Theme Article: Hot Chips

MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance

Peter Mattson
Google Brain

Vijay Janapa Reddi
Harvard University

Christine Cheng
Intel

Cody Coleman
Stanford University

Greg Diamos
Landing AI

David Kanter
Real World Technologies

Paulius Micikevicius
NVIDIA

David Patterson
Google Brain and University of California, Berkeley

Guenther Schmuelling
Microsoft Azure AI Infrastructure

Hanlin Tang
Intel

Gu-Yeon Wei
Harvard University

Carole-Jean Wu
Facebook and Arizona State University

***Abstract*—In this article, we describe the design choices behind MLPerf, a machine learning performance benchmark that has become an industry standard. The first two rounds of the MLPerf Training benchmark helped drive improvements to software-stack performance and scalability, showing a 1.3× speedup in the top 16-chip results despite higher quality targets and a 5.5× increase in system scale. The first round of MLPerf Inference received over 500 benchmark results from 14 different organizations, showing growing adoption.**

MACHINE LEARNING (ML) is transforming multiple industries, leading to a surge in hardware and software development. Most modern ML systems are built atop deep neural networks which are computationally demanding to train and deploy, thus their increasing use in industry is driving the rapid development of specialized hardware architectures and software frameworks.

Digital Object Identifier 10.1109/MM.2020.2974843
Date of publication 18 February 2020; date of current version 18 March 2020.

0272-1732 © 2020 IEEE

We need a performance benchmark to evaluate these competing ML systems. By providing clear metrics, benchmarking aligns research, engineering and marketing, and competitors across the industry in pursuit of the same objectives. For general-purpose computing, a consortium of chip vendors built the SPEC benchmark[1] in 1988, focusing the competition that drove computing performance for the next three decades. Earlier ML benchmarks include DeepBench[2] which focused on deep learning primitives, Fathom[3] which introduced a field-spanning set of ML benchmarks, and DAWNBench[4] which proposed time-to-convergence as a metric. See the work by Mattson *et al.*[5] and Reddi *et al.*[6] for more on related work.

MLPerf was founded in 2018 to combine the best of prior efforts: a broad benchmark set with a time-to-convergence metric and the support of an academic/industry consortium.

MLPerf was founded in 2018 to combine the best of prior efforts: a broad benchmark set with a time-to-convergence metric and the support of an academic/industry consortium. MLPerf contains two suites of ML benchmarks: one for training,[5] and one for inference.[6] MLPerf has released two rounds of results for the training suite and one round for the inference suite. Comparing the two rounds of training data shows MLPerf is encouraging improvements in performance and scalability; comparing all three rounds shows growing adoption. The remainder of this article describes the design choices faced by anyone seeking to benchmark ML performance, and how MLPerf navigated those choices to become a nascent industry standard.

DESIGN CHOICES

Designing any benchmark suite requires answering three big questions.

- Benchmark definition: How to specify a measurable task?
- Metric definition: How to measure performance?
- Benchmark selection: What set of tasks to measure?

Designing an ML benchmark suite requires answering additional questions.

- Implementation equivalence: ML accelerator architecture varies and there is no standard ML software stack. Submitters need to reimplement the benchmark for their hardware. How do we ensure that implementations are equivalent enough for fair comparison?
- Training hyperparameter equivalence: for training benchmarks, which hyperparameters are tunable?
- Training convergence variance: for training benchmarks, convergence times have relatively high variance. How do we make meaningful measurements?
- Inference weight equivalence: for inference benchmarks, are retrained or sparsified weights allowed?

This section describes how the MLPerf benchmark suites answer these questions.

MLPerf Training

Benchmark Definition We specify an MLPerf Training benchmark[5] as training a model on a specific data set to reach a target quality. For example, one benchmark measures training on the ImageNet data set until the image classification top-1 accuracy reaches 75.9%. However, this basic definition does not answer one critical question: do we specify which model to train? Specifying the model enables apples-to-apples performance comparisons of software or hardware alternatives because it requires all alternatives to process the same workload. However, not specifying the model encourages model improvements and hardware–software codesign. We created two divisions of results: a *Closed Division* that requires using a specific model for direct comparisons, and an *Open Division* that allows the use of any model to support model innovation.

Metric Definition There are two obvious metrics for training performance: *throughput*, the number of data processed per second, and *time-to-train*, the wall clock time it takes for the model to reach a target quality. These metrics could also be normalized by cost or power, which we

Table 1. MLPerf Training v0.5 Benchmarks.

Area	Problem	Data set	Model
Vision	Image recognition	ImageNet[7]	ResNet[7]
	Object detection	COCO[7]	SSD[7]
	Object segmentation	COCO[7]	Mask R CNN[7]
Language	Translation	WMT Eng.-German[7]	GNMT[7]
	Translation	WMT Eng.-German[7]	Transformer[7]
Commerce	Recommendation	Movielens-20M[7]	NCF[7]
Research	RL	Go, 9×9 board	MiniGo[7]

will address later on. Throughput has advantages. First, it is computationally inexpensive to measure because you do not train the model to completion. Second, it has a relatively low variance because the compute cost per datum is constant in most models. However, throughput can be increased at the cost of time-to-train by using optimizations like lower precision numerics or larger batch sizes.

We chose to use time-to-train because it accurately reflects the primary goal in choosing a training system: to fully train models as quickly as possible. Unfortunately, it is computationally expensive to fully train models. Furthermore, the number of epochs needed to train a model varies due to random weight initialization and stochastic floating-point ordering effects. However, we feel time-to-train is the least-bad alternative available.

Benchmark Selection Once we chose how to specify a benchmark, we needed to select a set of benchmarks. We first divided ML applications into five broad areas.

- Vision: image classification, object detection, segmentation, medical imaging.
- Speech: speech to text, text to speech.
- Language: translation, natural language processing (NLP).
- Commercial: recommenders, time-series.
- Research: reinforcement learning (RL) for games or robotics, generative adversarial networks (GANs).

We then tried to select one or two specific benchmarks within each area. In doing so, we chose models based on four characteristics.

- *Maturity*: We sought models that were near state-of-the-art but also showed evidence of growing adoption.
- *Variety*: We chose models that included a range of constructs such as a convolutional neural network (CNN), a recurrent neural network (RNN), attention, and an embedding table.
- *Complexity*: We chose model sizes to reflect current and anticipated market demands.
- *Practicality*: We chose only benchmarks with available data sets and models.

Table 1 shows the benchmarks in MLPerf Training v0.5. We chose ResNet as having relatively high accuracy and wide adoption. We added SSD and Mask R-CNN to cover two different points in the important vision complexity space. We chose transformer and GNMT for translation to increase variety by including attention and an RNN. We chose MiniGo for RL because it did not require an even more computationally expensive physics simulation. MLPerf Training presently omits medical imaging, speech-to-text, text-to-speech, NLP, time series, and GANs. Future versions of MLPerf will address these applications, starting with BERT in MLPerf training v0.7.

Implementation Equivalence ML benchmarks cannot function like conventional benchmarks in which fixed code is executed because it is not currently possible to write portable, scalable, high-performance ML code. There is no single ML framework supported by all architectures. Furthermore, ML code needs to be tuned for the architecture and system scale in order to achieve high performance.

Instead, MLPerf allows submitters to reimplement the benchmarks. However, this flexibility raises the question of implementation equivalence. MLPerf requires that all submitters to the closed division use the same model to enable apples-to-apples hardware comparisons, but what does it mean to use the same model? MLPerf provides a functional but unoptimized *reference* *implementation*. MLPerf rules require performing the same set of mathematical operations as the reference implementation to produce each output, using the same optimizer to update the weights, and using the same preprocessing and evaluation methods. Submitters are allowed to reorder data channels and parallel operations, use different numerical representations, and make a few other whitelisted changes.

Hyperparameter Tuning Different systems need different hyperparameters for optimal performance. Systems have varying levels of parallelism and therefore demand different batch sizes (numbers of inputs between weight updates). Similarly, different numerical representations affect training behavior and require changes to the learning rate schedule and other optimizer hyperparameters.

Finding the optimal hyperparameters requires exploring a many-dimensional space where each point is evaluated by training an ML model to convergence, which can take days on a single processor. Therefore, allowing submitters to do an unlimited hyperparameter search conveys an advantage on those with the most computational resources and/or best hyperparameter search strategy.

Currently, MLPerf tries to level the hyperparameter playing field while still allowing sufficient flexibility in two ways.

1. *Search limits*: MLPerf limits which hyperparameters can be tuned.
2. *Hyperparameter borrowing:* MLPerf allows submitters to "borrow" hyperparameters from other submissions and update their results prior to posting final results.

Variance The time to train a model to a target quality has relatively high variance. The time to train is roughly proportional to the number of epochs (passes over the training data) required. The number of epochs required varies because the starting weights are initialized to random values and because of nondeterministic floating-point ordering. This variance can be reduced by averaging the results of multiple runs. However, the reduction is proportional to the square root of the number of runs and each run is computationally costly; often multiple days on a single processor.

Table 2. MLPerf Inference v0.5 Benchmarks.

Area	Task	Data set	Model
Vision	Image classification	ImageNet (224×224)[7]	Resnet50-v1.5[7]
Vision	Image classification	ImageNet (224×224)[7]	MobileNets-v1 224p[7]
Vision	Object detection	COCO (1200×1200)[7]	SSD-ResNet34[7]
Vision	Object detection	COCO (300×300)[7]	SSD-MobileNets-v1[7]
Language	Machine translation	WMT Eng.-German[7]	GNMT[7]

MLPerf balances variance and computation cost by averaging over a number of runs but still accepting relatively high variance. MLPerf averages five runs for relatively stable vision benchmarks producing results that are roughly ±2.5% and ten runs for the other higher variance benchmarks producing results that are roughly ±5%.

MLPerf Inference

Benchmark Definition We define an MLPerf inference benchmark[6] as processing a series of inputs to a trained model to produce outputs that meet a quality target. We expand on this basic definition with four measurement scenarios shown in black in Figure 1, each of which addresses a class of use cases.

1. Single stream: a series of inputs are processed one after the other as in, for example, a cell mobile vision application.
2. Multiple stream: fixed-size batches of inputs are processed one after the other as in, for example, automotive vision.
3. Server: inputs arrive according to a Poisson distribution as in, for example, an online translation service.
4. Offline: all inputs are available at once as in, for example, a photo labeling application.

MLPerf inference then defines a benchmark for each of the above scenarios. Like MLPerf Training, MLPerf Inference has a Closed Division which mandates a specific model to enable direct comparisons and an Open Division which allows any model to encourage innovation.

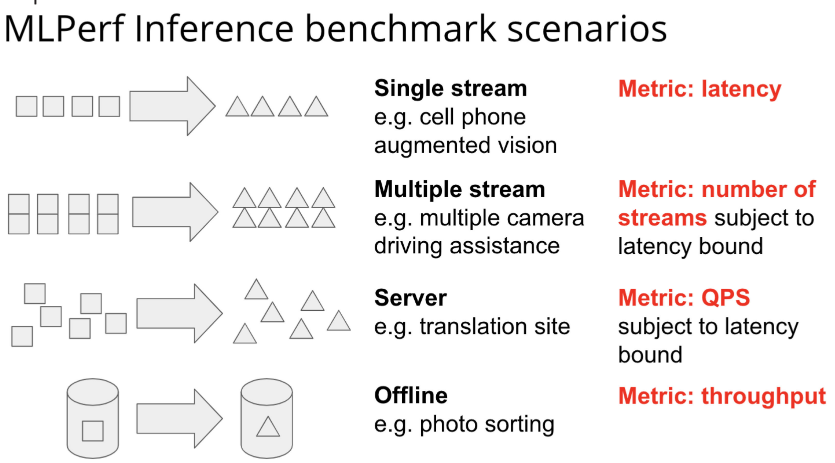

MLPerf Inference benchmark scenarios
Single stream
e.g. cell phone augmented vision
Metric: latency
Multiple stream
e.g. multiple camera driving assistance
Metric: number of streams subject to latency bound
Server
e.g. translation site
Metric: QPS subject to latency bound
Offline
e.g. photo sorting
Metric: throughput

Figure 1. MLPerf inference scenarios and metrics.

Metric Definition The ideal metric for measuring the performance of an inference system varies with the use case. For instance, a mobile vision application needs low latency, while an offline photo application demands high throughput. For this reason, each MLPerf inference benchmark scenario has a different metric, as shown in red in Figure 1.

1. Single stream: latency.
2. Multiple stream: number of streams subject to a latency bound.
3. Server: Poisson-arrival queries per second subject to a latency bound.
4. Offline: throughput.

Benchmark Selection The benchmark selection for MLPerf Inference was also driven by maturity, diversity, complexity, and practicality. However, we also needed to choose models with complexities suitable to a spectrum of hardware ranging from mobile devices to servers and support whatever models we chose in multiple scenarios. Thus, the initial version of MLPerf Inference focuses only on the most common vision tasks at different complexities. It complements the vision models with one moderate-size language model to increase model diversity. Future versions will expand this model selection and better align it with training.

Implementation Equivalence MLPerf Inference gives submitters the ability to reimplement the models to handle software stack and hardware diversity, again raising the question of model equivalence for the Closed Division. MLPerf Inference handles implementation equivalence by introducing two fundamental constraints.

1. All implementations must use a standardized load generator that implements the four scenarios and measures the corresponding metrics.
2. All implementations must use the same reference weights.

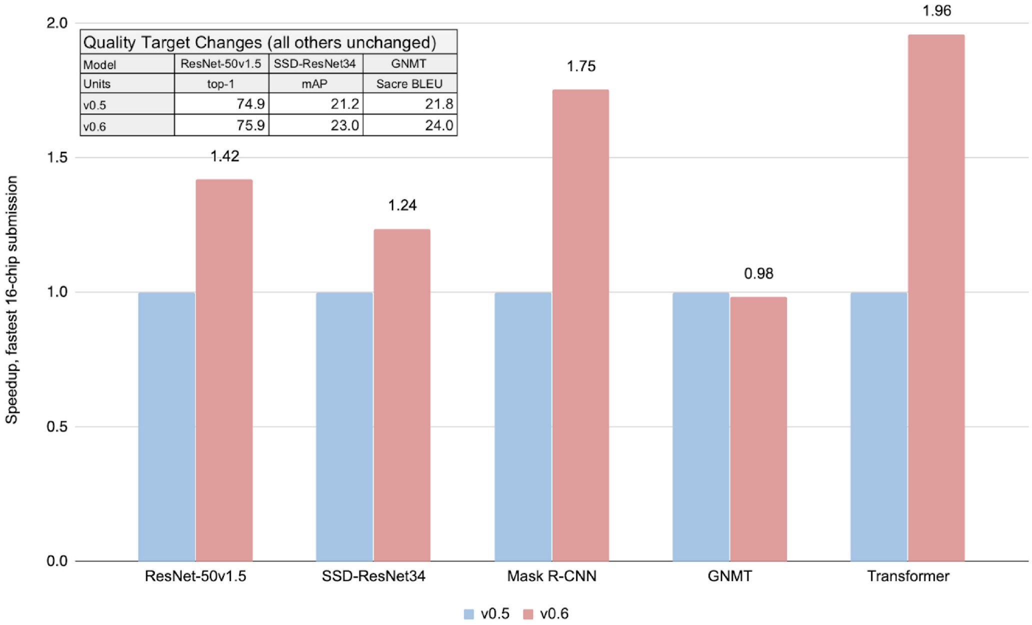
In addition to these two fundamental constraints, there is a short blacklist of forbidden optimizations including incorporating additional weights or other information about the data set, caching results to take advantage of repeated inputs, or sparsely evaluating the weights.

Quantization, Retraining, and Sparsity Inference systems can use quantized, retrained, or sparsified weights to increase computational efficiency at the cost of reduced accuracy, which may or may not match market requirements and can offer an unfair advantage in benchmarking. Different applications have different tolerance for accuracy loss, making it challenging to set an inference benchmark quality target. Further, retraining and sparsification techniques are a research area with significant proprietary technology and allowing either provides an advantage to those with the best methods.

For the initial version of MLPerf inference we took a simple approach by not allowing retraining or sparsification. Most quality targets are set to 99% of the quality that can be achieved with 32-bit floating point weights. These targets are somewhat lower than might be required for many applications, but we only allow post-training quantization. For future versions, we are investigating more flexible approaches to accuracy and allowing retraining and/or sparsification.

Presentation

Results or Single Summary Score Given a set of benchmark results, there are still choices



Quality Target Changes (all others unchanged)			
Model	ResNet-50v1.5	SSD-ResNet34	GNMT
Units	top-1	mAP	Sacre BLEU
v0.5	74.9	21.2	21.8
v0.6	75.9	23.0	24.0

Figure 2. Speedup in the fastest 16-chip entry from MLPerf Training version v0.5 to v0.6, despite more timed work due to increased quality targets as shown.

about how to present them in the most informative way. For instance, should results for one system on multiple benchmarks be combined to produce a single summary "score"? A score provides a consistent and easy way to communicate the bottom line. However, it assumes that systems are designed for general performance and that all benchmarks in the suite matter equally. We provide results instead of a single summary score because the range of ML use cases, from automotive vision to online recommendations, makes these assumptions incorrect: most users care about a subset of the benchmarks and many hardware architectures are specialized.

Scale Information and Normalization

Another presentation question is: should the results be normalized or include scale information? Different systems have different *scale factors* such as price, power consumption, and chip count. If system A performs slightly better than system B, but consumes twice as much power, which is a better design? In order to help consumers of benchmark results better utilize them, the results could be presented with one or more scale factors as additional information or normalized by a specific scale factor.

MLPerf currently provides scale information in the form of chip counts and future versions will include power measurements. MLPerf does not provide price because it is not a physical quantity and can vary over time and market. MLPerf does not normalize because the most important scale factor is different for different uses.

RESULTS

Benchmark suites aim to drive technological progress on faster hardware and software; we can measure this progress by comparing the best results on the benchmark suite over time. Comparing two rounds of MLPerf Training results shows progress in software-stack performance and scalability. The two rounds of results were collected approximately six months apart, and are driven by the same ML accelerators. Figure 2 compares five benchmarks that did not change significantly between the two rounds, though the target quality levels of three out of five benchmarks were increased, which increases the amount of work being timed.[8] The fastest 16-chip entries across the two rounds show an average 1.3× speedup *despite* more work required. Figure 3 shows an average 5.5× increase in system scale across the two rounds as submitters were able to effectively utilize

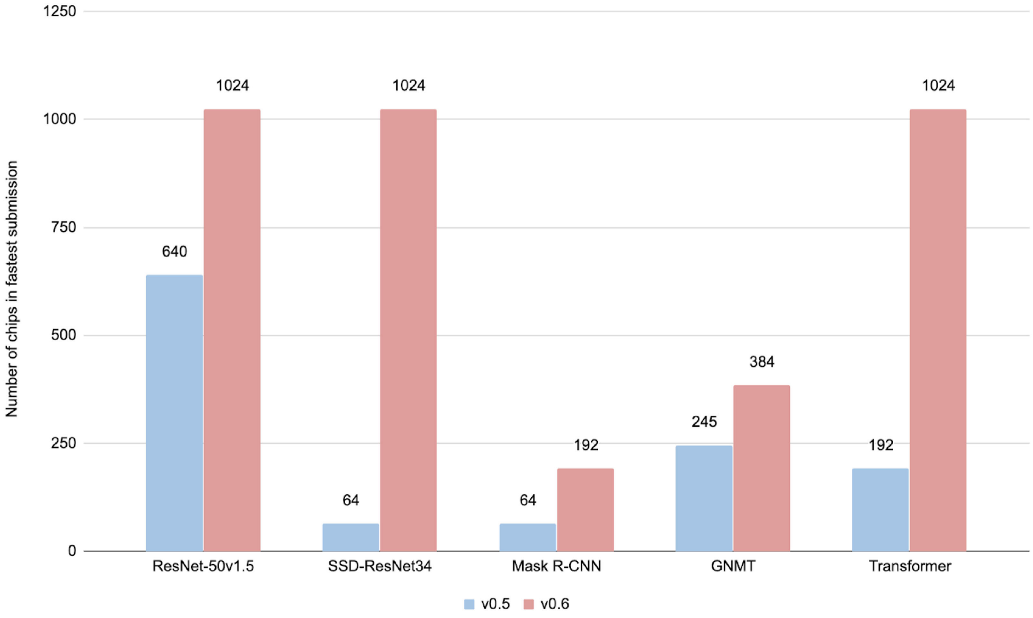


Figure 3. Increase in the number of chips used in the system that produced the fastest overall score from MLPerf Training version v0.5 to v0.6.

more chips.[8] Some of these improvements are benchmark-specific and some would have occurred without MLPerf, but many, based on first-hand observations of the authors, are generic and motivated by MLPerf. Over time, we expect similar improvements in hardware.

Since we only have a single round of MLPerf inference results, we cannot yet assess if it is driving performance improvements but can assess how well it handles diverse hardware as well as growth in adoption. Figure 4 shows that the systems submitted, ranging from embedded devices to cloud scale data center solutions, cover a very wide range of hardware that spans four orders of magnitude in terms of performance.[9] The number of submissions and results in each round of MLPerf is increasing: v0.5 had 3 submitters and 40+ results, training v0.6 had 5 submitters and 60+ results, inference v0.5 had 14 submitters and 500+ results.

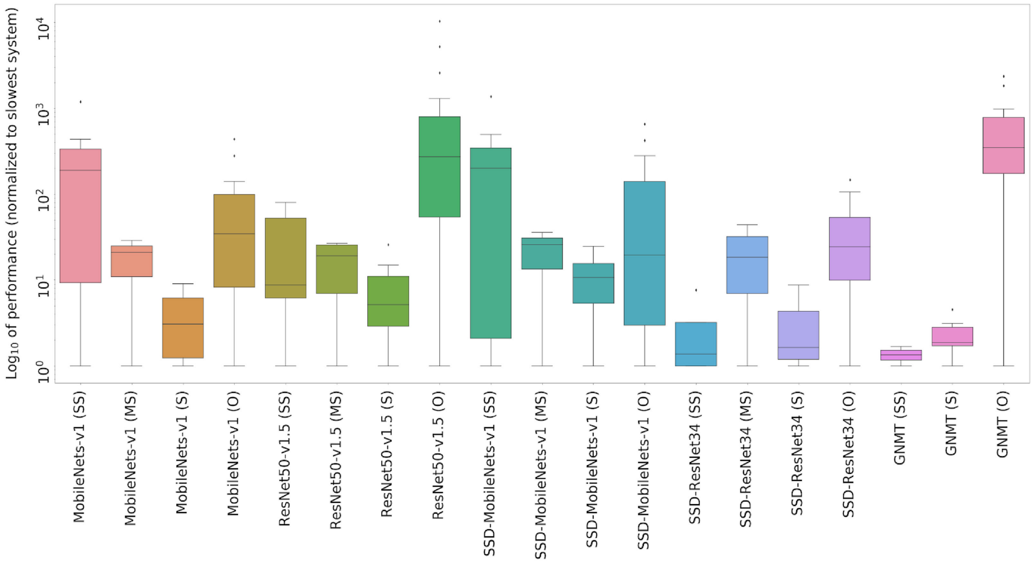


Figure 4. Normalized performance distribution in log scale from results in the closed division.

CONCLUSION

ML is a developing field and the MLPerf benchmark suites will need to evolve with the field; we have created an organization to enable that evolution. MLPerf inference and training are both driven by active working groups (WGs): a submitter's WG that maintains the rules, a special topics WG that explores deep technical issues, and a results WG that handles submission review and results presentation. Other WGs focus on specific topics such as power measurement or new benchmarks. We are creating a legal entity to provide a long term foundation for the effort.

We are developing a long-term benchmark roadmap. We aim to add new benchmarks to fully cover the five large ML areas we initially identified: vision, speech, language, commerce, and research. Over time, we will retire and replace benchmarks to keep pace with the field and to reduce the temptation to tune for benchmarks rather than real applications. We are recruiting a panel of academic and industry advisors for each area to ensure that MLPerf benchmarks are neutrally driven by research and industry needs.

The MLPerf effort is now supported by more than 65 companies and researchers from eight educational institutions. We welcome the involvement of engineers and researchers who are interested in helping us make MLPerf better by going to mlperf.org/get-involved.

Other future work includes the following.

- Creating a mobile application that can run select MLPerf inference benchmarks on smartphones.
- Improving reference implementations as starting points for development.
- Producing a "hyperparameter table" that maps system scale and precision to recommended hyperparameters for each benchmark.
- Developing better large public data sets for benchmarking and other purposes.
- Developing better software best practices for ML benchmarking and experimentation.

The MLPerf effort is now supported by more than 65 companies and researchers from eight educational institutions. We welcome the involvement of engineers and researchers who are interested in helping us make MLPerf better by going to mlperf.org/get-involved.

ACKNOWLEDGMENTS

MLPerf would not be possible without: B. Anderson, P. Bailis, V. Bittorf, M. Breughe, D. Brooks, M. Charlebois, D. Chen, W. Chou, R. Chukka, S. Davis, P. Deng, J. Duke, D. Dutta, D. Fick, J. S. Gardner, U. Gupta, K. Hazelwood, A. Hock, X. Huang, I. Hubara, S. Idgunji, T. B. Jablin, B. Jia, J. Jiao, D. Kang, P. Kanwar, N. Kumar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, D. Narayanan, T. Oguntebi, C. Osborne, G. Pekhimenko, L. Pentecost, A. T. R. Rajan, T. Robie, D. Sequeira, A. Sirasao, T. St. John, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, C. Young, B. Yu, G. Yuan, M. Zaharia, P. Zhang, A. Zhong, Y. Zhou, and many others.

REFERENCES

1. K. M. Dixit, "The SPEC benchmarks," *Parallel Comput.*, vol. 17, no. 10/11, pp. 1195–1209, 1991.
2. Baidu. "DeepBench: Benchmarking deep learning operations on different hardware," 2017. [Online]. Available: https://github.com/baidu-research/DeepBench
3. R. Adolf, S. Rama, B. Reagen, G.-Y. Wei, and D. Brooks, "Fathom: Reference workloads for modern deep learning methods," in *Proc. IEEE Int. Symp. Workload Characterization*, 2016, pp. 1–10.
4. C. Coleman *et al.*, "DAWNBench: An end-to-end deep learning benchmark and competition," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017.
5. P. Mattson *et al.*, "MLPerf training benchmark," 2019, *arXiv:1910.01500.*
6. V. Reddi *et al.*, "MLPerf inference benchmark," 2019, *arXiv:1911.02549.*
7. [Online]. Available: https://mlperf.org/dataset-and-model-credits/ presents full dataset and model citations.
8. [Online]. Available: https://mlperf.org/training-results-0-5/ and https://mlperf.org/training-results-0-6/ present complete results. https://github.com/mlperf/training_results_v0.6 and https://github.com/mlperf/training_results_v0.5 contain system details.
9. [Online]. Available: https://mlperf.org/inference-results/ present complete results and https://github.com/mlperf/inference_results_v0.5 contains system details.

Peter Mattson is currently a General Chair with MLPerf. He leads the ML Performance Metrics team at Google Brain. He received the Ph.D. degree from Stanford University, Stanford, CA, USA. Contact him at petermattson@google.com.

Vijay Janapa Reddi is currently an Inference Chair with MLPerf. He is an Associate Professor with Harvard University, Cambridge, MA, USA. He received the Ph.D. degree from Harvard University. Contact him at vj@ece.utexas.edu.

Christine Cheng is currently an Inference Chair with MLPerf. She is a Sr. Machine Learning Optimization Engineer with Intel, M.S., Stanford University, Stanford, CA, USA. Contact her at christine.cheng@intel.com.

Cody Coleman is currently a Research Chair with MLPerf. He is currently working toward the Ph.D. degree in computer science with Stanford University, Stanford, CA, USA. He received the M. Eng. degree from Massachusetts Institute of Technology, Cambridge, MA, USA. Contact him at cody@cs.stanford.edu.

Greg Diamos is currently a Data sets Chair with MLPerf. He leads AI Transformations team at Landing AI. He received the Ph.D. degree from Georgia Tech, Atlanta, GA, USA. Contact him at gregory.diamos@gmail.com.

David Kanter is currently an Inference and Power Chair with MLPerf. He is with Real World Technologies. He received the B.S. degree in mathematics and the B.A. degree in economics from the University of Chicago, Chicago, IL, USA. Contact him at dkanter@gmail.com.

Paulius Micikevicius is currently a Distinguished Engineer with NVIDIA. He received the Ph.D. degree from the University of Central Florida, Orlando, FL, USA. Contact him at pauliusm@nvidia.com.

David Patterson is currently a Google Brain Distinguished Engineer. He is a U.C. Berkeley Professor. He is a Vice-Chair Board of Directors of RISC-V Foundation. He received the Ph.D. degree from the University of California, Los Angeles, CA, USA. Contact him at pattrsn@cs.berkeley.edu.

Guenther Schmuelling is currently an Inference Results Chair with MLPerf. He is a Principal Tech Lead. He is with the Microsoft Azure AI Infrastructure. Contact him at guschmue@microsoft.com.

Hanlin Tang is currently a Sr. Director of AI Lab with Intel. He received the Ph.D. degree from Harvard University, Cambridge, MA, USA. Contact him at hanlin.tang@intel.com.

Gu-Yeon Wei is currently a Robert and Suzanne Case Professor of electrical engineering and computer science with Harvard University, Cambridge, MA, USA. He received the Ph.D. degree from Stanford University, Stanford, CA, USA. Contact him at gywei@g.harvard.edu.

Carole-Jean Wu is currently an Inference Chair with MLPerf. She is an Applied Research Scientist with Facebook. She is an Associate Professor at Arizona State University, Tempe, AZ, USA. She received the Ph.D. degree from Princeton University, Princeton, NJ, USA. Contact her at carolejeanwu@fb.com.