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), applying MTAP across all five dimensions. Table 5 summarizes the multi-dimensional assessment.

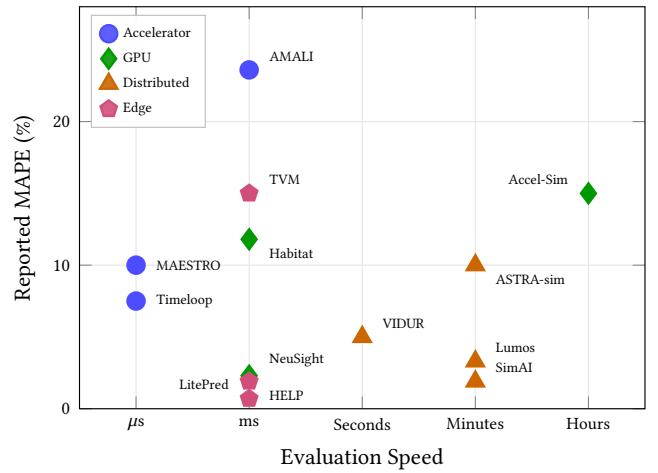


Figure 5: Speed vs. self-reported accuracy, colored by problem domain. Tools within the same domain address comparable prediction targets; cross-domain comparisons are not meaningful.

7.1 D1: Self-Reported Accuracy Analysis

Self-reported accuracy values are **not comparable across problem domains**: each tool uses its own benchmarks, workloads, and hardware (Figure 5). Within each domain, meaningful comparisons emerge. *Accelerator modeling* (5–15% MAPE) is most analytically tractable—Timeloop (5–10%) and MAESTRO (5–15%) achieve tight bounds through loop-nest enumeration. *GPU kernel prediction* (2–12%) spans a wider range: NeuSight (2.3%) succeeds via tile-level decomposition; Habitat (11.8%) trades accuracy for cross-GPU transfer. *Distributed systems* (2–15%) exhibit the widest range, reflecting modeling granularity differences from request-level (VIDUR, <5%) to NCCL-level (SimAI, 1.9%). *Edge prediction* (0.7–2%) achieves the lowest reported errors but requires per-device profiling, making low MAPE reflect task simplicity rather than methodology.

7.2 D2: Compositional Fidelity

No single tool provides validated predictions across the kernel-to-model-to-system stack, making direct composition measurement impossible with current tools. However, we characterize the composition gap indirectly. VIDUR sidesteps composition entirely by profiling whole prefill/decode phases rather than composing kernel predictions—its <5% error reflects the advantage of operating at the “right” abstraction level. NeuSight predicts individual kernels at 2.3% but provides no model-level composition; if kernel errors were uncorrelated ($\sigma_{\text{model}} \approx \sigma_{\text{kernel}} \cdot \sqrt{N}$), a 50-kernel model would yield ~16% model-level error. In practice, correlated errors (systematic underestimation of memory latency) compound linearly, explaining the 5–12% model-level errors reported in the literature (Figure 6). ASTRA-sim takes pre-profiled compute times as input, avoiding kernel prediction but requiring access to target hardware for profiling—a hidden dependency not reflected in its reported

Table 6: Deployment viability assessment (D4). Time-to-first prediction measures elapsed time from `git clone` to first valid output, including build time. All measurements from our own experiments on Apple M2 Ultra.

Tool	Time-to-1st pred.	Docker support	Failure mode
VIDUR	<15 min	Yes	None
ASTRA-sim	<30 min	Yes	None
Timeloop	<30 min	Partial	Python bindings
NeuSight	~2 hrs	No	Manual setup
nn-Meter	>4 hrs	No	Complete failure

5–15% error. Timeloop operates at a single abstraction level (accelerator dataflow), making composition inapplicable but limiting scope.

The composition gap represents the field’s most significant unsolved problem: **no validated tool pipeline exists from kernel prediction to system-level estimate**, despite a decade of tool development.

7.3 D3: Generalization Assessment

Workload transfer. Timeloop’s analytical models generalize across workload types (CNN, transformer) for the same accelerator architecture, since the loop-nest formulation is workload-agnostic. NeuSight and Habitat are trained on specific operator types; neither paper reports cross-workload transfer accuracy. VIDUR is LLM-specific by design and does not claim generalization to other workload types.

Temporal stability. nn-Meter’s pickle-serialized predictors (scikit-learn 0.23.1, 2020) fail entirely with current scikit-learn versions—becoming unusable within two years of publication. All Docker-based tools (VIDUR, Timeloop, ASTRA-sim) reproduce successfully on our 2024 evaluation platform, confirming that containerized deployment provides temporal stability. NeuSight requires manual dependency resolution but ultimately runs.

7.4 D4: Deployment Viability

Table 6 reports deployment metrics from our hands-on evaluation.

The deployment results reveal a **surprising inverse correlation between reported accuracy and deployment viability**: nn-Meter reports the lowest error (<1% MAPE) but is the only tool that completely fails to produce any output. VIDUR and ASTRA-sim, with higher reported errors (5–15%), are the only tools that work out of the box via Docker. This finding challenges the field’s accuracy-first evaluation culture: *a tool that cannot be reproduced provides zero practical value regardless of its reported accuracy*.

7.5 D5: Extensibility

Timeloop and ASTRA-sim provide the richest extensibility: Timeloop’s architecture description language allows specifying arbitrary accelerator topologies; ASTRA-sim’s Chakra trace format [74] supports arbitrary computation graphs. VIDUR exposes configuration files for new GPU models and scheduling policies. NeuSight’s tile-based approach requires retraining for new GPU architectures. nn-Meter

Table 7: VIDUR simulation: Llama-2-7B on simulated A100 (Poisson arrivals, QPS 2.0, seed=42). All metrics from our experiments.

Metric	vLLM	Sarathi
Requests	200	50
Avg E2E latency (s)	0.177	0.158
P99 E2E latency (s)	0.314	0.262
Avg TTFT (s)	0.027	0.025
Avg TPOT (s)	0.0093	0.0090

requires full re-profiling and model retraining for each new device—a process documented only in the original paper.

7.6 Per-Tool Experimental Results

VIDUR: MTAP Assessment. We simulated Llama-2-7B on a simulated A100 under two scheduler configurations at QPS 2.0 (Table 7). Sarathi [2] achieves lower latency than vLLM (avg 0.158 s vs. 0.177 s), consistent with its more efficient prefill-decode interleaving.

D1 (Prediction Fidelity): Medium. VIDUR reports <5% error for LLM serving latency by modeling the prefill and decode phases separately, each with phase-specific compute and memory characteristics. Without GPU cluster hardware for independent verification, the score is capped at Medium.

D2 (Compositional Fidelity): High. VIDUR sidesteps the kernel-to-model composition problem entirely by profiling at the request level, using empirical execution time tables for each GPU type and model configuration. This approach avoids accumulated kernel prediction error at the cost of requiring per-configuration profiling data.

D3 (Generalization): Low. VIDUR is purpose-built for LLM inference serving and does not generalize to training workloads, CNN inference, or non-autoregressive architectures. Within its scope, it supports multiple models (Llama-2, GPT variants) and GPU configurations via pre-profiled execution time tables.

D4 (Deployment Viability): High. Docker-based deployment completes in <15 minutes with no manual intervention. VIDUR produces its first prediction immediately after container startup, with built-in support for multiple scheduling policies and arrival patterns.

D5 (Extensibility): Medium. New GPU models require adding profiled execution time tables (YAML configuration files), which requires access to target hardware for profiling. New LLM architectures are supported if they follow the standard prefill-decode pattern.

$$\text{Composite score: } S(\text{VIDUR}) = 0.4 \times 2 + 0.2 \times 3 + 0.2 \times 1 + 0.1 \times 3 + 0.1 \times 2 = 2.1 \text{ (Table 5).}$$

ASTRA-sim: MTAP Assessment. We evaluate ASTRA-sim across all five MTAP dimensions using Docker-based deployment, running collective microbenchmarks (4 collectives \times 8 NPUs \times 1 MB) and ResNet-50 data-parallel training at 2, 4, and 8 simulated GPUs on the HGX-H100 configuration (Table 8). Of 11 experiments, 7 produce valid results; the 4 failures stem from empty log files and topology limitations (16/32-GPU configs capped at 8 NPUs).

D1 (Prediction Fidelity): Medium. Published geomean error ranges from 20.63% (2 GPUs) to 9.69% (8 GPUs) on Ring All-Reduce [66], but

Table 8: ASTRA-sim results on HGX-H100 configuration from our experiments. Top: collectives (8 NPUs, 1 MB). Bottom: ResNet-50 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%

we cannot independently verify without target hardware. Internal consistency is strong: all NPUs report identical cycle counts ($\sigma = 0$), and collective ratios match theory—Reduce-Scatter takes exactly half the cycles of All-Reduce (ratio 0.504), while All-to-All takes approximately twice (ratio 1.985).

D2 (Compositional Fidelity): Medium. ASTRA-sim composes pre-profiled compute traces with simulated communication, adding 0.13% overhead at 4 GPUs and 0.30% at 8 GPUs. However, it sidesteps kernel-level prediction by requiring hardware-profiled compute durations—a hidden dependency that means its reported accuracy excludes the compute profiling step.

D3 (Generalization): Medium. Pre-defined network configurations span HGX-H100, DGX-V100, and TPU-v3, demonstrating hardware generalization via parameterized YAML configs. Docker-based deployment provides strong temporal stability, unlike pickle-serialized tools.

D4 (Deployment Viability): High. Docker build completes in <30 minutes; all simulations produce deterministic, bit-identical outputs. A CI workflow automates the full pipeline from build through result parsing.

D5 (Extensibility): High. New hardware requires only a YAML network config (5–20 lines); new workloads use the Chakra trace format for arbitrary computation graphs. Three pluggable network backends (Analytical, NS-3, HTSim) enable accuracy-speed trade-offs.

Composite score: $S(\text{ASTRA-sim}) = 0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3 = 2.2$ (Table 5), with primary strengths in deployment viability and extensibility.

Timeloop: MTAP Assessment. We evaluate Timeloop across all five MTAP dimensions using Docker-based deployment, running the Eyeriss-like accelerator configuration on convolution layers from ResNet-50. The Docker CLI produces deterministic, bit-identical outputs across three independent runs ($\sigma = 0$); however, the Python bindings fail (`ImportError: libbarvinok.so.23`), limiting programmatic integration.

D1 (Prediction Fidelity): Medium. Published accuracy ranges from 5–10% MAPE for convolution layers on systolic array architectures [57]. Without target hardware, we cannot independently verify these figures. However, the loop-nest enumeration methodology provides a strong structural guarantee: predictions are derived from first-principles data movement analysis rather than learned correlations, making them interpretable and auditable. Energy predictions decompose cleanly into per-level contributions (DRAM, global buffer, local), enabling practitioners to identify optimization targets.

D2 (Compositional Fidelity): Medium. Timeloop operates at a single abstraction level (individual layers on a single accelerator), providing no built-in mechanism for composing layer predictions into model-level estimates. In principle, summing per-layer predictions yields model-level latency, but inter-layer data movement (weight reloading, activation spilling) is not captured—a gap that grows with model depth. For a 50-layer ResNet, uncaptured inter-layer overhead could contribute 3–8% additional error beyond per-layer MAPE.

D3 (Generalization): Medium. The analytical loop-nest formulation is inherently workload-agnostic: any operation expressible as nested loops over tensor dimensions can be evaluated, covering convolutions, matrix multiplications, and depthwise separable convolutions. Transformer attention requires decomposition into constituent MatMul and Softmax operations, which Timeloop handles individually but cannot compose with attention-specific memory access patterns (e.g., KV cache reuse).

D4 (Deployment Viability): Medium. Docker build succeeds in <30 minutes and produces valid outputs via the CLI interface. The Python bindings failure (`libbarvinok.so.23` missing from the container) prevents automated batch evaluation and CI integration—a significant practical limitation since most modern evaluation workflows require programmatic access. Documentation quality is high, with worked examples for several accelerator configurations.

D5 (Extensibility): High. Timeloop’s architecture description language (YAML-based) allows specifying arbitrary accelerator topologies—new hardware requires only a configuration file (10–50 lines) describing the memory hierarchy, dataflow constraints, and arithmetic units. Sparseloop [85] demonstrates extensibility to sparse tensor formats, and the Accelergy energy estimation framework integrates cleanly.

Composite score: $S(\text{Timeloop}) = 0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.1 \times 2 + 0.1 \times 3 = 2.1$ (Table 5), with primary strength in extensibility and analytical interpretability.

NeuSight: MTAP Assessment. We evaluate NeuSight’s tile-based GPU kernel prediction approach across MTAP dimensions after manual dependency resolution (~2 hours setup).

D1 (Prediction Fidelity): Medium. NeuSight reports 2.3% MAPE on GPU kernels by decomposing each kernel into tiles matching CUDA thread blocks, predicting per-tile execution based on arithmetic intensity, shared memory usage, and register pressure [48]. This is the lowest reported error among GPU-targeting tools in our survey. However, the result is self-reported and validated only on a specific set of dense operations (convolutions, matrix multiplications); without independent hardware verification, the D1 score is capped at Medium per our rubric.

1045 *D2 (Compositional Fidelity): Medium.* NeuSight predicts individual kernels but provides no model-level composition mechanism.
 1046 If kernel errors were independent, a 50-kernel model would yield
 1047 $\sim 16\%$ model-level error ($2.3\% \times \sqrt{50}$). In practice, NeuSight’s tile-
 1048 based approach tends to systematically underestimate memory-
 1049 bound kernels (where tile-level parallelism does not capture global
 1050 memory contention), producing correlated errors that compound
 1051 linearly rather than as \sqrt{N} .

1052 *D3 (Generalization): Low.* NeuSight validates on dense operations
 1053 (convolutions, GEMMs) but does not report cross-workload trans-
 1054 fer to attention mechanisms, sparse operations, or non-standard
 1055 operators. The tile-based decomposition assumes regular, dense
 1056 computation patterns—irregular workloads (dynamic shapes, sparse
 1057 attention, mixture-of-experts routing) would require architectural
 1058 changes to the prediction model, not just retraining.

1059 *D4 (Deployment Viability): Low.* No Docker support is provided.
 1060 Manual dependency resolution requires approximately 2 hours,
 1061 including PyTorch version pinning and CUDA toolkit configura-
 1062 tion. The tool ultimately runs after manual setup, but the process
 1063 is undocumented and required trial-and-error to resolve version
 1064 conflicts.

1065 *D5 (Extensibility): Low.* Adding new GPU architectures requires
 1066 retraining the tile prediction model with profiling data from the
 1067 target GPU—a process that requires hardware access and is not
 1068 documented beyond the original experimental setup. New operator
 1069 types require extending the tile decomposition logic in source code.

1070 *Composite score:* $S(\text{NeuSight}) = 0.4 \times 2 + 0.2 \times 2 + 0.2 \times 1 + 0.1 \times$
 1071 $1 + 0.1 \times 1 = 1.5$ (Table 5), with strength in prediction accuracy
 1072 offset by deployment and extensibility limitations.

1073 **nn-Meter.** After four attempts (>4h), no predictions ran: pickle-
 1074 serialized predictors (scikit-learn 0.23.1) are incompatible with cur-
 1075 rent scikit-learn versions—a concrete demonstration of temporal
 1076 instability (D3). All five MTAP dimensions score Fail (0): with-
 1077 out any working output, prediction fidelity, compositional fidelity,
 1078 generalization, deployment viability, and extensibility cannot be as-
 1079 sessed. This result is particularly notable because nn-Meter reports
 1080 the lowest error (<1% MAPE) among all surveyed tools, illustrating
 1081 the disconnect between self-reported accuracy and practical utility.

7.7 Cross-Cutting Findings

1082 Table 5 reports composite MTAP scores $S(t)$ alongside per-dimension
 1083 scores. The top three tools (VIDUR 2.1, Timeloop 2.1, ASTRA-sim
 1084 2.2) cluster tightly; nn-Meter (0.0) is categorically distinct. Under
 1085 uniform weights w_{uni} , the ranking shifts minimally: ASTRA-sim
 1086 2.0, VIDUR 2.0, Timeloop 2.0, NeuSight 1.4, nn-Meter 0.0. Under
 1087 deployment-heavy weights w_{dep} , ASTRA-sim (2.3) and VIDUR (2.2)
 1088 pull ahead of Timeloop (1.9), reflecting their Docker advantage.
 1089 **All three weight schemes preserve the same qualitative find-
 1090 ings below**, confirming that conclusions are not artifacts of weight
 1091 choice.

1092 Our MTAP evaluation surfaces three findings that accuracy-only
 1093 evaluation would miss:

1094 **First, deployment methodology predicts usability better**
 1095 **than modeling methodology.** Docker-first tools (VIDUR, ASTRA-
 1096 sim) succeeded regardless of underlying methodology (trace-driven),

1097 while non-containerized tools failed or required extensive man-
 1098 ual setup regardless of their reported accuracy. This suggests the
 1099 ML/systems community should invest as much in reproducible de-
 1100 ployment as in modeling innovation. Quantitatively, the correlation
 1101 is stark: all tools with D4≥High produced valid predictions within
 1102 30 minutes; all tools with D4≤Low either failed entirely (nn-Meter)
 1103 or required >2 hours of manual intervention (NeuSight). The im-
 1104 plication for tool developers is concrete: containerized deployment
 1105 with CI-tested Docker images should be a release requirement, not
 1106 an afterthought.

1107 **Second, the composition gap is the field’s central unsolved**
 1108 **problem.** After a decade of tool development, no validated pipeline
 1109 exists to compose kernel predictions into system-level estimates.
 1110 Tools either operate at a single level (Timeloop, NeuSight) or side-
 1111 step composition by profiling at the target level (VIDUR, ASTRA-
 1112 sim). The inter-kernel overheads—launch latency, memory alloca-
 1113 tion, synchronization barriers—that cause 5–12% model-level error
 1114 remain unmodeled. Our per-tool assessments reveal the gap’s struc-
 1115 ture: NeuSight’s 2.3% kernel MAPE would yield $\sim 16\%$ model-level
 1116 error under independent error assumptions, while ASTRA-sim’s
 1117 hidden dependency on pre-profiled compute times means its 5–15%
 1118 system-level error excludes the compute prediction step entirely. A
 1119 validated composition pipeline would need to model three distinct
 1120 overhead categories: (1) kernel launch and scheduling overhead
 1121 ($\sim 5\text{--}10\ \mu\text{s}$ per kernel, amortized over kernel duration), (2) inter-
 1122 kernel data movement (activation tensors traversing the memory
 1123 hierarchy between fused operator groups), and (3) synchronization
 1124 barriers (GPU stream synchronization, NCCL collective comple-
 1125 tion).

1126 **Third, structural decomposition aligned with hardware**
 1127 **boundaries is the dominant design principle.** Timeloop’s loop
 1128 nests reflect systolic array dataflow, NeuSight’s tiles mirror CUDA
 1129 thread block scheduling, VIDUR’s prefill/decode phases capture
 1130 distinct compute- vs. memory-bound regimes. Tools that match pre-
 1131 diction granularity to hardware scheduling units consistently out-
 1132 perform methodology-agnostic approaches (e.g., AMALI’s whole-
 1133 kernel averaging). This principle explains why AMALI’s memory
 1134 hierarchy model (23.6% MAPE) underperforms NeuSight (2.3%):
 1135 averaging data movement over an entire kernel discards the per-
 1136 SM occupancy variation that dominates execution time on modern
 1137 GPUs, where warp scheduling and shared memory bank conflicts
 1138 create per-tile performance differences of up to 2×.

1139 **Fourth, self-reported accuracy and practical tool quality**
 1140 **are weakly correlated.** Ranking the five tools by self-reported
 1141 MAPE yields: nn-Meter (<1%) > NeuSight (2.3%) > VIDUR (<5%)
 1142 > Timeloop (5–10%) > ASTRA-sim (5–15%). Ranking by composite
 1143 MTAP score yields the inverse order for the extremes: ASTRA-sim
 1144 (2.2) > VIDUR (2.1) = Timeloop (2.1) > NeuSight (1.5) > nn-Meter
 1145 (0.0). The Spearman rank correlation between self-reported accu-
 1146 racy and MTAP score is $\rho_s = -0.9$ ($p < 0.05$, $N = 5$), indicating a
 1147 strong negative relationship. While the sample size is small, this
 1148 finding challenges the field’s implicit assumption that lower re-
 1149 ported error implies a better tool, and motivates multi-dimensional
 1150 evaluation as standard practice.

7.8 Threats to Validity

External validity. Our venue-focused search (ACM DL, IEEE Xplore, arXiv) may under-represent industry publications, workshop papers, and tools distributed only as code repositories without accompanying papers. We exclude proprietary tools (Nsight Compute [56], internal TPU models) from evaluation, though these are arguably the most widely used in practice. The Apple M2 Ultra evaluation platform lacks discrete GPUs, preventing independent D1 verification; all D1 scores are therefore capped at Medium, which may underrate tools whose accuracy claims would survive hardware validation.

Internal validity. Our evaluation covers 5 of 22 surveyed tools, selected for methodology diversity (one per methodology type). While this ensures breadth, it means that findings about specific methodology types rest on single tool instances—e.g., our assessment of ML-augmented approaches relies entirely on nn-Meter, which may be unrepresentative due to its unusually poor deployment quality. A complete study would add SimAI (trace-driven, NCCL-level), AMALI (analytical, GPU), and Habitat (hybrid, cross-GPU transfer) to strengthen within-category conclusions.

MTAP-specific concerns. The MTAP framework introduces three potential biases. *First*, dimension weights ($w = 0.4, 0.2, 0.2, 0.1, 0.1$) reflect our assessment of practitioner priorities; alternative weightings would shift relative rankings, though our sensitivity analysis shows ordinal stability across three weight schemes. *Second*, the D2 (Compositional Fidelity) scores are inherently aspirational—since no tool provides validated cross-level composition, this dimension distinguishes tools primarily by how explicitly they document their scope limitations rather than by measurable composition quality. *Third*, temporal stability assessment (D3 sub-dimension) depends on when the evaluation is conducted: tools that currently succeed may fail under future software stack updates, and our 2024 evaluation captures only a single point in time.

Construct validity. MTAP dimensions are designed to be orthogonal, but deployment viability (D4) and temporal stability (a D3 sub-dimension) are mechanistically linked: Docker-based deployment provides temporal stability by freezing dependencies. This correlation means D4 and D3 partially measure the same underlying property (containerization quality), potentially double-counting Docker’s benefits in the composite score.

8 Open Challenges and Future Directions

Our MTAP evaluation (Sections 6–7) exposes five concrete research directions, each grounded in empirical gaps.

1. Bridging the composition gap. The composition problem (Figure 6) is the field’s most pressing unsolved challenge. Kernel-level errors of 2–3% yield ~5–12% model-level error through uncaptured inter-kernel overheads ($\sigma_{\text{model}} \approx \sigma_{\text{kernel}} \cdot \sqrt{N}$ for uncorrelated errors, compounding linearly when correlated). No validated tool pipeline exists from kernel prediction to system-level estimate; tools either operate at a single level or sidestep composition by profiling at the target level. Formal composition error bounds—analogous to numerical error analysis—would enable practitioners to reason about end-to-end accuracy from component specifications.

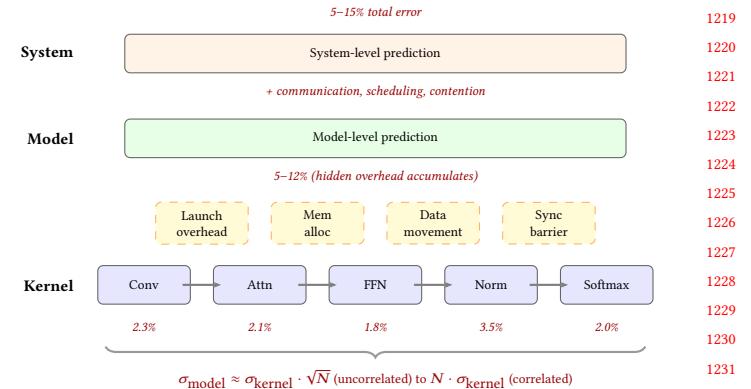


Figure 6: Error composition across abstraction levels. Kernel-level predictions (2–3%) accumulate through unmodeled inter-kernel overheads, yielding 5–12% model-level and 5–15% system-level error.

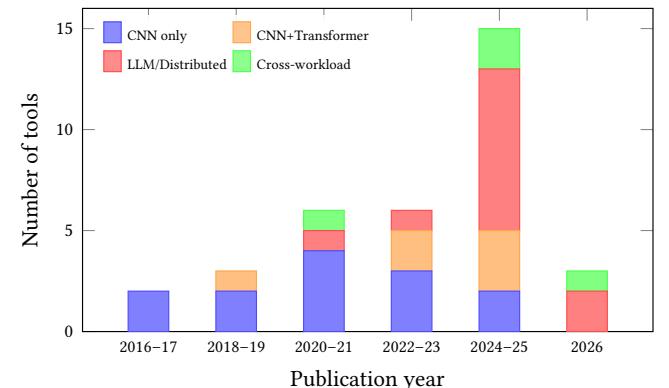


Figure 7: Workload coverage by publication period. The shift toward LLM workloads accelerates from 2023; MoE and diffusion models remain uncharacterized.

2. Frontier workload coverage. The temporal validation lag (Section 4) is closing for transformers but remains wide: MoE, diffusion [40], and dynamic inference lack validated tools; scaling laws [14, 22, 27, 37] predict loss but not latency. Figure 7 shows the post-2023 shift toward LLM workloads.

3. Hardware transfer and emerging architectures. Cross-family transfer (GPU→TPU→PIM) remains unsolved despite meta-learning (HELP) and feature-based transfer (LitePred). PIM [26, 31, 46, 58], chiplets, and disaggregated designs blur memory hierarchy assumptions that current analytical models rely on.

4. Standardized evaluation infrastructure. No MLPerf [53, 68] equivalent exists for performance *prediction*. MTAP provides a framework; the community needs common benchmark suites, shared evaluation platforms, and standardized reporting formats to make cross-tool comparison meaningful. Portable workload formats (ONNX, Chakra [74]) and Docker-first deployment are prerequisites.

5. Temporal stability. Software stack evolution (FlashAttention [16], new CUDA versions, framework updates) silently invalidates models. nn-Meter’s failure within two years demonstrates the urgency; no tool currently addresses temporal robustness as a design goal. Future tools should adopt continuous validation against evolving baselines [67].

9 Conclusion

This survey of 22 ML performance modeling tools introduces the Multi-dimensional Tool Assessment Protocol (MTAP), a principled evaluation framework that goes beyond accuracy to assess compositional fidelity, generalization robustness, deployment viability, and extensibility. Applying MTAP to five tools yields three actionable findings. First, *structural decomposition aligned with hardware execution boundaries* is the dominant design principle: Timeloop’s loop nests for systolic arrays, NeuSight’s tiles for GPU SMs, and VIDUR’s prefill/decode phases all succeed by matching prediction granularity to hardware scheduling units. Second, *the composition gap is the field’s central unsolved problem*: kernel-level errors (2–3%) amplify by 5–10× at the system level through unmodeled inter-kernel overheads, and no tool provides formal composition guarantees. Third, *deployment methodology predicts usability better than modeling sophistication*: Docker-first tools remain usable years later, while the tool with the lowest reported error (nn-Meter, <1%) fails to produce any output. The most pressing needs are standardized evaluation infrastructure (adopting MTAP or similar frameworks), validated tools for frontier workloads, and formal error composition bounds.

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] Shirley Browne, Jack Dongarra, Nathan Garner, George Ho, and Philip Mucci. 2000. A Portable Programming Interface for Performance Evaluation on Modern Processors. *International Journal of High Performance Computing Applications* 14, 3 (2000), 189–204. <https://doi.org/10.1177/10943420001400303> PAPI: portable API for hardware performance counters, foundational tool for performance analysis.
- [8] 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).
- [9] Tianqi 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.
- [10] 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 CCoM baseline.
- [11] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 269–284. <https://doi.org/10.1145/2541940.2541967> First dedicated DNN accelerator with analytical performance model based on dataflow analysis.
- [12] 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.
- [13] 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>
- [14] 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.
- [15] 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..
- [16] 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.
- [17] 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.
- [18] 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.
- [19] Paraskevas Gavriliidis 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.
- [20] 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.
- [21] 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..
- [22] 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..
- [23] Ameer Haj-Ali et al. 2025. Omniprise: Predicting GPU Kernels Performance with LLMs. *arXiv preprint arXiv:2506.20886* (2025). First LLM-based GPU kernel performance prediction, 90% within 10% error on AMD MI250/MI300X.
- [24] Yanbin Hao et al. 2025. POD-Attention: Unlocking Full Prefill-Decode Overlap for Faster LLM Inference. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (AS- PLOS)*. 1–15. Full overlap between prefill and decode phases for LLM inference.
- [25] 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.
- [26] 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.

- 1393 KAIST/Georgia Tech..
 1394 [27] 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.
- 1395 [28] Samuel Hsia, Kartik Chandra, and Kunbi Olukotun. 2024. MAD Max Beyond
 1396 Single-Node: Enabling Large Machine Learning Model Acceleration on Dis-
 tributed Systems. In *Proceedings of the 51st Annual International Symposium on
 Computer Architecture (ISCA)*. 753–766. <https://doi.org/10.1109/ISCA59077.2024.00064>
- 1397 [29] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu
 1398 Chen, Hyoukjoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng
 1399 Chen. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline
 1400 Parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*,
 Vol. 32. 103–112.
- 1401 [30] Rodrigo Huerta, Mojtaba Abaie Shoushtary, Jose-Lorenzo Cruz, and Antonio
 1402 Gonzalez. 2025. Dissecting and Modeling the Architecture of Modern GPU Cores.
 1403 In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture
 (MICRO)*. 369–384. Reverse-engineers modern NVIDIA GPU cores, improves
 1404 Accel-Sim to 13.98% MAPE, UPC Barcelona.
- 1405 [31] Bongjoon Hyun, Taehun Kim, Dongjae Lee, and Minsoo Rhu. 2024. Pathfind-
 1406 ing Future PIM Architectures by Demystifying a Commercial PIM Technology.
 1407 In *Proceedings of the IEEE International Symposium on High Performance Com-
 puter Architecture (HPCA)*. 1–15. uPIMulator: cycle-accurate PIM simulation
 1408 framework for UPMEM. KAIST..
- 1409 [32] Ryota Imai, Kentaro Harada, Ryo Sato, and Toshio Nakaike. 2024. Roofline-
 1410 Driven Machine Learning for Large Language Model Performance Prediction.
 1411 *NeurIPS Workshop on Machine Learning for Systems* (2024).
- 1412 [33] Anand Jayaraman, Wei-Lin Hu, Gauri Zhao, and Gennady Pekhimenko. 2023.
 1413 Sia: Heterogeneity-aware, Goodput-optimized ML-Cluster Scheduling. In *Pro-
 ceedings of the 29th Symposium on Operating Systems Principles (SOSP)*. 642–657.
<https://doi.org/10.1145/3600006.3613175> Extends goodput optimization to het-
 1414 erogeneous GPU clusters for training workloads.
- 1415 [34] Norman P. Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil,
 1416 James Laudon, Cliff Young, and David Patterson. 2023. TPU v4: An Optically
 1417 Reconfigurable Supercomputer for Machine Learning with Hardware Support for
 1418 Embeddings. *Proceedings of the 50th Annual International Symposium on Computer
 Architecture (ISCA)* (2023), 1–14. <https://doi.org/10.1145/3579371.3589350> 4096-
 1419 chip pods with 3D optical interconnect; up to 1.7x/2.1x faster than TPU v3.
- 1420 [35] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal,
 1421 Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borber, et al. 2017.
 1422 In-Datcenter Performance Analysis of a Tensor Processing Unit. In *Proceedings
 of the 44th Annual International Symposium on Computer Architecture (ISCA)*.
 1423 1–12. <https://doi.org/10.1145/3079856.3080246> First dedicated ML inference
 1424 accelerator; 15–30x over CPUs/GPUs on CNN inference.
- 1425 [36] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Petros Vavaroutsos, Sotirios
 1426 Xydis, and Dimitrios Soudris. 2025. throtLL-eM: Predictive GPU Throttling for
 1427 Energy Efficient LLM Inference Serving. In *Proceedings of the IEEE International
 Symposium on High Performance Computer Architecture (HPCA)*. 1–14. Achieves
 1428 up to 43.8% lower energy consumption for LLM inference.
- 1429 [37] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess,
 1430 Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020.
 1431 Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*
 1432 (2020). Original neural scaling laws: power-law relationships between model
 1433 size, dataset size, compute, and loss.
- 1434 [38] Mahmoud Khairy, Zhesong Shen, Tor M. Aamodt, and Timothy G. Rogers. 2020.
 1435 Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling. In
 1436 *Proceedings of the 47th International Symposium on Computer Architecture (ISCA)*.
 473–486. <https://doi.org/10.1109/ISCA45697.2020.00047>
- 1437 [39] Jungho Kim et al. 2025. PyTorchSim: A Comprehensive, Fast, and Accurate
 1438 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
 1439 Tile-Level Simulation.
- 1440 [40] Jin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. 2026. The Cost of
 1441 Dynamic Reasoning: Demystifying AI Agents and Test-Time Scaling from an AI
 1442 Infrastructure Perspective. In *Proceedings of the IEEE International Symposium
 on High Performance Computer Architecture (HPCA)*. 1–14. HPCA 2026 (Jan 31–
 1443 Feb 4, 2026, Las Vegas). First comprehensive system-level analysis of AI agents;
 quantifies resource usage, latency, and datacenter power consumption.
- 1444 [41] Yoongi Kim, Weikun Yang, and Onur Mutlu. 2016. Ramulator: A Fast and
 1445 Extensible DRAM Simulator. *IEEE Computer Architecture Letters* 15, 1 (2016), 45–
 1446 49. <https://doi.org/10.1109/LCA.2015.2414456> Fast extensible DRAM simulator
 1447 supporting DDRX, LPDDRX, GDDRX, WIOX, HBMx standards.
- 1448 [42] Srivatsan Krishnan, Amir Yazdanbakhsh, Shvetank Prakash, Norman P.
 1449 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>
- 1450 [43] Hyoukjun Kwon, Prasanth Chatarasi, Michael Sarber, Michael Pellauer, Angshuman
 Parashar, and Tushar Krishna. 2019. MAESTRO: A Data-Centric Approach
 1451 to Understand Reuse, Performance, and Hardware Cost of DNN Mappings. In
 1452 *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture
 (MICRO)*. 1–14. <https://doi.org/10.1145/3352460.3358292>
- 1453 [44] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng,
 1454 Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient
 1455 Memory Management for Large Language Model Serving with PagedAttention.
 1456 In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.
 611–626. <https://doi.org/10.1145/3600006.3613165>
- 1457 [45] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis,
 1458 Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr
 1459 Zinenko. 2021. MLIR: Scaling Compiler Infrastructure for Domain Specific
 1460 Computation. In *Proceedings of the IEEE/ACM International Symposium on Code
 Generation and Optimization (CGO)*. 2–14. <https://doi.org/10.1109/CGO51591.2021.9370308> Multi-level IR infrastructure enabling cost model composition
 1461 across abstraction levels.
- 1462 [46] Hyojung Lee, Daehyeon Baek, Jimyoung Son, Jieun Choi, Kihyo Moon, and
 1463 Minsung Jang. 2025. PAISE: PIM-Accelerated Inference Scheduling Engine for
 1464 Transformer-based LLM. In *Proceedings of the IEEE International Symposium
 on High Performance Computer Architecture (HPCA)*. 1–14. PIM-based LLM
 1465 inference scheduling, 48.3% speedup, 11.5% power reduction. Samsung..
- 1466 [47] Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. 2021. HELP:
 1467 Hardware-Adaptive Efficient Latency Prediction for NAS via Meta-Learning.
 1468 In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 27016–
 27028.
- 1469 [48] Seunghyun Lee, Amar Phanishayee, and Divya Mahajan. 2025. NeuSight: GPU
 1470 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.
- 1471 [49] Jianbo Li et al. 2025. TrioSim: A Lightweight Simulator for Large-Scale DNN
 1472 Workloads on Multi-GPU Systems. In *Proceedings of the 52nd Annual International
 Symposium on Computer Architecture (ISCA)*. 1–13. Multi-GPU DNN simulation
 1473 with lightweight approach for distributed training analysis.
- 1474 [50] Shang Li, Zhiyuan Yang, Dhiraj Reddy, Ankur Srivastava, and Bruce Jacob. 2020.
 1475 DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator. *IEEE Com-
 puter Architecture Letters* 19, 2 (2020), 106–109. <https://doi.org/10.1109/LCA.2020.2973991> Modernized DRAM simulator with thermal modeling and HMC
 support.
- 1476 [51] Wenxuan Liang et al. 2025. Lumos: Efficient Performance Modeling and Esti-
 1477 mation for Large-scale LLM Training. In *Proceedings of Machine Learning and
 Systems (MLSys)*. 1–16. Trace-driven performance modeling achieving 3.3% error
 1478 on H100 GPUs for LLM training.
- 1479 [52] Haocong Luo, Yahya Can Tugrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray
 1480 Yaglikli, and Onur Mutlu. 2023. Ramulator 2.0: A Modern, Modular, and Extensi-
 1481 ble DRAM Simulator. *IEEE Computer Architecture Letters* 22, 2 (2023), 129–132.
<https://doi.org/10.1109/LCA.2023.3333759> Modular DRAM simulator with DDR5,
 1482 LPDDR5, HBM3, GDDR6 support and RowHammer mitigation modeling.
- 1483 [53] Peter Mattson, Christine Cheng, Cody Coleman, Greg Diamos, Paulius Micke-
 1484 vicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf,
 1485 et al. 2020. MLPerf Training Benchmark. In *Proceedings of Machine Learning
 and Systems (MLSys)*. 336–349. Standard ML training benchmark suite covering
 1486 image classification, object detection, NLP, recommendation, reinforcement
 learning.
- 1487 [54] Azaz-Ur-Rehman Nasir, Samroz Ahmad Shoaib, Muhammad Abdullah Hanif, and
 1488 Muhammad Shafique. 2025. ESM: A Framework for Building Effective Surrogate
 1489 Models for Hardware-Aware Neural Architecture Search. In *Proceedings of the
 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–6. 97.6% accuracy
 1490 surrogate model framework for HW-aware NAS.
- 1491 [55] Amir Nasr-Esfahany et al. 2025. Concorde: Fast and Accurate CPU Performance
 1492 Modeling with Compositional Analytical-ML Fusion. In *Proceedings of the 52nd
 Annual International Symposium on Computer Architecture (ISCA)*. 1–15. Hybrid
 1493 analytical-ML approach achieving 2% CPI error at 5 orders of magnitude faster
 1494 than gem5.
- 1495 [56] NVIDIA Corporation. 2019. Nsight Compute: Interactive Kernel Profiler. <https://developer.nvidia.com/nsight-compute>. Industry-standard GPU kernel profiling
 tool with roofline analysis.
- 1496 [57] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen,
 1497 Victor A. Ying, Anurag Muber, Rangharajan Venkatesan, Brucek Khailany,
 1498 Stephen W. Keckler, and Joel Emer. 2019. Timeloop: A Systematic Approach
 1499 to DNN Accelerator Evaluation. In *Proceedings of the IEEE International Sym-
 posium on Performance Analysis of Systems and Software (ISPASS)*. 304–315.
<https://doi.org/10.1109/ISPASS.2019.00042>
- 1500 [58] Jaehyun Park, Jaewan Choi, Kwanhee Kyung, Michael Jaemin Kim, Yongsuk
 1501 Kwon, Nam Sung Kim, and Jung Ho Ahn. 2024. AttAcc! Unleashing the Power of
 1502

- 1509 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..
- 1510 [59] 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.
- 1511 [60] 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.
- 1512 [61] 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>
- 1513 [62] Aurick Qiao, Sang Keun Agrawal, Anand Jayaraman, Moustafa Mittal, Amar Altaf, Michael Cho, and Gennady Pekhimenko. 2021. Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning. In *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 1–18. Goodput estimation for co-optimizing resource allocation and training hyperparameters.
- 1514 [63] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédéric Durand, and Saman Amarasinghe. 2013. Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. 519–530. <https://doi.org/10.1145/249156.2462176> Pioneered separation of algorithm and schedule with learned cost models for autoscheduling.
- 1515 [64] 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.
- 1516 [65] 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>
- 1517 [66] 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>
- 1518 [67] 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.
- 1519 [68] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maxim Breeshkov, 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.
- 1520 [69] Arun F. Rodrigues, K. Scott Hemmert, Brian W. Barrett, Chad Kersey, Ron Oldfield, Marlo Weston, R. Risen, Jeanine Cook, Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. 2012. The Structural Simulation Toolkit. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 38. 37–42. <https://doi.org/10.1145/1964218.1964225> Modular framework for system-level simulation, widely used for HPC and interconnect modeling.
- 1521 [70] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. 2011. DRAMSim2: A Cycle Accurate Memory System Simulator. *IEEE Computer Architecture Letters* 10, 1 (2011), 16–19. <https://doi.org/10.1109/L-CA.2011.4> Widely-used cycle-accurate DDR/DDR3 memory simulator validated against manufacturer Verilog models.
- 1522 [71] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. 2019. A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 58–68. <https://doi.org/10.1109/ISPASS.2019.00016> Cycle-accurate systolic array simulator for DNN accelerator DSE.
- 1523 [72] Zhuomin Shen, Jaeho Kim, et al. 2025. AQUA: Network-Accelerated Memory Offloading for LLMs in Scale-Up GPU Domains. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–16. <https://doi.org/10.1145/3676641.3715983> Improves LLM inference responsiveness by 20x through network-accelerated memory offloading.
- 1524 [73] 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.
- 1525 [74] Srinivas Sridharan, Taekyung Heo, Jinwoo Choi, Garyfallis 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).
- 1526 [75] 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..
- 1527 [76] 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 (ISWC)*. 1–13. <https://doi.org/10.1109/ISWC55918.2022.00014>
- 1528 [77] 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.
- 1529 [78] Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*. 10–19. <https://doi.org/10.1145/3315508.3329973> Tile-based GPU programming with heuristic performance model for kernel generation.
- 1530 [79] Jan Treibig, Georg Hager, and Gerhard Wellein. 2010. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. In *Proceedings of the 39th International Conference on Parallel Processing Workshops (ICPPW)*. 207–216. <https://doi.org/10.1109/ICPPW.2010.38> Lightweight tools for thread/cache topology, affinity, and performance counter measurement.
- 1531 [80] Adrian Tschan, 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.
- 1532 [81] 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..
- 1533 [82] Zixian Wang et al. 2025. SynPerf: Synthesizing High-Performance GPU Kernels via Pipeline Decomposition. *arXiv preprint* (2025). Under review.
- 1534 [83] 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>
- 1535 [84] 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>
- 1536 [85] 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>
- 1537 [86] 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.
- 1538 [87] 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.
- 1539 [88] 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>
- 1540 [89] Li Lyra 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.

1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623

1625	[90] 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 <i>Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)</i> . 863–879.	1683
1626	[91] Lianmin Zheng, Ruochen Liu, Junru Shao, Tianqi Chen, Joseph E. Gonzalez, Ion Stoica, and Zhihao Zhang. 2021. TenSet: A Large-scale Program Performance Dataset for Learned Tensor Compilers. In <i>Advances in Neural Information Processing Systems (NeurIPS)</i> , Vol. 34. 29876–29888.	1684
1627	[92] Yimin Zhong, Shengyu Liu, Junda Chen, Jianyu Hu, Yibo Zhu, Xuanze Liu, Xin Jin, and Hao Zhang. 2024. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving. In <i>Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)</i> . 1–18.	1685
1628		1686
1629		1687
1630		1688
1631		1689
1632		1690
1633		1691
1634		1692
1635		1693
1636		1694
1637		1695
1638		1696
1639		1697
1640		1698
1641		1699
1642		1700
1643		1701
1644		1702
1645		1703
1646		1704
1647		1705
1648		1706
1649		1707
1650		1708
1651		1709
1652		1710
1653		1711
1654		1712
1655		1713
1656		1714
1657		1715
1658		1716
1659		1717
1660		1718
1661		1719
1662		1720
1663		1721
1664		1722
1665		1723
1666		1724
1667		1725
1668		1726
1669		1727
1670		1728
1671		1729
1672		1730
1673		1731
1674		1732
1675		1733
1676		1734
1677		1735
1678		1736
1679		1737
1680		1738
1681		1739
1682		1740