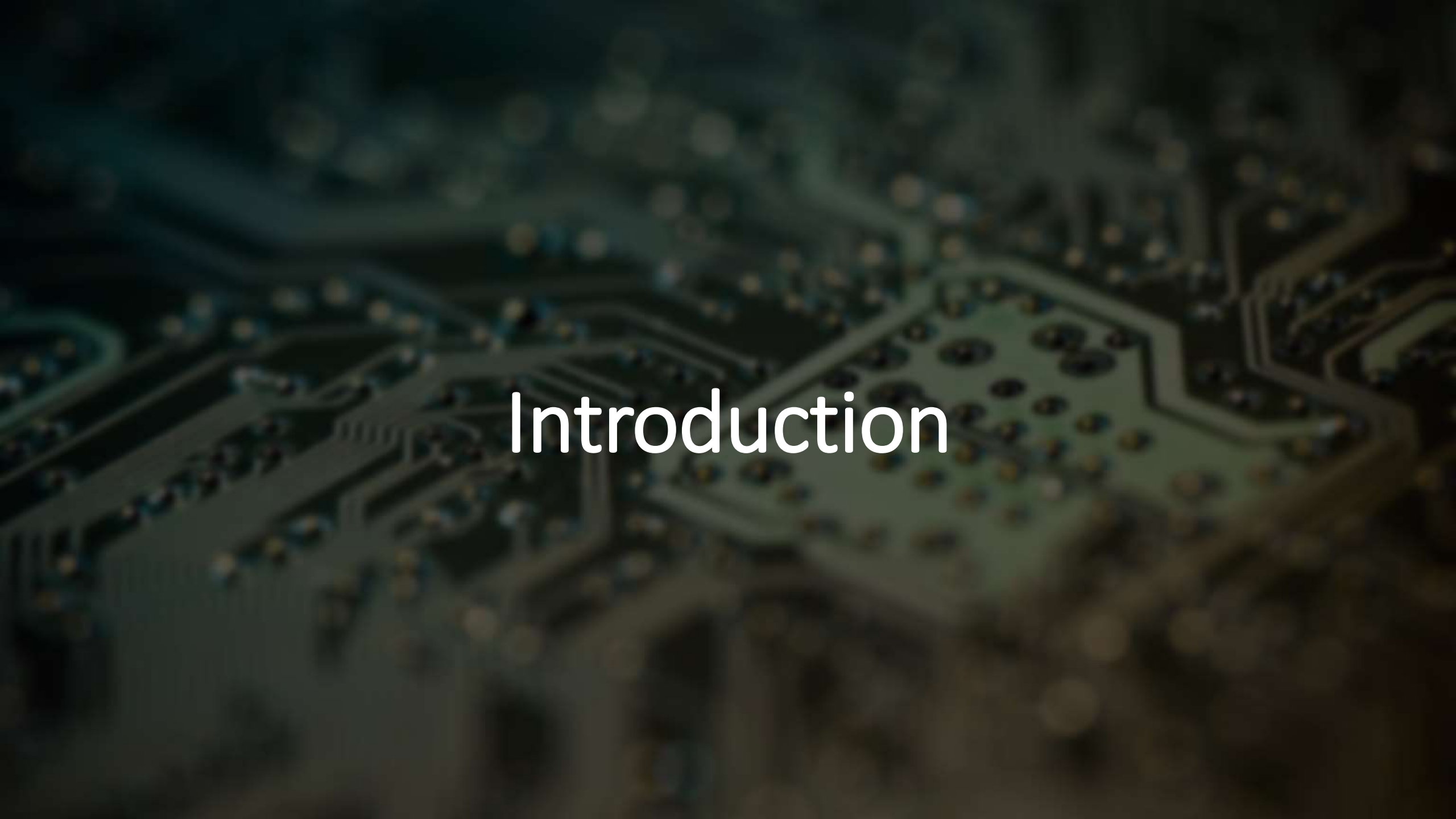
PULP-TrainLib: Enabling On-Device Training for RISC-V Multi-Core MCUs through Performance-Driven Autotuning

Davide Nadalini[1,2], Manuele Rusci[2], Giuseppe Tagliavini[3], Leonardo Ravaglia[2], Luca Benini[2,4], and Francesco Conti[2]

[1]DAUIN, Politecnico di Torino, Torino, Italy, davide.nadalini@polito.it
[2]DEI, University of Bologna, Bologna, Italy, {d.nadalini,manuele.rusci,luca.benini,f.conti}@unibo.it
[3]DISI, University of Bologna, Bologna, Italy, giuseppe.tagliavini@unibo.it
[4]IIS, ETH Zurich, Zurich, Switzerland, lbenini@iis.ee.ethz.ch

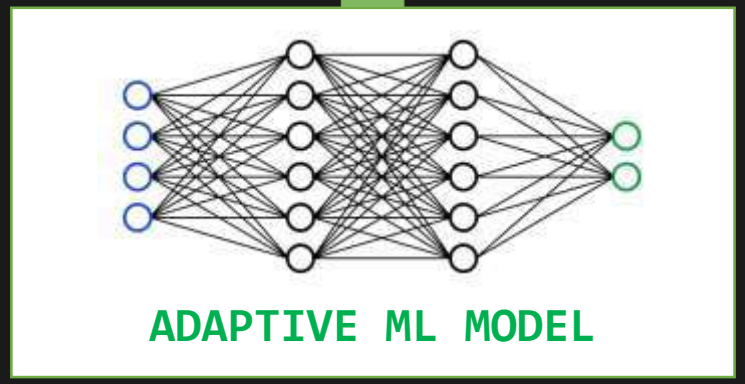
OVERVIEW

- Intro
  - Motivation (On-Device Training)
  - Contribution
- PULP-TrainLib
  - PULP-TrainLib
  - AutoTuner
- Results
  - In-Depth Analysis of AutoTuner’s Results
  - Training TinyML Models
  - Comparison with the State-Of-Art
- Conclusion
- Q. & A.


Introduction

MOTIVATION

MINIVAN!



ADAPTIVE ML MODEL

ML model's **parameters** updated based on environment-specific data

STANDARD DNN TRAINING

DEPLOY

SERVER

TRAIN

DATASET

Wireless to Cloud
Human Intervention
...

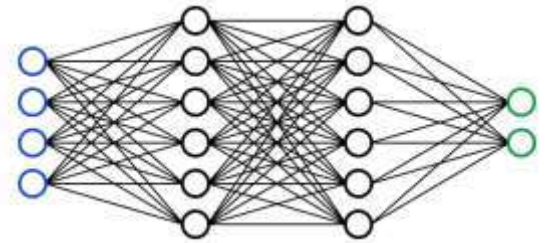
DATA COLLECTION

PRONE TO PRIVACY ISSUES, LIMITED SCALING, INCREASED POWER BUDGET

ON-DEVICE TRAINING

UPDATE

RETRAIN ON-DEVICE



ADAPTIVE ML MODEL

- Continual Learning
- Transfer Learning
- Federated Learning

NEW SAMPLE

BACKPROPAGATION -BASED

TRAINING ON MCUs

SERVER

EDGE GPU

MCU

> 100 W | 1-50 W | < 1 W

High Power
High Performance

Low Power

Ultra-Low Power

State of the Art On-Device Training on MCUs:
AIfES for Arduino (Fraunhofer IMS)

Suitable for IoT Edge Devices

[1]https://github.com/Fraunhofer-IMS/AIfES_for_Arduino

CONTRIBUTION

PULP-TrainLib, the first Open-Source[1] training library for multicore RISC-V MCUs

AutoTuner, a HW-in-the-loop tool to optimize PULP-TrainLib

PULP-TrainLib + AutoTuner on an **8-Core PULP** Platform

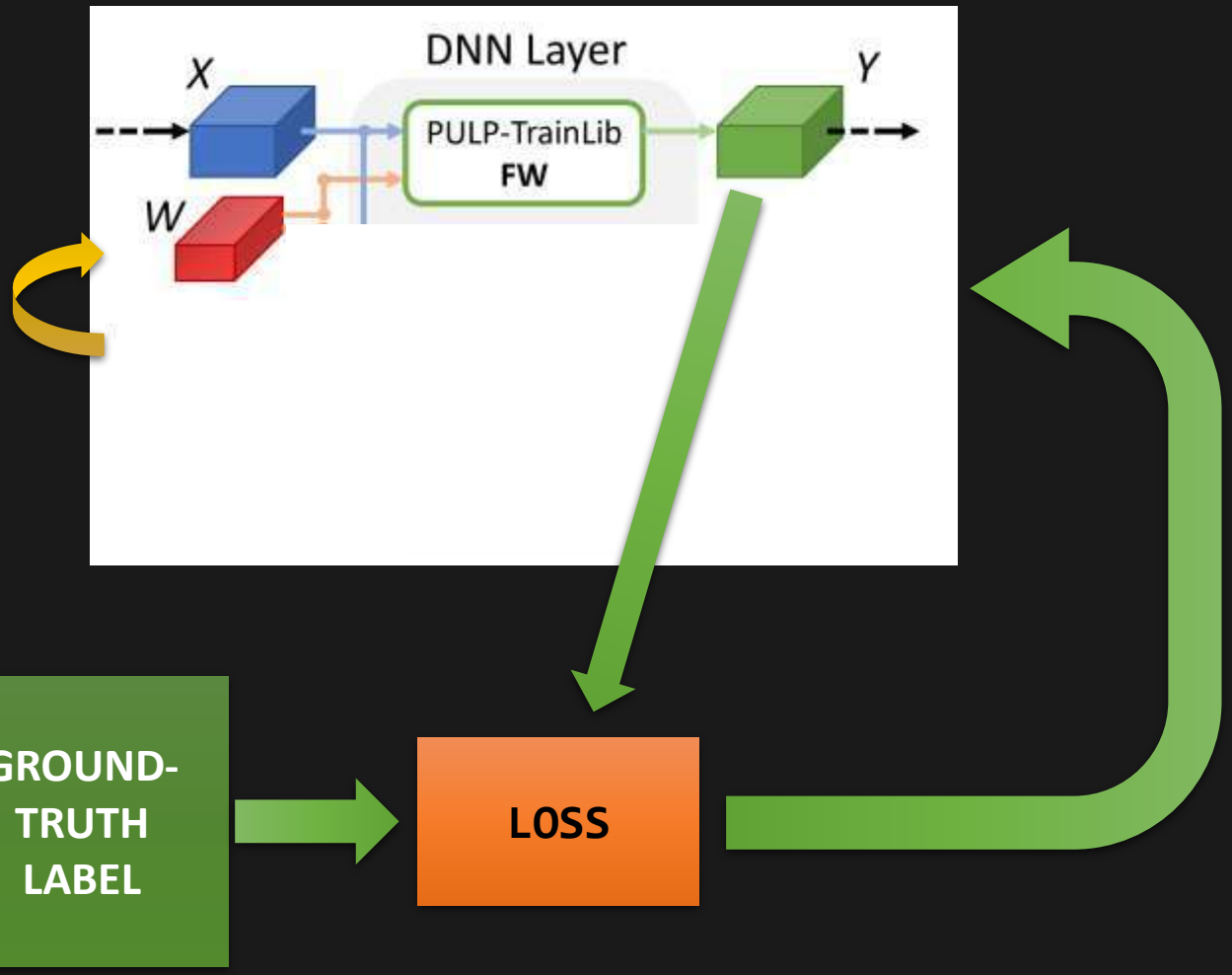
[1]https://github.com/pulp-platform/pulp-trainlib

PULP-TrainLib

TRAINING WITH BACKPROPAGATION

GOAL: Update Deep Learning Model's W

- ❑Forward Pass
- ❑Loss Function
- ❑Weight Gradient
- ❑Input Gradient
- ❑Weight Update



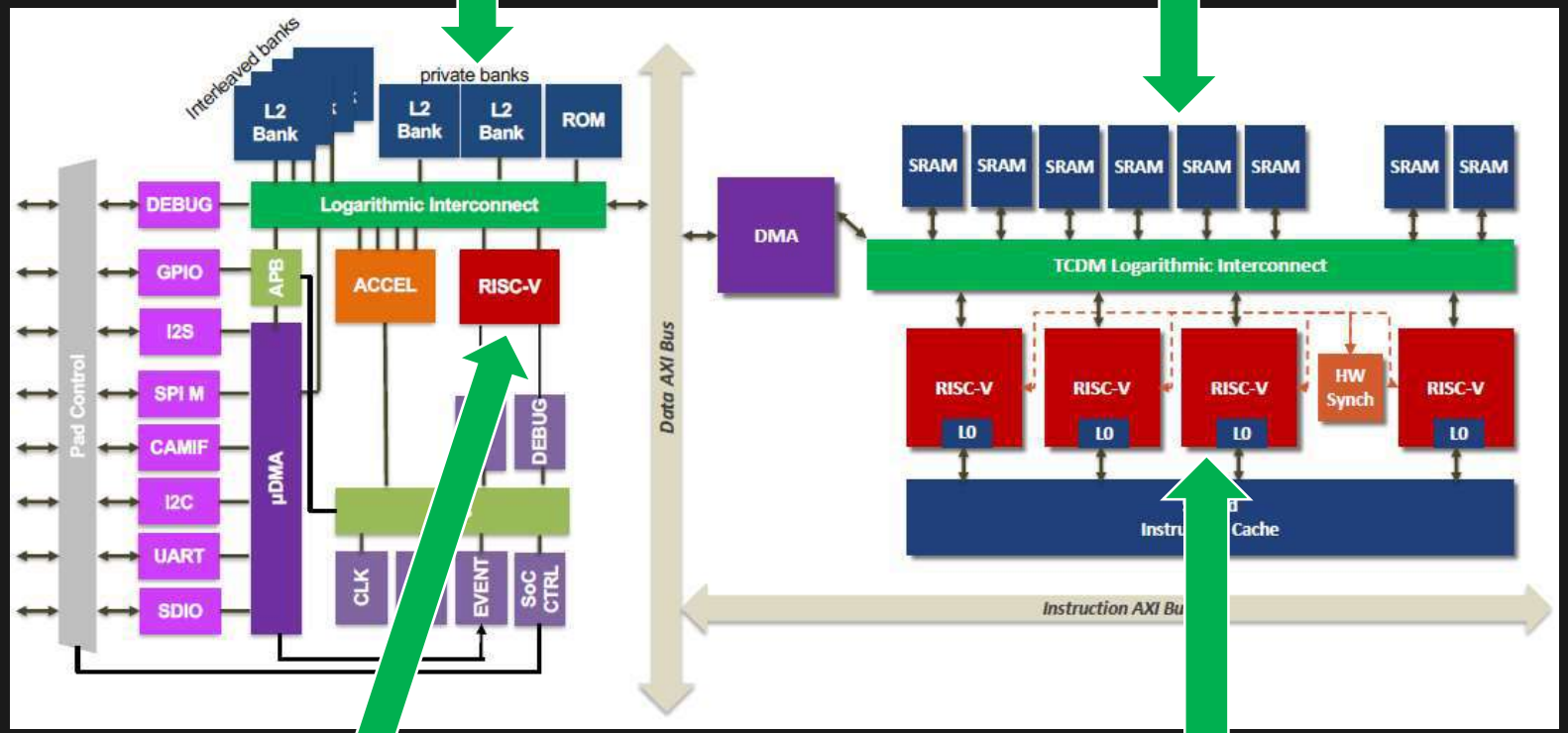
THE PULP PLATFORM

PULP (Parallel Ultra-Low-Power): computational platform for **energy-efficient** and **scalable edge computing**, based on **RISC-V** cores.

EMBODIMENTS:
SoC (MCU), GVSoC

512 kB - 2 MB
SRAM (L2)

64 kB - 128 kB
SRAM (L1)



RISC-V MCU
(Fabric Controller)

Cluster of 8 RISC-V Cores
RV32IMFC-Xpulp

PULP-TRAINLIB

The First SW DNN (Deep Neural Network) training library for multicore RISC-V MCUs

Loss Functions

Optimizers

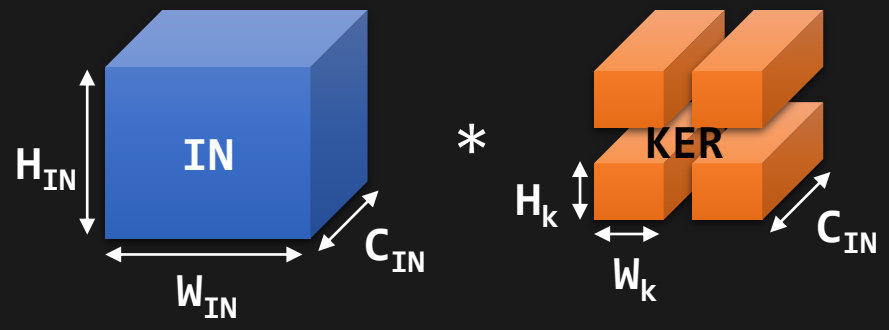
```
function train_conv2d(X, W, Y, label) {
  pulp_conv2d_fp32_fw_cl(X, W, Y);
  dY = loss(Y, label);
  pulp_conv2d_fp32_bw_input_grads_cl(dY, W, dX);
  pulp_conv2d_fp32_bw_param_grads_cl(dY, X, dW);
  W = update(W, dW);
}
```

Forward and **Backward** Training Primitives

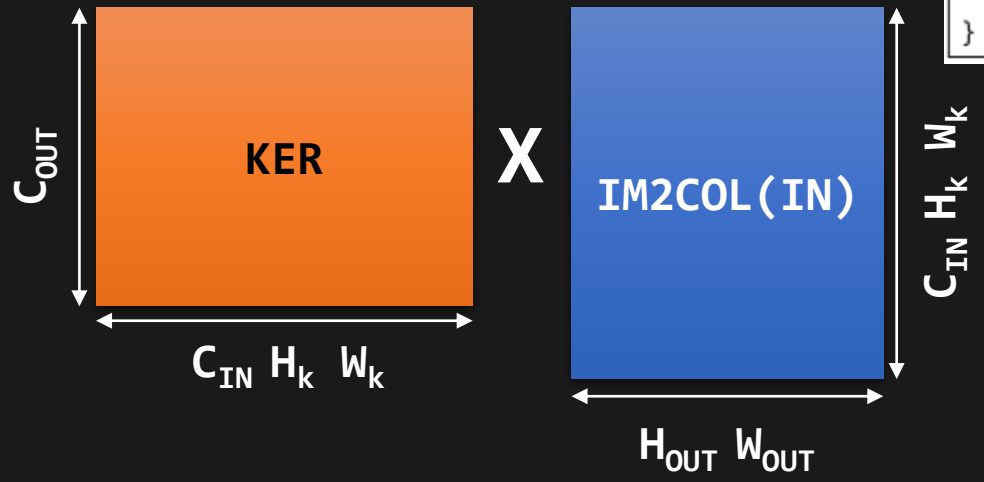
PERFORMANCE TUNABLE

Floating Point FP32 SUPPORTED LAYERS: Fully-Connected, DepthWise Separable Convolution, 2D Convolution

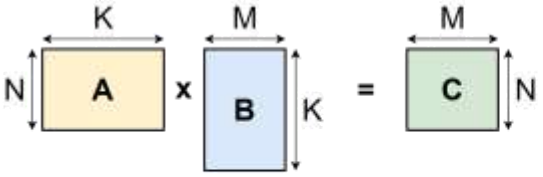
PULP-TRAINLIB'S PRIMITIVES



CONV2D FORWARD



UNROLLING + **PARALLELIZATION**




```
function mm (A, B, C, sizes) {
   for i in N:
      for j in M:
         for k in K:
            // C[i, j] += A[i, k] * B[k, j]
            ld      rsA, Aik
            ld      rsB, Bkj
            fmac    rdC, rsA, rsB
         st      rdC, addr
}
```

(a)

```
function mm_unroll_2x1 (A, B, C, sizes) {
   for i in N/2:
      for j in M:
         for k in K:
            // C[i, j] += A[i, k] * B[k, j]
            // C[i+1, j] += A[i+1, k] * B[k, j]
            ld      rsA1, Ai1k
            ld      rsA2, Ai2k
            ld      rsB, Bkj
            fmac    rdC1, rsA1, rsB
            fmac    rdC2, rsA2, rsB
         st      rdC1
         st      rdC2
}
```

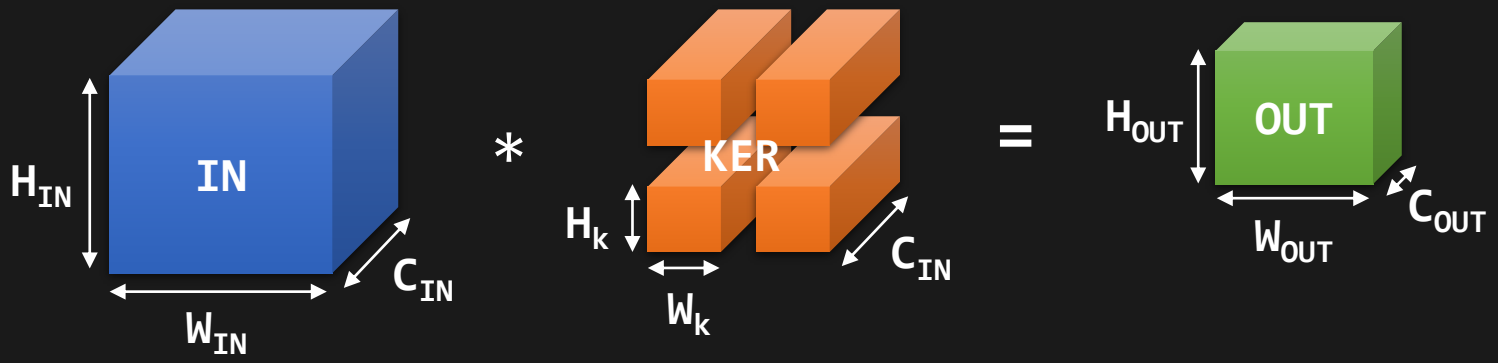
Split on N_{CORES}

(b)

Table 1: PULP-TrainLib's Optimized MM Kernels

MM Type	Unrolling Factor	Parallelism	Description
mm	–	N or M	Naive MM
mm_uJ	$J = 2$	N or M	Unroll J inner products in K
mm_unroll_1xV	$U = 2, 4, 8$	N or M	Unroll U columns of C
mm_unroll_Ux1	$V = 2, 4, 8$	N or M	Unroll V rows of C
mm_unroll_UxV	$U, V = 2, 4$	N or M	Unroll U,V rows and columns of C

DEPLOYING A DNN TRAINING TASK



DEPLOYMENT ON MCU

TENSOR SIZE MAY EXCEED L1

TILING

TILE SIZES NOT MULTIPLE OF UNROLLING FACTORS

OPTIMAL MM

AUTOTUNER

PRE-DEPLOYMENT, OFFLINE TOOL

EXTERNAL TILER

PULP SoC RESOURCES
L1: 64 kB
N_{CORES}: 8

POINTWISE CONVOLUTION
STEP: WG-BW
IN: 64x25x5
KER: 64x1x1x64
OUT: 64x25x5

80.38 kB

TileList

TileShape 1
IN: 64x25x5
KER: 64x1x1x16
OUT: 16x25x5

TileShape 2
IN: 32x25x5
KER: 32x1x1x32
OUT: 32x25x5

For **TileShape** in **TileList**:
For **MM** in **Table 1**:

HW-in-the-loop
GVSoC

AUTOTUNER

2.4x improvement vs parallel, non-unrolled MM

TileShape 1
IN: 64x25x5
KER: 64x1x1x16
OUT: 16x25x5

Size
44.1 kB

Optimal MM
mm_unroll_2x4

Experimental Results

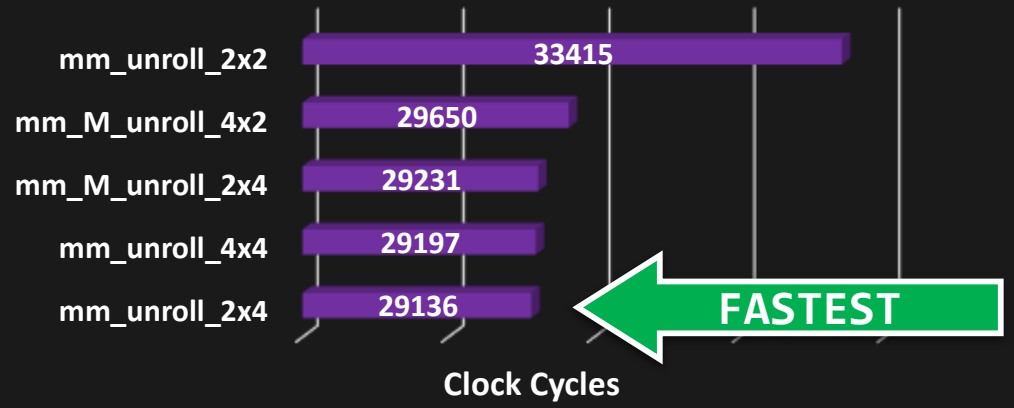
AUTOTUNER - UNDER THE HOOD

POINTWISE CONVOLUTION
STEP: WG-BW
IN: 64x25x5
KER: 64x1x1x64
OUT: 64x25x5

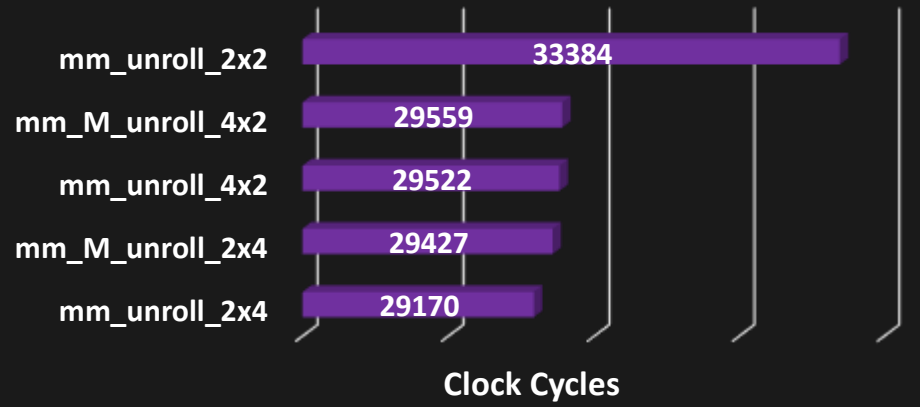
TileShape 1
IN: 64x25x5
KER: 64x1x1x16
OUT: 16x25x5

TileShape 2
IN: 32x25x5
KER: 32x1x1x32
OUT: 32x25x5

BOTH REQUIRE 4 TILES!

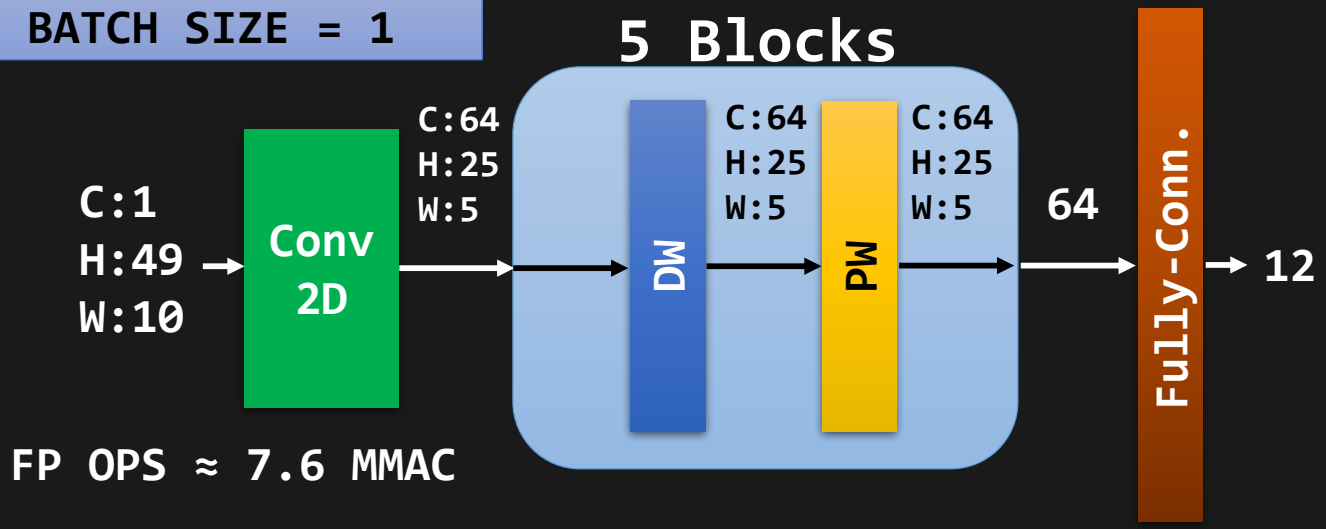


PointWise WG-BW (with TileShape 1)
MAC/Clock: 4.39

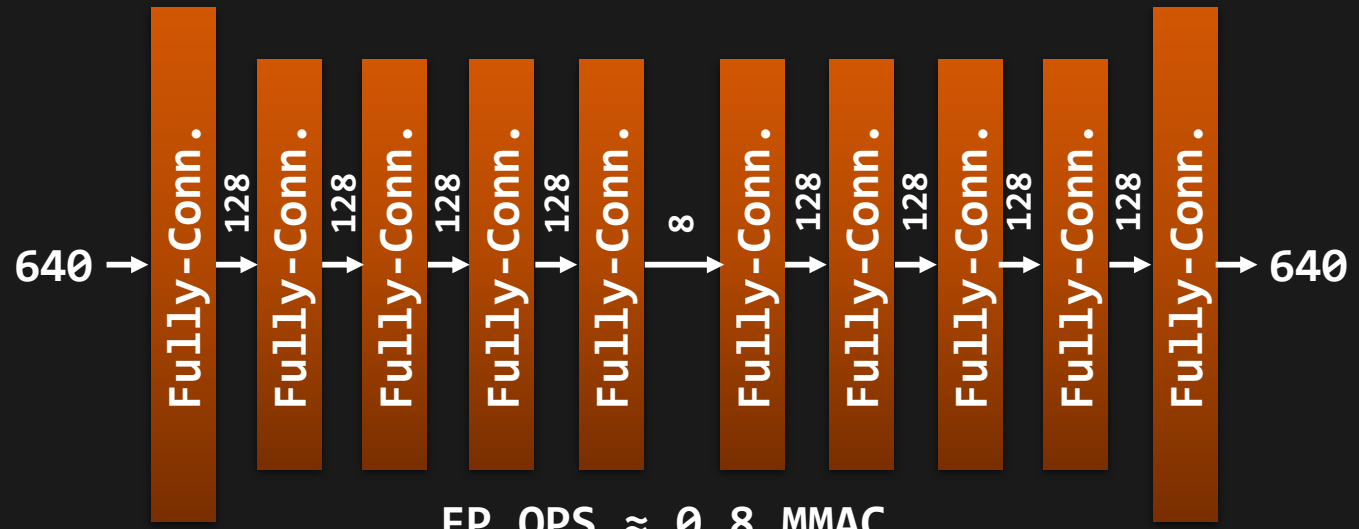


COMPLETE TINYML MODELS' TRAINING

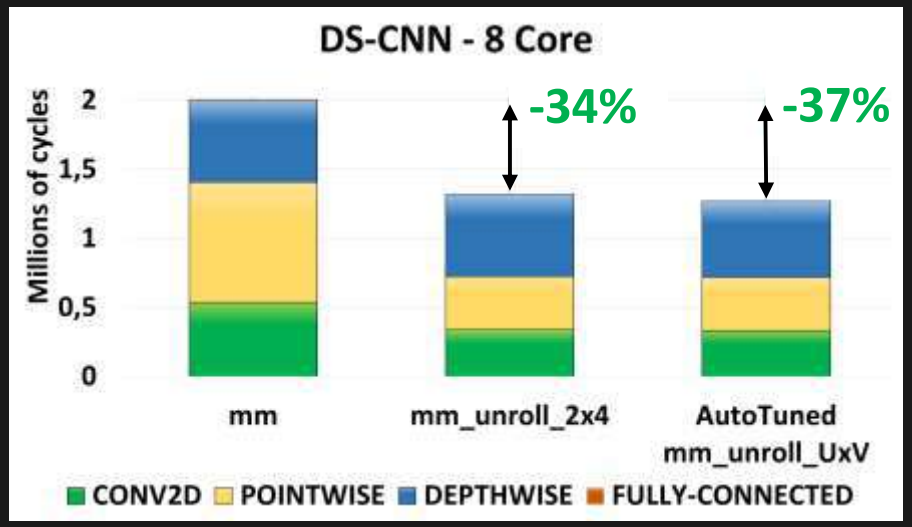
ONLINE LEARNING
BATCH SIZE = 1

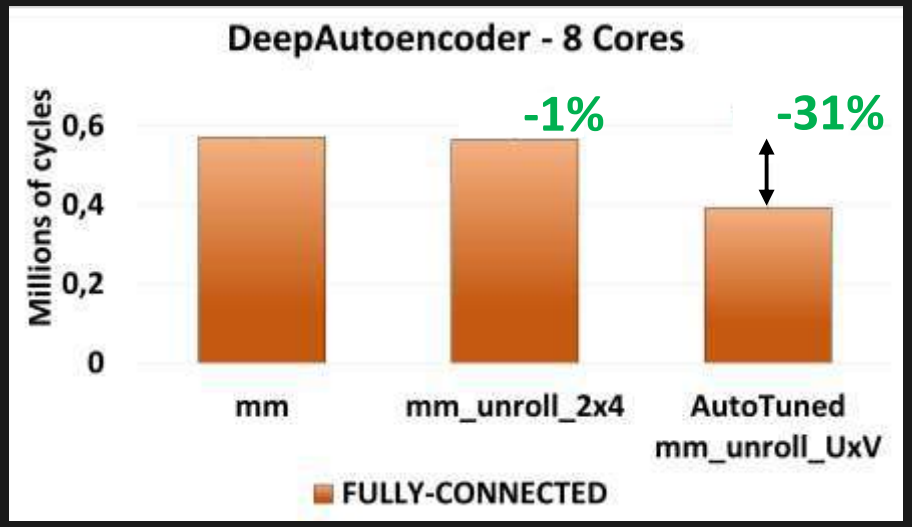


FP OPS ≈ 7.6 MMAC



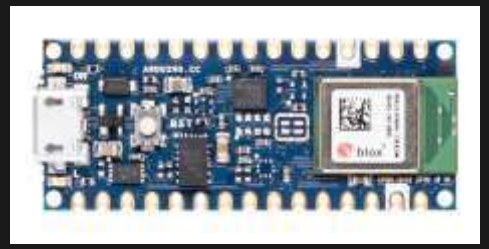
FP OPS ≈ 0.8 MMAC





COMPARISON WITH SOA - AIfES

AIfES: DNN on-device training/inference library for Arduino (with ARM MCUs)



Our target:
ARM CORTEX M4
vs
GVSoC

Training a Deep Autoencoder

AIfES core:
ARM-CMSIS MM

PULP ISA
(2x wrt ARM)
+ AutoTuner

PARALLELISM
+ AutoTuner
(6.82x speedup wrt 1 RISC-V Core)

Conclusion

SUMMARY

PULP-TrainLib

+

AutoTuner

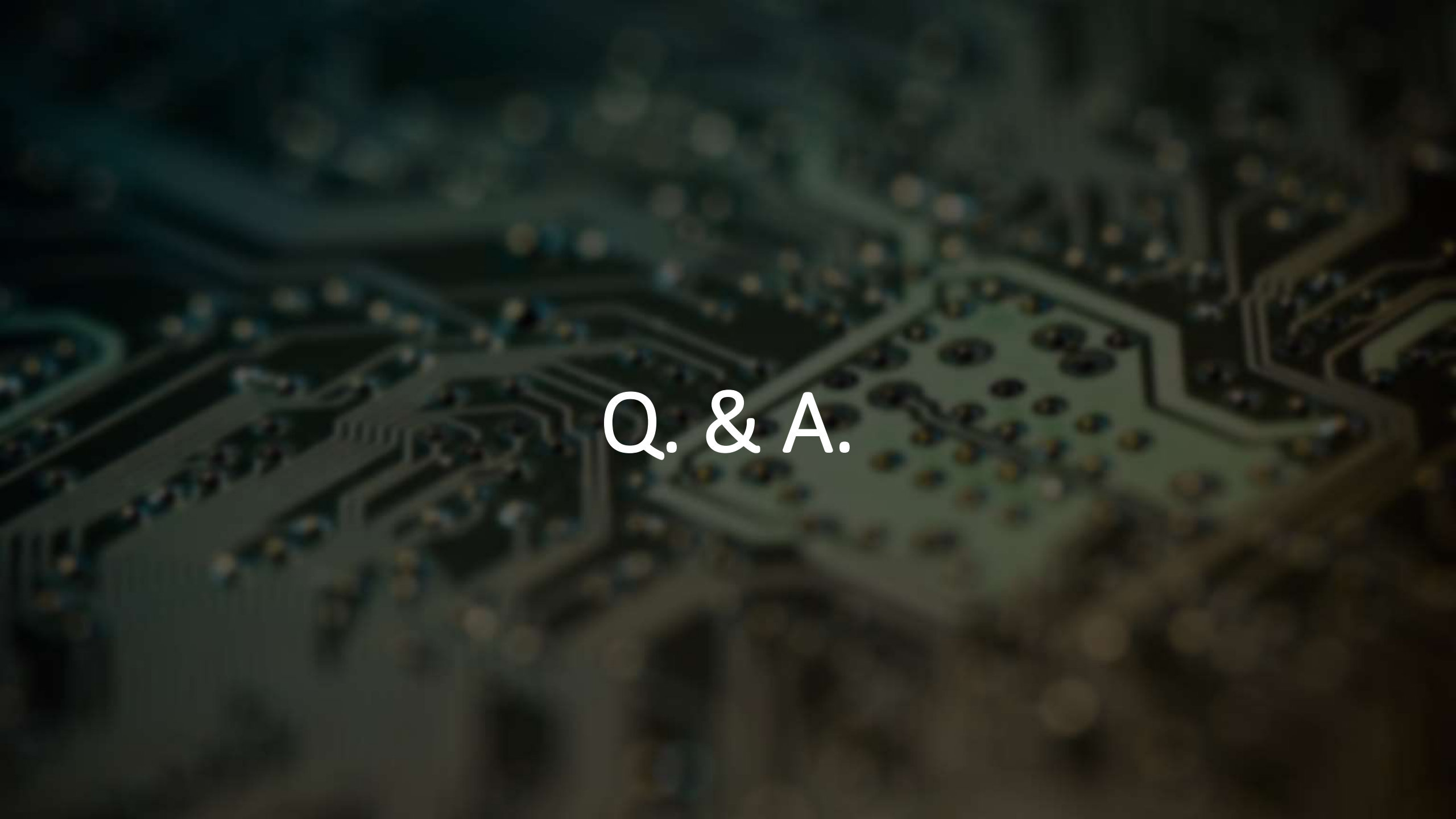
The first **Open-Source SW Library for On-Device Training** on RISC-V Multicore MCUs

Up to **4.39 MAC/clock on 8 RISC-V cores** to execute FP32 DNN training primitives (with AutoTuner)

Almost real time training of TinyML models:
0.9 ms (Deep Autoencoder), 3 ms (DS-CNN)
@450 MHz (PULP-based Vega SoC)

30.7x faster performance **wrt AIfES** on Cortex M4

THANK YOU FOR YOUR ATTENTION


Q. & A.