# BENCHMARKING TINYML SYSTEMS: CHALLENGES AND DIRECTION

**Colby R. Banbury** [1]  **Vijay Janapa Reddi** [1]  **Max Lam** [1]  **William Fu** [1]  **Amin Fazel** [2]  **Jeremy Holleman** [3][4]
**Xinyuan Huang** [5]  **Robert Hurtado** [6]  **David Kanter** [7]  **Anton Lokhmotov** [8]  **David Patterson** [9][10]  **Danilo Pau** [11]
**Jae-sun Seo** [12]  **Jeff Sieracki** [13]  **Urmish Thakker** [14]  **Marian Verhelst** [15][16]  **Poonam Yadav** [17]

## ABSTRACT

Recent advancements in ultra-low-power machine learning (TinyML) hardware promises to unlock an entirely new class of smart applications. However, continued progress is limited by the lack of a widely accepted benchmark for these systems. Benchmarking allows us to measure and thereby systematically compare, evaluate, and improve the performance of systems and is therefore fundamental to a field reaching maturity. In this position paper, we present the current landscape of TinyML and discuss the challenges and direction towards developing a fair and useful hardware benchmark for TinyML workloads. Furthermore, we present our four benchmarks and discuss our selection methodology. Our viewpoints reflect the collective thoughts of the TinyMLPerf working group that is comprised of over 30 organizations.

## 1 INTRODUCTION

Machine learning (ML) inference on the edge is an increasingly attractive prospect due to its potential for increasing energy efficiency (Fedorov et al., 2019), privacy, responsiveness (Zhang et al., 2017), and autonomy of edge devices. Thus far, the field edge ML has predominately focused on mobile inference which has led to numerous advancements in machine learning models such as exploiting pruning, sparsity, and quantization. But in recent years, there have major been strides in expanding the scope of edge systems. Interest is brewing in both academia (Fedorov et al., 2019; Zhang et al., 2017) and industry (Flamand et al., 2018; Warden, 2018a) towards expanding the scope of edge ML to microcontroller-class devices.

The goal of "TinyML" (tinyML Foundation, 2019) is to bring ML inference to ultra-low-power devices, typically under a milliWatt, and thereby break the traditional power barrier preventing widely distributed machine intelligence. By performing inference on-device, and near-sensor, TinyML enables greater responsiveness and privacy while avoiding the energy cost associated with wireless communication, which at this scale is far higher than that of compute (War-

den, 2018b). Furthermore, the efficiency of TinyML enables a class of smart, battery-powered, always-on applications that can revolutionize the real-time collection and processing of data. This emerging field, which is the culmination of many innovations, is poised only further to accelerate its growth in the coming years.

To unlock the full potential of the field, hardware software co-design is required. Specifically, TinyML models must be small enough to fit within the tight constraints of MCU-class devices (e.g., a few hundred kB of memory and limited onboard compute horsepower in the order of MHz processor clock speed), thus limiting the size of the input and the number of layers (Zhang et al., 2017) or necessitating the use lightweight, non-neural network-based techniques (Kumar et al., 2017). TinyML tools are broadly defined as anything that enables the design, mapping, and deployment of TinyML algorithms including aggressive quantization techniques (Wang et al., 2019), memory aware neural architecture searches (Fedorov et al., 2019), frameworks (TensorFlow), and efficient inference libraries (Lai et al., 2018; Garofalo et al., 2019). Efforts in TinyML hardware include improving inference on the next generation of general-purpose MCUs (arm; Flamand et al., 2018), developing hardware specialized for low power inference, and creating novel architectures intended only as inference engines for specific tasks (Moons et al., 2018).

The complexity and dynamicity of the field obscure the measurement of progress and make dynamism design decisions intractable. In order to enable the continued innovation, a fair and reliable method of comparison is needed. Since progress is often the result of increased hardware capability,

---

[1]Harvard University [2]Samsung Semiconductor, Inc. [3]Syntiant [4]University of North Carolina, Charlotte [5]Cisco Systems [6]California State Polytechnic University, Pomona [7]Real World Insights [8]dividiti [9]University of California, Berkeley [10]Google [11]STMicroelectronics, Italy [12]Arizona State University [13]Reality AI [14]Arm ML Research Lab [15]KU Leuven [16]Interuniversity Microelectronics Centre (IMEC) [17]University of York. Correspondence to: Colby R. Banbury <cbanbury@g.harvard.edu>.

a reliable TinyML hardware benchmark is required.

In this paper, we discuss the challenges and opportunities associated with the development of a TinyML hardware benchmark. Our short paper is a call to action for establishing a common benchmarking for TinyML workloads on emerging TinyML hardware to foster the development of TinyML applications. The points presented here reflect the ongoing effort of the TinyMLPerf working group that is currently comprised of over 30 organizations and 75 members.

The rest of the paper is organized as follows. In Section 2, we discuss the application landscape of TinyML, including the existing use cases, models, and datasets. In Section 3, we describe the existing TinyML hardware solutions, including outlining improvements to general-purpose MCUs and the development of novel architectures. In Section 4, we discuss the inherent challenges of the field and how they complicate the development of a benchmark. In Section 5, we describe the existing benchmarks that relate to TinyML and identify the deficiencies that still need to be filled. In Section 6 we discuss the progress of the TinyMLPerf working group thus far and describe the four benchmarks. In Section 7, we concluded the paper and discuss future work.

## 2 TINY USE CASES, MODELS & DATASETS

In this section we attempt to summarize the field of TinyML by describing a set of representative use cases (Section 2.1), their relevant datasets (Section 2.2), and the model architectures commonly applied to these specific use cases (Section 2.3).

### 2.1 Use Cases

Despite the general lack of maturity within the field, there are a number of well established TinyML use cases. We categorize the application landscape of tiny ML by input type in Table 3, which in the context of TinyML systems plays a crucial role in the use case definition.

Audio wake words is already a fairly ubiquitous example of always-on ML inference. Audio wake words is generally a speech classification problem that achieves very low power inference by limiting the label space, often to two labels: "wake word" and "not wake word" (Zhang et al., 2017).

Anomaly detection and predictive maintenance are commonly deployed on MCUs in factory settings where audio, motor bearing, or IMU data can be used to detect faults in products or equipment.

Other deployed TinyML applications, like activity recognition from IMU data (Hassan et al., 2018), rely on low feature dimensionality to fit within the tight constraints of the platforms. Some use cases have been proven viable, but have yet to reach end users because they are too new, like

visual wake words (Chowdhery et al., 2019).

Many traditional ML use cases can be considered futuristic TinyML tasks. As ultra-low-power inference hardware continues to improve, the threshold of viability expands. Tasks like large label space image classification or object counting are well suited for low-power always-on applications but are currently too compute and memory hungry for today's TinyML hardware.

Furthermore, TinyML has a significant role to play in future technology. For example, many of the fundamental features of augmented reality (AR) glasses are always-on and battery-powered. Due to tight real time constraints, these devices cannot afford the latency of offloading computation to the cloud, an edge server, or even an accompanying mobile device. Thus, due to shared constraints, AR applications can benefit significantly from progress in the field of TinyML.

### 2.2 Datasets

There are a number of open-source datasets that are relevant to TinyML usecases. Table 3 breaks them down by the type of data. Despite the availability of these datasets, the majority of deployed TinyML models are trained on much larger, proprietary datasets. The open-source datasets that are competitively large are not TinyML specific. The lack of large, TinyML focused, open-source datasets slows the progress of academic research and limits the ability of a benchmark to represent real workloads accurately.

### 2.3 Models

Table 3 lists common model types for TinyML use cases. Although neural networks (NN) are a dominant force in traditional ML, it is common to use non-NN based solutions like decision trees (Kumar et al., 2017), for some TinyML use cases, due to their low compute and memory requirements.

Machine learning on MCU-class devices has only recently become feasible; therefore, the community has yet to produce models that have become widely accepted as MobileNets have become for mobile devices. This makes the task of selecting representative models challenging. However, immaturity also brings opportunity as our decisions can help direct future progress. Selecting a subset of the currently available models, outlining the rules for quality versus accuracy trade-offs, and prescribing a measurement methodology that can be faithfully reproduced will encourage the community to develop new models, runtimes, and hardware that progressively outperform one another.

*Table 1.* Survey of TinyML Use Cases, Models, and Datasets

| INPUT TYPE | USE CASES | MODEL TYPES | DATASETS |
|---|---|---|---|
| AUDIO | AUDIO WAKE WORDS<br>CONTEXT RECOGNITION<br>CONTROL WORDS<br>KEYWORD DETECTION | DNN<br>CNN<br>RNN<br>LSTM | SPEECH COMMANDS (WARDEN, 2018A)<br>AUDIOSET (GEMMEKE ET AL., 2017)<br>EXTRASENSORY (VAIZMAN ET AL., 2017) |
| IMAGE | VISUAL WAKE WORDS<br>OBJECT DETECTION<br>IMAGE CLASSIFICATION<br>GESTURE RECOGNITION<br>OBJECT COUNTING<br>TEXT RECOGNITION | DNN<br>CNN<br>SVM<br>DECISION TREES<br>KNN<br>LINEAR | VISUAL WAKE WORDS (CHOWDHERY ET AL., 2019)<br>CIFAR10 (KRIZHEVSKY ET AL., 2009B)<br>MNIST (LECUN & CORTES, 2010)<br>IMAGENET (DENG ET AL., 2009)<br>DVS128 GESTURE (AMIR ET AL., 2017) |
| PHYSIOLOGICAL /<br>BEHAVIORAL<br>METRICS | SEGMENTATION<br>FORECASTING<br>ACTIVITY DETECTION | DNN<br>DECISION TREE<br>SVM<br>LINEAR | PHYSIONET (GOLDBERGER ET AL., 2000)<br>HAR (CRAMARIUC, 2019)<br>DSA (ALTUN ET AL., 2010)<br>OPPORTUNITY (ROGGEN ET AL., 2010)<br>UCI EMG (LOBOV ET AL., 2018) |
| INDUSTRY<br>TELEMETRY | SENSING (LIGHT, TEMP, ETC)<br>ANOMALY DETECTION<br>MOTOR CONTROL<br>PREDICTIVE MAINTENANCE | DNN<br>DECISION TREE<br>SVM<br>LINEAR<br>NAIVE BAYES | UCI AIR QUALITY (DE VITO ET AL., 2008)<br>UCI GAS (VERGARA ET AL., 2012)<br>NASA'S PCOE (SAXENA & GOEBEL, 2008) |
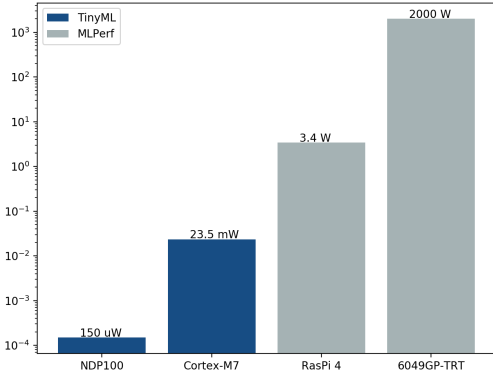


*Figure 1.* A logarithmic comparison of the active power consumption between TinyML systems and those supported by MLPerf. TinyML systems can be up to four orders of magnitude smaller in the power budget as compared to state-of-the-art MLPerf systems.

## 3 TINY HARDWARE CONSTRAINTS

TinyML hardware is defined by its ultra-low power consumption, which is often in the range of 1 mWatt and below. At the top of this range are efficient 32-bit MCUs, like those based on the Arm Cortex-M7 or RISC-V PULP processors, and at the bottom are novel ultra-low-power inference engines. Even the largest TinyML devices consume drastically less power than the smallest traditional ML devices. Figure 1 shows the logarithmic comparison of the active power consumption between TinyML devices and those currently supported by MLPerf (v0.5 inference results from the open and closed divisions). TinyML devices can be up to four orders of magnitude smaller in the power budget as compared to state-of-the-art MLPerf systems.

The advent of low-power, cheap 32-bit MCUs have revolutionized the compute capability at the very edge. Cortex-M based platforms are now regularly performing tasks that were previously infeasible at this scale, mostly due to support for single instruction multiple data (SIMD) and digital signal processing (DSP) instructions. This fast vector math supports NN and highly efficient SVM implementations, it also accelerates many feature computations using 8bit fixed point arithmetic.

A feature of MCUs is the prevalence of on-chip SRAM and embedded Flash. Thus, when models can fit within the tight on-chip memory constraints, they are free of the costly DRAM accesses that hamper traditional ML. Widespread adoption and dispersion of TinyML are reliant on the capa-

bility of these platforms.

Although general-purpose MCUs provide flexibility, the highest TinyML performance efficiency comes from specialized hardware. Novel architectures can achieve performance in the range of one micro Joule per inference (Holleman, 2019). These specialized devices expand the boundaries of ML to the ultra low power end of TinyML processors.

# 4 CHALLENGES

TinyML systems present a number of unique challenges to the design of a performance benchmark that can be used to measure and quantify performance differences between various systems systematically. We discuss the four primary obstacles and postulate how they might be overcome.

## 4.1 Low Power

Low power consumption is one of the defining features of TinyML systems. Therefore, a useful benchmark should ostensibly profile the energy efficiency of each device. However, there are many challenges in fairly measuring energy consumption. Firstly, as illustrated in Figure 1, TinyML devices can consume drastically different amounts of power, which makes maintaining accuracy across the range of devices difficult.

Secondly, determining what falls under the scope of the power measurement is difficult to determine when data paths and pre-processing steps can vary significantly between devices. Other factors like chip peripherals and underlying firmware can impact the measurements. Unlike traditional high-power ML systems, TinyML systems do not have spare cores to load the System-Under-Test (SUT) with minimal overheads.

## 4.2 Limited Memory

Due to their small size, TinyML systems often have tight memory constraints. While traditional ML systems like smartphones cope with resource constraints in the order of a few GBs, tinyML systems are typically coping with resources that are two orders of magnitude smaller.

Memory is one of the primary motivating factors for the creation of a TinyML specific benchmark. Traditional ML benchmarks use inference models that have drastically higher peak memory requirements (in the order of gigabytes) than TinyML devices can provide. This also complicates the deployment of a benchmarking suite as any overhead can significantly impact power consumption or even make the benchmark too big to fit. Individual benchmarks must also cover a wide range of devices; therefore, multiple levels of quantization and precision should be represented in the benchmarking suite. Finally, a variety of benchmarks should

be chosen such that the diversity of the field is supported.

## 4.3 Hardware Heterogeneity

Despite its nascency, TinyML systems are already diverse in their performance, power, and capabilities. Devices range from general-purpose MCUs to novel architectures, like in event-based neural processors (Brainchip) or memory compute (Kim et al., 2019). This heterogeneity poses a number of challenges as the system under test (SUT) will not necessarily include otherwise standard features, like a system clock or debug interface. Furthermore, the task of normalizing performance results across heterogeneous implementations is a key challenge.

Today's state-of-the-art benchmarks are not designed to handle the challenges readily. They need careful re-engineering to be flexible enough to handle the extent of hardware heterogeneity that is commonplace in the TinyML ecosystem.

## 4.4 Software Heterogeneity

There are three distinct methods for model deployment on to TinyML systems: hand coding, code generation, and ML interpreters.

Hand coding often produces the best results as it allows for low-level, application specific optimizations; however, the task is time consuming and the impact of the optimizations are often opaque to anyone but the original design team. Moreover, hand coding limits the ability to share knowledge and adopt new methods, which is detrimental to the rate of progress in TinyML. From a benchmarking perspective, hand coded submission will likely produce the best numerical results at the cost of reproducibility, comparability and time.

Code generation methods produce well optimized code without the significant effort of hand coding by abstracting and automating system level optimizations. However, code generation does not address the issues with comparability, as each major vendor has their own set of proprietary tools and compilers, which also makes portability a challenge.

ML interpreters allow for significant portability as their abstract structure is the same across platforms. TensorFlow Lite for Microcontrollers, a popular ML framework for TinyML, uses an interpreter to call individual kernels, like convolution, during run time. The framework is independent of the model architecture, therefore new models can be easily swapped in. Additionally, the reference kernels can be individually optimised and changed to fit the platform. This method comes with a small overhead in binary size and performance. From a benchmarking perspective, this abstraction separates the impact of the model architecture on the system level performance, which makes results more generalizable.

*Table 2.* Existing Benchmarks

| BENCHMARK | ML? | POWER? | TINY? |
|---|---|---|---|
| COREMARK | × | √ | √ |
| MLMARK | √ | × | × |
| MLPERF INFERENCE | √ | √ | × |
| TINYML REQUIREMENTS | √ | √ | √ |

A benchmark suite must balance optimality with portability, and comparibility with representativeness. A TinyML benchmark should support many options for model deployment but the impact of that choice on the results must be carefully evaluated.

## 5 RELATED WORK

There are a number of ML related hardware benchmarks, however, none that accurately represent the performance of TinyML workloads on tiny hardware. Table 2 shows a sampling of the widely accepted industry benchmarks that are directly applicable to the discussion on TinyML systems.

EEMBC CoreMark (Gal-On & Levy) has become the standard performance benchmark for MCU-class devices due to its ease of implementation and use of real algorithms. Yet, CoreMark does not profile full programs, nor does it accurately represent machine learning inference workloads.

EEMBC MLMark (Torelli & Bangale) addresses these issues by using actual ML inference workloads. However, the supported models are far too large for MCU-class devices and are not representative of TinyML workloads. They require far too much memory (GBs) and have significant run times. Additionally, while CoreMark supports power measurements with ULPMark-CM (EEMBC), MLMark does not, which is critical for a TinyML benchmark.

MLPerf, a community-driven benchmarking effort, has recently introduced a benchmarking suite for ML inference (Reddi et al., 2019) and has plans to add power measurements. However, much like MLMark, the current MLPerf inference benchmark precludes MCUs and other resource-constrained platforms due to a lack of small benchmarks and compatible implementations.

As Table 2 summarizes, there is a clear and distinct need for a TinyML benchmark that caters to the unique needs of ML workloads, makes power a first-class citizen and prescribes a methodology that suits TinyML.

## 6 BENCHMARKS

To overcome theses challenges, we adopt a set of principles for the development of a robust TinyML benchmarking suite and select a set of 4 benchmarks.

### 6.1 Open and Closed Divisions

As previously stated, TinyML is a diverse field, therefore not all systems can be accommodated under strict rules, however, without strict rules, direct comparison of the hardware becomes more difficult. To address this issue, we adopt MLPerf's open and closed structure. More traditional TinyML solutions can submit to the closed division where submissions must use a model that is considered equivalent to the reference model. TinyML systems that fall outside the bounds of the "closed" benchmark can submit results to the open division which will allow submissions to deviate as necessary from the closed reference. We believe this structure increases the inclusivity of the bechmarking suite while maintaining the comparability of the results.

Additionally, the open division allows for submissions to demonstrate novel software optimizations. Software based organizations can submit results using the reference platform while altering the model or inference engine to demonstrate the relative advantage of their unique solutions.

### 6.2 Use Cases

Our use case selection process prioritized diversity, feasibility, and industry relevance. Diversity to ensure our benchmark suite covered as much of the field as possible, feasibility in terms of access to open source datasets and models, and relevance to real world applications.

The group has selected four use cases to target: audio wake words, visual wake words, image classification, and anomaly detection. Audio wake words refers to the common, keyword spotting task (e.g. "Alexa", "Ok Google", and "Hey Siri"). Visual wake words is a binary image classification task that indicates if a person is visible in the image or not. The image classification use cases targets small label set size image classification. Anomaly detection is a broader use case that classifies time series data as "normal" or "abnormal". We specifically select audio anomaly detection as our use case due to the availability of a relevant dataset.

These use cases have been selected to represent the broad range of TinyML. They encompass three distinct input data types and range from relatively resource hungry (visual wake words) to light weight (anomaly detection). Furthermore the models traditionally used for these use cases are varied therefore the benchmarking suite can support a diverse set of ML techniques.

### 6.3 Dataset Selection

The group has selected a dataset for each use case, as shown in Table 3. The datasets help specify the use cases, are used to train the reference models, and are sampled to create the tests sets used during the measurement on device. Further-

*Table 3.* TinyMLPerf Benchmarking Suite

| USE CASE | DATASETS | MODEL | |
|----------|----------|-------|---|
| AUDIO WAKE WORDS | SPEECH COMMANDS (WARDEN, 2018A) | DS-CNN (ZHANG ET AL., 2017) | |
| VISUAL WAKE WORDS | VISUAL WAKE WORDS DATASET (CHOWDHERY ET AL., 2019) | DS-CNN (TFLM-PERSON-DETECTION) | |
| IMAGE CLASSIFICATION | CIFAR10(KRIZHEVSKY ET AL., 2009A) | RESNET 8 (HE ET AL., 2016) | |
| ANOMALY DETECTION | TOYADMOS (TOY CAR)(KOIZUMI ET AL., 2019) | DEEP AUTOENCODER (KOIZUMI ET AL., 2020) | |

more, the datasets can be used to train a new or modified model in the open division. We have selected datasets that are open, well known, and relevant to industry use cases.

### 6.4 Model Selection

The group has selected four reference models. These reference models are the benchmark workloads in the closed division and act as a baseline for the open division. The DS-CNN described in (Zhang et al., 2017) have been selected for audio wake words. The MobilenetV1(Howard et al., 2017) used in the TensorFlow Lite for Microcontrollers person detection example (TFLM-Person-Detection) has been selected for visual wake words. An eight layer ResNet model (He et al., 2016) has been selected for image classification. The baseline deep autoencoder from Task 2 of DCASE2020 competetition has been selected for anomaly detection. (Koizumi et al., 2020). The models were selected, based on industry input, to be representative of their respective use cases.

### 6.5 Metrics

The benchmarking suite will primarily measure inference latency with the option to measure energy consumption. The scope of the the the measurements is determined by each benchmark. In the open division the accuracy of the model must remain within a set threshold of the closed division model.

### 6.6 Future work

Perfection is often the enemy of good, therefore, to fill the community's need for comparability, our priority is to quickly establish a set of minimum viable benchmarks and iteratively address deficiencies. The benchmarking suite will continue to evolve to meet the needs of the community.

We plan to accept result submissions in March of 2021.

### 7 CONCLUSION

In conclusion, TinyML is an important and rapidly evolving field that requires comparability amongst hardware innovations to enable continued progress and stability. In this paper, we reviewed the current landscape of TinyML, including highlighting the need for a hardware benchmark. Additionally, we analyzed challenges associated with developing said benchmark and discussed a path forward. Finally, we have selected use cases, datasets, and models for our four benchmarks.

If you would like to contribute to the effort, join the working group here: https://groups.google.com/u/4/a/mlcommons.org/g/tiny

The benchmark suite is available here: https://github.com/mlcommons/tiny

### REFERENCES

Helium: Enhancing the capabilities of the smallest devices. URL https://www.arm.com/why-arm/technologies/helium.

Altun, K., Barshan, B., and Tunçel, O. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*, 43:3605–3620, 10 2010. doi: 10.1016/j.patcog.2010.04.019.

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Nolfo, C. D., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., and Modha, D. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern*

*Recognition (CVPR)*, pp. 7388–7397, July 2017. doi: 10.1109/CVPR.2017.781.

Brainchip. Akida neuromorphic system on chip. URL https://www.brainchipinc.com/products/akida-neuromorphic-system-on-chip.

Chowdhery, A., Warden, P., Shlens, J., Howard, A., and Rhodes, R. Visual wake words dataset. *CoRR*, abs/1906.05721, 2019. URL http://arxiv.org/abs/1906.05721.

Cramariuc, A.-C. P. I. M. B. Precis har, 2019. URL http://dx.doi.org/10.21227/mene-ck48.

De Vito, S., Massera, E., Piga, M., and Martinotto, L. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B Chemical*, 129:750–757, 02 2008. doi: 10.1016/j.snb.2007.09.060.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

EEMBC. Ulpmark - an eembc benchmark. URL https://www.eembc.org/ulpmark/index.php.

Fedorov, I., Adams, R. P., Mattina, M., and Whatmough, P. Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers. In *Advances in Neural Information Processing Systems 32*, pp. 4978–4990. Curran Associates, Inc., 2019.

Flamand, E., Rossi, D., Conti, F., Loi, I., Pullini, A., Rotenberg, F., and Benini, L. Gap-8: A risc-v soc for ai at the edge of the iot. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 1–4, July 2018. doi: 10.1109/ASAP.2018.8445101.

Gal-On, S. and Levy, M. Exploring coremark - a benchmark maximizing simplicity and efficacy. Technical report. URL https://www.eembc.org/techlit/articles/coremark-whitepaper.pdf.

Garofalo, A., Rusci, M., Conti, F., Rossi, D., and Benini, L. Pulp-nn: accelerating quantized neural networks on parallel ultra-low-power risc-v processors. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2164): 20190155, Dec 2019. ISSN 1471-2962. doi: 10.1098/rsta.2019.0155. URL http://dx.doi.org/10.1098/rsta.2019.0155.

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

Hassan, M. M., Uddin, M. Z., Mohamed, A., and Almogren, A. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, 81:307–313, 2018.

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

Holleman, J. The speed and power advantage of a purpose-built neural compute engine, Jun 2019. URL https://www.syntiant.com/post/keyword-spotting-power-comparison.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Kim, H., Chen, Q., Yoo, T., Kim, T. T.-H., and Kim, B. A 1-16b precision reconfigurable digital in-memory computing macro featuring column-mac architecture and bit-serial computation. In *ESSCIRC 2019-IEEE 45th European Solid State Circuits Conference (ESSCIRC)*, pp. 345–348. IEEE, 2019.

Koizumi, Y., Saito, S., Uematsu, H., Harada, N., and Imoto, K. Toyadmos: A dataset of miniature-machine operating sounds for anomalous sound detection. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 313–317. IEEE, 2019.

Koizumi, Y., Kawaguchi, Y., Imoto, K., Nakamura, T., Nikaido, Y., Tanabe, R., Purohit, H., Suefusa, K., Endo, T., Yasuda, M., and Harada, N. Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring. In *arXiv e-prints: 2006.05822*, pp. 1–4, June 2020. URL https://arxiv.org/abs/2006.05822.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009a.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). 2009b. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Kumar, A., Goyal, S., and Varma, M. Resource-efficient machine learning in 2 KB RAM for the internet of things. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1935–1944, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/kumar17a.html.

Lai, L., Suda, N., and Chandra, V. Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus, 2018.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Lobov, S., Krilova, N., Kastalskiy, I., Kazantsev, V., and Makarov, V. Latent factors limiting the performance of semg-interfaces. *Sensors*, 18:1122, 04 2018. doi: 10.3390/s18041122.

Moons, B., Bankman, D., Yang, L., Murmann, B., and Verhelst, M. Binareye: An always-on energy-accuracy-scalable binary cnn processor with all memory on chip in 28nm cmos. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4. IEEE, 2018.

Reddi, V. J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C.-J., Anderson, B., Breughe, M., Charlebois, M., Chou, W., Chukka, R., Coleman, C., Davis, S., Deng, P., Diamos, G., Duke, J., Fick, D., Gardner, J. S., Hubara, I., Idgunji, S., Jablin, T. B., Jiao, J., John, T. S., Kanwar, P., Lee, D., Liao, J., Lokhmotov, A., Massa, F., Meng, P., Micikevicius, P., Osborne, C., Pekhimenko, G., Rajan, A. T. R., Sequeira, D., Sirasao, A., Sun, F., Tang, H., Thomson, M., Wei, F., Wu, E., Xu, L., Yamada, K., Yu, B., Yuan, G., Zhong, A., Zhang, P., and Zhou, Y. Mlperf inference benchmark, 2019.

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Sagha, H., Bayati, H., Creatura, M., and d. R. Millàn, J. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, pp. 233–240, June 2010. doi: 10.1109/INSS.2010.5573462.

Saxena, A. and Goebel, K. Turbofan engine degradation simulation data set, 2008. URL http://ti.arc.nasa.gov/project/prognostic-data-repository.

TensorFlow. Tensorflow lite for microcontrollers. URL https://www.tensorflow.org/lite/microcontrollers.

TFLM-Person-Detection. Tensorflow lite for microcontrollers person detection example. URL https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/micro/examples/person_detection.

tinyML Foundation. tinyml summit, 2019. URL https://www.tinymlsummit.org/.

Torelli, P. and Bangale, M. Measuring inference performance of machine-learning frameworks on edge-class devices with the mlmark benchmark. URL https://www.eembc.org/techlit/articles/MLMARK-WHITEPAPER-FINAL-1.pdf.

Vaizman, Y., Ellis, K., and Lanckriet, G. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, 16(4):62–74, October 2017. ISSN 1558-2590. doi: 10.1109/MPRV.2017.3971131.

Vergara, A., Vembu, S., Ayhan, T., Ryan, M., Homer, M., and Huerta, R. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, s 166–167:320–329, 05 2012. doi: 10.1016/j.snb.2012.01.074.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition, 2018a.

Warden, P. why the future of machine learning is tiny, 2018b. URL https://petewarden.com/2018/06/11/why-the-future-of-machine-learning-is-tiny/.

Zhang, Y., Suda, N., Lai, L., and Chandra, V. Hello edge: Keyword spotting on microcontrollers, 2017.