On-Device Domain Learning for Keyword Spotting on Low-Power Extreme Edge Embedded Systems

Cristian Cioflan[†], Lukas Cavigelli[‡], Manuele Rusci[‖], Miguel de Prado[¶], Luca Benini[†§]

[†]Integrated Systems Laboratory, ETH Zurich; [‡]Zurich Research Center, Huawei Technologies;
[‖]ESAT, KU Leuven; [¶]VERSES AI; [§]DEI, University of Bologna;

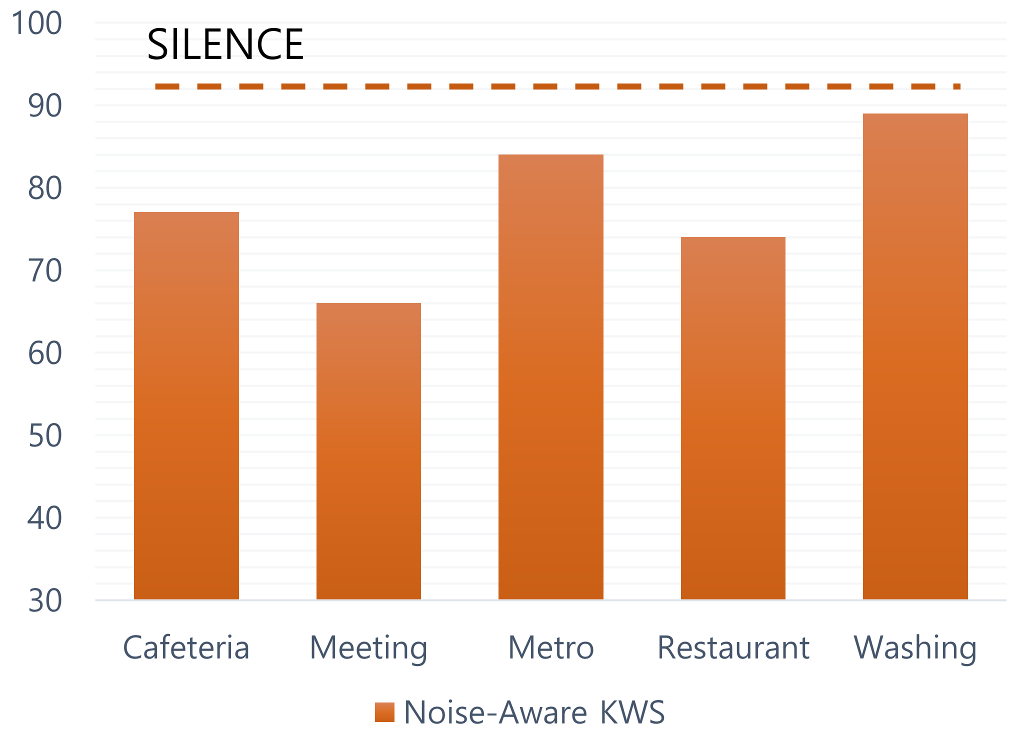
2024 IEEE International Conference on Artificial Intelligence Circuits and Systems

Keyword Spotting at the extreme edge

- Voice-controlled personal assistants
- Drones controlled remotely to investigate hard-to-reach locations
- Hearing devices adapted to the environment conditions

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**



Noise-Aware Keyword Spotter (NA-KWS)

- Trained for **generic** robustness
- Accuracy drop compared to a noiseless model trained in noiseless conditions

How to mitigate the performance degradation?

- **Server-side training** on on-site data [Lopez-Espejo2021, Ng2022]
  - × Does not respect **privacy**
  - × Communication will reduce **device lifetime**
  - × User-specific **labeled data** is scarce

How to mitigate the performance degradation?

- **Server-side training** on on-site data [Lopez-Espejo2021, Ng2022]
  - × Does not respect **privacy**
  - × Communication will reduce **device lifetime**
  - × User-specific **labeled data** is scarce

- **On-device training** (by backpropagation) must address
  - ✓Limited **storage** – tens of MB (e.g., data, model parameters)
  - ✓Limited **memory** – hundreds of kB (e.g., activations, gradients)
  - ✓Real-time operation – minimize **latency** (∝ #operations)
  - ✓Always-on devices – minimize **energy consumption**

On-Device Learning frameworks

Method	Target device	Proposed optimization	Retrainable layers	Data type
[Ren2021]	Arduino Nano 33	Retrain last (additional) layer	Linear	FP32
[Lin2022]	STM32	Quantized Sparse Update	Convolutions, Linear	INT8, FP32
[Nadalini2022]	Multicore RISC-V MCUs	Parallelism, SIMD, loop unrolling	Convolutions, Linear	FP32, FP16

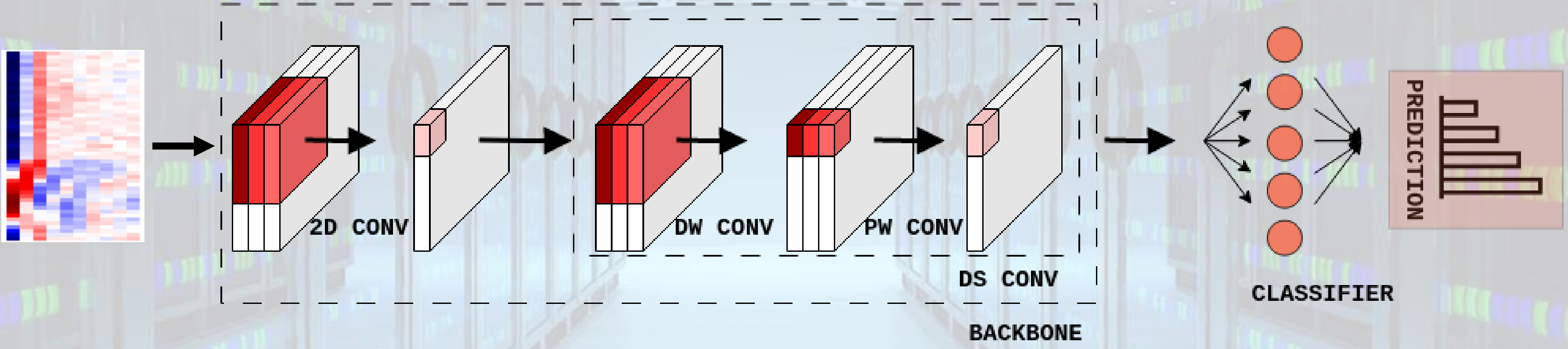
We exploit the framework proposed by [Nadalini2022]

- It addresses latency and energy consumption
- We additionally consider memory & storage constraints

to achieve **end-to-end on-device domain learning for keyword spotting**

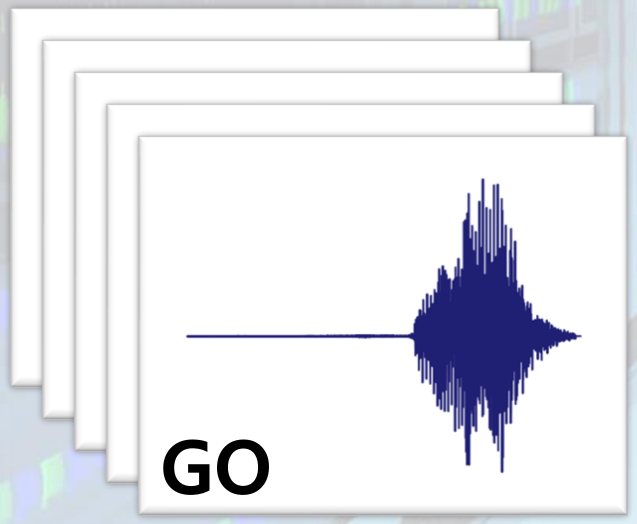
On-Device Domain Learning – the methodology

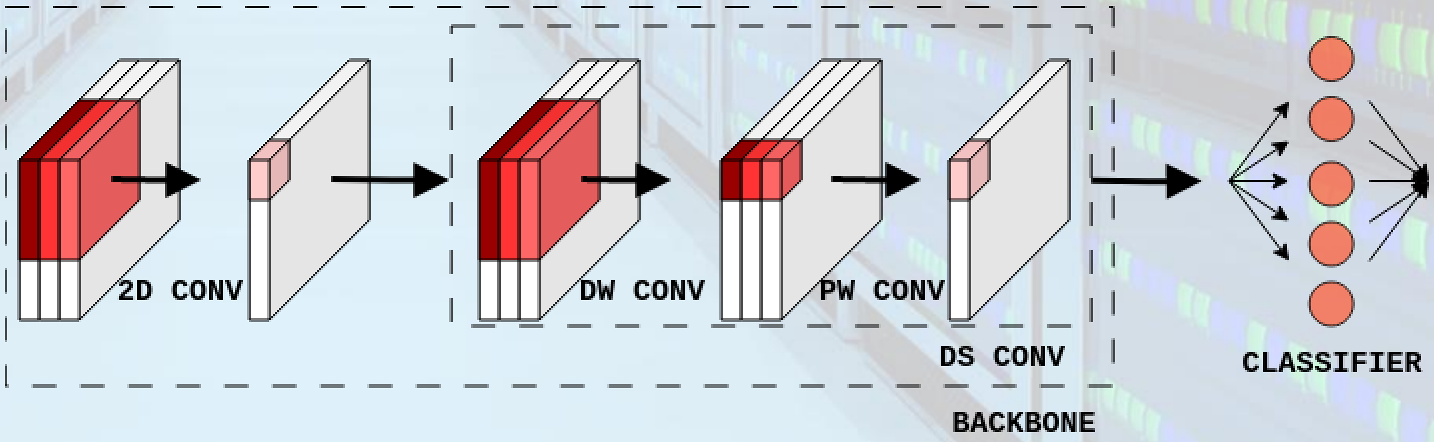
- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server [Cioflan2022]



On-Device Domain Learning – the methodology

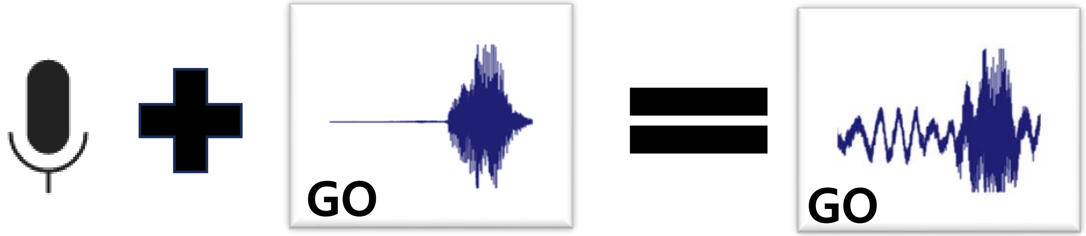
- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server [Cioflan2022]
  - Deploy KWS model
  - Store pre-recorded utterances and labels





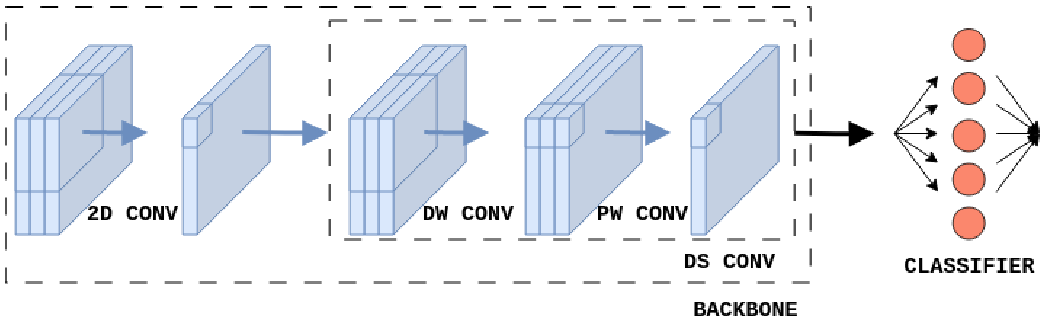
On-Device Domain Learning – the methodology

- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server [Cioflan2022]
  - Deploy KWS model
  - Store pre-recorded utterances and labels
- Adapt to new environments
  - Record noise from the environment
  - Augment pre-recorded utterances



On-Device Domain Learning – the methodology

- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server [Cioflan2022]
  - Deploy KWS model
  - Store pre-recorded utterances and labels
- Adapt to new environments
  - Record noise from the environment
  - Augment pre-recorded utterances
  - **On-device (supervised) learning**

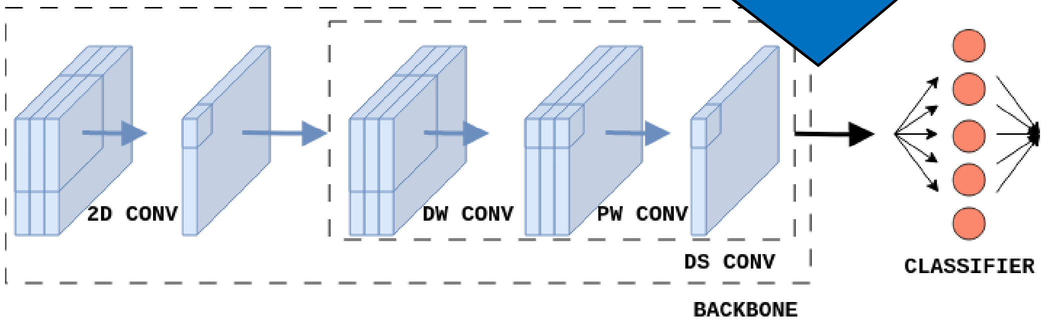


1. Forward pass – compute the activations
2. Backward pass
   1. Compute the loss considering the ground truth (pre-recorded)
   2. Compute the gradients through *backpropagation*
   3. Update the parameters

On-Device Domain Learning – the methodology

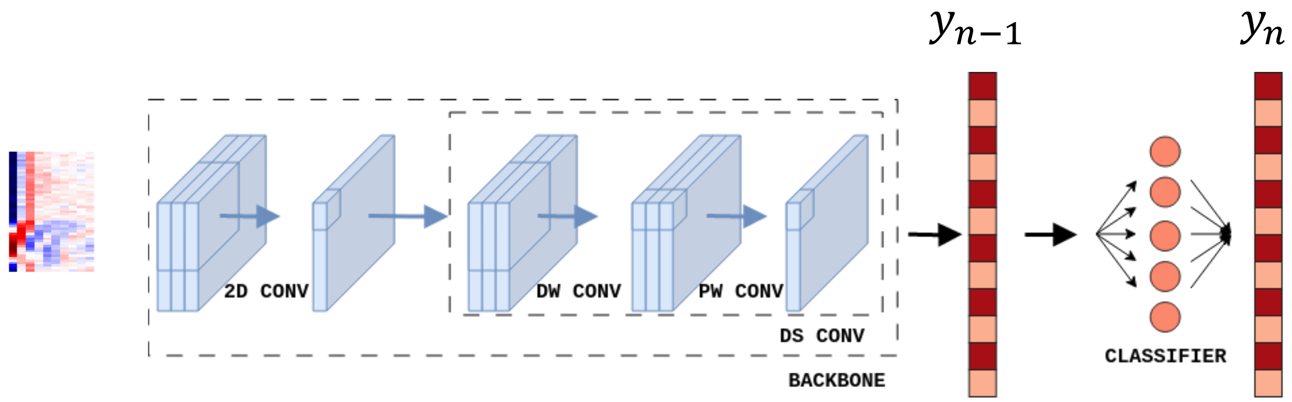
- Enable on-device keyword spotting
  - Train (and quantize) NA-KWS model – on the server
  - Deploy KWS model
  - Store pre-recorded utterances and labels
- Adapt to new environments
  - Record noise from the environment
  - Augment pre-recorded utterances
  - **On-device (supervised) learning**

e.g., freezing the backbone, updating the classifier



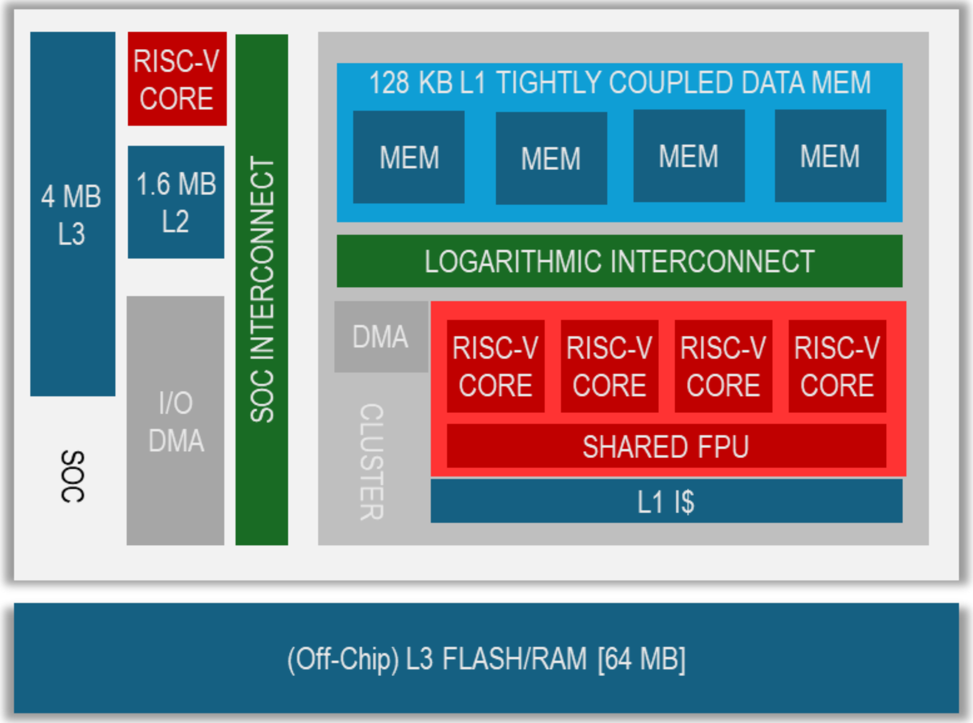
1. Forward pass – compute the activations
2. Backward pass
   1. Compute the loss considering the ground truth (pre-recorded)
   2. Compute the gradients through *backpropagation*
   3. Update the parameters

Forward pass – compute the activations



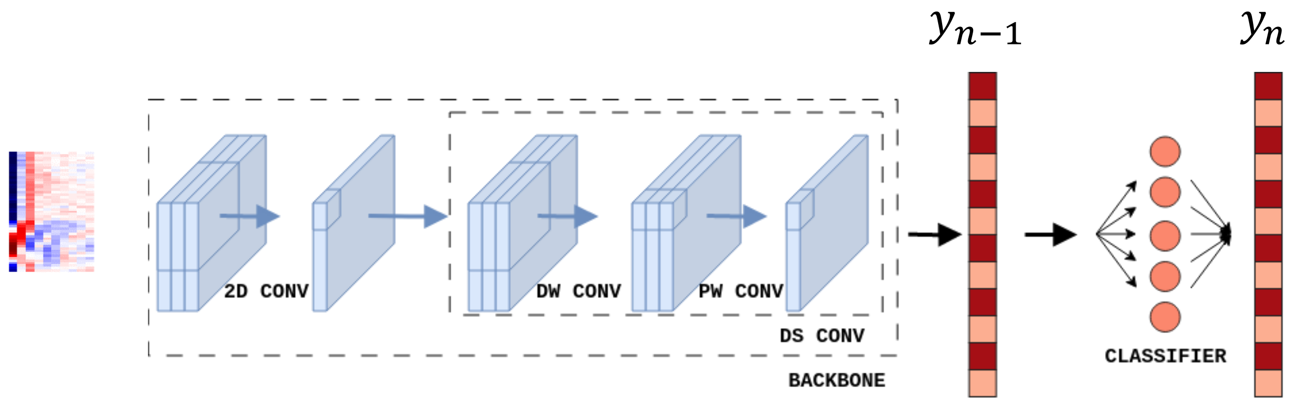
$$y_n = W_n \cdot y_{n-1}$$

```
pulp_backbone_int8_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
```



Backward pass – compute the loss

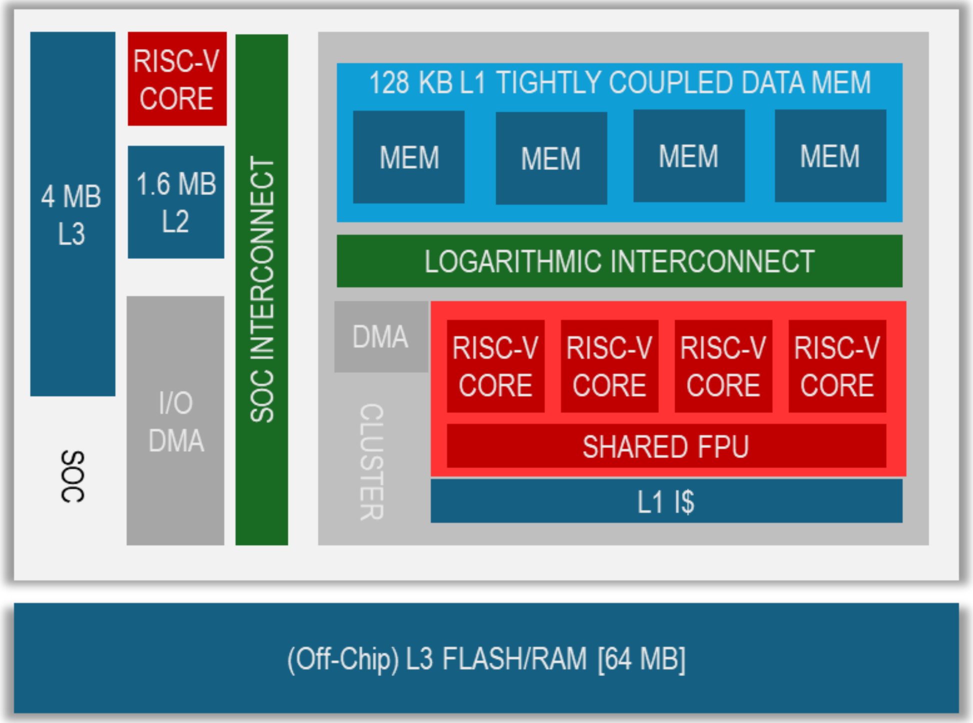
```
pulp_backbone_int8_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
pulp_CrossEntropyLoss(&loss_args);
```



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

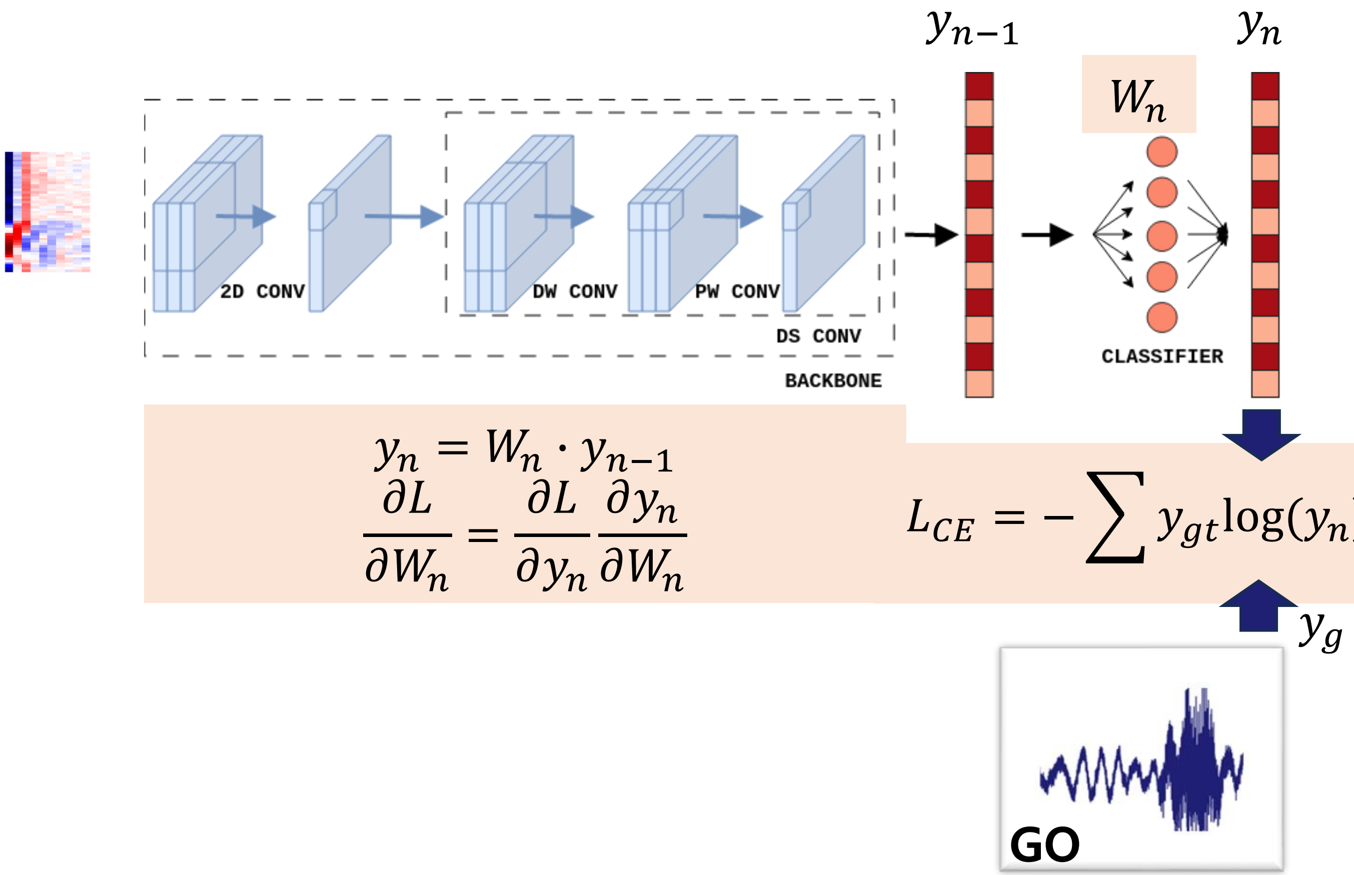
$$L_{CE} = -\sum y_{gt} \log(y_n)$$

y_g



Backward pass – compute the gradients (backpropagation)

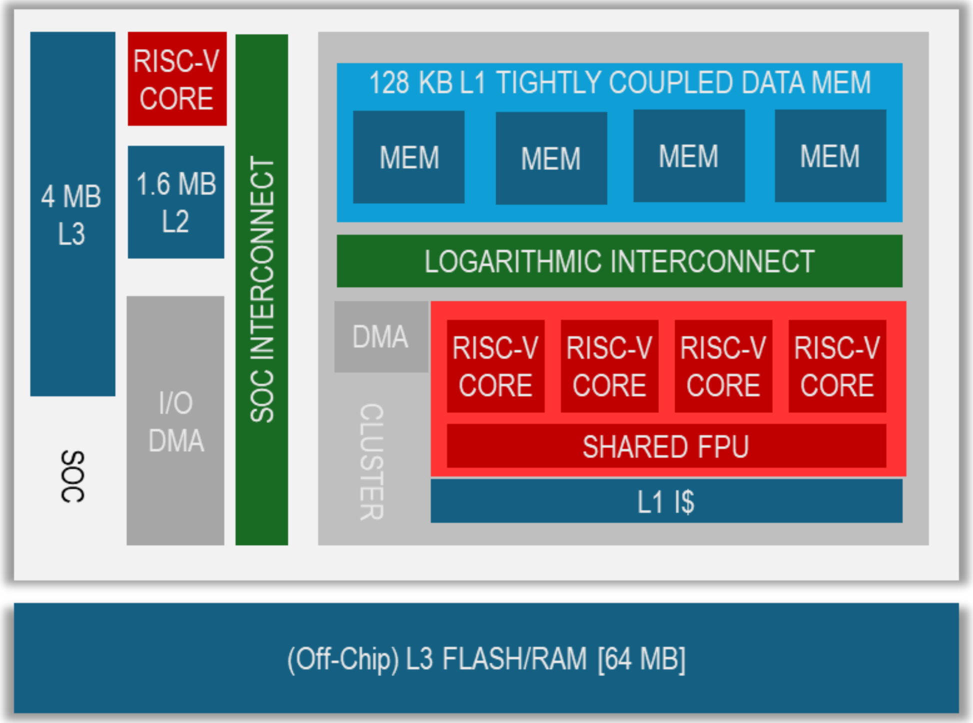
```
pulp_backbone_int8_fw_cl(&args);
pulp_linear_fp32_fw_cl(&args);
pulp_CrossEntropyLoss(&loss_args);
pulp_linear_fp32_bw_cl(&l1_args);
```



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

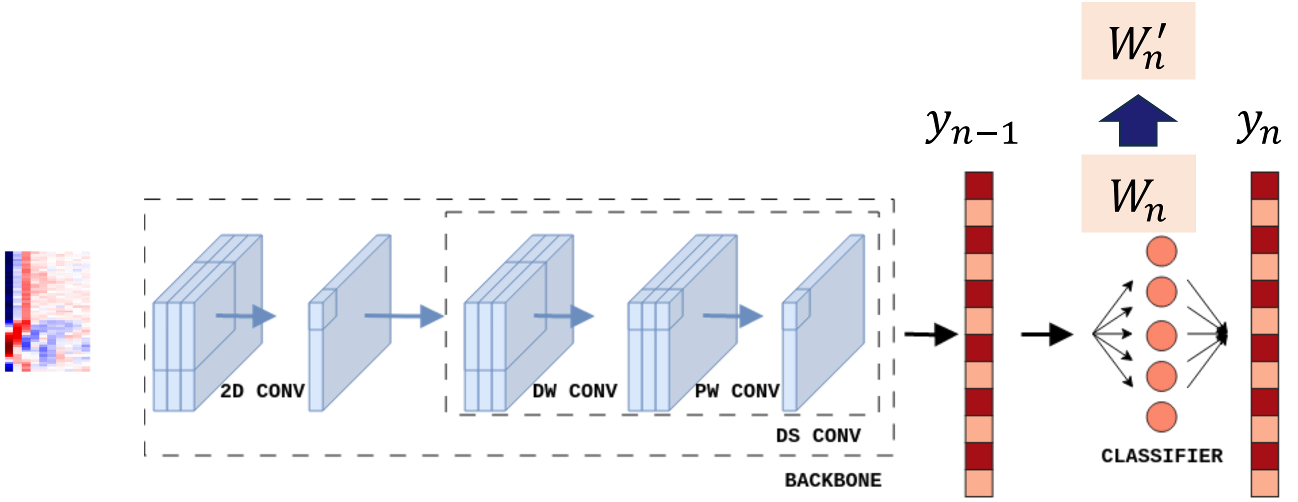
$$\frac{\partial L}{\partial W_n} = \frac{\partial L}{\partial y_n}\frac{\partial y_n}{\partial W_n}$$

$$L_{CE} = -\sum y_{gt}\log(y_n)$$



Backward pass – update the weights

```
pulp_backbone_int8_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);
```



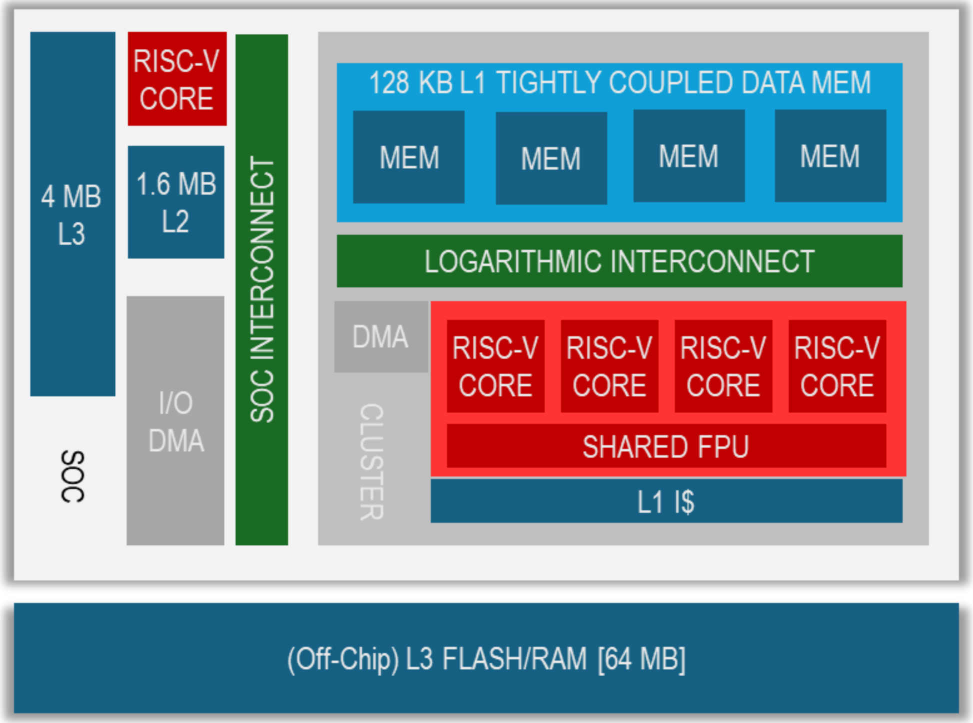
$$y_n = W_n \cdot y_{n-1}$$

$$\frac{\partial L}{\partial W_n} = \frac{\partial L}{\partial y_n}\frac{\partial y_n}{\partial W_n}$$

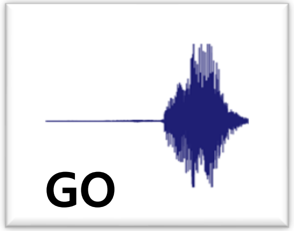
$$W_n' = W_n - \eta \cdot \frac{\partial L}{\partial W_n}$$

$$L_{CE} = -\sum y_{gt}\log(y_n)$$

y_g

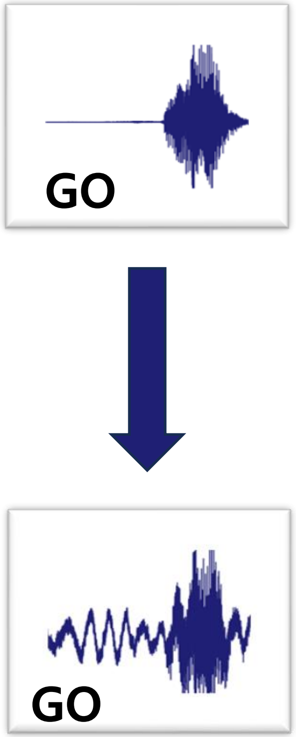


Domain Adaptation – experimental setup



- Google Speech Commands [Warden2018]
- 1-second audio @ 16 kHz
- {train,ODDA}:validation:test – 80:10:10

Domain Adaptation – experimental setup

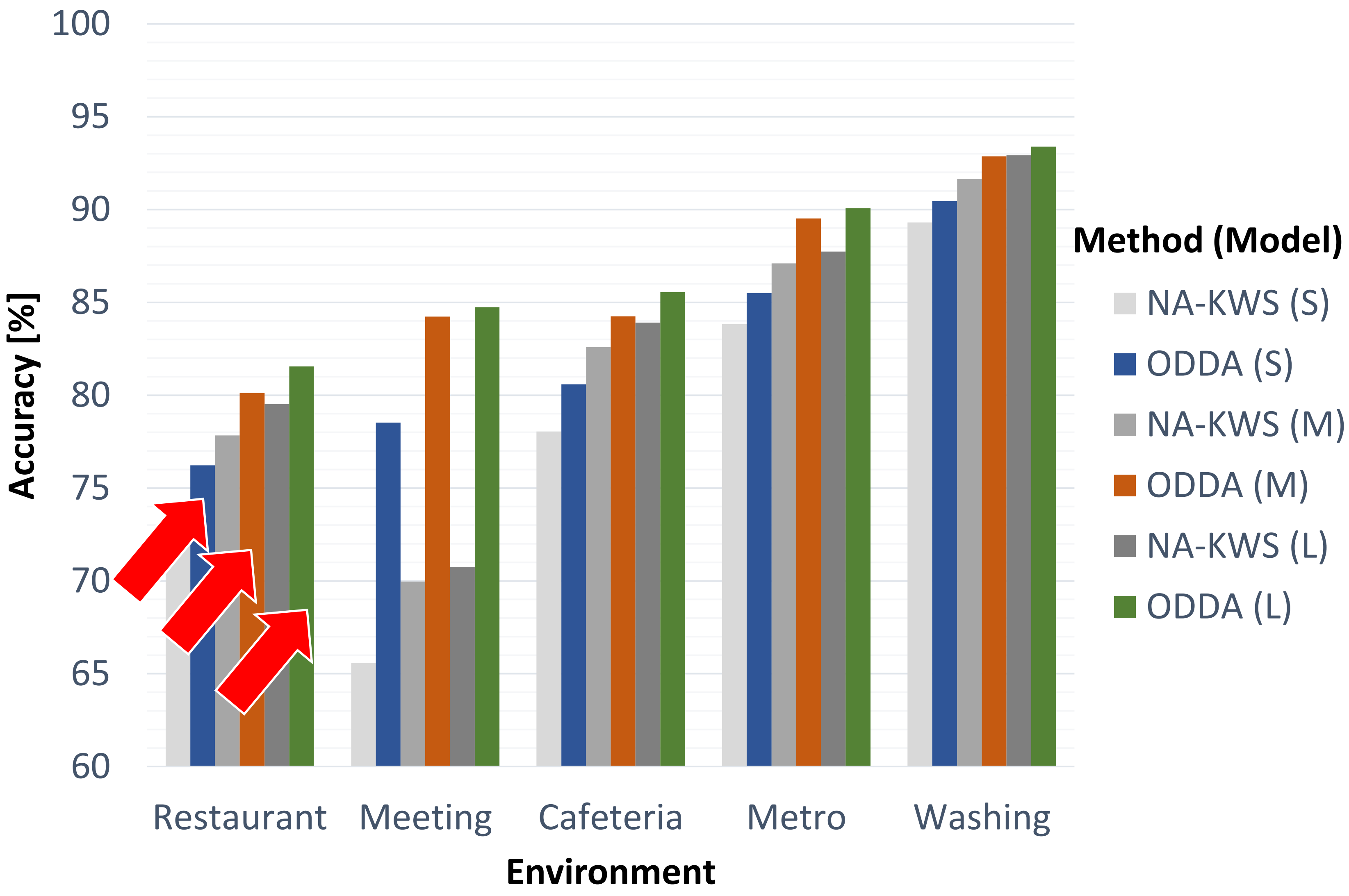


- Google Speech Commands [Warden2018]
- 1-second audio @ 16 kHz
- {train,ODDA}:validation:test – 80:10:10

- DEMAND [Thiemann2013]
- Real-world noises; SNR = 0 dB
- 5 leave-one-out adaptation targets (cafeteria, restaurant, meeting, metro, washing)

Domain Adaptation increases KWS accuracy in all environments

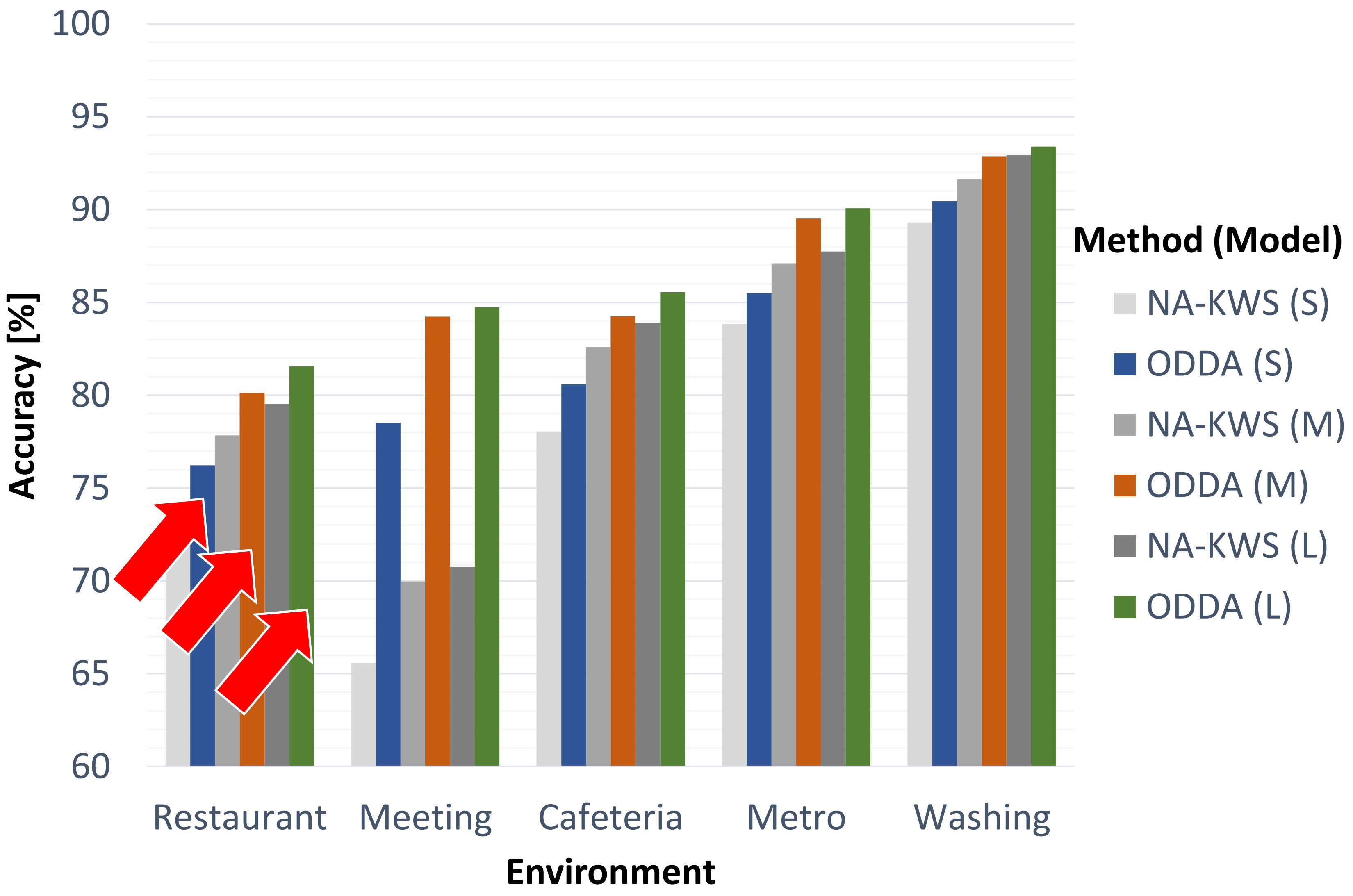
- Accuracy increases by 1%-14% on GSC-12 over noise robust NA-KWS models



Domain Adaptation increases KWS accuracy in all environments

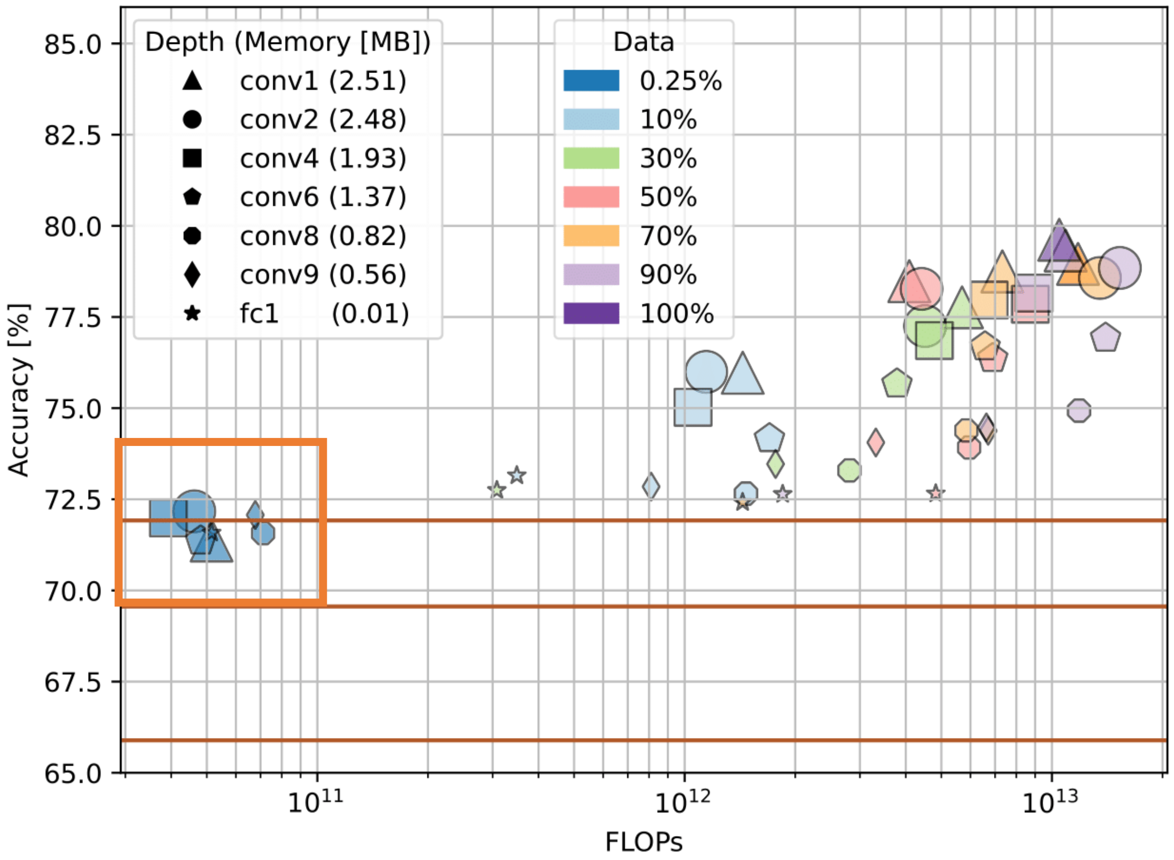
- Accuracy increases by 1%-14% on GSC-12 over noise robust NA-KWS models
- The impact of ODDA increases for models with lower capacity

DS-CNN Model	Params. [kB]	Compute [MFLOPs]
S	23.7	2.95
M	138.1	17.2
L	416.7	51.1



Resource-constrained On-Device Domain Learning

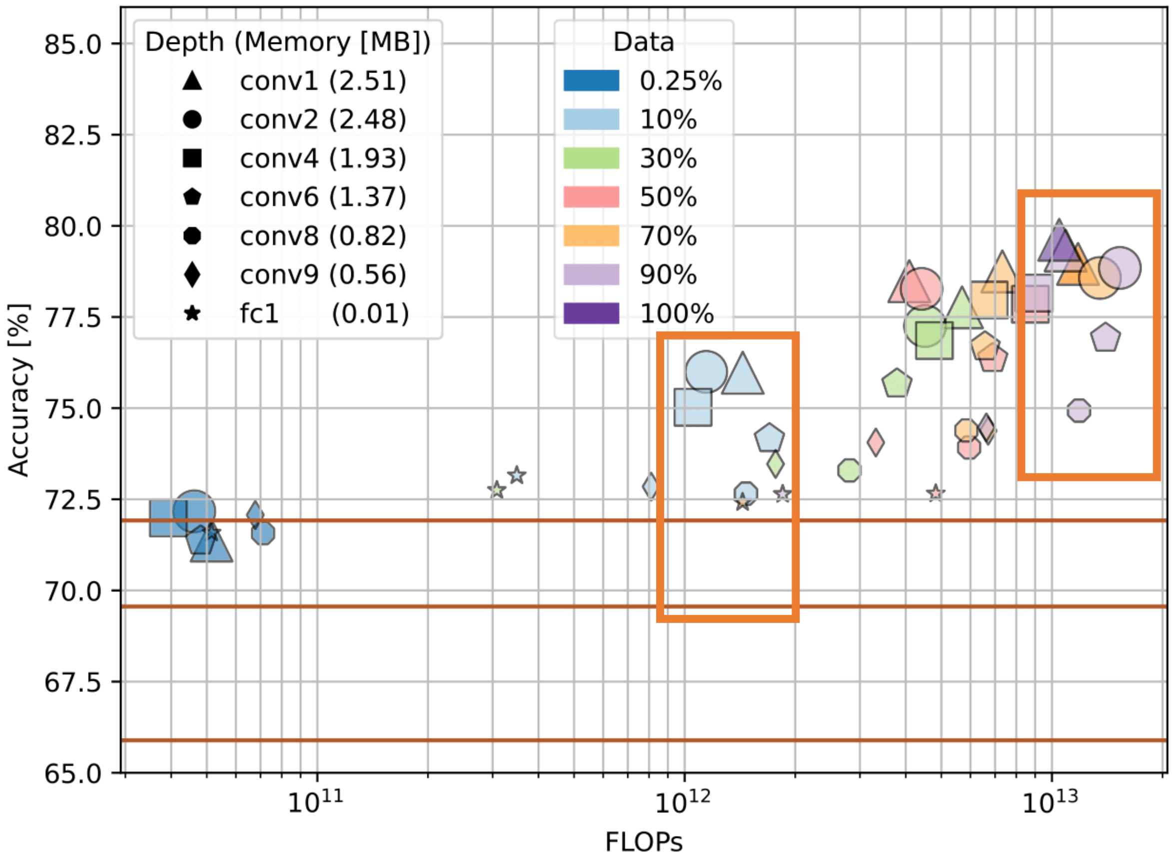
- Update ***fc_1* layer** of DS-CNN S
  - 10 kB on-chip L1 memory
  - 3 MB storage for pre-recorded samples
  - DS-CNN S_{ODDA} = DS-CNN $S_{NA\text{-}KWS}$ **+ 5.5%**



Resource-constrained On-Device Domain Learning

- Update ***fc_1* layer** of DS-CNN S
  - 10 kB on-chip L1 memory
  - 3 MB storage for pre-recorded samples
  - DS-CNN S_{ODDA} = DS-CNN $S_{NA\text{-}KWS}$ **+ 5.5%**

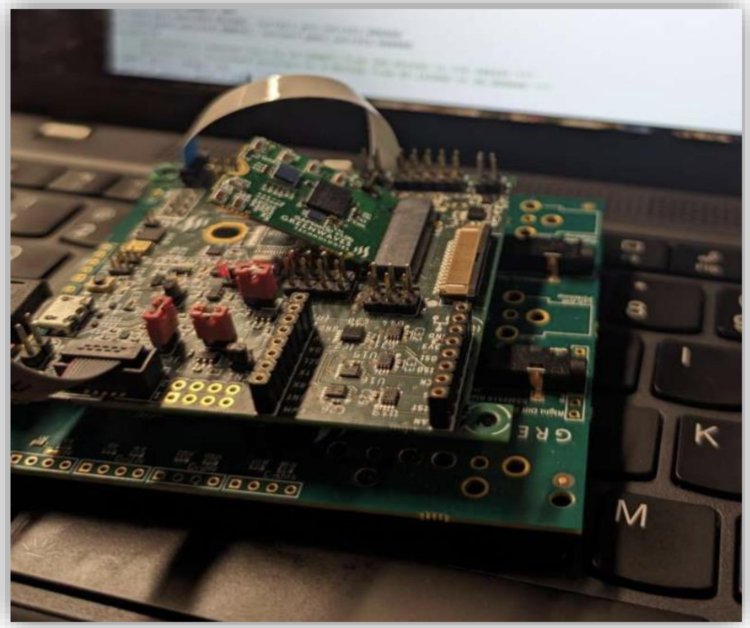
- Refine backbone and classifier
  - +1.2% over fc_1 update using 10% of pre-recorded samples
  - +6% over fc_1 update using 100% of pre-recorded samples



Implementation on GAP9

- Greenwaves GAP9 – based on PULP Vega [Rossi2022]

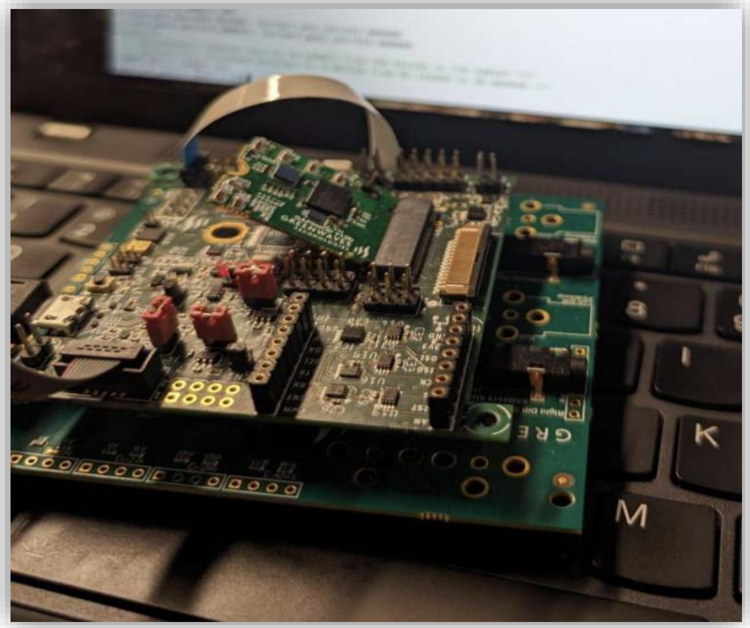
- Low-power mode: 240 MHz, 650 mV
  - On-device learning in ½ mJ, ready in 11 ms per sample



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

Implementation on GAP9

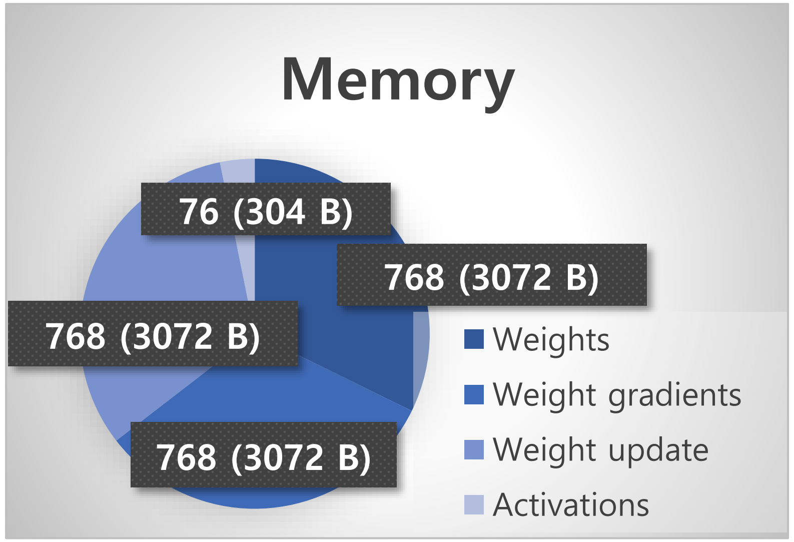
- Greenwaves GAP9 – based on PULP Vega [Rossi2022]
- Low-power mode: 240 MHz, 650 mV
  - On-device learning in ½ mJ, ready in 11 ms per sample
  - 10 kB of L1 memory for backpropagation



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

Implementation on GAP9

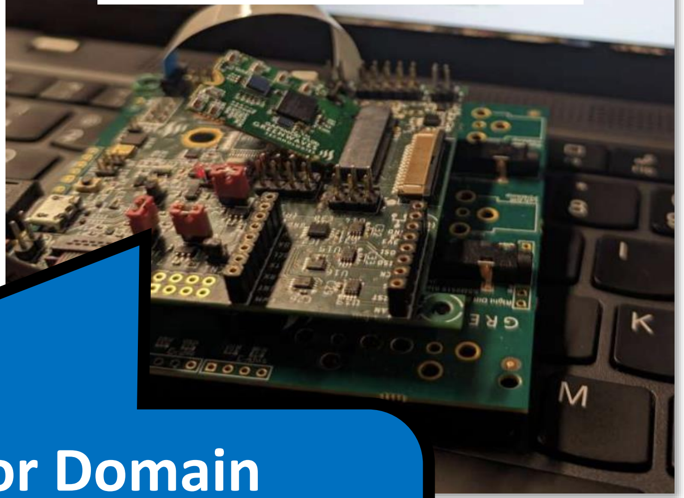
- Greenwaves GAP9 – based on PULP Vega [Rossi2022]
- Low-power mode: 240 MHz, 650 mV
  - On-device learning in ½ mJ, ready in 11 ms per sample
  - 10 kB of L1 memory for backpropagation



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

Implementation on GAP9

GREENWAVES TECHNOLOGIES



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

Live Demonstration: On-Device Learning for Domain Adaptation on Low-Power Extreme Edge Embedded Systems

C2L-C on Thursday (10:50 – 12:20)

DS-CNN Model						ergy [µJ]
S						424
M	17.2	138.1	25.5	9.18	24.16	988
L	51.1	416.7	40.9	11	55.04	2313

Conclusions

- On-Device Domain Adaptation improves the accuracy over noise robust keyword spotting models by specializing on the target noise
  - Accuracy gains up to 12% over NA-KWS at 0 dB for DS-CNN S
  - Enables word recognition in non-stationary speech noise

- On-Device Domain Adaptation operates on tinyML GAP9 platform
  - +6% over NA-KWS in only 14 s
  - 424 µJ per sample for DS-CNN S
  - 10 kB of memory for backpropagation

Conclusions

- On-Device Domain Adaptation improves the accuracy over noise robust keyword spotting models by specializing on the target noise
- On-Device Domain Adaptation operates on tinyML GAP9 platform

- What are we working on now?
  - Pairing efficient on-device-learning with state-of-the-art (linear) attention-based backbones [Scherer2024]
  - Expanding the methodology – from domain adaptation to domain (& class) incremental learning

References

[Lopez-Espejo2021] I. Lopez-Espejo et al., “A novel loss function ´and training strategy for noise-robust keyword spotting,” *IEEE/ACM TASLP*, 2021.

[Ng2022] D. Ng et al., “Convmixer: Feature interactive convolution with curriculum learning for small footprint and noisy far-field keyword spotting,” in *ICASSP*, 2022.

[Ren2021] H. Ren et al., “Tinyol: Tinyml with onlinelearning on microcontrollers,” in *IJCNN,* 2021.

[Lin2022] J. Lin et al., “On-device training under 256kb memory,” in *NeurIPS,* 2022.

[Nadalini2022] D. Nadalini et al., “Pulp-trainlib: Enabling on-device training for risc-v multi-core mcus through performance-driven autotuning,” in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2022.

[Tatman2017] R. Tatman, C. Kasten, "Effects of Talker Dialect, Gender & Race on Accuracy of Bing Speech and YouTube Automatic Captions," in *Interspeech*, 2017.

[Savoldi2022] B. Savoldi et al, "Under the Morphosyntactic Lens: A Multifaceted Evaluation of Gender Bias in Speech Translation," in *Annual Meeting of the Association for Computational Linguistics*, 2022.

[Cioflan2022] C. Cioflan et al., "Towards On-device Domain Adaptation for Noise-Robust Keyword Spotting," *AICAS*, 2024.

[Warden2018] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *arXiv preprint arXiv:1804.03209*, 2018.

[Thiemann2013] J. Thiemann et al. "DEMAND: a collection of multi-channel recordings of acoustic noise in diverse environments," in *Proc. Meetings Acoust,* 2013.

[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.

[Scherer2024] M. Scherer et al., “Work In Progress: Linear Transformers for TinyML”, in *DATE*, 2024.

Conclusions

- On-Device Domain Adaptation improves the accuracy over noise robust keyword spotting models by specializing on the target noise
  - Accuracy gains up to 12% over NA-KWS at 0 dB for DS-CNN S
  - Enables word recognition in non-stationary speech noise
- On-Device Domain Adaptation operates on tinyML GAP9 platform
  - +6% over NA-KWS in extreme-edge conditions
  - 424 µJ per epoch for DS-CNN S
  - 10 kB of memory for backpropagation
- What are we working on now?
  - Pairing efficient on-device-learning with state-of-the-art (linear) attention-based backbones [Scherer2024]
  - Expanding the methodology – from domain adaptation to domain (& class) incremental learning

Q&A

Cristian Cioflan
cioflanc@ethz.ch