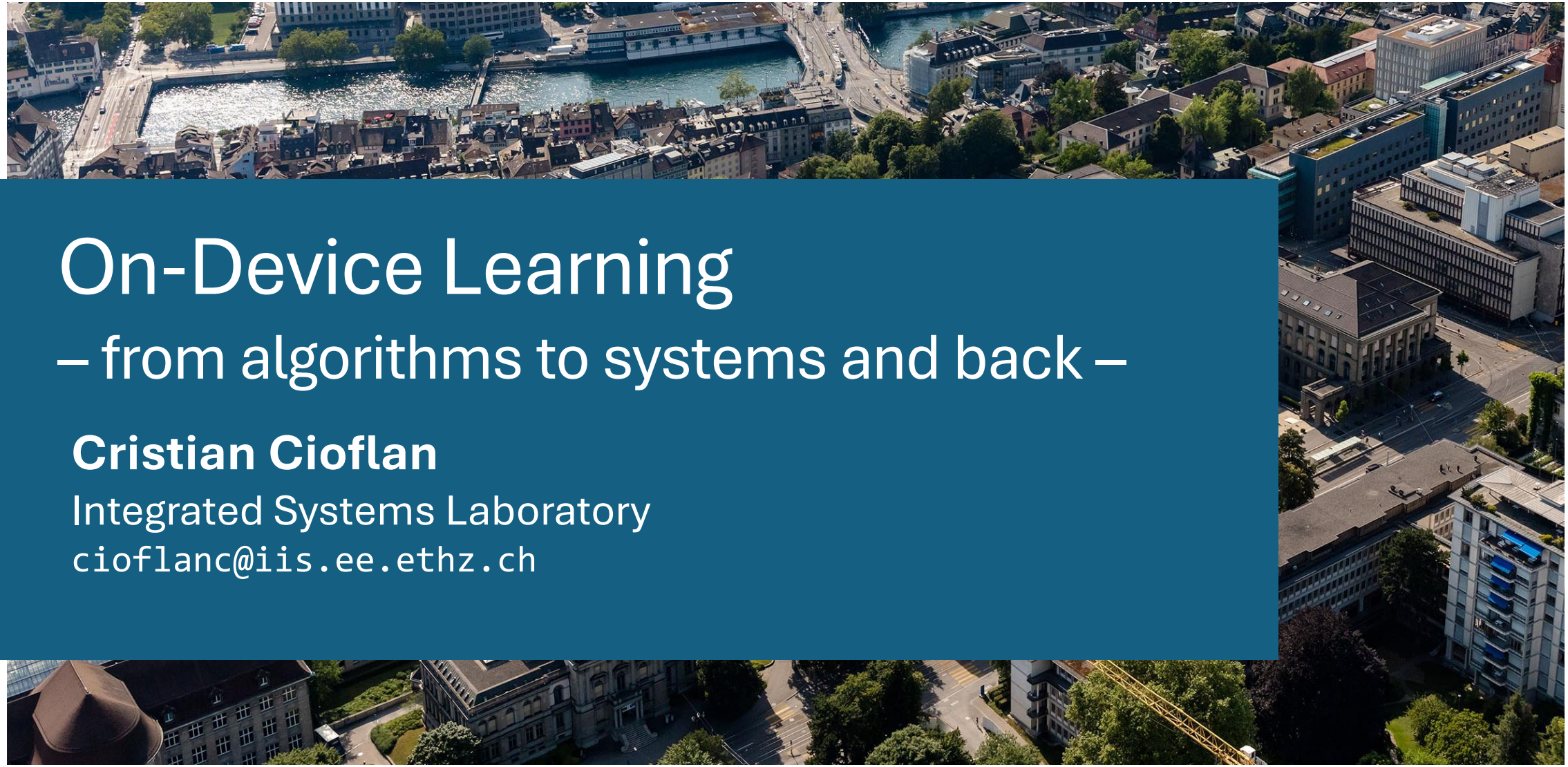
ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

PULP
Parallel Ultra Low Power



On-Device Learning

– from algorithms to systems and back –

Cristian Cioflan
Integrated Systems Laboratory
cioflanc@iis.ee.ethz.ch

Bio

- PhD student in the Integrated Systems Laboratory @ ETHZ
  - B.Eng. from University Polytehnica of Bucharest
  - M.Sc. from ETH Zurich

Bio

- PhD student in the Integrated Systems Laboratory @ ETHZ
  - B.Eng. from University Polytehnica of Bucharest
  - M.Sc. from ETH Zurich
- Main research interests
  - TinyML for audio systems
  - On-Device Learning
  - Sensor fusion
  - Neural Architecture Search

Cristian Cioflan

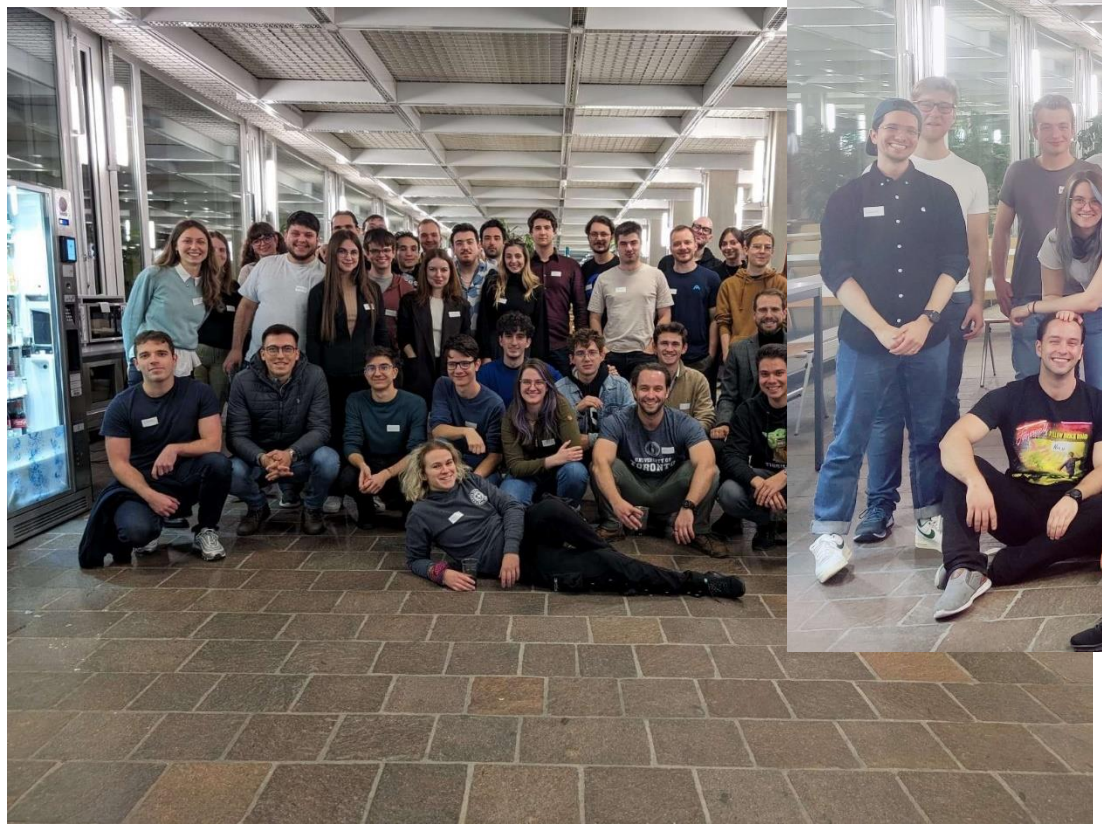
Cristian aspires to become the master of ML on constrained embedded devices and strives to bring "Aha!" moments to our students.

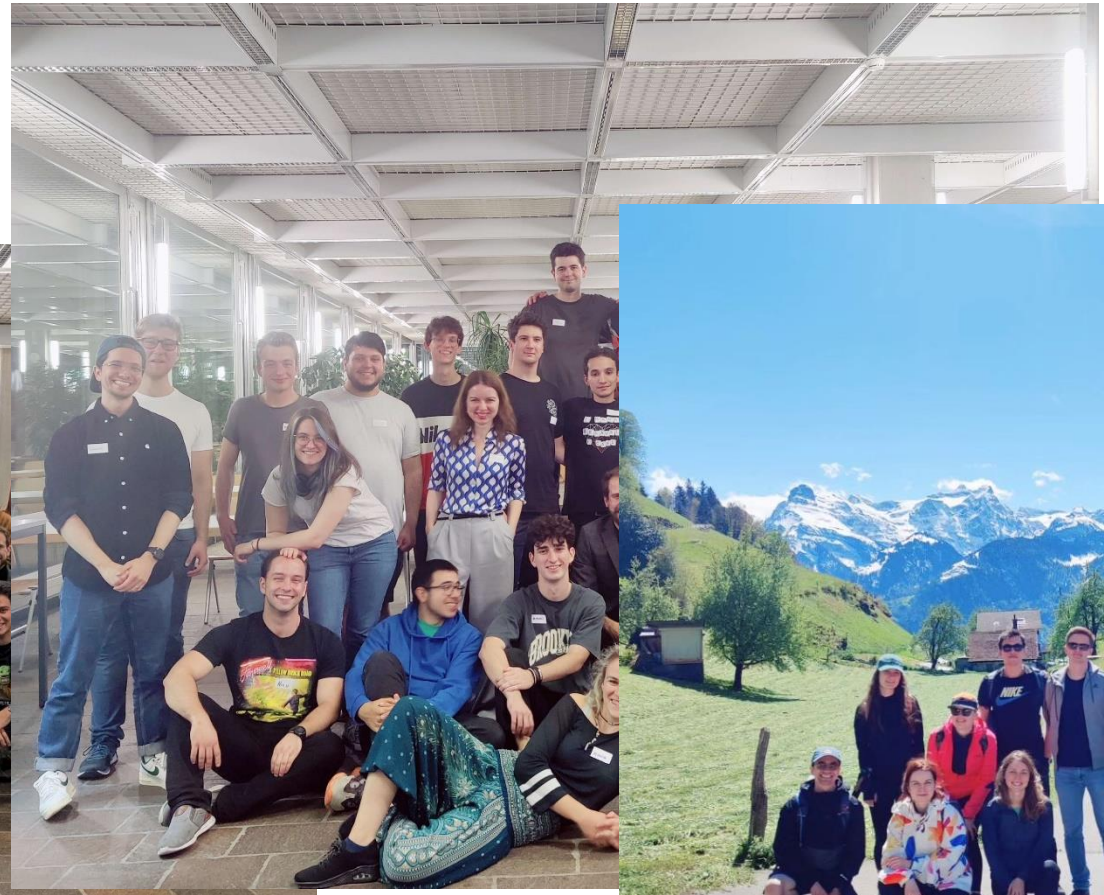
ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

Afterwork Bio





Co-founded ARICH - the Romanian Student Association in Switzerland

ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

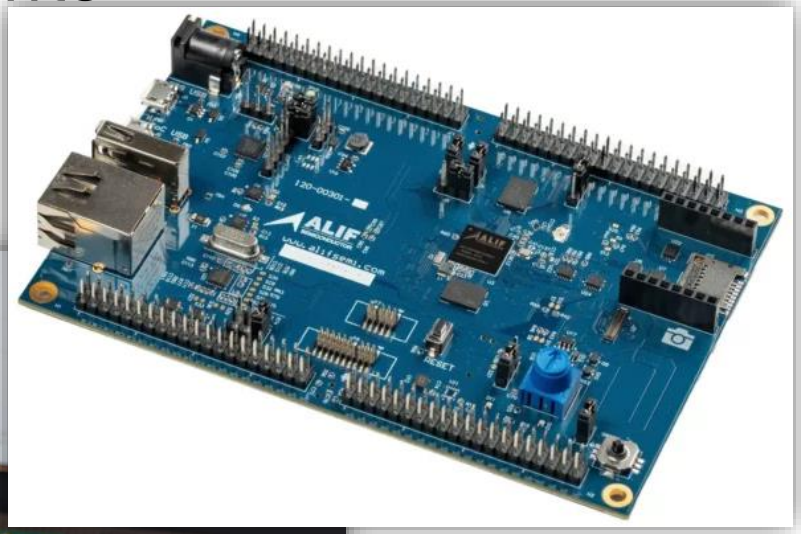
ARICH

On-Device Learning

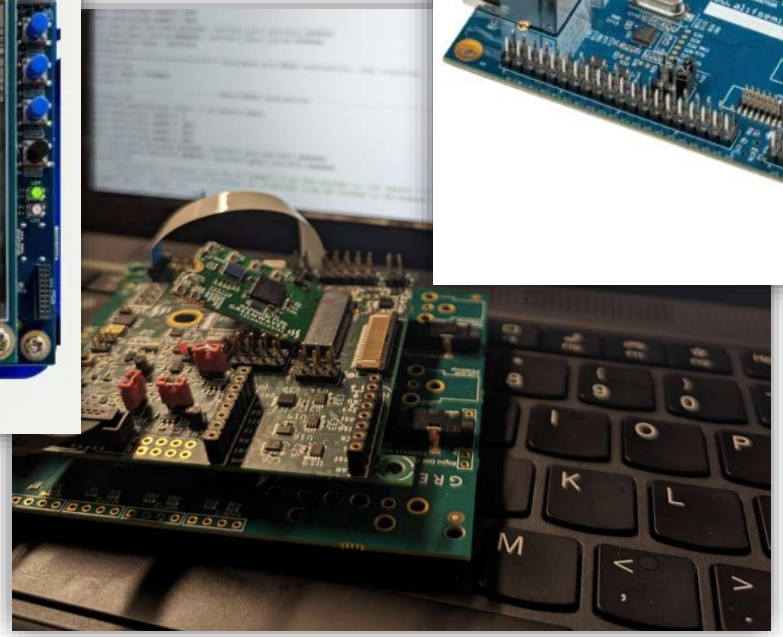
What is On-Device Learning?

- First, the devices – tinyML domain & its constraints
  - mW of power
  - embeddable
  - low-cost
- MCUs w/ NPUs

STM32N6

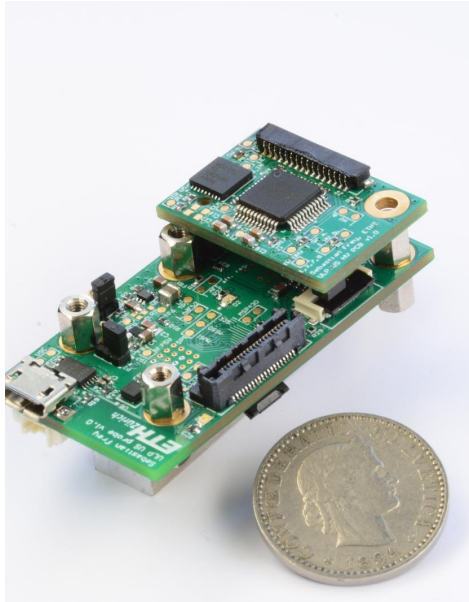


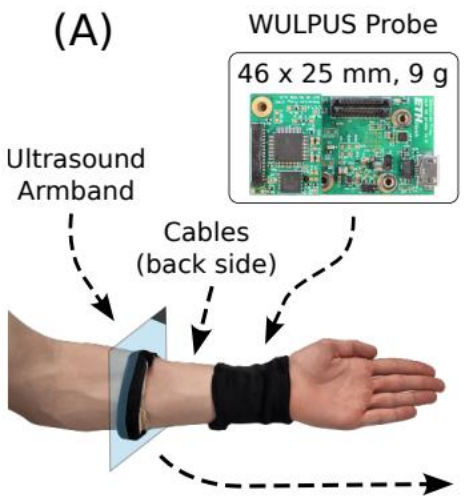
Alif E1C w/ Ethos-U55

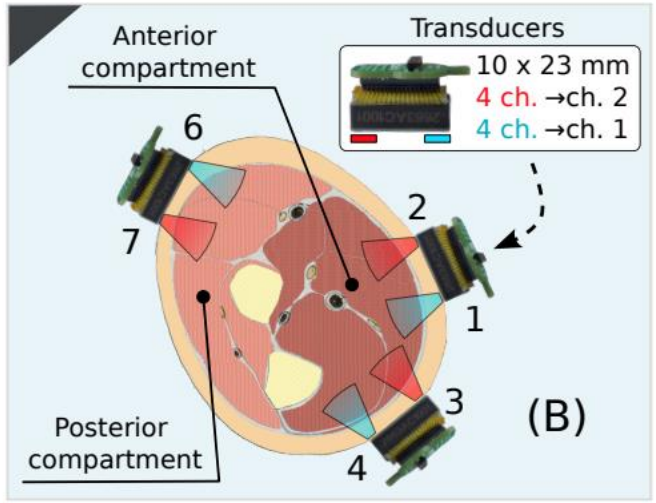


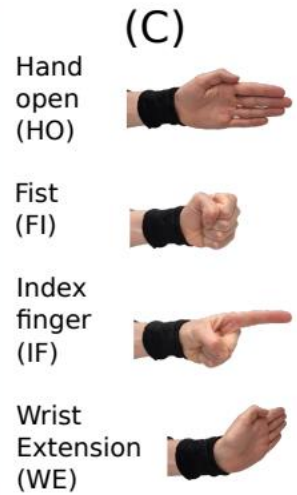
GAP9

Extreme-Edge Deep Learning





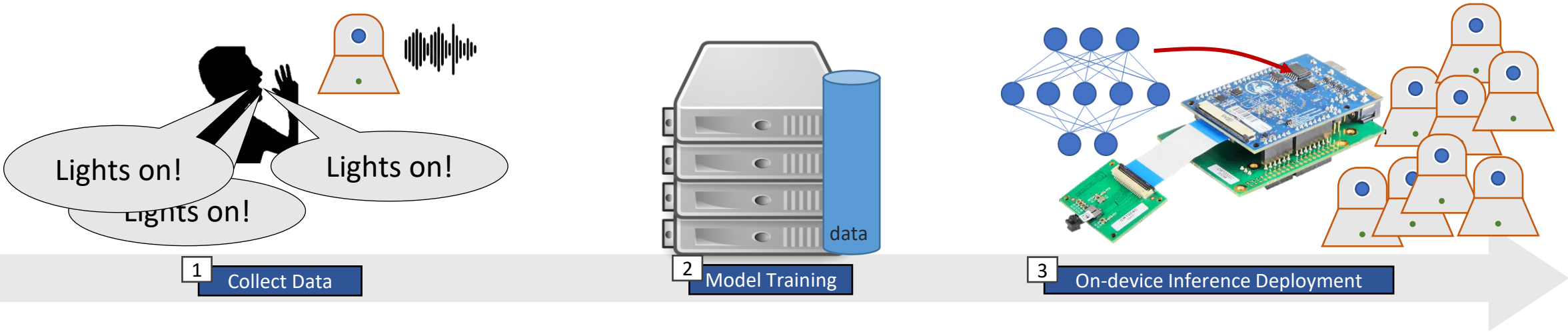




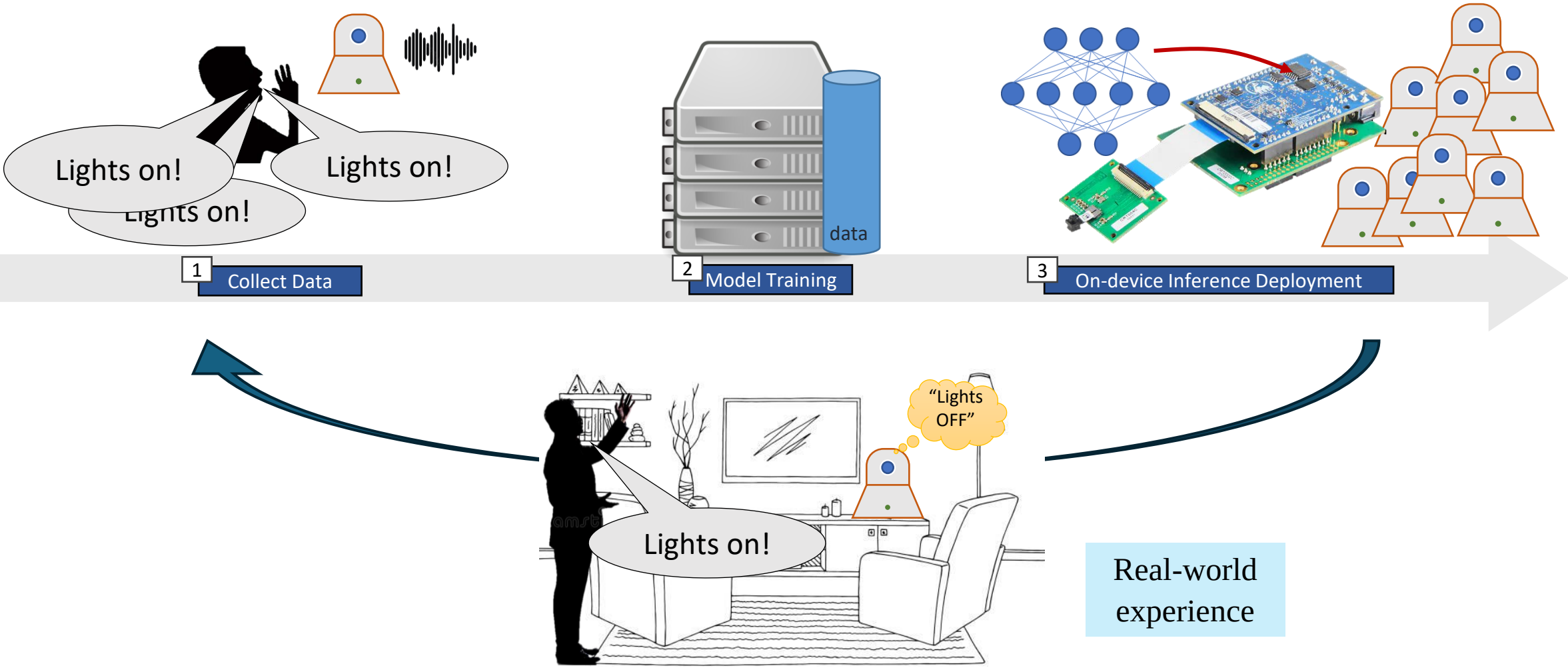
NanoSLAM (Simultaneous Localization And Mapping) [Niculescu2024]

Ultrasound-based gesture recognition [Vostrikov2023]

Train-Once-Deploy-Everywhere

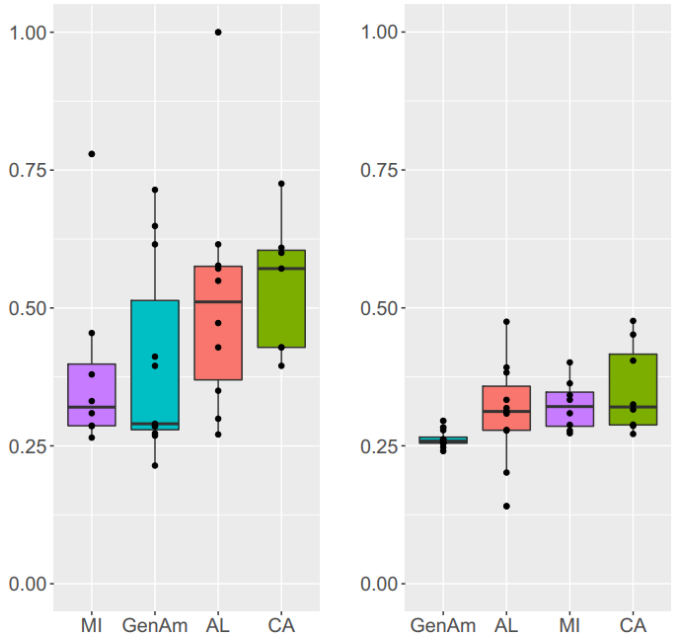


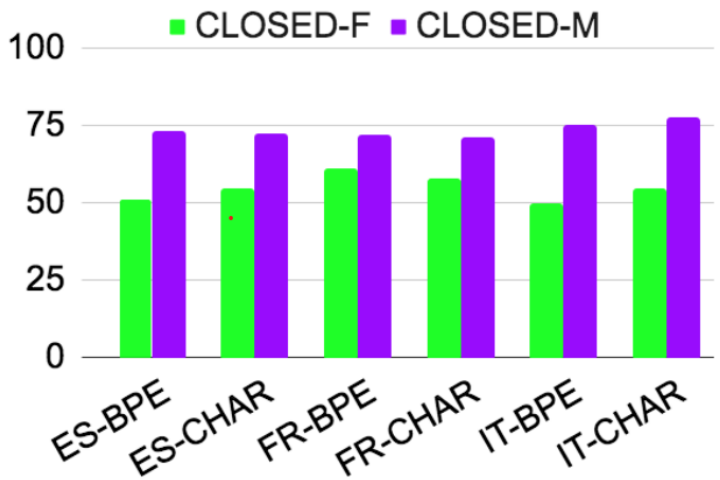
Train-Once-Deploy-Everywhere



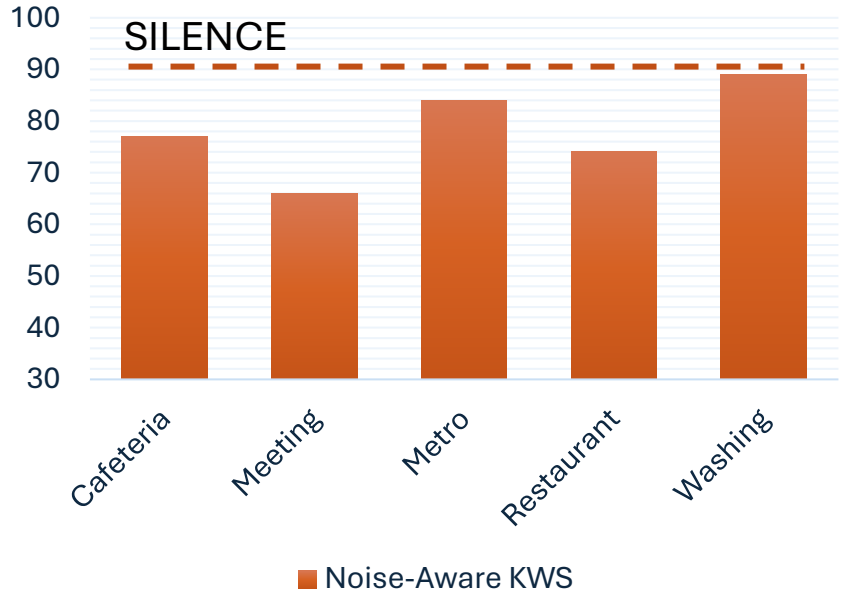
Accuracy degrades in real-world conditions

- Unknown environments where pretraining (offline) ≠ target (online) data
  - Domain shifts, differences in sensors, knowledge expansion
  - Accents, genders, background noises





Feminine vs. masculine accuracy on function words for two speech translation model, in three languages [Savoldi2022]



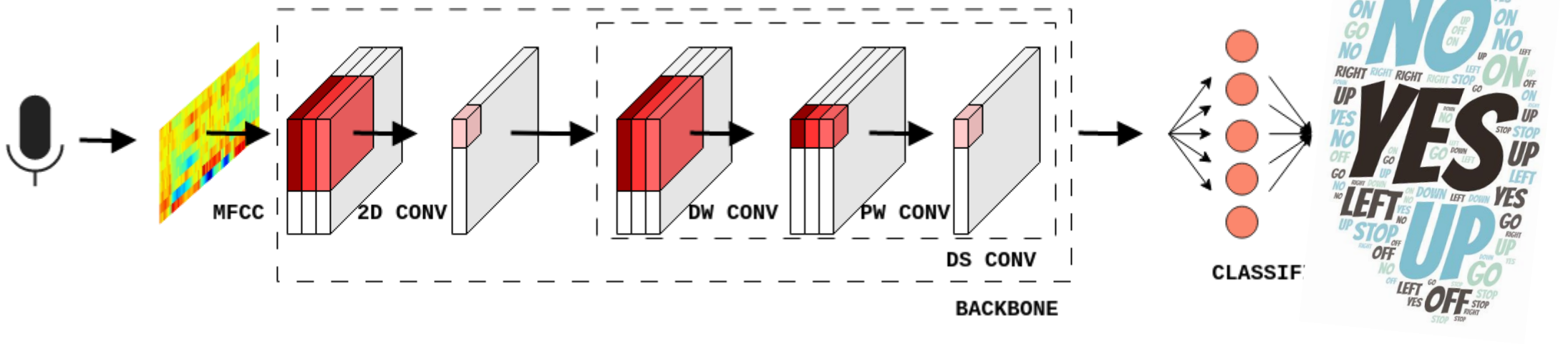
Keyword spotting accuracy drops by 3%-26% compared to silent environments [Cioflan2024]

On-Device Learning

A case study

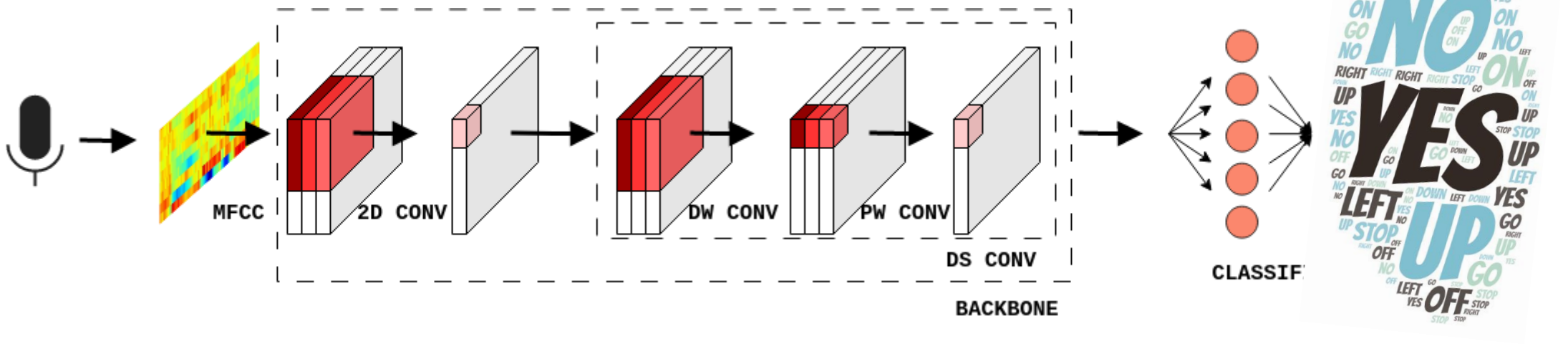
Keyword spotting – a case study for ODL

- **Process** an audio signal
- **Recognize** a **target** word from a **predefined set**



Keyword spotting – **92% in clean conditions**

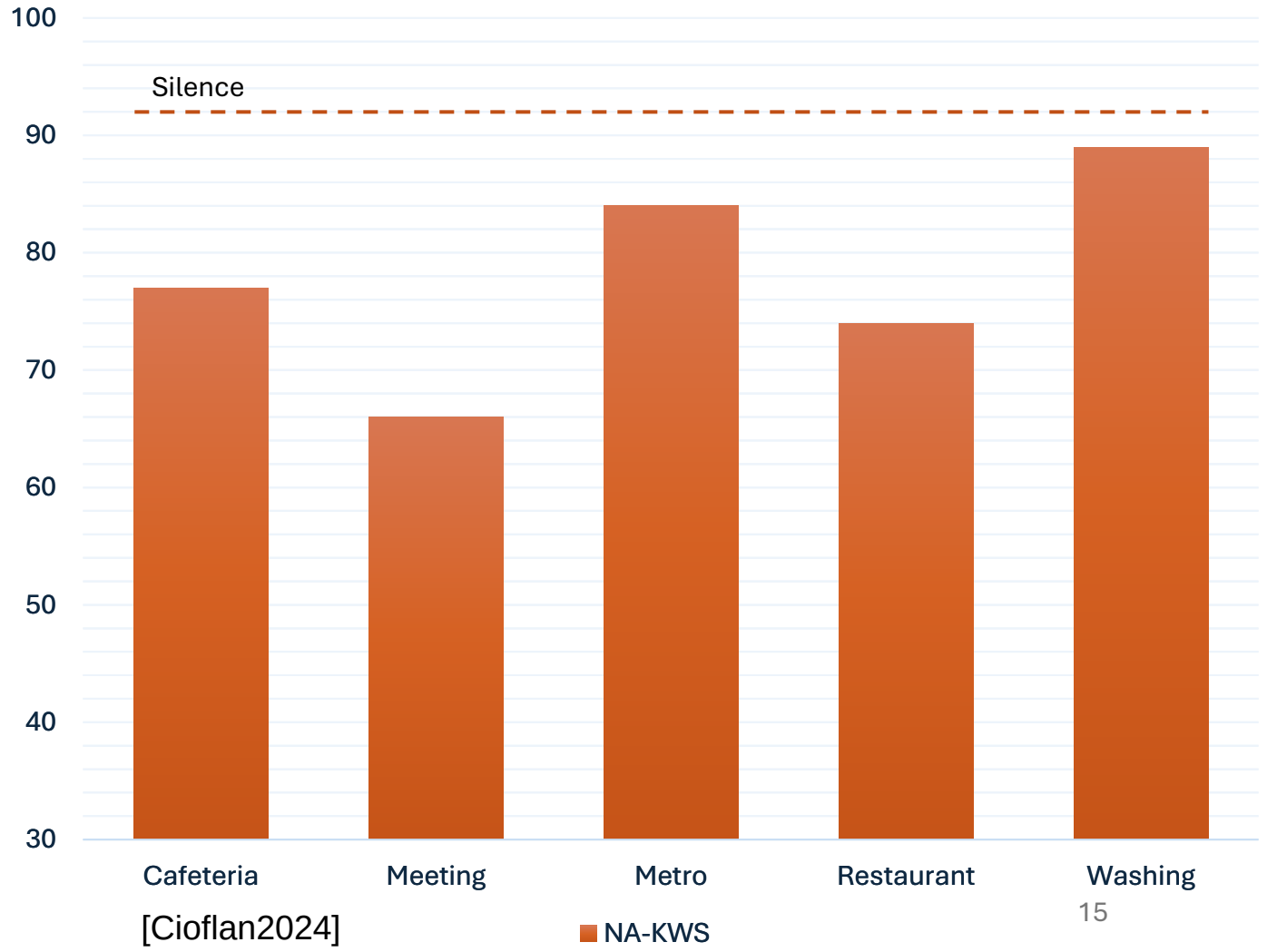
- **Process** an audio signal
- **Recognize** a **target** word from a **predefined set**



Quiet rooms are not the norm...

Can a KWS system still recognize the words?

- **Noise-Aware KWS**
  - Noise-augmented KWS at (pre)training time



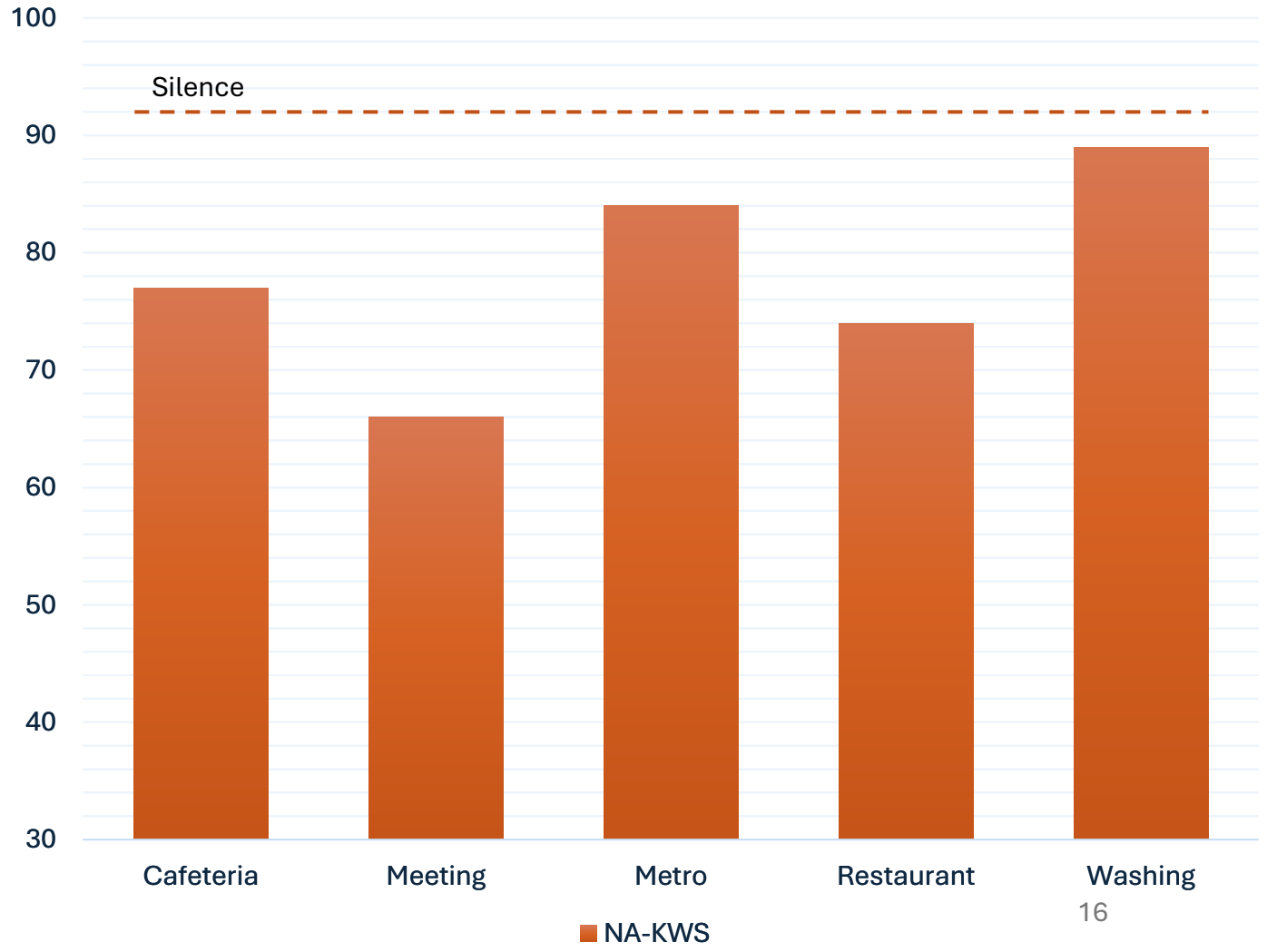
[Cioflan2024]

Can a KWS system still recognize the words?

- **Noise-Aware KWS**
  - Noise-augmented KWS at (pre)training time

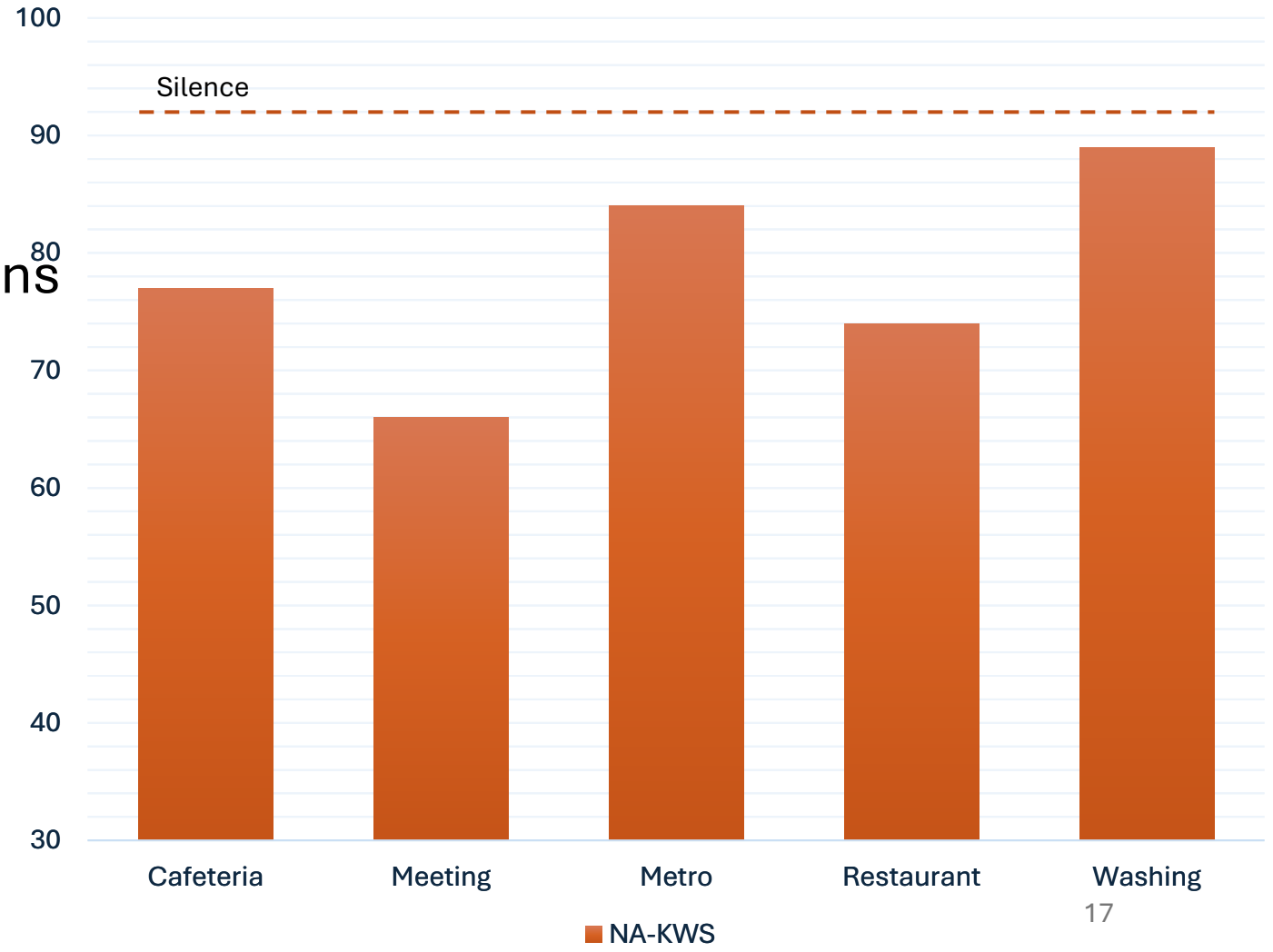
Terminology

pretraining = training (on the server)
pretraining ≠ OD(C)L
OD(C)L = adaptation = fine-tuning (at the edge)



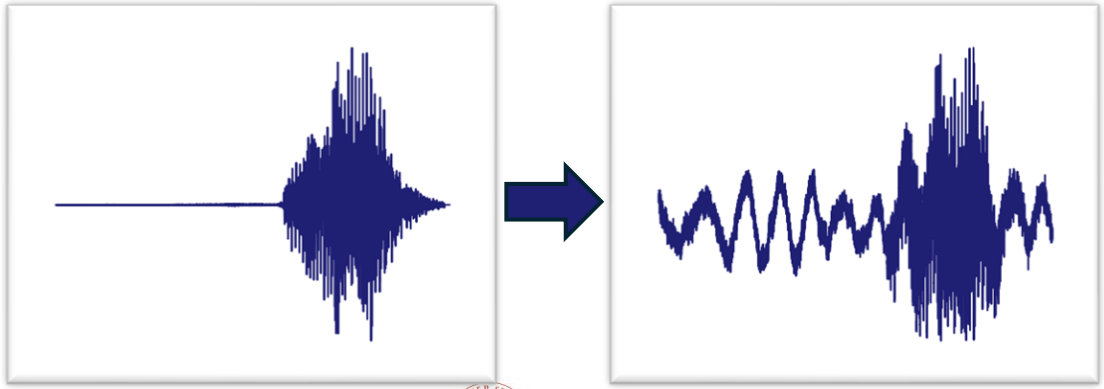
Can a KWS system still recognize the words?

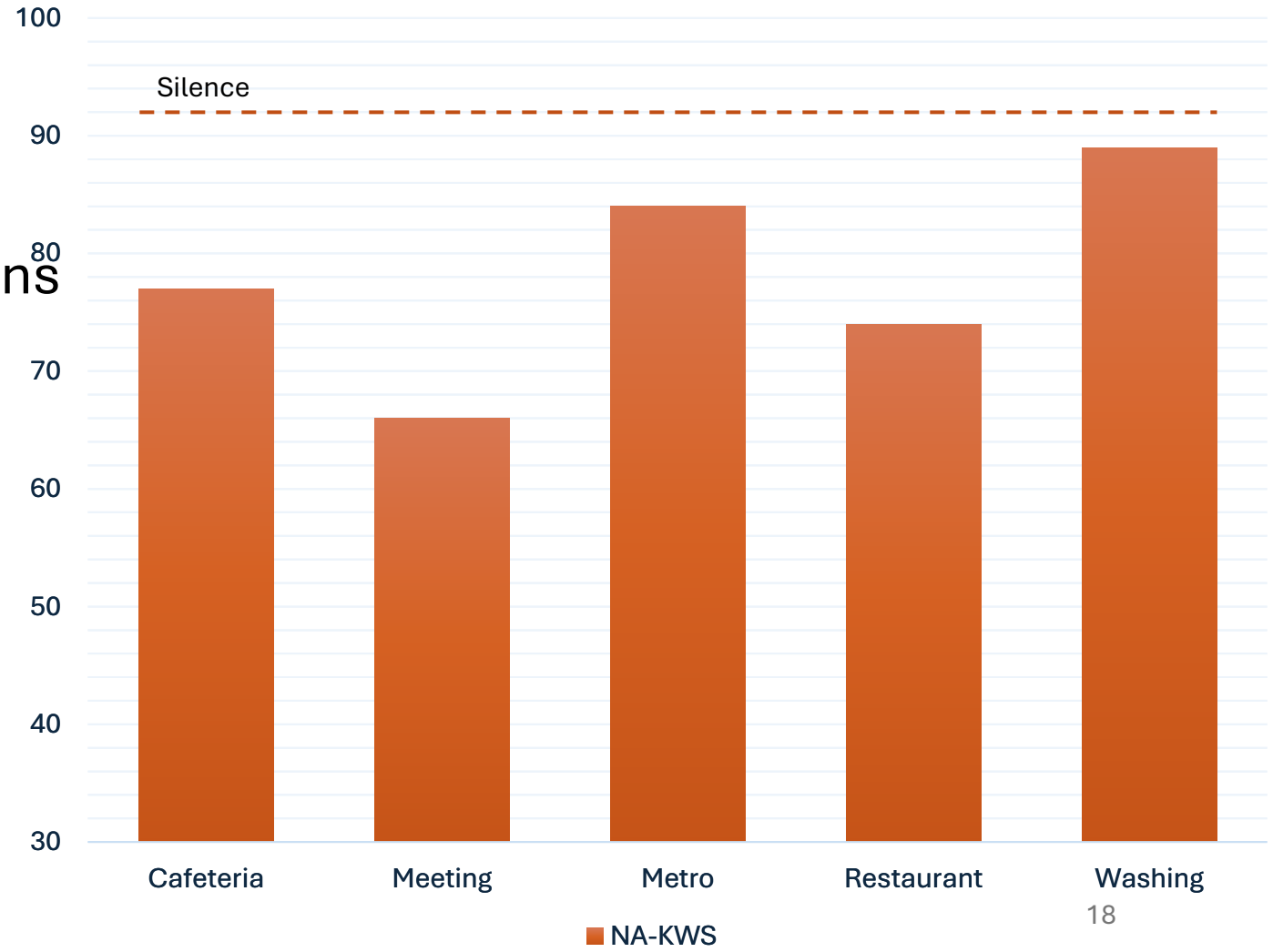
- **Noise-Aware KWS**
  - Noise-augmented KWS at (pre)training time
  - 3% to 26% accuracy reductions over silent environments



Can a KWS system still recognize the words?

- **Noise-Aware KWS**
  - Noise-augmented KWS at (pre)training time
  - 3% to 26% accuracy reductions over silent environments
  - Difficult to separate the target from the noise





ETH zürich

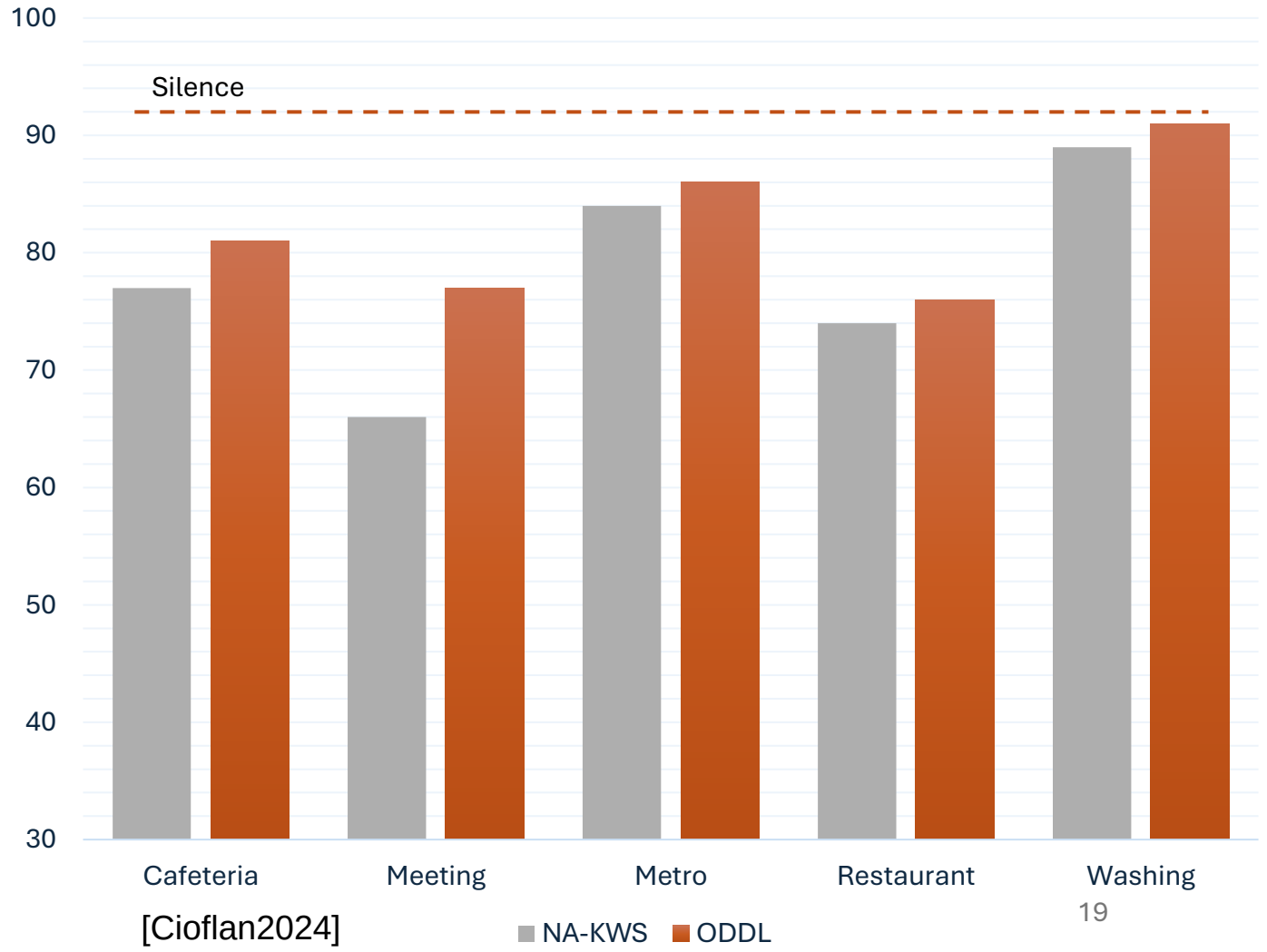
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

On-Device Domain Learning

Improving the KWS accuracy in noisy environments

- **Accuracy increments** by 4% on average over **NA-KWS**
- 13% on **speech noise**



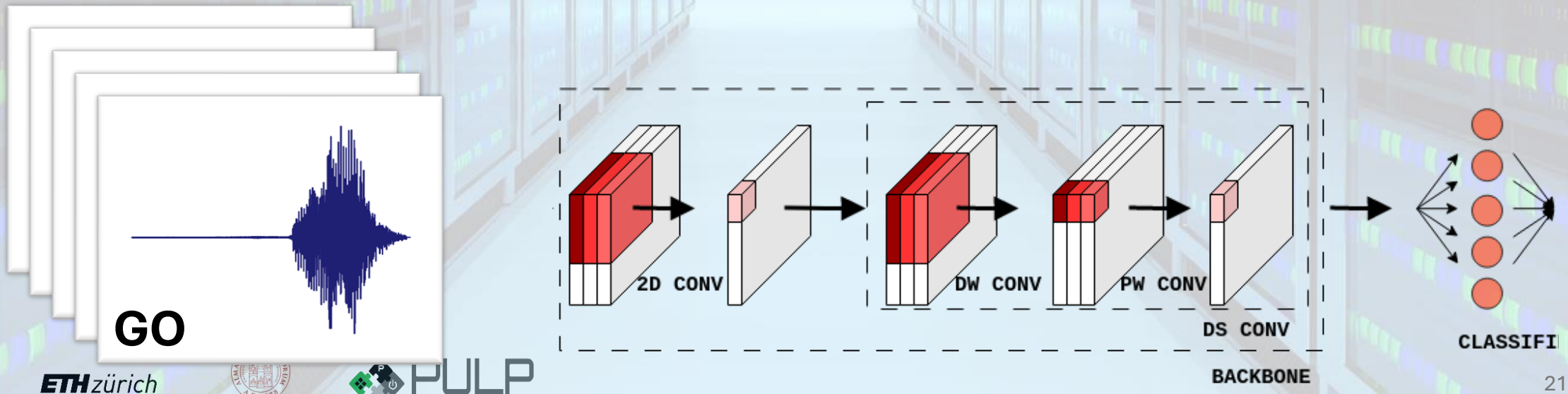
[Cioflan2024]

The Methodology

- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server

The Methodology

- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server
  - Deploy KWS model
  - Store pre-recorded utterances and labels

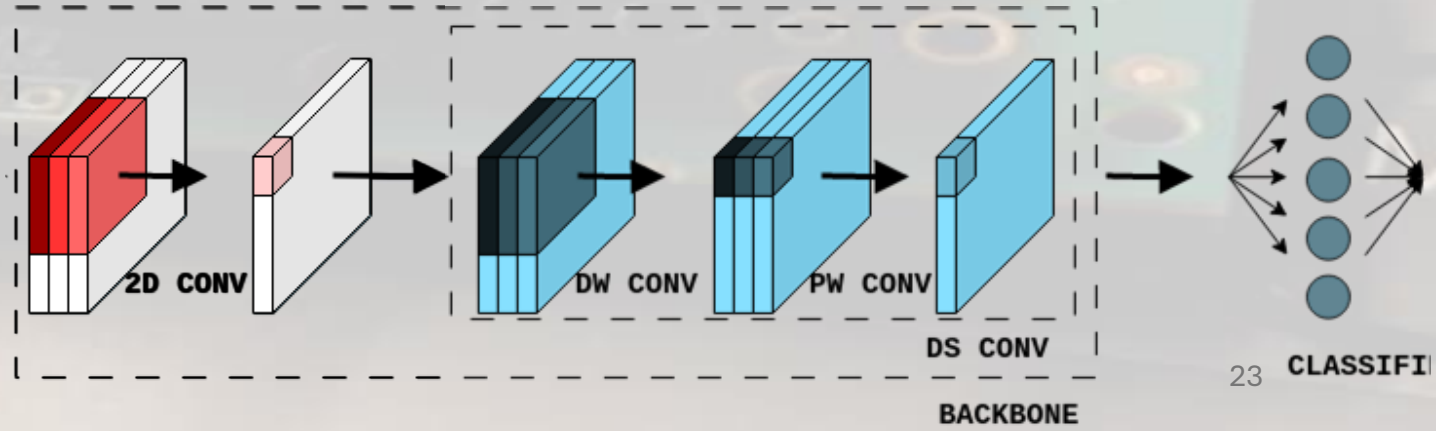


The Methodology

- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server
  - Deploy KWS model
  - Store pre-recorded utterances and labels
- Adapt to new environments
  - Record noise from the environment
  - Augment pre-recorded utterances

The Methodology

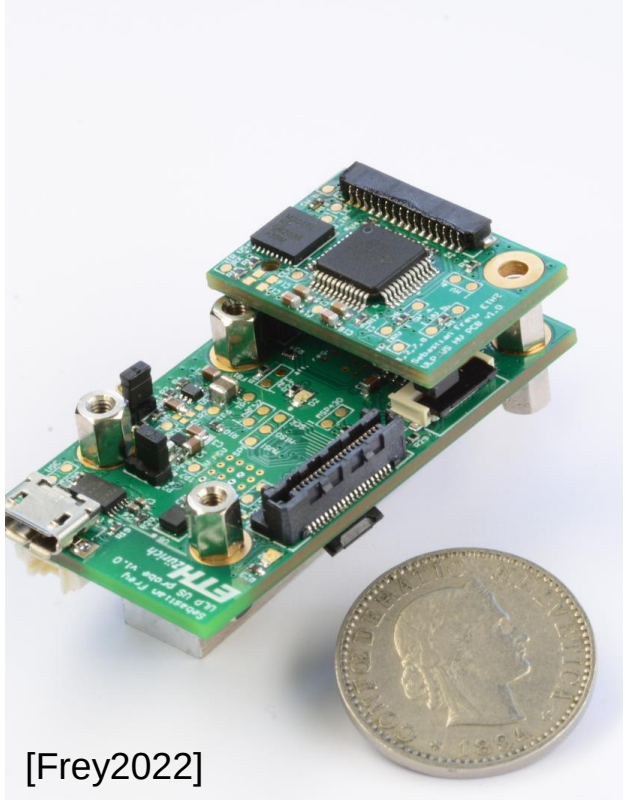
- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server
  - Deploy KWS model
  - Store pre-recorded utterances and labels
- Adapt to new environments
  - Record noise from the environment
  - Augment pre-recorded utterances
  - On-device learning



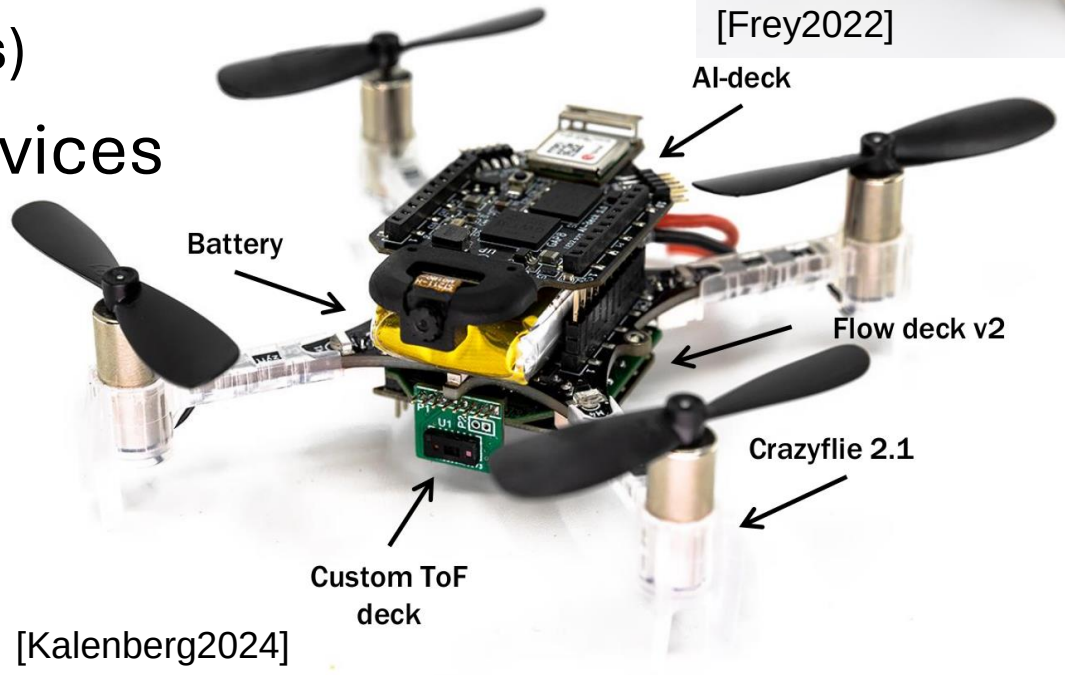
Let’s have a closer look at the target (extreme edge) devices

The embedded perspective

- Embedded, miniaturized devices
  - Limited **storage** (e.g., data, model parameters)
  - Limited **memory** (e.g., activations, gradients)
- Real-time operation
  - Minimize **latency** ($\propto$ **#operations**)
- Always-on, battery operated devices
  - Minimize **energy consumption**

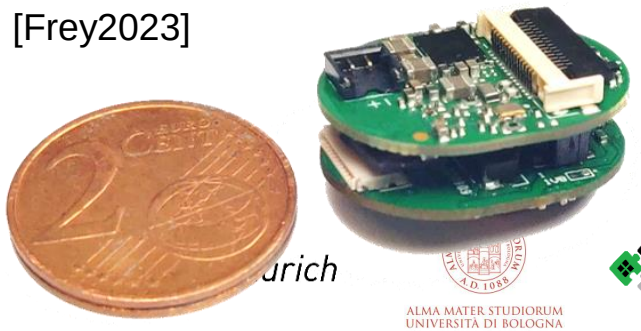


[Frey2022]



[Kalenberg2024]

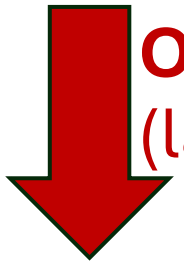
[Frey2023]



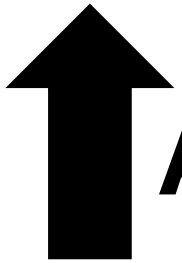
On-Device Learning has costs

Storage

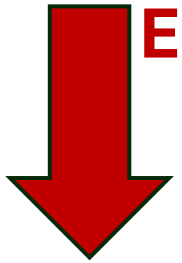
Memory



Operations
(latency)



Accuracy

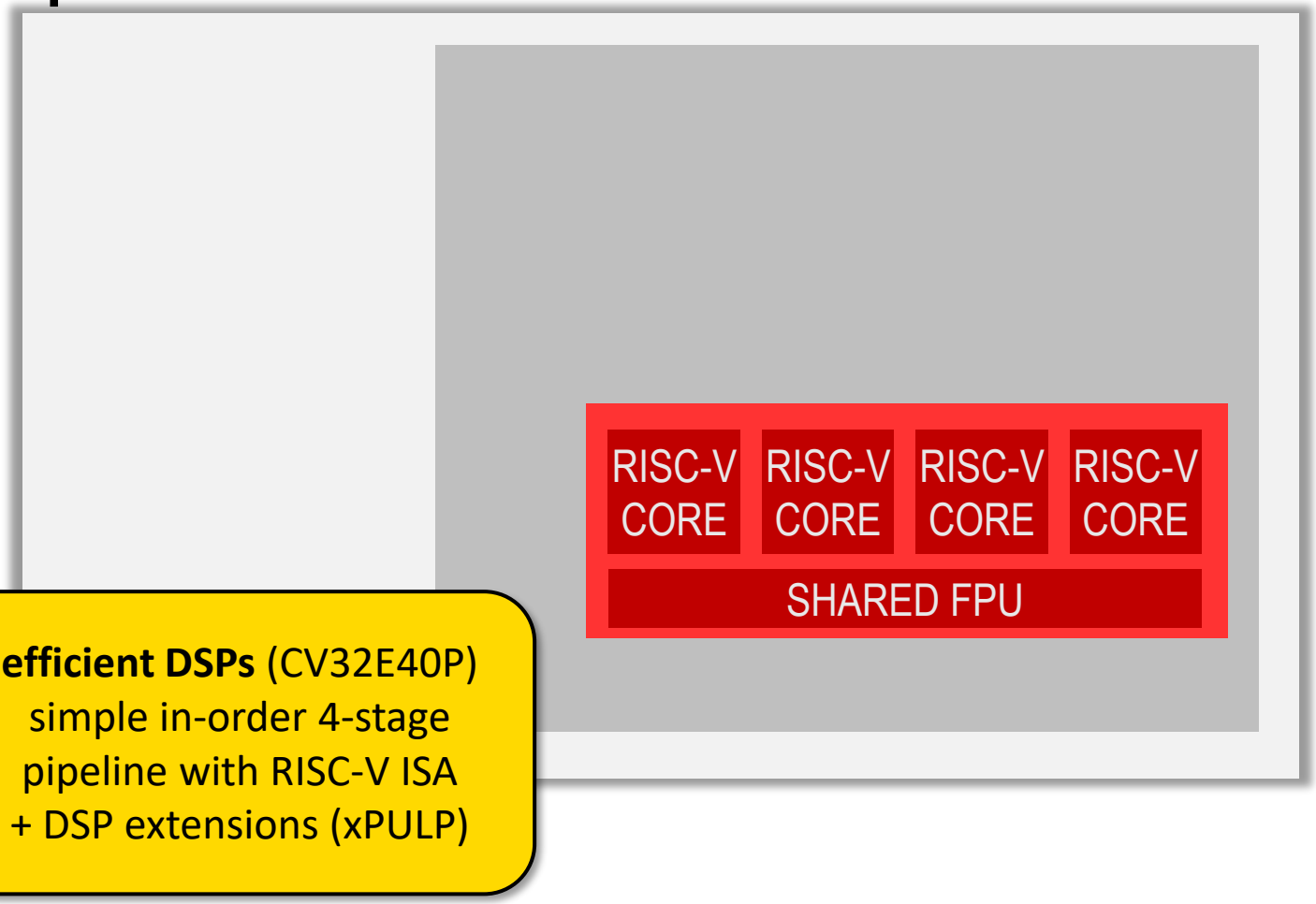


Energy

Parallel Ultra-Low Power Paradigm: multiple cores

PULP Paradigm: multiple cores

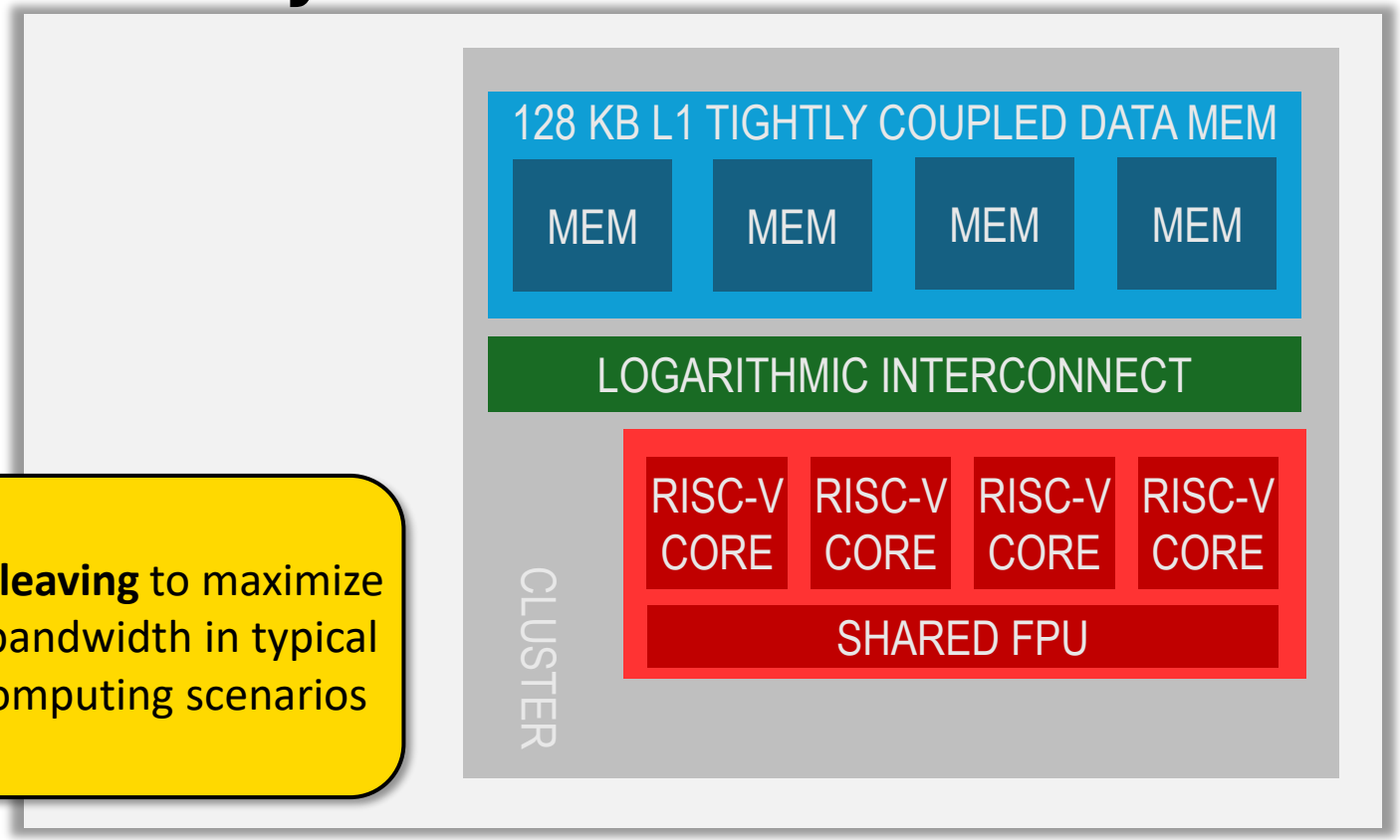
- Heterogeneous compute units
  - General purpose RISC-V cores
  - Shared FPUs



- **Operations (latency)**
- **Energy**

PULP Paradigm: low-latency shared TCDM

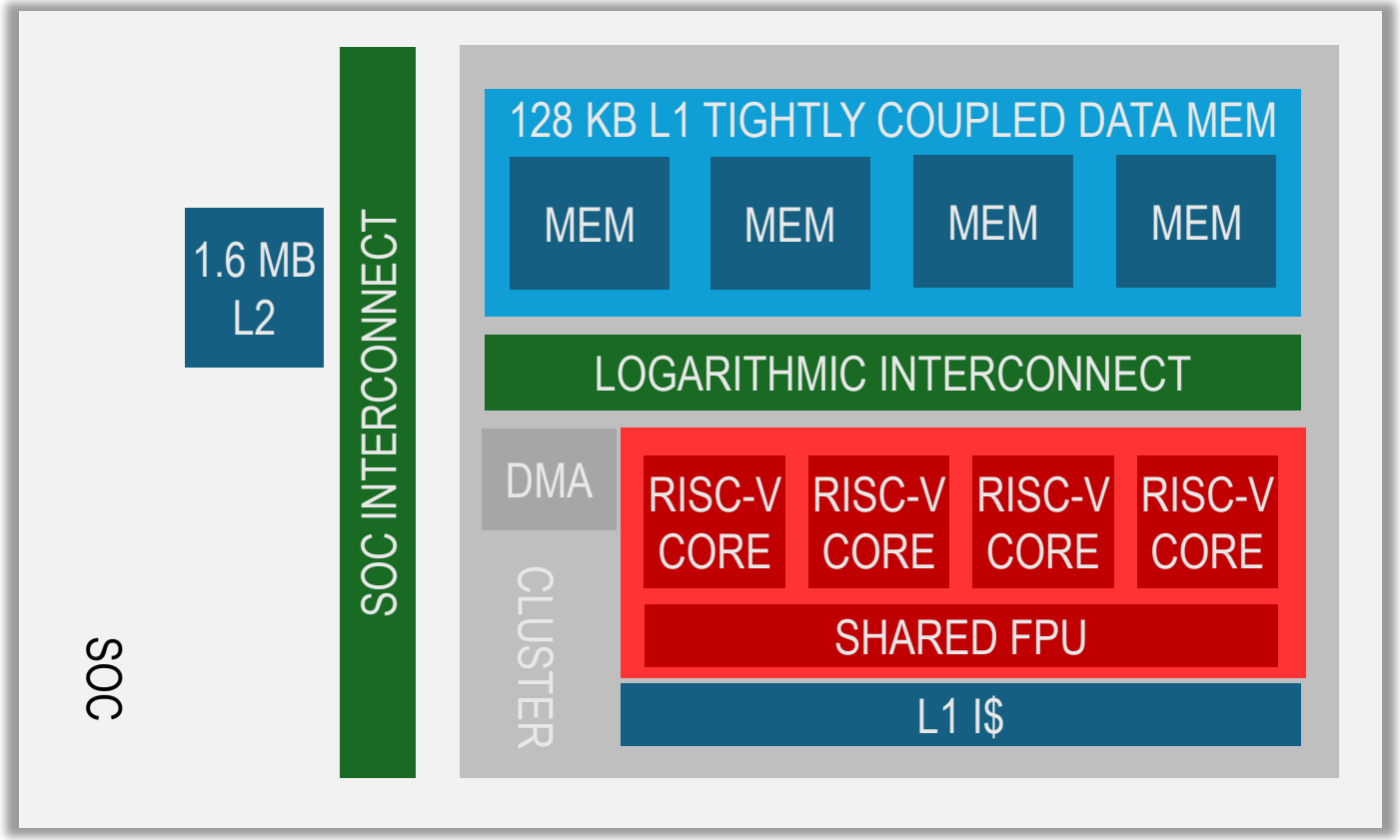
- Hierarchical memory architecture
  - L1 – single-cycle access
  - TCDM – cluster domain



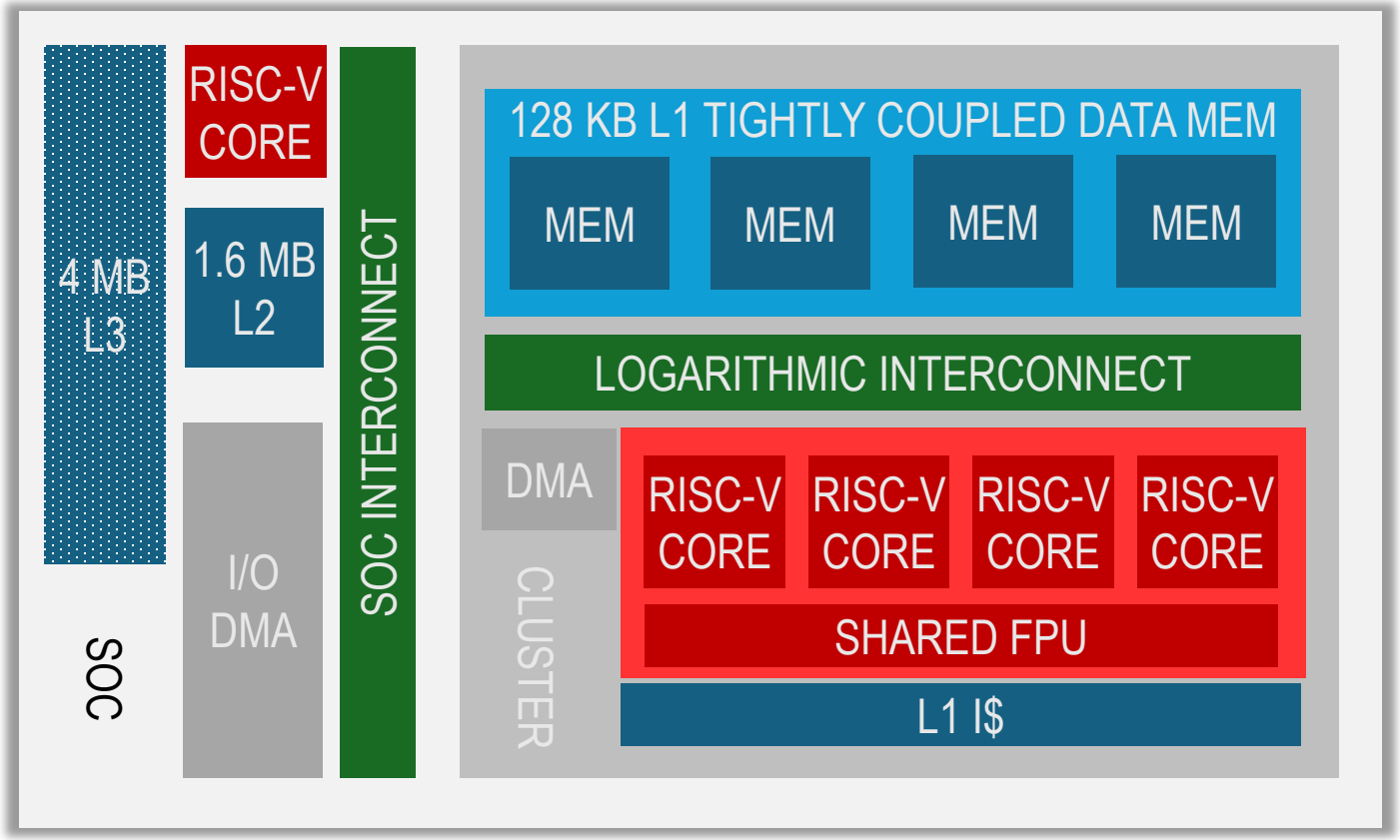
bank interleaving to maximize available bandwidth in typical parallel computing scenarios

PULP Paradigm: DMA & I$ for L2 mem. access

- Hierarchical memory architecture
  - L1 – single-cycle access
    - TCDM – cluster domain
  - L2 – SRAM – SoC domain

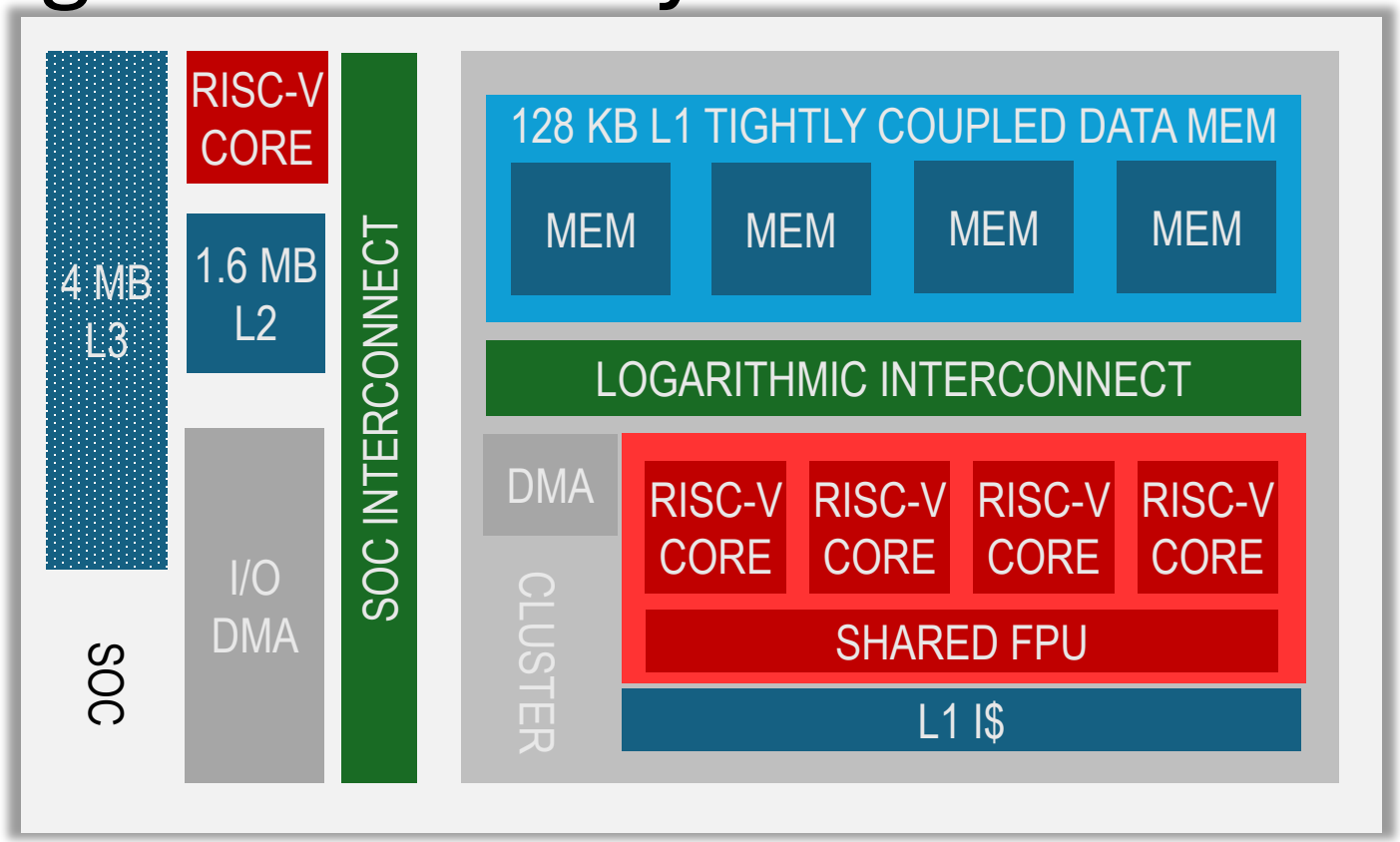


PULP Paradigm: the cluster accelerates the host



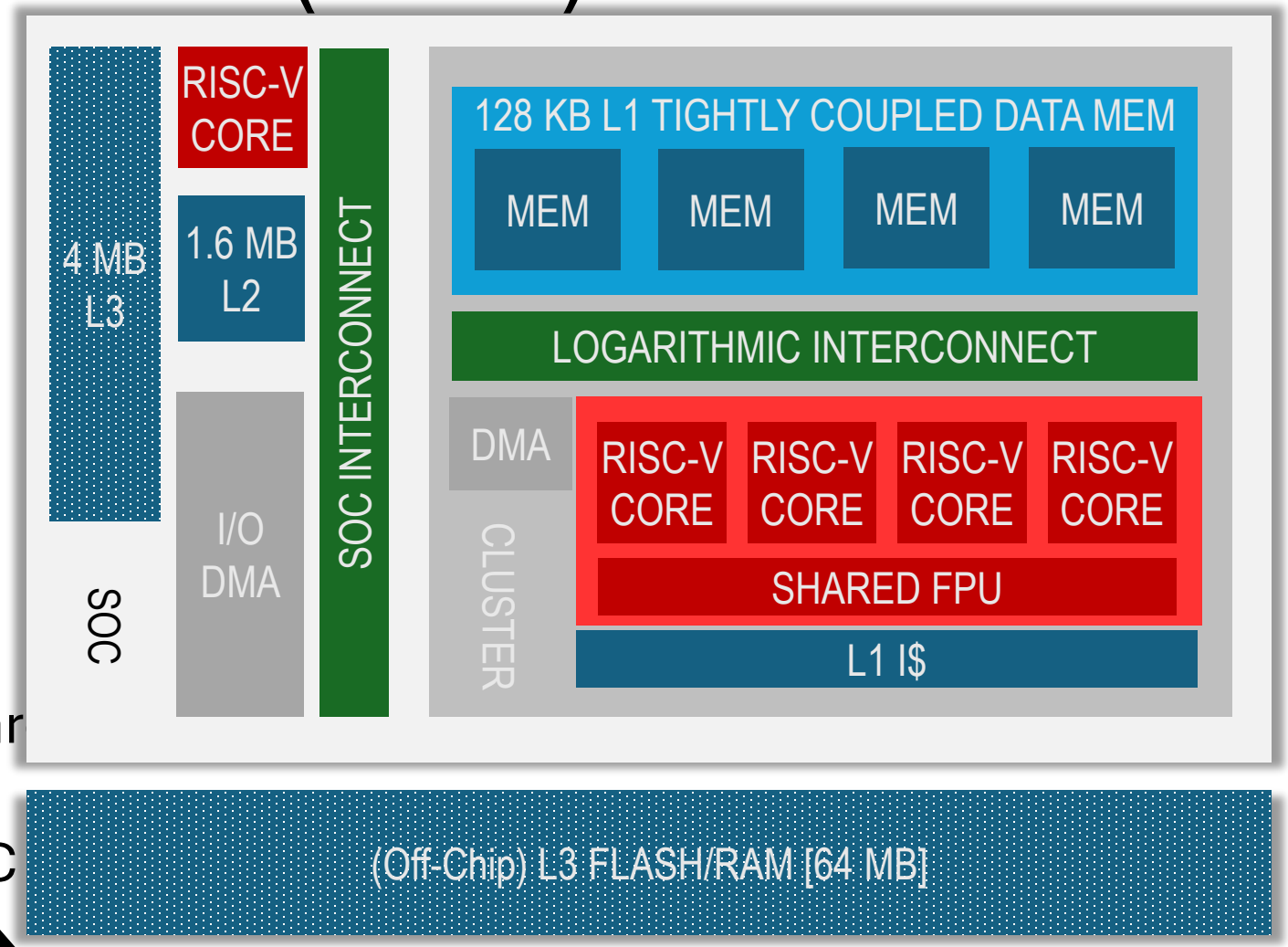
PULP Paradigm: storage and memory

- Hierarchical memory architecture
  - L1 – single-cycle access
    - TCDM – cluster domain
  - L2 – SRAM – SoC domain
  - L3 – Non-volatile
    - On-chip MRAM
    - Off-chip mem. through SPI



Parallel Ultra-Low Power (PULP) Platform

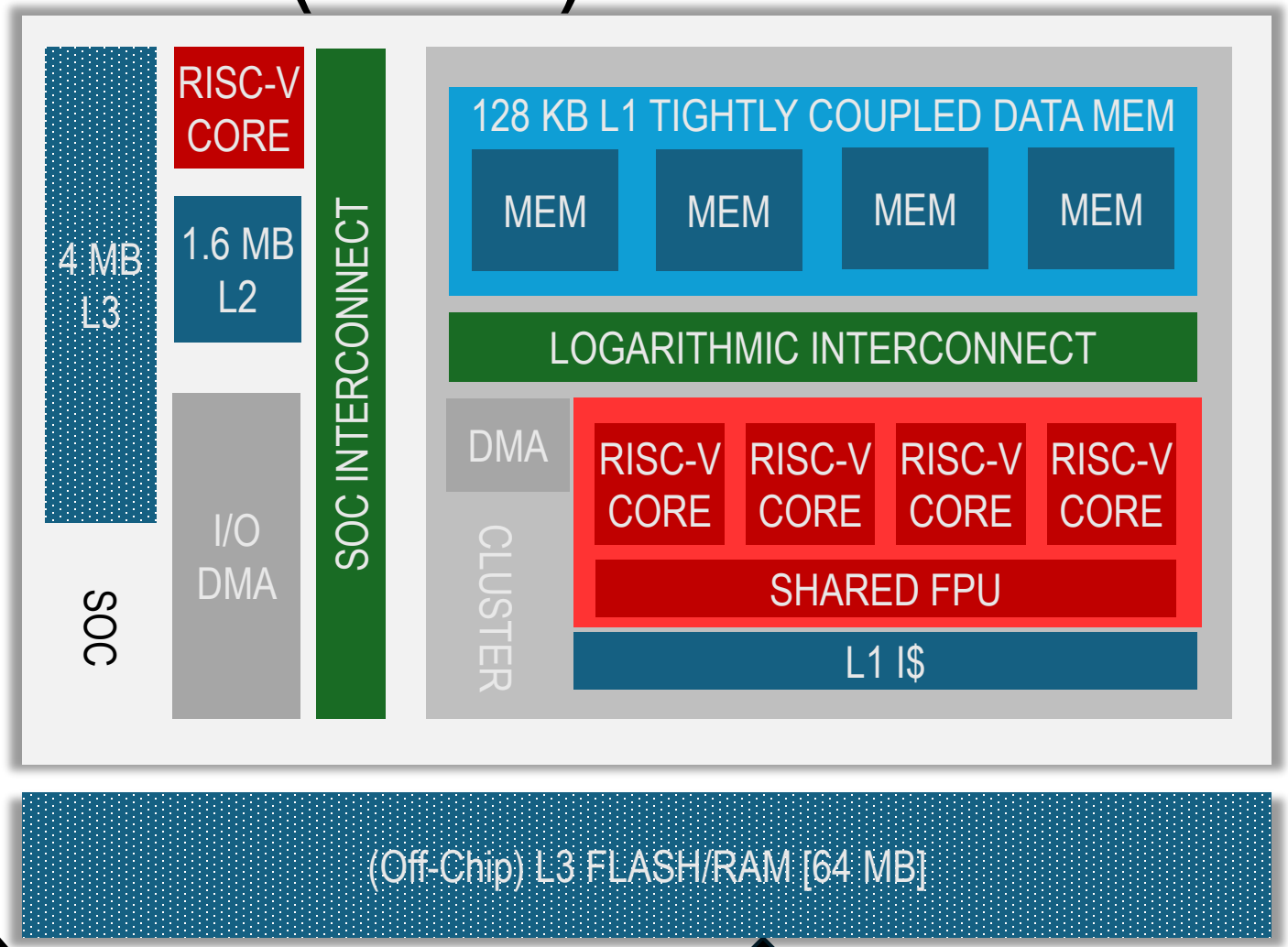
- Hierarchical memory architecture
- Heterogeneous compute units
- PULP SDK
  - PULP toolchain – compile and exploit features
  - PMSIS – PULP MCU Softwar Interface Standard
  - Targets boards/RTL/GVSOC



github.com/pulp-platform/pulp-sdk

Parallel Ultra-Low Power (PULP) Platform

- Hierarchical memory architecture
- Heterogeneous compute units
- PULP SDK & GVSOC
  - Instruction set simulator
    - Timing model
    - Virtual models of devices



github.com/pulp-platform/gvsoc

Accuracy

ODL: The math, the algorithm, and the system

Deep dive into hardware-aware learning

- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server
  - Deploy KWS model
  - Store pre-recorded utterances and labels
- Adapt to new environments
  - Record noise from the environment
  - Augment pre-recorded utterances
  - On-device learning

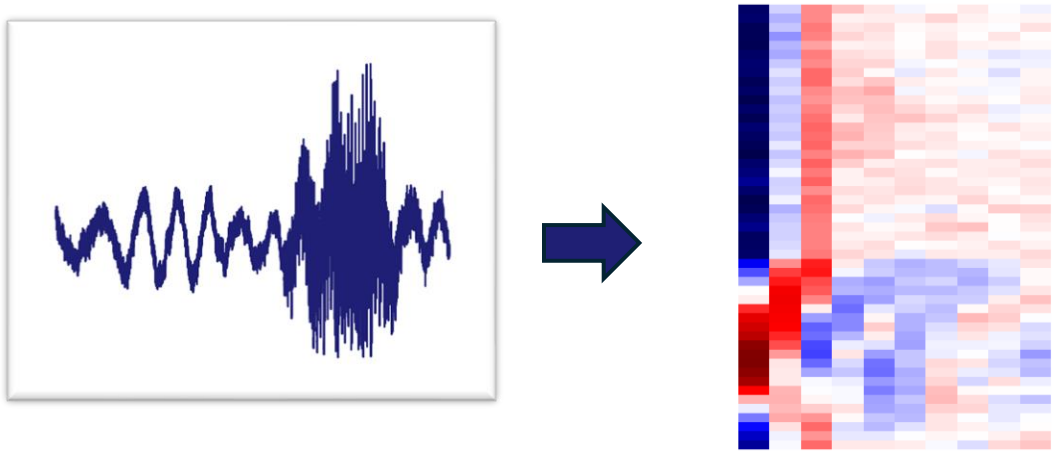
Deep dive into hardware-aware learning

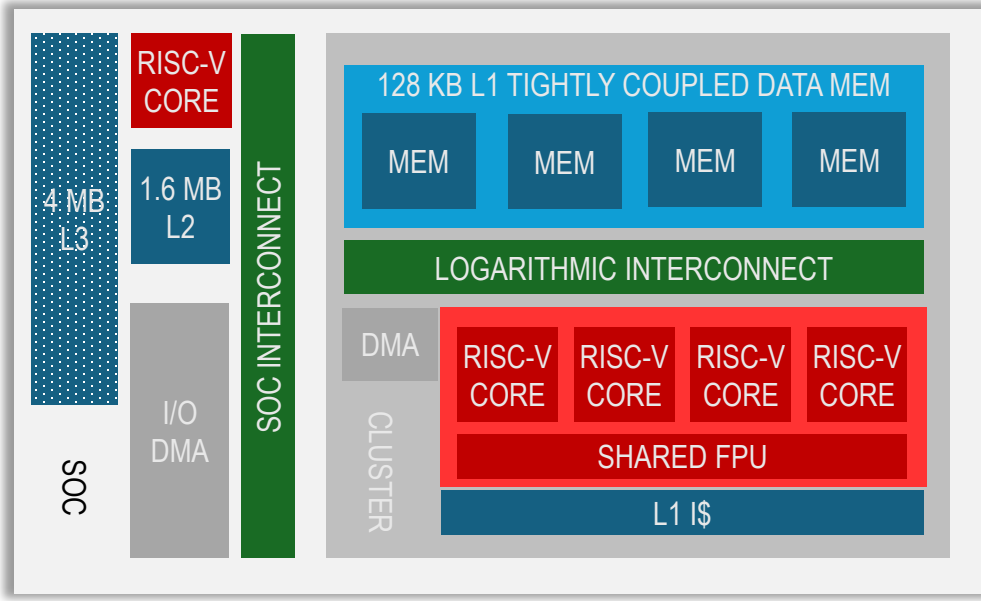
- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server
  - Deploy KWS model
  - Store pre-recorded utterances and labels
- Adapt to new environments
  - Record noise from the environment
  - Augment pre-recorded utterances
  - **On-device learning**

1. **Forward pass – compute the activations**
2. **Backward pass**
   1. **Compute the loss considering the ground truth (pre-recorded)**
   2. **Compute the gradients through *backpropagation***
   3. **Update the parameters**

Forward pass
Backbone features

- Usually preceded by a preprocessing step





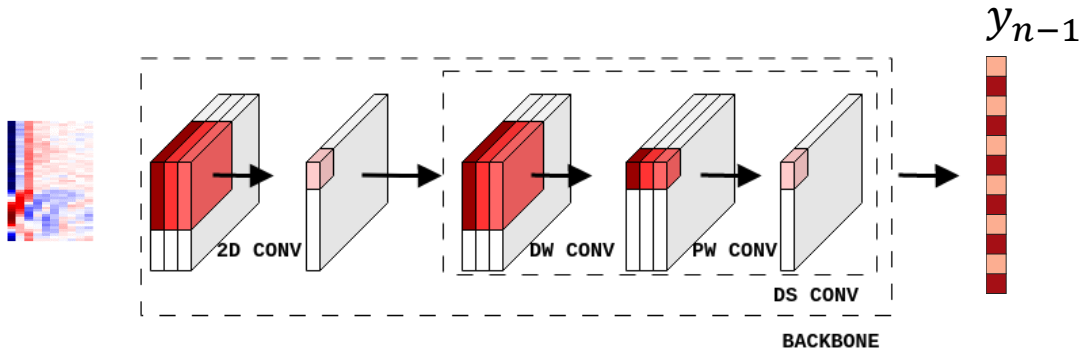
(Off-Chip) L3 FLASH/RAM [64 MB]

ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

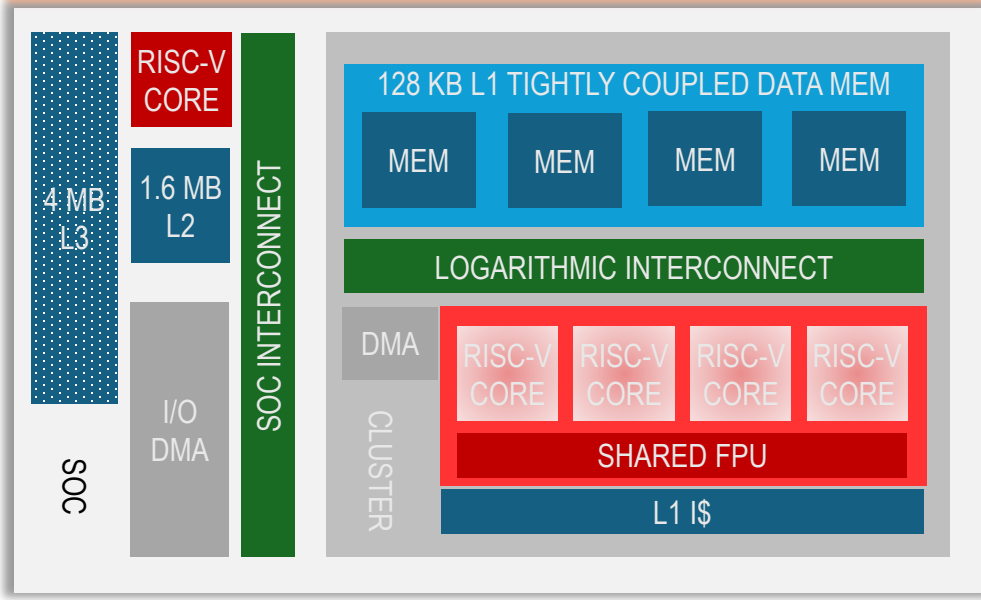
Forward pass
Backbone features

- Forward pass - the neural network approximates a **mapping function**

$$y_{n-1} = f_{n-1}(input)$$




```
pulp_backbone_fp32_fw_cl(&args);
```



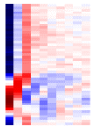
(Off-Chip) L3 FLASH/RAM [64 MB]

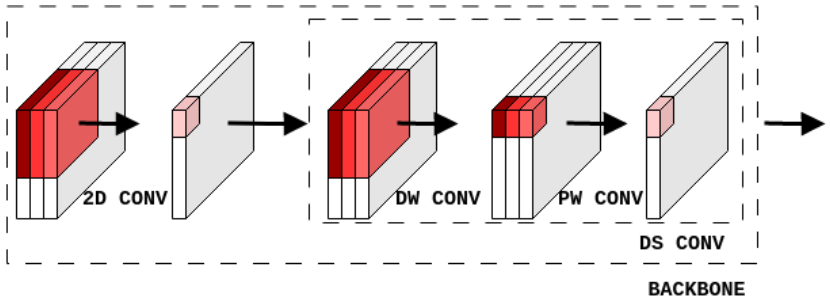
ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

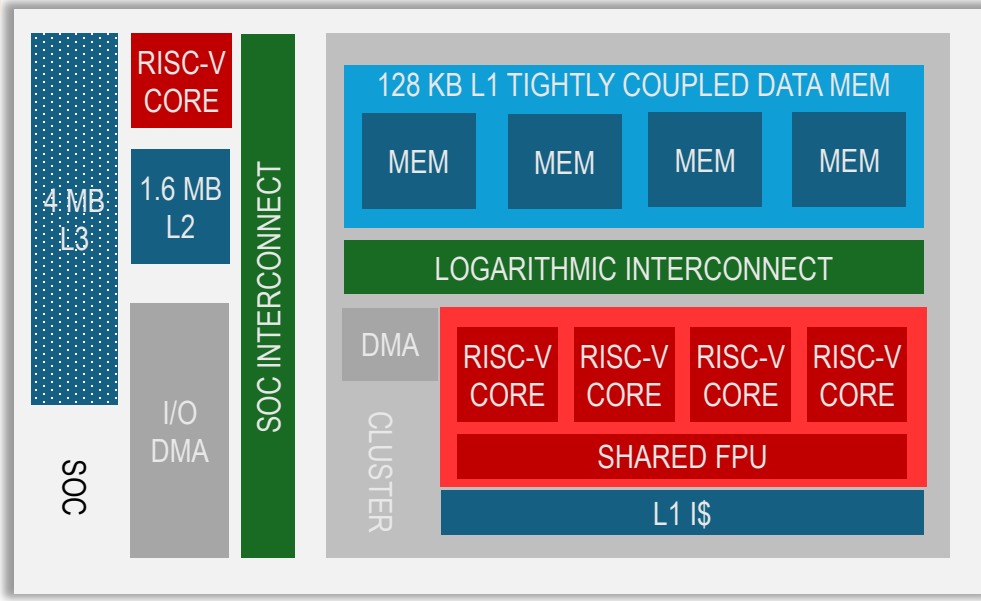
Forward pass
Classifier features

- Forward pass - the neural network approximates a **mapping function**

$$y_{n-1} = f_{n-1}(input)$$
$$y_n = f_n(z_n) = f_n(W_n \cdot x_n + b_n) = f_n(W_n \cdot y_{n-1} + b_n)$$







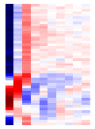
(Off-Chip) L3 FLASH/RAM [64 MB]

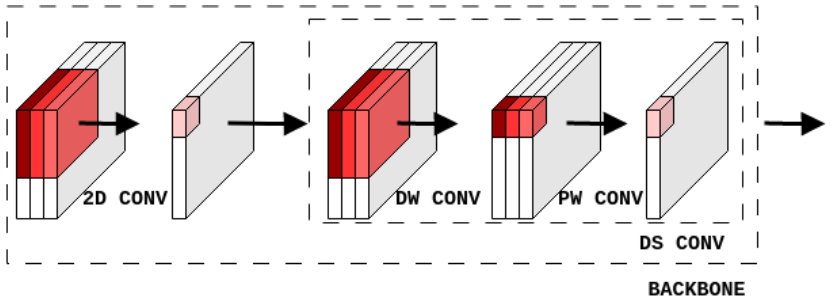
ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

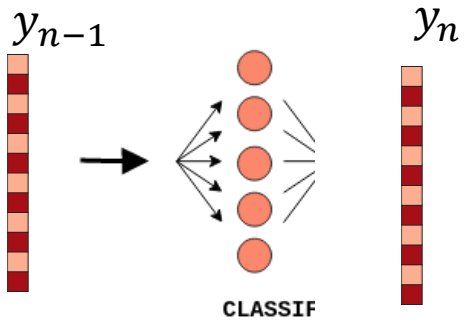
Forward pass
Classifier features

- Forward pass - the neural network approximates a **mapping function**

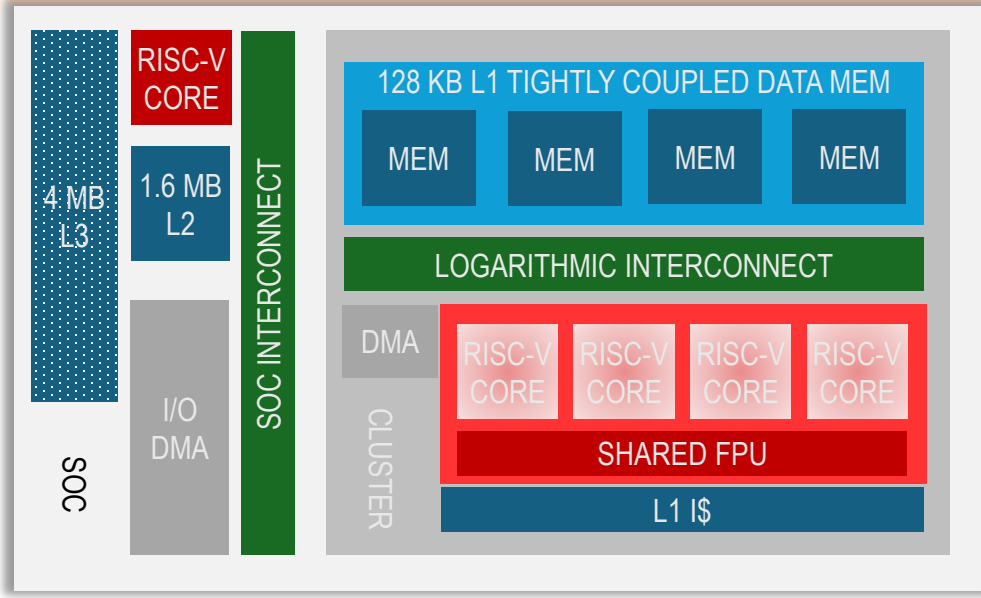
$$y_{n-1} = f_{n-1}(input)$$
$$y_n = f_n(z_n) = f_n(W_n \cdot x_n + b_n) = f_n(W_n \cdot y_{n-1} + b_n)$$








```
pulp_backbone_fp32_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
```



(Off-Chip) L3 FLASH/RAM [64 MB]

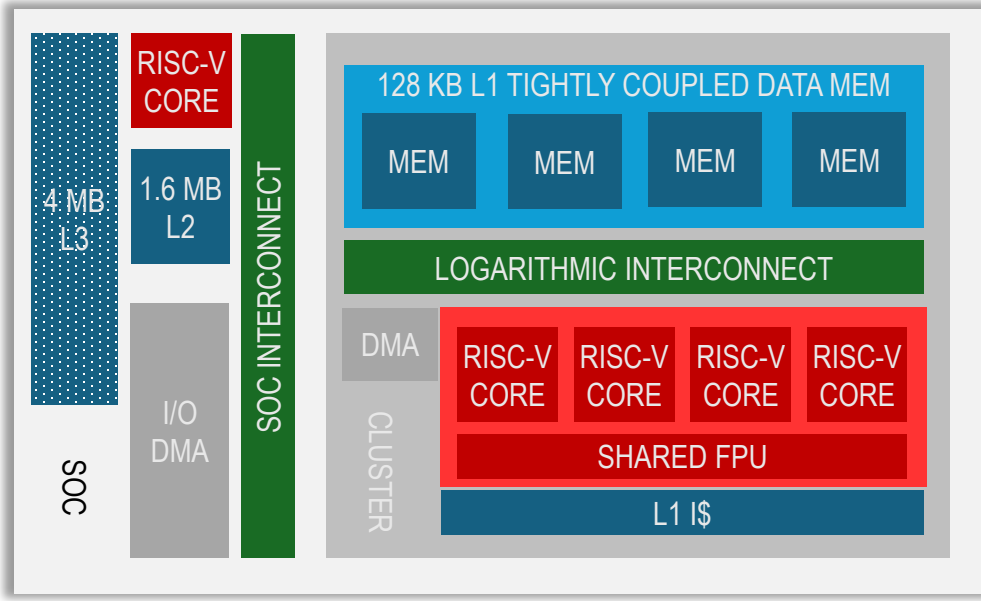
ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

Backward pass
Compute the loss

- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$\min_{input} L(y_n(input), y_{gt})$$

  - where y_{gt} represents the label



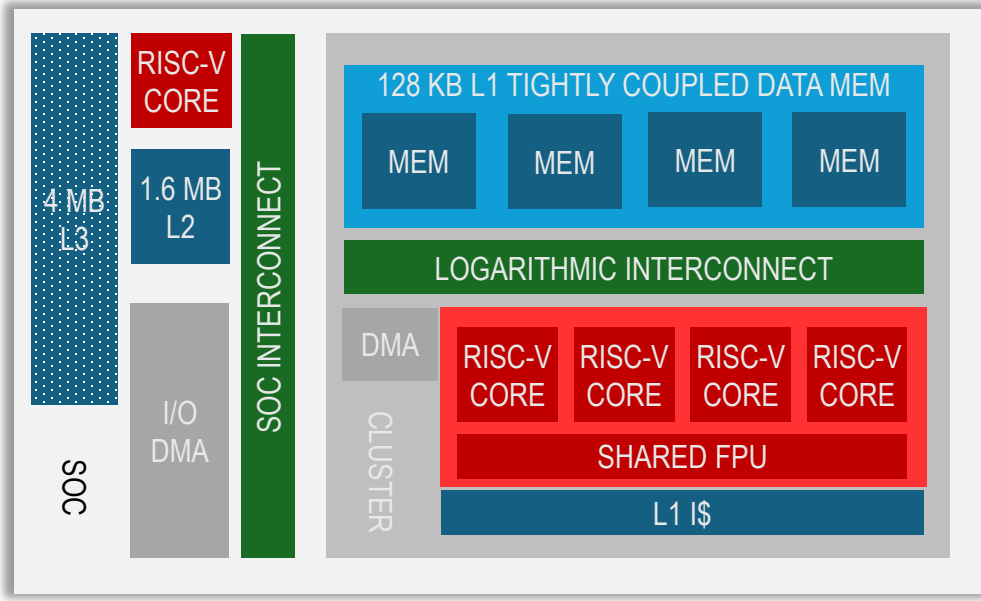
Backward pass
Compute the loss

- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$\min_{input} L(y_n(input), y_{gt})$$

$$let\ L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

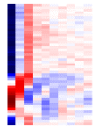
  - where y_{gt} represents the label
  - averaged over S samples

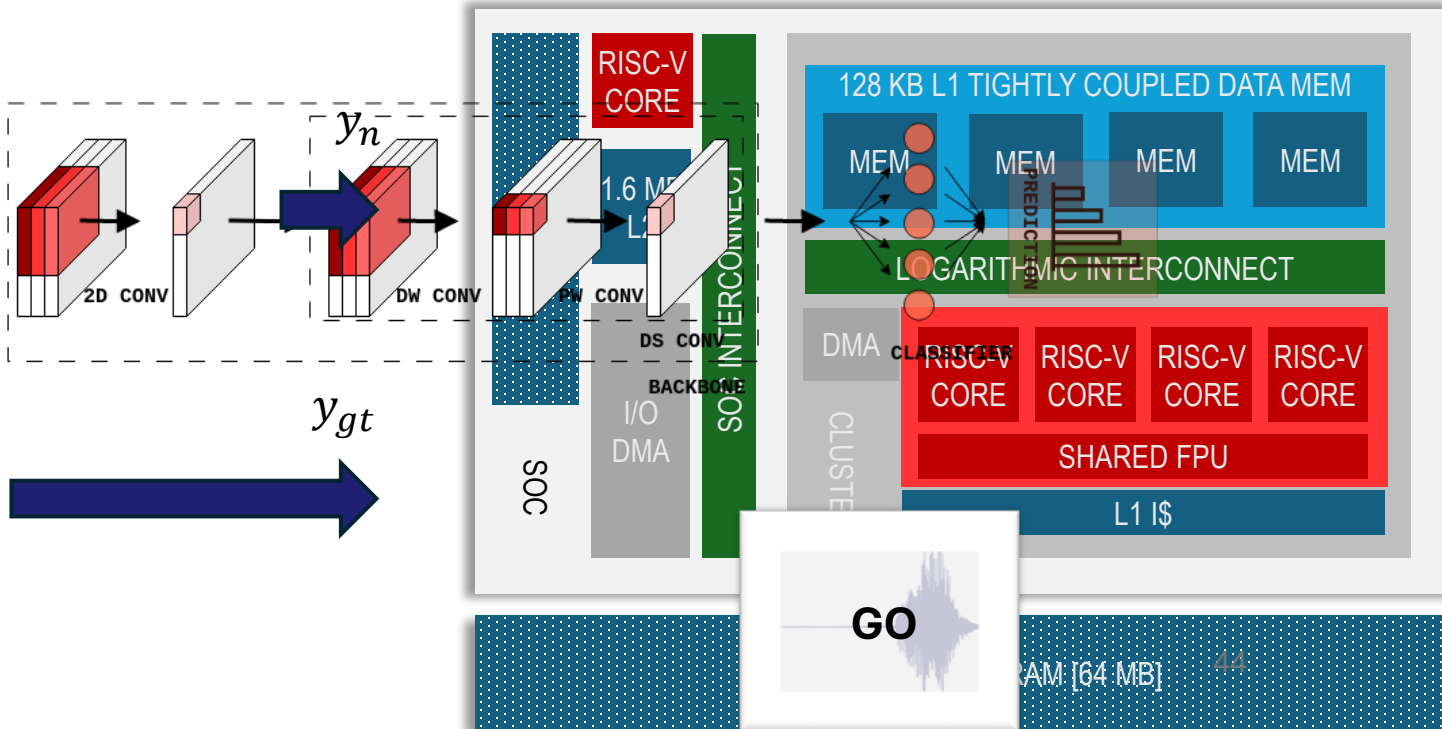


Backward pass
Compute the loss

- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$L_{MSE} = \frac{1}{S}\,(y_n(input) - y_{gt})^2$$



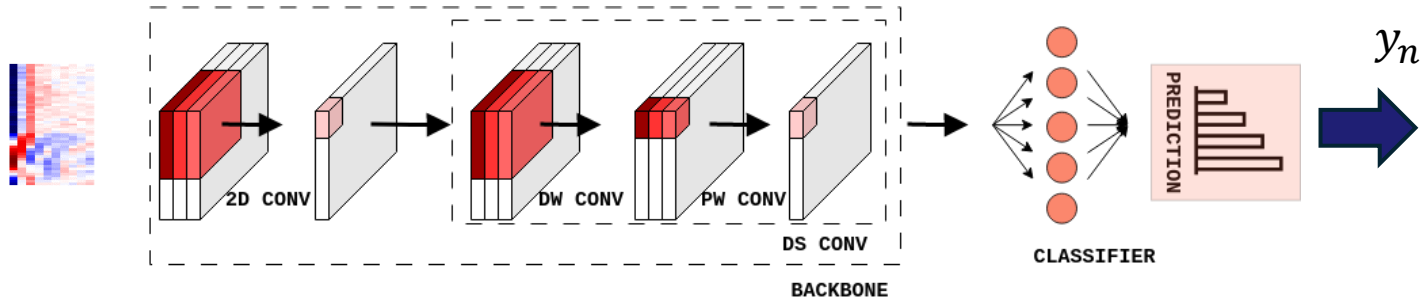


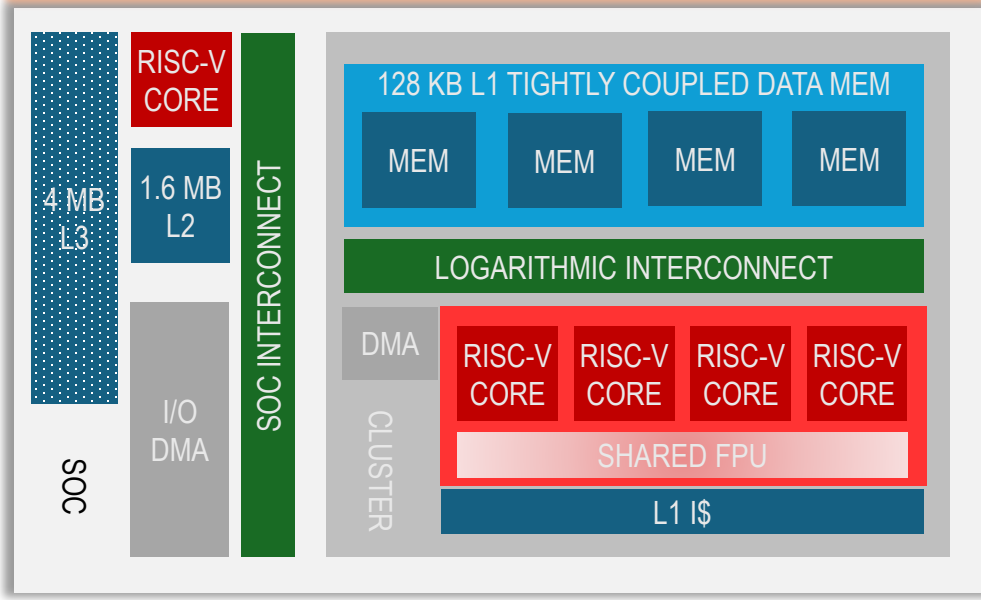
Backward pass
Compute the loss

- Backward pass via backpropagation (**train** model parameters

$$L_{MSE} = \frac{1}{S}\,(y_n(input) - y_{gt})^2$$

```
pulp_backbone_fp32_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
pulp_MSELoss(&loss_args);
```





(Off-Chip) L3 FLASH/RAM [64 MB]

ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

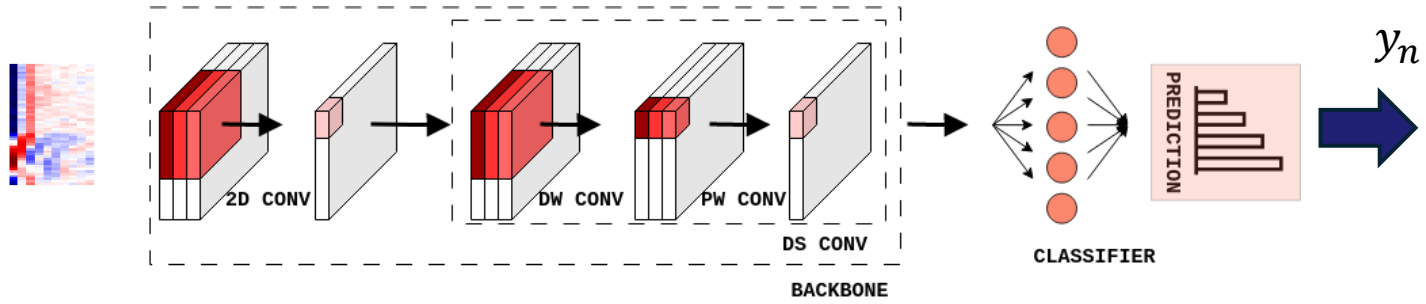
Backward pass
Compute the loss

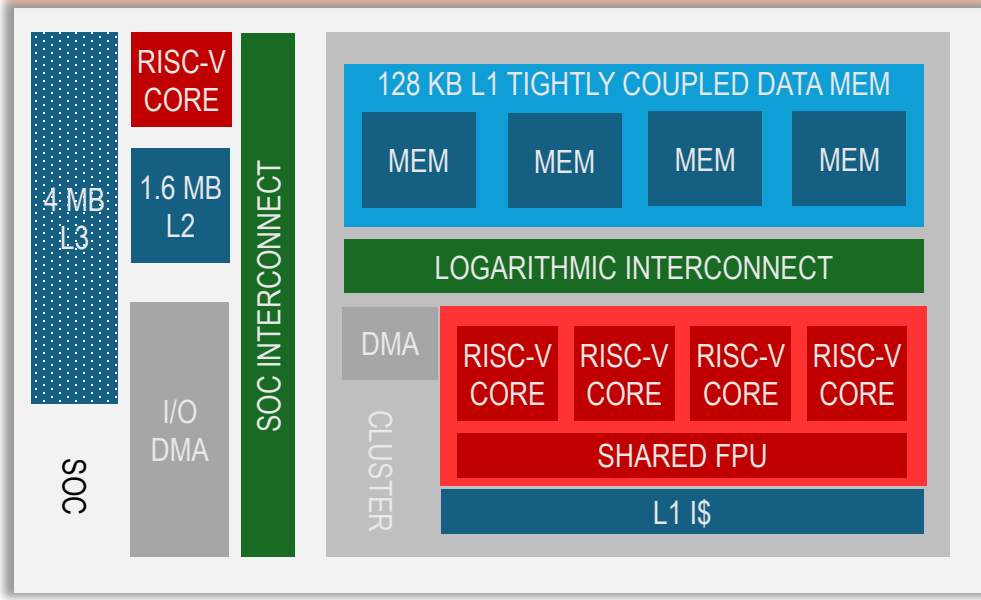
- backpropagation (**train**

Change weights & biases

$$L_{MSE} = \frac{1}{S}\,(y_n(input) - y_{gt})^2$$

```
pulp_backbone_fp32_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
pulp_MSELoss(&loss_args);
```





(Off-Chip) L3 FLASH/RAM [64 MB]

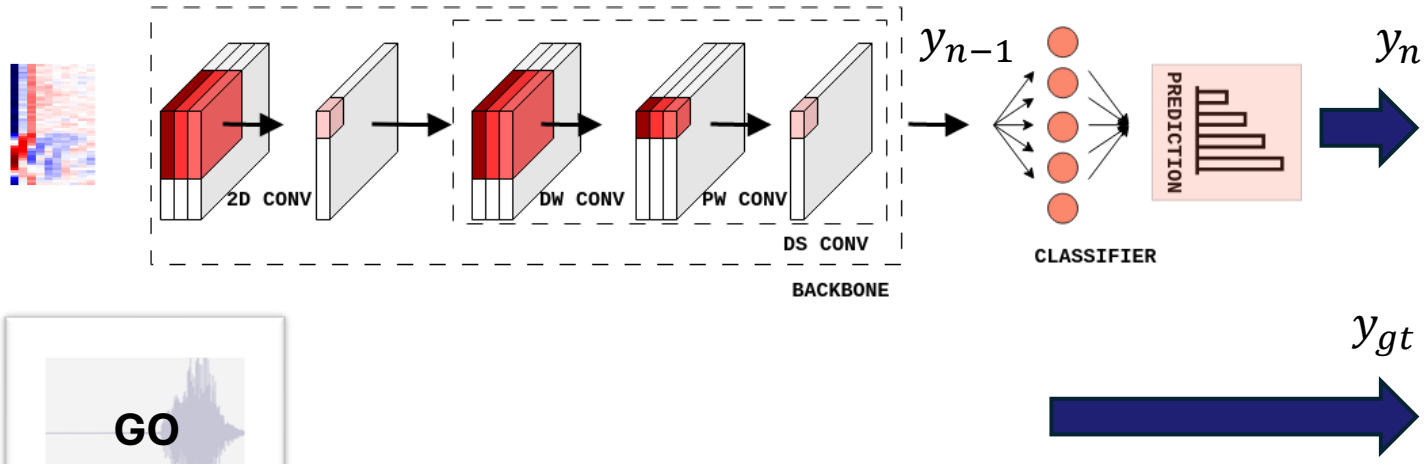
ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

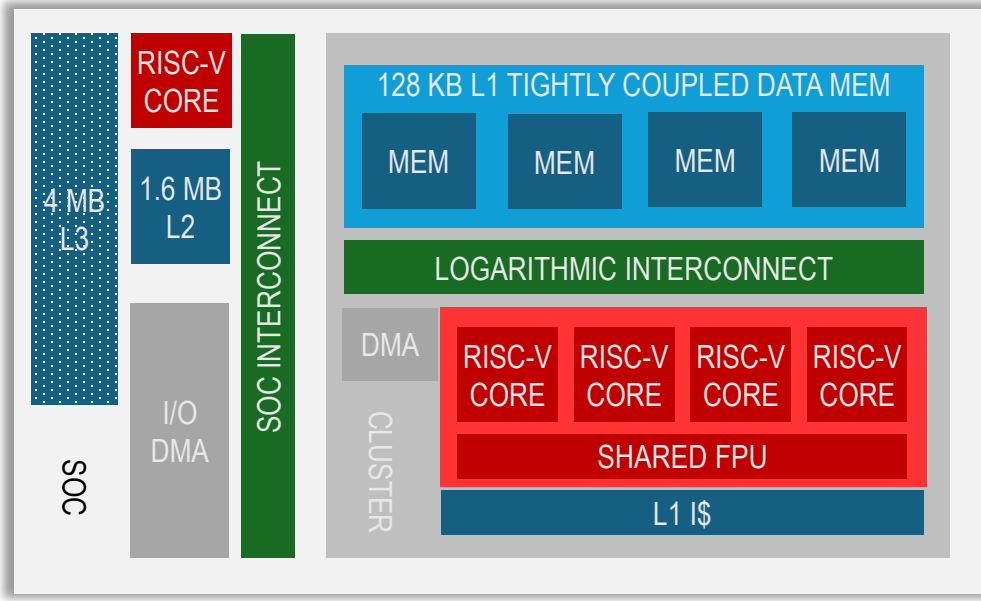
Backward pass
Compute the gradients (backpropagation)

- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$y_n = f_n\left(W_n \cdot y_{n-1} + b_n\right)$$

$$L_{MSE} = \frac{1}{S}\left(y_n(input) - y_{gt}\right)^2$$





(Off-Chip) L3 FLASH/RAM [64 MB]

ETH zürich
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
PULP Parallel Ultra Low Power

Backward pass

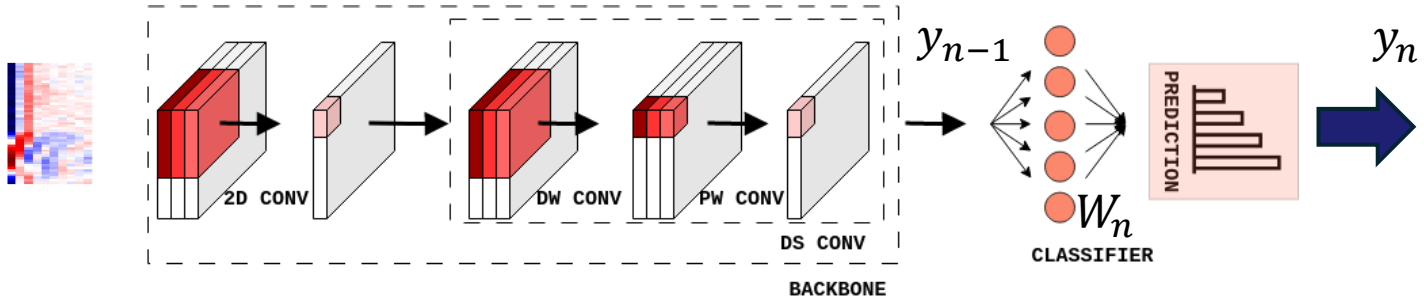
Compu nts (backpropagation)

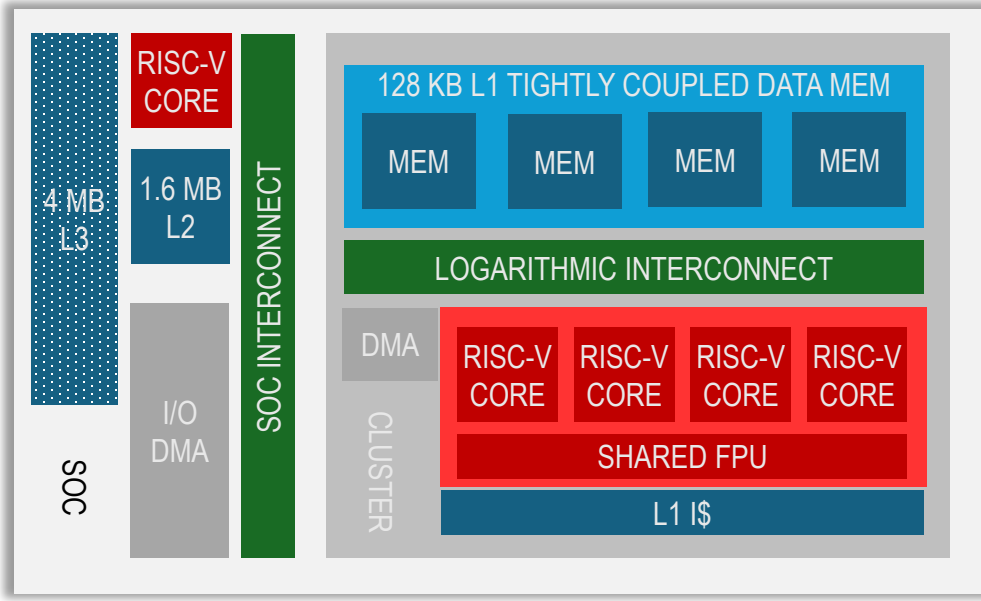
Let us assume no bias

- Backwa ropagation (**training**) – **learning** the model parameters

$$y_n = f_n\,(W_n \cdot y_{n-1} + 0)$$

$$L_{MSE} = \frac{1}{S}\,(y_n(input) - y_{gt})^2$$





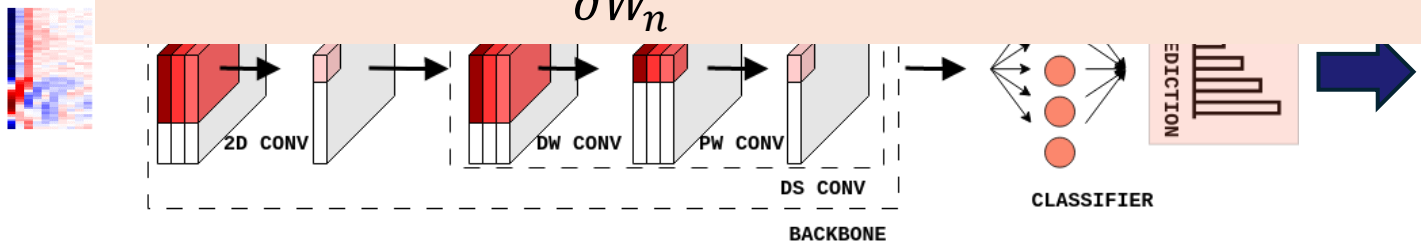
Backward pass
Compute the gradients (backpropagation)

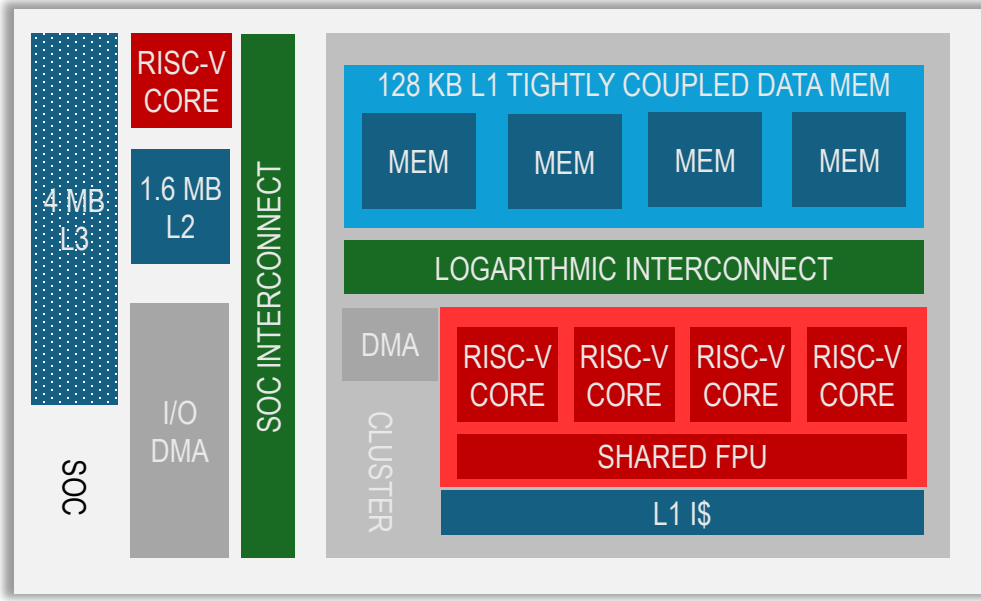
- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$y_n = f_n\,(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}\,(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n}$$





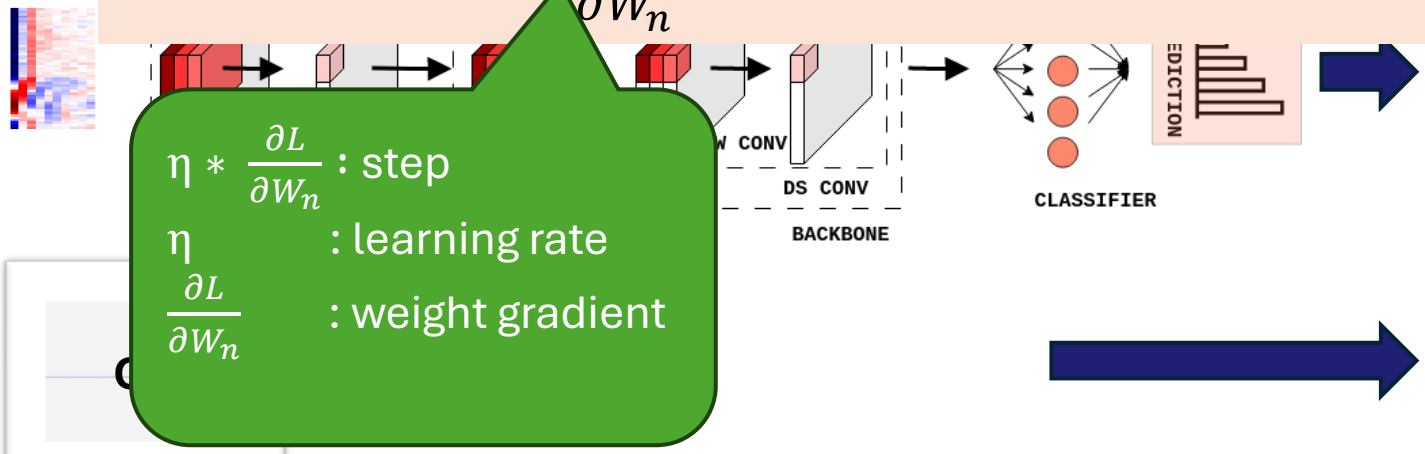
Backward pass
Compute the gradients (backpropagation)

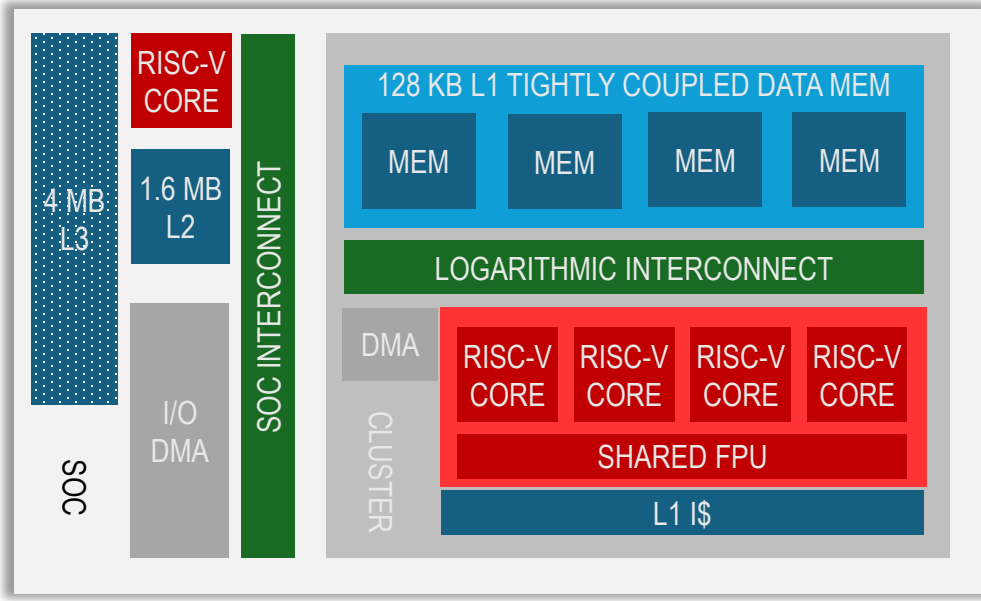
- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$y_n = f_n(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n}$$





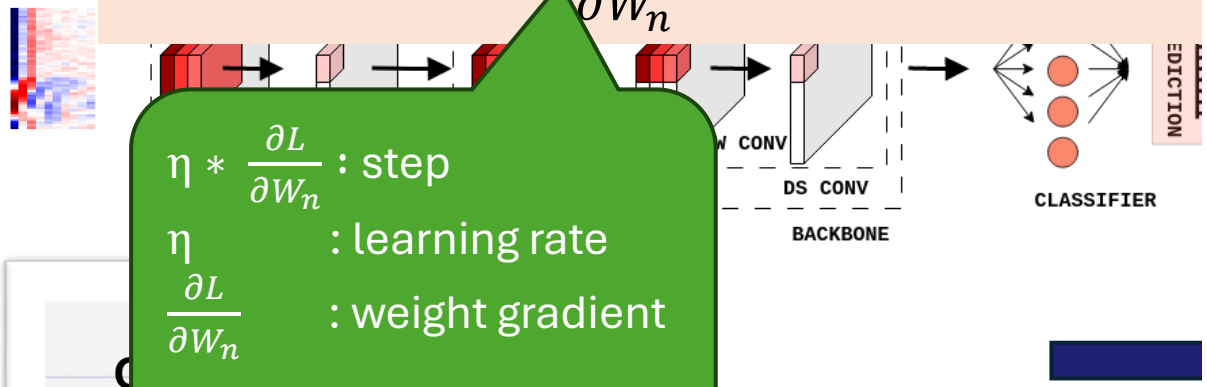
Backward pass
Compute the gradients (backpropagation)

- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$y_n = f_n(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n}$$

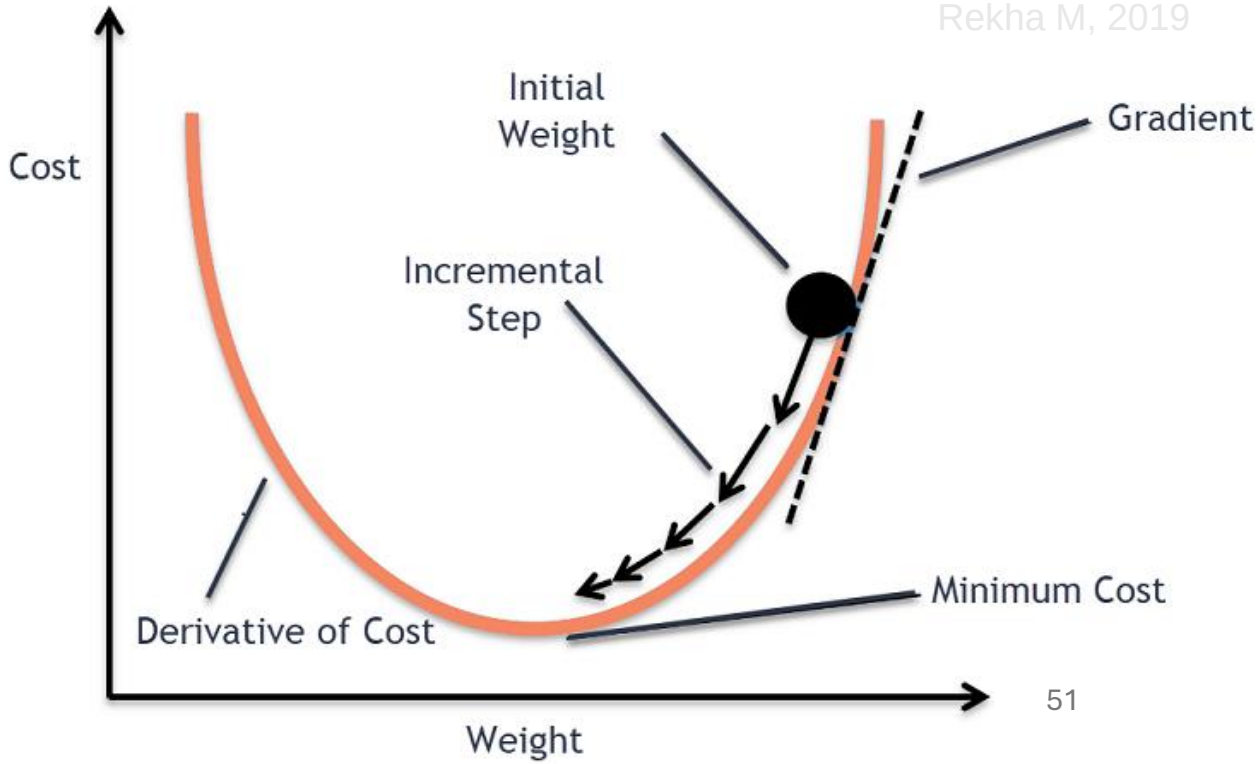


$\eta * \frac{\partial L}{\partial W_n}$: step

η : learning rate

$\frac{\partial L}{\partial W_n}$: weight gradient

Rekha M, 2019



Backward pass
Compute the gradients (backpropagation)

- Backward pass via back … g the model parameters

Chain rule

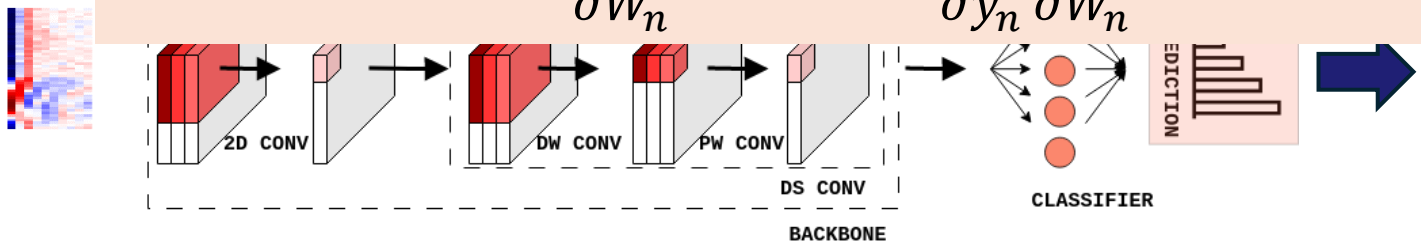
Let $y = f(g(x)), u = g(x)$. Then $y = f(u)$ and

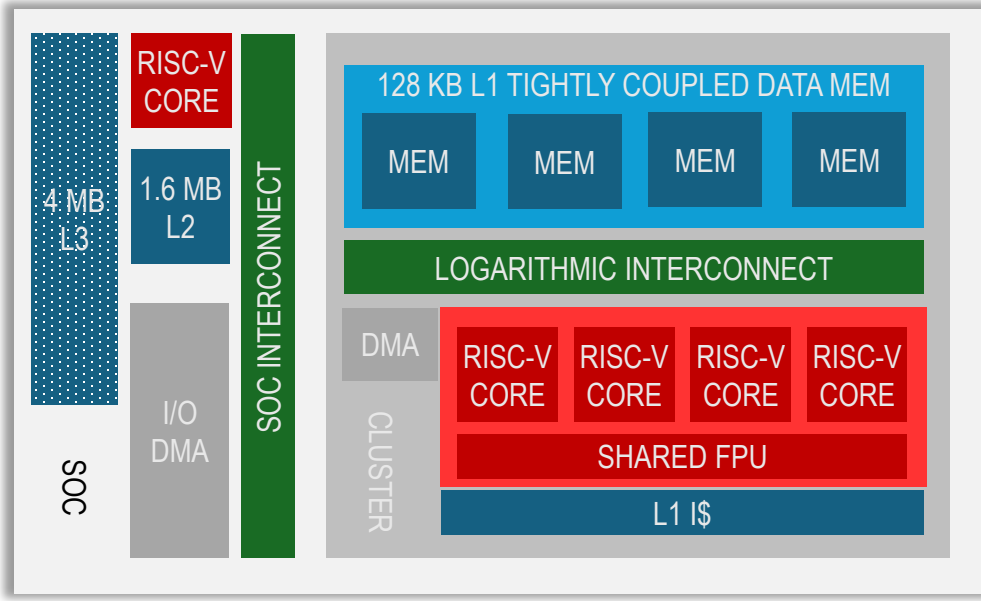
$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

$$y_n = f_n(z_n) = f_n(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$$





Backward pass
Compute the gradients (backpropagation)

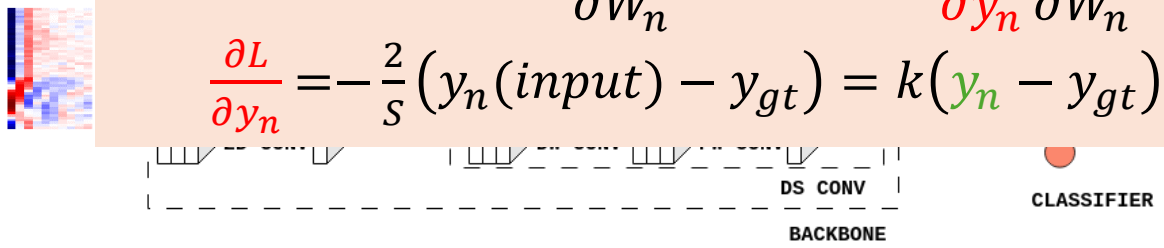
- Backward pass via backpropagation (**training**) – **learning** the model parameters

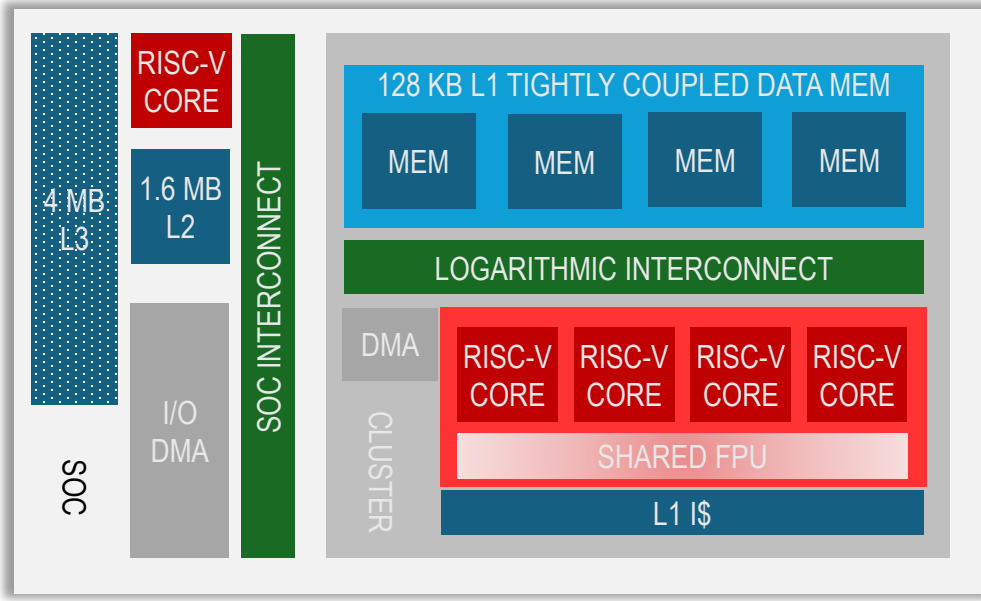
$$y_n = f_n(z_n) = f_n(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$$

$$\frac{\partial L}{\partial y_n} = -\frac{2}{S}(y_n(input) - y_{gt}) = k(y_n - y_{gt})$$





Backward pass
Compute the gradients (backpropagation)

- Backward pass via backpropagation (**training**) – **learning** the model parameters

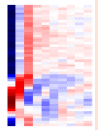
$$y_n = f_n(z_n) = f_n(W_n \cdot y_{n-1})$$

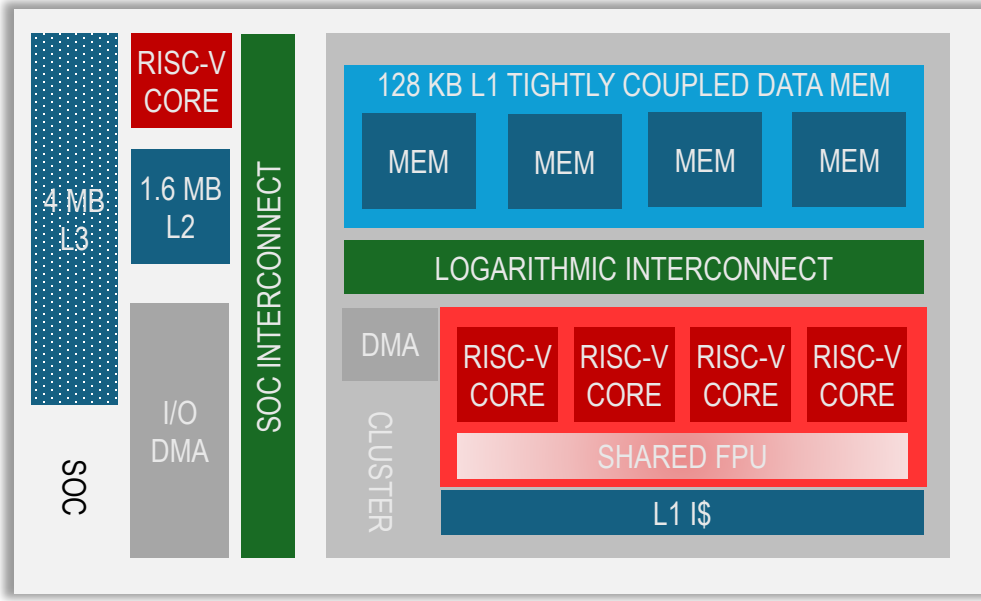
$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$$

$$\frac{\partial L}{\partial y_n} = -\frac{2}{S}(y_n(input) - y_{gt}) = k(y_n - y_{gt})$$

$$\frac{\partial z_n}{\partial W_n} = y_{n-1}$$





Backward pass
Compute the gradients (backpropagation)

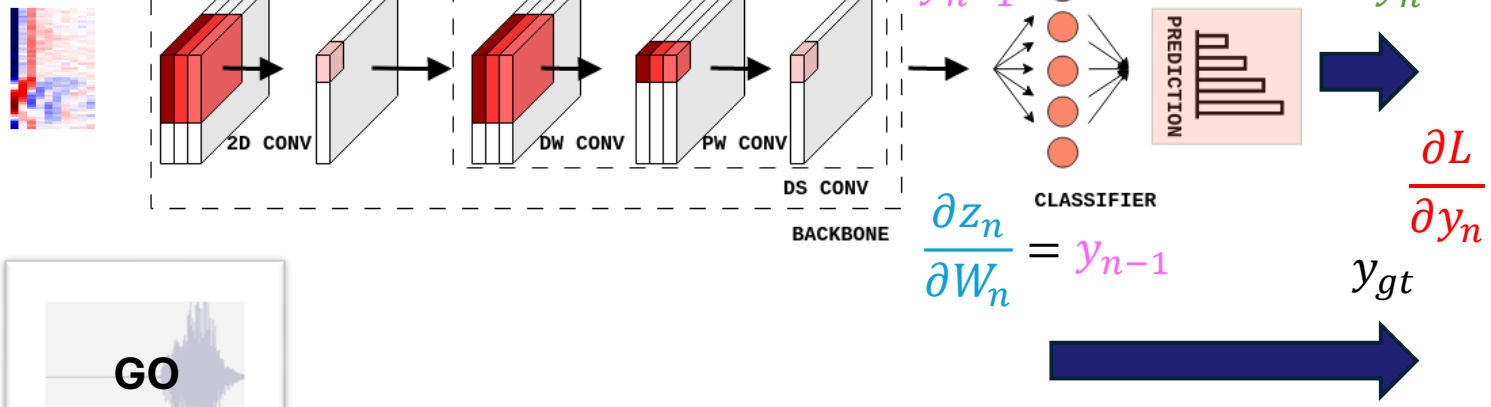
$$y_n = f_n(z_n) = f_n(W_n \cdot y_{n-1})$$

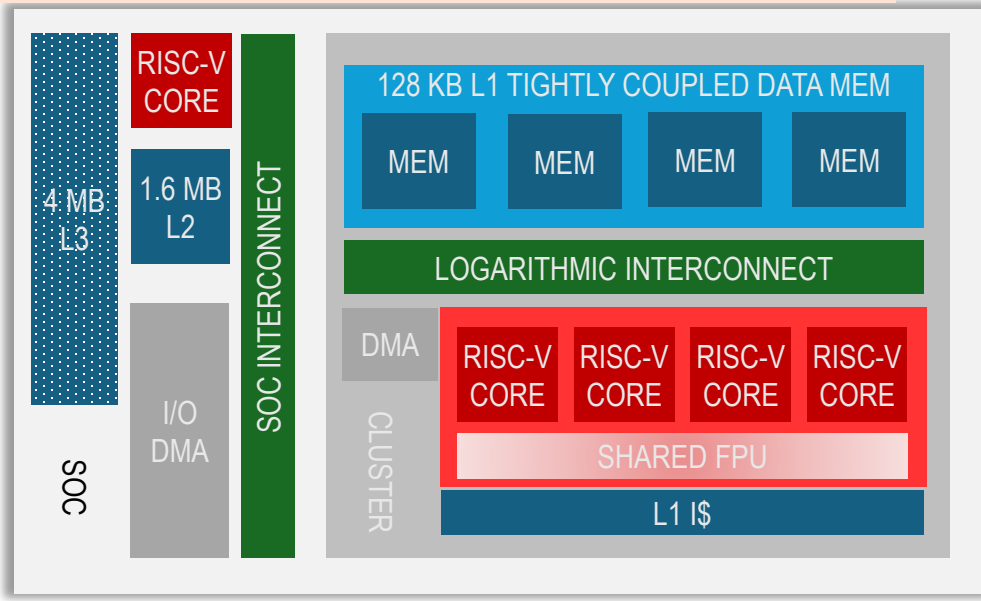
$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$$

$$\frac{\partial L}{\partial y_n} = -\frac{2}{S}(y_n(input) - y_{gt}) = k(y_n - y_{gt})$$

$$\frac{\partial z_n}{\partial W_n} = y_{n-1}$$





Backward pass
Compute the gradients (backpropagation)

- Backward pass via backpropagation (**training**) – **learning** the model parameters

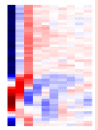
$$y_n = f_n(z_n) = f_n(W_n \cdot y_{n-1})$$

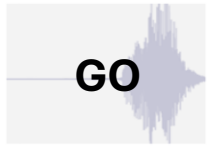
$$L_{MSE} = \frac{1}{n}(y_n(input) - y_{gt})^2$$

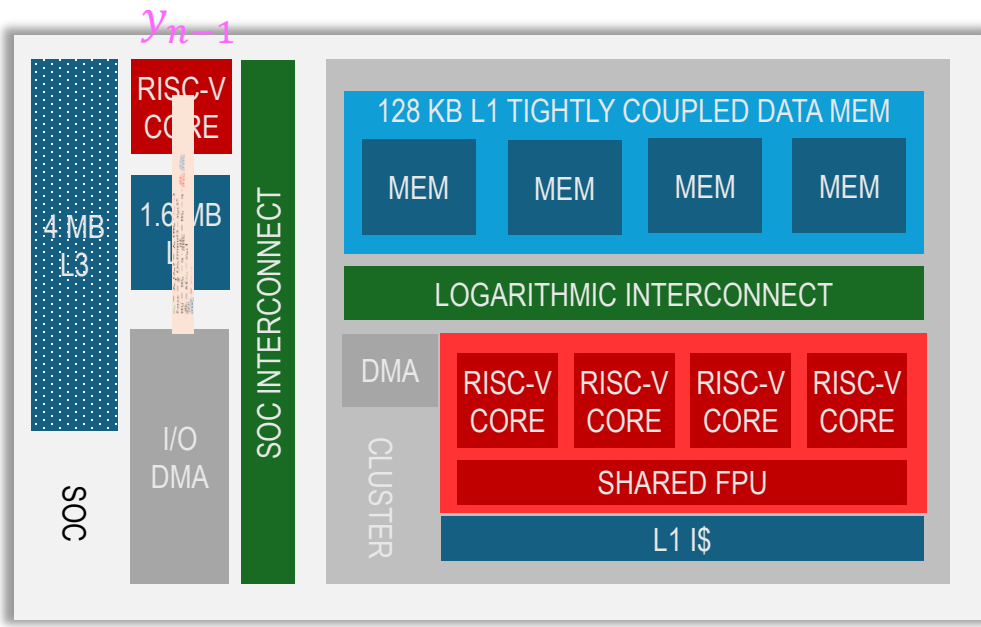
$$W_n = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$$

$$\frac{\partial z_n}{\partial W_n} = y_{n-1}$$

Keep the activations in the memory







(Off-Chip) L3 FLASH/RAM [64 MB]

ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

Backward pass
Compute the gradients (backpropagation)

- Backward pass via backpropagation (**train** model parameters

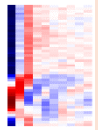
$$y_n = f_n(z_n) = f_n(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

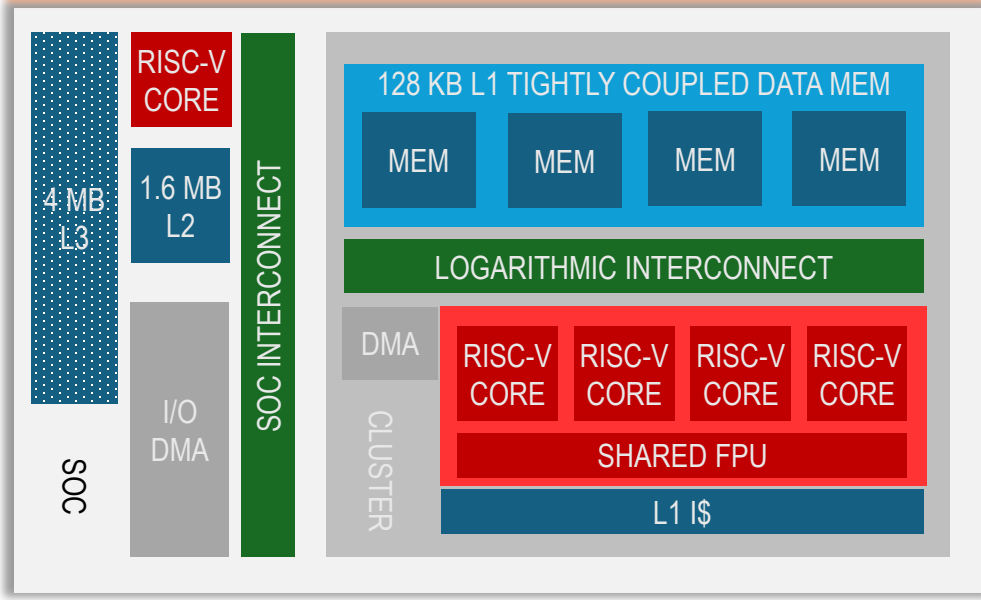
$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$$

$$\frac{\partial L}{\partial y_n} = -\frac{2}{S}(y_n(input) - y_{gt}) = k(y_n - y_{gt})$$

$$\frac{\partial z_n}{\partial W_n} = y_{n-1}$$




```
pulp_backbone_fp32_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
pulp_MSELoss(&loss_args);
pulp_linear_fp32_bw_cl(&l1_args);
```



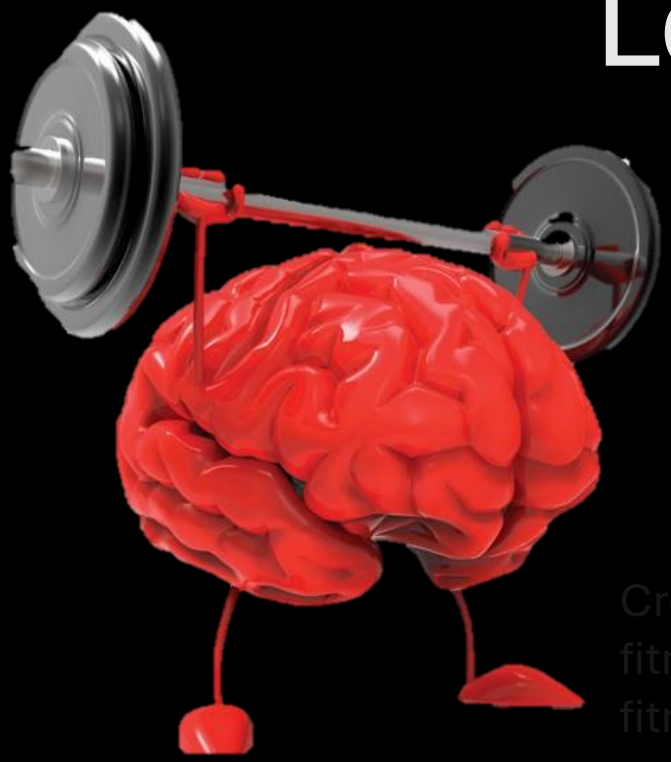
(Off-Chip) L3 FLASH/RAM [64 MB]

ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

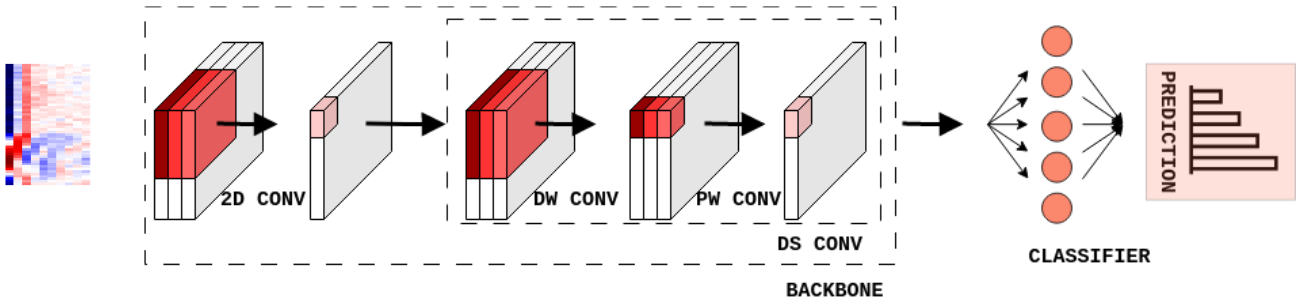
Let’s try now something more interesting



Credit: https://www.swspotlight.com/articles/health-and-fitness/stay-fit-students-flex-their-mental-muscles-in-brain-fitness-classes/

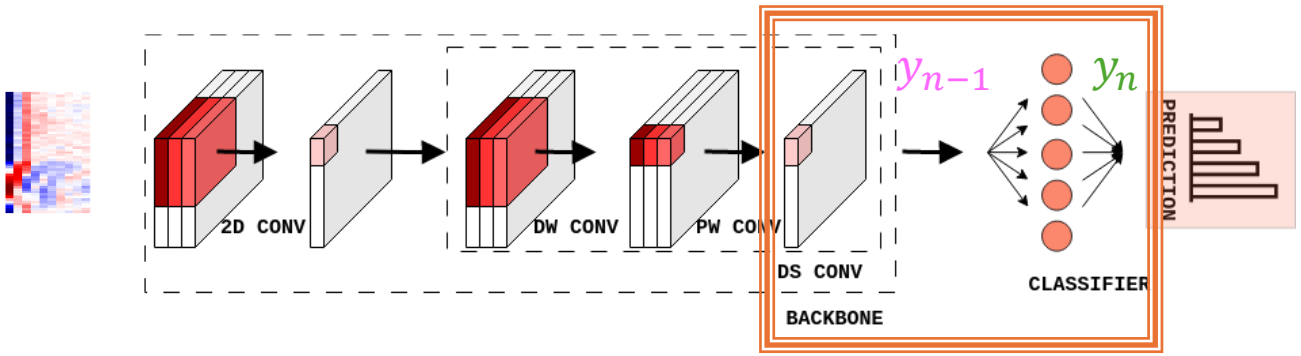
What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1})$$



What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1}) = f_n\big(W_n \cdot f_n(W_{n-1} * y_{n-2})\big)$$

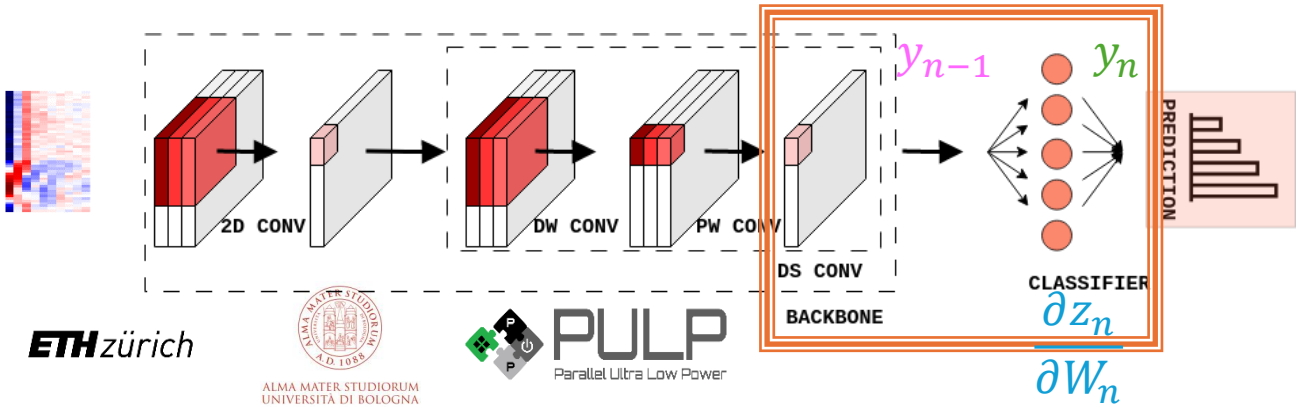


What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1}) = f_n\left(W_n \cdot f_n(W_{n-1} * y_{n-2})\right)$$

$$L_{MSE} = \frac{1}{S}\left(y_n(input) - y_{gt}\right)^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$$



ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

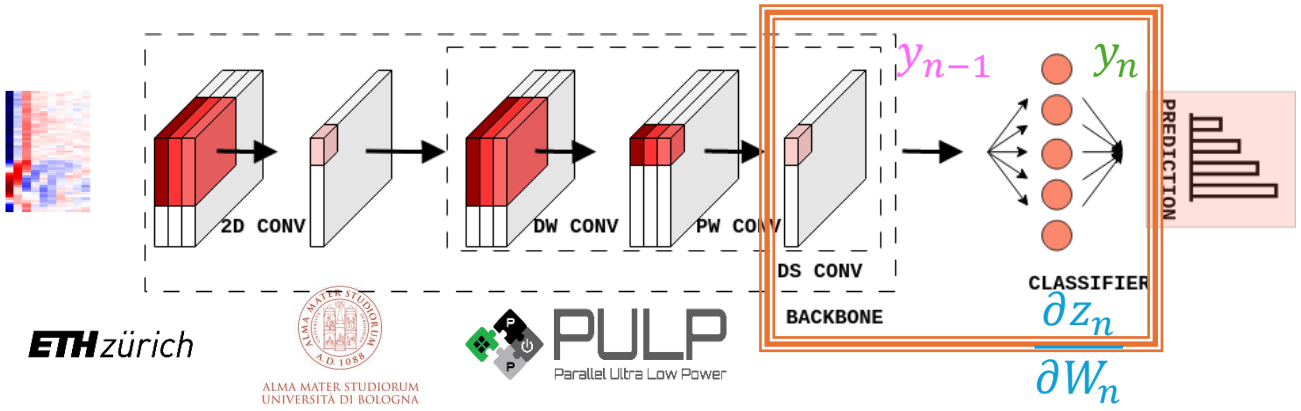
PULP Parallel Ultra Low Power

What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1}) = f_n\left(W_n \cdot f_n(W_{n-1} * y_{n-2})\right)$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$



What if we increase the update depth?

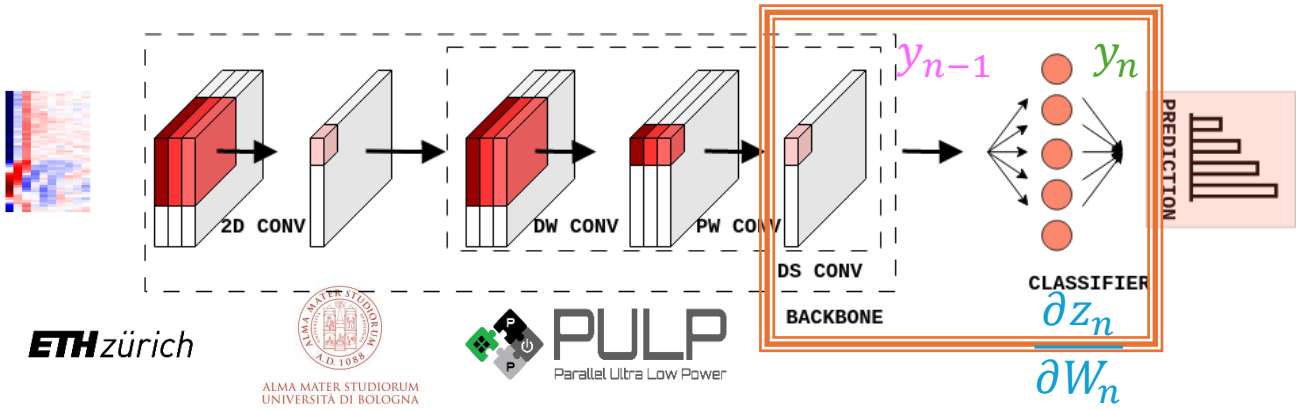
$$y_n = f_n(W_n \ldots) = f(W \ldots f(W_{n-1} * y_{n-2}))$$

$$L_{MSE} = \frac{1}{S}(\ldots$$

Weight gradients

$$W_n' = W_n - \ldots \frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt}) y_{n-1}$$

$$W_{n-1}' = W_{n-1} - \eta \cdot \frac{\partial L}{\partial W_{n-1}}$$



ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

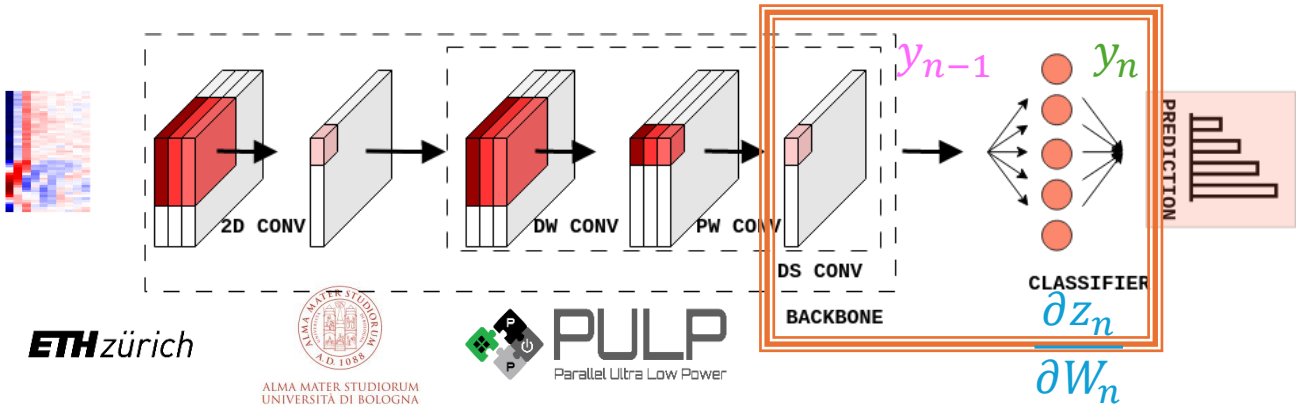
What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1}) = f_n\left(W_n \cdot f_n(W_{n-1} * y_{n-2})\right)$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$

$$W_{n-1}' = W_{n-1} - \eta \cdot \frac{\partial L}{\partial W_{n-1}} = W_{n-1} - \eta \cdot \frac{\partial L}{\partial y_{n-1}}\frac{\partial z_{n-1}}{\partial W_{n-1}}$$



ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

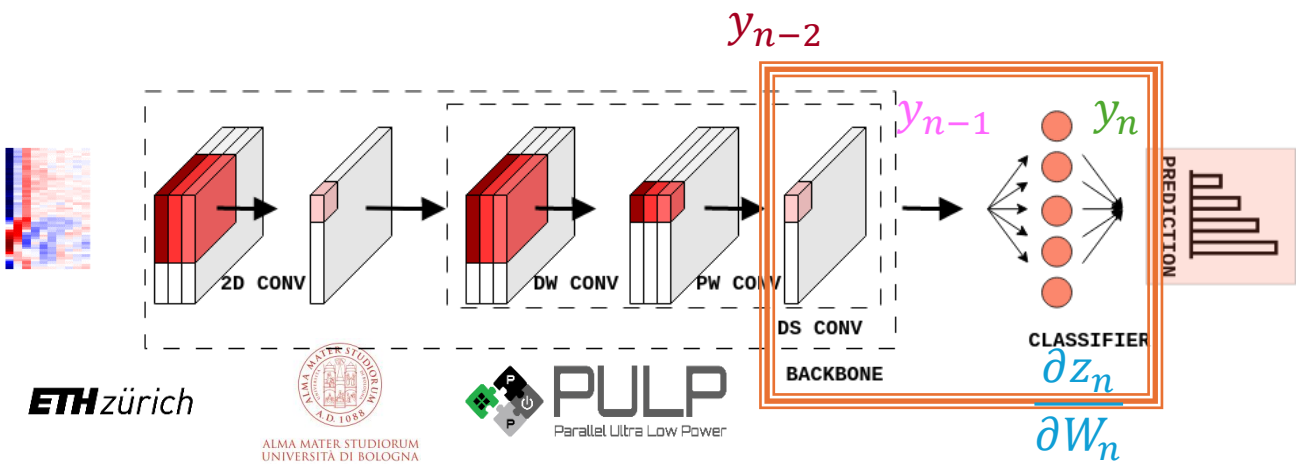
What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1}) = f_n\left(W_n \cdot f_n(\boldsymbol{W_{n-1}} * y_{n-2})\right)$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$

$$W_{n-1}' = W_{n-1} - \eta \cdot \frac{\partial L}{\partial W_{n-1}} = W_{n-1} - \eta \cdot \frac{\partial L}{\partial y_{n-1}}\frac{\partial z_{n-1}}{\boldsymbol{\partial W_{n-1}}}$$



ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

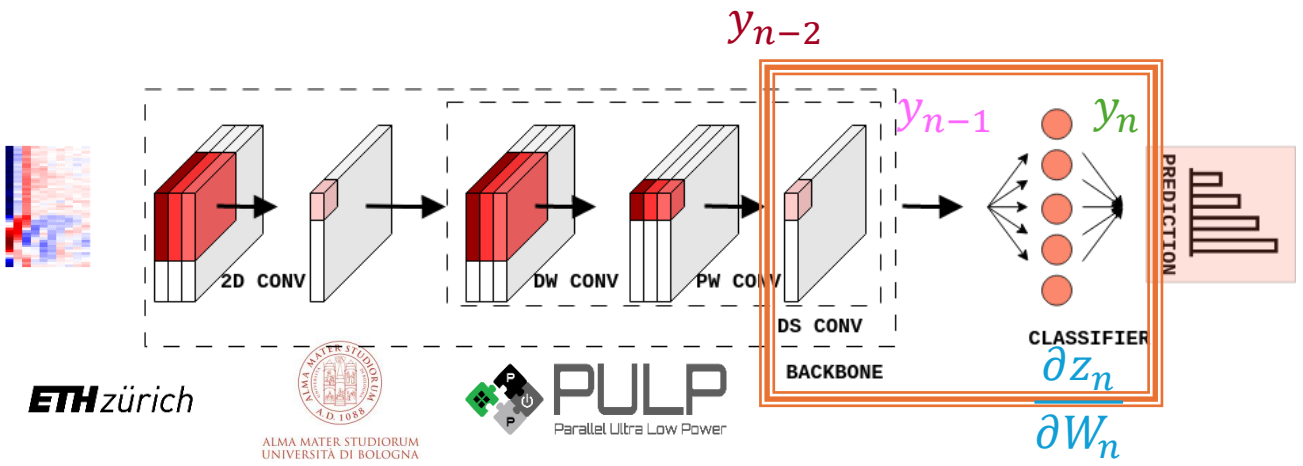
What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1}) = f_n\big(W_n \cdot f_n(W_{n-1} * y_{n-2})\big)$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$

$$W_{n-1}' = W_{n-1} - \eta \cdot \frac{\partial L}{\partial W_{n-1}} = W_{n-1} - \eta \cdot \frac{\partial L}{\partial y_{n-1}}y_{n-2}$$



ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

What if we increase the update depth?

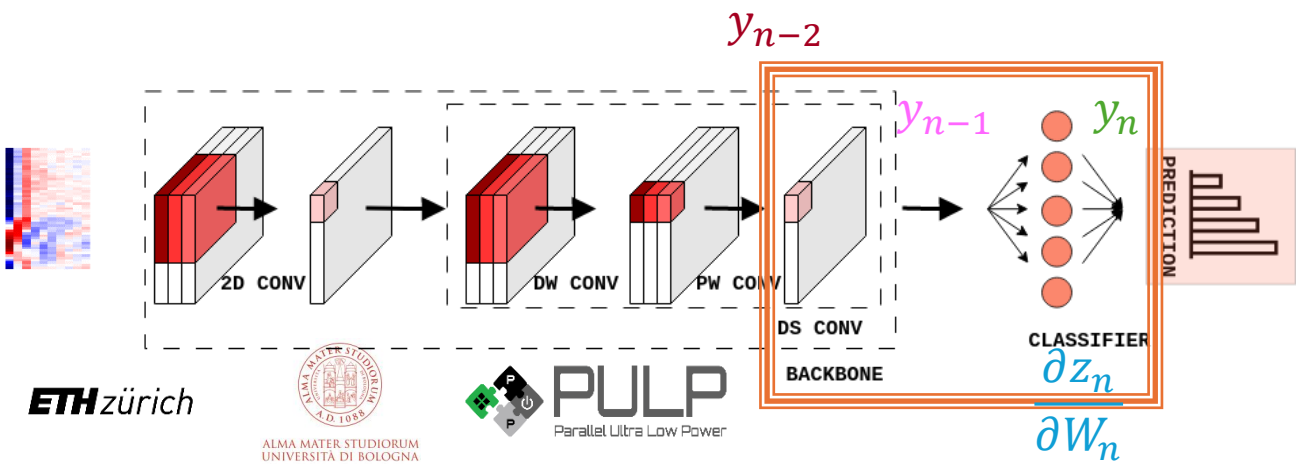
$$y_n = f_n(W_n \cdot y_{n-1}) = f_n\big(W_n \cdot f_n(W_{n-1} \cdot y_{n-2})\big)$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial y_n}{\partial W_n} = W_n - \eta \cdot 2(y_n - y_{gt})y_{n-1}$$

Input gradients

$$W_{n-1}' = W_{n-1} - \eta \cdot \frac{\partial L}{\partial W_{n-1}} = W_{n-1} - \eta \cdot \frac{\partial L}{\partial y_{n-1}}y_{n-2}$$



ETH zürich

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

What if we increase the update depth?

$$y_n = f_n(W_n \cdot y_{n-1}) = f_n(W_n * f_{n-1}(W_{n-1} * y_{n-2}))$$

$$L_{MSE} = \frac{1}{c}(y_n(i$$

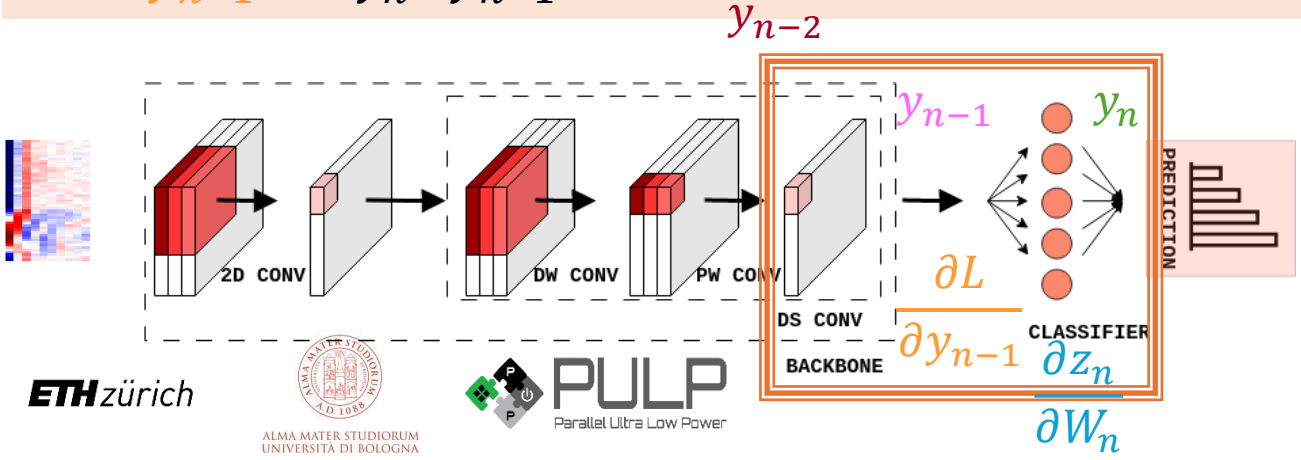
$$\eta \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$

$$-1 - \eta \cdot \frac{\partial L}{\partial W_{n-1}} = W_{n-1} - \eta \cdot \frac{\partial L}{\partial y_{n-1}} y_{n-2}$$

$$\frac{\partial L}{\partial y_{n-1}} = \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial y_{n-1}}$$

Input gradients

Weight gradients



What if we increase the update depth?

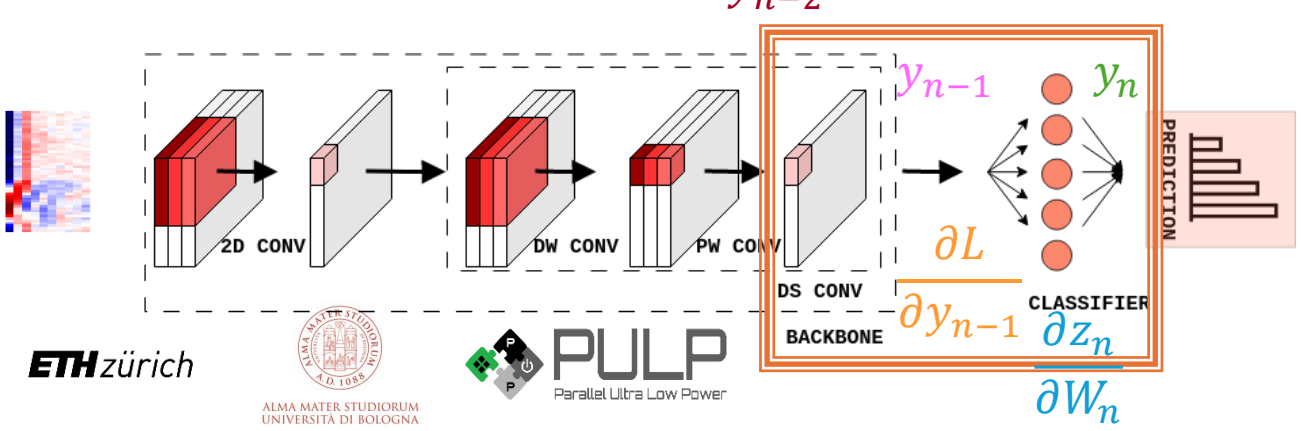
$$y_n = f_n(\boldsymbol{W_n} \cdot y_{n-1}) = f_n(W_n \cdot f_n(W_{n-1} * y_{n-2}))$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$

$$W_{n-1}' = W_{n-1} - \eta \cdot \frac{\partial L}{\partial W_{n-1}} = W_{n-1} - \eta \cdot \frac{\partial L}{\partial y_{n-1}} y_{n-2}$$

$$\frac{\partial L}{\partial y_{n-1}} = \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial y_{n-1}} = \frac{\partial L}{\partial y_n}\boldsymbol{W_n}$$



What if we increase the update depth?

$$y_n = f_n(\boldsymbol{W_n} \cdot y_{n-1}) = f_n\big(W_n \cdot f_n(W_{n-1} * y_{n-2})\big)$$

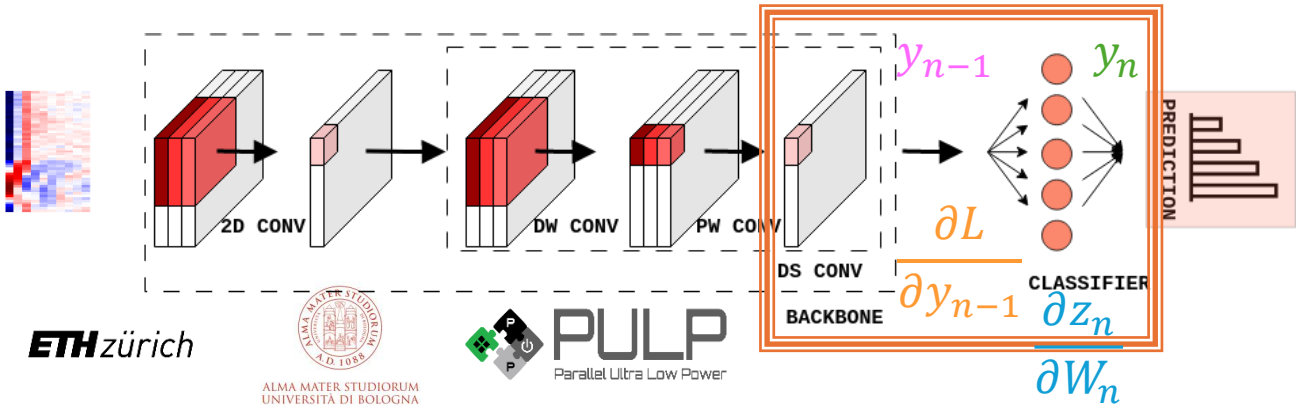
$$L = \frac{1}{2}(y_n(input) - y_{gt})^2$$

$$W_n = W_n - \eta \cdot \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$

$$\frac{\partial L}{\partial W_{n-1}} = W_{n-1} - \eta \cdot \frac{\partial L}{\partial y_{n-1}} y_{n-2}$$

$$\frac{\partial L}{\partial y_{n-1}} = \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial y_{n-1}} = \frac{\partial L}{\partial y_n}\boldsymbol{W_n}$$

Keep the gradients in the memory



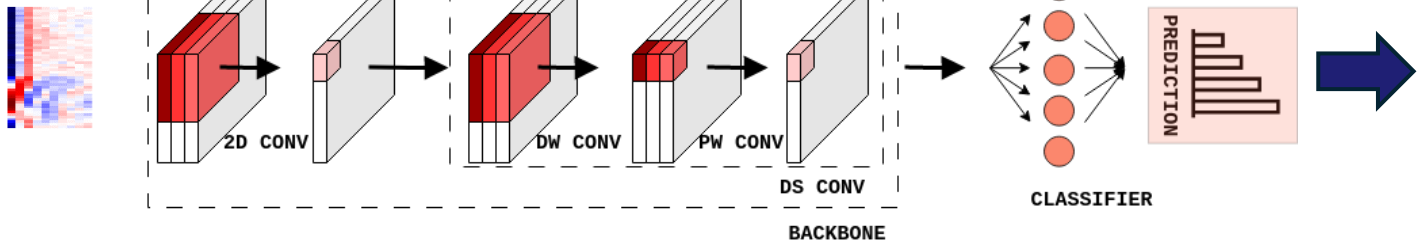
Backward pass
Update the weights

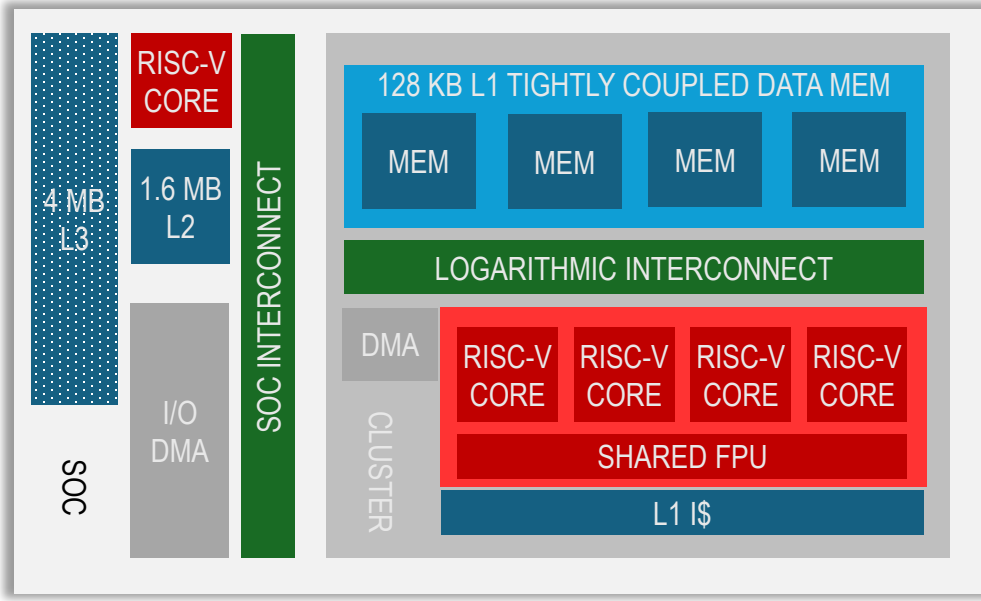
- Backward pass via backpropagation (**training**) – **learning** the model parameters

$$y_n = f_n(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$





Backward pass
Update the weights

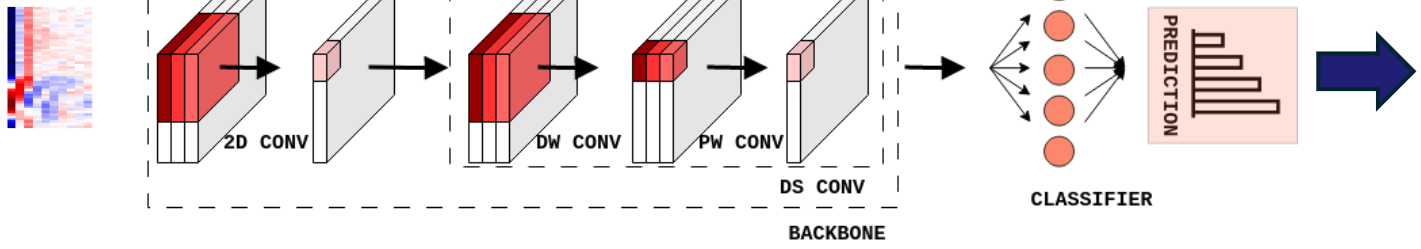
- Backward pass via backpropagation (**train** model parameters

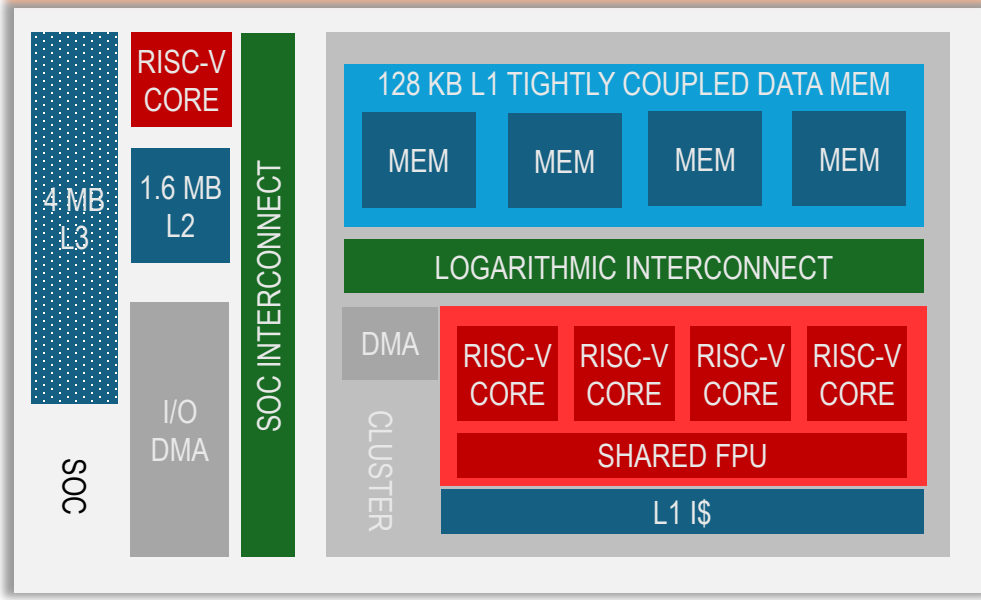
$$y_n = f_n(W_n \cdot y_{n-1})$$

$$L_{MSE} = \frac{1}{S}(y_n(input) - y_{gt})^2$$

$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n} = W_n - \eta \cdot k(y_n - y_{gt})y_{n-1}$$

```
pulp_backbone_fp32_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
pulp_MSELoss(&loss_args);
pulp_linear_fp32_bw_cl(&l1_args);
pulp_gradient_descent_fp32(&l1_args);
```





(Off-Chip) L3 FLASH/RAM [64 MB]

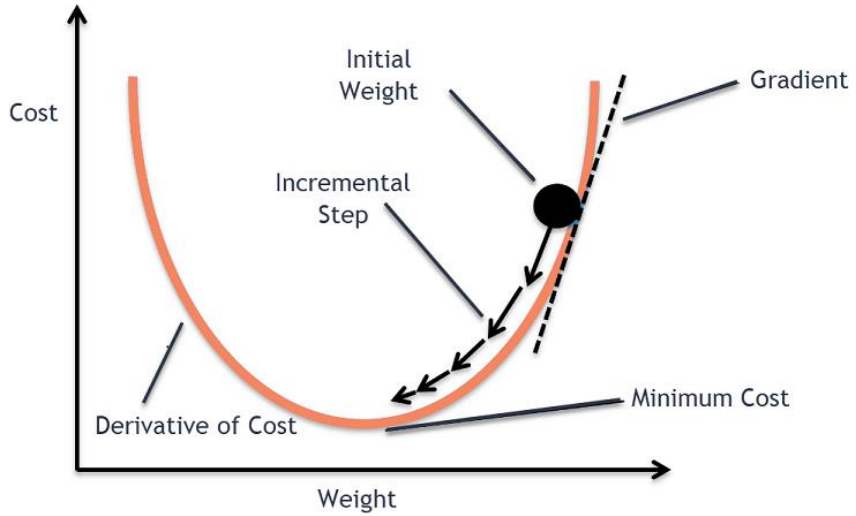
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

PULP Parallel Ultra Low Power

Revisiting Deep Learning – Batched update

- Update formula is input-dependent
  - $W_n' = W_n - \eta \cdot k\left(y_n(input) - y_{gt}\right)y_{n-1}(input)$

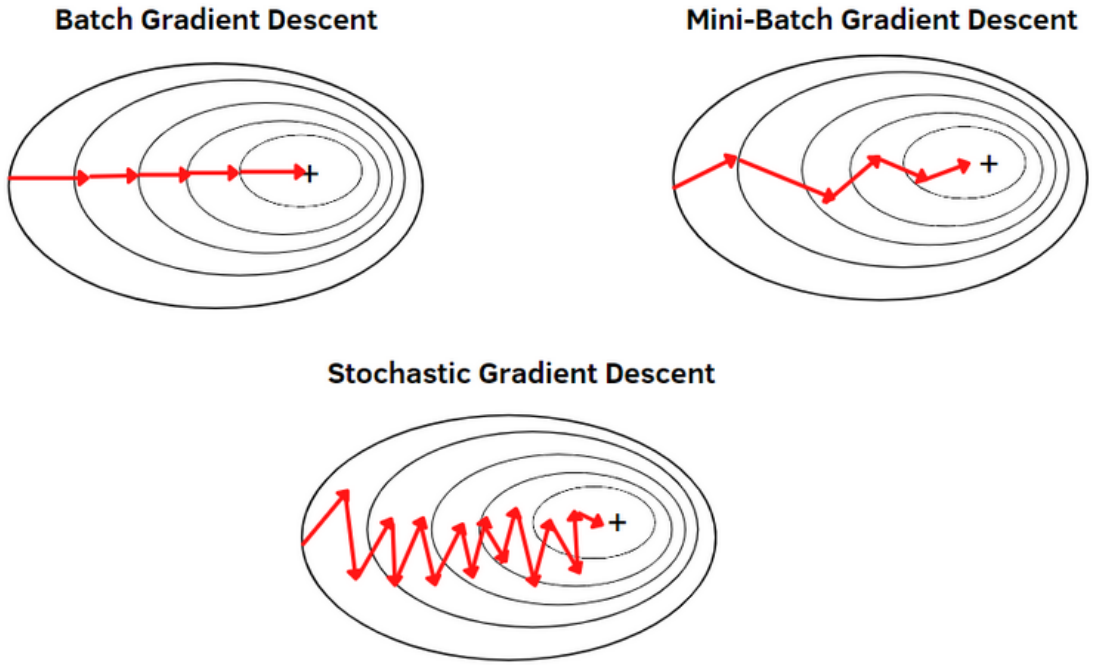
Rekha M, 2019



Revisiting Deep Learning – Batched update

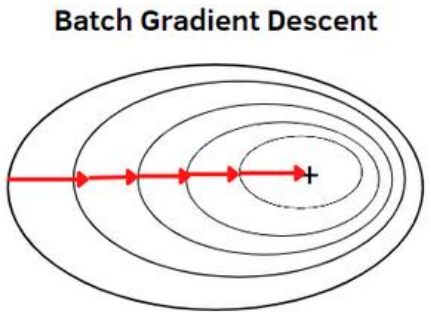
- Update formula is input-dependent
  - $W_n' = W_n - \eta \cdot k\left(y_n(input) - y_{gt}\right)y_{n-1}$

Kurtis Pykes, 2020



Batched Gradient Descent

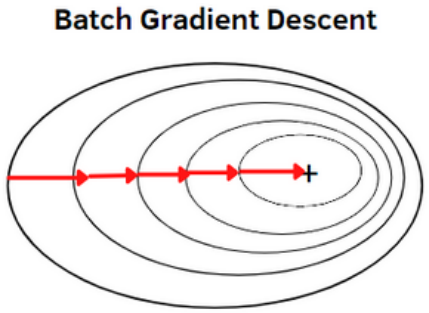
- Update formula is input-dependent
  - $W_n' = W_n - \eta \cdot k\left(y_n(input) - y_{gt}\right) y_{n-1}(input)$

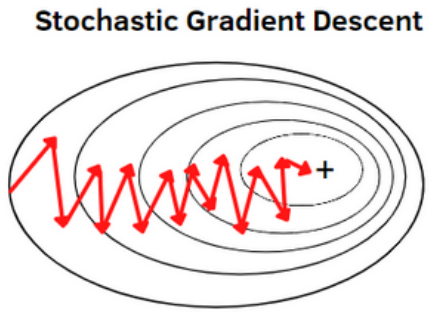


- See all the data simultaneously
- Excellent for smooth manifolds

Stochastic Gradient Descent

- Update formula is input-dependent
  - $W_n' = W_n - \eta \cdot k\left(y_n(input) - y_{gt}\right)y_{n-1}(input)$

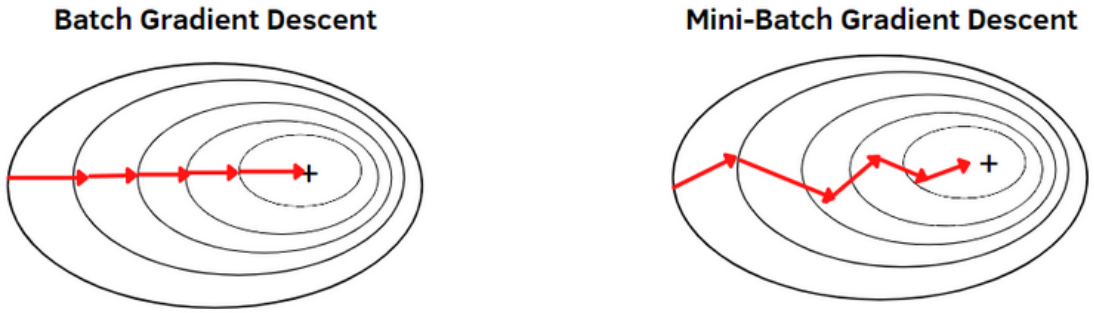


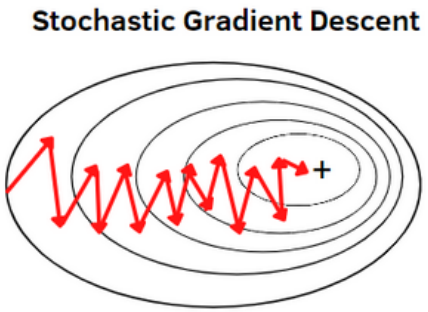


- See one data at a time
- Minimal memory cost

Mini-Batch Gradient Descent

- Update formula is input-dependent
  - $W_n' = W_n - \eta \cdot k\left(y_n(input) - y_{gt}\right)y_{n-1}(input)$





- See **some** data at a time
- Convergence rate
- Computational and memory efficient

Backpropagation has costs

```
// forward pass
for i in 1, L
```

$$y_i = f_i(W_i \cdot y_{i-1} + b_i)$$

```
compute loss
// backward pass
for i in L, 1
```

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial y_i} \cdot y_{i-1}$$

$$\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial y_{i+1}} \cdot W_i$$

```
update weights
```

Backpropagation has costs

```
// forward pass
for i in 1, L
```

$$y_i = f_i(W_i \cdot y_{i-1} + b_i)$$

```
compute loss
// backward pass
for i in L, 1
```

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial y_i} \cdot y_{i-1}$$

$$\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial y_{i+1}} \cdot W_i$$

```
update weights
```

- Storage
  - Frozen weights
  - Frozen biases

Backpropagation has costs

```
// forward pass
for i in 1, L
```

$$y_i = f_i(W_i \cdot y_{i-1} + b_i)$$

```
compute loss
// backward pass
for i in L, 1
```

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial y_i} \cdot y_{i-1}$$

$$\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial y_{i+1}} \cdot W_i$$

```
update weights
```

- Storage
  - Frozen weights
  - Frozen biases
- Memory (down to layer l)
  - $\sum_{i=l}^{L} W_i$ (weights)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial y_{i+1}}$ (input grads)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial W_i}$ (weight grads)
  - $\sum_{i=l-1}^{L} W_i$(activations)

Backpropagation has costs

```
// forward pass
for i in 1, L
```

$$y_i = f_i(W_i \cdot y_{i-1} + b_i)$$

```
compute loss
// backward pass
for i in L, 1
```

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial y_i} \cdot y_{i-1}$$

$$\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial y_{i+1}} \cdot W_i$$

```
update weights
```

$$Memory_{total} = Memory_{sample=1} x \frac{\#\ samples}{\#\ batches}$$

- Memory (down to layer l)
  - $\sum_{i=l}^{L} W_i$ (weights)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial y_{i+1}}$ (input grads)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial W_i}$ (weight grads)
  - $\sum_{i=l-1}^{L} W_i$(activations)

Backpropagation has costs

```
// forward pass
for i in 1, L
```

$$y_i = f_i(W_i \cdot y_{i-1} + b_i)$$

```
compute loss
// backward pass
for i in L, 1
```

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial y_i} \cdot y_{i-1}$$

$$\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial y_{i+1}} \cdot W_i$$

```
update weights
```

- Storage
  - Frozen weights
  - Frozen biases
- Memory (down to layer l)
  - $\sum_{i=l}^{L} W_i$ (weights)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial y_{i+1}}$ (input grads)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial W_i}$ (weight grads)
  - $\sum_{i=l-1}^{L} W_i$(activations)
- Operations (latency)
  - forward pass
  - gradients
  - update ↓↓↓

Backpropagation has costs

```
// forward pass
for i in 1, L
```

$$y_i = f_i(W_i \cdot y_{i-1} + b_i)$$

```
compute loss
// backward pass
for i in L, 1
```

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial y_i} \cdot y_{i-1}$$

$$\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial y_{i+1}} \cdot W_i$$

```
update weights
```

- Storage
  - Frozen weights
  - Frozen biases
- Memory (down to layer l)
  - $\sum_{i=l}^{L} W_i$ (weights)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial y_{i+1}}$ (input grads)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial W_i}$ (weight grads)
  - $\sum_{i=l-1}^{L} W_i$(activations)
- Operations (latency)
  - forw
  - grad
  - upd

forward pass + input gradients + weight gradients ≅ 3 forward passes

Backpropagation has costs

```
// forward pass
for i in 1, L
    y_i = f_i(W_i · y_{i-1} + b_i)
compute loss
// backward pass
for i in L, 1
    ∂L/∂W_i = ∂L/∂y_i · y_{i-1}
    ∂L/∂y_i = ∂L/∂y_{i+1} · W_i
update weights
```

- Storage
  - Frozen weights
  - Frozen bias
- Memory (dow… layer l)
  - $\sum_{i=l}^{L} W_i$ (weights)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial y_{i+1}}$ (input grads)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial W_i}$ (weight grads)
  - $\sum_{i=l-1}^{L} W_i$(activations)
- Operations (latency)

forward pass +
input gradients +
weight gradients
≅ 3 forward passes x #data x #epochs

Backpropagation has costs

```
// forward pass
for i in 1, L
```

$$y_i = f_i(W_i \cdot y_{i-1} + b_i)$$

```
compute loss
// backward pass
for i in L, 1
```

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial y_i} \cdot y_{i-1}$$

$$\frac{\partial L}{\partial y_i} = \frac{\partial L}{\partial y_{i+1}} \cdot W_i$$

```
update weights
```

- Storage
  - Frozen weights
  - Frozen biases
- Memory (down to layer l)
  - $\sum_{i=l}^{L} W_i$ (weights)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial y_{i+1}}$ (input grads)
  - $\sum_{i=l}^{L} \frac{\partial L}{\partial W_i}$ (weight grads)
  - $\sum_{i=l-1}^{L} W_i$(activations)
- Operations (latency)
  - forward pass
  - gradients
  - update ↓↓↓
- Energy
  - $E = = P \cdot t = = P \cdot Eff \cdot f \cdot OPs$

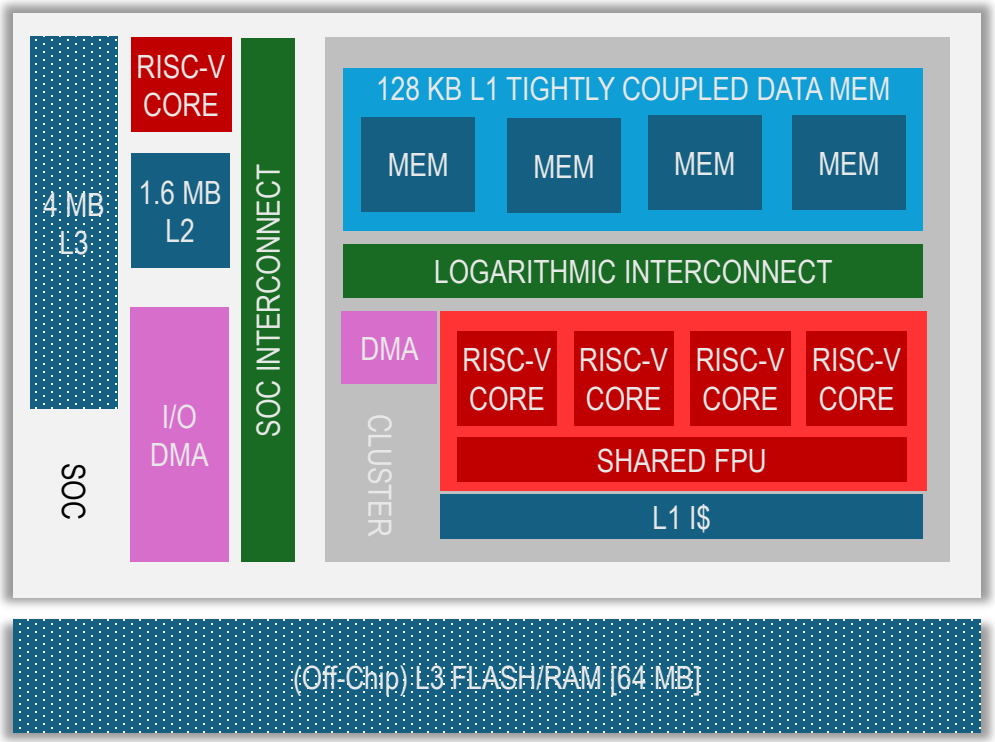
Backpropagation has costs

Storage
- Frozen weights
- Frozen biases

Memory (down to layer l)
- $\sum_{i=l}^{L} W_i$ (weights)
- $\sum_{i=l}^{L} \frac{\partial L}{\partial y_{i+1}}$ (input grads)
- $\sum_{i=l}^{L} \frac{\partial L}{\partial W_i}$ (weight grads)
- $\sum_{i=l-1}^{L} W_i$(activations)

Operations (latency)
- forward pass
- gradients
- update ↓↓↓

Energy
- $E = = P \cdot t = = P \cdot Eff \cdot f \cdot OPs$



Accuracy

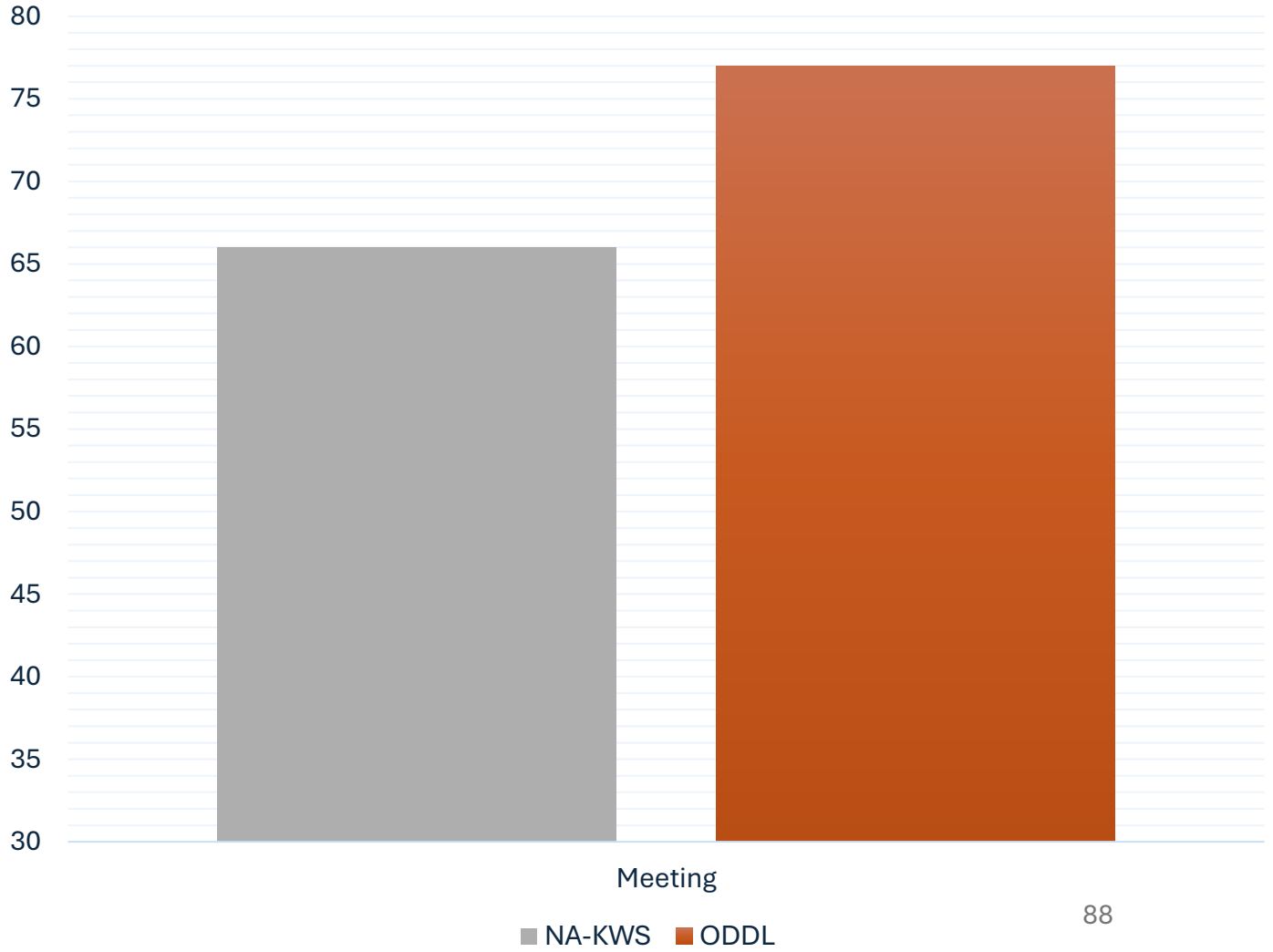
How do learning costs impact an On-Device Learning application?

Results

Improving the KWS accuracy in noisy environments

- **Accuracy increments** by 4% on average over **NA-KWS**
- 13% on **speech noise**

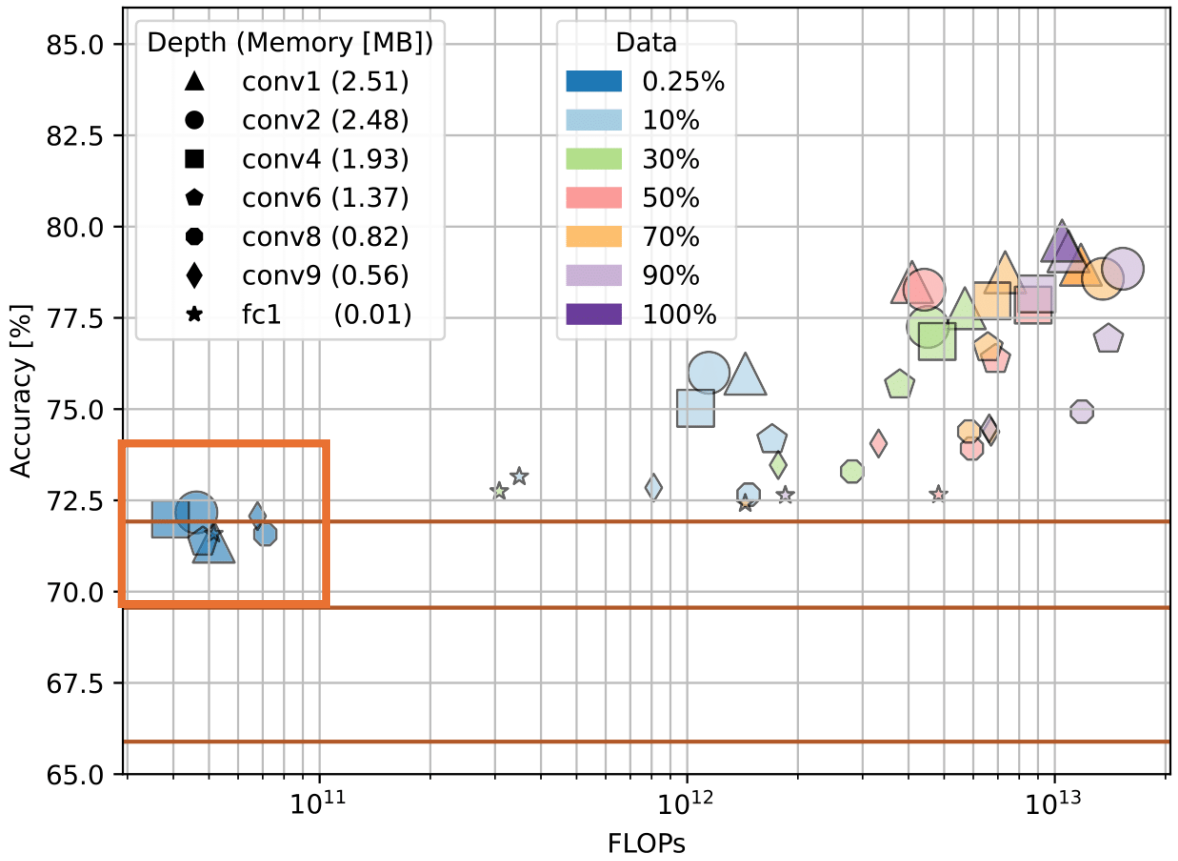
- Does it work well under embedded constraints?



Results

Memory requirements for effective learning

- **Refine fc_1 layer**
  - 10 kB on-chip L1 memory
  - $S_{ODDA} = S_{NA\text{-}KWS}$ **+ 5.5%**

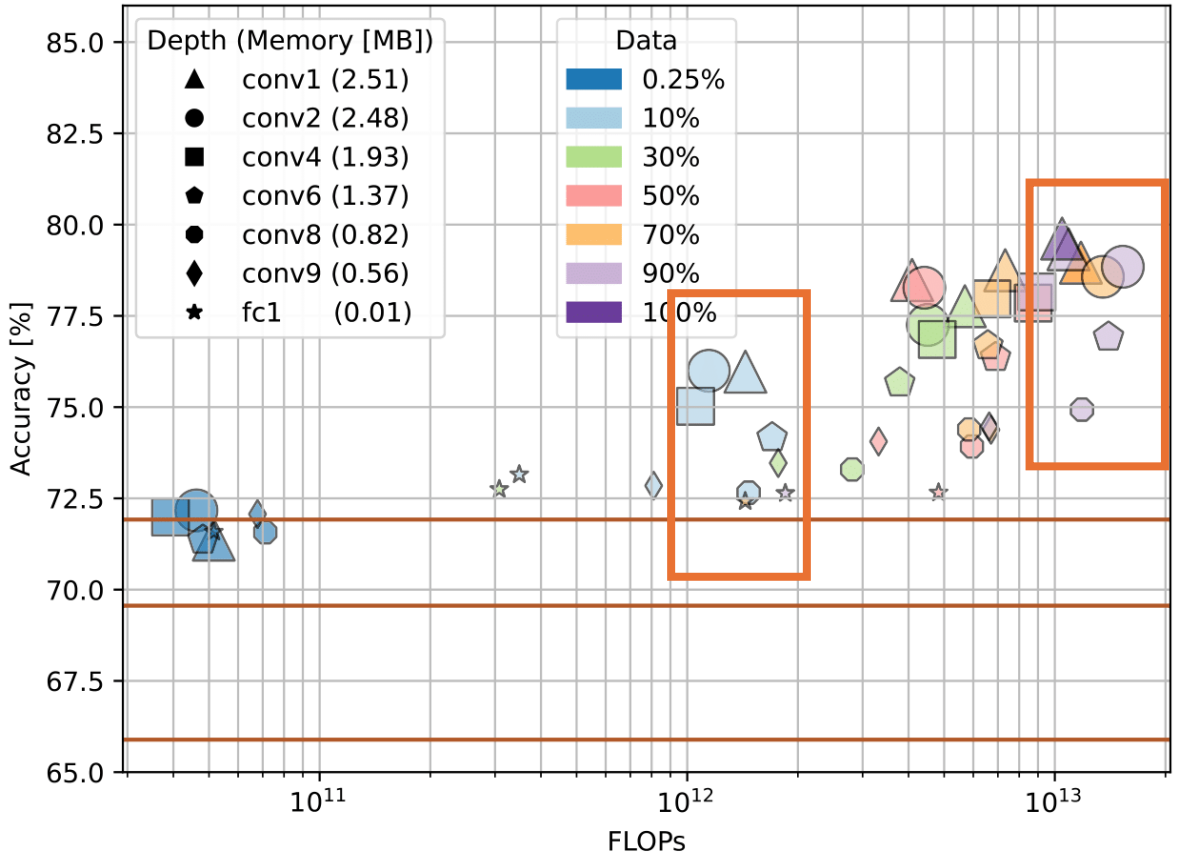


Results

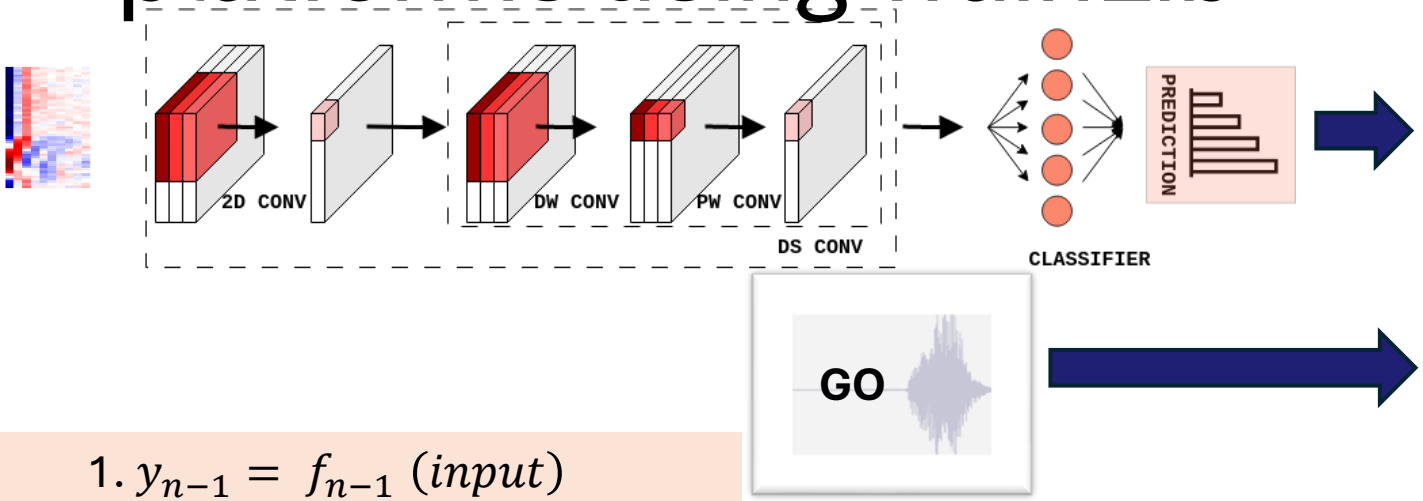
Memory requirements for effective learning

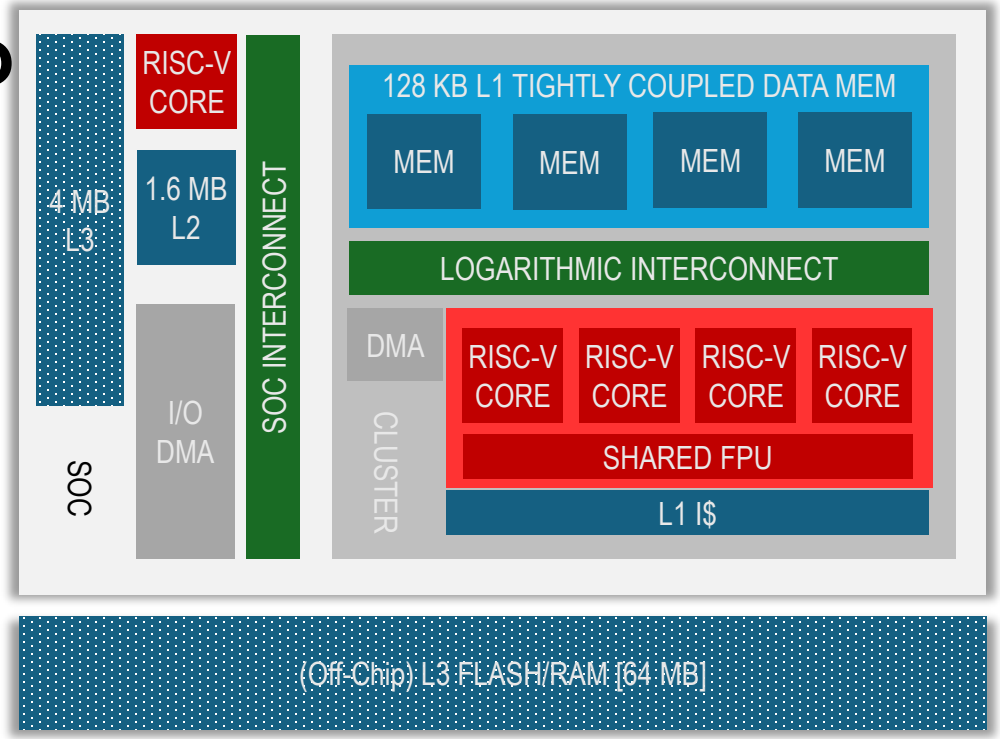
- **Refine fc_1 layer**
  - 10 kB on-chip L1 memory
  - $S_{ODDA} = S_{NA\text{-}KWS}$ **+ 5.5%**

- Refine backbone and classif.
  - +1.2% over *fc1* update using 10% of pre-recorded samples
  - +6% over *fc1* update using 100% of pre-recorded samples



On-Device Learning on PULP platforms using TrainLib





1. $y_{n-1} = f_{n-1}\ (input)$
2. $y_n = f_n\ (W_n \cdot y_{n-1} + b_n)$
3. $L = -\sum_{cls} y_{gt_{cls}} \log(y_{n_{cls}})$
4. $\frac{\partial L}{\partial W_n} = \frac{\partial L}{\partial y_n}\frac{\partial z_n}{\partial W_n}$
5. $W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n}$

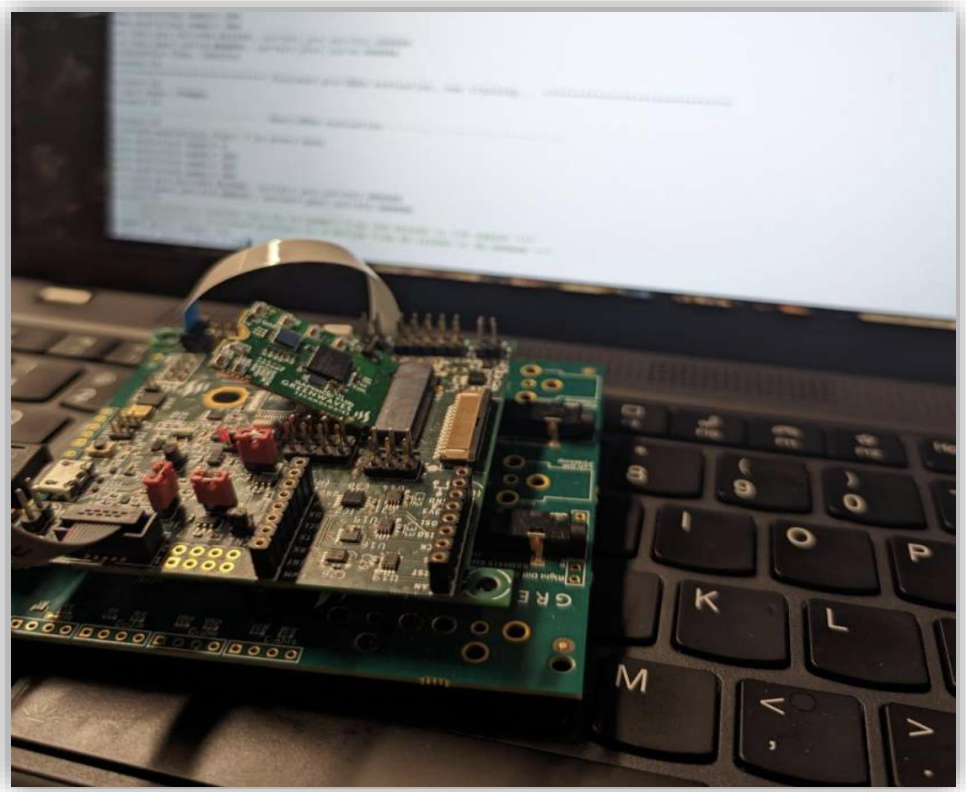
```
pulp_backbone_fp32_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
pulp_CrossEntropyLoss(&loss_args);
pulp_linear_fp32_bw_cl(&l1_args);
pulp_gradient_descent_fp32(&l1_args);
```

Memory management

- Pre-recorded (clean) utterances and labels stored in L3 memory
- Pretrained model stored in L3 memory
- Weights, activations stored in L2 memory during forward pass
- Weights, activations, gradients stored in L1 during backward pass

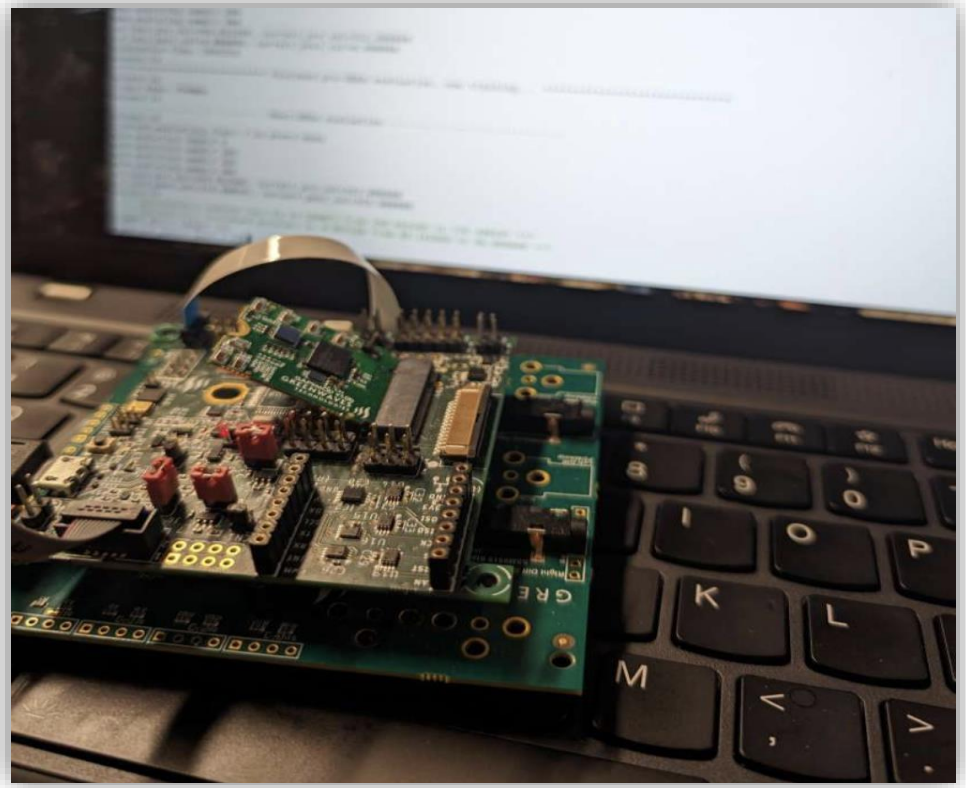
Results
Implementation on GAP 9

- Greenwaves GAP9 – based on PULP Vega [Rossi2022]
- Low-power mode: 240 MHz, 650 mV



Results Implementation on GAP 9

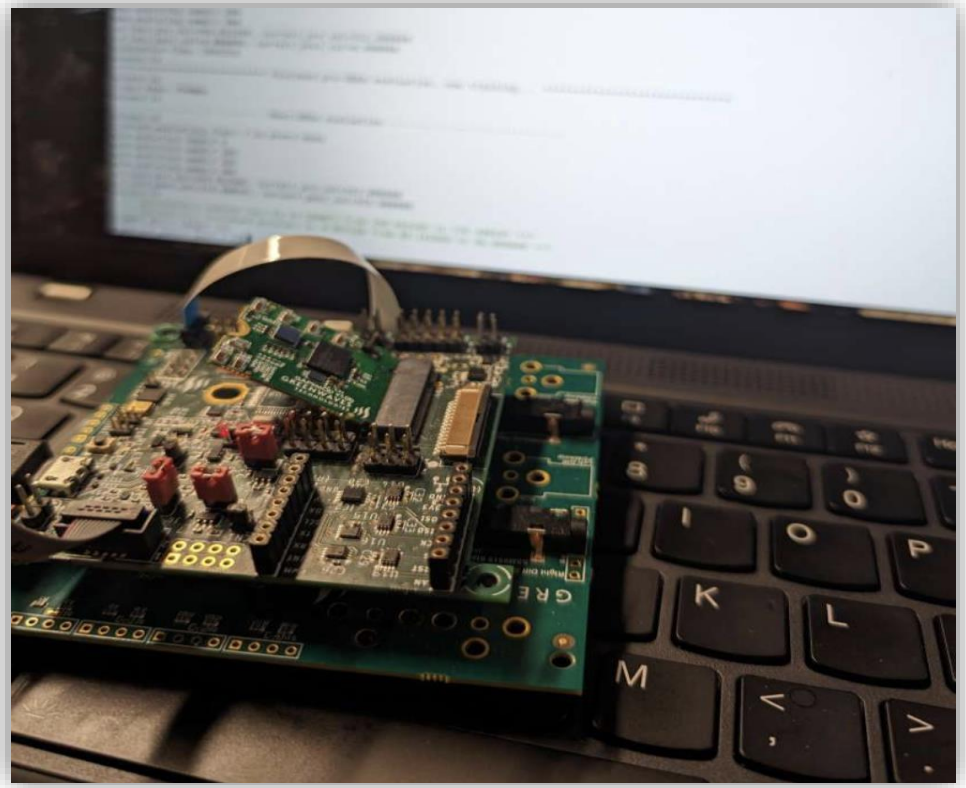
- Greenwaves GAP9 – based on PULP Vega [Rossi2022]
- Low-power mode: 240 MHz, 650 mV
  - On-device learning in **½ mJ**, ready in **11 ms**



DS-CNN Model	Compute [MFLOps]	Storage [kB]	Memory [kB]	Eff. [FLOPs/ cycle]	Compute time [ms]	Energy [µJ]
S	2.95	23.7	9.5	4.94	**10.89**	**424**
M	17.2	138.1	25.5	9.18	24.16	988
L	51.1	416.7	40.9	11	55.04	2313

Results Implementation on GAP 9

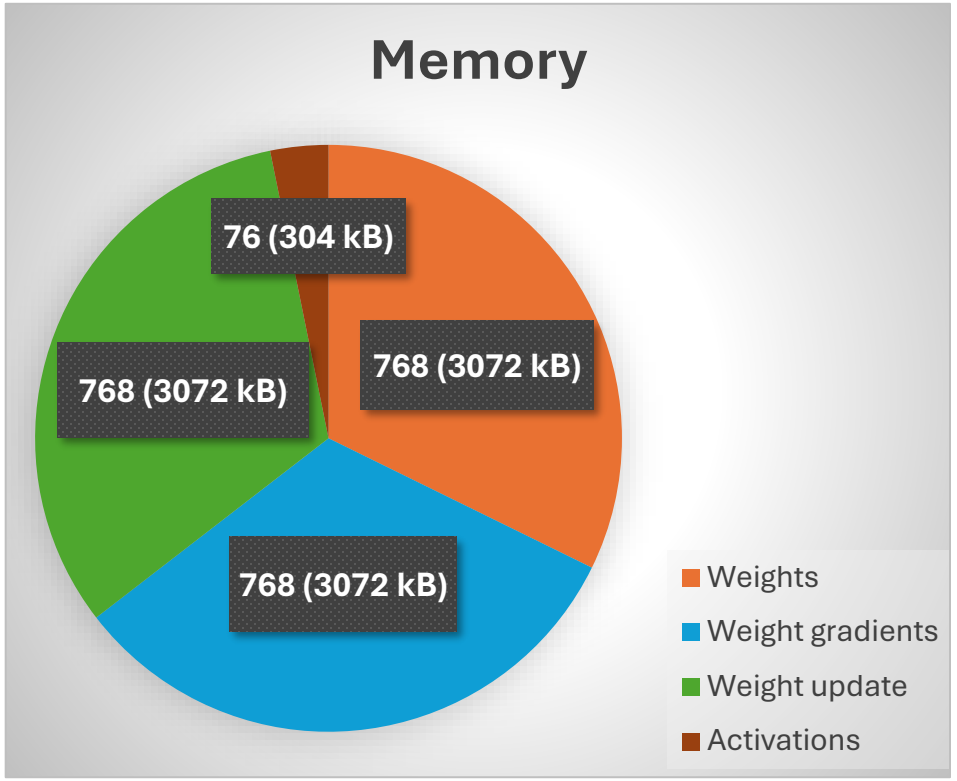
- Greenwaves GAP9 – based on PULP Vega [Rossi2022]
- Low-power mode: 240 MHz, 650 mV
  - On-device learning in **½ mJ**, ready in **11 ms**
  - **10 kB** of L1 memory for backpropagation



DS-CNN Model	Compute [MFLOps]	Storage [kB]	Memory [kB]	Eff. [FLOPs/ cycle]	Compute time [ms]	Energy [µJ]
S	2.95	23.7	**9.5**	4.94	10.89	424
M	17.2	138.1	25.5	9.18	24.16	988
L	51.1	416.7	40.9	11	55.04	2313

Results Implementation on GAP 9

- Greenwaves GAP9 – based on PULP Vega [Rossi2022]
- Low-power mode: 240 MHz, 650 mV
  - On-device learning in **½ mJ**, ready in **11 ms**
  - **10 kB** of L1 memory for backpropagation

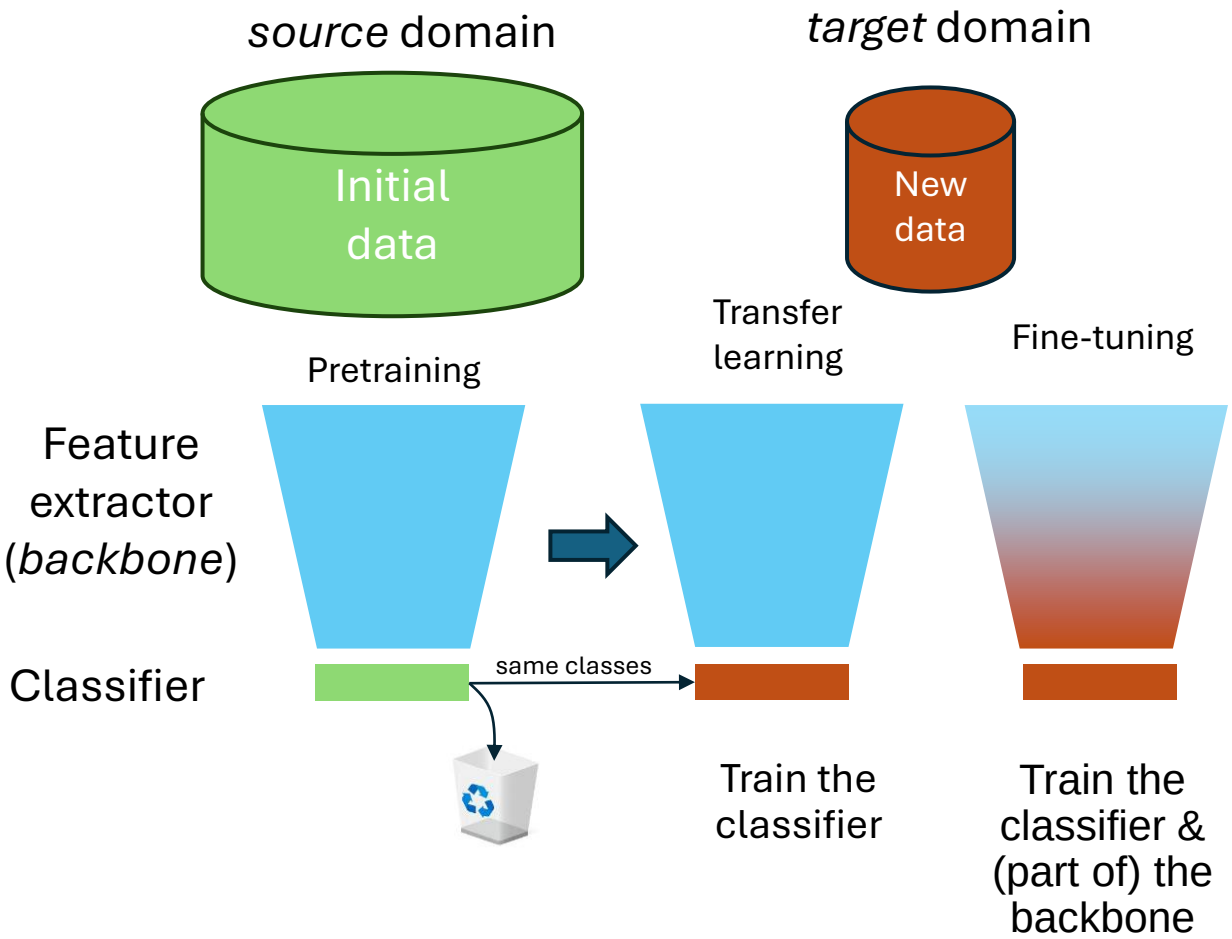


DS-CNN Model	Compute [MFLOps]	Storage [kB]	Memory [kB]	Eff. [FLOPs/ cycle]	Compute time [ms]	Energy [µJ]
S	2.95	23.7	**9.5**	4.94	10.89	424
M	17.2	138.1	25.5	9.18	24.16	988
L	51.1	416.7	40.9	11	55.04	2313

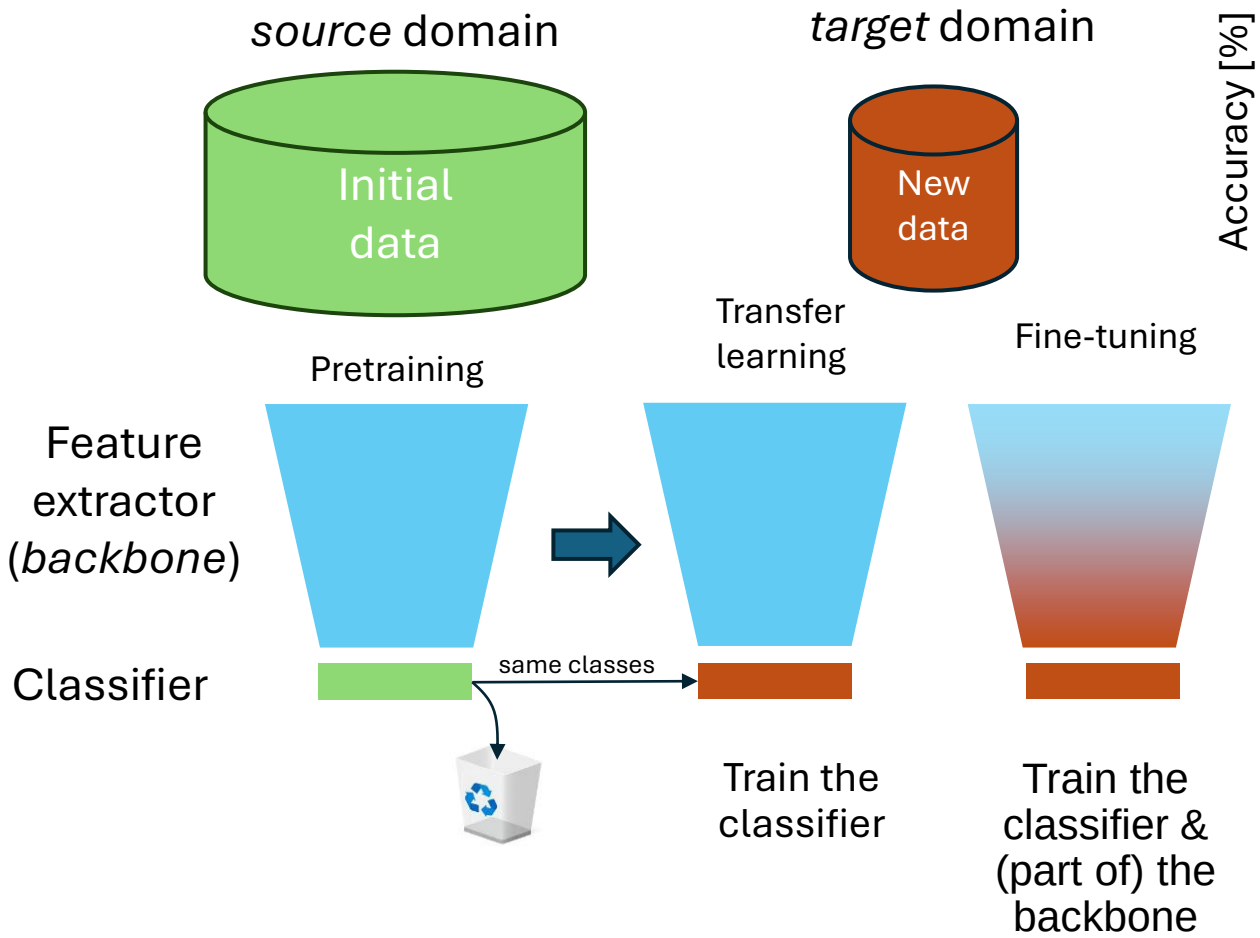
And now for something completely different

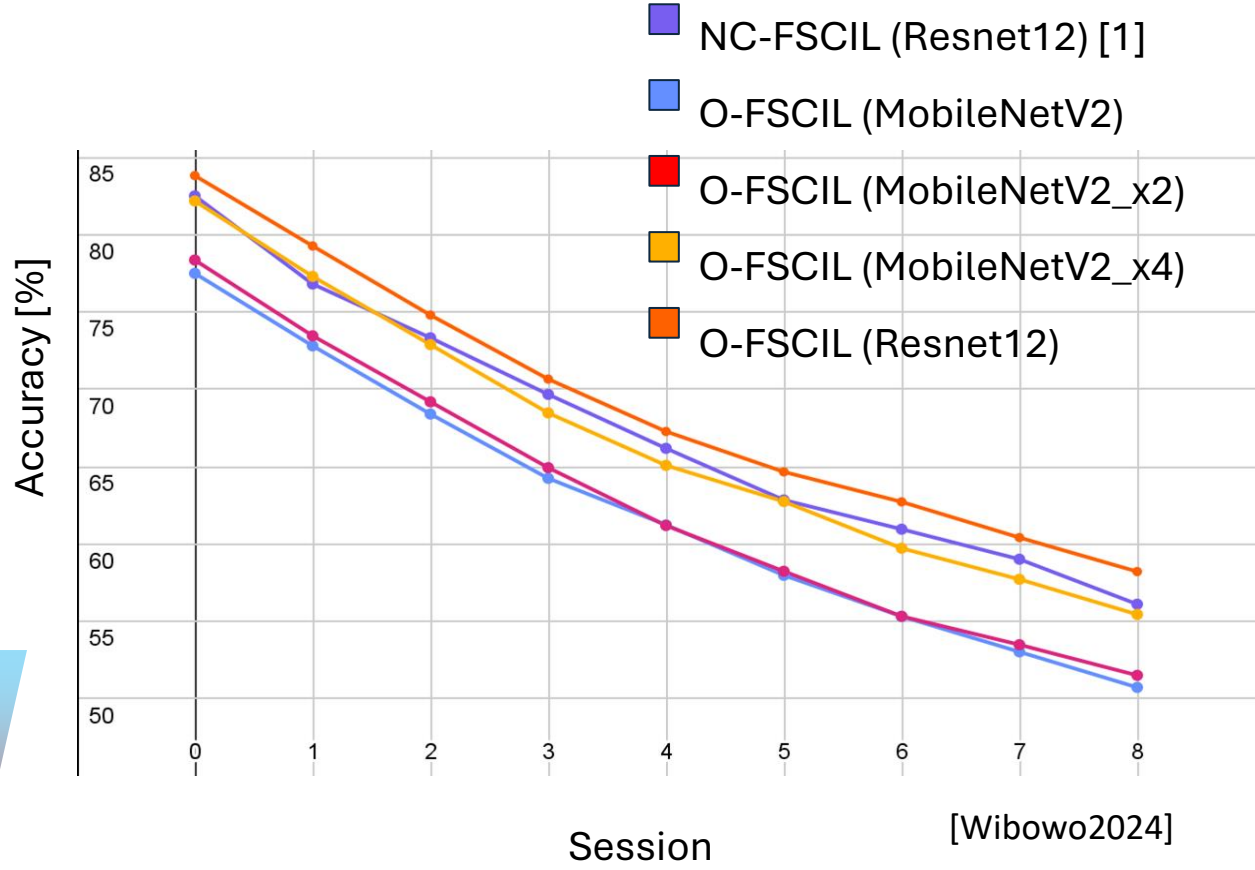
Credit: https://www.swspotlight.com/articles/health-and-fitness/stay-fit-students-flex-their-mental-muscles-in-brain-fitness-classes/

Domain adaptation
Transfer Learning & Fine-tuning



Catastrophic forgetting





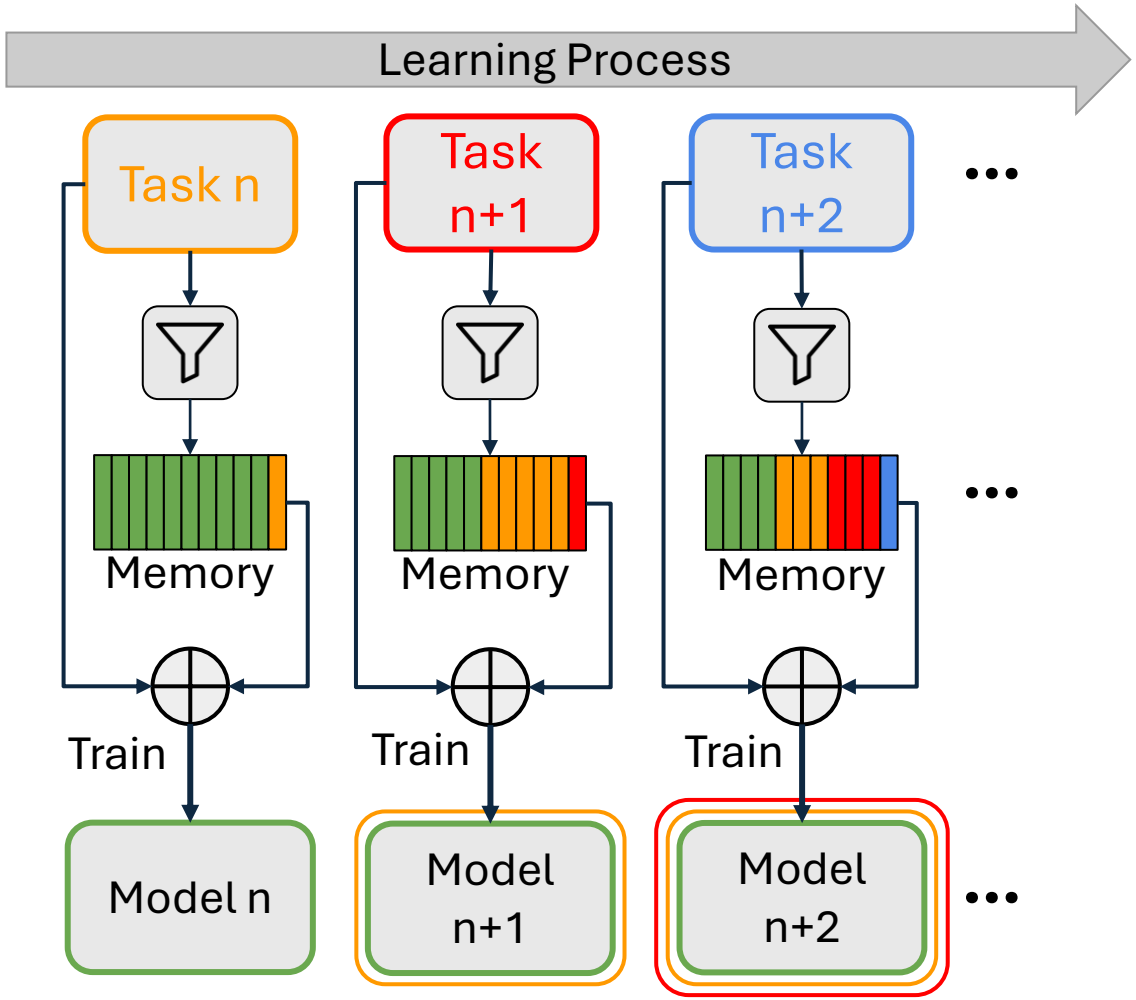
[Wibowo2024]

- Continual Learning – strategies to mitigate catastrophic forgetting

On-Device Continual learning

Algorithmic strategies

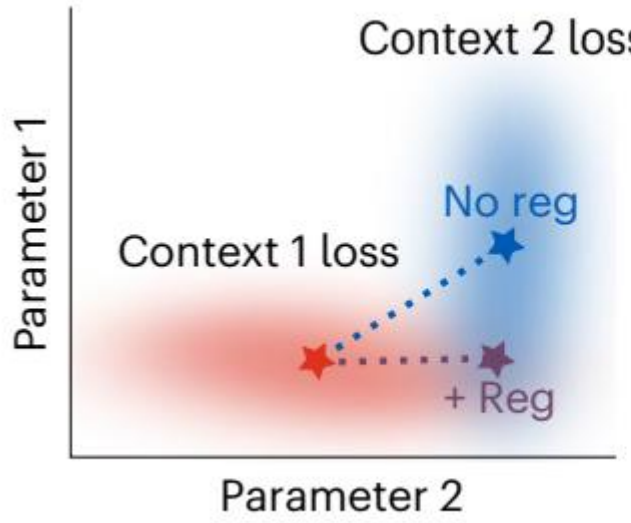
- Replay-based
  - Experience replay [Rolnick2019]
  - Extended Class Balancing Reservoir Sampling [Amalapuram2024]



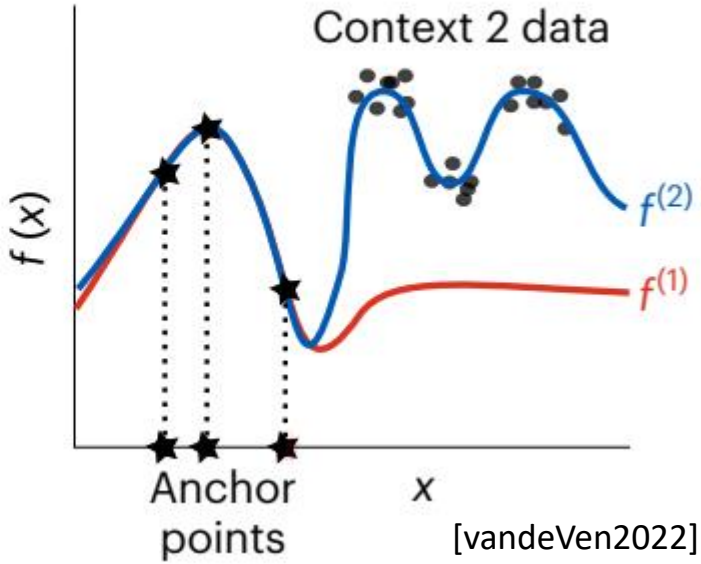
On-Device Continual learning
Algorithmic strategies

- Replay-based
  - Experience replay [Rolnick2019]
  - Extended Class Balancing Reservoir Sampling [Amalapuram2024]
- Regularization-based
  - Elastic Weight Consolidation [Kirkpatrick2017]
  - Learning without Forgetting [Li2016]

Parameter regularization



Functional regularization



[vandeVen2022]

On-Device Continual learning
System-level strategies

- Selective layer update
  - Retrain only the classifier [Ren2021]
  - Retrain only the biases [Cai2020]
  - Sparse Update [Lin2022]
- Save checkpoints
  - Select checkpoints through exhaustive search [Khan2023]
  - Select checkpoints through Mixed Integer Linear Programming and use paging to copy activations to off-chip memory [Patil2022]
- Reduce precision
  - During training [Lin2022]
  - In the replay buffer [Ravaglia2021]

Conclusions

- Backpropagation at the extreme edge is expensive
  - Storage, memory, operations, latency
  - Embrace accuracy-complexity trade-offs

- Hardware-Software co-design is required for efficient CL at the extreme edge
  - Few-shot supervised, semi-supervised, and unsupervised learning
  - Heterogeneous computing (e.g., cluster & accelerator → at-MRAM acc.)

References

[Niculescu2024] V. Niculescu, T. Polonelli, M. Magno and L. Benini, “NanoSLAM: Enabling Fully Onboard SLAM for Tiny Robots”, 2024, IEEE Internet of Things Journal

[Vostrikov2023] S. Vostrikov, M. Anderegg, C. Leitner, L. Benini and A. Cossettini, “Hand Gesture Recognition via Wearable Ultra-Low Power Ultrasound and Gradient-Boosted Tree Classifiers”, IEEE International Ultrasonics Symposium (IUS), 2023

[Tatman2017] R. Tatman, C. Kasten, "Effects of Talker Dialect, Gender & Race on Accuracy of Bing Speech and YouTube Automatic Captions," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, 2017.

[Savoldi2022] B. Savoldi et al, "Under the Morphosyntactic Lens: A Multifaceted Evaluation of Gender Bias in Speech Translation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022.

[Cioflan2024] C. Cioflan, L. Cavigelli, M. Rusci, M. De Prado and L. Benini, "On-Device Domain Learning for Keyword Spotting on Low-Power Extreme Edge Embedded Systems," *IEEE 6th International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2024*

[Frey2022] S. Frey, S. Vostrikov, L. Benini and A. Cossettini, "WULPUS: a Wearable Ultra Low-Power Ultrasound probe for multi-day monitoring of carotid artery and muscle activity," *IEEE International Ultrasonics Symposium (IUS)*, Venice, Italy, 2022

[Frey2023] S. Frey, M. Guermandi, S. Benatti, V. Kartsch, A. Cossettini and L. Benini, "BioGAP: a 10-Core FP-capable Ultra-Low Power IoT Processor, with Medical-Grade AFE and BLE Connectivity for Wearable Biosignal Processing," *IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2023.

[Kalenberg2024] K. Kalenberg *et al*., "Stargate: Multimodal Sensor Fusion for Autonomous Navigation On Miniaturized UAVs," in *IEEE Internet of Things Journal (Early access)*

[Rossi2022] D. Rossi *et al*., "Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode," in *IEEE Journal of Solid-State Circuits*, 2022

[Wibowo2024] Y.E. Wibowo et al., “12 mJ per Class On-Device Online Few-Shot Class-Incremental Learning”, IEEE Design, Automation and Test in Europe (DATE). 2024

References

[Rolnick2019] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, G. Wayne, “Experience Replay for Continual Learning”, Advances in Neural Information Processing Systems 32 (NeurIPS), 2019.

[Amalapuram2024] S.K. Amalapuram, S. Channappayya, B. Reddy Tamma, “Augmented Memory Replay-based Continual Learning Approaches for Network Intrusion Detection”, Advances in Neural Information Processing Systems 36 (NeurIPS), 2023.

[Li2016] Z. Li, D. Hoiem, “Learning Without Forgetting”, 14th European Conference on Computer Vision (ECCV), 2016.

[Kirkpatrick2017] J. Kirkpatrick et al., “Overcoming catastrophic forgetting in neural networks”, 2016, Proceedings of the National Academy of Sciences

[vandeVen2022] van de Ven, G.M., Tuytelaars, T. & Tolias, A.S. “Three types of incremental learning”. Nature Machine Intelligence 4, 2022.

[Ren2021] Ren, Haoyu, Darko Anicic, and Thomas A. Runkler. "TinyOL: TinyML with Online-Learning on Microcontrollers." International Joint Conference on Neural Networks (IJCNN), 2021.

[Cai2020] Cai, Han, Chuang Gan, Ligeng Zhu, and Song Han. “TinyTL: Reduce memory, not parameters for efficient on-device learning.“, Advances in Neural Information Processing Systems 33 (NeurIPS 2020) .

[Lin2022] Lin, Ji, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. "On-device training under 256kb memory.“, Advances in Neural Information Processing Systems 35 (NeurIPS 2022).

[Patil2022] Patil, Shishir G., Paras Jain, Prabal Dutta, Ion Stoica, and Joseph Gonzalez. “POET: Training neural networks on tiny devices with integrated rematerialization and paging." International Conference on Machine Learning (ICML), 2022.

[Khan2023] Khan, Osama, Gwanjong Park, and Euiseong Seo. "DaCapo: An On-Device Learning Scheme for Memory-Constrained Embedded Systems." Transactions on Embedded Computing Systems (TECS) 2023.

[Ravaglia2021] Ravaglia, Leonardo, Manuele Rusci, Davide Nadalini, Alessandro Capotondi, Francesco Conti, and Luca Benini. "A tinyml platform for on-device continual learning with quantized latent replays." IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), 2021.

PULP Platform

Open Source Hardware, the way it should be!

Cristian Cioflan cioflanc@iis.ee.ethz.ch

Institut für Integrierte Systeme – ETH Zürich
Gloriastrasse 35
Zürich, Switzerland
DEI – Università di Bologna
Viale del Risorgimento 2
Bologna, Italy

@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

And now for something completely different

Credit: https://www.swspotlight.com/articles/health-and-fitness/stay-fit-students-flex-their-mental-muscles-in-brain-fitness-classes/

PULP-TrainLib

github.com/pulp-platform/pulp-trainlib

Davide Nadalini, Università di Bologna

d.nadalini@unibo.it

Find the slides here and the tutorial here.