AIBench Training: Balanced Industry-Standard AI Training Benchmarking

Fei Tang[1,3], Wanling Gao[1,2,3], Jianfeng Zhan[1,2,3], Chuanxin Lan[1], Xu Wen[1], Lei Wang[1,2,3], Chunjie Luo[1,3], Zheng Cao[4], Xingwang Xiong[1], Zihan Jiang[1], Tianshu Hao[1], Fanda Fan[1], Fan Zhang[1], Yunyou Huang[2], Jianan Chen[1], Mengjia Du[1], Rui Ren[1], Chen Zheng[1], Daoyi Zheng[5], Haoning Tang[6], Kunlin Zhan[7], Biao Wang[8], Defei Kong[9], Minghe Yu[10], Chongkang Tan[11], Huan Li[12], Xinhui Tian[13], Yatao Li[14], Junchao Shao[15], Zhenyu Wang[16], Xiaoyu Wang[17], Jiahui Dai[2], and Hainan Ye[2]

[1]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
{tangfei, gaowanling, zhanjianfeng, lanchuanxin, wenxu, wanglei_2011, luochunjie}@ict.ac.cn
[2]International Open Benchmark Council (BenchCouncil)
[3]University of Chinese Academy of Sciences
[4]Alibaba, zhengzhi.cz@alibaba-inc.com
[5]Baidu, zhengdaoyi@baidu.com
[6]Tencent, haoningtang@tencent.com
[7]58.com, zhankunlin@58.com
[8]NetEase, bjwangbiao@corp.netease.com
[9]ByteDance, kongdefei@bytedance.com
[10]Zhihu, yuminghe@zhihu.com
[11]Lenovo, tanck1@lenovo.com
[12]Paypal, huanli1@paypal.com
[13]Moqi, xinhuit@moqi.ai
[14]Microsoft Research Asia, yatli@microsoft.com
[15]JD.com, shaojunchao@imdada.cn
[16]CloudTa, wangzhenyu@cloudta.com.cn
[17]Intellifusion, wang.xiaoyu@intellif.com

***Abstract*—Earlier-stage evaluations of a new AI architecture/system need affordable AI benchmarks. Only using a few AI component benchmarks like MLPerf alone in the other stages may lead to misleading conclusions. Moreover, the learning dynamics are not well understood, and the benchmarks' shelf-life is short. This paper proposes a balanced benchmarking methodology. We use real-world benchmarks to cover the factors space that impacts the learning dynamics to the most considerable extent. After performing an exhaustive survey on Internet service AI domains, we identify and implement nineteen representative AI tasks with state-of-the-art models. For repeatable performance ranking (RPR subset) and workload characterization (WC subset), we keep two subsets to a minimum for affordability. We contribute by far the most comprehensive AI training benchmark suite.**

The evaluations show: (1) AIBench Training (v1.1) outperforms MLPerf Training (v0.7) in terms of diversity and representativeness of model complexity, computational cost, convergent rate, computation, and memory access patterns, and hotspot functions; (2) Against the AIBench full benchmarks, its RPR subset shortens the benchmarking cost by 64%, while maintaining the primary workload characteristics; (3) The performance ranking shows the single-purpose AI accelerator like TPU with the optimized TensorFlow framework performs better than that of GPUs while losing the latter's general support for various AI models. The specification, source code, and performance numbers are available from the AIBench homepage https://www.benchcouncil.org/aibench-training/index.html.

***Index Terms*—Benchmark; Deep Learning; Learning Dynamics; Training; Subsetting; Workload Characterization;**

Jianfeng Zhan is the corresponding author.

I. Introduction

The AI advancements have brought breakthroughs in processing images, video, speech, and audio [1]–[3], boosting industry-scale deployments of massive AI algorithms, systems, and architectures. Unfortunately, the AI training's learning dynamics are dynamic, volatile, and unpredictable: a slight change of models, hyper-parameters, or optimizing strategies may influence the final accuracy or the training convergence rate, which is not understood well theoretically. Meanwhile, the benchmarks' shelf-life is short as state-of-the-art AI models evolve very fast. These situations raise severe AI benchmarking challenges.

First, the prohibitive cost of training a state-of-the-art AI model raises a serious benchmarking challenge. Some mixed-

978-1-7281-8643-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPASS51385.2021.00014

TABLE I
COMPARISON OF AIBENCH TRAINING AND MLPERF TRAINING

		AIBench Training v1.1	MLPerf Training v0.7
Methodology		Balanced methodology considering conflicting requirements	According to commercial and research relevance
Task		Nineteen tasks and models	six tasks and eight models
Dataset		Text, image, 3D, audio, and video data	Text and image data
Algorithm behavior	Computation	0.09 to 282830 MFLOPs	0.21 to 29000 MFLOPs
	Complexity	0.03 to 110 million parameters	5.2 to 110 million parameters
	Optimizer categories	5	5
	Loss function categories	14	6
System behavior	Hotspot functions	30	9
	Convergence	6 to 96 epochs	3 to 49 epochs
Micro-architecture behavior	Achieved occupancy	0.12 to 0.61	0.12 to 0.54
	IPC efficiency	0.02 to 0.77	0.02 to 0.74
	Gld efficiency	0.28 to 0.94	0.52 to 0.85
	Gst efficiency	0.27 to 0.98	0.75 to 0.98
	DRAM utilization	0.08 to 0.61	0.08 to 0.61

precision optimizations indeed improve traditional performance metrics like throughput. Still, they adversely affect the final model's quality, which we can only observe by running an entire training session [4], [5]—training an AI model (a component benchmark) to achieve a state-of-the-art quality target. So running an entire training session is mandatory. Unfortunately, it is prohibitively costly, often taking several weeks to run a complete training session on a small-scale system. Furthermore, the architecture community heavily relies upon simulations with slowdowns varying wildly from 10X to 1000X, which further exaggerates the challenge.

Second, there are conflicting benchmarking requirements: affordable vs. comprehensive in different stages. On the one hand, earlier-stage evaluations of a new architecture or system need affordable AI benchmarks to reduce the portability cost. Meanwhile, affordable benchmarks are also necessary to provide valuable performance implications in ranking off-the-shelf systems or architectures for promoting its adoption.

On the other hand, later-stage evaluations or purchasing off-the-shelf systems need detailed evaluations using comprehensive benchmarks to avoid benchmarketing [6], and using a few AI component benchmarks like MLPerf alone may lead to misleading or unfair conclusions in the other stages. For example, our experiments find that TPU reflects hugely high performance for Image Classification, while supports limited models officially considering the huge portability cost, which is not the case for GPUs. Meanwhile, the initial design input or workload characterization needs to consider various computation and memory access patterns to avoid over-optimization for some specific workloads.

There are other shelf-life, scalability, and repeatability challenges, which we detail in Section II-A. AIBench is a systematic AI benchmarking project tackling the challenges mentioned above. It distills and abstracts real-world application scenarios across Datacenter, HPC [7], IoT [8], and Edge [9], into the scenario [10], training, inference [11], micro [12], and synthetic benchmarks [13], which we detail in Section II-B. This paper focuses on AIBench Training and its subsets.

We present a balanced methodology to meet conflicting benchmarking requirements in different stages. We use real-world benchmarks to cover the factors space that impacts the learning dynamics to the most considerable extent. The factors which we consider include the commonly used building blocks, model layers, loss functions, optimizers, FLOPs, and different-scale parameter sizes. We identify and include nineteen representative AI tasks from one of the essential domains– Internet Services to guarantee the benchmarks' representativeness and diversity.

We keep two subsets for repeatable performance ranking (RPR subset) and workload characterization (WC subset) to a minimum for affordability. The criteria for the RPR subset are the diversity of model complexity, computational cost, convergence rate, repeatability, and having widely-accepted metrics or not. The criteria for the WC subset are of micro-architectural characteristics. We currently consider occupancy, IPC, global load, global store, and dram utilization, which are GPU architecture-dependent. Table I summarizes the differences of AIBench Training v1.1 against MLPerf Training v0.7 [1].

Our contributions are as follows:

- We propose a balanced benchmarking methodology that meets conflicting requirements in different stages.
- We perform by far the most comprehensive workload characterization on AIBench (v1.1) and MLPerf (v0.7). We found that the nineteen benchmarks of AIBench reflect distinct and different computation and memory access patterns from that of MLPerf. AIBench outperforms MLPerf in terms of the diversity and representativeness of model complexity, computational cost, convergent rate, computation, memory access patterns, and hotspot functions. Over MLPerf, AIBench reduces the benchmarking cost while avoiding error-prone design or benchmarketing.
- We perform a performance ranking of nine state-of-the-practice AI accelerators using the AIBench RPR subset. The ranking list shows the single-purpose AI accelerator like TPU with the optimized TensorFlow framework performs better than that of GPUs while losing the latter's general support for a variety of AI models.

The rest of this paper is organized as follows. Section II summarizes the challenges, methodology, and related work. Section III presents the design and implementation. Section IV presents the workload characterization. Section V presents performance ranking. Section VI draws a conclusion.

II. CHALLENGES, METHODOLOGY, AND RELATED WORK

[1]We run a smaller dataset for Recommendation in MLPerf v0.7, and the original dataset is too large to run on a single GPU card.

TABLE II
REPRESENTATIVE AI TASKS IN INTERNET SERVICE DOMAIN.

Internet Service	Core Scenario	Involved AI Problem Domain
Search Engine	Content-based image retrieval (e.g., face, scene)	Object detection; Classification; Spatial transformer; Face embedding; 3D face recognition
	Advertising and recommendation	Advertising; Recommendation
	Maps search and translation	3D object reconstruction; Text-to-Text translation; Speech recognition; Neural architecture search; Nature Language Processing
	Data annotation and caption (e.g., text, image)	Text summarization; Image-to-Text
	Search result ranking	Learning-to-rank
	Image resolution enhancement	Image generation; Image-to-Image
	Data storage space and transfer optimization	Image compression; Video prediction
Social Network	Friend or community recommendation	Recommendation; Face embedding; 3D face recognition
	Vertical search (e.g., image, people)	Classification; Spatial transformer; Object detection; Nature Language Processing
	Language translation	Text-to-Text translation; Neural architecture search
	Automated data annotation and caption	Text summarization; Image-to-Text; Speech recognition
	Anomaly detection (e.g., spam image detection)	Classification
	Image resolution enhancement	Image generation; Image-to-Image
	Photogrammetry (3D scanning)	3D object reconstruction
	Data storage space and transfer optimization	Image compression; Video prediction
	News feed ranking	Advertising; Learning-to-rank
E-commerce	Product searching	Classification; Spatial transformer; Object detection; Nature Language Processing
	Product recommendation and advertising	Recommendation
	Language and dialogue translation	Text-to-Text translation; Speech recognition; Neural architecture search
	Automated data annotation and caption	Text summarization; Image-to-Text
	Virtual reality (e.g., virtual fitting)	3D object reconstruction; Image generation; Image-to-Image
	Data storage space and transfer optimization	Image compression; Video prediction
	Product ranking	Learning to rank
	Facial authentication and payment	Face embedding; 3D face recognition;

This section presents challenges, methodology, and related work.

A. AI Benchmarking Challenges

Currently, state-of-the-art AI models evolve very fast. The learning dynamics of how huge parameters and system configurations impact are not fully understood, which raises serious AI benchmarking challenges.

First, the prohibitive cost of training a state-of-the-art AI model raises a serious benchmarking challenge (Cost Challenge #1). Besides, there are conflicting benchmarking requirements in different stages (Conflicting Requirements Challenge #2). We detail these two challenges in Section I.

There are massive AI tasks and models. Meanwhile, an AI model's shelf-life is short [14] as the state-of-the-art models evolve very fast. Intuitively, to improve affordability, one option is to single out the representative subset from massive AI tasks and models. It is necessary to develop a reasonable AI benchmarking methodology to justify the choices for and update AI tasks, models, and data sets. Another option is to provide different sizes of data inputs. However, a smaller dataset may significantly impact model accuracy, stability, convergence, etc. Another option is to profiling hotspot functions of AI workloads as microbenchmark. The microbenchmarks are affordable to perform a fair comparison of competing systems by isolating hardware and software from statistical optimizations [5], but it can not reflect learning dynamics.

Second, the benchmarks' shelf-life is short [14] as the state-of-the-art models evolve very fast, which raises another challenge (Shelf-life Challenge #3). However, it often takes more than one year to walk through benchmark design, implementation, community adoption, and wide-scale testing. A synthetic benchmark like ParaDNN (2020) [14] seems useful as it can traverse more expansive parameter spaces. Unfortunately, it can not model learning dynamics: using ParaDNN, we can not observe an entire-training session.

Third, an AI task's problem scale is often fixed, which raises a scalability challenge (Scalability Challenge #4). The current benchmarking practices show the largest-scale system can finish an ImageNet training around several ten seconds, which can not constantly stress the system test. Unfortunately, designing a scalable AI benchmark is not trivial. It is essential to model learning dynamics when scaling up a problem size with a synthetic benchmark.

Finally, the benchmark mandates being repeatable [7], while deep learning's nature is stochastic, allowing multiple different but equally valid solutions [5] (Repeatability Challenge #5). Previous work manifests HPC AI's uncertainty by run-to-run variation in terms of epochs-to-quality [7] and the effect of scaling training on time-to-quality [5]. Run-to-run variation of an AI model depends on many factors, such as the model's initial state, the order of data traversal, and the gradient descent algorithms. The variation between different AI models depends mainly on the model structures.

B. AIBench Methodology

AIBench distills and abstracts real-world application scenarios across Datacenter, HPC (HPC AI500), IoT, and Edge, into the scenario, training, inference, micro, and synthetic benchmarks, as shown in Fig. 1.

AIBench Scenario benchmarks [10] are proxies to industry-scale real-world applications scenarios. Each scenario benchmark models the critical paths of a real-world application scenario as a permutation of the AI and non-AI modules. Edge AIBench [9] is an instance of the scenario benchmark suites, modeling end-to-end performance across IoT, edge, and Datacenter.

AIBench Training and AIBench Inference [11] cover nineteen representative AI tasks with state-of-the-art models to guarantee diversity and representativeness. AIBench Micro [12] provides the intensively-used hotspot functions, profiled from AIBench Training and Inference, for simulation-based architecture researches. AIBench Training provides two subsets for repeatable performance ranking (RPR subset) and workload characterization (WC subset) to improve affordability; It keeps the benchmark subsets to a minimum while maintaining representativeness.

Based on the AIBench Training RPR subset, we provide HPC AI500 [7], [15] to evaluate large-scale HPC AI systems. AIoTBench [8] implements AIBench Inference on various IoT and embedded devices, emphasizing diverse light-weight AI frameworks and models. Using a whole-picture workload characterization methodology [16], we present the AIBench Inference subset from a perspective of inherent workload characterization.

As complementary to real-world benchmarks, we plan to provide several synthetic benchmarks (AIBench Synthetic [13]) with scalable problem sizes to model learning dynamics to tackle Challenges #3-4. We aim to achieve scalability, which supports auto-generation of diverse models with different combinations of building blocks and connections, to investigate the impact of varying model structures.

The rest of this subsection details the AIBench Training Methodology. AIBench Training adopts a balanced AI benchmarking methodology considering comprehensiveness, representativeness, affordability, and portability. Our methodology widely investigates AI tasks and models and covers the algorithm-level, system-level, and microarchitectural-level factors space to the most considerable extent. From the algorithm level, we consider the commonly used building blocks, model layers, loss functions, optimizers, FLOPs, and different-scale parameter sizes; From the system level, we evaluate the convergent rate and hot functions. From the microarchitectural level, we identify their computation and memory access patterns. We provide real-world AI workloads to achieve comprehensiveness and representativeness. Besides, we propose two AIBench Training subset: RPR and WC subsets to achieve affordability. We give the hotspot functions as microbenchmarks (AIBench Micro) to achieve portability for simulator-based research after profiling.

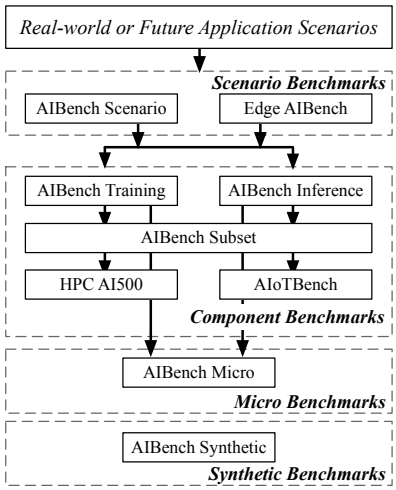

Fig. 1. AIBench Methodology.

1) Performing a detailed survey of the critical domain rather than a rough survey of a variety of domains: As it is impossible to investigate all AI domains, we single out one of the essential AI domains–Internet services for the detailed survey. In cooperation with seventeen prominent industry partners, our survey covers diverse business types like Internet services, streaming videos, machine translation, Q&A community, online payment, etc.

2) Include as most as possible representative benchmarks: We believe the prohibitive cost of training a model to a state-of-the-art quality cannot justify including only a few AI benchmarks. Instead, using only a few AI component benchmarks may lead to error-prone design: over-optimization for some specific workloads or benchmarketing.

For Internet services, to the best of our knowledge, we identify and include as most as possible representative AI tasks, models, and datasets into the benchmark suite to guarantee benchmarks' representativeness and diversity. Meanwhile, we consider the primitives' diversity in the AI models.

The past successful benchmark practice also witnesses this strategy. The cost of execution time for other benchmarks like HPC [17], SPECCPU [18] on simulators, is also prohibitively costly. However, the representativeness and coverage of a widely accepted benchmark suite are paramount important. For example, SPECCPU 2017 [18] contains 43 benchmarks. The other examples include PARSEC3.0 (30) [19], TPC-DS (99) [20].

3) Keep the benchmark subsets to a minimum: For two different purposes, we choose two minimum subsets according to different criteria: We choose the RPR subset based on diversity of model complexity, computational cost, convergence rate, repeatability, having the widely-accepted metrics or not, and choose the WC subset according to the representativeness of system or micro-architecture characteristics.

Using the subset for ranking is also witnessed by the past practice. For example, Top500 [21]–a super computer ranking–

only reports HPL [17] and HPCG [22]–two benchmarks out of 20+ representative HPC benchmarks like HPCC [23], NPB [24].

4) Consider the full benchmarks, their subsets, and microbenchmarks as indispensable: Different stages have conflicting benchmarking requirements. The initial design inputs to a new system/architecture need comprehensive workload characterization. For earlier-stage evaluations of a new system or architecture, which even adopts simulation-based methods, heavy benchmarking is a significant burden. Thus, concise, portable, and lightweight benchmarks are of great significance. While later-stage evaluations of a new architecture or system or purchasing a commercial off-the-shelf one need detailed evaluations using comprehensive benchmarks to avoid error-prone design or benchmarketing.

For initial design inputs, we perform detailed workload characterization. For later-stage evaluations of or purchasing a new system/architecture, we run the full benchmarks or selectively run some benchmarks to locate the bottlenecks quickly. We run an affordable subset for earlier-stage evaluations of a new system/architecture or ranking commercial off-the-shelf systems/architectures.

C. Related Work

DawnBench (2017) [4] is the first benchmark that notices the necessity of performing so-called end-to-end benchmarking. It proposes time-to-accuracy as the main metric, which requires training a model to state-of-the-art accuracy. AIBench (2018) [10], [46], [47] is the first benchmark that notices the necessity of modeling the critical paths of a real-world application scenario. ParaDNN (2020) [14] is the first synthetic AI benchmark. The HPL-AI (2019) [48] is a micro benchmark that uses mixed-precision LU decomposition to achieve upper bound FLOPS performance. It is scalable, but LU decomposition is not relevant to most of the AI workloads in AIBench [7]. Both ParaDNN and HPL-AI can not model the learning dynamics without considering the model quality.

AIBench and MLPerf are two systematic AI benchmarking projects. They are concurrent and complemental. The AIBench suites are by far the most comprehensive AI benchmark suites tackling Challenges #1-5 mentioned above. MLPerf (2019) [5] includes seven benchmarks for training and five benchmarks for inference. MLPerf performs the most large-scale testing, but it fails to present a benchmarking methodology to justify the choice for and update AI tasks, models, and data sets. Besides, they fail to consider the conflicting requirements, shelf-life, scalability challenges (Challenges #2-4) mentioned above.

Other related benchmark projects include DeepBench (2016) [49], BenchNN (2012) [50], BenchIP (2018) [51], Fathom (2016) [52] and TBD (2018) [53]. They either consist of microbenchmarks [49], or only evaluate throughput while ignoring model quality [52], [53] or only include simple shallow neural networks such as multi-layer perceptron [50].

III. The Benchmark Designs and Implementations

This section illustrates the benchmark designs and implementations, including selections of workloads, datasets, quality targets (Section III-A), metrics, and reference implementations (Section III-B). For all benchmarks, we set the target qualities according to the experiments and referenced papers.

A. The Benchmark Decisions

To cover a broad spectrum of representative AI tasks in Internet services, we thoroughly analyze the essential application scenarios among three primary Internet services, including search engines, social networks, and e-commerce, as shown in Table II. In total, we identify nineteen representative AI tasks, for each of which we implement a state-of-the-art model as a component benchmark, as shown in Table III. Our benchmarks are constantly evolving and keep updating to use state-of-the-art models. Due to space limitation, we give a brief introduction.

Image Classification is a fundamental AI task to classify an image into multiple categories, which the industry widely uses.

Object Detection aims to find objects of certain target classes with precise localization in each image and be one of the most important tasks in computer vision.

Learning-to-Rank is to train models for ranking tasks using machine learning methods.

Image Generation is to generate similar images with the input image.

Text-to-Text Translation is to translate a sequence of words into another language and is one of the most representative tasks in natural language processing.

Image-to-Text is to generate a description for each image automatically.

Image-to-Image Translation is to learn image mapping of two different domains, which we use for style conversion, object conversion, seasonal conversion, and photo enhancement.

Speech Recognition is to recognize speech audio and translate it into a text.

Face Embedding is to verify a face by learning an embedding into the Euclidean space.

3D Face Recognition performs identification of 3D face images.

Recommendation is essential in the industry and widely used in commercial advertisements.

Video Prediction predicts the effect of physical interactions.

Image Compression aims to reduce the cost of storage or transmission using deep learning.

3D Object Reconstruction is to capture a real object's shape and appearance.

Text Summarization generates a headline or a summary and is an essential task in natural language processing.

Spatial Transformer provides spatial transformation capabilities, which we can use to process distorted and deformed objects.

TABLE III
COMPONENT BENCHMARKS IN AIBENCH.

No.	Component Benchmark	Algorithm	Dataset	Target Quality
TrC1	Image Classification	ResNet50 [25]	ImageNet	74.9% (accuracy)
TrC2	Image Generation	WassersteinGAN [26]	LSUN	N/A
TrC3	Text-to-Text translation	Transformer [27]	WMT English-German	55% (accuracy)
TrC4	Image-to-Text	Neural Image Caption Model [28]	Microsoft COCO	4.2 (perplexity)
TrC5	Image-to-Image Translation	CycleGAN [29]	Cityscapes	N/A
TrC6	Speech Recognition	DeepSpeech2 [30]	Librispeech	23.5% (WER)
TrC7	Face Embedding	Facenet [31]	VGGFace2, LFW	90% (accuracy)
TrC8	3D Face Recognition	3D face models [32]	77,715 samples from 253 face IDs	94.64% (accuracy)
TrC9	Object Detection	Faster R-CNN [33]	VOC2007	76% (mAP)
TrC10	Recommendation	Neural collaborative filtering [34]	MovieLens	63.5% (HR@10)
TrC11	Video Prediction	Motion-Focused predictive models [35]	Robot pushing dataset	72 (MSE)
TrC12	Image Compression	Recurrent neural network [36]	ImageNet	0.99 (MS-SSIM)
TrC13	3D Object Reconstruction	Convolutional encoder-decoder network [37]	ShapeNet dataset	45.83% (IU)
TrC14	Text Summarization	Sequence-to-sequence model [38]	Gigaword dataset	41 (Rouge-L)
TrC15	Spatial Transformer	Spatial transformer networks [39]	MNIST	99% (accuracy)
TrC16	Learning-to-Rank	Ranking distillation [40]	Gowalla	14% (accuracy)
TrC17	Neural Architecture Search	Reinforcement learning [41]	PTB [42]	100 (perplexity)
TrC18	Advertising	DLRM [43]	Kaggle Display Advertising Challenge Dataset [44]	78.9% (accuracy)
TrC19	Nature Language Processing (NLP)	BERT [45]	Wikipedia	71.2% (accuracy)

Neural Network Search automatically designs neural networks.

Advertising is to display the most relevant ads to customers.

Natural Language Processing (NLP) is to train a language model, which we use for many tasks like translation and question answer.

B. Reference Implementations and Metrics

Reference Implementations. AIBench Training provides reference implementations and corresponding running scripts, datasets, and monitoring tools for each component benchmark. For each AI task, we provide implementations on both TensorFlow [1] and PyTorch frameworks.

Metrics. AIBench Training uses time-to-quality as a metric, which is the wall clock time in training a model to achieve a target quality [4].

IV. EVALUATION

This section compares AIBench Training (v1.1) against MLPerf Training (v0.7) (in short AIBench vs. MLPerf) from the perspectives of the model and micro-architectural characteristics (Section IV-B), quantify the run-to-run variation, and measure their benchmarking cost (Section IV-C). Also, we propose the RPR and WC subsets to achieve affordability and representativeness for repeatable performance ranking and workload characterization (Section IV-D), respectively. Further, we characterize micro-architectural behaviors from the perspectives of runtime breakdown, hotspot functions, and stall analysis (Section IV-E).

TABLE IV
HARDWARE CONFIGURATION DETAILS.

CPU Configurations			
CPU Type		Intel CPU Core	
Intel ®Xeon E5-2620 v3		12 cores@2.40G	
L1 DCache	L1 ICache	L2 Cache	L3 Cache
12 × 32 KB	12 × 32 KB	12 × 256 KB	15MB
Memory	Ethernet	Hyper-Threading	
64GB, DDR3	1Gb	Disabled	
GPU Configurations v1: Nvidia Titan XP			
Cuda Cores	3840	Memory	12GB, GDDR5X
GPU Configurations v2: Nvidia Titan RTX			
Cuda Cores	4608	Memory	24GB, GDDR6

A. Experimental Configuration

We conducted experiments on two servers equipped with different GPUs: We use the TITAN XP GPUs for workload characterization and perform training experiments on the TITAN RTX GPUs. The other configurations of the servers are the same, and the configurations of GPUs and servers are shown in Table IV. The operating system is ubuntu 16.04 with the Linux kernel 4.4, and the other software is CUDA 10, python 3.7, and PyTorch 1.10. In the rest of this section, we mainly evaluate the reference PyTorch implementations of AIBench in Section IV-C2. The AIBench Traning homepage shows the performance numbers using other frameworks.

B. The Comparison of AIBench against MLPerf

1) Model Characteristics: In this subsection, we first characterize the AIBench and MLPerf benchmarks' model characteristics from the perspectives of model complexity, computational cost, and convergent rate.

The characterization approach is like that of [54]. The differences have two points. First, they only evaluate different AI models of the same Image Classification task, and we assess twenty AI tasks and twenty-four models, which are the union set of AIBench and MLPerf. Second, they report Top-1 accuracy, and we report the convergent rate—the cost of training a model.

We use the total amount of learnable parameters, FLOPs of a single forward computation, and the number of epochs to achieve a target quality (e.g., accuracy) to characterize the above three characteristics, respectively. We use the OpCounter [55] to estimate the FLOPs and learnable parameters for both AIBench and MLPerf benchmarks. Since the tool can not count some operations, the reported numbers may be smaller than the actual ones. We do not report the numbers of the reinforcement learning model for both AIBench (Neural Architecture Search) and MLPerf (Game) shown in Table I because the FLOPs and learnable parameters vary significantly from different epochs. We do not report the values of BERT and DLRM models for both AIBench (NLP, Advertising) and MLPerf (NLP, Recommendation) because the training is iteration-based, not epoch-based. The value of Video Prediction is not reported due to the limited support of OpCounter.

For each benchmark of AIBench and MLPerf, we train the model to achieve a target quality. Specifically, the target quality is 74.9% (accuracy) for Image Classification, 55% (accuracy) for Text-to-Text translation, 4.2 (perplexity–the smaller is the better) for Image-to-Text, 23.5% (WER–the smaller is the better) for Speech Recognition, 90% (accuracy) for Face Embedding, 94.59% (accuracy) for 3D Face Recognition, 76.7% (mAP) for Object Detection, 60% (HR@10) for Recommendation, 72 (MSE–the smaller is the better) for Video Prediction, 0.99 (MS-SSIM) for Image Compression, 45% (IU) for 3D Object Reconstruction, 41 (Rouge-L) for Text Summarization, 99% (accuracy) for Spatial Transformer, 13.9% (accuracy) for Learning-to-Rank, 100 (perplexity) for Neural Architecture Search, 78.9% for Advertising, and 71.2% for Nature Language Processing. For the MLPerf benchmarks, the target quality is 37.7 (BBOX) for Object Detection (heavy), 22.47 (mAP) for Object Detection (light), 22.21 (BLEU) for Translation (recurrent), 25.25 (BLEU) for Translation (nonrecurrent). Note that AIBench and MLPerf use the same models and datasets for Image Classification, NLP, and Advertising (Recommendation task in MLPerf), so their numbers for these tasks are consistent in the rest of this paper.

Fig. 2 shows the model characteristics. From the computation cost perspective, AIBench ranges from 0.09 to 282830 M-FLOPs, while MLPerf varies from 0.213248 to 24500 M-FLOPs–a much narrower range. From the perspective of model complexity, the number of learnable parameters of AIBench ranges from 0.03 million to 68.4 million, while MLPerf only covers a range of 5.2 to 49.53 million. From the convergent rate perspective, the required epochs of AIBench range from 6 to 96, while MLPerf only covers a range of 3 to 49. Thus, only using MLPerf cannot cover the diversities of different AI models.

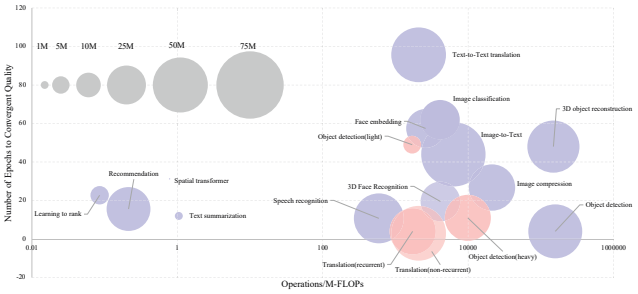

Fig. 2. The Comparisons of AIBench against MLPerf from the Perspectives of Model Complexity, Computational Cost, and Convergent Rate.

Object Detection and 3D Object Reconstruction have the gigantic FLOPs, while Learning-to-Rank has the smallest. Image-to-Text is the most complex model, while the Spatial Transformer is the least. Text-to-text translation requires the largest epochs to converge, while the remaining models converge within 60 epochs.

Then we further investigate the optimizer and loss function categories. From the perspective of optimizers, both AIBench and MLPerf cover five optimizers, which are adam, adamw, RMSprop, SGD, and lamb for AIBench, and adam, lamb, lars, lazy, and SGD for MLPerf. From the perspective of loss function categories, AIBench covers fourteen loss functions, including BCELoss, BCEWithLogitsLoss, ChamferLoss, CrossEntropyLoss, CTCLoss, GANLoss, L1Loss, NLLLoss, SigmoidLogLoss, SmoothL1Loss, SoftmaxLoss, TripletLoss, Earth-Mover distance, and first order Euclidean distance, while MLPerf only covers six loss functions, which are BCELoss, CrossEntropyLos, LogLoss, SmoothL1Loss, SoftmaxLoss and VirtualLoss.

2) Micro-architectural Characteristics: GPU architecture contains multiple streaming multiprocessors (SM), each of which has a certain number of CUDA cores, registers, caches, warp schedulers, etc. We choose five micro-architectural metrics to compare AIBench and MLPerf from a computation and memory access pattern perspective, including achieved occupancy, ipc_efficiency, gld_efficiency, gst_efficiency, and dram_utilization. Achieved_occupancy represents the ratio of the average active warps per active cycle to the maximum number of warps provided by a multiprocessor [56]. Ipc_efficiency indicates the ratio of the executed instructions per cycle to the theoretical number [56]. Gld_efficiency and gst_efficiency represent the ratio of requested global memory load/store throughput to required global memory load/store throughput, respectively [56]. Dram_utilization means the utilization level of the device memory relative to peak utilization [56].

Fig. 3 presents the computation and memory access patterns of the twenty-seven AI benchmarks (19 of AIBench, 8 of MLPerf). We find that AIBench captures distinct computation and memory patterns under different scenarios, e.g., processing text, image, audio, video, and various tasks of the same scenario, e.g., Image Classification and Image Generation,

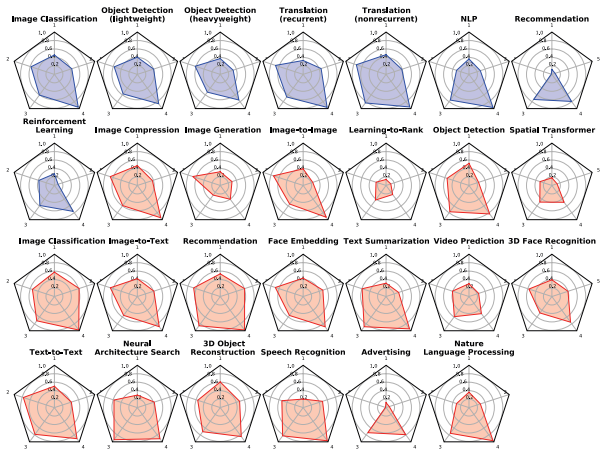

Fig. 3. The Computation and Memory Access Patterns of 24 Benchmarks from MLPerf (8) and AIBench (19) (1: achieved_occupancy; 2: ipc_efficiency; 3: gld_efficiency; 4: gst_efficiency; 5: dram_utilization).

while MLPerf not.

Through Fig. 2 and Fig. 3, we conclude that over AIBench, MLPerf has a significantly smaller coverage in terms of model complexity, computational cost, convergent rate, and computation and memory access patterns. AIBench outperforms MLPerf in terms of representativeness and diversity, which Table I also confirms.

C. Repeatability and Benchmarking Cost Evaluation

In this subsection, we describe the run-to-run variation and measure the benchmarking cost of AIBench against MLPerf.

1) Run-to-run Variation: Repeatability [57] refers to the variation in repeat measurements (different runs instead of different epochs using the same benchmark implementation under identical configurations) made on the same system under test.

Due to the stochastic nature of AI, different runs of the same benchmark under the same system are different. HPC AI500 [7] uses the coefficient of variation(the standard deviation / mean) of the number of epochs to quantify the run-to-run variation[2], and Table V (Column 5 and 6) shows the results.

2) Evaluate Benchmarking Cost: Running entire training sessions of all AIBench benchmarks is prohibitively costly. Table V (Columns 3 and 4) lists the elapsed time for both an epoch and the total time for a training session to the target quality. Image Classification, Nature Language Processing, and Speech Recognition are the top-three most time-consuming benchmarks, which take about 164.25 hours in total. Supposing that we run each of the above three benchmarks five times, the time consumption reaches 34.2 days. If we do these for all nineteen benchmarks, the time will go 45.5 days, which is not affordable for most industries and academia. So, a concise, portable, and lightweight subset is of great significance.

[2]For Advertising and Nature Language Processing, it uses the coefficient of variation of the number of iterations.

[3]This benchmark uses the TensorFlow version, since the PyTorch version has no pre-trained model.

TABLE V
TRAINING COSTS AND RUN-TO-RUN VARIATION (COLUMN 5 AND 6 ARE FROM HPC AI500 [7]) OF NINETEEN BENCHMARKS. THE TOTAL TIME RECORDS THE WHOLE TRAINING TIME TO ACHIEVE A TARGET QUALITY.

No.	Component Benchmark	Time Per Epoch (second)	Total Time (hour)	Variation	Repeat Times
TrC1	Image Classification	4440	76.25	1.12%	8
TrC2	Image Generation	3935.75	N/A	N/A	N/A
TrC3	Text-to-Text translation	64.83	1.72	9.38%	6
TrC4	Image-to-Text	845.02	10.21	23.53%	5
TrC5	Image-to-Image	251.67	N/A	N/A	N/A
TrC6	Speech Recognition	14326.86	42.78	12.08%	4
TrC7	Face Embedding	214.73	3.43	5.73%	8
TrC8	3D Face Recognition	36.99	12.02	37.11%	10
TrC9	Object Detection	1859.96	2.06	0	10
TrC10	Recommendation	36.72	0.16	9.95%	5
TrC11	Video Prediction	24.99	2.11	11.83%	4
TrC12	Image Compression	763.44	5.67	22.49%	4
TrC13	3D Object Reconstruction	28.41	0.38	16.07%	4
TrC14	Text Summarization	1923.33	6.41	24.72%	5
TrC15	Spatial Transformer	6.38	0.06	7.29%	4
TrC16	Learning-to-Rank	60.1	0.14	1.90%	4
TrC17	Neural Architecture Search	932.79	7.47	6.15%	6
TrC18	Advertising	1.77	2.28	0.12%	4
TrC19	Nature Language Processing (NLP)[3]	143.92	45.22	19.51%	5

Besides, we also evaluate the benchmarking cost of running a training session for MLPerf. To achieve a target quality, the time costs for MLPerf are 76.25 hours for Image Classification, 2.28 hours for Recommendation, 73.34 hours for Object Detection (heavyweight), 23.7 hours for Object Detection (lightweight), 16.52 hours for Translation (recurrent), 22 hours for Translation (nonrecurrent), 45.22 hours for NLP. For Reinforcement Learning in MLPerf, we train the model for more than 96 hours, and the pro move prediction reaches 34%, while the target is 40%. Hence, running all seven benchmarks in MLPerf to achieve a target quality one time, the time cost goes more than 355.31 hours, even larger than running all benchmarks in AIBench. The reason is that for some workloads, AIBench uses a different model or dataset against MLPerf. For example, for Object Detection, AIBench uses Faster R-CNN, a fundamental model instead of Mask R-CNN, and uses the VOC dataset rather than the COCO dataset to achieve affordability. If we repeat MLPerf benchmarks five times, the time cost will be more than 74 days.

D. The AIBench Subsets

For performance ranking and workload characterization purposes, we choose two subsets.

1) How to Choose the Two Subsets?: We need to keep the subset repeatable, fair, affordable, and representative for the repeatable performance ranking purpose, so we keep the subset to a minimum from the following perspectives and criteria.

Reflecting diverse model complexity, computational cost, and convergent rate. Specifically, we intend to choose the

benchmarks that cover different aspects of Fig. 2 as much as possible.

Run-to-run variation. Repeatability is an important selection criterion of the subset. To avoid too much run-to-run variation, we choose the benchmarks with variance under 2%.

Widely accepted evaluation metrics. A benchmark should have widely accepted performance metrics so that runs from different users have consistent termination conditions. So, we exclude the GAN-based models even if GANs are particularly important for content generation.

For the workload characterization purpose, we aim to select minimum workloads with the most representative system or micro-architectural characteristics, while the repeatability and consistent termination conditions are not mandatory.

2) The Subsets Decision: AIBench Training provides two subsets for repeatable performance ranking (RPR subset) and workload characterization (WC subset) to improve affordability.

RPR subset. The RPR subset includes three benchmarks: Image Classification, Object Detection, and Learning-to-Rank. To satisfy the first criterion, they cover different ranges of numbers in terms of FLOPs and learnable parameters (both small for Learning-to-Rank, medium for Image Classification, and large for Object Detection), and different convergent rates (small epochs for Object Detection, medium for Learning-to-Rank, and large for Image Classification). As for the second criterion, three benchmarks have the low run-to-run variation, 1.12% for Image Classification, 1.9% for Learning-to-Rank, and 0% for Object Detection. Besides, they have widely accepted evaluation metrics.

WC subset. The WC subset also includes three benchmarks: Spatial Transformer, Image-to-Text, and Speech-to-Text. This subset reflects the most representative micro-architectural workload characteristics since they are the nearest to the centroid of three clusters, respectively, according to the K-means clustering results on AIBench Training benchmarks in [7].

Comparing to the full benchmark of AIBench and MLPerf, the AIBench RPR subset shortens the training time by 64% and 78%, respectively, and the AIBench WC subset shortens by 76% and 85%, respectively.

E. Micro-architectural Behaviors

This subsection characterizes the micro-architectural behaviors from the perspectives of runtime breakdown, hotspot function analysis, and stall analysis.

1) Runtime Breakdown: We evaluate the PyTorch implementations with Pytorch 1.1.0. The dataset for each benchmark is as follows: ImageNet (137 GB) for Image Classification and Image Compression; LSUN (42.8 GB) for Image Generation; VGGFace2 (36 GB) for Face Embedding; Microsoft COCO (13 GB) for Image-to-Text; VOC2007 (439 MB) for Object Detection; MNIST (9.5 MB) for Spatial Transformer; Cityscapes (267 MB) for Image-to-Image; MovieLens (190 MB) for Recommendation; Librispeech (59.3 GB) for Speech Recognition; Gowalla (107 MB) for Learning-to-Rank; WMT English-German (1.2 MB) for Text-to-Text translation; Robot pushing dataset (137 GB) for Video Prediction; ShapeNet dataset (6.8 GB) for 3D Object Reconstruction; Gigaword dataset (277 MB) for Text Summarization; 3D face data (37 GB) for 3D Face Recognition; PTB dataset (4.9 MB) for Neural Architecture Search, respectively, Kaggle Display Advertising Challenge Dataset (4.3 GB) for Advertising, and Wikipedia (76 GB) for NLP.

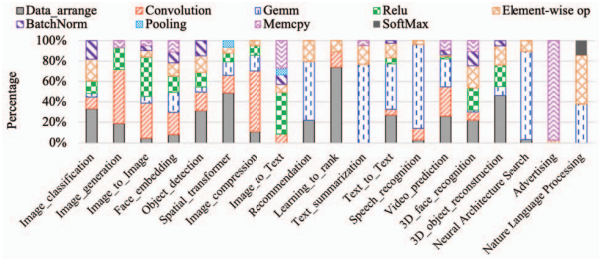

Fig. 4. Runtime Breakdown of the AIBench Benchmarks.

The overall execution performance of these component benchmarks varies in terms of IPC, which measures the executed instructions per cycle. Fig. 3 shows that the IPC efficiency ranges from 0.02 (Advertising) to 0.77 (Text-to-Text translation). Some benchmarks like Learning-to-Rank have extremely low IPC comparing to other benchmarks. To reveal the factors that impact the performance significantly, we first conduct runtime breakdown analysis and decompose the benchmarks into the hotspot functions. Then we explore the GPU execution efficiency in terms of different percentages of stalls.

We use NVProf to trace the runtime breakdown and find the hotspot functions that occupy more than 80% of runtime in total. Since each run involves dozens of function calls, we single out the functions that occupy large proportions of runtime and classify them into several categories of kernels according to their computation logic. Through statistics, we find that the most time-consuming functions among all component benchmarks have much in common, and they are classified into nine categories of kernels: data arrangement, convolution, general matrix multiply (gemm), batch normalization, element-wise operation, relu activation, pooling, memory copy, and softmax, spanning from computation kernels to memory access kernels. Note that each kernel contains a bunch of functions that solve a similar issue. For example, a gemm kernel includes single or double precision floating general matrix multiply. Fig. 4 shows the runtime breakdown of nineteen benchmarks of AIBench, using the average number of all involved functions within each category of kernels. Note that this figure does not consider the remaining 20% functions. Further, we summarize typical functions for each category of kernels that occupy a large proportion of runtime among the component benchmarks, as shown in Table VI. We find that Learning-to-Rank spends too much time on data arrangement operations from Fig. 4, and the corresponding function call is maxwell_scudnn_128x32_stridedB_splitK_interior_nn with

an IPC of 0.98. This is the reason that Learning-to-Rank has the low IPC of 0.99. We believe that the nine categories of kernels and these corresponding functions are the optimization points for CUDA library optimizations and micro-architectural optimizations.

TABLE VI
HOTSPOT FUNCTIONS.

Micro Benchmark	Function Name
Data Arrangement	1) maxwell_scudnn_128x128_stridedB_splitK_interior_nn; 2) maxwell_scudnn_128x32_stridedB_splitK_interior_nn; 3) maxwell_scudnn_128x128_stridedB_interior_nn
Convolution	1) maxwell_scudnn_winograd_128x128_ldg1_ldg4_tile148n_nt; 2) wgrad_alg0_engine; 3) fft2d_r2c_32x32
GEMM	1) maxwell_sgemm_128x64_nt; 2) maxwell_sgemm_128x64_nn; 3) sgemm_32x32x32_NN_vec;
BatchNorm	1) cudnn::detail::bn_fw_tr_1C11_kernel_NCHW; 2) cudnn::detail::bn_bw_1C11_kernel_new; 3) batch_norm_backward_kernel; 4) at::native::batch_norm_backward_kernel
Relu	1) maxwell_scudnn_128x128_relu_small_nn; 2) maxwell_scudnn_128x128_relu_interior_nn; 3) maxwell_scudnn_128x32_relu_interior_nn
Element-wise	1) element-wise add kernel; 2) element-wise threshold kernel; 3) element-wise mul kernel
Pooling	1) MaxPoolBackward; 2) AvePoolForward
Memcpy	1) CUDA memcpy HtoD; 2) CUDA memcpy DtoD
Softmax	1) SoftMax_compute

2) Hotspot Function Analysis: Hotspot function identification is of great significance for bottleneck locating and code optimization. We compare the essential hotspot functions identified by AIBench and MLPerf.

Fig. 5 shows the numbers of hotspot functions within each category of occupying different time percentages identified by AIBench and MLPerf. We find that MLPerf only covers a fraction of hotspot functions over AIBench. For example, within the category that occupies more than 10% of runtime, the number of hotspot functions profiled from AIBench is 30, while only 9 for MLPerf. Thus, MLPerf omits many hotspot functions that occurred in a broad spectrum of AI tasks.

We further profile the AIBench RPR subset to find whether they capture the primary hotspot functions. Our evaluation shows that even though the RPR subset captures the least number of hotspot functions compared to the full benchmarks of AIBench and MLPerf, it still covers the most time-consuming and frequently-appearing functions within these benchmarks like maxwell_scudnn_128x128_stridedB_splitK_interior_nn (e.g., occupying 17% running time for 3D Object Reconstruction).

In conclusion, for AIBench, the full benchmarks and the subset are two indispensable parts. Over MLPerf, the complete benchmarks of AIBench provide comprehensive workload characterization and detailed evaluation while reducing the training time by 37%. Against the AIbench full benchmarks, its subset further shortens the benchmarking cost by 41% while maintaining the primary workload characteristics.

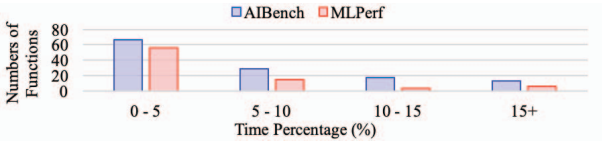

Fig. 5. The Number of Hotspot Functions Identified by AIBench and MLPerf.

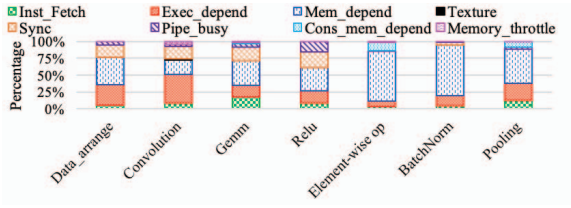

Fig. 6. Stall Breakdown of the Hotspot Functions.

3) Stall Analysis: We analyze the percentage of stalls of eight kinds of stalls focusing on the above eight kernel categories. Instruction fetch stall (Inst_fetch) indicates the next assembly instruction has not yet been fetched. Execution dependency stall (Exe_depend) is because an input required by the instruction is not, however, available. Memory dependency stall (Mem_depend) is because a memory operation cannot be performed due to the required resources not being available or fully utilized. Texture stall (Texture) is because of the under-utilization of the texture sub-system. Synchronization stall (Sync) is due to a syncthreads call. Constant memory dependency stall (Const_mem_depend) is because of an immediate constant cache miss. Pipe busy stall (Pipi_busy) is because a compute operation cannot be performed because the compute pipeline is busy. Memory throttle stall (Mem_throttle) is due to large pending memory operations [56].

Fig. 6 shows the breakdown of eight stalls of the hotspot functions. The top two GPU execution stalls are memory dependency stalls and execution dependency stalls. For example, for Element-Wise kernels, the memory dependency stalls occupy a substantial proportion of 70%, resulting in a low IPC of 0.86 on average. The memory dependency stalls may occur due to high cache misses.Possible optimizations include optimizing data alignment, data locality, and data access patterns. The execution dependency stalls may occur due to low instruction-level parallelism, and exploiting ILP may alleviate partial execution dependency stalls.

V. PERFORMANCE RANKING

In this section, we use the AIBench RPR subset to rank and report the performance of GPUs and TPUs. For GPU evaluations, we deploy them on our local servers. For TPU evaluation, we rent a Google Cloud environment. Table VII lists their configurations. Note that the other server configurations for GPUs are consistent with the illustrations in Section IV-A.

Our benchmarks support distributed training, and the performance difference of multi-cards is consistent with single-

TABLE VII
GPU AND TPU CONFIGURATIONS.

GPU Type	Arch	GPU Cores	Memory	FP32	Year
RTX 2080S	Turing	3072	8GB	11.15 TFLOPS	2019
RTX 2080 Ti	Turing	4352	11GB	13.1 TFLOPS	2018
TITAN RTX	Turing	4608	24GB	16.3 TFLOPS	2018
TITAN XP	Pascal	3840	12GB	12.15 TFLOPS	2017
TITAN V	Volta	5120	12GB	15 TFLOPS	2017
P100	Pascal	3584	16GB	9.3 TFLOPS	2016
P40	Pascal	3840	24GB	12 TFLOPS	2016
TPU Type	**Arch**	**TPU Cores**	**Memory**	**Bfloat16**	**Year**
TPU v3	N/A	8	128GB	420 TFLOPS	2018
TPU v2	N/A	8	64GB	180 TFLOPS	2017

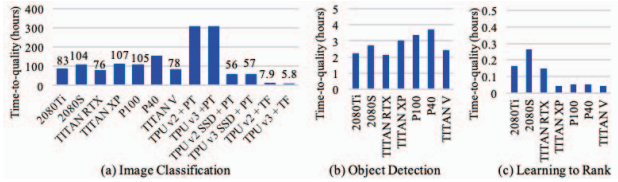

Fig. 7. GPU and TPU Ranking using AIBench RPR subset. PT and TF indicate PyTorch and TensorFlow versions. SSD represents a solid-state disk. TPU currently does not support Object Detection (Faster R-CNN) and Learning-to-Rank [59].

card, so we only report single-card performance ranking here. We evaluate the GPU and TPU listed in Table VII and use the PyTorch version of the AIBench RPR subset. The metric is time-to-quality. As shown in Fig. 7, the RPR subset has deficient performance on TPU compared to GPU, with a deterioration of 2 to 4 times. The reason is that TPU connects the host virtual machine of Google Cloud through the network instead of being embedded on the motherboard like GPU. Hence, the data feeding is highly input-bound for Image Classification unless there are many workers to feed in data and sufficient RAM to maintain many worker threads [58]. Our experiments show that when using TPU, the average I/O Wait ratio of the virtual machine is more than 80%, which is indeed a severe I/O bottleneck.

Considering the I/O bottleneck of PyTorch on TPU, we replace HDD with SSD to accelerate I/O processing (TPU v2/3 SSD+PT in Fig. 7(a)). Image classification performance improves about 5.3 times compared to the original for both TPU v2 and v3 and is even better than all tested GPUs. We further evaluate the TensorFlow version's performance on TPU (TPU v2/3+TF in Fig. 7(a)). The TensorFlow version on TPU achieves the best performance. For Image Classification, it spends 7.8 hours on TPU v2 and 5.8 hours on TPU v3, which is about eight times higher than that of the PyTorch version on TPU with SSD average, and 9.6 times higher than the best performance on GPU (TITAN RTX: 76 hours). The reason is that TPU provides a mechanism specialized for TensorFlow, which reads the data in a distributed way from cloud storage directly and caches them into memory to fully utilize the high-speed network bandwidth and achieve fast data access. Although TPU reflects extremely high performance for Image Classification, it supports limited models officially, bringing the huge portability cost, which is not the case for many general-purpose accelerators like GPUs.

VI. CONCLUSION

This paper summarizes AI benchmarking challenges: prohibitive cost, conflicting requirements, short shelf-life, scalability, and repeatability. AIBench is the first benchmark project that systematically tackles the above challenges: it distills and abstracts real-world application scenarios into the scenario, training, inference, micro, and synthetic AI benchmarks.

We present a balanced industry-standard methodology to meet the conflicting benchmarking requirements in different stages. We use real-world benchmarks to cover the factors space that impacts the learning dynamics to the most considerable extent. We identify and implement nineteen representative AI tasks with state-of-the-art models. For two different purposes: repeatable performance ranking (RPR subset) and workload characterization (WC subset), we keep two subsets to a minimum for affordability.

Our evaluations show AIBench outperforms MLPerf in terms of the diversity and representativeness of model complexity, computational cost, convergent rate, computation and memory access patterns, and hotspot functions; AIBench reduces the benchmarking cost while avoiding error-prone design or benchmarketing. With respect to the AIBench full benchmarks, its RPR subset shortens the benchmarking cost by 64%, while maintaining the primary workload characteristics.

ACKNOWLEDGMENT

We are very grateful to the anonymous reviewers. This work is supported by the Standardization Research Project of Chinese Academy of Sciences No.BZ201800001.

REFERENCES

[1] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.
[2] B. Smith and G. Linden, "Two decades of recommender systems at amazon. com," *Ieee internet computing*, vol. 21, no. 3, pp. 12–18, 2017.
[3] K. Hazelwood *et al.*, "Applied machine learning at facebook: A datacenter infrastructure perspective," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 620–629.
[4] C. Coleman *et al.*, "Dawnbench: An end-to-end deep learning benchmark and competition," *Training*, vol. 100, no. 101, p. 102, 2017.
[5] P. Mattson *et al.*, "Mlperf training benchmark," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 336–349.
[6] J. Gray, "Database and transaction processing performance handbook." 1993.
[7] Z. Jiang, W. Gao, F. Tang, X. Xiong, L. Wang, C. Lan, C. Luo, H. Li, and J. Zhan, "Hpc ai500: Representative, repeatable and simple hpc ai benchmarking," *arXiv preprint arXiv:2102.12848*, 2021.
[8] C. Luo, F. Zhang, C. Huang, X. Xiong, J. Chen, L. Wang, W. Gao, H. Ye, T. Wu, R. Zhou, and J. Zhan, "Aiotbench: Towards comprehensive benchmarking mobile and embedded device intelligence," *2018 BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing (Bench18)*, 2018.
[9] T. Hao, Y. Huang, X. Wen, W. Gao, F. Zhang, C. Zheng, L. Wang, H. Ye, K. Hwang, Z. Ren, and J. Zhan, "Edge aibench: Towards comprehensive end-to-end edge computing benchmarking," *2018 BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing (Bench18)*, 2018.

[10] W. Gao, F. Tang, J. Zhan, X. Wen, L. Wang, Z. Cao, C. Lan, C. Luo, X. Liu, and Z. Jiang, "Aibench scenario: Scenario-distilling ai benchmarking," *arXiv preprint arXiv:2005.03459*, 2020.

[11] G. Wanling *et al.*, "Aibench inference: Characterizing the inherent characterization of ai inference workloads," *BenchCouncil Technical Report*, 2021.

[12] T. Fei *et al.*, "Aibench micro: Profiling and understanding hotspot functions of ai workloads," *BenchCouncil Technical Report*, 2021.

[13] T. Fei *et al.*, "Aibench synthetic: A synthetic ai benchmark suite modelling learning dynamics," *BenchCouncil Technical Report*, 2021.

[14] Y. Wang, G.-Y. Wei, and D. Brooks, "A systematic methodology for analysis of deep learning hardware and software platforms," *Proceedings of Machine Learning and Systems*, vol. 2020, pp. 30–43, 2020.

[15] Z. Jiang *et al.*, "Hpc ai500: A benchmark suite for hpc ai systems," *2018 BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing (Bench18)*, 2018.

[16] L. Wang, X. Xiong, J. Zhan, W. Gao, X. Wen, G. Kang, , and F. Tang, "Wpc: Whole-picture workload characterization across intermediate representation, isa, and microarchitecture," *Technical Report*, 2021.

[17] J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: past, present and future," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003.

[18] https://www.spec.org/cpu2017/.

[19] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 72–81.

[20] http://www.tpc.org/tpcds/.

[21] https://www.top500.org/.

[22] J. Dongarra, M. A. Heroux, and P. Luszczek, "High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems," *International Journal of High Performance Computing Applications*, vol. 30, 2015.

[23] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The hpc challenge (hpcc) benchmark suite," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, vol. 213, no. 10.1145. Citeseer, 2006, pp. 1 188 455–1 188 677.

[24] https://www.nas.nasa.gov/publications/npb.html.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[28] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 652–663, 2017.

[29] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[30] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*, 2016, pp. 173–182.

[31] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[32] R.-L. Vieriu, S. Tulyakov, S. Semeniuta, E. Sangineto, and N. Sebe, "Facial expression recognition under a wide range of head poses," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 1. IEEE, 2015, pp. 1–7.

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[34] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[35] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, 2016, pp. 64–72.

[36] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.

[37] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision," in *Advances in Neural Information Processing Systems*, 2016, pp. 1696–1704.

[38] R. Nallapati, B. Zhou, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[39] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[40] J. Tang and K. Wang, "Ranking distillation: Learning compact ranking models with high performance for recommender system," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[41] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4095–4104.

[42] M. Mitchell, B. Santorini, M. Marcinkiewicz, and A. Taylor, "Treebank-3 ldc99t42 web download," *Philidelphia: Linguistic Data Consortium*, vol. 3, p. 2, 1999.

[43] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini *et al.*, "Deep learning recommendation model for personalization and recommendation systems," *arXiv preprint arXiv:1906.00091*, 2019.

[44] https://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset.

[45] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[46] W. Gao, C. Luo, L. Wang, X. Xiong, J. Chen, T. Hao, Z. Jiang, F. Fan, M. Du, Y. Huang *et al.*, "Aibench: towards scalable and comprehensive datacenter ai benchmarking," in *International Symposium on Benchmarking, Measuring and Optimization*. Springer, 2018, pp. 3–9.

[47] W. Gao, F. Tang, L. Wang, J. Zhan, C. Lan, C. Luo, Y. Huang, C. Zheng, J. Dai, Z. Cao *et al.*, "Aibench: an industry standard internet service ai benchmark suite," *arXiv preprint arXiv:1908.08998*, 2019.

[48] "Hpl-ai mixed-precision benchmark," https://icl.bitbucket.io/hpl-ai/.

[49] "Deepbench," https://svail.github.io/DeepBench.

[50] T. Chen, Y. Chen, M. Duranton, Q. Guo, A. Hashmi, M. Lipasti, A. Nere, S. Qiu, M. Sebag, and O. Temam, "Benchnn: On the broad potential application scope of hardware neural network accelerators," in *Workload Characterization (IISWC), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 36–45.

[51] J.-H. Tao, Z.-D. Du, Q. Guo, H.-Y. Lan, L. Zhang, S.-Y. Zhou, L.-J. Xu, C. Liu, H.-F. Liu, S. Tang *et al.*, "Benchip: Benchmarking intelligence processors," *Journal of Computer Science and Technology*, vol. 33, no. 1, pp. 1–23, 2018.

[52] R. Adolf, S. Rama, B. Reagen, G.-Y. Wei, and D. Brooks, "Fathom: reference workloads for modern deep learning methods," in *Workload Characterization (IISWC)*. IEEE, 2016, pp. 1–10.

[53] H. Zhu, M. Akrout, B. Zheng, A. Pelegris, A. Jayarajan, A. Phanishayee, B. Schroeder, and G. Pekhimenko, "Benchmarking and analyzing deep neural network training," in *2018 IEEE International Symposium on Workload Characterization (IISWC)*, 2018, pp. 88–100.

[54] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[55] https://github.com/Lyken17/pytorch-OpCounter.

[56] https://docs.nvidia.com/cuda/profiler-users-guide/index.html.

[57] J. Bartlett and C. Frost, "Reliability, repeatability and reproducibility: analysis of measurement errors in continuous variables," *Ultrasound in Obstetrics and Gynecology: The Official Journal of the International Society of Ultrasound in Obstetrics and Gynecology*, vol. 31, no. 4, pp. 466–475, 2008.

[58] https://cloud.google.com/tpu/docs/tutorials/resnet-pytorch.

[59] https://cloud.google.com/tpu/docs/tutorials/support-matrix?hl=en.