CS249r: Acceleration for ML: GPUs, TPUs and FPGAs

Oct 23

Course Logistics

Assignment - Why, What, How

- Assignment 1
  - **Why**: Vision (Camera) use cases + Dataset generation
  - **What**: End-to-end ML workflow
  - **How**: Frameworks (Edge Impulse) + Embedded Hardware (Nicla)
- Assignment 2
  - **Why**: Audio (Microphone) use cases
  - **What**: ML Pipeline (Data preprocessing + Model optimizations)
  - **How**: NAS (Edge Impulse) + Quantization/Pruning (TFLite)
- Assignment 3 - Model development
  - **Why**: Time series (IMU) use cases
  - **What**: ML frameworks programming (Tensorflow/TFLite/TFLM)
  - **How**: Bare metal implementation of end-to-end ML workflow
- Assignment 4 - Responsible AI
  - **Why**: Sustainability + Responsibility
  - **What**: Sustainability-aware system design and safe AI
  - **How**: TinyML Footprint + Adversarial Nibbler

Assignment 2

- Due tonight 11:59pm
- Any questions?
- Please check out the **#assignment2** slack channel for updates and answers to frequently asked questions

Assignment 3: Magic Nicla Wand

Due: November 6 at 11:59 pm

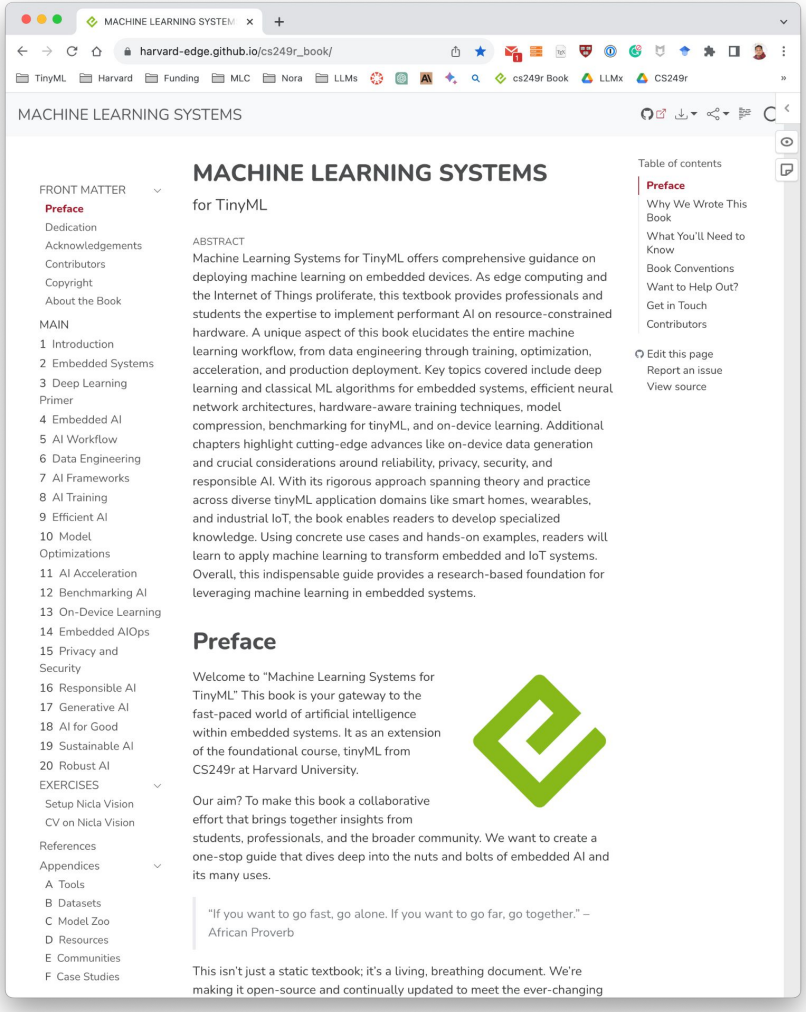
Objective:

- Explore Tensorflow ecosystem (Tensorflow -> Tensorflow Lite -> Tensorflow Lite Micro)
- Model Optimization (quantization/pruning) using IMU data from Arduino Nicla Vision

Extra Credit: Deployment of model on Nicla

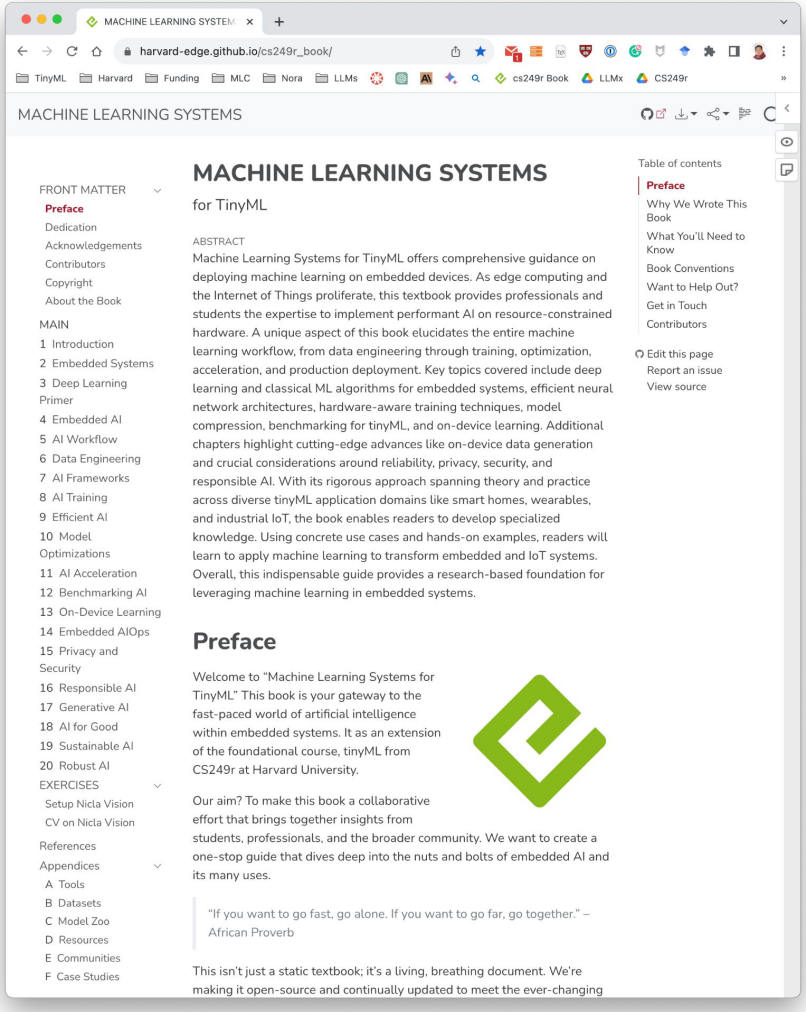
Scribing (again!)

- Week of topic
  - Meet with Matthew & VJ
  - Create a google doc
  - Share with staff
  - Iterate on rough outline with VJ/Matt
  - After 👍 from staff, put changes in a **single forked repo**
  - Create **one PR** with the entire chapter
  - Classmates will peer review using the PR



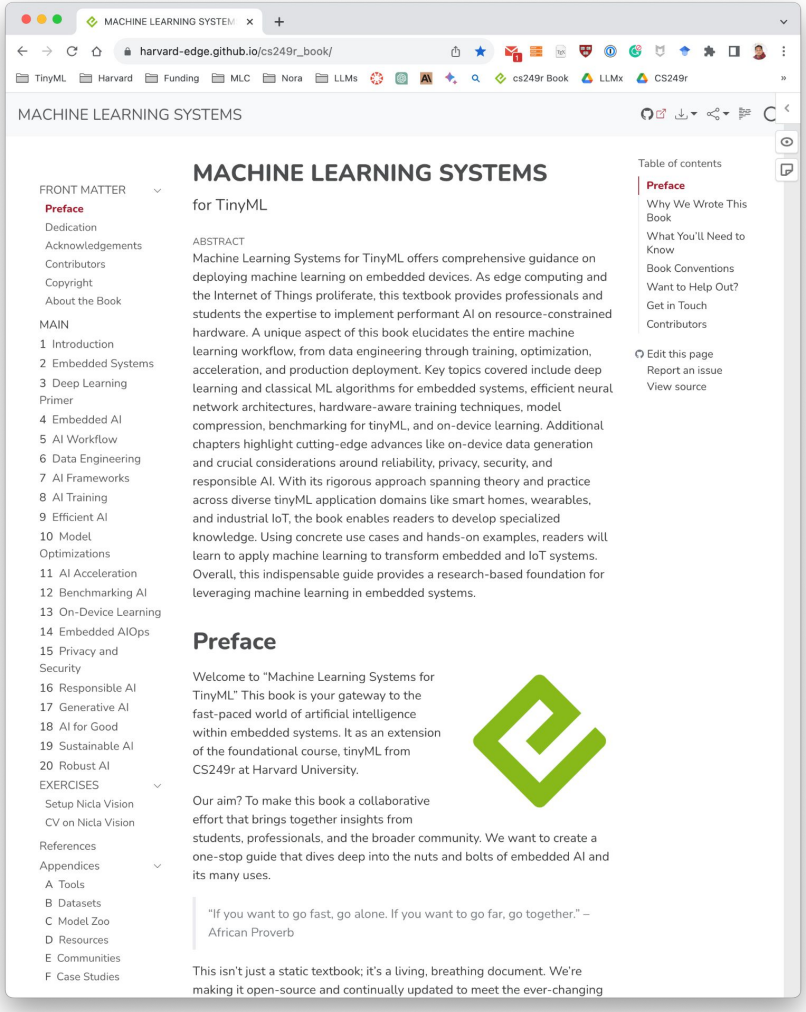
Scribing

- Grading (20% of your grade)
  - Part 1:
    - i. Paper review 10%
      - Content creation - 5%
      - Content curation - 5%
  - Part 2
    - i. Paper presentation 10%
      - References - 3%
      - Figures - 2%
      - Clarity - 5%



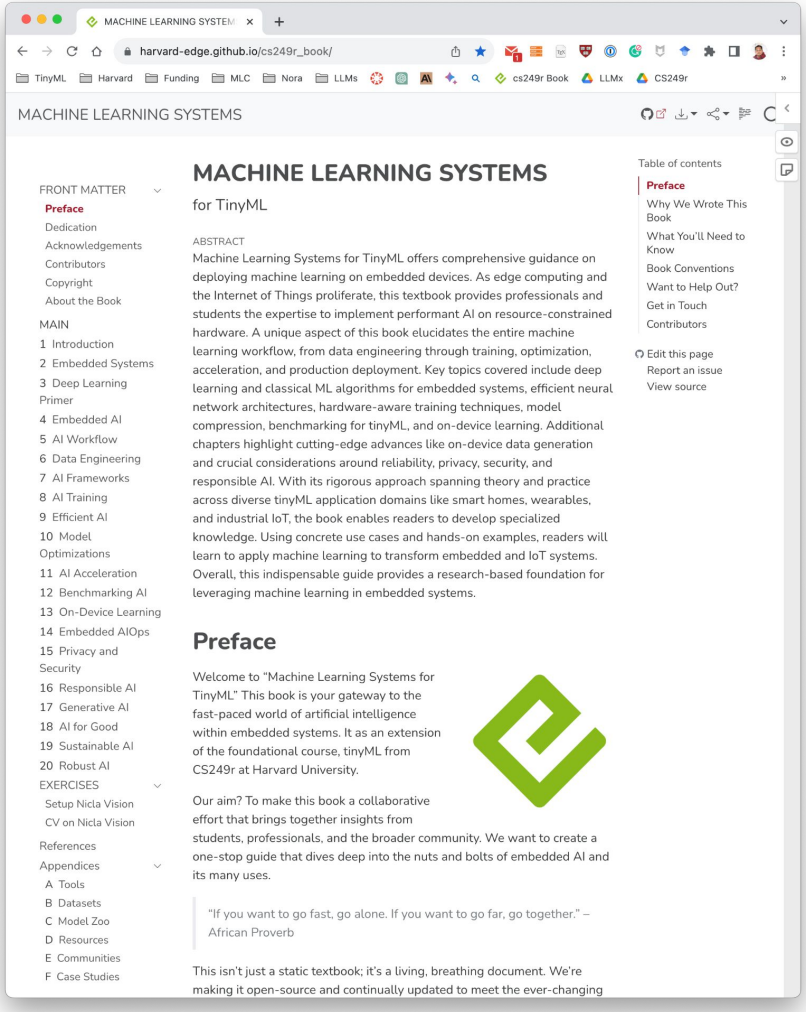
Scribing

- AI Frameworks
  - Available as a PR
  - Will be merged soon (EOD or tomorrow)



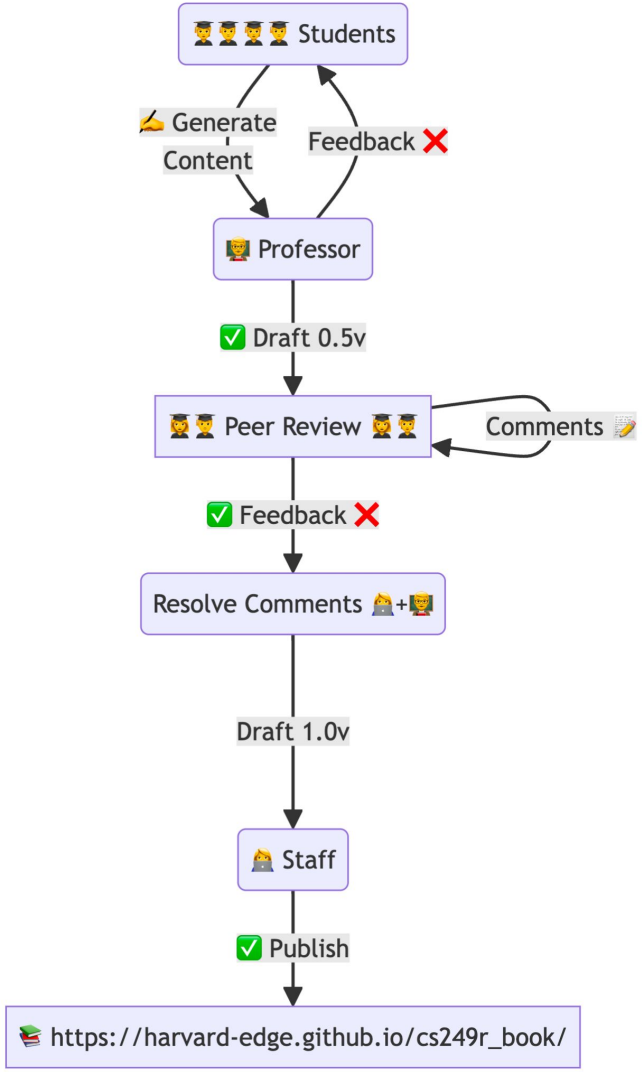
Scribing (again!)

- This week (Oct 23)
  - Review Model optimizations chapter
- Next week
  - Review On-device learning chapter
  - Review of Benchmarking AI



Assignment Schedule Updates

- Assignment 2
  - Due: October 23rd (Monday)
- Mid-Project Review
  - Due: October 30th (Monday)
- Assignment 3
  - Due: November 6th (Monday)
- Assignment 4 Part 1
  - Due: November 20th (Monday)
- Assignment 4 Part 2
  - Due: November 27th (Monday)
- Project Presentations
  - Due: December 4th (Monday)
- Final Report
  - Due: December 11th (Monday)



Projects

Mid-Project Presentation (**5 slides/3 mins**) – Oct 30th

Oct 30	MLOps	Daniel Situnayake, Head of Machine Learning at Edge Impulse	Required • Edge Impulse (paper) • Hidden Technical Debt in Machine Learning (paper) Optional • Data Cascades in High-Stakes AI (paper) • Pytorch RPC (paper)	None	Mid-Project Review Due

A	B	C
	Paper Reading Table Leads	
	MLOps	Hidden Technical Debt
Table 1		Vijay Edupuganti
Table 2		Annie Landefeld
Table 3		Andrew Bass
Table 4		Aghyad Deeb
Table 5		Jared Ni

Project Update Slide Template

Put your slides in here

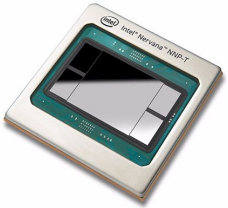
- Add your title slide
- Answer the questions with the template slides
- Stay on time

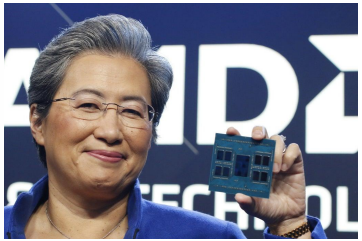
Lecture

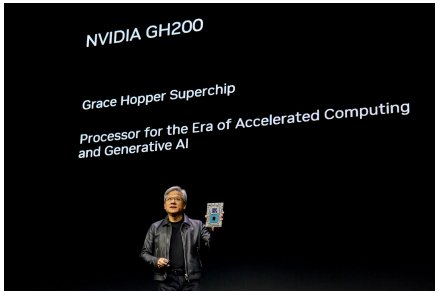
Course Topics

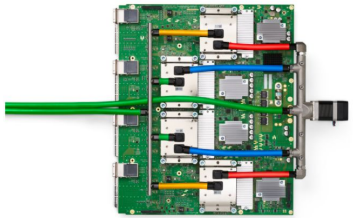
1. Overview and Introduction to Embedded Machine Learning
2. Data Engineering
3. Embedded Machine Learning Frameworks
4. Efficient Model Representation and Compression
5. ~~Performance Metrics and Benchmarking of ML Systems~~
6. Learning on the Edge
7. Hardware Acceleration for Edge ML: GPUs, TPUs and FPGAs
8. Embedded MLOps
9. Secure and Privacy-Preserving On-Device ML
10. Responsible AI
11. Sustainability at the Edge
12. Generative AI at the Edge

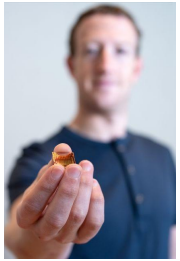
Benchmarking ML Systems

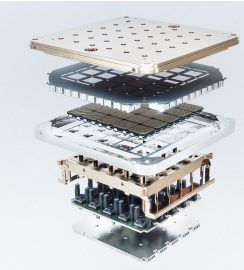

intel
Intel® Habana® NNP-T




NVIDIA GH200
Grace Hopper Superchip
Processor for the Era of Accelerated Computing and Generative AI







LIGHTMATTER
COILTRONICS
VP3-0760
4YPN18 K
XILINX
ZYNQ
6-2301984-1
0516-A\129
10.10.10.161

AI Chip Landscape

S.T.

Tech Giants/System

Google
Microsoft
facebook
aws
IBM
Alibaba Group 阿里巴巴集团
HUAWEI
Baidu 百度
Hewlett Pack Enterprise
FUJITSU
DELL
Western Digital
NOKIA
LG

IC Vender/Fabless

intel
SAMSUNG
NVIDIA
QUALCOMM
AMD
XILINX
MEDIATEK
UNISOC
Rockchip
NationalChip

Automated Driving

NXP
Texas Instruments
RENESAS
TOSHIBA
ST

Startup in China

Cambricon
地平线 Horizon Robotics
BITMAIN
intelliFusion
Think Force
NextVPU
Canaan
Enflame
清微智能 TsingMicro
亿智科技
Smart Voice
Chiplatelli
Rokid
AISPEECH 思必驰
云知声 Unisound

Startup Worldwide

Cerebras
WAVE COMPUTING
Graphcore
SambaNova
habana
HAILO
thinci
KALRAY
groq
Tachyum
Esperanto TECHNOLOGIES
PEZY Computing
Eta Compute
GREENWAVES TECHNOLOGIES

FPGA

Achronix
flexlogix

Processing in Memory

MYTHIC
SYNTIANT
gyrfalcon technology

Optical Computing

LIGHTELLIGENCE
LIGHTMATTER
Optalysys

neuromorphic

aiCTX
brainchip
Preferred Networks
Kneron
NOVUMIND

More on https://basicmi.github.io/AI-Chip/

IP/Design Sevice

arm
SYNOPSYS
Imagination
CEVA
cadence
SiFive
ARTERIS IP
Moortec

Design service with In-house IP

VeriSilicon
BROADCOM
GUC
alchip
FARADAY
eSilicon

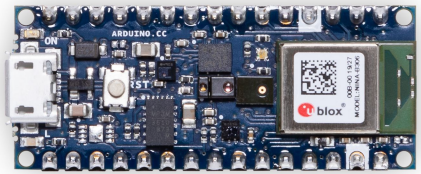
Compilers

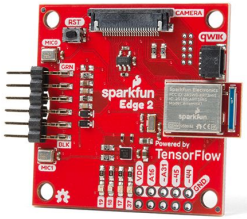
TensorFlow XLA
GLOW
tvm
NVIDIA TensorRT
plaidML
nGraph
Tiramisu Compiler
ONNC

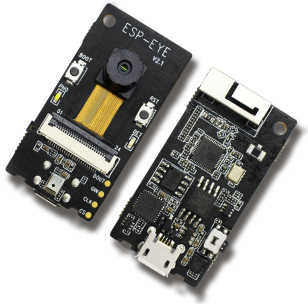
Benchmarks

MLPerf
AI - Benchmark
AI Matrix
AIIA
DAWNBench


himax

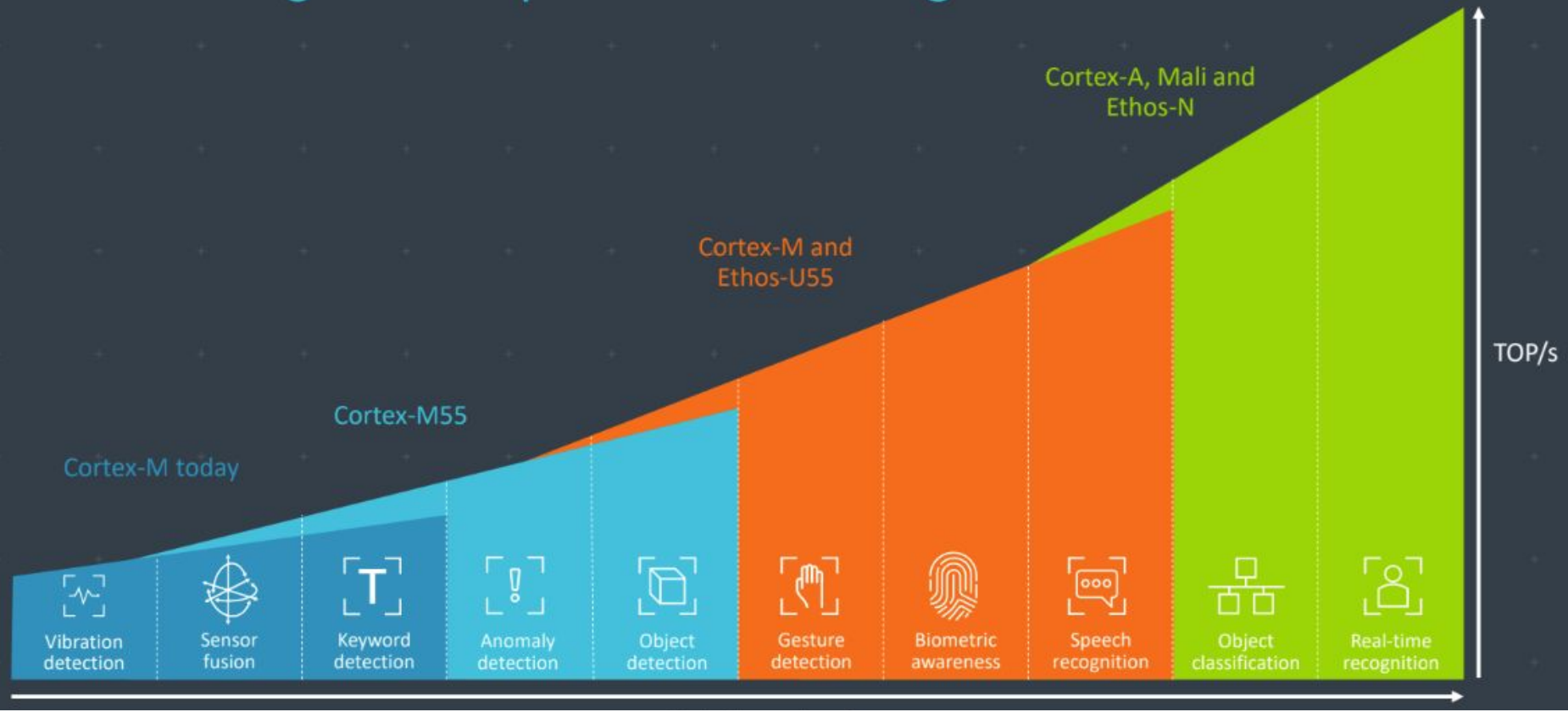

ARDUINO.CC
ON
RST
blox


RST
CAMERA
TWIK
sparkfun
Edge 2
Powered by
TensorFlow

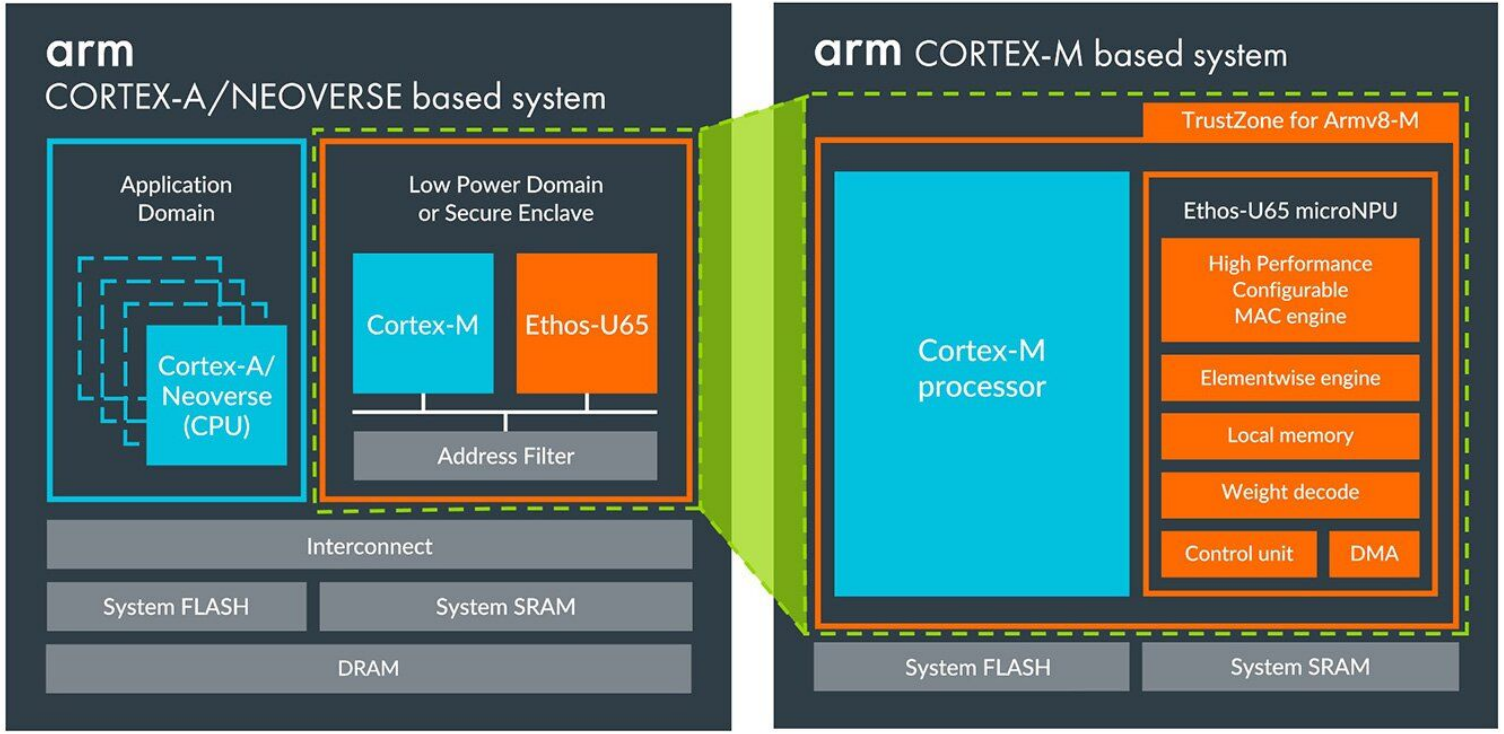

ESP-EYE
V2.1

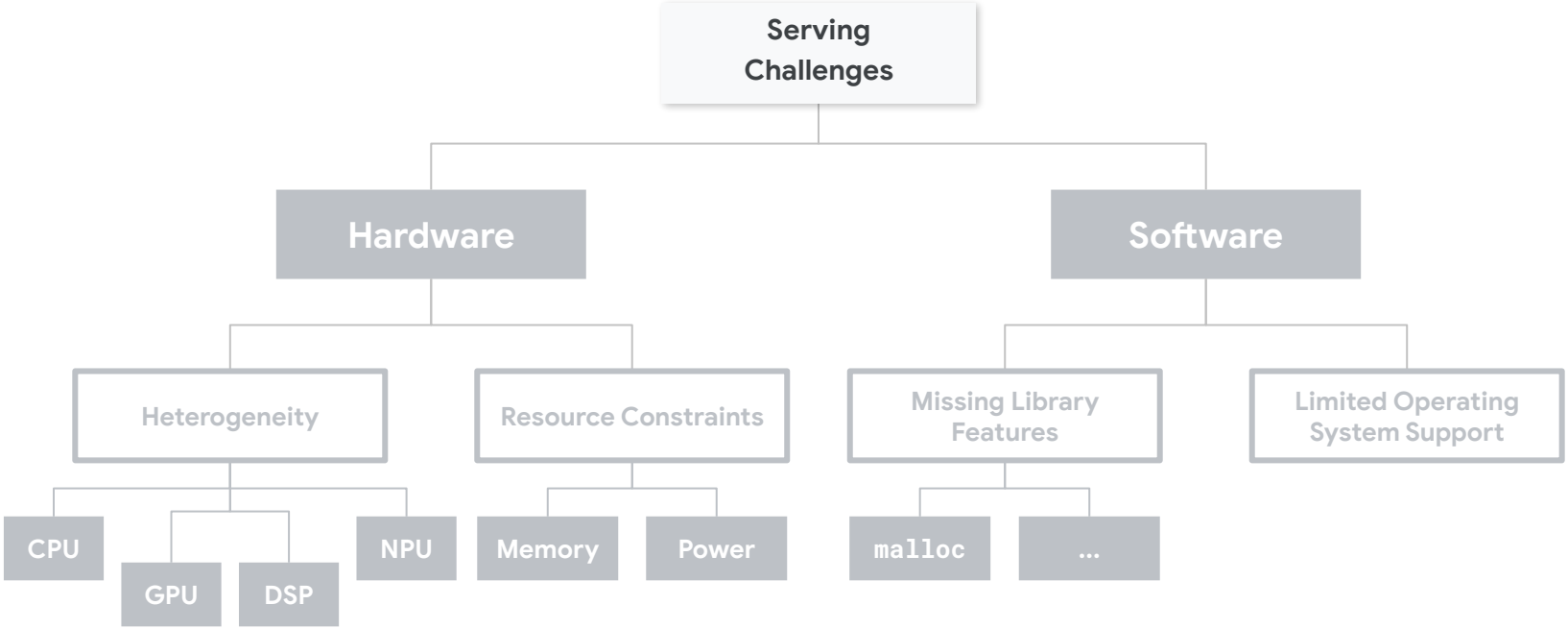
	Board	MCU / ASIC	Clock	Memory	Sensors	Radio
	Himax WE-I Plus EVB	HX6537-A 32-bit EM9D DSP	400 MHz	2MB flash 2MB RAM	Accelerometer, Mic, Camera	None
	Arduino Nano 33 BLE Sense	32-bit nRF52840	64 MHz	1MB flash 256kB RAM	Mic, IMU, Temp, Humidity, Gesture, Pressure, Proximity, Brightness, Color	BLE
	SparkFun Edge 2	32-bit ArtemisV1	48 MHz	1MB flash 384kB RAM	Accelerometer, Mic, Camera	BLE
	Espressif EYE	32-bit ESP32-D0WD	240 MHz	4MB flash 520kB RAM	Mic, Camera	WiFi, BLE

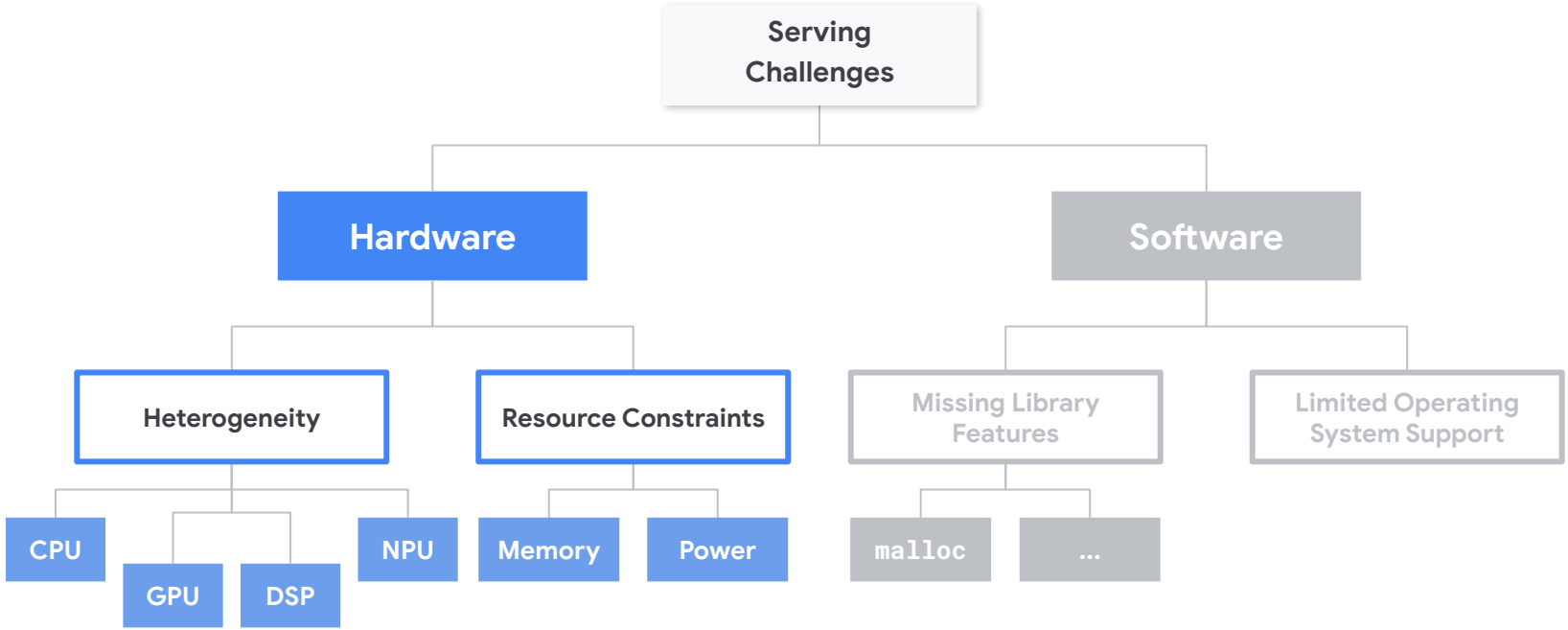
Broadest Range of ML-optimized Processing Solutions

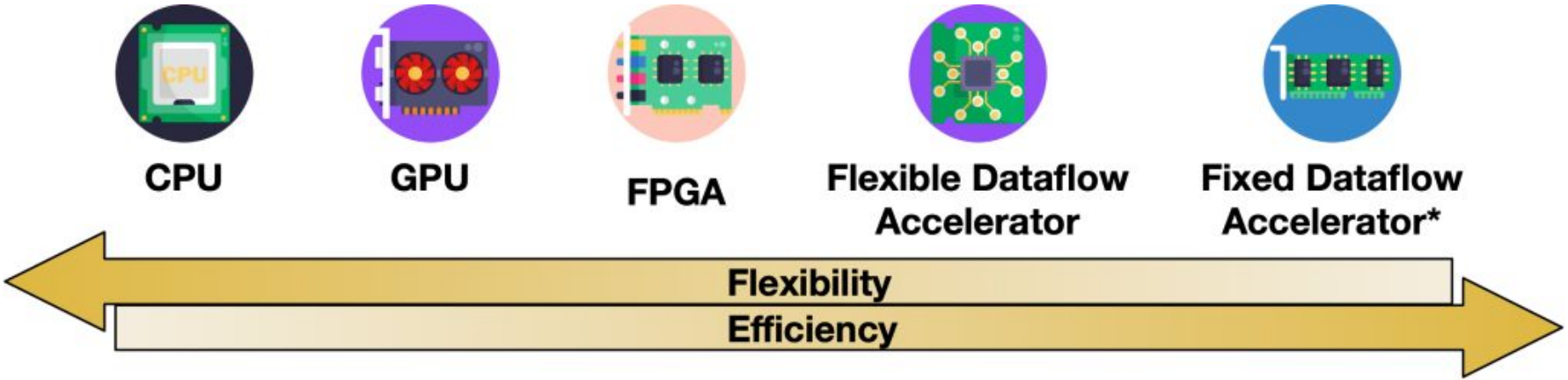


Different Systems Applicable to Ethos-U




Serving Challenges
Hardware
Software
Heterogeneity
Resource Constraints
Missing Library Features
Limited Operating System Support
CPU
GPU
DSP
NPU
Memory
Power
malloc
...


Serving Challenges
Hardware
Software
Heterogeneity
Resource Constraints
Missing Library Features
Limited Operating System Support
CPU
GPU
DSP
NPU
Memory
Power
malloc
...



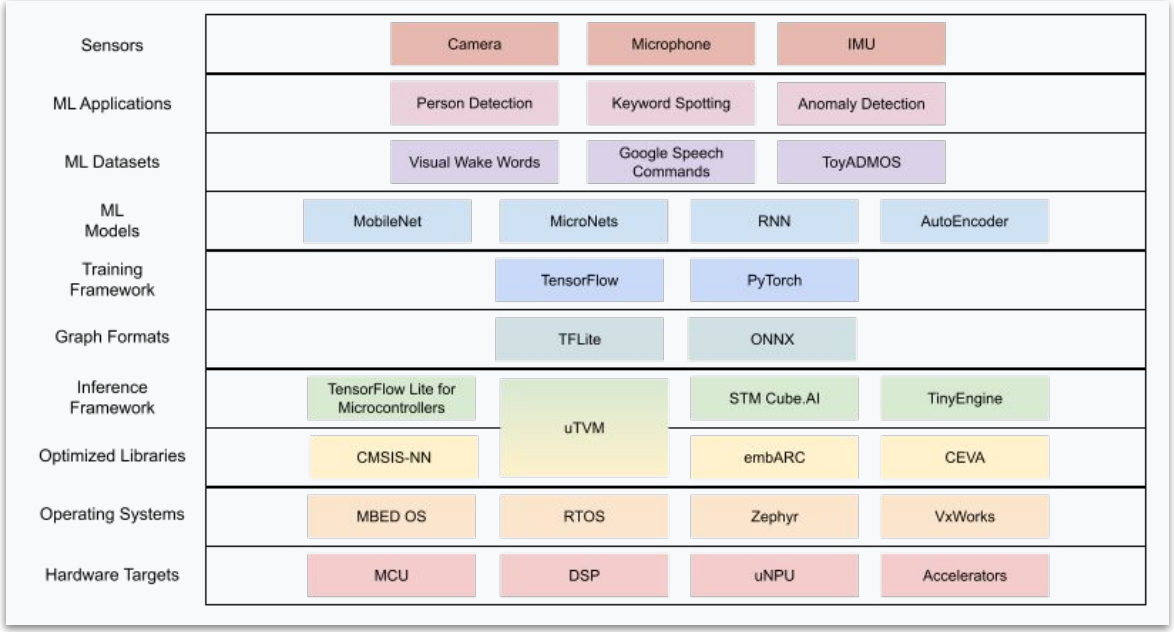
* When the dataflow is optimal for a workload; Efficiency can drop under workload change


ML
Hardware X


ML
Hardware Y

TinyML System Stack is Complicated

- Machine learning system stack is **complicated**
- Many **different** models, datasets, models, frameworks, formats, compilers, libraries, operating systems, targets
- The **cross-product** makes it challenging to decipher system performance



Apples-to-apples comparison

What task?
What model?
What dataset?
What batch size?
What quantization?
What software libraries?
...

bench·mark

/ˈben(t)SHmärk/

See definitions in:

All | Technology | Surveying

noun

1. a standard or point of reference against which things may be compared or assessed.
   "a benchmark case"

   Similar: standard | point of reference | basis | gauge | criterion | specification

2. a surveyor's mark cut in a wall, pillar, or building and used as a reference point in measuring altitudes.

verb

evaluate or check (something) by comparison with a standard.
"we are **benchmarking** our performance **against** external criteria"

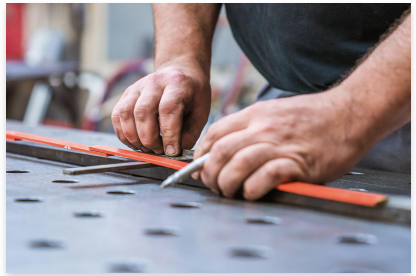
Definitions from Oxford Languages

Feedback

Benchmarking

Use to

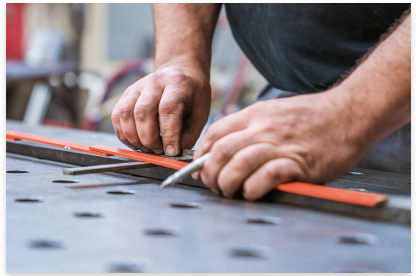
- **Compare** solutions



Benchmarking

Use to

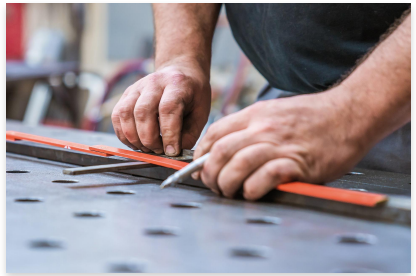
- **Compare** solutions
- **Inform** selection



Benchmarking

Use to

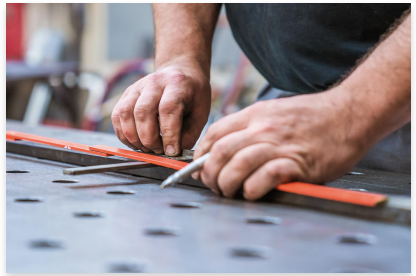
- **Compare** solutions
- **Inform** selection
- **Measure** and track progress



Benchmarking

Use to

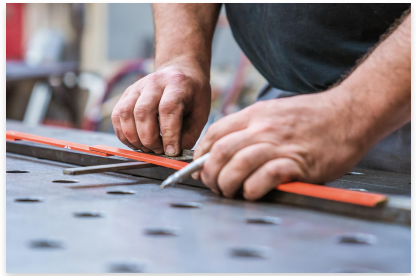
- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar**, advance** the field



Benchmarking

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field



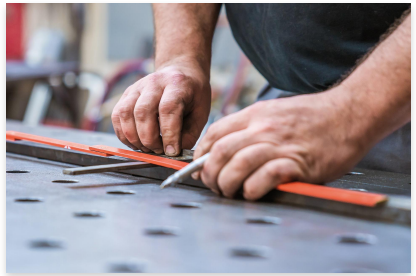
Requires

- **Methodology** that is both fair and rigorous

Benchmarking

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field



Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus

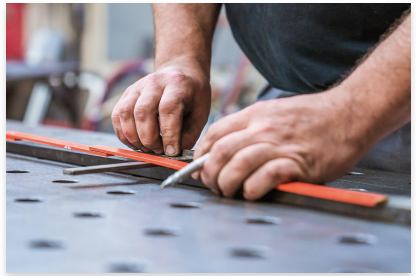
Benchmarking

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field

Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus



Provides

- **Standardization** of use cases and workloads

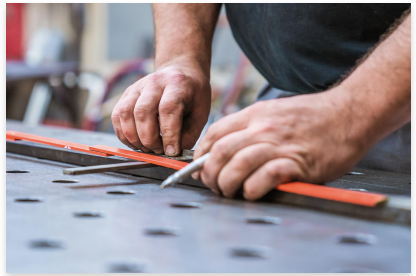
Benchmarking

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field

Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus



Provides

- **Standardization** of use cases and workloads
- **Comparability** across heterogeneous HW/SW systems

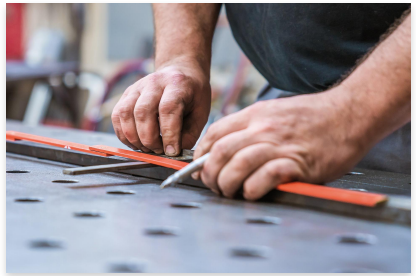
Benchmarking

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field

Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus



Provides

- **Standardization** of use cases and workloads
- **Comparability** across heterogeneous HW/SW systems
- **Complex characterization** of system compromises

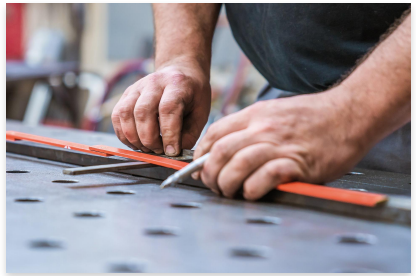
Benchmarking

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field

Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus



Provides

- **Standardization** of use cases and workloads
- **Comparability** across heterogeneous HW/SW systems
- **Complex characterization** of system compromises
- **Verifiable and Reproducible** results

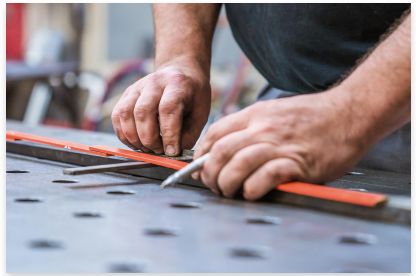
Benchmarking

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar**, advance** the field

Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus



Provides

- **Standardization** of use cases and workloads
- **Comparability** across heterogeneous HW/SW systems
- **Complex characterization** of system compromises
- **Verifiable and Reproducible** results

MLPerf

Goals

Enforce performance result replicability to ensure reliable results

MLPerf

Goals

MLPerf

Enforce performance result replicability to ensure reliable results

Use **representative workloads**, reflecting production use-cases

Goals

MLPerf

Enforce performance result replicability to ensure reliable results

Use **representative workloads**, reflecting production use-cases

Encourage innovation to improve the state-of-the-art of ML

Goals

MLPerf

Enforce performance result replicability to ensure reliable results

Use **representative workloads**, reflecting production use-cases

Encourage innovation to improve the state-of-the-art of ML

Accelerate progress in ML via **fair and useful measurement**

Goals

MLPerf

Enforce performance result replicability to ensure reliable results

Use **representative workloads**, reflecting production use-cases

Encourage innovation to improve the state-of-the-art of ML

Accelerate progress in ML via **fair and useful measurement**

Serve both the **commercial and research communities**

Goals

MLPerf

Enforce performance result replicability to ensure reliable results

Use **representative workloads**, reflecting production use-cases

Encourage innovation to improve the state-of-the-art of ML

Accelerate progress in ML via **fair and useful measurement**

Serve both the **commercial and research communities**

Keep **benchmarking affordable** so that all can participate

Wide Array of ML Tasks

Task Category	Use Case
Audio	Audio Wake Words Context Recognition Control Words Keyword Detection
Image	Visual Wake Words Object Detection Gesture Recognition Object Counting Text Recognition
Physiological / Behavioral Metrics	Segmentation Anomaly Detection Forecasting Activity Detection
Industry Telemetry	Sensing Predictive Maintenance Motor Control

Wide Array of ML Tasks

Task Category	Use Case	Model Type
Audio	Audio Wake Words Context Recognition Control Words Keyword Detection	DNN CNN RNN LSTM
Image	Visual Wake Words Object Detection Gesture Recognition Object Counting Text Recognition	DNN CNN SVM Decision Tree KNN Linear
Physiological / Behavioral Metrics	Segmentation Anomaly Detection Forecasting Activity Detection	DNN Decision Tree SVM Linear
Industry Telemetry	Sensing Predictive Maintenance Motor Control	DNN Decision Tree SVM Linear Naive Bayes

Wide Array of ML Tasks

Task Category	Use Case	Model Type	Datasets
Audio	Audio Wake Words Context Recognition Control Words Keyword Detection	DNN CNN RNN LSTM	Speech Commands Audioset ExtraSensory Freesound DCASE
Image	Visual Wake Words Object Detection Gesture Recognition Object Counting Text Recognition	DNN CNN SVM Decision Tree KNN Linear	Visual Wake Words CIFAR10 MNIST ImageNet DVS128 Gesture
Physiological / Behavioral Metrics	Segmentation Anomaly Detection Forecasting Activity Detection	DNN Decision Tree SVM Linear	Physionet HAR DSA Opportunity
Industry Telemetry	Sensing Predictive Maintenance Motor Control	DNN Decision Tree SVM Linear Naive Bayes	UCI Air Quality UCI Gas UCI EMG NASA's PCoE

A Principled Approach to Subsetting

Big Questions	**Inference**
1. Benchmark definition	What is the definition of a benchmark task?

A Principled Approach to Subsetting

Big Questions	**Inference**
1. Benchmark definition	What is the definition of a benchmark task?
2. Benchmark selection	Which benchmark task to select?

A Principled Approach to Subsetting

Big Questions	**Inference**
1. Benchmark definition	What is the definition of a benchmark task?
2. Benchmark selection	Which benchmark task to select?
3. Metric definition	What is the measure of “performance” in ML systems?

A Principled Approach to Subsetting

Big Questions	**Inference**
1. Benchmark definition	What is the definition of a benchmark task?
2. Benchmark selection	Which benchmark task to select?
3. Metric definition	What is the measure of “performance” in ML systems?
4. Implementation equivalence	How do submitters run on different hardware/software systems?

A Principled Approach to Subsetting

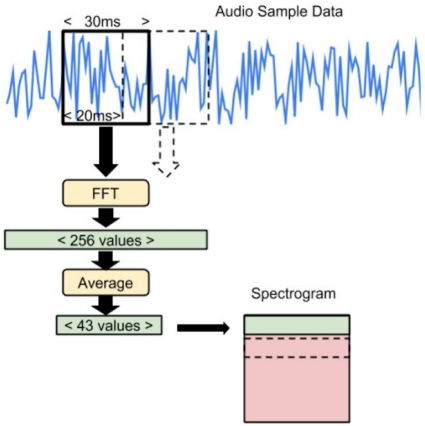
Big Questions	Inference
1. Benchmark definition	What is the definition of a benchmark task?
2. Benchmark selection	Which benchmark task to select?
3. Metric definition	What is the measure of “performance” in ML systems?
4. Implementation equivalence	How do submitters run on different hardware/software systems?
5. Issues with optimizations	Quantization, calibration, and/or retraining?

A Principled Approach to Subsetting

Big Questions	Inference
1. Benchmark definition	What is the definition of a benchmark task?
2. Benchmark selection	Which benchmark task to select?
3. Metric definition	What is the measure of “performance” in ML systems?
4. Implementation equivalence	How do submitters run on different hardware/software systems?
5. Issues with optimizations	Quantization, calibration, and/or retraining?
6. Results	Do we normalize and/or summarize results?

MLPerf “Tiny” Tasks

Keyword Spotting



Warden, Pete. "Speech commands: A dataset for limited-vocabulary speech recognition." *arXiv preprint arXiv:1804.03209* (2018).

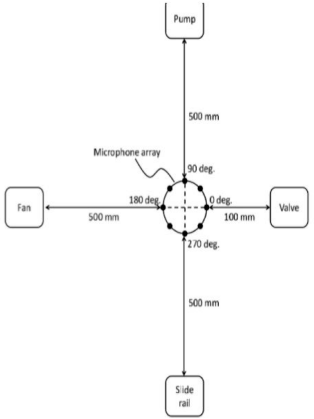
Visual Wake Words

(a) ‘Person’

(b) ‘Not-person’

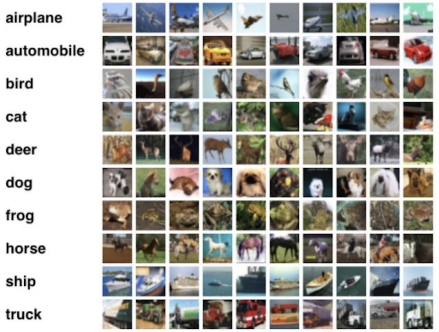
Chowdhery, Aakanksha, et al. "Visual wake words dataset." *arXiv preprint arXiv:1906.05721* (2019).

Anomaly Detection



Purohit, Harsh, et al. "MIMII dataset: Sound dataset for malfunctioning industrial machine investigation and inspection." *arXiv preprint arXiv:1909.09347* (2019).

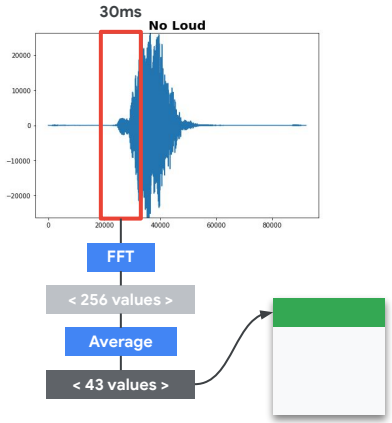
Tiny Image Classification



Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009): 7.

MLPerf “Tiny” Tasks

Keyword Spotting



Warden, Pete. "Speech commands: A dataset for limited-vocabulary speech recognition." *arXiv preprint arXiv:1804.03209* (2018).

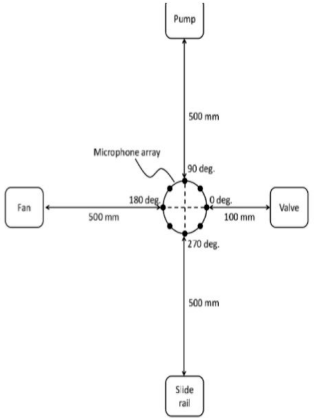
Visual Wake Words

(a) ‘Person’

(b) ‘Not-person’

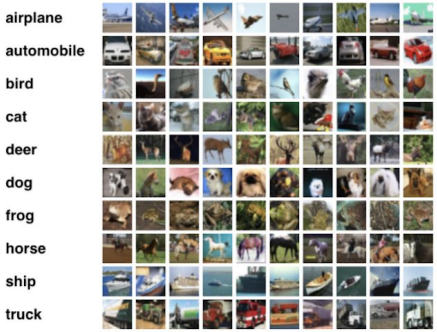
Chowdhery, Aakanksha, et al. "Visual wake words dataset." *arXiv preprint arXiv:1906.05721* (2019).

Anomaly Detection

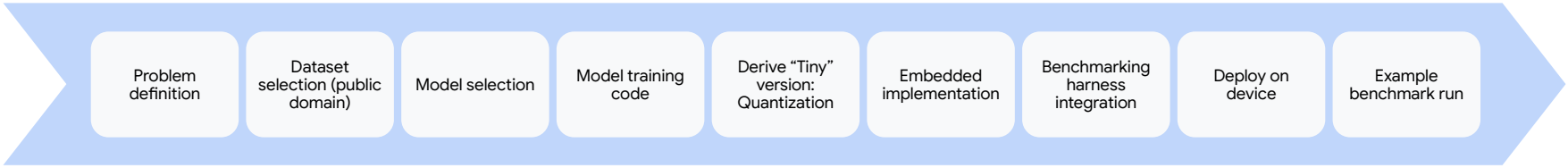


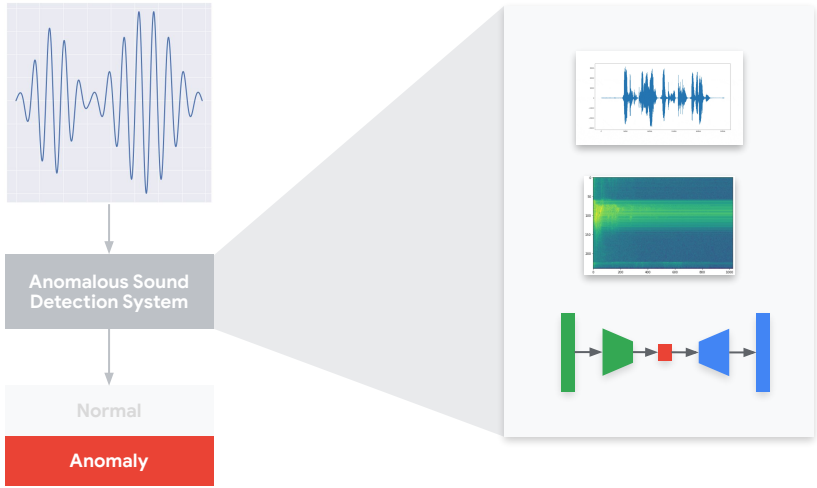
Purohit, Harsh, et al. "MIMII dataset: Sound dataset for malfunctioning industrial machine investigation and inspection." *arXiv preprint arXiv:1909.09347* (2019).

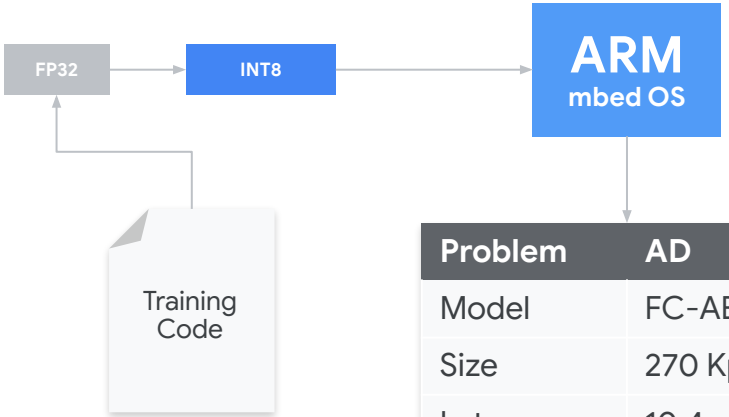
Tiny Image Classification



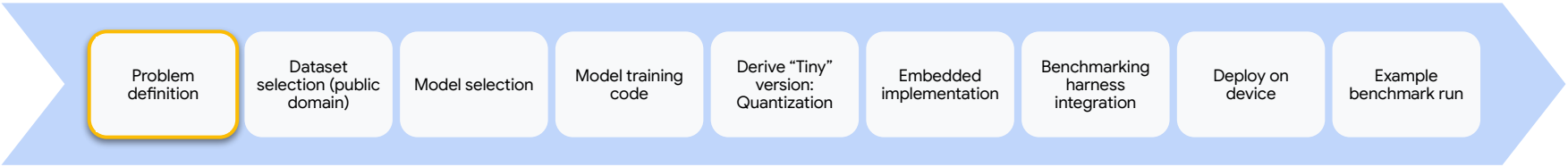
Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009): 7.

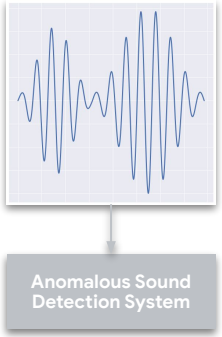


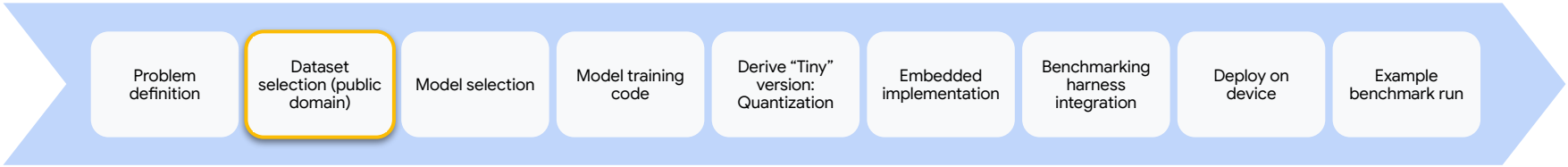


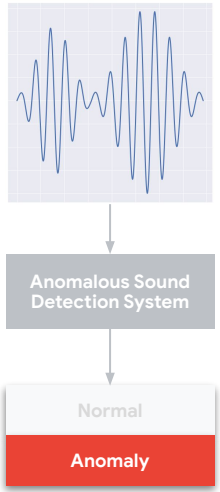


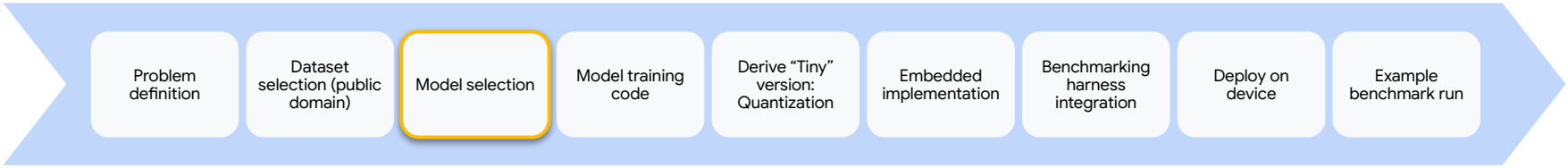
Problem	AD
Model	FC-AE
Size	270 Kpar
Latency	10.4 ms/inf.
Accuracy	.86 AUC
Energy	516 µJ/inf.

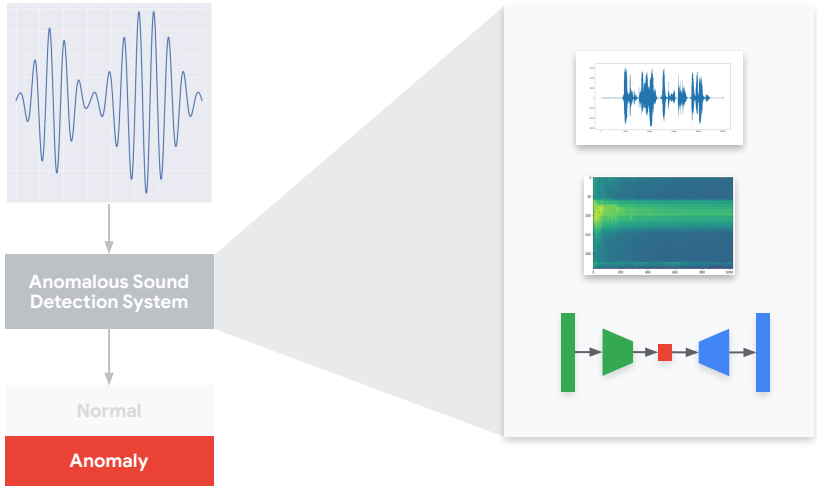

Problem definition
Dataset selection (public domain)
Model selection
Model training code
Derive “Tiny” version: Quantization
Embedded implementation
Benchmarking harness integration
Deploy on device
Example benchmark run

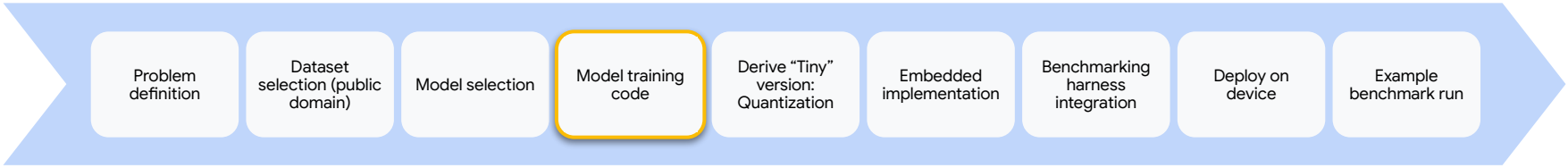

Anomalous Sound Detection System

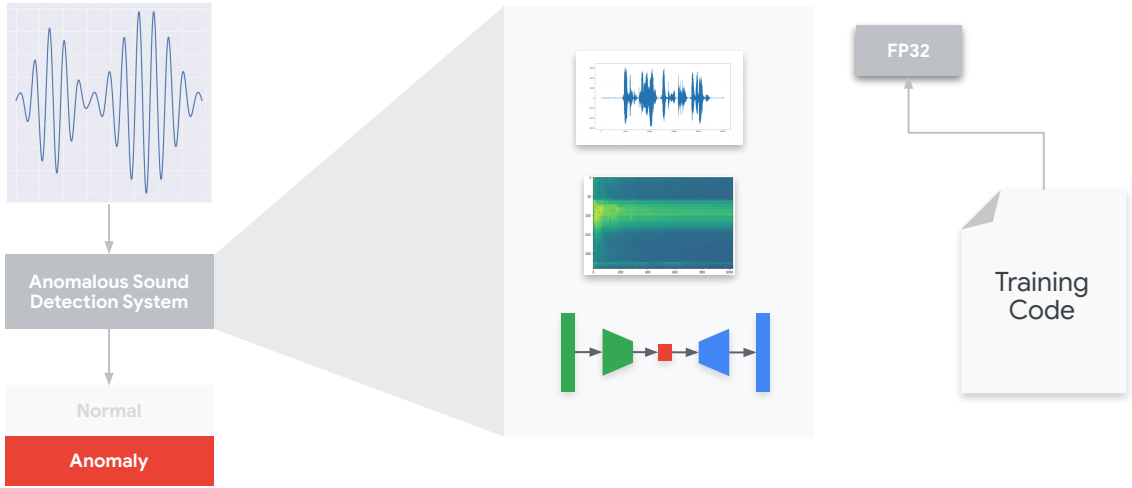

Problem definition
Dataset selection (public domain)
Model selection
Model training code
Derive “Tiny” version: Quantization
Embedded implementation
Benchmarking harness integration
Deploy on device
Example benchmark run

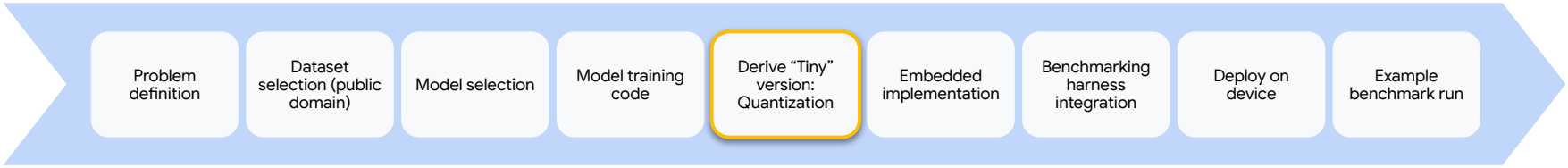

Anomalous Sound Detection System
Normal
Anomaly

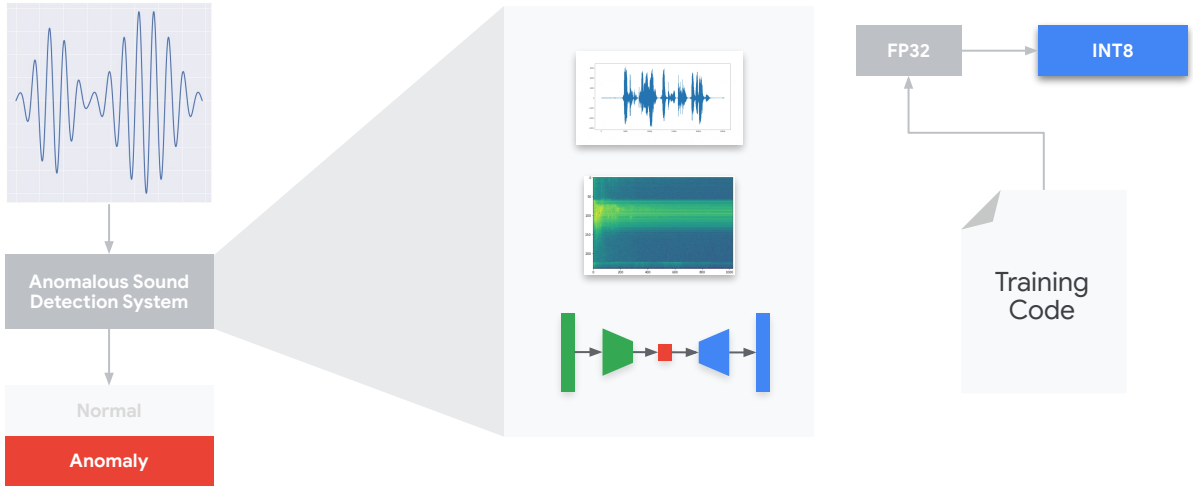

Problem definition
Dataset selection (public domain)
Model selection
Model training code
Derive “Tiny” version: Quantization
Embedded implementation
Benchmarking harness integration
Deploy on device
Example benchmark run

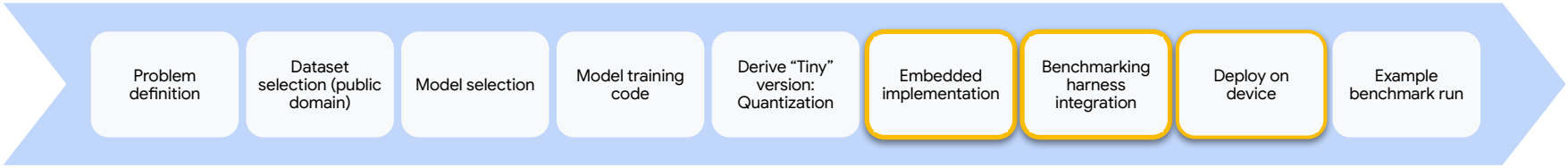

Anomalous Sound Detection System
Normal
Anomaly


Problem definition
Dataset selection (public domain)
Model selection
Model training code
Derive “Tiny” version: Quantization
Embedded implementation
Benchmarking harness integration
Deploy on device
Example benchmark run

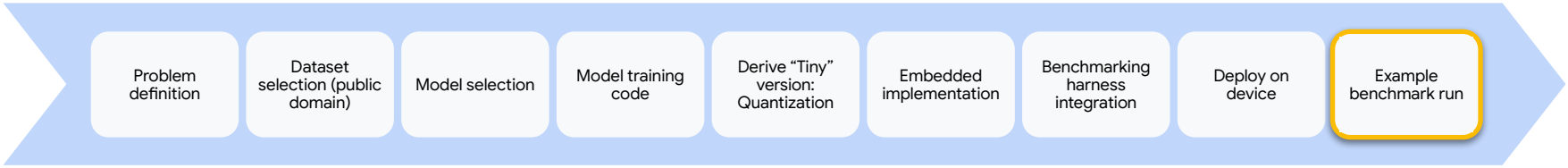

Anomalous Sound Detection System
Normal
Anomaly
FP32
Training Code

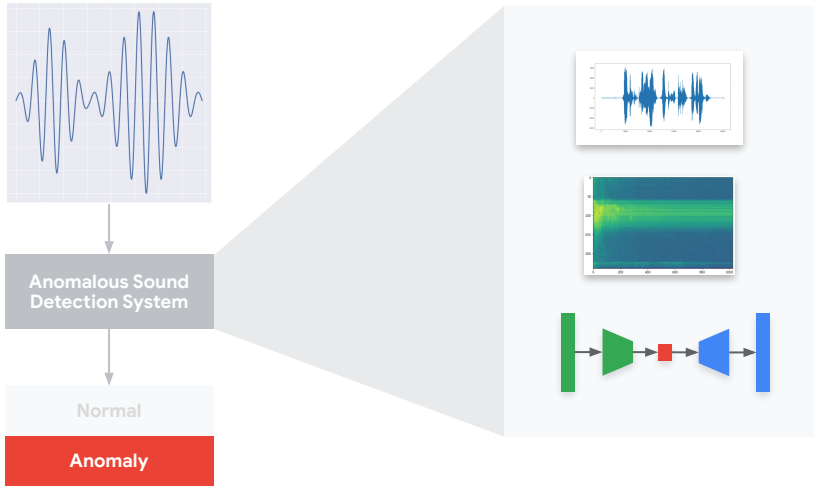

Problem definition
Dataset selection (public domain)
Model selection
Model training code
Derive “Tiny” version: Quantization
Embedded implementation
Benchmarking harness integration
Deploy on device
Example benchmark run

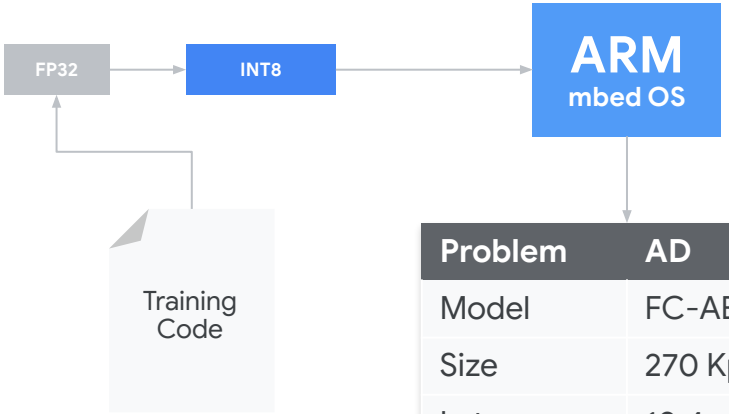

Anomalous Sound Detection System
Normal
Anomaly
FP32
INT8
Training Code


Problem definition
Dataset selection (public domain)
Model selection
Model training code
Derive "Tiny" version: Quantization
Embedded implementation
Benchmarking harness integration
Deploy on device
Example benchmark run


Anomalous Sound Detection System
Normal
Anomaly
FP32
INT8
ARM
mbed OS
Training Code







Problem	AD
Model	FC-AE
Size	270 Kpar
Latency	10.4 ms/inf.
Accuracy	.86 AUC
Energy	516 µJ/inf.

Metrics

Latency

Small fast dataset

Loop of inferences

No data-dependent execution

a.

```
Runtime requirements have been met.
Performance results for window 10:
  # Inferences :         1000
  Runtime     :       10.524 sec.
  Throughput  :       95.020 inf./sec.
Runtime requirements have been met.
----------------------------------------
Median throughput is 95.019 inf./sec.
----------------------------------------
```

Metrics

Latency

Small fast dataset

Loop of inferences

No data-dependent execution

a.

```
Runtime requirements have been met.
Performance results for window 10:
  # Inferences :          1000
  Runtime     :        10.524 sec.
  Throughput  :        95.020 inf./sec.
Runtime requirements have been met.
----------------------------------------
Median throughput is 95.019 inf./sec.
----------------------------------------
```

Accuracy

Evaluate on larger dataset

Top-1 accuracy & AUC

CLOSED: meet threshold
v.
OPEN: part of the metrics

Metrics

Latency

Small fast dataset

Loop of inferences

No data-dependent execution

a.

```
Runtime requirements have been met.
Performance results for window 10:
  # Inferences :          1000
  Runtime      :        10.524 sec.
  Throughput   :        95.020 inf./sec.
Runtime requirements have been met.
----------------------------------------
Median throughput is 95.019 inf./sec.
----------------------------------------
```

Accuracy

Evaluate on larger dataset

Top-1 accuracy & AUC

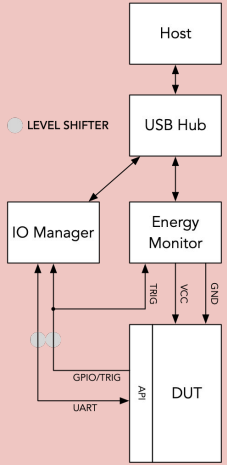
CLOSED: meet threshold
v.
OPEN: part of the metrics

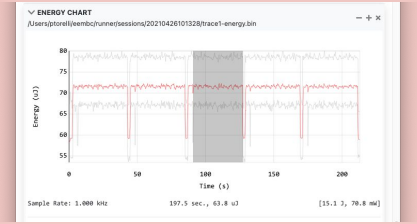
Energy

No "cherry-picking"

Power Monitor setup

Median result





V1.0 Results

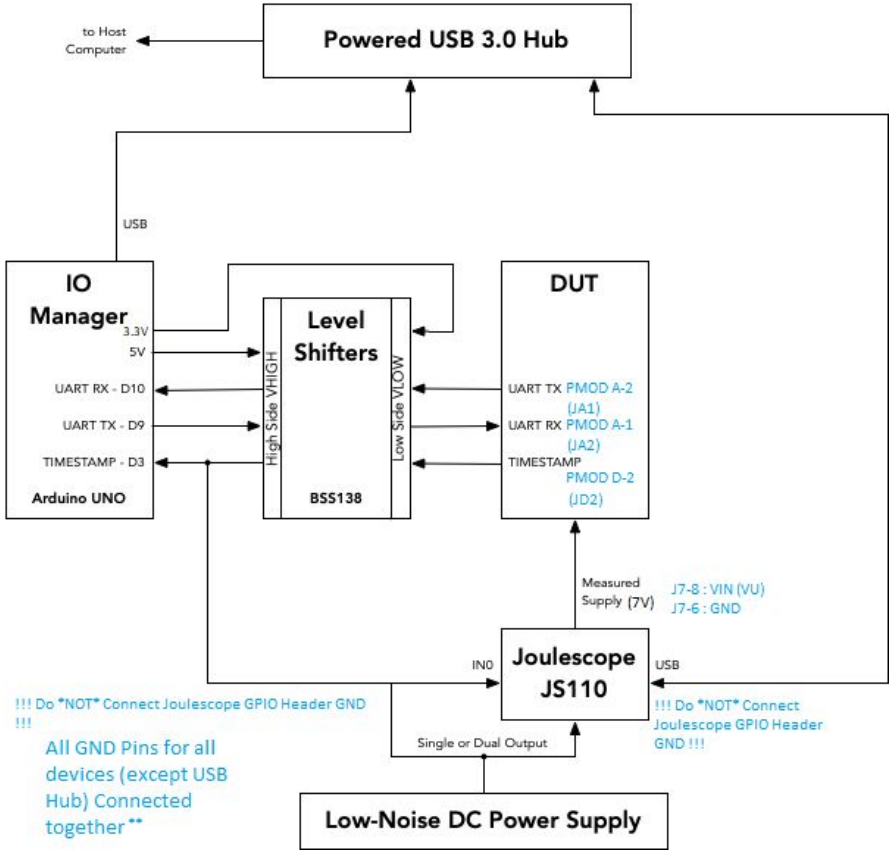
mlcommons.org/en/inference-tiny-10

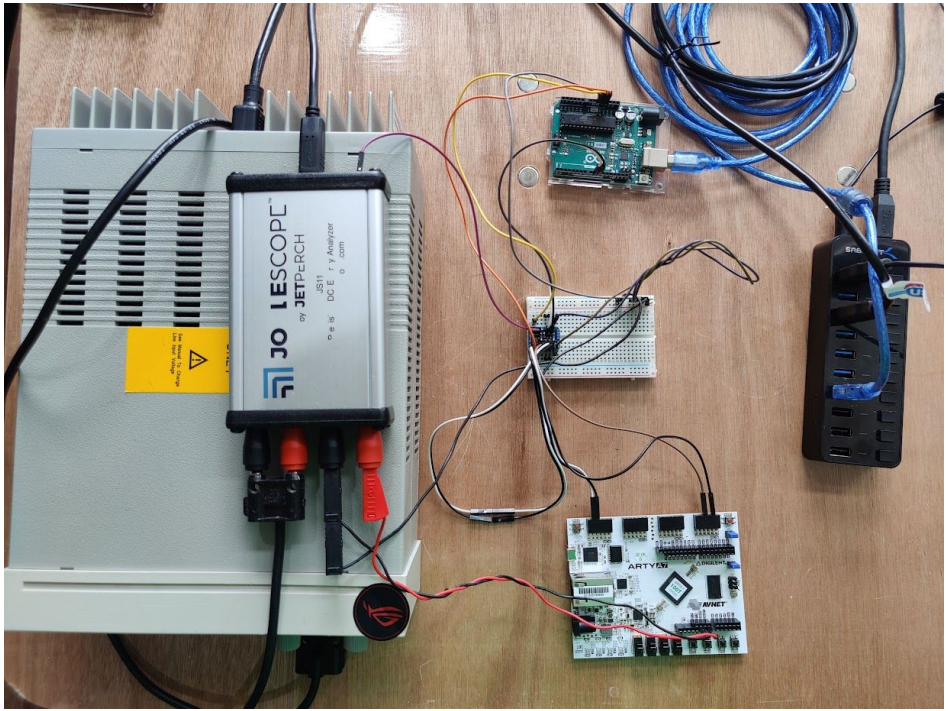
Submitter	Board Name	SoC Name	Processor(s) & Number	Accelerator(s) & Number
Greenwaves Technologies	GAP9 EVK	GAP9	RISC-V Core (1+9)	NE16 (1)
Greenwaves Technologies	GAP9 EVK	GAP9	RISC-V Core (1+9)	NE16 (1)
OctoML	NRF5340DK	nRF5340	Arm® Cortex®-M33	
OctoML	NUCLEO-L4R5ZI	STM32L4R5ZIT6U	Arm® Cortex®-M4	
OctoML	NUCLEO-L4R5ZI	STM32L4R5ZIT6U	Arm® Cortex®-M4	
Plumerai	B_U585I_IOT02A	STM32U585	Arm® Cortex®-M33	
Plumerai	CY8CPROTO-062-4343w	PSoC 62 MCU	Arm® Cortex®-M4	
Plumerai	DISCO-F746NG	STM32F746	Arm® Cortex®-M7	
Plumerai	NUCLEO-L4R5ZI	STM32L4R5ZIT6U	Arm® Cortex®-M4	
Silicon Labs	xG24-DK2601B	EFR32MG24	Arm® Cortex®-M33	Silicon Labs MVP(1) (78 MHz, 1.8V)
STMicroelectronics	NUCLEO-H7A3ZI-Q	STM32H7A3ZIT6Q	Arm® Cortex®-M7	
STMicroelectronics	NUCLEO-L4R5ZI	STM32L4R5ZIT6U	Arm® Cortex®-M4	
STMicroelectronics	NUCLEO-U575ZI-Q	STM32U575ZIT6Q	Arm® Cortex®-M33	
Syntiant	NDP9120-EVL	NDP120	M0 + HiFi	Syntiant Core 2 (98MHz, 1.1V)
Syntiant	NDP9120-EVL	NDP120	M0 + HiFi	Syntiant Core 2 (30MHz, 0.9V)
Qualcomm Innovation Center	Next Generation Snapdragon Mobile Platform HDK	Next Generation Snapdragon Mobile Platform	Qualcomm Kryo CPU(1)	Qualcomm Sensing Hub(1)

Select Keyword Spotting Results

Submitter	SoC	Accelerator	Accuracy	Latency (mS)	Energy (uJ)
Syntiant	NDP120	Syntiant Core 2	90%	1.48	43.8
STMicroelectronics	STM32U575ZIT6Q	None	90%	44.2	1138.5

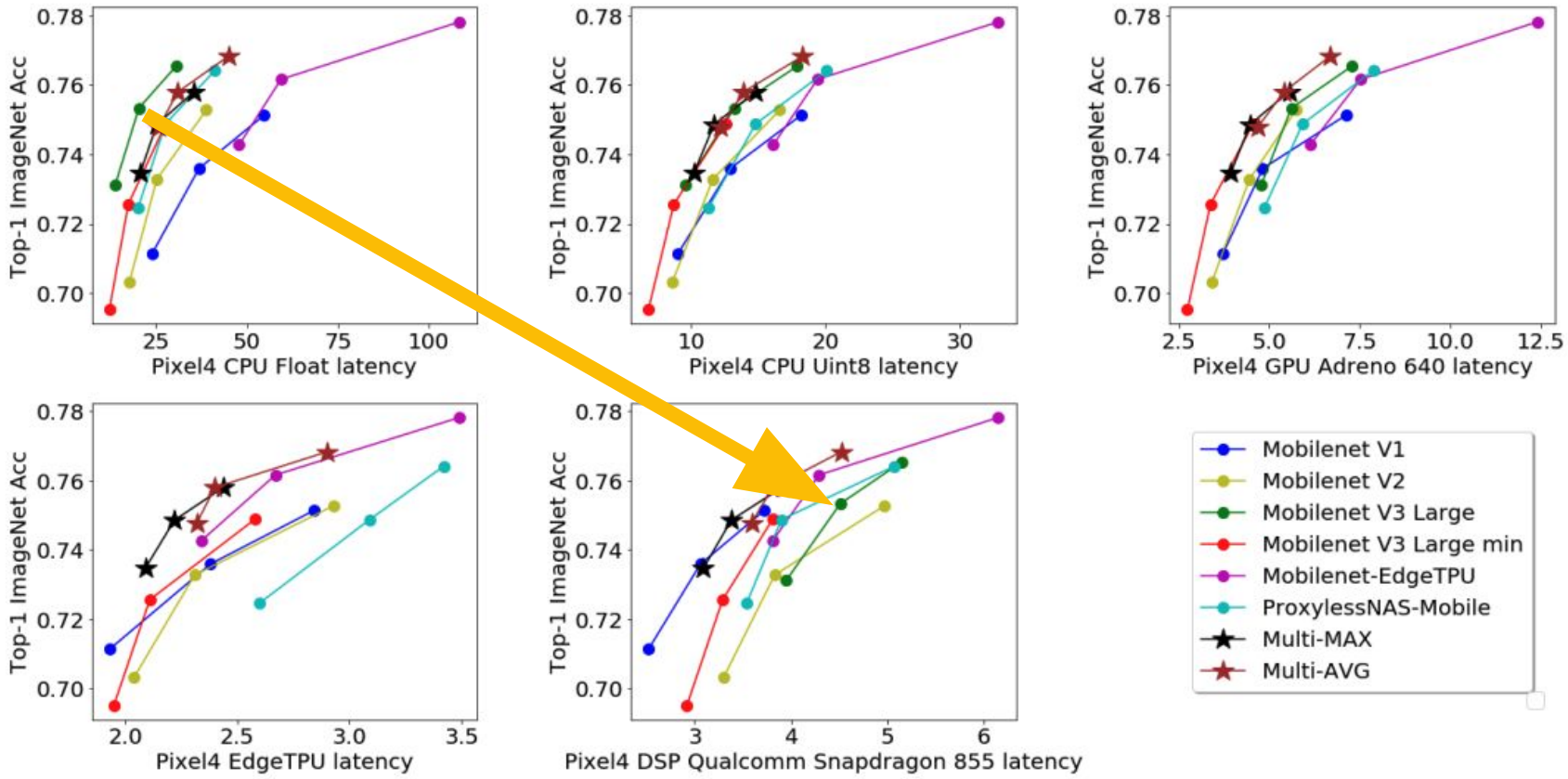
FPGA Energy Configuration





Hardware Lottery Problem

Source:
Chu, Grace, et al. "Discovering multi-hardware mobile models via architecture search." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.



Emerging TinyML Use Cases

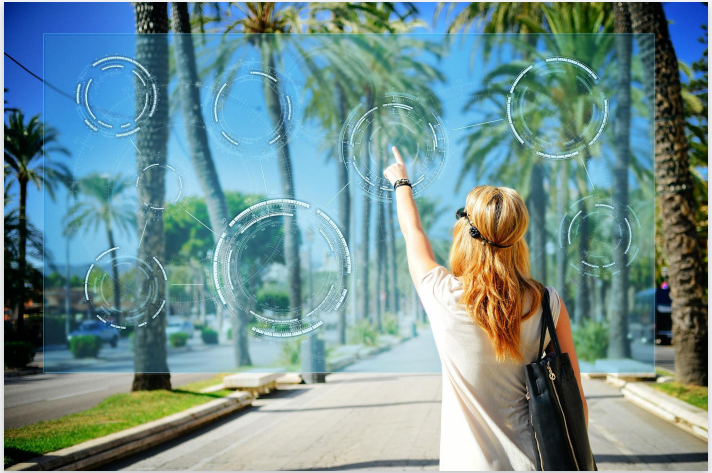
Example: Smart shoes

- Kicking
- Penalty kicking
- Passing
- Dribbling
- ...

Emerging TinyML Use Cases

Example: Augmented Reality

- Eye tracking
- Hand tracking
- Computer vision
- Superresolution
- ...



Toward Emerging Multi-DNN Models

Pipelined DNNs

Keyword Spotting

Speech Processing

- Back-to-back execution
- Execution dependency

Toward Emerging Multi-DNN Models

Pipelined DNNs

Keyword Spotting

Speech Processing

- Back-to-back execution
- Execution dependency

Concurrent DNNs

Eye Tracking

Obstacle Detection

Video Processing

- Concurrent execution
- Execution deadline

Toward Emerging Multi-DNN Models

Pipelined DNNs

Keyword Spotting

Speech Processing

- Back-to-back execution
- Execution dependency

Concurrent DNNs

Eye Tracking

Obstacle Detection

Video Processing

- Concurrent execution
- Execution deadline

Concurrent & Pipelined DNNs

Obstacle Detection

Eye Tracking → Foveated Rendering

- Challenges from both pipelined and concurrent

MLPerf

Enforce performance result replicability to ensure reliable results

Use **representative workloads**, reflecting production use-cases

Encourage innovation to improve the state-of-the-art of ML

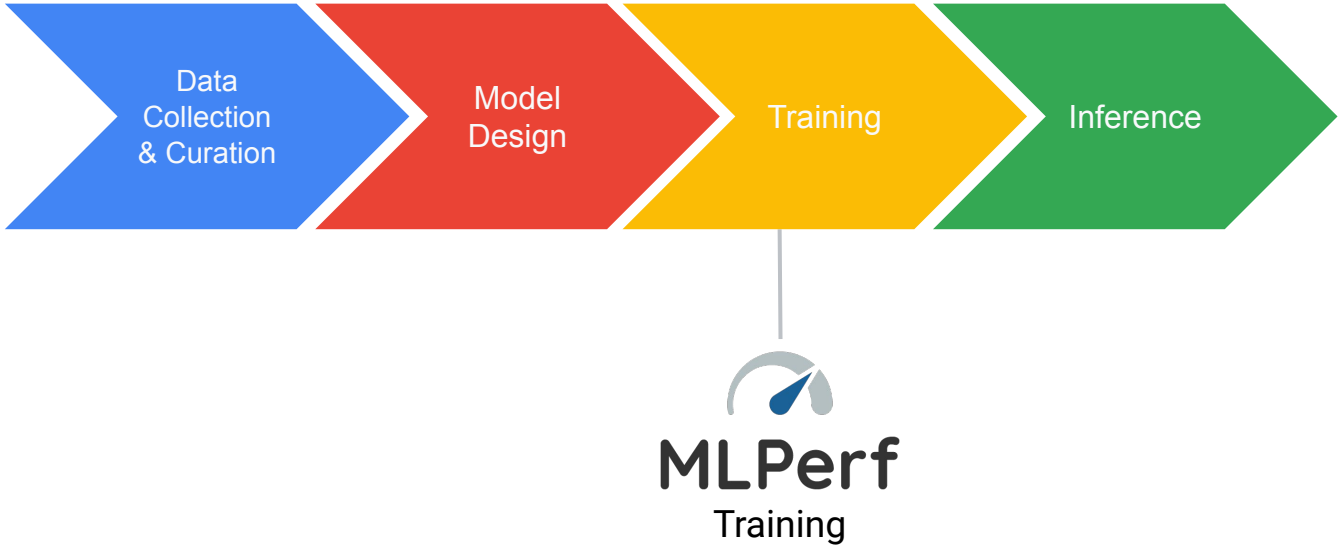
Accelerate progress in ML via **fair and useful measurement**

Serve both the **commercial and research communities**

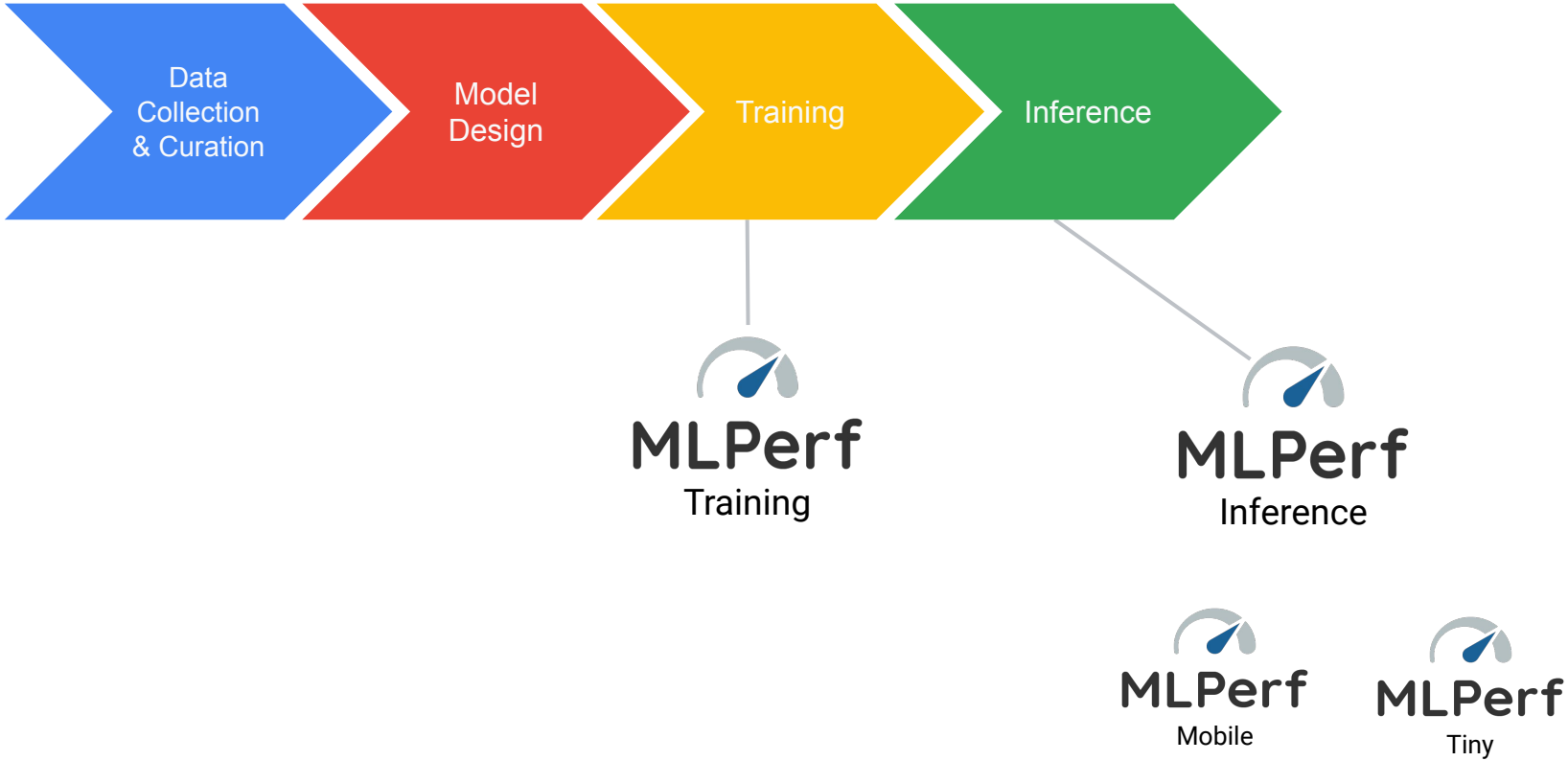
Keep **benchmarking affordable** so that all can participate

Landscape of AI benchmarking

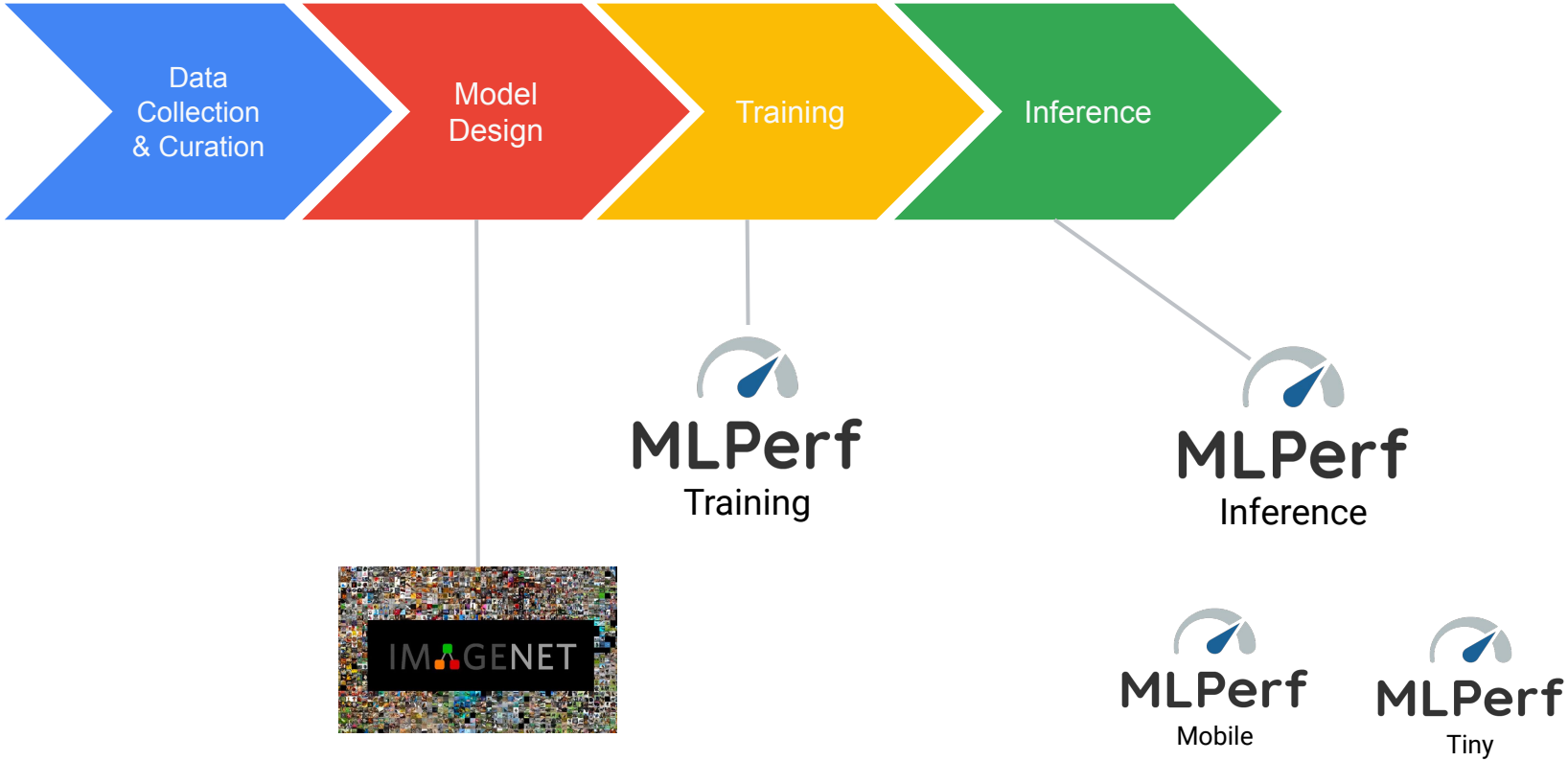
Landscape of AI benchmarking



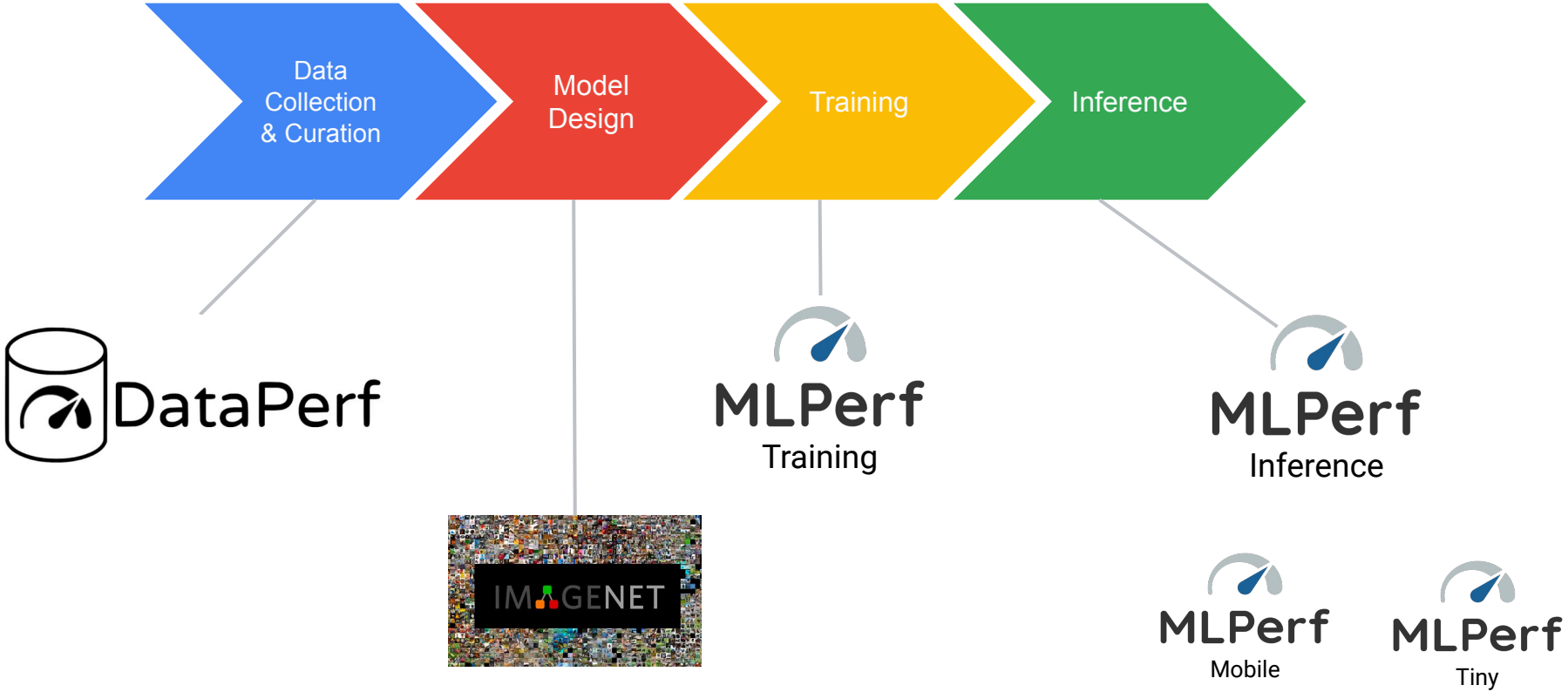
Landscape of AI benchmarking



Landscape of AI benchmarking



Landscape of AI benchmarking



MLPerf Tiny Benchmark

Colby Banbury* **Vijay Janapa Reddi*** **Peter Torelli†** **Jeremy Holleman**

Csaba Kiraly¶ **Pietro Montino*** **David Kanter**** **Sebastian Ahmed††**

Urmish Thakker[I] **Antonio Torrini[II]** **Peter Warden§** **Jay Cordaro ‡** **Giusep**

Javier Duarte[IV] **Stephen Gibellini‡** **Videet Parekh[V]** **Honson Tran[V]**

Niu Wenxu[VII] **Xu Xuesong[VII]**

Abstract

Advancements in ultra-low-power *tiny* machine learning (TinyM
promise to unlock an entirely new class of smart applications. Ho
tinued progress is limited by the lack of a widely accepted and easily
benchmark for these systems. To meet this need, we present MLPe
first industry-standard benchmark suite for ultra-low-power tiny machi
systems. The benchmark suite is the collaborative effort of more th
nizations from industry and academia and reflects the needs of the
MLPerf Tiny measures the accuracy, latency, and energy of machi
inference to properly evaluate the tradeoffs between systems. Additiona
Tiny implements a modular design that enables benchmark submitters
benefits of their product, regardless of where it falls on the ML deploy
in a fair and reproducible manner. The suite features four benchmark
spotting, visual wake words, image classification, and anomaly detect

1 Introduction

Machine learning (ML) inference on the edge is an increasingly attractive prospe
for increasing energy efficiency [4], privacy, responsiveness, and autonomy o
far, the field edge ML has predominantly focused on mobile inference, but i
have been major strides towards expanding the scope of edge ML to ultra
The field, known as "TinyML" [1], achieves ML inference under a milliWatt
the traditional power barrier preventing widely distributed machine intellige
inference on-device, and near-sensor, TinyML enables greater responsivenes
avoiding the energy cost associated with wireless communication, which at th
than that of compute [5]. Furthermore, the efficiency of TinyML enables a cl
powered, always-on applications that can revolutionize the real-time collectio
data. Deploying advanced ML applications at this scale requires the co-optimiz
the ML deployment stack to achieve the maximum efficiency. Due to this comp

*Harvard University, †EEMBC, ‡Syntiant ||UNC Charlotte §Google ¶Digital
**MLCommons ††Qualcomm ‡‡STMicroelectronics ¶SambaNova Systems ||Silicon Lab
[V]Latent AI [VI]Fermilab [VII]Peng Cheng Labs

Preprint. Under review.

MLPerf Inference Ben

Vijay Janapa Reddi,* Christine Cheng,† David Ka
Guenther Schmuelling,¶ Carole-Jean Wu,|| Brian Anderso
Mark Charlebois,†† William Chou,†† Ramesh Chukka,† Co
Pan Deng,[xxvii] Greg Diamos,[vi] Jared Duke,§ Dave Fick,[xii] J. S
Sachin Idgunji,** Thomas B. Jablin,§ Jeff Jiao,[xv] Tom St
David Lee,[xvi] Jeffery Liao,[xvii] Anton Lokhmotov,[xviii] Franci
Paulius Micikevicius,** Colin Osborne,[xix] Gennady Pekhimenko,
Dilip Sequeira,** Ashish Sirasao,[xxi] Fei Sun,[xxiii] Hanlin T
Frank Wei,[xxv] Ephrem Wu,[xxi] Lingjie Xu,[xxviii] Koichi
George Yuan,** Aaron Zhong,[xv] Peizhao Zhang,

*Harvard University †Intel ‡Real World Insights §Google ¶Microsoft
‡‡Stanford University [x]Myrtle [xi]Landing AI [xii]Mythic [xiii]Advanta
[xv]Alibaba T-Head [xvi]Facebook (formerly at MediaTek) [xvii]OPPO (forn
[xx]University of Toronto & Vector Institute [xxi]Xilinx [xxii]Tesla [xxiii]
[xxiv]Centaur Technology [xxv]Alibaba Cloud [xxvi]General Motors [xxvii]Tencent

***Abstract*—Machine-learning (ML) hardware and software system demand is burgeoning. Driven by ML applications, the number of different ML inference systems has exploded. Over 100 organizations are building ML inference chips, and the systems that incorporate existing models span at least three orders of magnitude in power consumption and five orders of magnitude in performance; they range from embedded devices to data-center solutions. Fueling the hardware are a dozen or more software frameworks and libraries. The myriad combinations of ML hardware and ML software make assessing ML-system performance in an architecture-neutral, representative, and reproducible manner challenging. There is a clear need for industry-wide standard ML benchmarking and evaluation criteria. MLPerf Inference answers that call. In this paper, we present our benchmarking method for evaluating ML inference systems. Driven by more than 30 organizations as well as more than 200 ML engineers and practitioners, MLPerf prescribes a set of rules and best practices to ensure comparability across systems with wildly differing architectures. The first call for submissions garnered more than 600 reproducible inference-performance measurements from 14 organizations, representing over 30 systems that showcase a wide range of capabilities. The submissions attest to the benchmark's flexibility and adaptability.**

***Index Terms*—Machine Learning, Inference, Benchmarking**

I. Introduction

Machine learning (ML) powers a variety of applications from computer vision ([20], [18], [34], [29]) and natural-language processing ([50], [16]) to self-driving cars ([55], [6]) and autonomous robotics [32]. Although ML-model training has been a development bottleneck and a considerable expense [4], inference has become a critical workload. Models can serve as many as 200 trillion queries and perform over 6 billion translations a day [31]. To address these growing computational demands, hardware, software, and system developers have focused on inference performance for a variety of use cases by
ware. Estimates
specialized infe
20 companies a

Each ML sy
trading off later
result is many
data sets, fram
inference engi
performance ne
including but n
tion, object det
automatic spee
dations. Even f
many ML mod
of scenarios fr
continuously a
multiple camer
ML tasks have
time processing
model's functi
specific, and th
To quantify the
that is architect

Both academ
ML inference
AIMatrix [3], E
industry, as wel
and DAWNBen
substantial con
developed with
signers. As a r
machine learni

DataPerf: Benchmarks for Data-Centric AI Development

Mark Mazumder[1], Colby Banbury[1], Xiaozhe Yao[2], Bojan Karlaš[2], William Gaviria Rojas[3], Sudnya Diamos[3], Greg Diamos[4], Lynn He[5], Alicia Parrish [9], Hannah Rose Kirk[18], Jessica Quaye[1], Charvi Rastogi[12], Douwe Kiela[10,22], David Jurado[7,21], David Kanter[7], Rafael Mosquera[7,21], Juan Ciro[7,21], Lora Aroyo[9], Bilge Acun[8], Lingjiao Chen[10], Mehul Smriti Raje[3], Max Bartolo[17,20], Sabri Eyuboglu[10], Amirata Ghorbani[10], Emmett Goodman[10], Oana Inel[19], Tariq Kane[3,9], Christine R. Kirkpatrick[11], Tzu-Sheng Kuo[12], Jonas Mueller[13], Tristan Thrush[10], Joaquin Vanschoren[14], Margaret Warren[15], Adina Williams[8], Serena Yeung[10], Newsha Ardalani[8], Praveen Paritosh[7], Lilith Bat-Leah[7], Ce Zhang[2], James Zou[10], Carole-Jean Wu[8], Cody Coleman[3], Andrew Ng[4,5,10], Peter Mattson[9], and Vijay Janapa Reddi[1]

[1]Harvard University, [2]ETH Zurich, [3]Coactive.AI, [4]Landing AI, [5]DeepLearning.AI, [7]MLCommons, [8]Meta, [9]Google, [10]Stanford University, [11]San Diego Supercomputer Center, UC San Diego, [12]Carnegie Mellon University, [13]Cleanlab, [14]Eindhoven University of Technology, [15]Institute for Human and Machine Cognition, [16]Kaggle, [17]Cohere, [18]University of Oxford, [19]University of Zurich, [20]University College London, [21]Factored, [22]Contextual AI

Abstract

Machine learning research has long focused on models rather than datasets, and prominent datasets are used for common ML tasks without regard to the breadth, difficulty, and faithfulness of the underlying problems. Neglecting the fundamental importance of data has given rise to inaccuracy, bias, and fragility in real-world applications, and research is hindered by saturation across existing dataset benchmarks. In response, we present DataPerf, a community-led benchmark suite for evaluating ML datasets and data-centric algorithms. We aim to foster innovation in data-centric AI through competition, comparability, and reproducibility. We enable the ML community to iterate on datasets, instead of just architectures, and we provide an open, online platform with multiple rounds of challenges to support this iterative development. The first iteration of DataPerf contains five benchmarks covering a wide spectrum of data-centric techniques, tasks, and modalities in vision, speech, acquisition, debugging, and diffusion prompting, and we support hosting new contributed benchmarks from the community. The benchmarks, online evaluation platform, and baseline implementations are open source, and the MLCommons Association will maintain DataPerf to ensure long-term benefits to academia and industry.

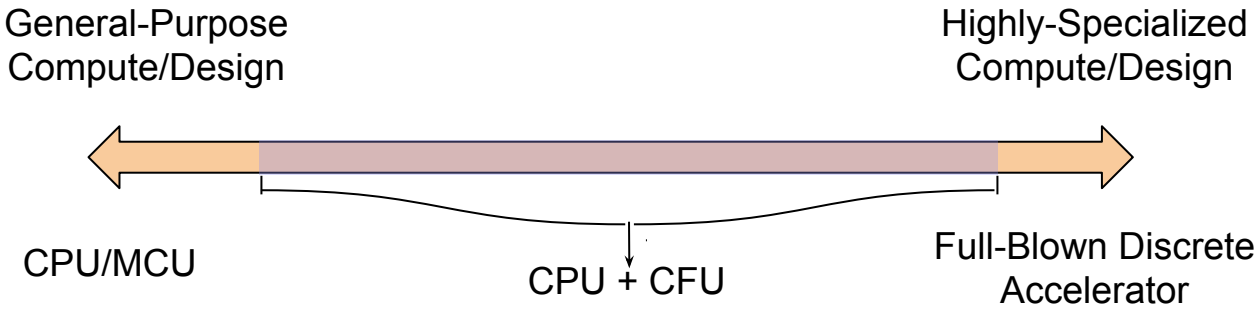
1 Introduction

Machine learning research has overwhelmingly focused on improving models rather than on improving datasets. Large public datasets such as ImageNet [14], Freebase [7], Switchboard [22], and SQuAD [44] serve as compasses for benchmarking model performance. Consequently, researchers eagerly adopt the largest existing dataset without fully considering its breadth, difficulty and fidelity to the underlying problem. Critically, better data quality [2] is increasingly necessary to improve generalization, avoid bias, and aid safety in data cascades [48]. Without high-quality training data models can exhibit performance discrepancies leading to reduced accuracy and persistent fairness

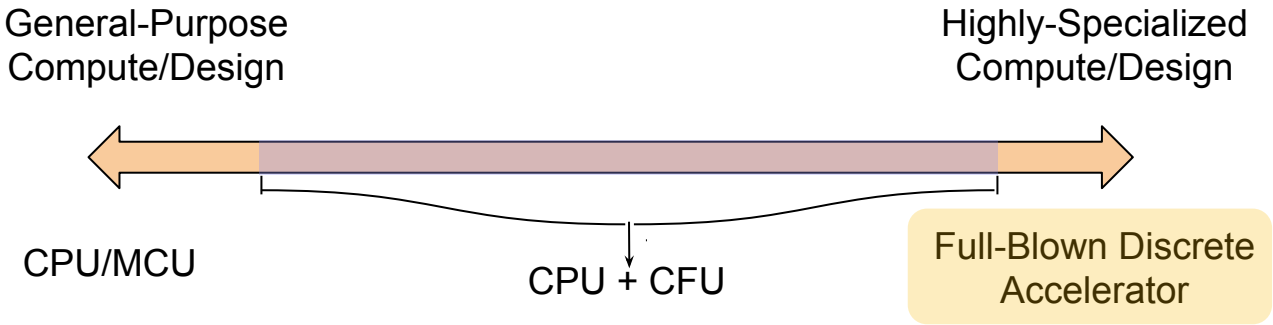
37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks.

Paper Discussions

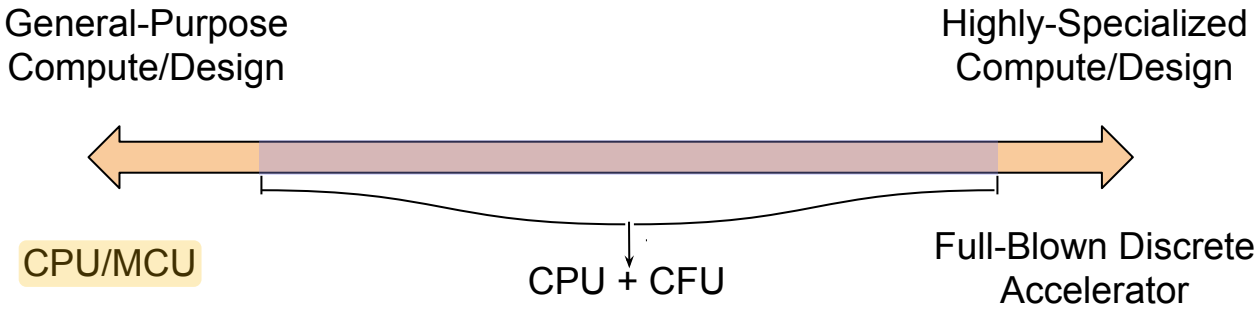
The Need for Agile and Full-Stack Frameworks



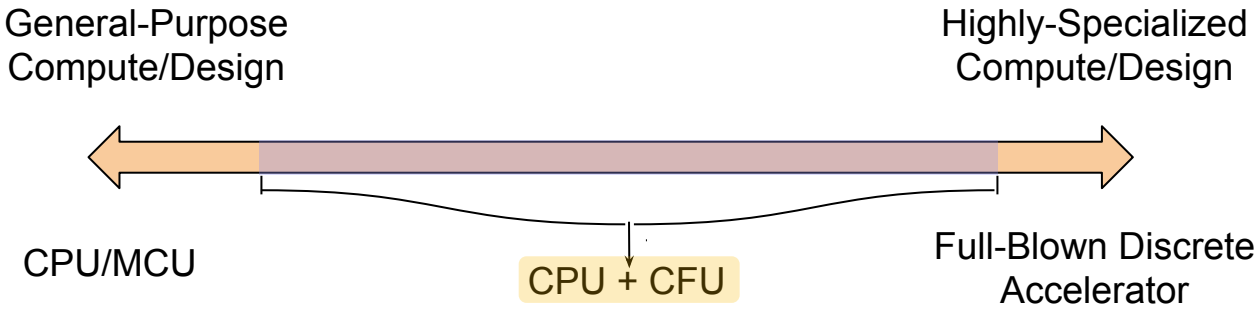
The Need for Agile and Full-Stack Frameworks



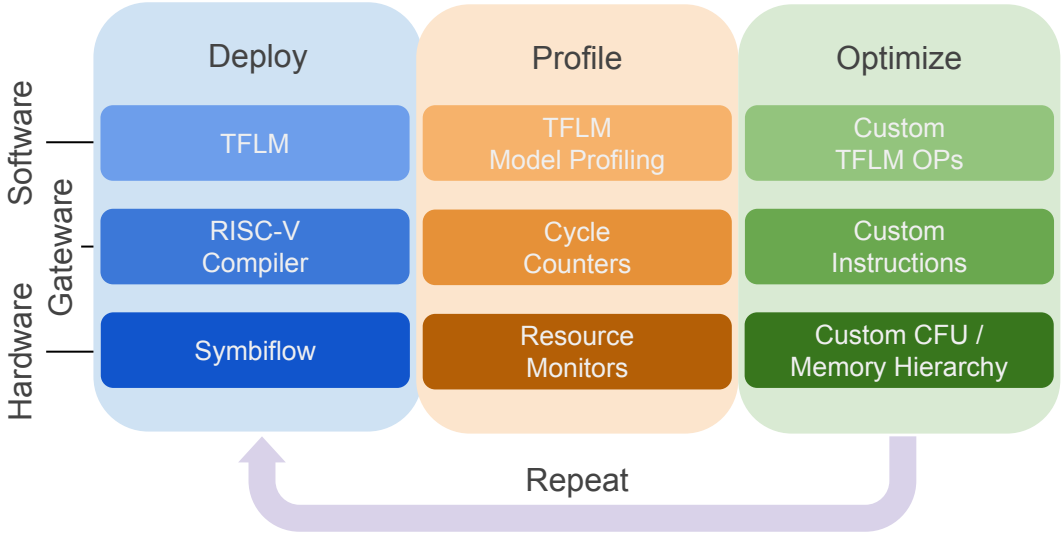
The Need for Agile and Full-Stack Frameworks



The Need for Agile and Full-Stack Frameworks

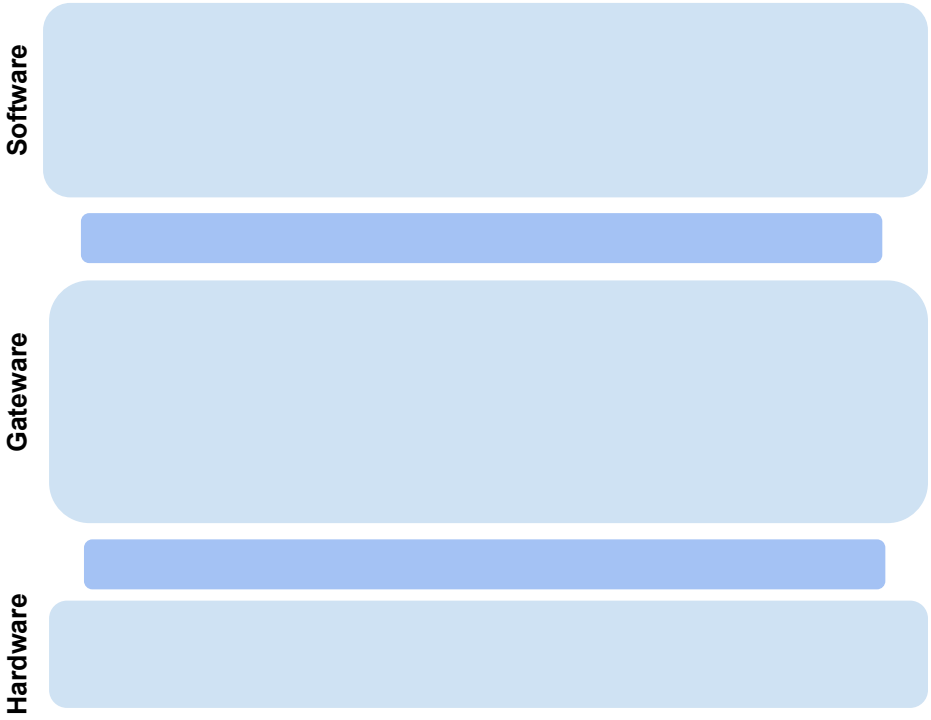


The Need for Agile and Full-Stack Frameworks



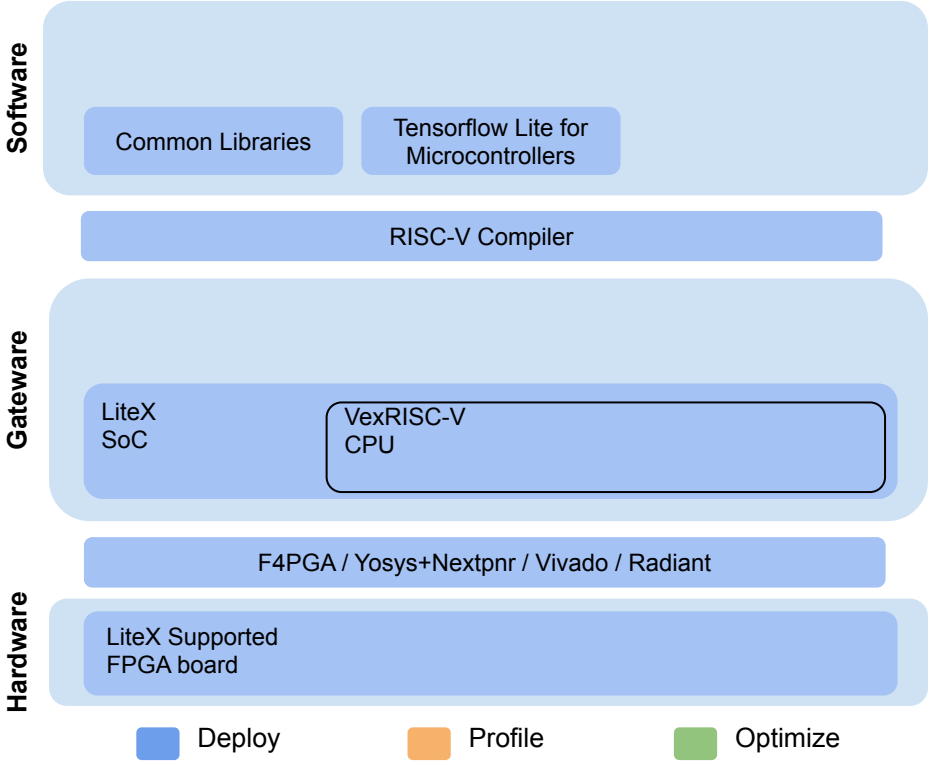
Agile Design Methodology

The Need for Agile and Full-Stack Frameworks



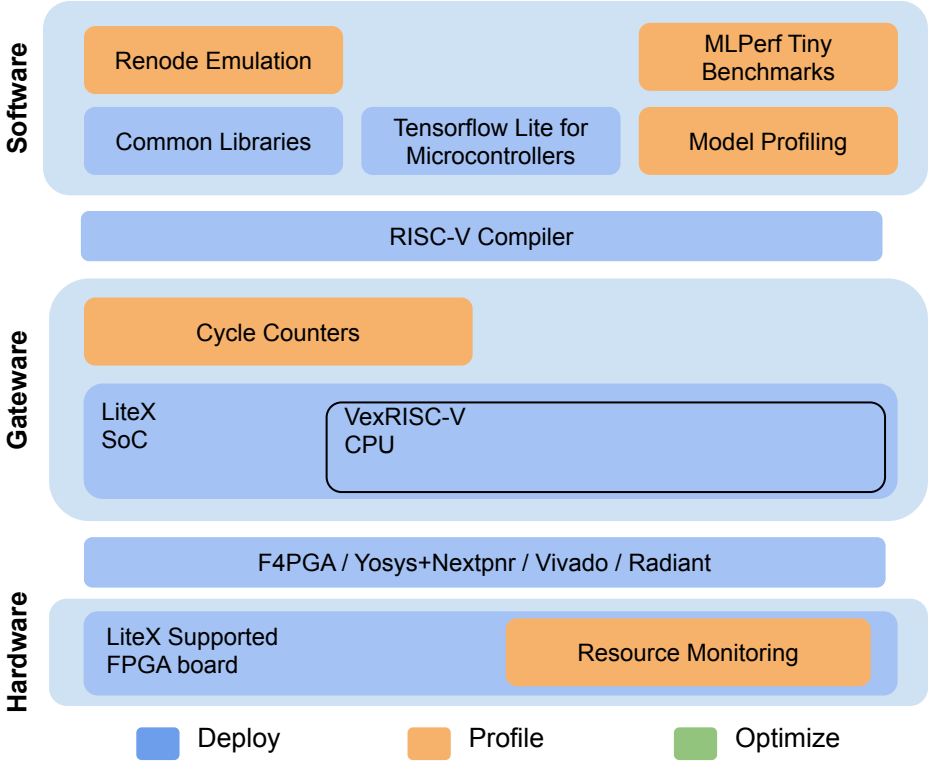
Full-Stack Open-Source Framework

The Need for Agile and Full-Stack Frameworks



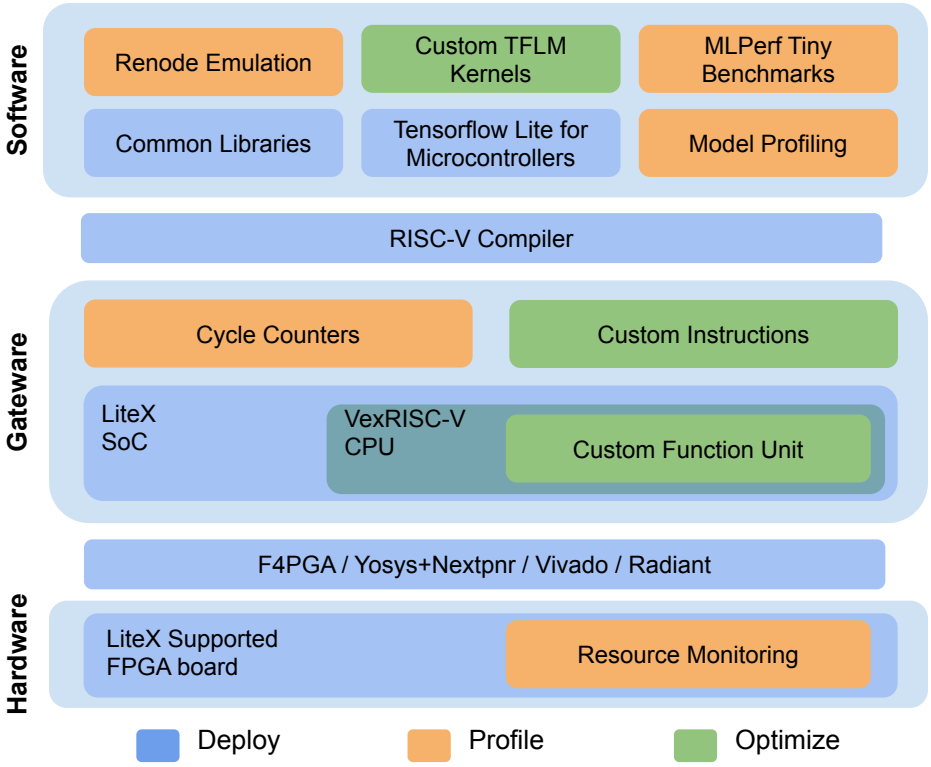
Full-Stack Open-Source Framework

The Need for Agile and Full-Stack Frameworks



Full-Stack Open-Source Framework

The Need for Agile and Full-Stack Frameworks



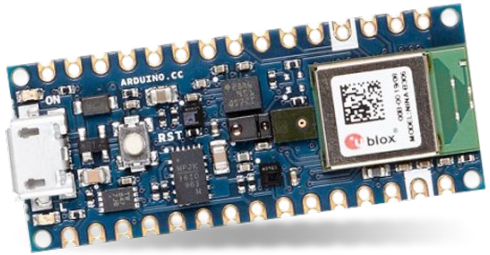
Full-Stack Open-Source Framework

Related Works

	Open Source	Full Stack	Full SoC	Tightly Coupled/ Specialized ISA	Fine-Grained Accelerated ML Ops	Hardware & Engineer In-The-Loop	Stock Compiler	Automated CPU↔Accelerator Design Space Exploration	TinyML Focus
CFU Playground	✓	✓	✓	✓	✓	✓	✓	✓	✓
Chipyard [20]	✓	✓	✓	✓	✓	✓	✓	✗	✗
Centrifuge [21]	✓	✓	✓	✓	✓	✓	✗	✗	✗
Embedded Scalable Platform [22]	✓	✓	✓	✗	✗	✓	✓	✗	✗
Gemmini [23]	✓	✓	✓	✗	✗	✗	✓	✗	✗
hls4ml [24]	✓	✗	✓	✗	✗	✓	✗	✗	✓
Deepburning [25]	✓	✓	✓	✗	✗	✗	✗	✗	✗
DNN-Weaver [26]	✓	✓	✗	✓	✗	✗	✗	✗	✗
DNN-Builder [27]	✓	✓	✗	✗	✗	✗	✗	✗	✗
FINN [28]	✓	✗	✗	✗	✗	✗	✗	✗	✗

TABLE III: Comparison of CFU Playground with open-source toolchains supporting custom hardware design for ML workloads. CFU Playground focuses on *open-source* development across the *full system* stack, while providing varying levels of flexibility for hardware and software (co-)design.

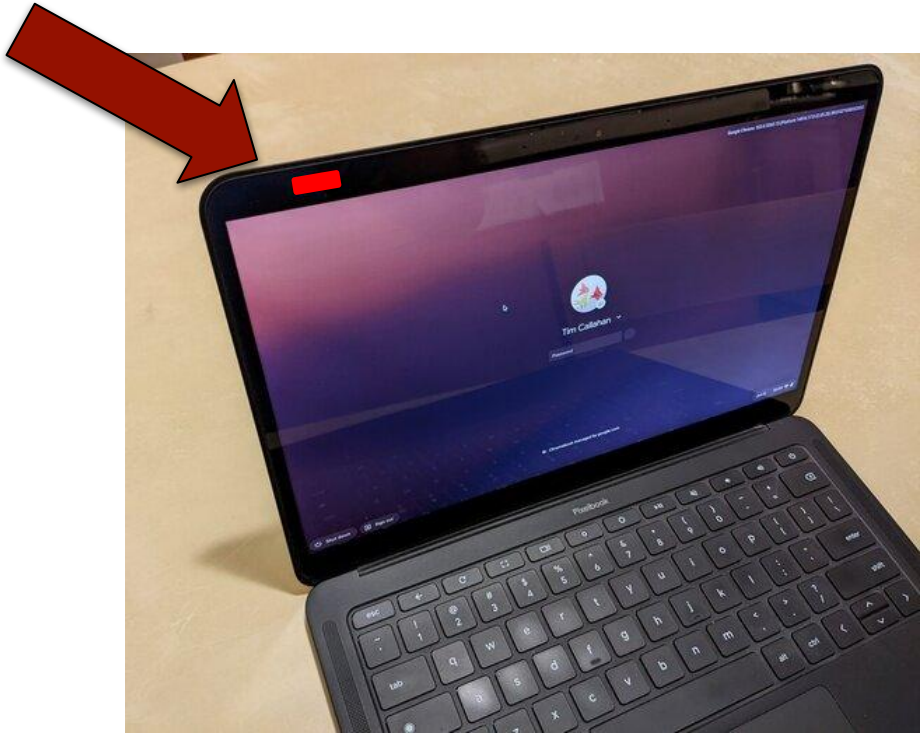
Accelerating TinyML on FPGAs



MCUs: **KBs** of RAM, **Fixed/slow** processor

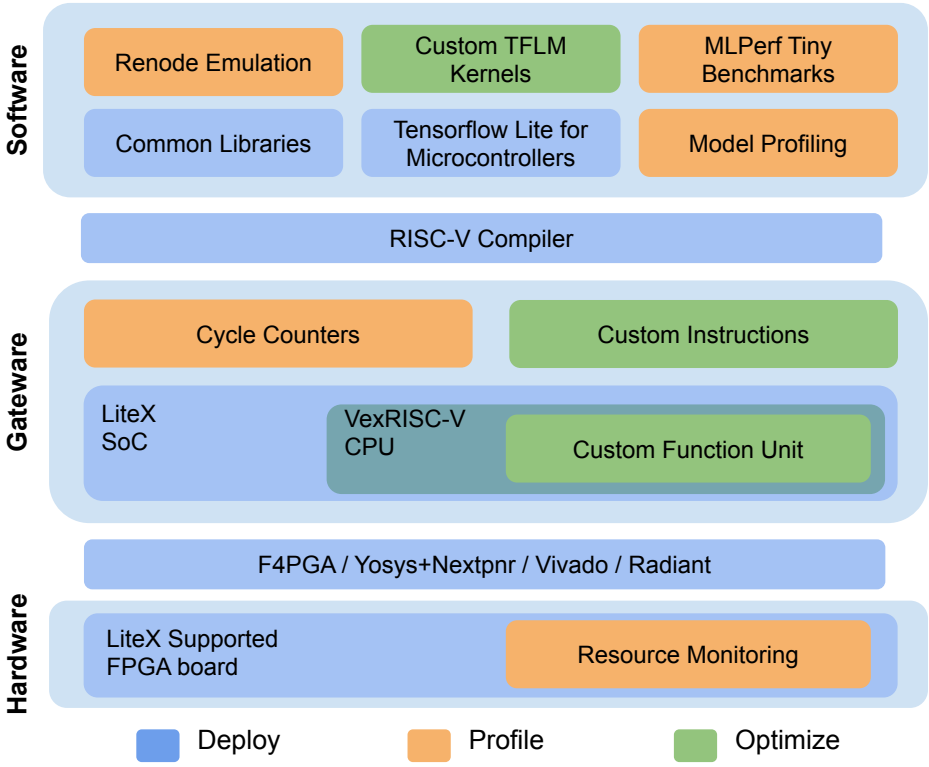
Specialized Hardware Customization (on FPGAs)

Real World Use Case



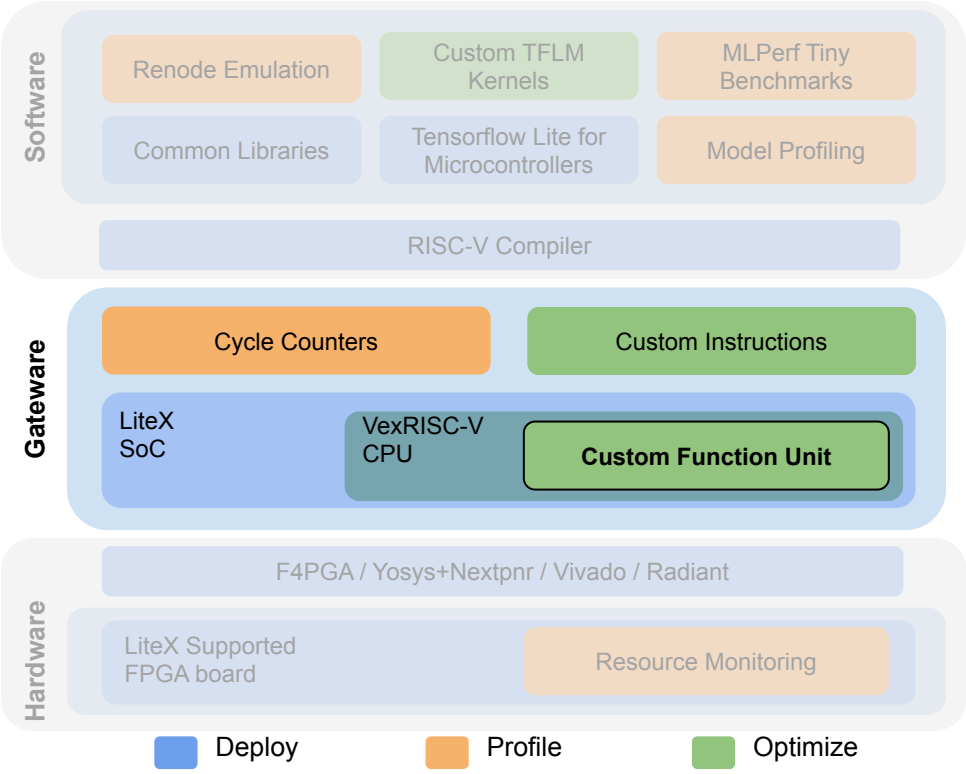
Chromebook Sensor Designed with CFU Playground

CFU Playground



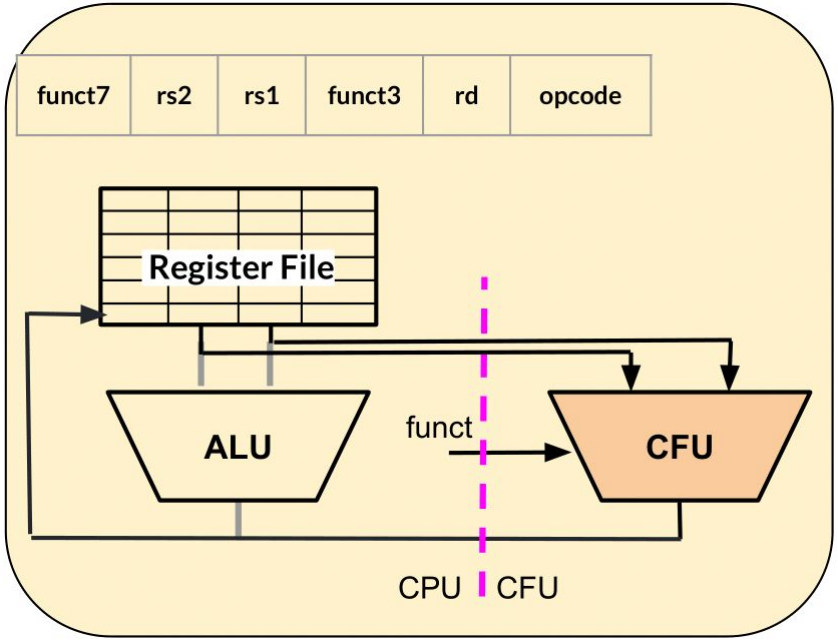
Full-Stack Open-Source Framework

CFU Playground



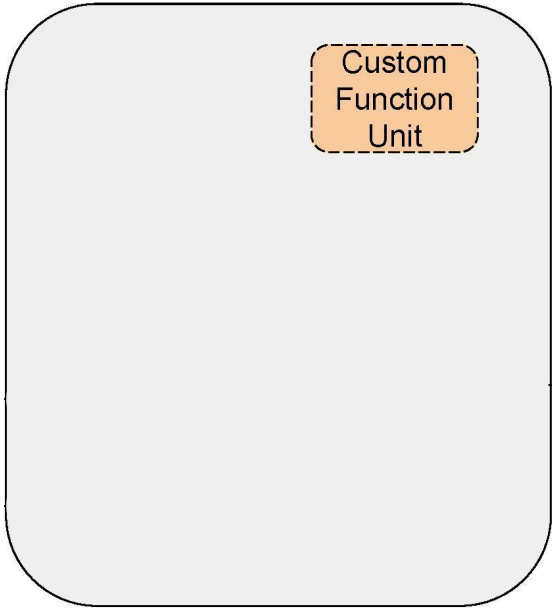
Full-Stack Open-Source Framework

Custom Function Units (CFU)

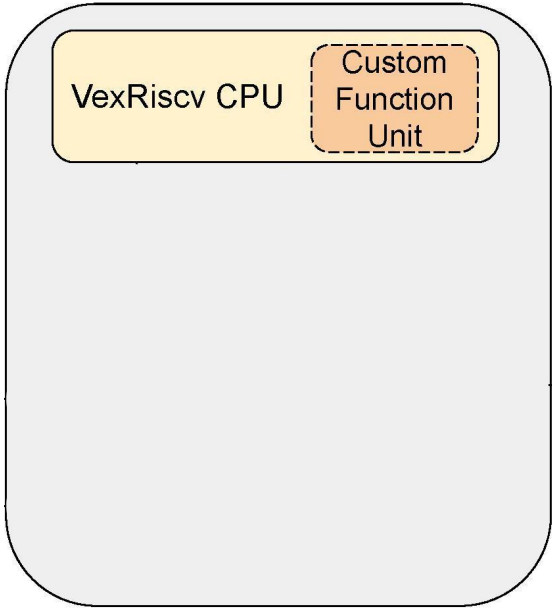


Acceleration via Custom Function Unit (CFU)

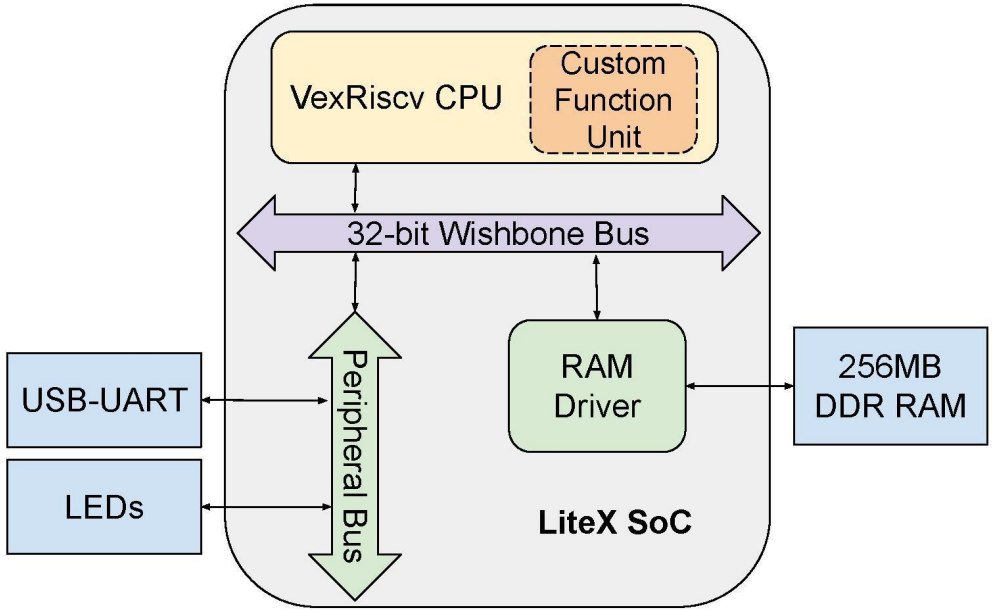
System On-Chip (SoC) Integration



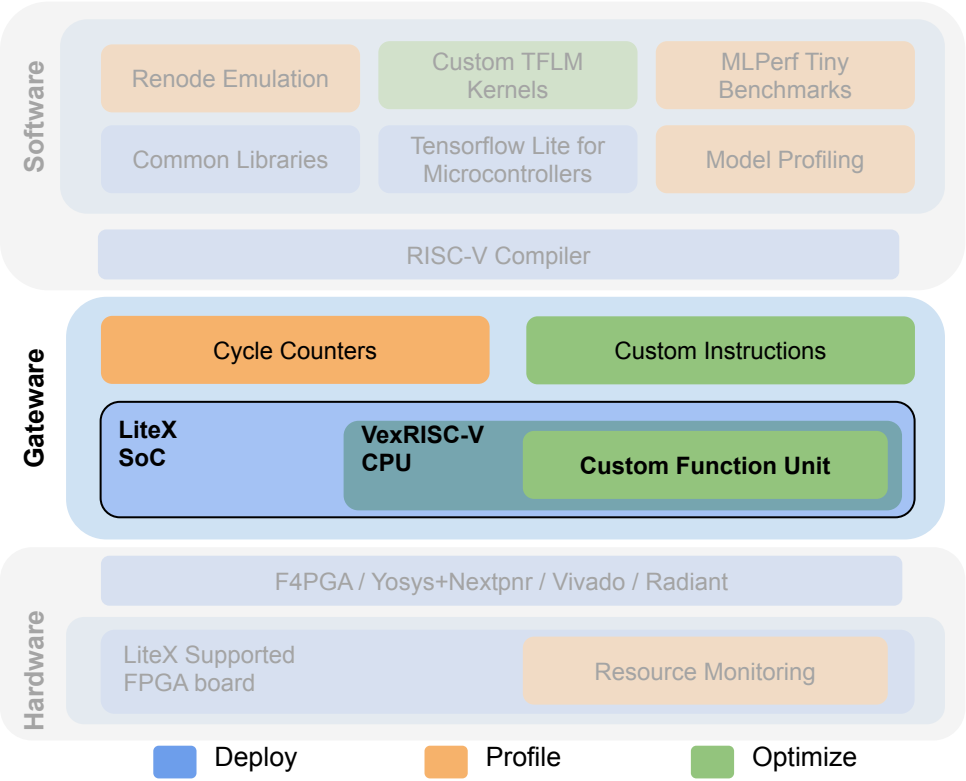
System On-Chip (SoC) Integration



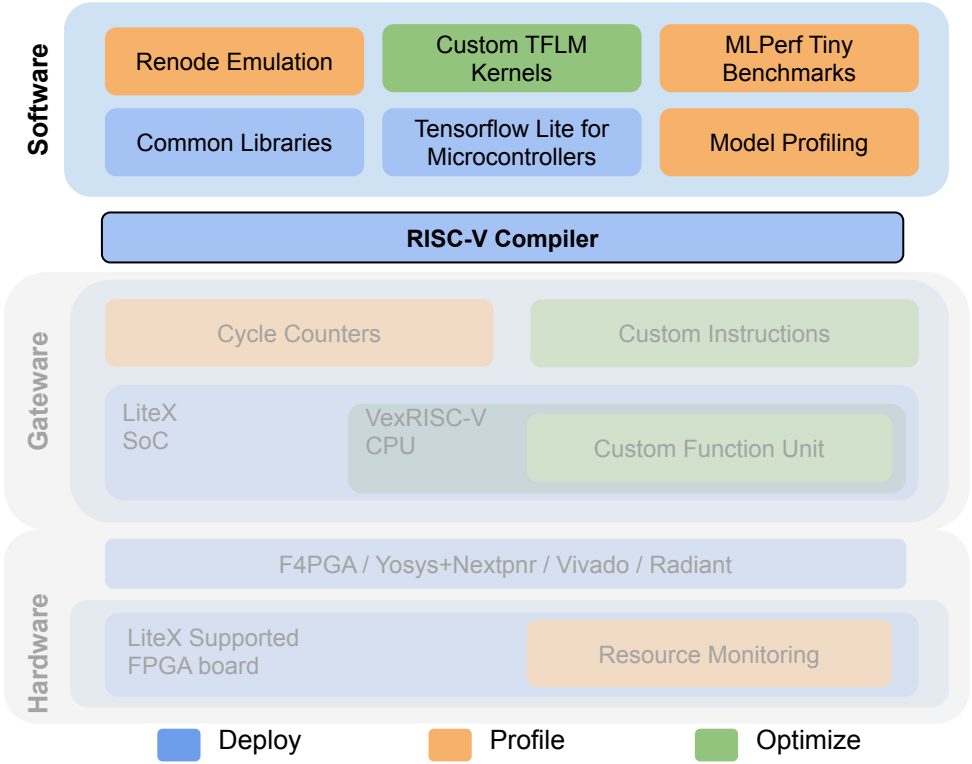
System On-Chip (SoC) Integration



System On-Chip (SoC) Integration



CFU Software Interface



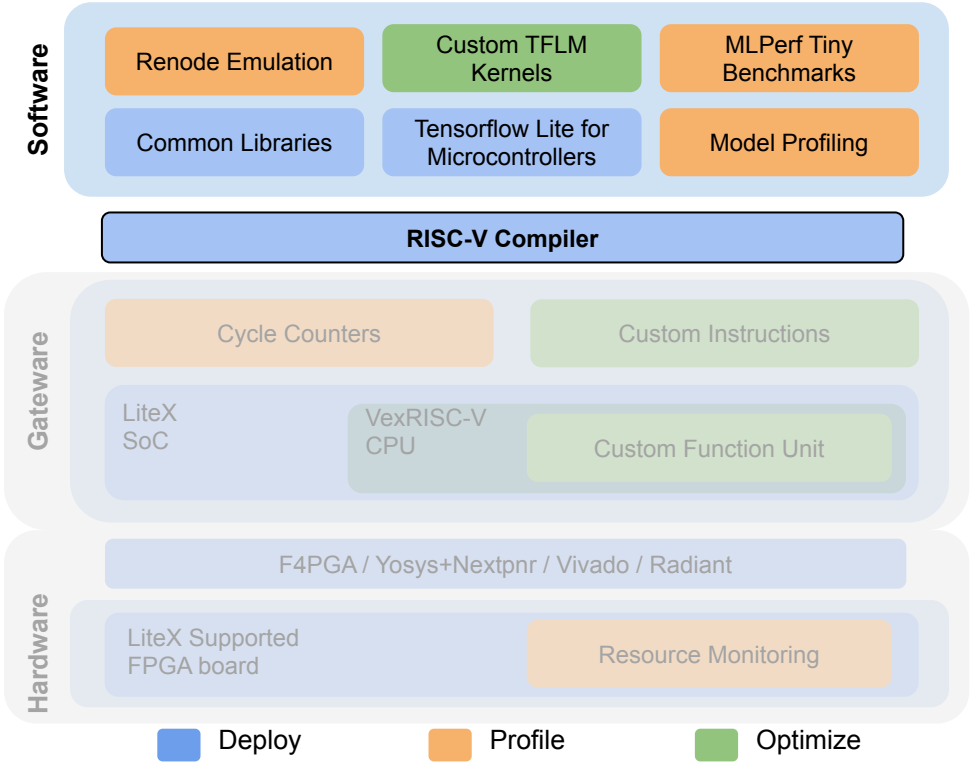
Access new instruction as C function call:

```
rslt = cfu_op(funct3, funct7, op1, op2);
```

Compile-time constants

C / C++ variables / expressions

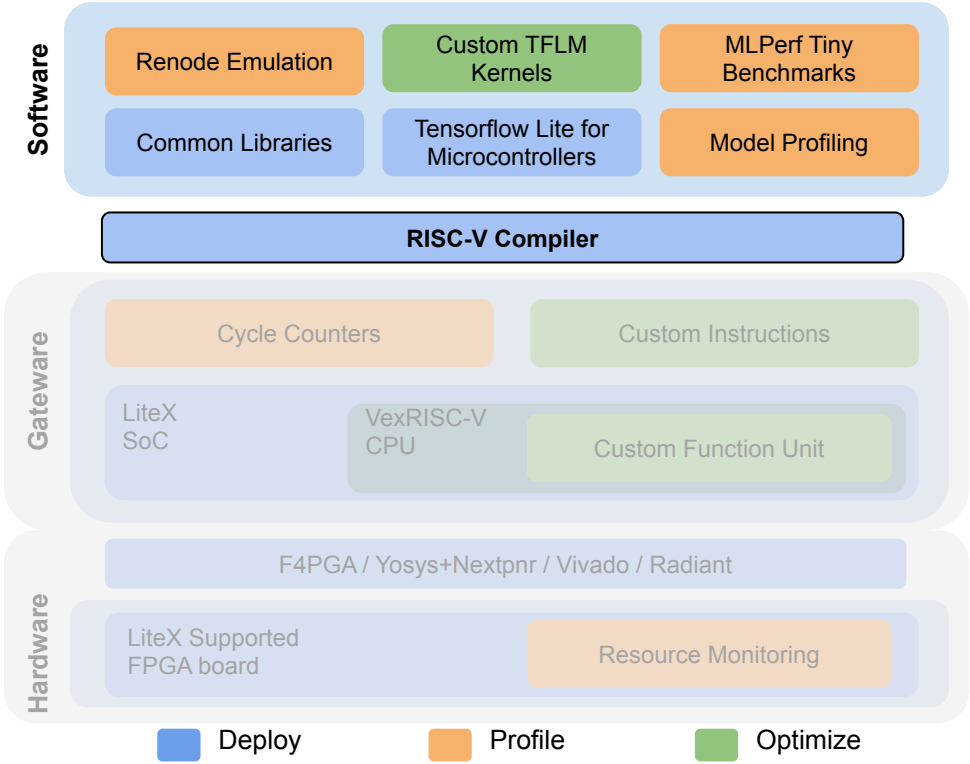
CFU Software Interface



Custom instruction macros intermix with plain C code

```
t1 = *x;
t2 = cfu_op(0, 0, t1, b);
t3 = cfu_op(1, 0, t2, b);
*x = t3;
```

CFU Software Interface



Custom instruction macros intermix with plain C code

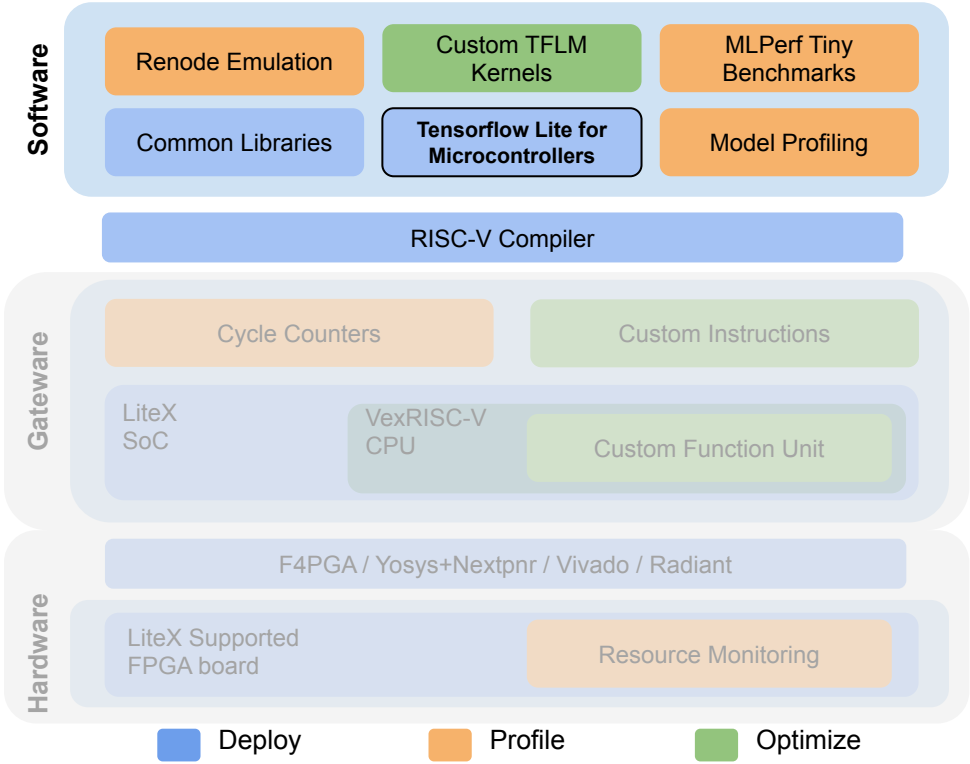
```
t1 = *x;
t2 = cfu_op(0, 0, t1, b);
t3 = cfu_op(1, 0, t2, b);
*x = t3;
```

Compiled and disassembled:

```
400001a0:  00812783   lw        a5,8(sp)
400001a4:  00d7878b   cfu[0,0]  a5, a5, a3
400001a8:  00d7978b   cfu[0,1]  a5, a5, a3
400001ac:  00f12423   sw        a5,8(sp)
```

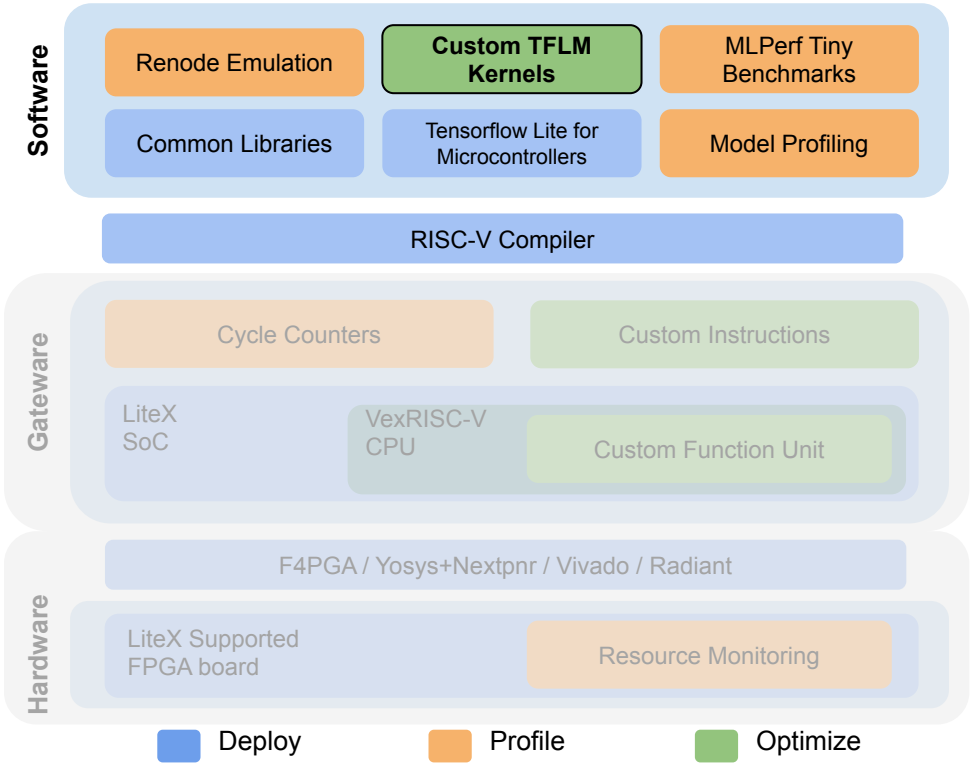
No overhead!

ML Deployment Framework



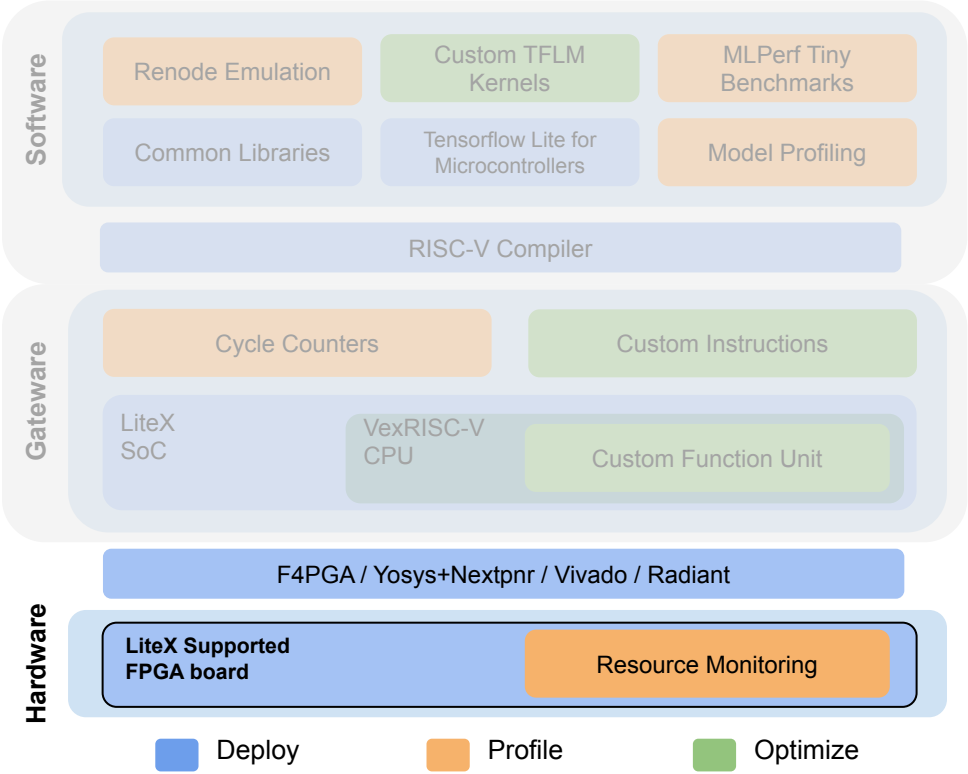

```
const int32_t input_offset = params.input_offset;  // r = s(q - Z)

for (int batch = 0; batch < batches; ++batch) {
  for (int out_y = 0; out_y < output_height; ++out_y) {
    const int in_y_origin = (out_y * stride_height) - pad_height;
    for (int out_x = 0; out_x < output_width; ++out_x) {
      const int in_x_origin = (out_x * stride_width) - pad_width;
      for (int out_channel = 0; out_channel < output_depth; ++out_channel) {
        int32_t acc = 0;
        for (int filter_y = 0; filter_y < filter_height; ++filter_y) {
          const int in_y = in_y_origin + dilation_height_factor * filter_y;
          for (int filter_x = 0; filter_x < filter_width; ++filter_x) {
            const int in_x = in_x_origin + dilation_width_factor * filter_x;

            // Zero padding by omitting the areas outside the image.
            const bool is_point_inside_image =
                (in_x >= 0) && (in_x < input_width) && (in_y >= 0) &&
                (in_y < input_height);

            if (!is_point_inside_image) {
              continue;
            }

            for (int in_channel = 0; in_channel < input_depth; ++in_channel) {
              int32_t input_val = input_data[Offset(input_shape, batch, in_y,
                                                    in_x, in_channel)];
              int32_t filter_val = filter_data[Offset(
                  filter_shape, out_channel, filter_y, filter_x, in_channel)];

              acc += filter_val * (input_val + input_offset);
            }
          }
        }
        (use acc)
```

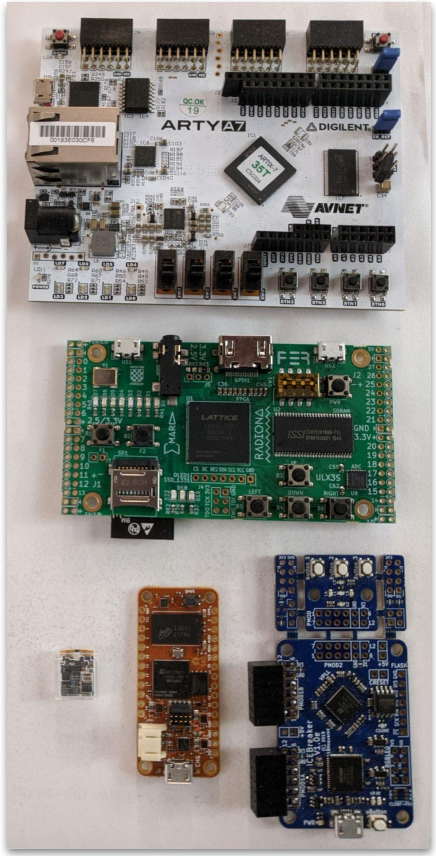
Accelerated Kernels




```
const int32_t input_offset = params.input_offset;  // r = s(q - Z)
// CFU: copy input_offset into the CFU
cfu_init_offset(input_offset);

for (int batch = 0; batch < batches; ++batch) {
  for (int out_y = 0; out_y < output_height; ++out_y) {
    const int in_y_origin = (out_y * stride_height) - pad_height;
    for (int out_x = 0; out_x < output_width; ++out_x) {
      const int in_x_origin = (out_x * stride_width) - pad_width;
      for (int out_channel = 0; out_channel < output_depth; ++out_channel) {

        //int32_t acc = 0;
        // CFU: set the CFU internal acc to ZERO
        cfu_clear_acc();

        for (int filter_y = 0; filter_y < filter_height; ++filter_y) {
          const int in_y = in_y_origin + dilation_height_factor * filter_y;
          for (int filter_x = 0; filter_x < filter_width; ++filter_x) {
            const int in_x = in_x_origin + dilation_width_factor * filter_x;

            ...

            for (int in_channel = 0; in_channel < input_depth; ++in_channel) {
              int32_t input_val = input_data[Offset(input_shape, batch, in_y,
                                                    in_x, in_channel)];
              int32_t filter_val = filter_data[Offset(
                  filter_shape, out_channel, filter_y, filter_x, in_channel)];

              // acc += filter_val * (input_val + input_offset);
              // CFU: add-multiply-accumulate in the CFU
              cfu_macc_with_offset(filter_val, input_val);
            }
          }
        }

        // CFU: retrieve final acc value from the CFU
        int32_t acc = cfu_get_acc();
```

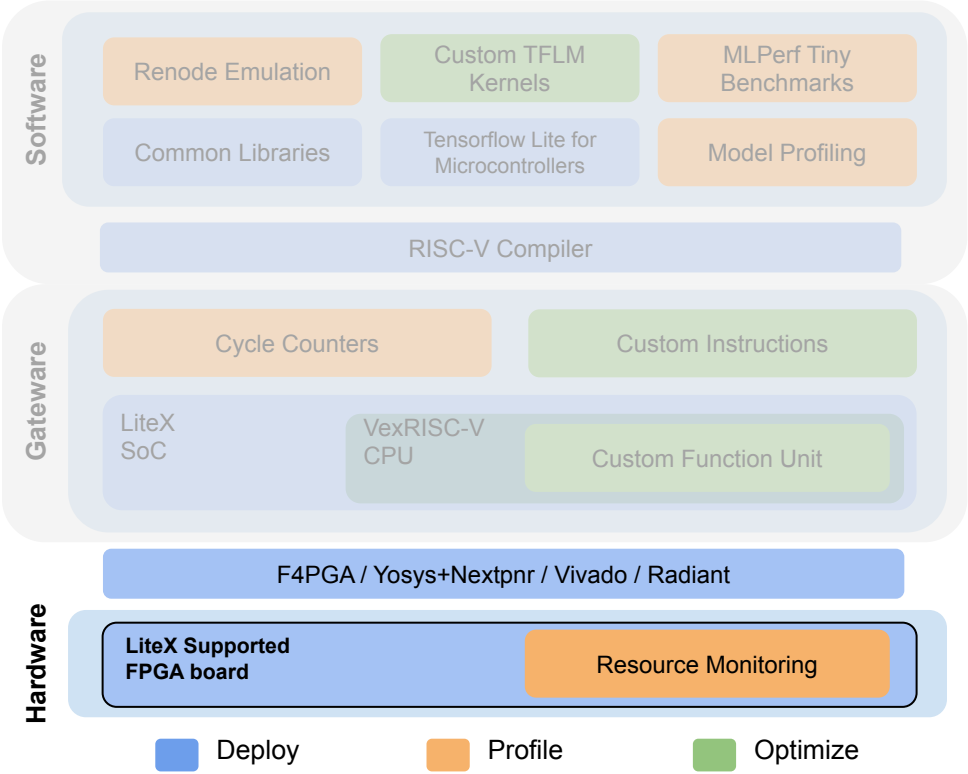
Hardware In-The-Loop

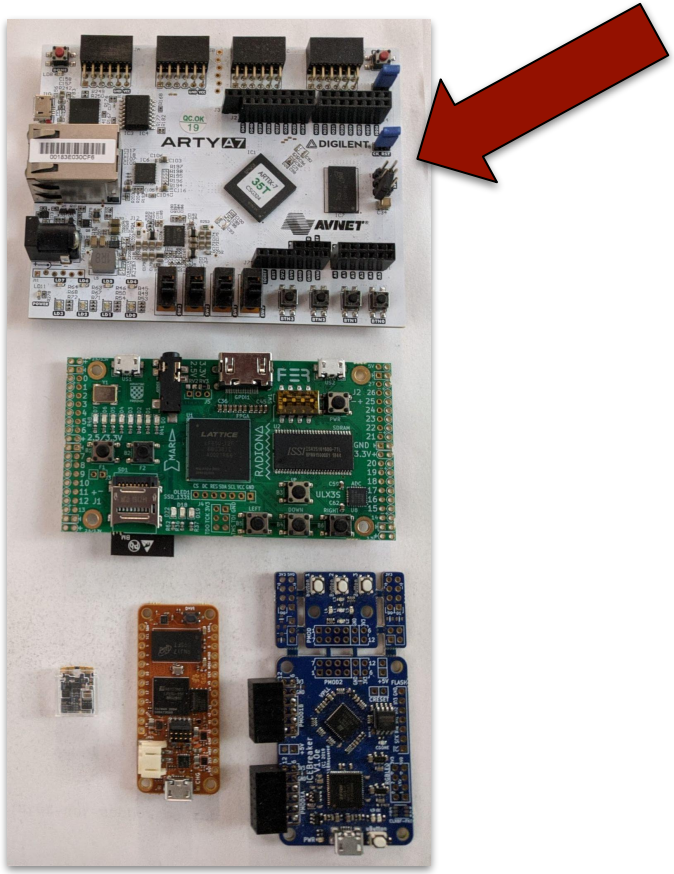




Diverse Family of FPGA Boards

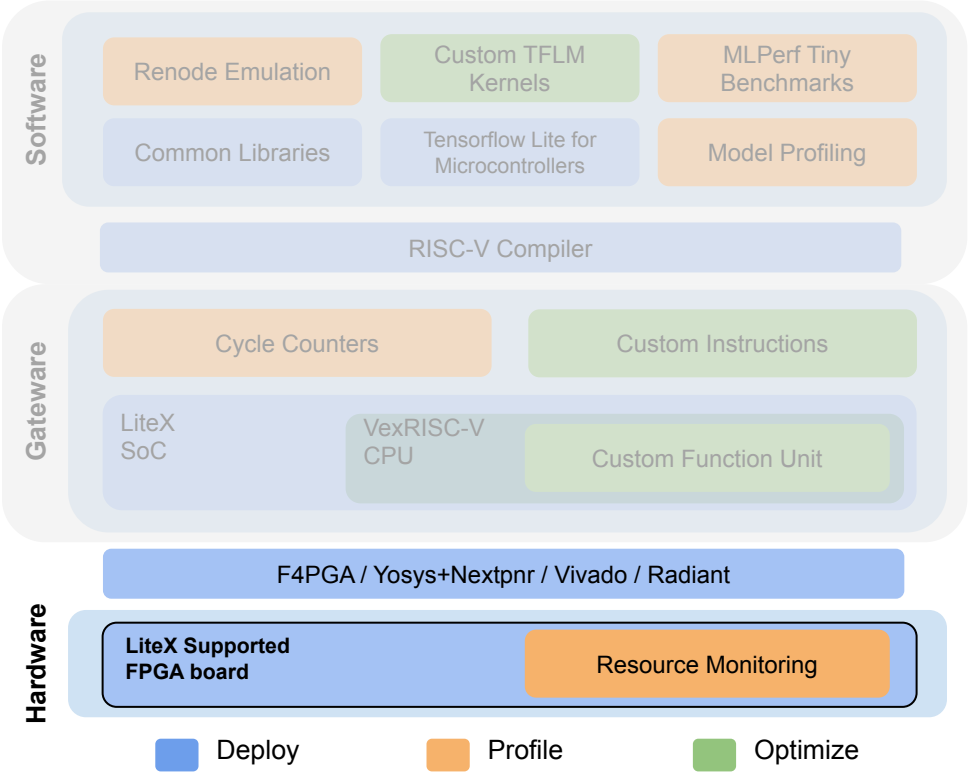
Hardware In-The-Loop

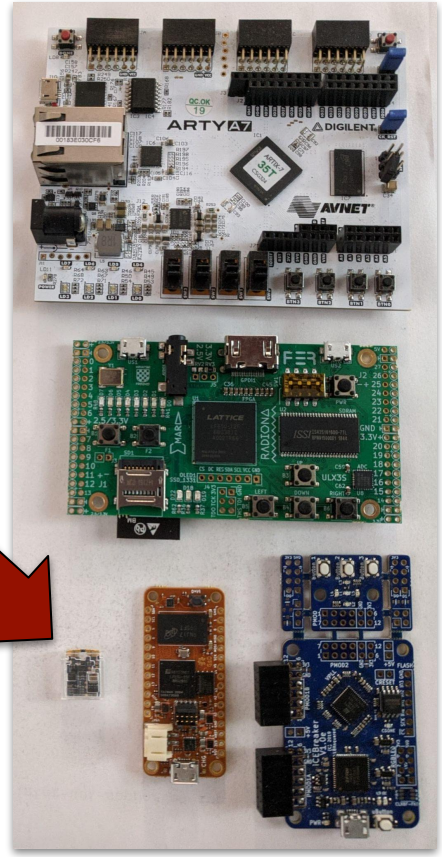




Diverse Family of FPGA Boards

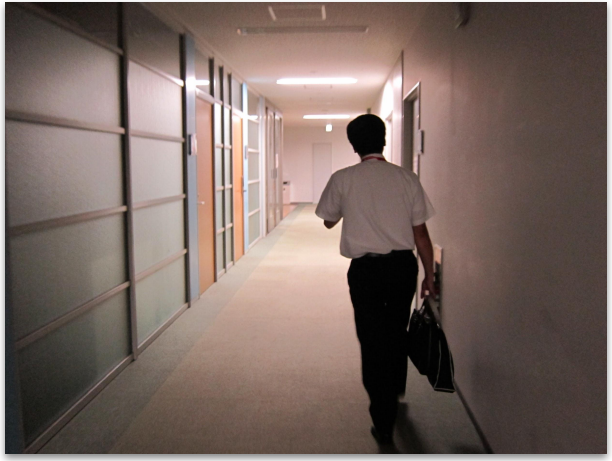
Hardware In-The-Loop

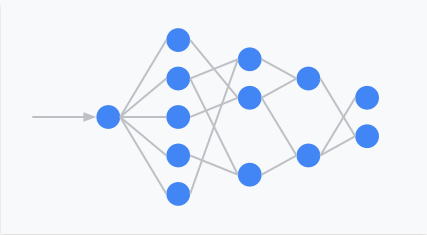


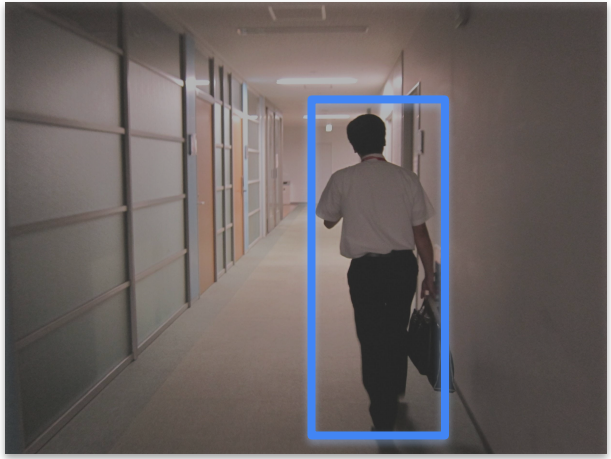


Diverse Family of FPGA Boards

FPGA Acceleration for Image Classification

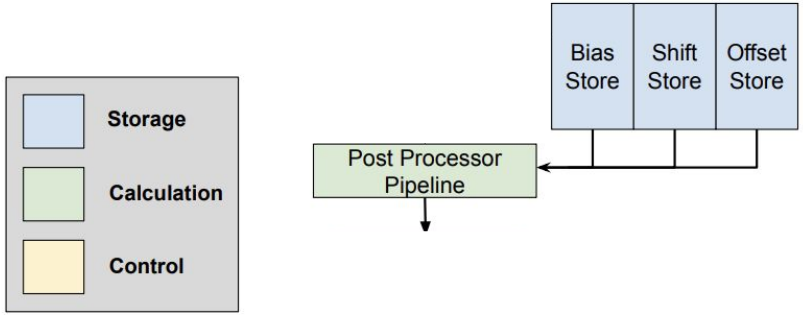






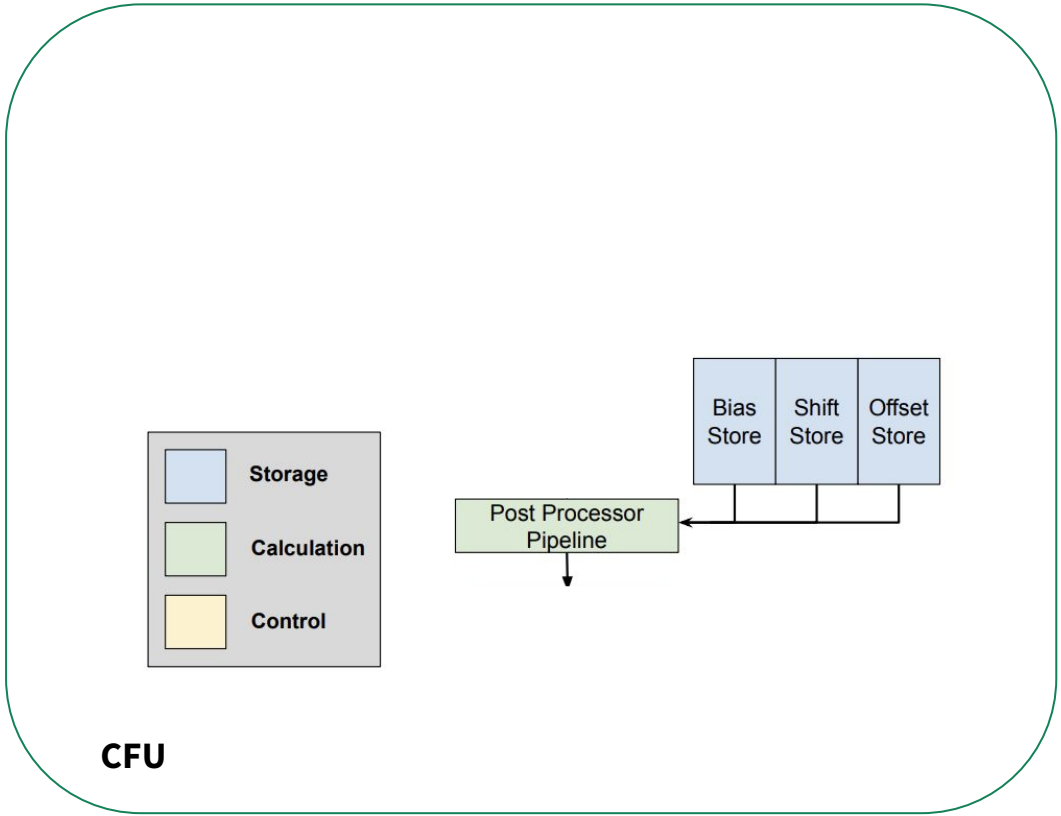
FPGA Acceleration for Image Classification

Start with Software Optimizations!

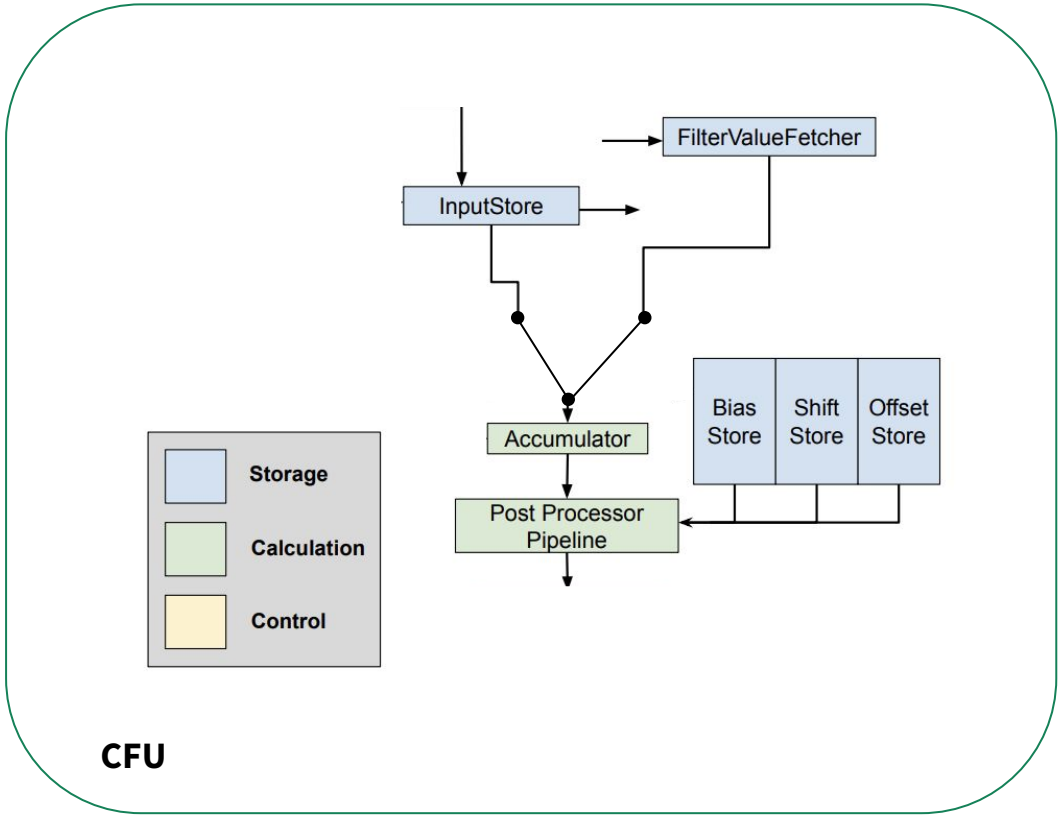


CFU

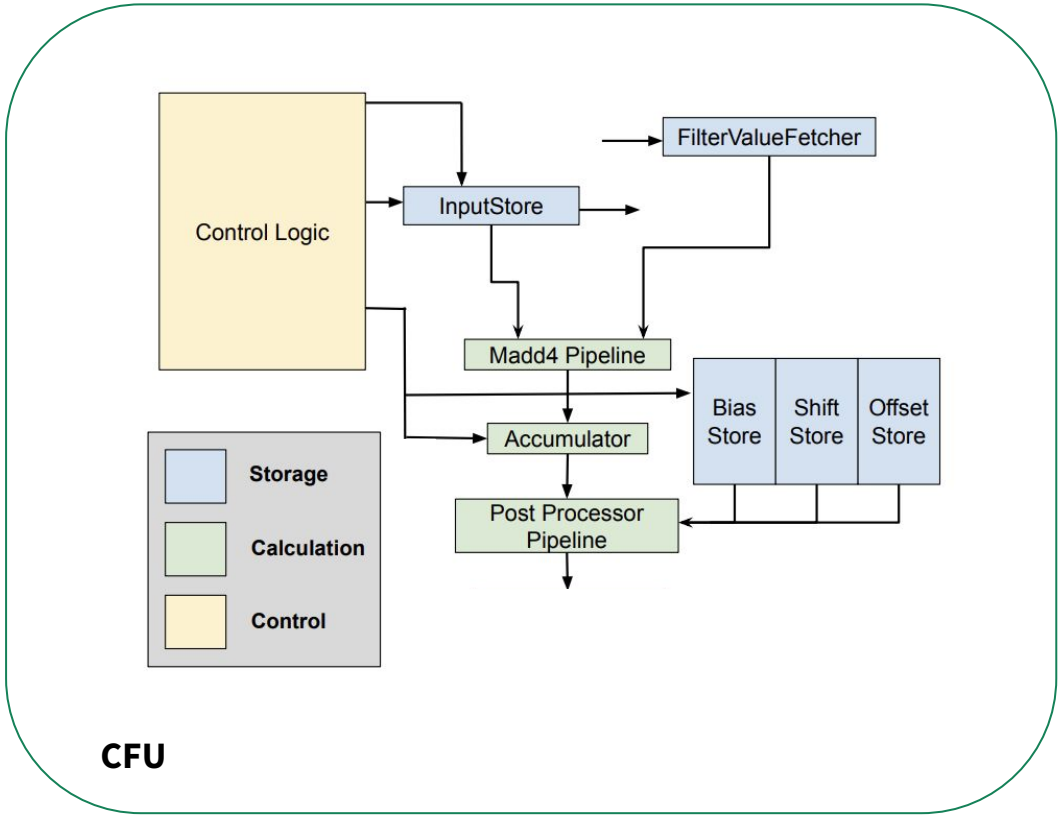
FPGA Acceleration for Image Classification



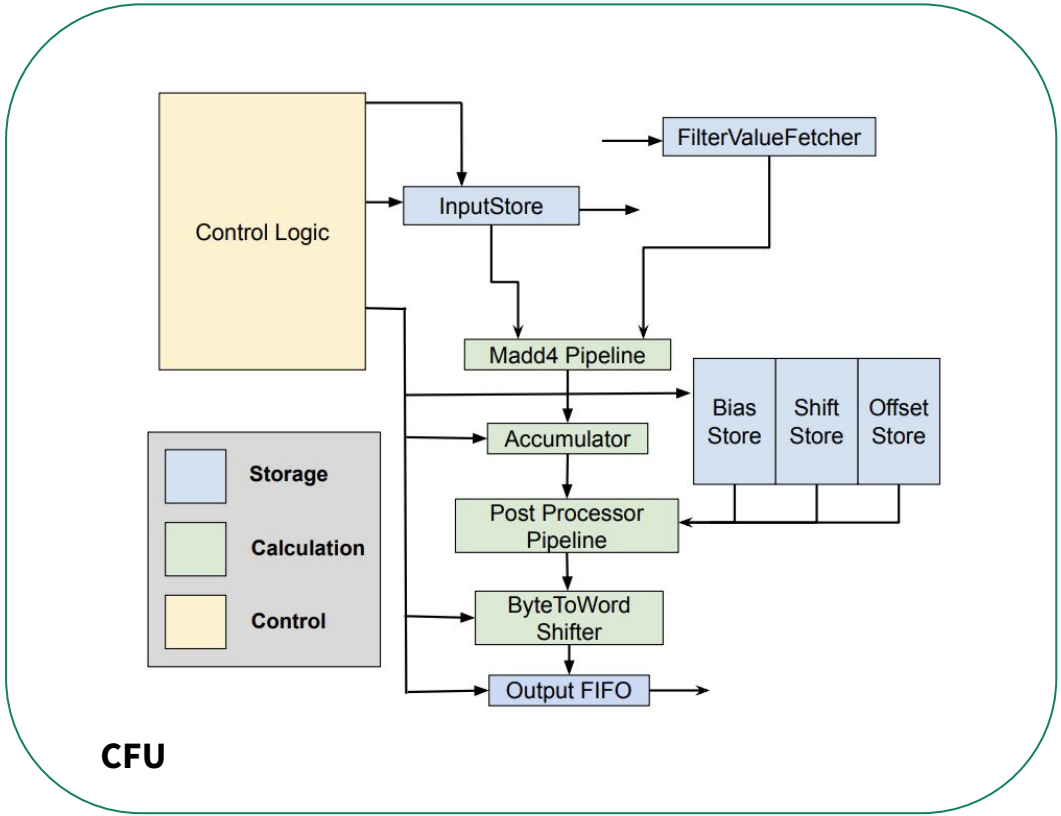
FPGA Acceleration for Image Classification



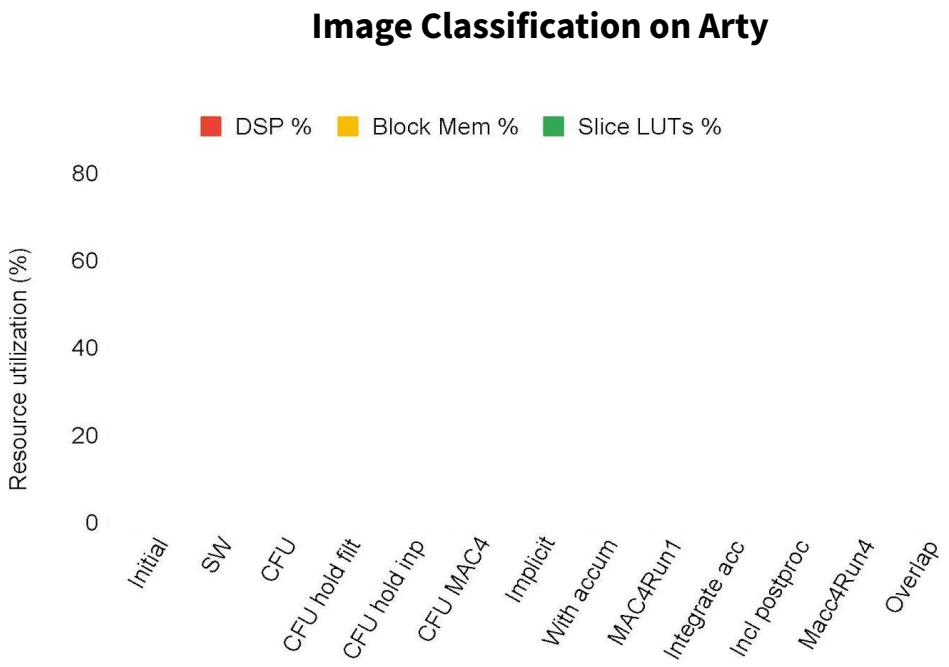
FPGA Acceleration for Image Classification



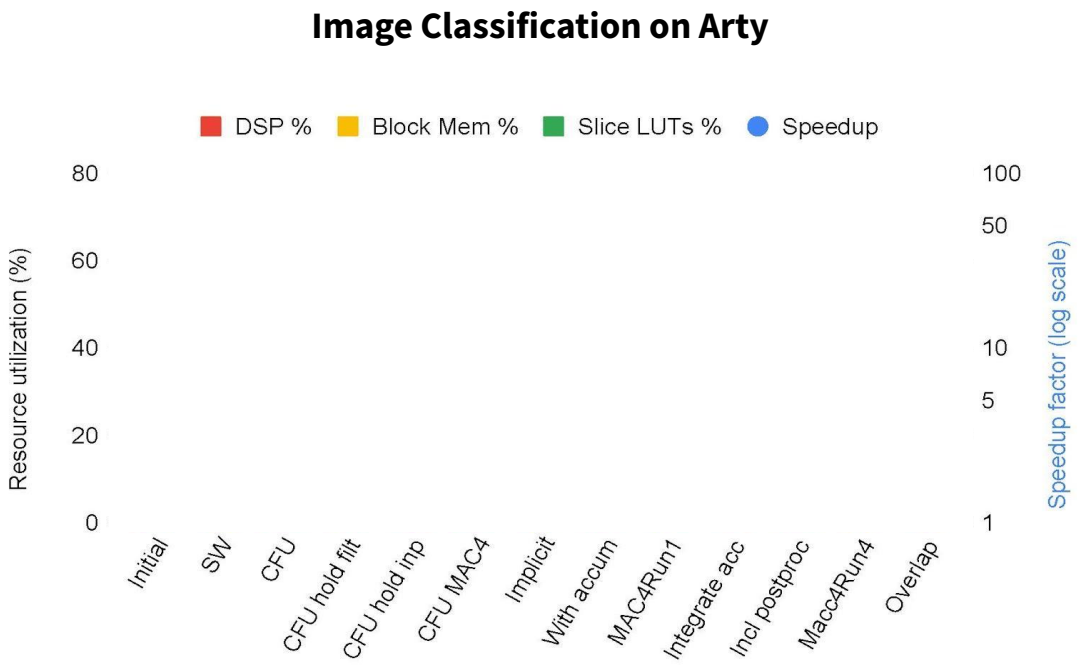
FPGA Acceleration for Image Classification



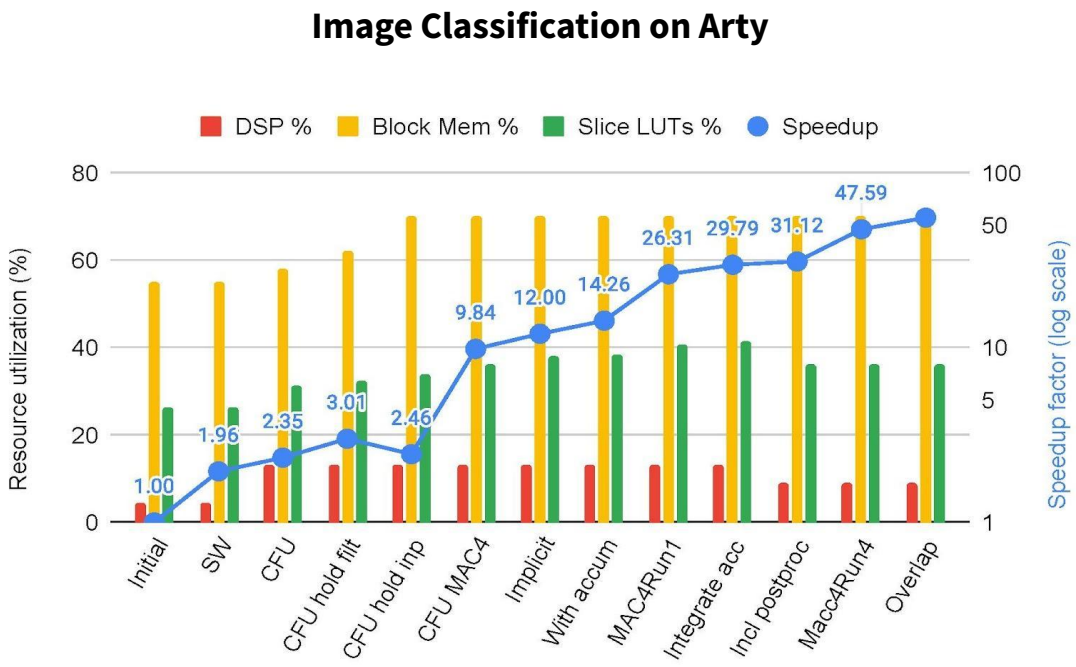
FPGA Acceleration for Image Classification



FPGA Acceleration for Image Classification

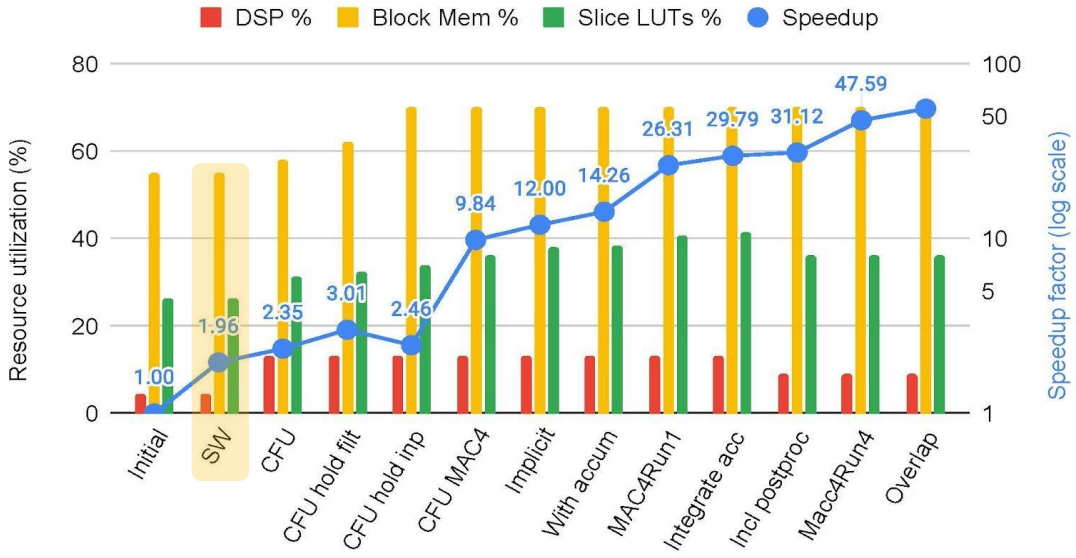


FPGA Acceleration for Image Classification



FPGA Acceleration for Image Classification

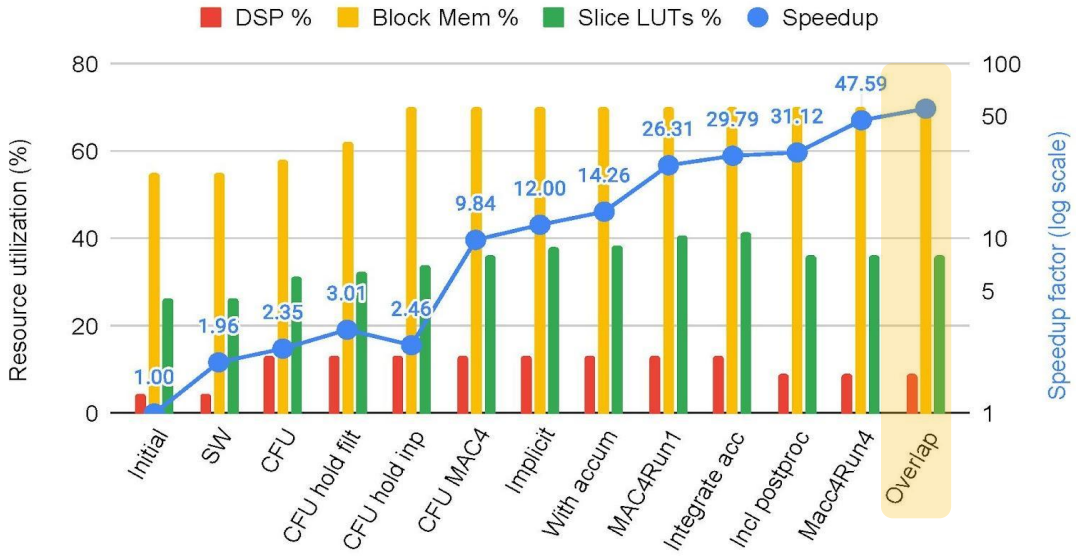
Image Classification on Arty



2x speedup from SW Optimizations

FPGA Acceleration for Image Classification

Image Classification on Arty



Total 55x speedup in 5 weeks

Human Presence Sensor

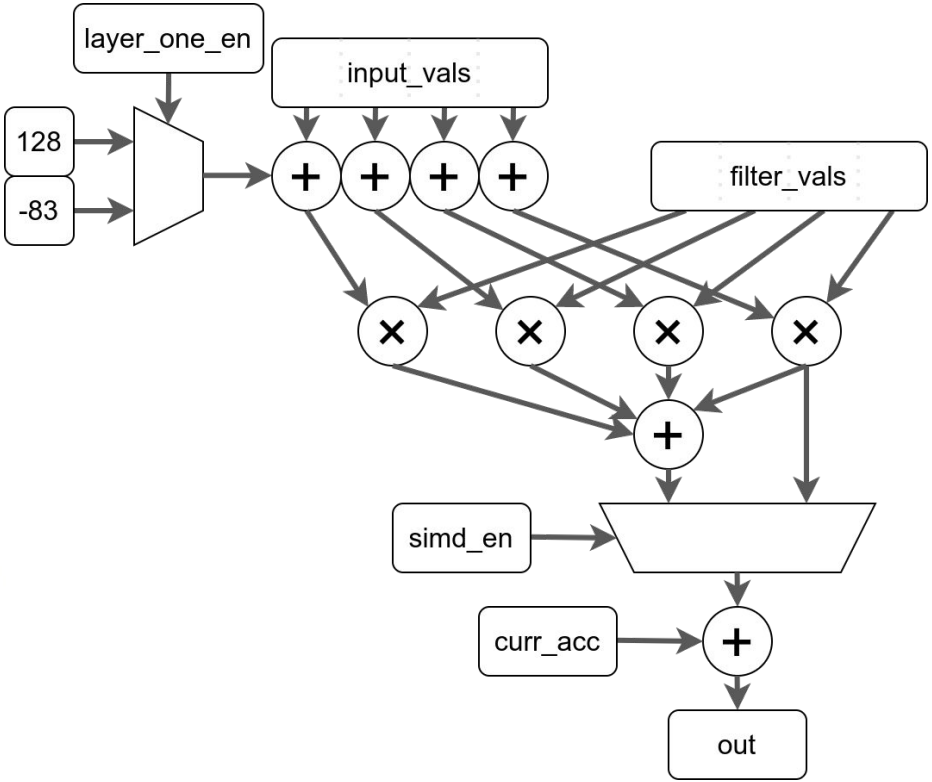
- In Chromebook:
  - An isolated camera+ML subsystem embedded in the display bezel

- User features:
  - Keep awake while present
  - Dim on leave
  - Wake on approach
  - Eavesdropper warning

FPGA Acceleration for Keyword Spotting

FOMU FPGA

FPGA Acceleration for Keyword Spotting



75× speedup on model inference

How it started:

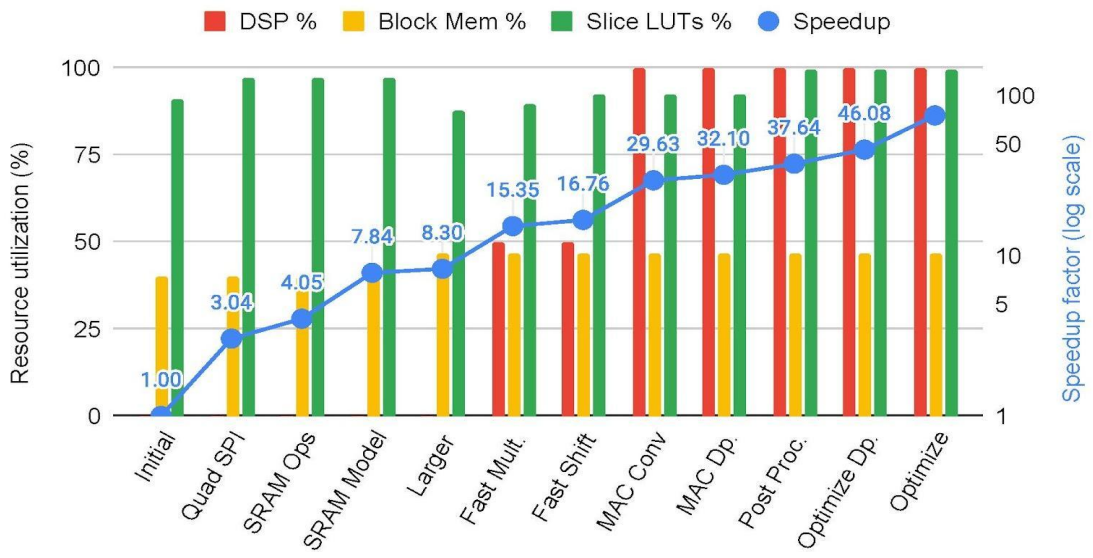
```
Running MLCommons Tiny V0.1 Keyword Spotting
Error_reporter OK!
Input: 490 bytes, 4 dims: 1 49 10 1


Tests for kws model
===================
 0: Run with "down" input
 1: Run with "go" input
 2: Run with "left" input
 g: Run golden tests (check for expected outputs)
 x: eXit to previous menu
kws>
```

How it's going:

```
Running MLCommons Tiny V0.1 Keyword Spotting
Error_reporter OK!
Input: 490 bytes, 4 dims: 1 49 10 1


Tests for kws model
===================
 0: Run with "down" input
 1: Run with "go" input
 2: Run with "left" input
 g: Run golden tests (check for expected outputs)
 x: eXit to previous menu
kws>
```

FPGA Acceleration for Keyword Spotting

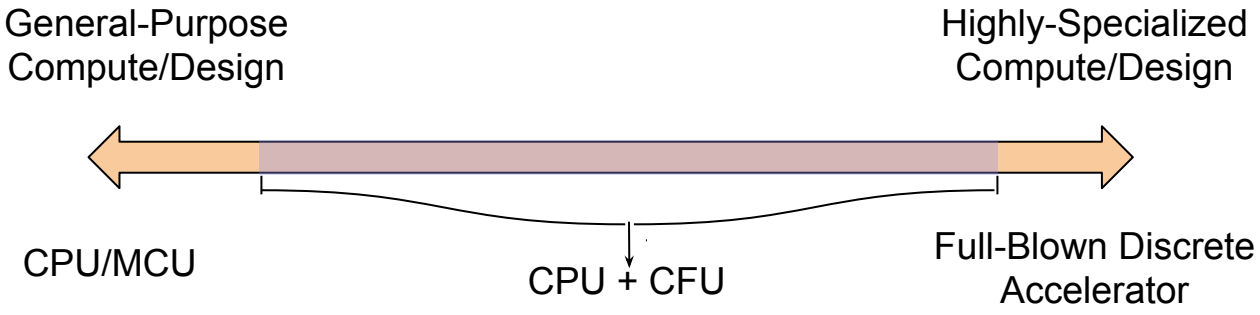
Keyword Spotting on FOMU



75x speedup in under 4 weeks

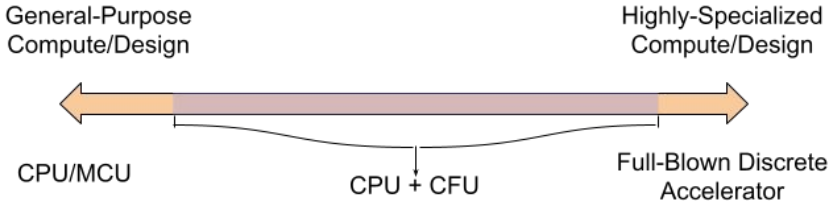
Design Space Exploration

(CFU) Accelerator vs (Soft) CPU

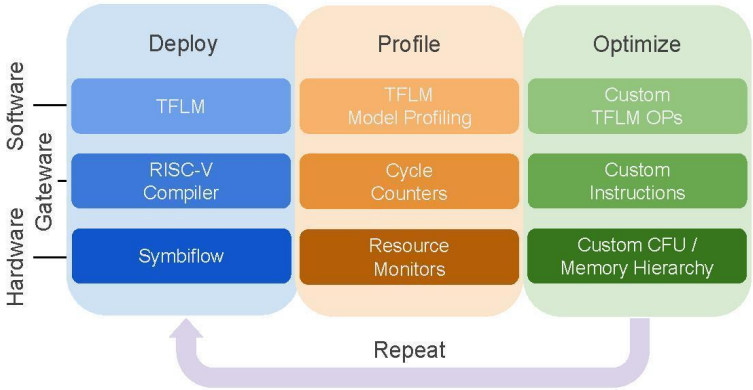


(Manual) Design Space Exploration

(CFU) Accelerator vs (Soft) CPU

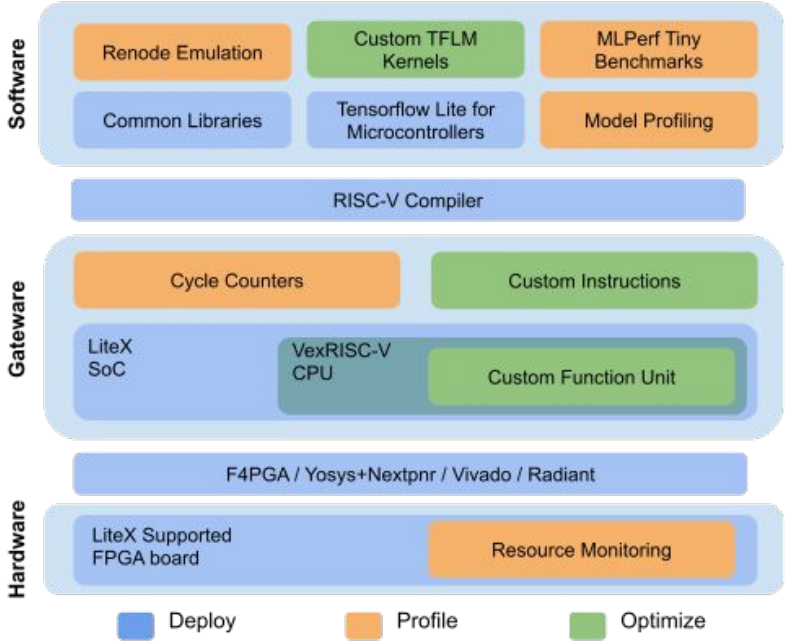


+



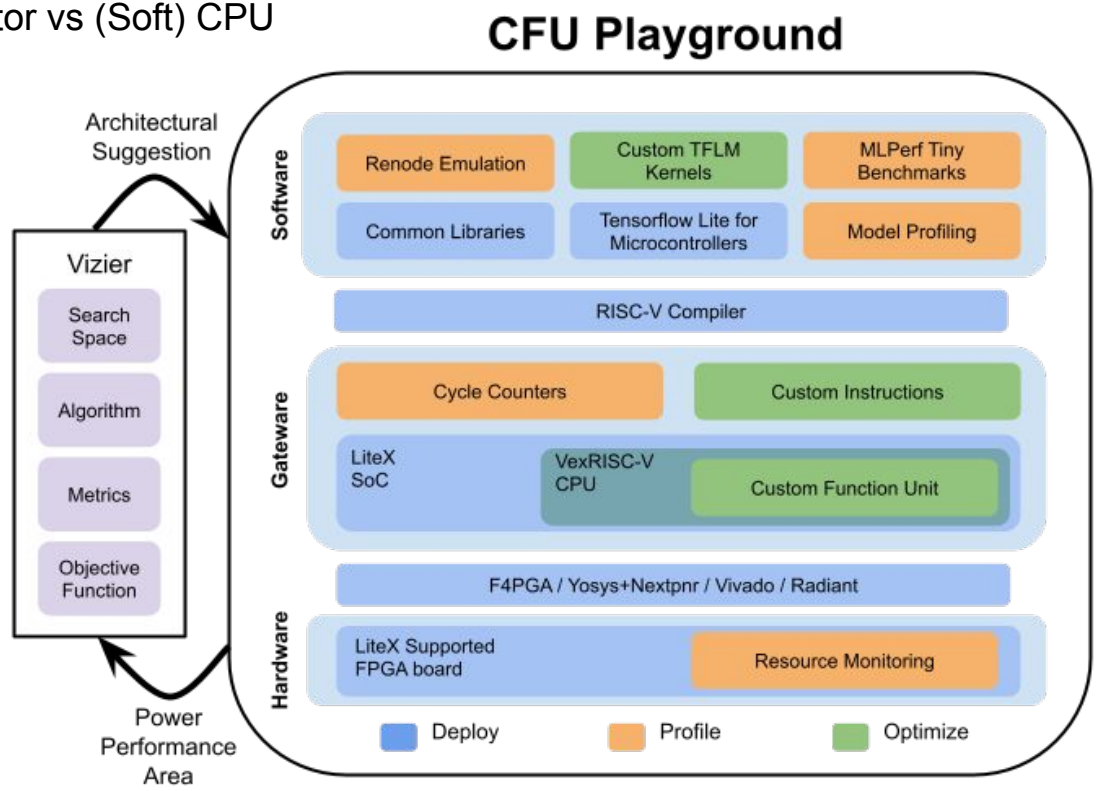
(Automated) Design Space Exploration

(CFU) Accelerator vs (Soft) CPU



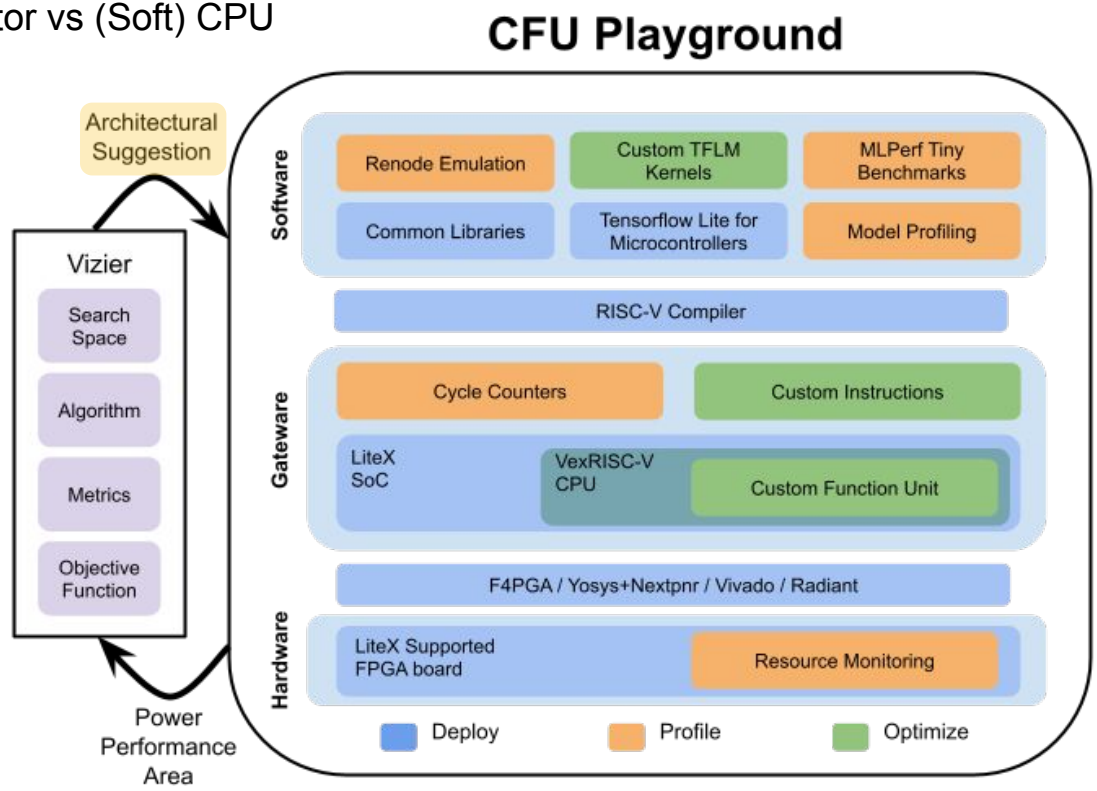
(Automated) Design Space Exploration

(CFU) Accelerator vs (Soft) CPU



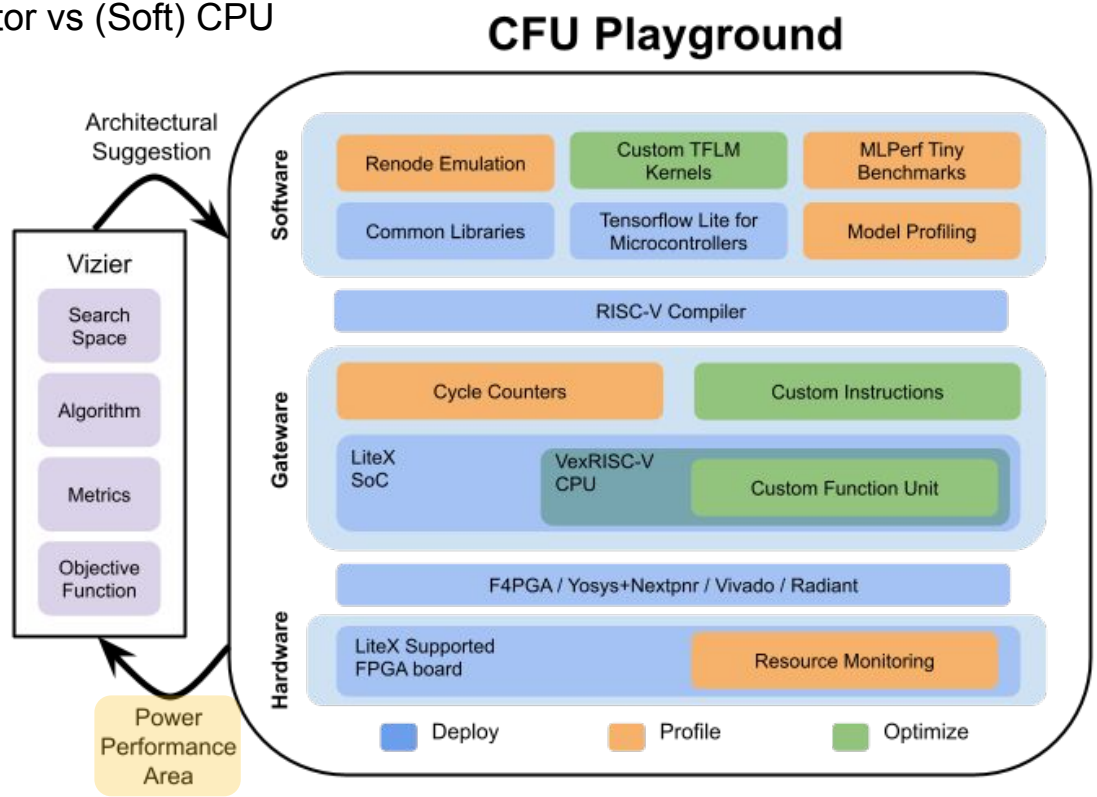
(Automated) Design Space Exploration

(CFU) Accelerator vs (Soft) CPU

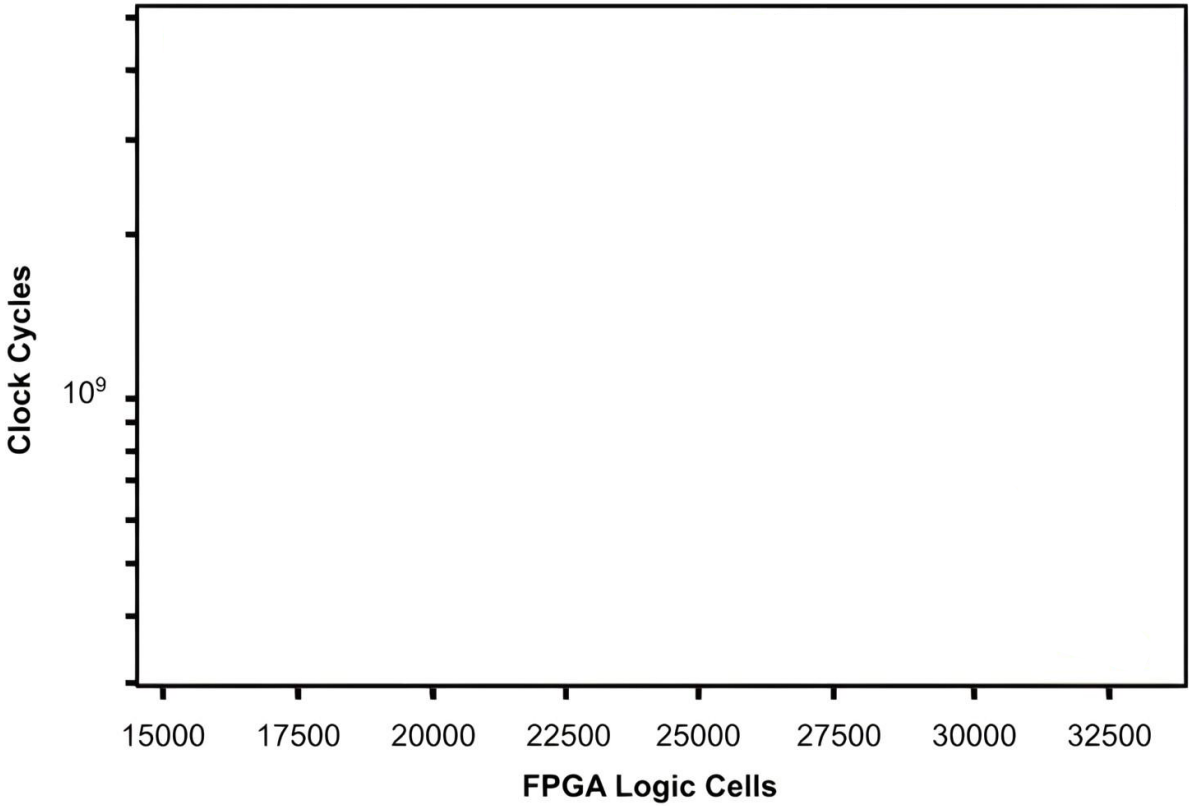


(Automated) Design Space Exploration

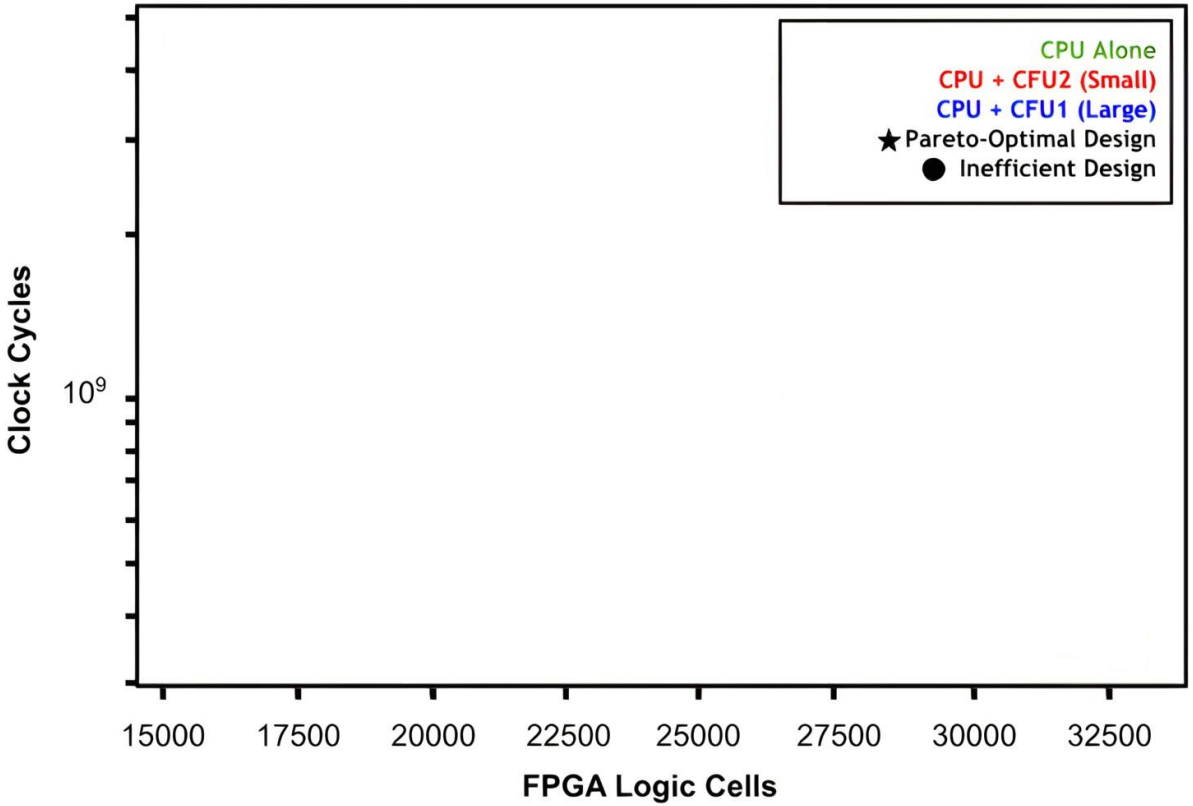
(CFU) Accelerator vs (Soft) CPU



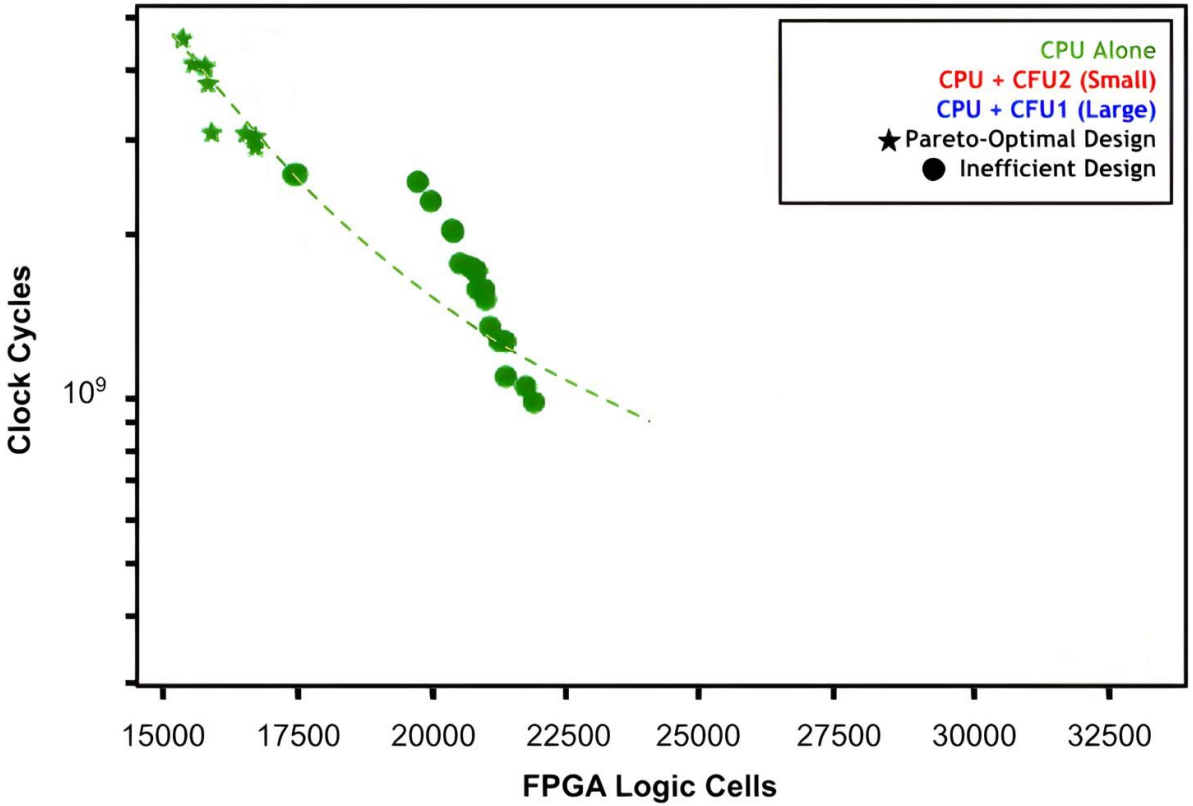
Design Space Exploration: CFU vs CPU



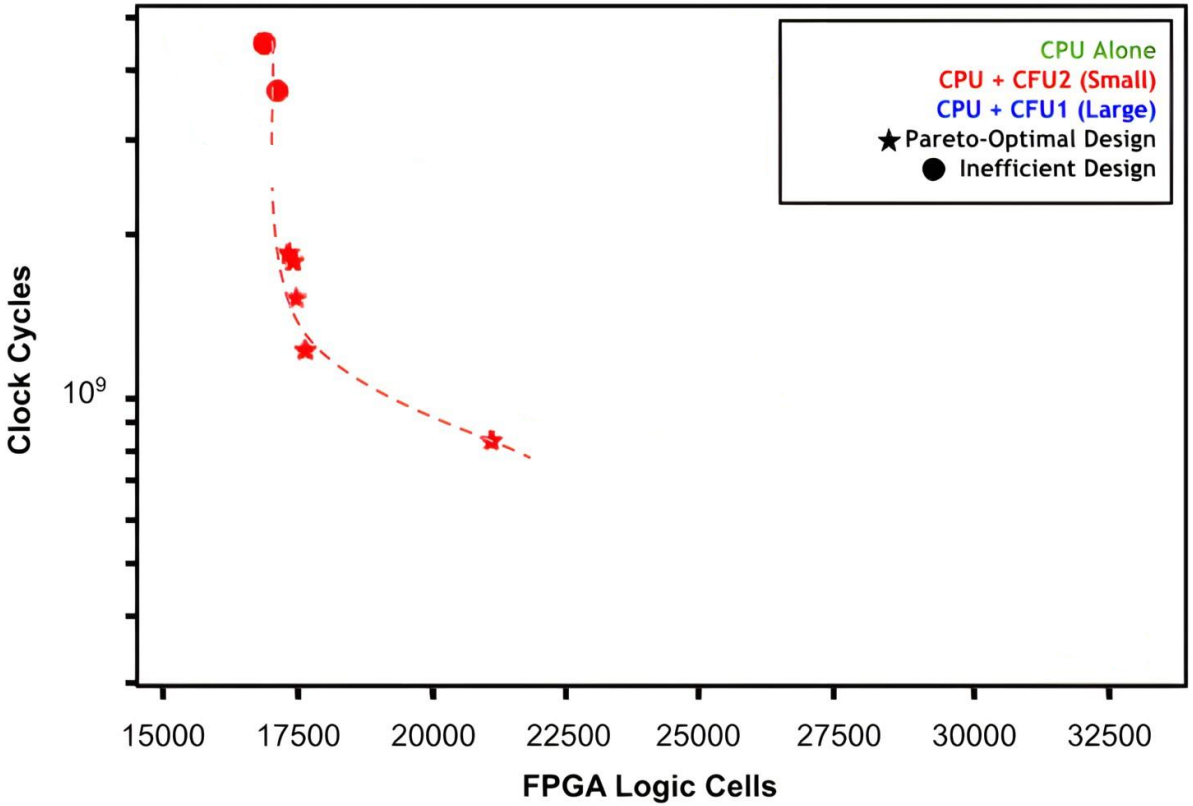
Design Space Exploration: CFU vs CPU



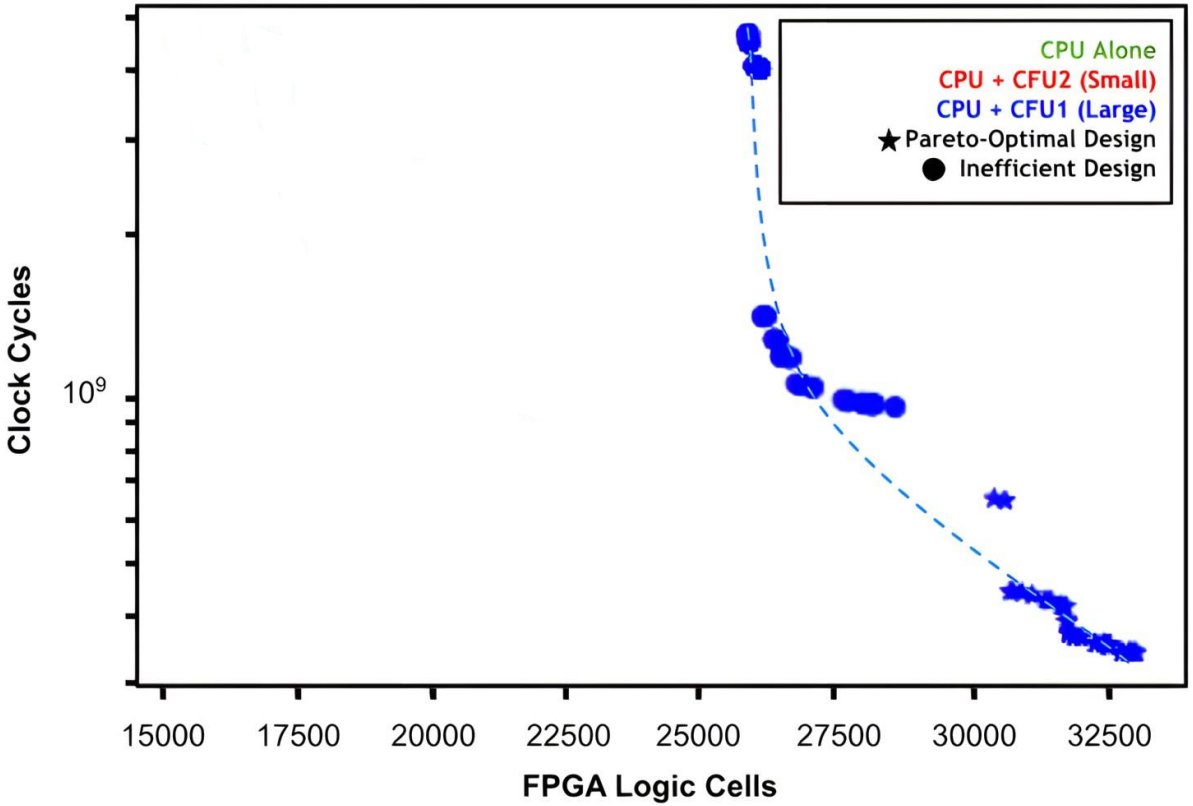
Design Space Exploration: CFU vs CPU



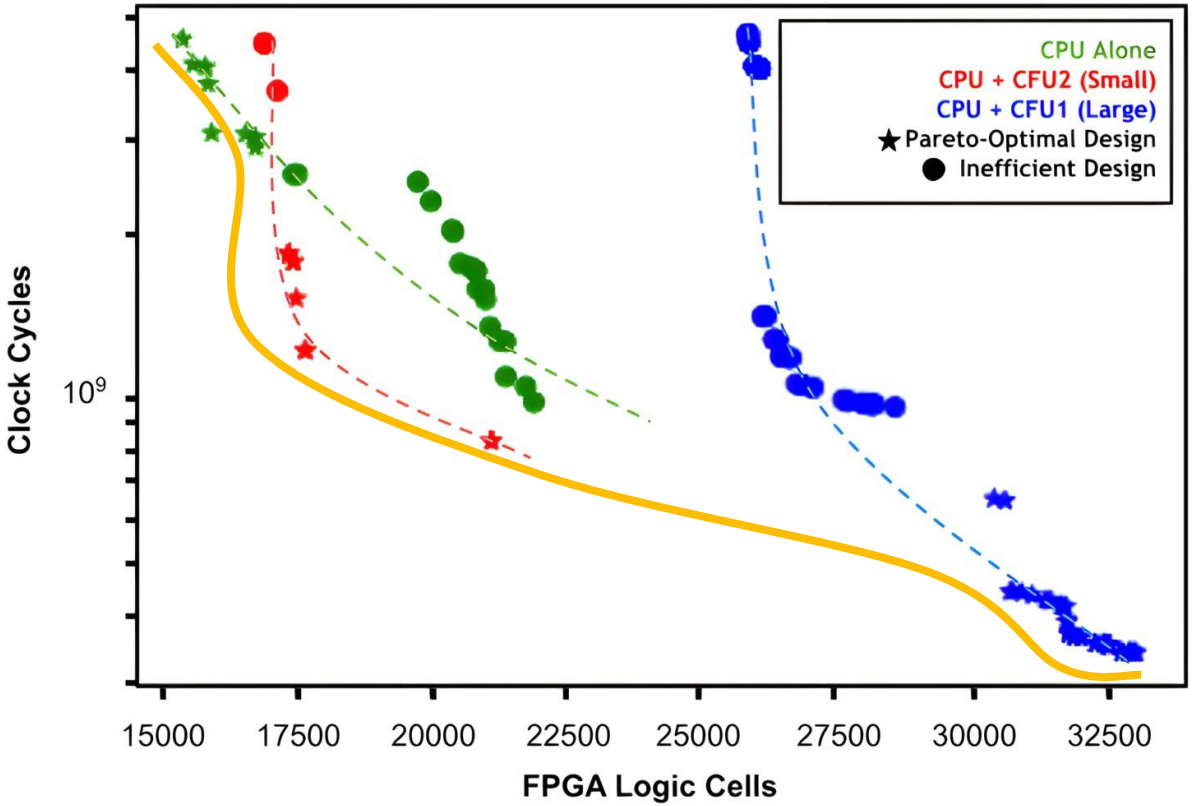
Design Space Exploration: CFU vs CPU



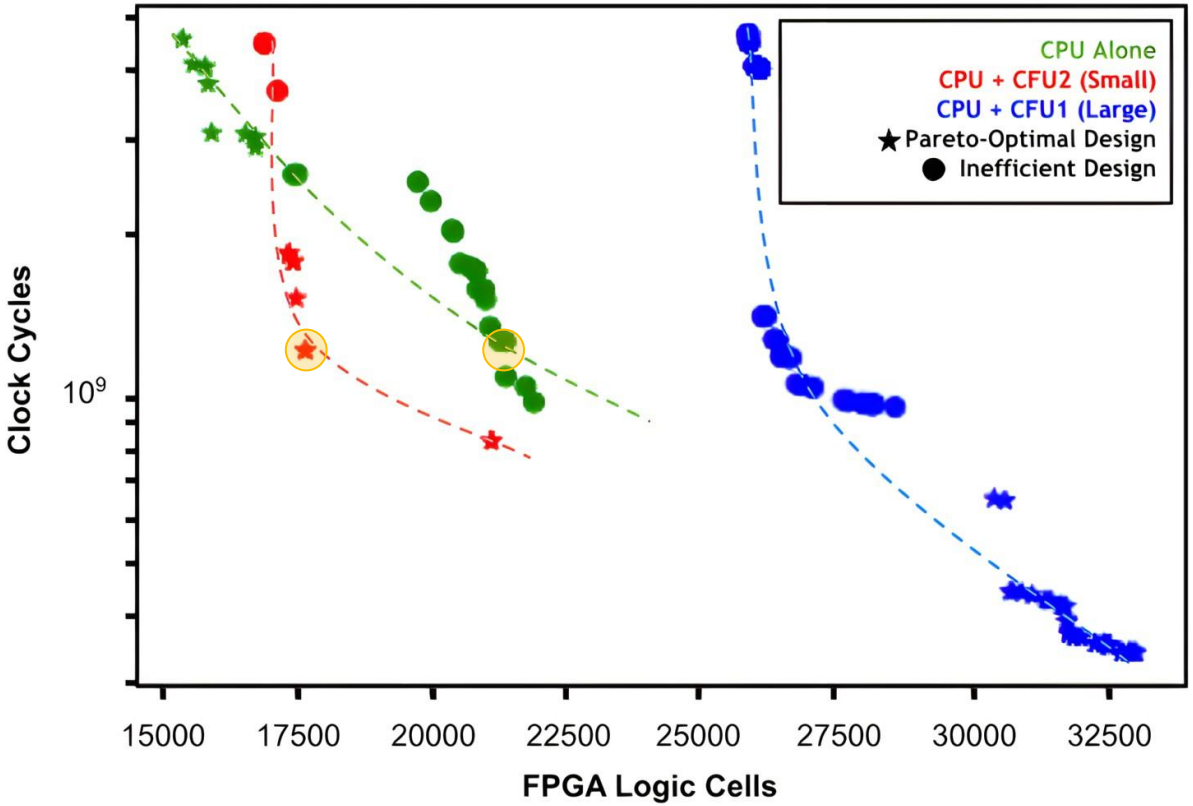
Design Space Exploration: CFU vs CPU



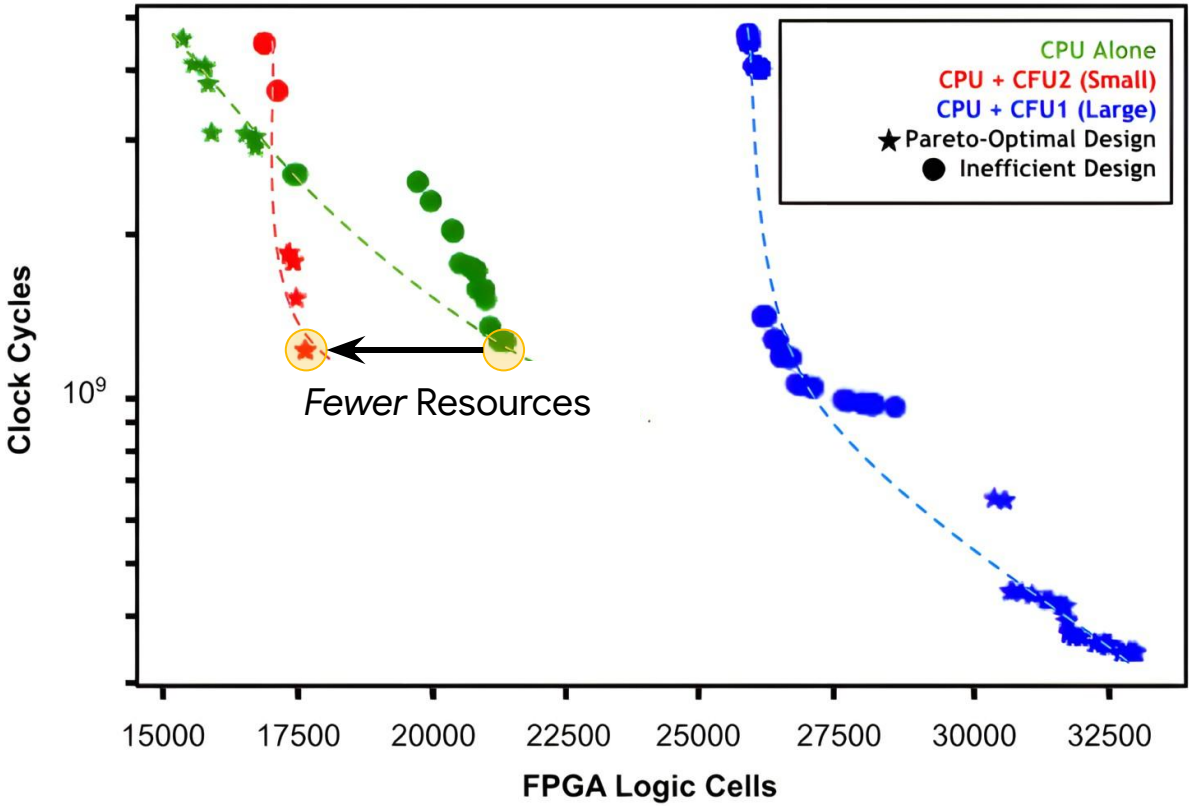
Design Space Exploration: CFU vs CPU



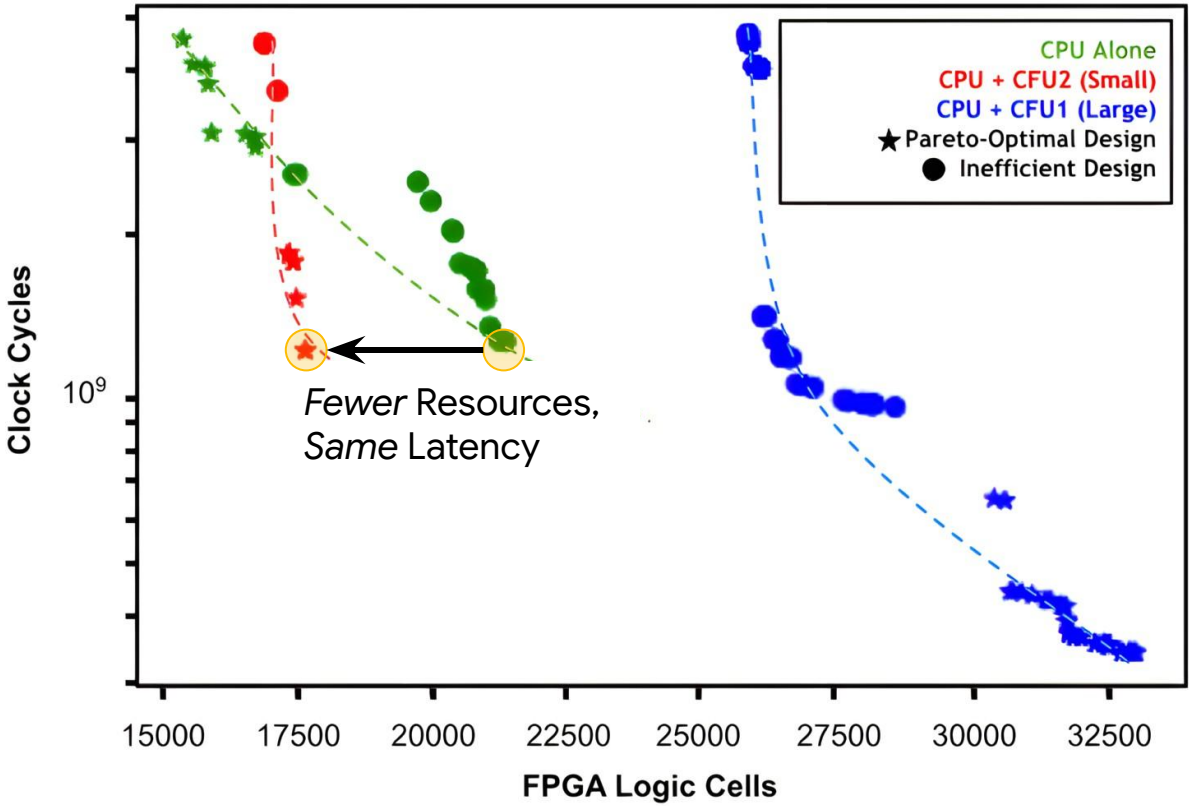
Design Space Exploration: CFU vs CPU



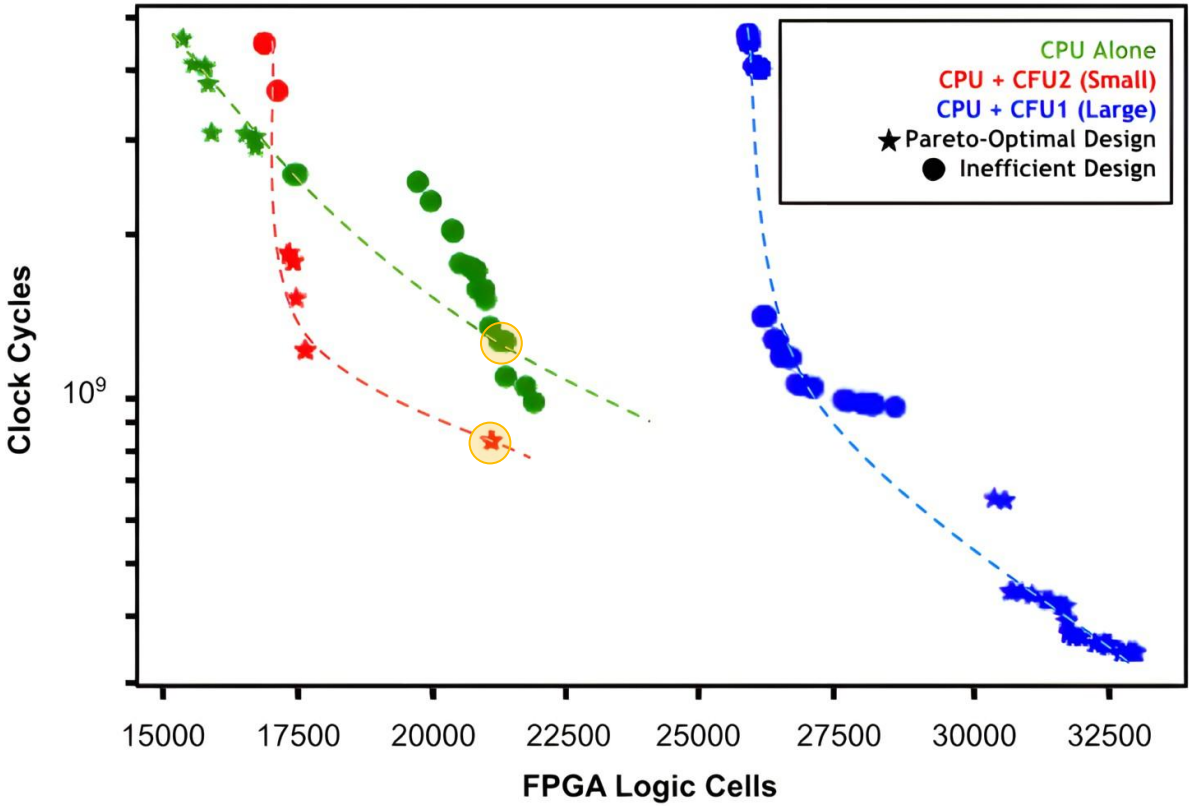
Design Space Exploration: CFU vs CPU



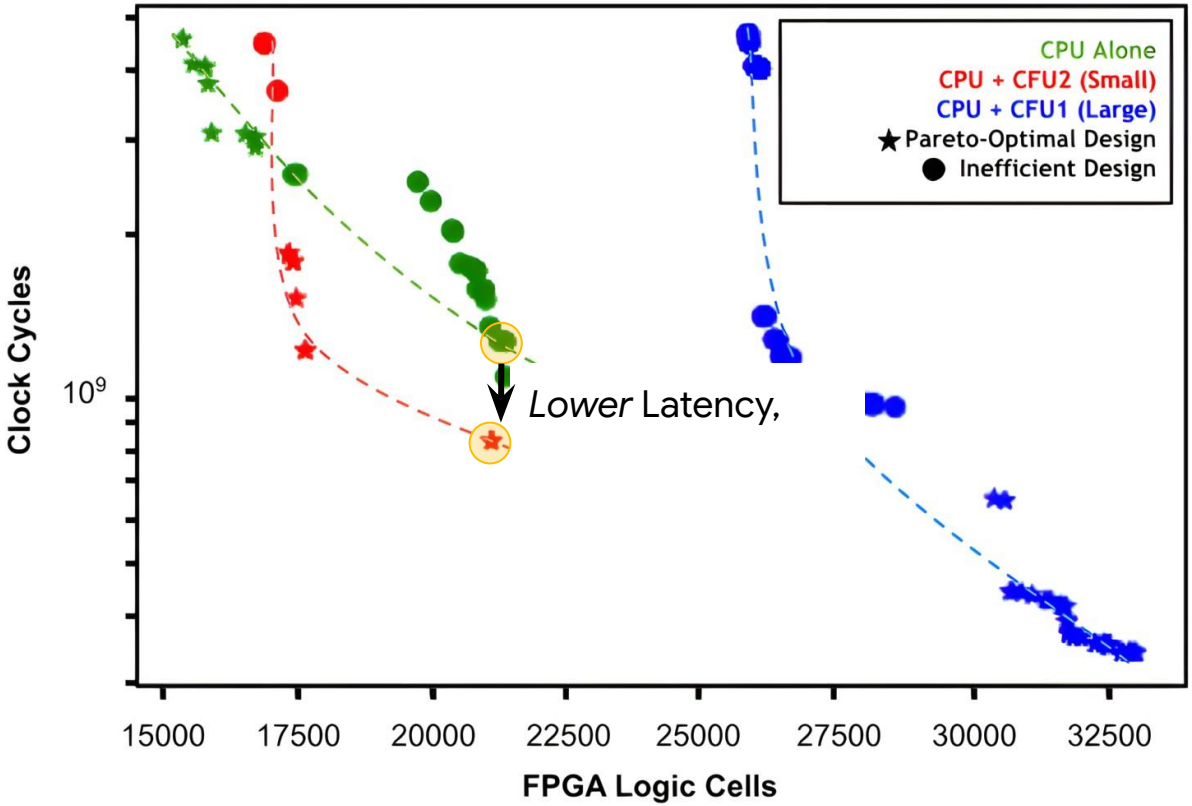
Design Space Exploration: CFU vs CPU



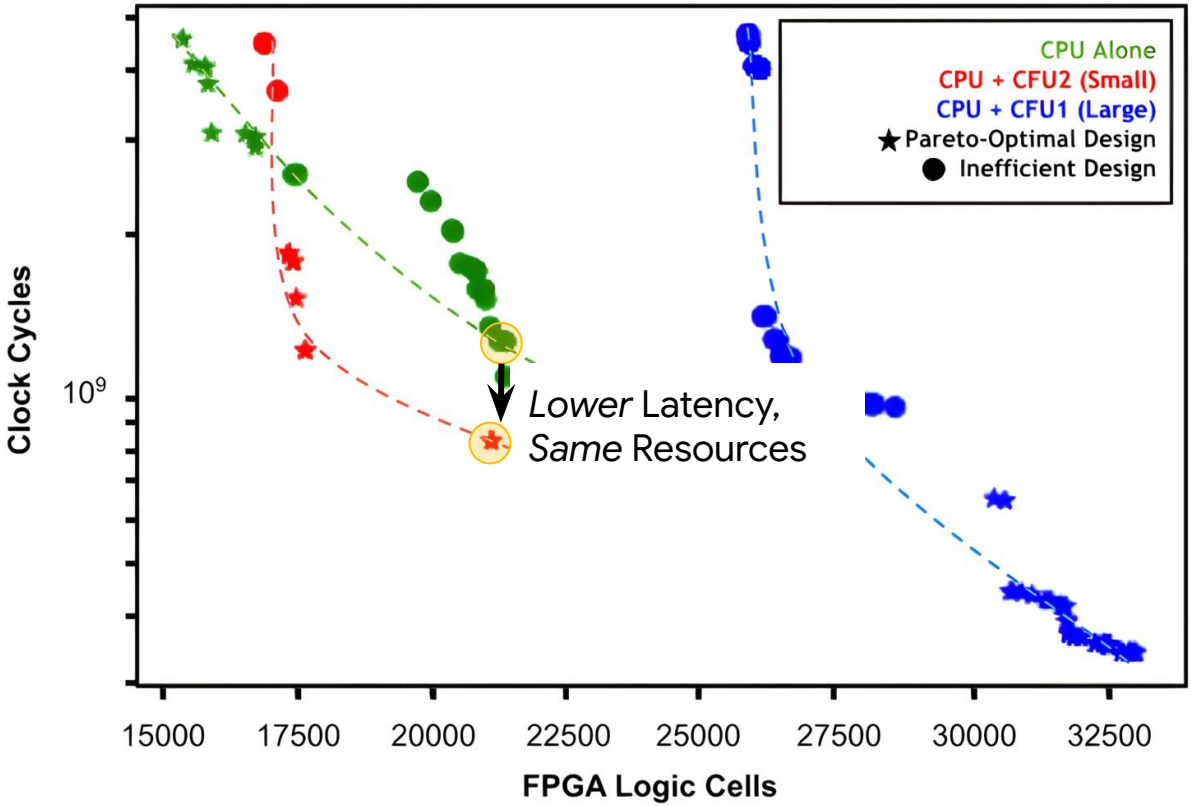
Design Space Exploration: CFU vs CPU



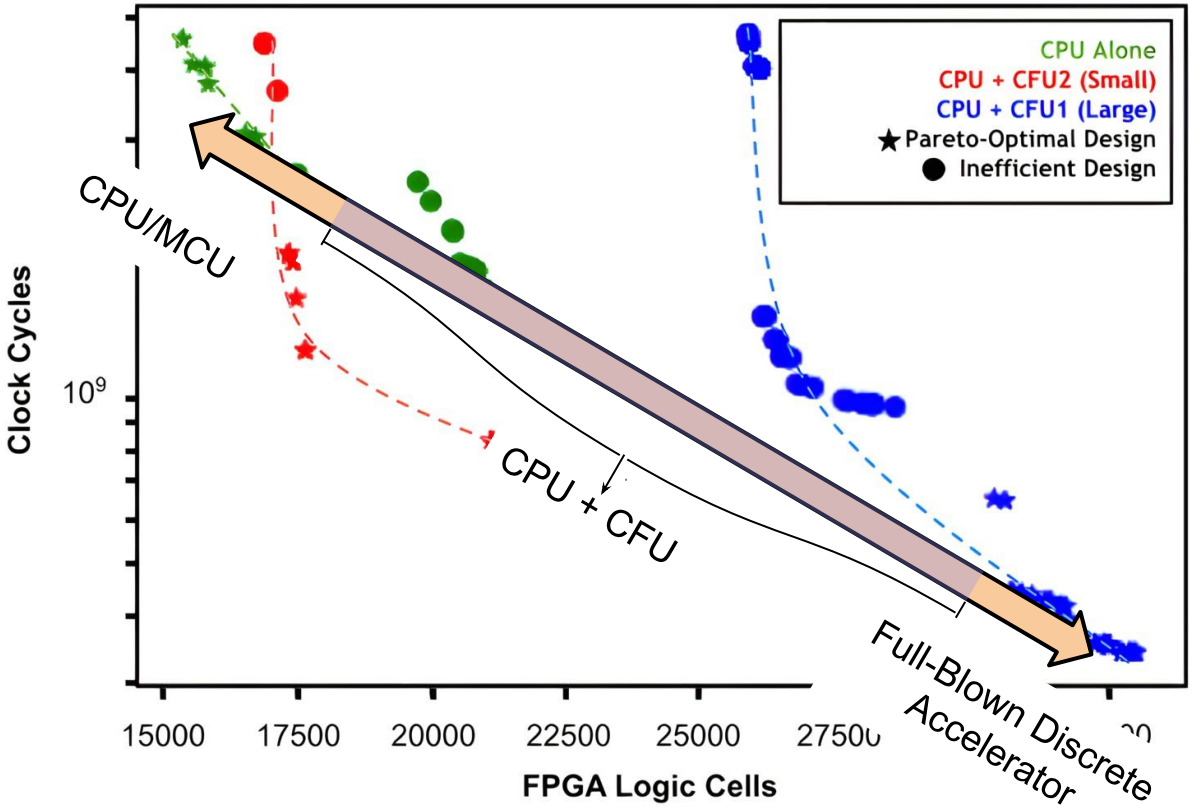
Design Space Exploration: CFU vs CPU



Design Space Exploration: CFU vs CPU



Design Space Exploration: CFU vs CPU



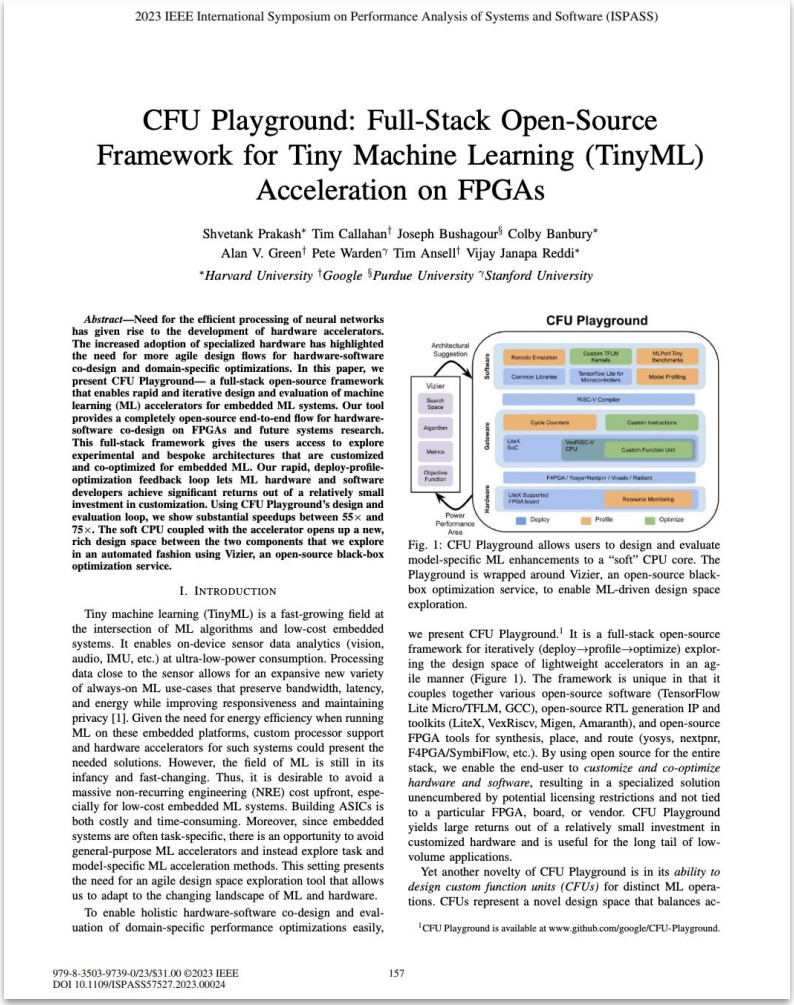
Key Takeaways

1. Full-stack framework that integrates open-source tools to facilitate community-driven research.

2. Agile methodology to iteratively design and evaluate tightly-coupled, bespoke TinyML accelerators.

3. Unique model-specific resource allocation trade-offs between CFU, CPU, and memory.

4. Automated design space exploration of the CPU paired with a CFU using Vizier.



2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)

CFU Playground: Full-Stack Open-Source Framework for Tiny Machine Learning (TinyML) Acceleration on FPGAs

Shvetank Prakash* Tim Callahan† Joseph Bushagour§ Colby Banbury*
Alan V. Green† Pete Wardenγ Tim Ansell† Vijay Janapa Reddi*
*Harvard University †Google §Purdue University γStanford University

***Abstract*—Need for the efficient processing of neural networks has given rise to the development of hardware accelerators. The increased adoption of specialized hardware has highlighted the need for more agile design flows for hardware-software co-design and domain-specific optimizations. In this paper, we present CFU Playground— a full-stack open-source framework that enables rapid and iterative design and evaluation of machine learning (ML) accelerators for embedded ML systems. Our tool provides a completely open-source end-to-end flow for hardware-software co-design on FPGAs and future systems research. This full-stack framework gives the users access to explore experimental and bespoke architectures that are customized and co-optimized for embedded ML. Our rapid, deploy-profile-optimization feedback loop lets ML hardware and software developers achieve significant returns out of a relatively small investment in customization. Using CFU Playground's design and evaluation loop, we show substantial speedups between 55× and 75×. The soft CPU coupled with the accelerator opens up a new, rich design space between the two components that we explore in an automated fashion using Vizier, an open-source black-box optimization service.**

I. Introduction

Tiny machine learning (TinyML) is a fast-growing field at the intersection of ML algorithms and low-cost embedded systems. It enables on-device sensor data analytics (vision, audio, IMU, etc.) at ultra-low-power consumption. Processing data close to the sensor allows for an expansive new variety of always-on ML use-cases that preserve bandwidth, latency, and energy while improving responsiveness and maintaining privacy [1]. Given the need for energy efficiency when running ML on these embedded platforms, custom processor support and hardware accelerators for such systems could present the needed solutions. However, the field of ML is still in its infancy and fast-changing. Thus, it is desirable to avoid a massive non-recurring engineering (NRE) cost upfront, especially for low-cost embedded ML systems. Building ASICs is both costly and time-consuming. Moreover, since embedded systems are often task-specific, there is an opportunity to avoid general-purpose ML accelerators and instead explore task and model-specific ML acceleration methods. This setting presents the need for an agile design space exploration tool that allows us to adapt to the changing landscape of ML and hardware.

To enable holistic hardware-software co-design and evaluation of domain-specific performance optimizations easily,

Fig. 1: CFU Playground allows users to design and evaluate model-specific ML enhancements to a "soft" CPU core. The Playground is wrapped around Vizier, an open-source black-box optimization service, to enable ML-driven design space exploration.

we present CFU Playground.[1] It is a full-stack open-source framework for iteratively (deploy→profile→optimize) exploring the design space of lightweight accelerators in an agile manner (Figure 1). The framework is unique in that it couples together various open-source software (TensorFlow Lite Micro/TFLM, GCC), open-source RTL generation IP and toolkits (LiteX, VexRiscv, Migen, Amaranth), and open-source FPGA tools for synthesis, place, and route (yosys, nextpnr, F4PGA/SymbiFlow, etc.). By using open source for the entire stack, we enable the end-user to *customize and co-optimize hardware and software*, resulting in a specialized solution unencumbered by potential licensing restrictions and not tied to a particular FPGA, board, or vendor. CFU Playground yields large returns out of a relatively small investment in customized hardware and is useful for the long tail of low-volume applications.

Yet another novelty of CFU Playground is in its *ability to design custom function units (CFUs)* for distinct ML operations. CFUs represent a novel design space that balances ac-

[1] CFU Playground is available at www.github.com/google/CFU-Playground.

979-8-3503-9739-0/23/$31.00 ©2023 IEEE
DOI 10.1109/ISPASS57527.2023.00024

Paper Discussions

Why are we reading these papers?

What are the important things we learn from these papers?

What can we compare and contrast with these papers?

Guest Speaker

Hardik Sharma

Hardik is the Director of Hardware Engineering at Bigstream. His research interests are domain specific hardware architectures for accelerating machine learning. He led the development of the first open-source FPGA-based hardware acceleration stack for DNNs at Georgia Tech.

Website

Google Scholar