TINYMLPERF
Development of a Benchmark Suite for tinyML Systems

Csaba Kiraly, Digital Catapult
Colby Banbury, Harvard University
IMPORTANCE OF BENCHMARKING

GIVEN A TASK AT HAND HOW DO WE KNOW WHICH IS THE **RIGHT SYSTEM**?

HOW DO WE **COMPARE** THE DIFFERENT SOLUTIONS?
BENCHMARKING IS ALL ABOUT …

- **Comparability** across hardware and software
- **Standardization** of use cases and workloads
- **Measuring** progress via rigorous methodology
- **Community** building and consensus generation
NEED FOR A TINYML SPECIFIC BENCHMARK

- MLPerf Inference
- DawnBench
- Mobile AI Bench
- MLPerf Mobile
- tinyMLPerf

- Embedded Systems
- Coremark
- Embench
- ULPMark
BENCHMARK DESIGN CHALLENGES

- Number and diversity of use cases for tinyML
  - Which use cases should we focus on?

- TinyML innovation is in HW, in SW, in tooling, in algorithms, etc.
  - How do we support fair comparison and innovation?

- Embedded system design is always a compromise
  - What metrics to choose to fairly evaluate systems?
BENCHMARK DESIGN CHALLENGES

- **Number and diversity of use cases for tinyML**
  - Which use cases should we focus on?

- **TinyML innovation is in HW, in SW, in tooling, in algorithms, etc.**
  - How do we support fair comparison and innovation?

- **Embedded system design is always a compromise**
  - What metrics to choose to fairly evaluate systems?
# Relevant Use Cases

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Use Case</th>
<th>Model Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>Audio Wake Words</td>
<td>DNN</td>
</tr>
<tr>
<td></td>
<td>Context Recognition</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Control Words</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Keyword Detection</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>Visual Wake Words</td>
<td>DNN</td>
</tr>
<tr>
<td></td>
<td>Object Detection</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Gesture Recognition</td>
<td>SVM</td>
</tr>
<tr>
<td></td>
<td>Object Counting</td>
<td>Decision Tree</td>
</tr>
<tr>
<td></td>
<td>Text Recognition</td>
<td>KNN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linear</td>
</tr>
<tr>
<td>Industry / Telemetry</td>
<td>Segmentation</td>
<td>DNN</td>
</tr>
<tr>
<td></td>
<td>Anomaly Detection</td>
<td>Decision Tree</td>
</tr>
<tr>
<td></td>
<td>Forecasting</td>
<td>SVM</td>
</tr>
<tr>
<td></td>
<td>Activity Detection</td>
<td>Linear</td>
</tr>
</tbody>
</table>
## RELEVANT USE CASES & COMMUNITY

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Use Case</th>
<th>Model Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>Audio Wake Words</td>
<td>DNN</td>
</tr>
<tr>
<td></td>
<td>Context Recognition</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Control Words</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Keyword Detection</td>
<td>LSTM</td>
</tr>
<tr>
<td>Image</td>
<td>Visual Wake Words</td>
<td>DNN</td>
</tr>
<tr>
<td></td>
<td>Object Detection</td>
<td>SVM</td>
</tr>
<tr>
<td></td>
<td>Gesture Recognition</td>
<td>Decision Tree</td>
</tr>
<tr>
<td></td>
<td>Object Counting</td>
<td>KNN</td>
</tr>
<tr>
<td></td>
<td>Text Recognition</td>
<td>Linear</td>
</tr>
<tr>
<td>Industry / Telemetry</td>
<td>Segmentation</td>
<td>DNN</td>
</tr>
<tr>
<td></td>
<td>Anomaly Detection</td>
<td>Decision Tree</td>
</tr>
<tr>
<td></td>
<td>Forecasting</td>
<td>SVM</td>
</tr>
<tr>
<td></td>
<td>Activity Detection</td>
<td>Linear</td>
</tr>
</tbody>
</table>

**WG lead:** Vijay Janapa Reddi + Colby Banbury @ Harvard

**Benchmark harness:** Peter Torelly @ EEMBC + Nat Jeffries @ Google

- Jeremy Holleman @ Syntiant
- Nat Jeffries @ Google
- Pietro Montino
- Csaba Kiraly @ DC
SELECTED BENCHMARKS

Keyword Spotting


Visual Wake Words


Anomaly Detection


Tiny Image Classification

BENCHMARK DESIGN CHALLENGES

- Number and diversity of use cases for tinyML
  - Which use cases should we focus on?

- TinyML innovation is in HW, in SW, in tooling, in algorithms, etc.
  - How do we support fair comparison and innovation?

- Embedded system design is always a compromise
  - What metrics to choose to fairly evaluate systems?
Modular Design

Direct Comparison
CLOSED DIVISION

Closed Submission A
- Compiler
- Inference Framework
- Kernels
- SoC
- Accelerator

Closed Submission B
- Compiler
- Inference Framework
- Kernels
- SoC
- Accelerator

VS.

Demonstrate Improvement
OPEN DIVISION

TinyMLPerf
- Training Data
- Training Script
- Model Architecture
- Pruning
- Quantization
- Compiler
- Inference Framework
- Kernels
- SoC
- CPU
- Accelerator

Open Submission
- Training Script
- Model Architecture

2X Faster vs. Reference
BENCHMARK DESIGN CHALLENGES

- Number and diversity of use cases for tinyML
  - Which use cases should we focus on?

- TinyML innovation is in HW, in SW, in tooling, in algorithms, etc.
  - How do we support fair comparison and innovation?

- Embedded system design is always a compromise
  - What metrics to choose to fairly evaluate systems?
METRICS OF INTEREST

- What metrics to choose, how to define
  - latency: ms/inference, or ms/task?
  - accuracy: which metric?
  - energy: what should be included?

- (Some) measurement difficulties
  - pre/post processing
  - single inference vs. more complex processing
  - accurate power measurement
  - accurate timestamping

- EEMBC Benchmark Runner
MEASURING THOSE METRICS

Latency

Accurate timestamping
Evaluated on series of inferences
Data-dependant execution?

Accuracy

Top-1 accuracy & AUC
CLOSED: meet threshold
OPEN: part of the metrics

Energy

Power Monitor integration
No “cherry-picking”
Median result

---

Runtime requirements have been met.
Performance results for window 10:
# Inferences : 1000
Runtime : 10.524 sec.
Throughput : 95.029 inf./sec.
Runtime requirements have been met.

Median throughput is 95.019 inf./sec.
ALL TOGETHER: THE FULL BENCHMARK FLOW

Workgroup

Benchmark definition

- Problem definition
- Dataset selection (public domain)
- Model selection / training
- Derive “Tiny” version: Quantization, etc.
- Write C implementation
- Integrate with benchmarking harness
- Deploy on device
- Benchmark
  - ML performance
  - Power consumption
- Review and Audit

Reference implementation

Submitter

Open submission

Closed submission
ALL TOGETHER: ANOMALY DETECTION EXAMPLE

1. Problem definition
2. Dataset selection (public domain)
3. Reference model selection
4. Derive “Tiny” version: Quantization, etc.
5. Write C implementation
6. Integrate with benchmarking harness
7. Deploy on device
8. Benchmark
9. Review and Audit
EXAMPLE: ANOMALY DETECTION

Input

Anomalous Sound Detection System

Output

Normal

Anomaly

- Pre-processing: Spectrogram

- Sliding window

- Anomaly score on individual window
  AutoEncoder

- Post-processing:
  average score
REFERENCE IMPLEMENTATION

- Open platform
- Widely known, available, and affordable
- ARM Cortex-M4 with FPU
- 2 MB Flash
- 640KB SRAM

ST NUCLEO-L4R5ZI

- Open source
- Portability of reference implementations

- Open source
- Full toolchain with everything we need
- Improving quantization support
TINYMLPERF: WHERE NEXT?

V0.1.0 is out bootstrapping the process

Just the beginning …
Grow the community
Refine and define use cases towards V0.2.0
Premier Sponsor
Automated TinyML

Zero-code SaaS solution

Create tiny models, ready for embedding, in just a few clicks!

Compare the benchmarks of our compact models to those of TensorFlow and other leading neural network frameworks.

Executive Sponsors
Optimized models for embedded (e.g. TensorFlow Lite Micro)

Optimized low-level NN libraries (i.e. CMSIS-NN)

RTOS such as Mbed OS

Arm Cortex-M CPUs and microNPUs

Profiling and debugging tooling such as Arm Keil MDK

Application

Supported by end-to-end tooling

Connect to high-level frameworks

Connect to Runtime

AI Ecosystem Partners

Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm
TinyML for all developers

Dataset
Acquire valuable training data securely

Enrich data and train ML algorithms

Edge Device
Real sensors in real time
Open source SDK

Embedded and edge compute deployment options

Test
Test impulse with real-time device data flows

www.edgeimpulse.com
Advancing AI research to make efficient AI ubiquitous

Power efficiency
- Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization
- Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning
- Robust learning through minimal data, unsupervised learning, on-device learning

Perception
- Object detection, speech recognition, contextual fusion

Reasoning
- Scene understanding, language understanding, behavior prediction

Action
- Reinforcement learning for decision making

A platform to scale AI across the industry

Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.
Syntiant Corp. is moving artificial intelligence and machine learning from the cloud to edge devices. Syntiant’s chip solutions merge deep learning with semiconductor design to produce ultra-low-power, high performance, deep neural network processors. These network processors enable always-on applications in battery-powered devices, such as smartphones, smart speakers, earbuds, hearing aids, and laptops. Syntiant's Neural Decision Processors™ offer wake word, command word, and event detection in a chip for always-on voice and sensor applications.

SYNTIANT

Syntiant was recently named a **CES® 2021 Best of Innovation Awards Honoree**, shipped over **10M units worldwide**, and **unveiled the NDP120** part of the NDP10x family of inference engines for low-power applications.

www.syntiant.com  @Syntiantcorp
Platinum Sponsors
Part of your life. Part of tomorrow.

www.infineon.com
Add Advanced Sensing to your Product with Edge AI / TinyML

Pre-built Edge AI sensing modules, plus tools to build your own

Reality AI solutions
- Prebuilt sound recognition models for indoor and outdoor use cases
- Solution for industrial anomaly detection
- Pre-built automotive solution that lets cars "see with sound"

Reality AI Tools® software
- Build prototypes, then turn them into real products
- Explain ML models and relate the function to the physics
- Optimize the hardware, including sensor selection and placement

https://reality.ai  info@reality.ai  @SensorAI  LinkedIn Reality AI
Gold Sponsors
Adaptive AI for the Intelligent Edge

Latent AI

Latentai.com
Build Smart IoT Sensor Devices From Data

SensiML pioneered TinyML software tools that auto generate AI code for the intelligent edge.

• End-to-end AI workflow
• Multi-user auto-labeling of time-series data
• Code transparency and customization at each step in the pipeline

We enable the creation of production-grade smart sensor devices.

sensiml.com
Silver Sponsors
Copyright Notice

The presentation(s) in this publication comprise the proceedings of tinyML® EMEA Technical Forum 2021. The content reflects the opinion of the authors and their respective companies. This version of the presentation may differ from the version that was presented at tinyML EMEA. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyML.org