POSTER: TPE: AI-MicroBMT: A Unified Benchmark Tool for Deployment-Aware Performance Characterization of Diverse AI Accelerators

Jonghyun Shin
jonghyun_shin@snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Dongmyong Shin
demyshin@snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Jeongnam Youn
jnyoun@snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Dae-Hwan Kim
kimdh@ssc.ac.kr
Suwon Science College
Hwaseong, Republic of Korea

Soojung Ryu
sue.ryu@snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Xuan Truong Nguyen*
truongnx@capp.snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Hyuk-Jae Lee*
hjlee@capp.snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Abstract

Comparing usable performance is hard due to diverse accelerator architectures and vendor deployment paths. AI-MicroBMT unifies evaluation with a shared architecture and a hardware abstraction layer (HAL). A feasibility gate verifies deployment, quality retention, and CPU-relative latency. The Unified Deployment Score (UDS) summarizes coverage, performance, quality, resolution robustness, and optional efficiency via user-weighted components. Across 67 classification models and a resolution sweep, feasibility depends on the backend and optimal devices shift by deployment profile. UDS enables deployment choices under explicit constraints and priorities.

CCS Concepts

• **General and reference** → **Evaluation**; • **Computer systems organization** → *Embedded systems.*

Keywords

AI accelerators, microbenchmarking, deployment feasibility, hardware abstraction layer.

1 Introduction

Accelerator memory, parallelism, compilation, post-training quantization (PTQ), and runtimes make identical source models differ in deployability, quality, latency, and throughput. Compilation success alone cannot establish useful acceleration. MLPerf Inference standardizes scenarios and quality constraints [8]. AI-MicroBMT provides a shared architecture and hardware abstraction layer (HAL), and deployment-oriented interpretation through a feasibility gate and Unified Deployment Score (UDS). The platform is available at https://ai-bmt.com; deployment scripts, analysis code, and measurement manifests are at https://github.com/kinsingo/AI-MicroBMT.

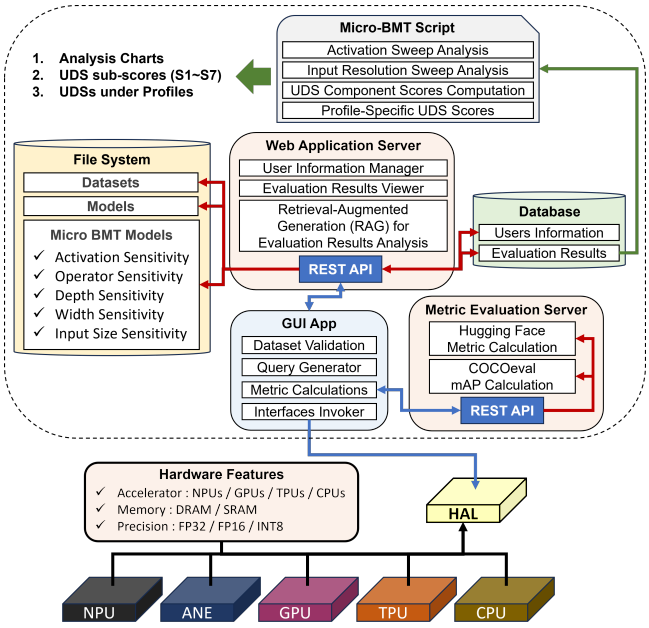


Figure 1: AI-MicroBMT architecture. Shared evaluation invokes device-specific HALs; logged results feed UDS component and profile analysis.

2 Methods

2.1 AI-MicroBMT Architecture

Figure 1 shows shared dataset validation, query generation, HAL execution, quality evaluation, and logging. Single-stream queries issue one sample; offline enables supported batching, asynchronous execution, or multicore parallelism. Logged models, scenarios, toolchains, deployment settings, hosts, and outcomes feed analysis scripts computing UDS components and profile scores.

*Corresponding authors: truongnx@capp.snu.ac.kr and hjlee@capp.snu.ac.kr.

This work is licensed under a Creative Commons Attribution 4.0 International License.
PACT '26, Chicago, IL, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2915-7/2026/10
https://doi.org/10.1145/3838684.3846853

2.2 Hardware Abstraction Layer

Backends implement model loading, device-specific preprocessing/data layout, and inference calls. Some NPUs require PTQ or a compilation step; preprocessing adapts validated inputs. The platform supplies common inputs, query semantics, metrics, and logging. Six non-CPU backends reuse this evaluation logic across Linux/macOS and Arm/x86-64 without modifying the core pipeline.

2.3 Feasibility Gate and UDS

Feasibility Gate. For device d and task t, let c indicate deployment success, A denote Top-1 accuracy, and L denote single-stream latency. With Apple M4 CPU as baseline, speedup is $s_{d,t} = L_{\text{cpu},t}/L_{d,t}$ and relative accuracy drop is $\Delta_{d,t} = 100(A_{\text{cpu},t} - A_{d,t})/A_{\text{cpu},t}$. The base-suite gate is

$$I^{\text{base}}_{d,t} = \mathbf{1}\begin{Bmatrix} c_{d,t} = 1, & A_{d,t} \geq A_{\min}, \\ \Delta_{d,t} \leq \tau_{\text{acc}}, & s_{d,t} \geq \theta \end{Bmatrix}.$$

We set $A_{\min} = 15\%$, $\tau_{\text{acc}} = 4\%$, and $\theta = 1$: an absolute guard against near-zero quality, relative accuracy-drop tolerance, and CPU-parity latency. These thresholds are configurable.

UDS Components. S_1 measures gate-passing coverage. Among passing tasks, S_2 measures capped, normalized CPU-relative single-stream speedup; S_4 measures accuracy retention; S_5 measures CPU-relative offline throughput gain. S_3 combines resolution coverage with scaling from single-stream to offline relative to each model's best scaling, preventing reliance on one favorable input size. Optional S_6 measures offline throughput per reported peak compute to expose unrealized capacity; S_7 measures offline throughput per watt to expose power-inefficient speed. All components lie in $[0, 1]$; normalization caps depend on the device set.

Aggregate scores. For device d, the base score is $\text{UDS}_{\text{base}}(d; \tau_{\text{acc}}, \theta) = \sum_{i=1}^{5} w_i S_i(d)$, $\sum_i w_i = 1$. When comparable metadata are available, the extended score is $\text{UDS}_{\text{ext}}(d; \tau_{\text{acc}}, \theta) = \sum_{i=1}^{7} \tilde{w}_i S_i(d)$, $\sum_i \tilde{w}_i = 1$. Components that are excluded are assigned a weight of zero, and the weights of all remaining components must add up to 1.

3 Results

Setup. The 67-model base suite comprises 47 architecture variants across diverse model families, plus 20 activation variants of MobileNetV2 [9] and ResNet-18 [3], evaluated at 224 × 224 on 10,000 ImageNetV2-TopImages samples [7]. S_3 sweeps MobileNetV2, ResNet-50, and ResNet-101 over 26 input sizes (48–448, step 16). Measurements use the C++ HAL with warmup. CPU/ANE/RTX use FP32-source ONNX models; other NPUs use PTQ or compilation with their own runtime paths. Hosts, runtimes, and deployed precision differ, characterizing deployed systems and toolchains.

Feasibility. Table 1 separates latency, quality, and deployment failures. ANE incurs no quality/deployment failures, yet 13 models are slower than the CPU, leaving 54/67 feasible. DeepX M1 and Hailo-8 each have 48 feasible cases but differ in the balance of latency, quality, and deployment failures. Execution-success counts would hide this distinction.

Table 1: Feasibility over 67 models under the canonical gate (θ=1, τ_{acc}=4%, $A_{\min}$=15%). Assign C_4 first for compile/PTQ/runtime failure or $A < A_{\min}$; otherwise, C_3 exceeds the accuracy-drop tolerance, C_2 passes quality but misses CPU parity, and C_1 passes the gate ($S_1 = C_1/67$).

Accelerator	C_1	C_2	C_3	C_4
DeepX M1	**48**	5	3	11
Mobilint-ARIES	**63**	0	4	0
Hailo-8	**48**	1	6	12
Qualcomm QCS6490	**44**	9	12	2
Apple M4 ANE	**54**	13	0	0
RTX PRO 6000 Max-Q	**67**	0	0	0

Table 2: Highest-scoring devices under each UDS profile (score; weights). Fixed inputs disable S_3 ($w_3 = 0$); extended profiles use $w_6 = w_7$. ANE is excluded from both comparisons because comparable standalone power is unavailable.

Profile	Input	Base UDS	Extended UDS
Accuracy-first	Var.	ARIES (.813) (.15, .20, .15, .30, .20)	DeepX M1 (.804) (.13, .13, .13, .18, .13, .15, .15)
Accuracy-first	Fixed	RTX (.887) (.18, .24, 0, .34, .24)	DeepX M1 (.785) (.16, .16, 0, .21, .16, .155, .155)
Latency-first	Var.	ARIES (.789) (.15, .30, .15, .20, .20)	DeepX M1 (.786) (.13, .18, .13, .13, .13, .15, .15)
Latency-first	Fixed	RTX (.848) (.18, .34, 0, .24, .24)	DeepX M1 (.767) (.16, .21, 0, .16, .16, .155, .155)
Throughput-first	Var.	ARIES (.830) (.15, .20, .15, .20, .30)	DeepX M1 (.795) (.13, .13, .13, .13, .18, .15, .15)
Throughput-first	Fixed	RTX (.878) (.18, .24, 0, .24, .34)	ARIES (.787) (.16, .16, 0, .16, .21, .155, .155)

Profile sensitivity. Table 2 favors ARIES for variable-input base profiles and RTX for fixed inputs. Extended profiles favor DeepX M1 except for fixed-input throughput priority, which favors ARIES. Scores under different weights are not direct performance differences. This profile dependence is the intended behavior of UDS: it exposes deployment-priority sensitivity rather than prescribing a universal accelerator ranking.

Efficiency profiles use vendor metadata [1, 2, 4–6]: S_6 uses RTX FP32 peaks versus NPU INT8 TOPS; S_7 uses specified TDP, maximum power, or range upper bounds. These are neither equal-precision efficiency nor uniformly measured energy comparisons. UDS is a deployment decision aid conditioned on the suite, normalization, metadata, and priorities.

Acknowledgments

This work was partly supported by Institute of Information & Communications Technology Planning and Evaluation (IITP) grants funded by the Korea government (MSIT) under the Development of a Benchmark Test (BMT) Platform Technology for Edge AI Semiconductor (RS-2024-00399936) and the Graduate School of Artificial Intelligence Semiconductor (IITP-2026-RS-2023-00256081).

References

[1] DEEPX. [n. d.]. DX-M1. Product specifications. Retrieved September 8, 2026 from https://deepx.ai/products/dx-m1/

[2] Hailo Technologies Ltd. 2023. *Hailo-8 M.2 Key M ET Module Data Sheet.* Rev. 2.2. https://premio.blob.core.windows.net/premio/uploads/resource/data-sheet/Hailo-8/Hailo-8%20M.2%20Key%20M%20ET%20Datasheet%20Rev2.2.pdf

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 770–778. doi:10.1109/CVPR.2016.90

[4] Mobilint, Inc. [n. d.]. ARIES. Product specifications. Retrieved September 8, 2026 from https://www.mobilint.com/aries

[5] NVIDIA Corporation. 2025. *NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition.* Datasheet 3519233. https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/workstation-datasheet-blackwell-rtx-pro-6000-max-q-nvidia-3519233.pdf

[6] Qualcomm Technologies, Inc. 2025. *Qualcomm Dragonwing QCS6490/QCM6490 Processors.* Product brief 87-28733-1 Rev. F. https://docs.qualcomm.com/doc/87-28733-1/87-28733-1_REV_E_QUALCOMM_QCS6490_QCM6490_Processors_Product_Brief.pdf

[7] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do ImageNet Classifiers Generalize to ImageNet?. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 5389–5400. https://proceedings.mlr.press/v97/recht19a.html

[8] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, Ramesh Chukka, Cody Coleman, Sam Davis, Pan Deng, Greg Diamos, Jared Duke, Dave Fick, J. Scott Gardner, Itay Hubara, Sachin Idgunji, Thomas B. Jablin, Jeff Jiao, Tom St. John, Pankaj Kanwar, David Lee, Jeffery Liao, Anton Lokhmotov, Francisco Massa, Peng Meng, Paulius Micikevicius, Colin Osborne, Gennady Pekhimenko, Arun Tejusve Raghunath Rajan, Dilip Sequeira, Ashish Sirasao, Fei Sun, Hanlin Tang, Michael Thomson, Frank Wei, Ephrem Wu, Lingjie Xu, Koichi Yamada, Bing Yu, George Yuan, Aaron Zhong, Peizhao Zhang, and Yuchen Zhou. 2020. MLPerf Inference Benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 446–459. doi:10.1109/ISCA45697.2020.00045

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4510–4520. doi:10.1109/CVPR.2018.00474