Friedrich-Alexander-Universität
Technische Fakultät

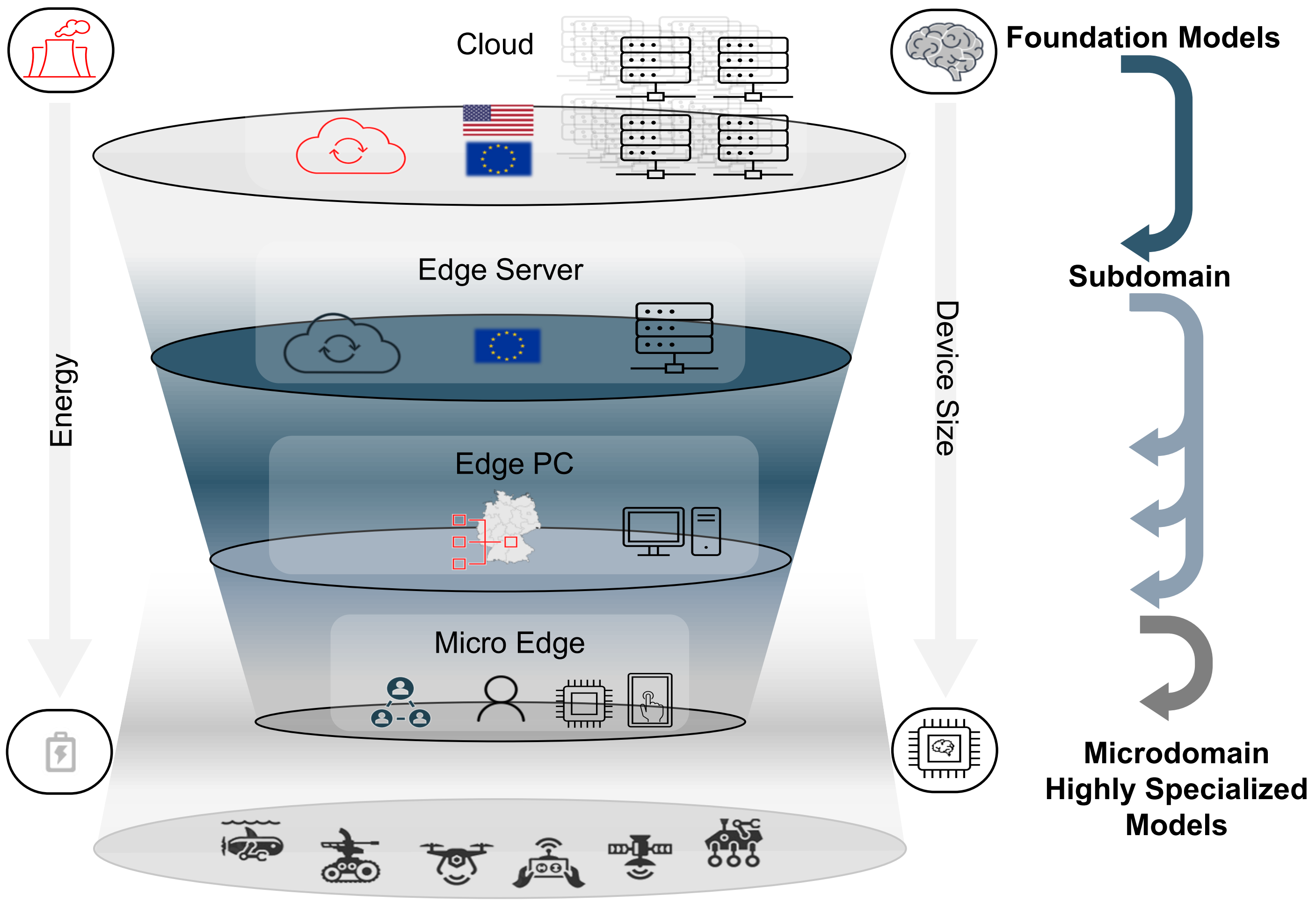
On-Device Training of Deep Neural Networks on Cortex-M Microcontrollers

Deep Learning on Narrow Resources

Mark Deutel
10.10.2025

Motivation

The Cloud-Edge-Continuum



Quelle: https://stock.adobe.com/de/images/robotics-industry-glyph-icon-set-with-robot-and-bot-technology-artificial-intelligence-ai-machine-learning-ml-automated-and-remote-control-smart-chip-android-toy-and-more-tech-symbols/265786856

Motivation

Efficient AI – Advantages

Efficient AI

Energy Efficient

Re-use of energy-efficient hardware

Fast

Local processing of data directly at the sensor

Private & Autonomous

No communication to the cloud via an error-prone network

Tiny

No extra space for servers or large industrial PCs required

Motivation

DNN Deployment on Microcontrollers – An Easy Task?

Deep Neural Network Architectures[1]

Metrics	AlexNet	VGG 16	ResNet 50
# Layers	8	16	54
Total Weights	61 M	138 M	25.5 M
Total MACs*	724 M	15.5 G	3.9 G

1. Sze, Vivienne, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. „Efficient Processing of Deep Neural Networks: A Tutorial and Survey“. 2017.

Target Micro Controllers

Metrics	Raspberry Pi Pico	Arduino Nano 33 BLE Sense
Processor	ARM Cortex M0+	ARM Cortex M4
Clock Speed	133 MHz	64 MHz
Flash memory	2 MB	1 MB
SRAM	256 KB	256 KB

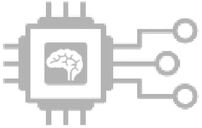
Significant gap between DNN requirements and available resources

- Low processor speed vs. large number of mathematical operations
- Strict memory limitations vs. large number of weights and big inputs/feature maps
- Floating point datatypes vs. hardware often focused on integer arithmetic

* Multiply-Accumulate Operations

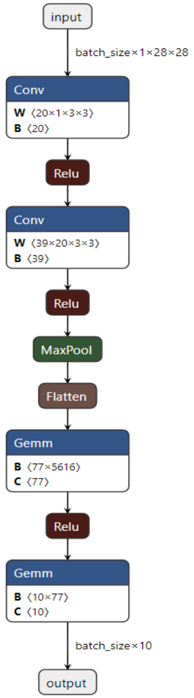
Motivation

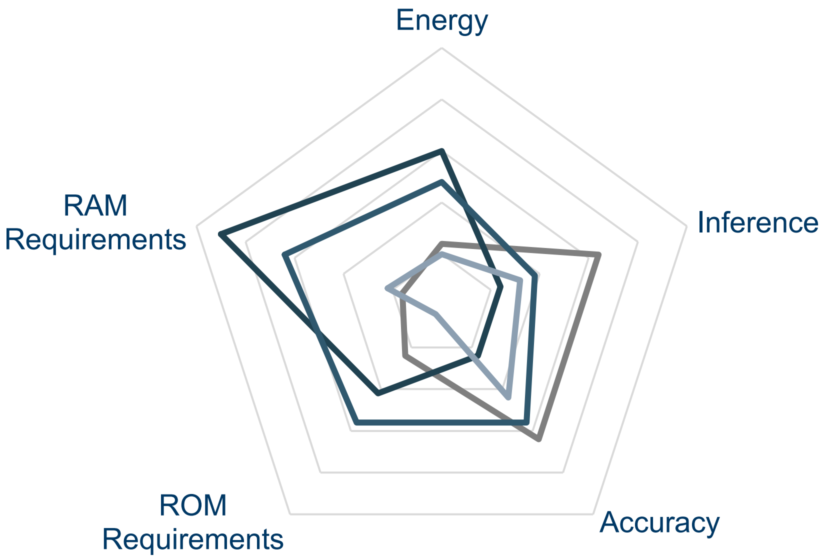
DNN Deployment on Microcontrollers – An Optimization Problem



DNN Architecture, Dataset

Deployment on Target System





Motivation

DNN Deployment on Microcontrollers – Domain Shift

The world we live in is dynamic and ever-changing. The data used to train a model is highly unlikely to be the same as what it will encounter in the real world.

Training

92% Accuracy

Deployment

75% Accuracy

DNN Deployment on Microcontrollers

A Fully-Automated End-to-End DSE Pipeline

Training & Compression → Lowering & Mapping → Continual Fine-Tuning

Dataset, Model, Hyper-parameter

DNN Pruning & Quantization

Pruning

→ How to prune?
→ What to prune?
→ When to prune?

Quantization

→ How to perform?
→ When to apply?

Deployment on Target

Data Representation

→ Optimized Memory Layout

Target Code Generation

→ Convert Graph to Code
→ Graph Optimizations
→ Algorithmic Optimization

On-Device Training

[DHM+24a], [DSP+25]

Fully Quantized Training

→ Quantized Backpropagation
→ Partial Gradient Updates

Unsupervised Learning

→ Variational Autoencoder for unsupervised classification

Deployed Model

Suggest next set of Hyperparameters using Multi-Objective Optimization based on accuracy, ROM, RAM, and FLOPS

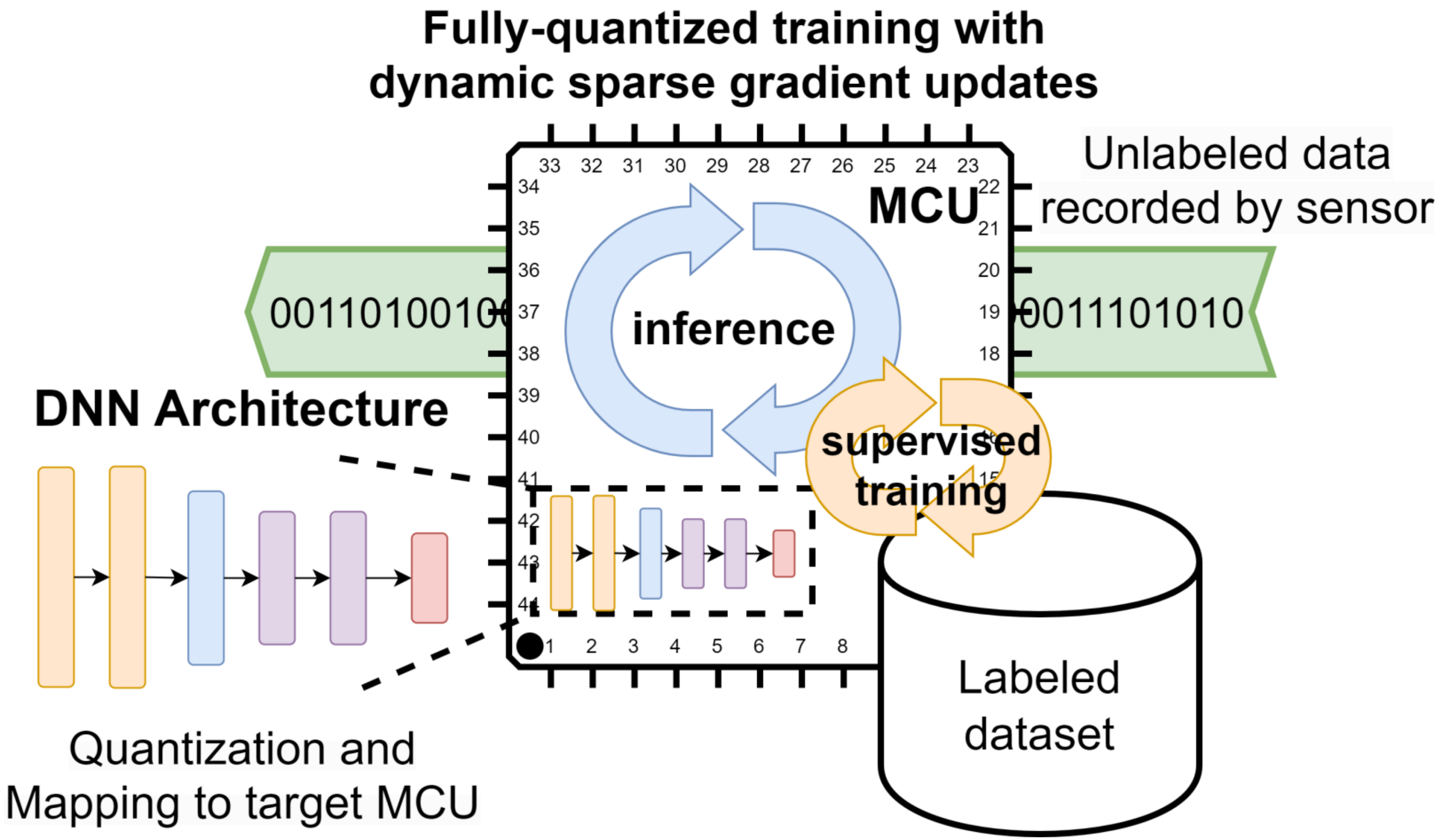
offline | **online**

- Introduction and Motivation
- On-Device Training of DNNs on Microcontrollers
- Unsupervised On-Device Training
- Conclusion

On-Device Training of DNNs on Microcontrollers

On-Device Training on Microcontrollers

Framework [DHM+24a]

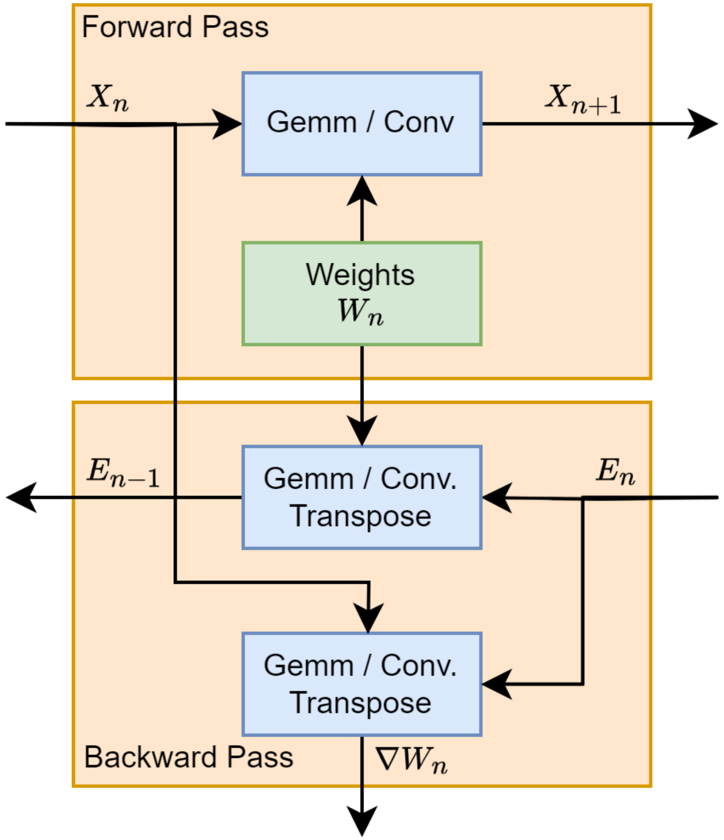


On-Device Training on Microcontrollers

Challenges [DHM+24a]

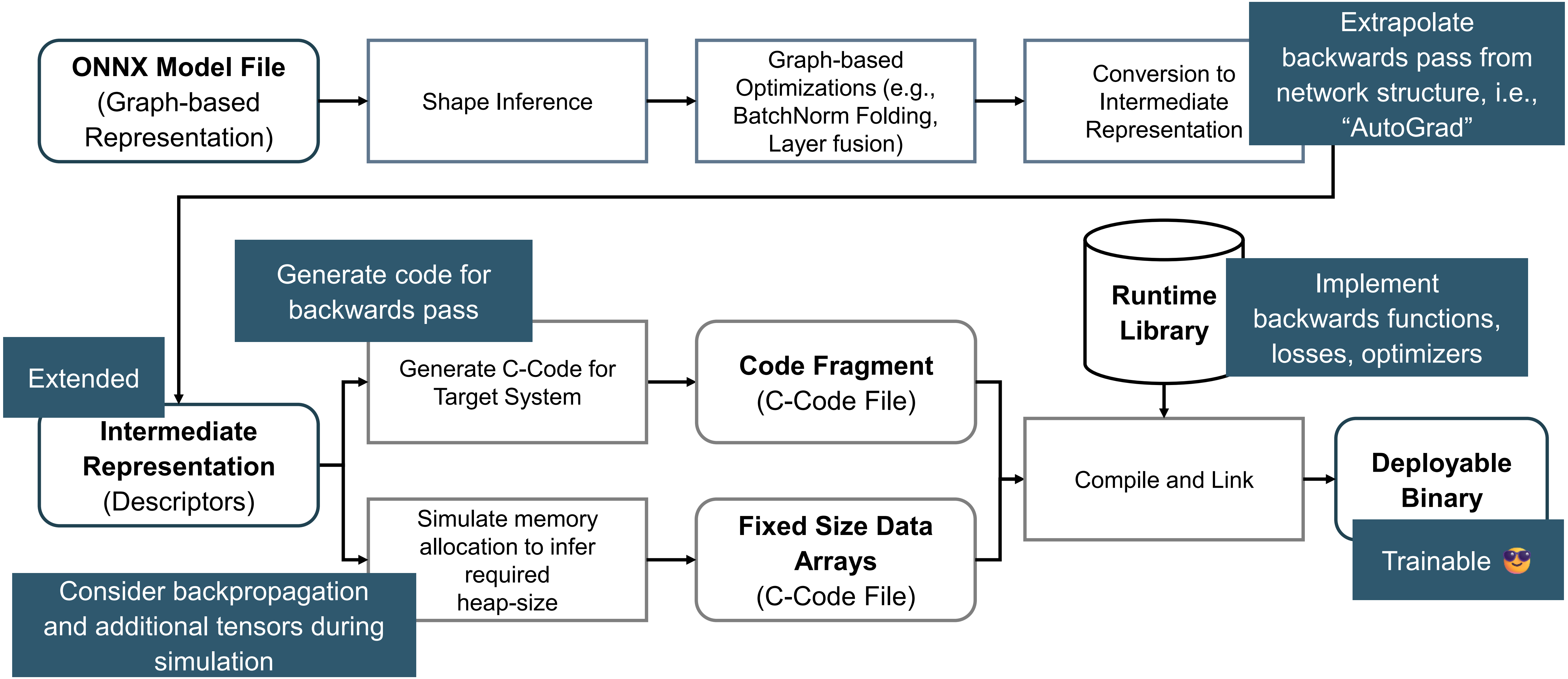
*The main challenge is to handle the additional **computational overhead** and **memory requirements** within the limitation of resource constraint MCUs*

- **Backpropagation (BP):**
  - Requires the calculation of one/two partial derivatives of each $f_l(x)$ (i.e., the function executed in the forward pass)
  - Some tensors from the forward pass need to be saved for backpropagation
- **Stochastic Gradient Descent (SGD):**
  - **Weights** are **updated** over a (mini-)batch of samples
  - Accumulate partial derivatives of each sample in a gradient buffer until one batch is full
- How to get **labels** required for training?
  - More about this later!



On-Device Training on Microcontrollers

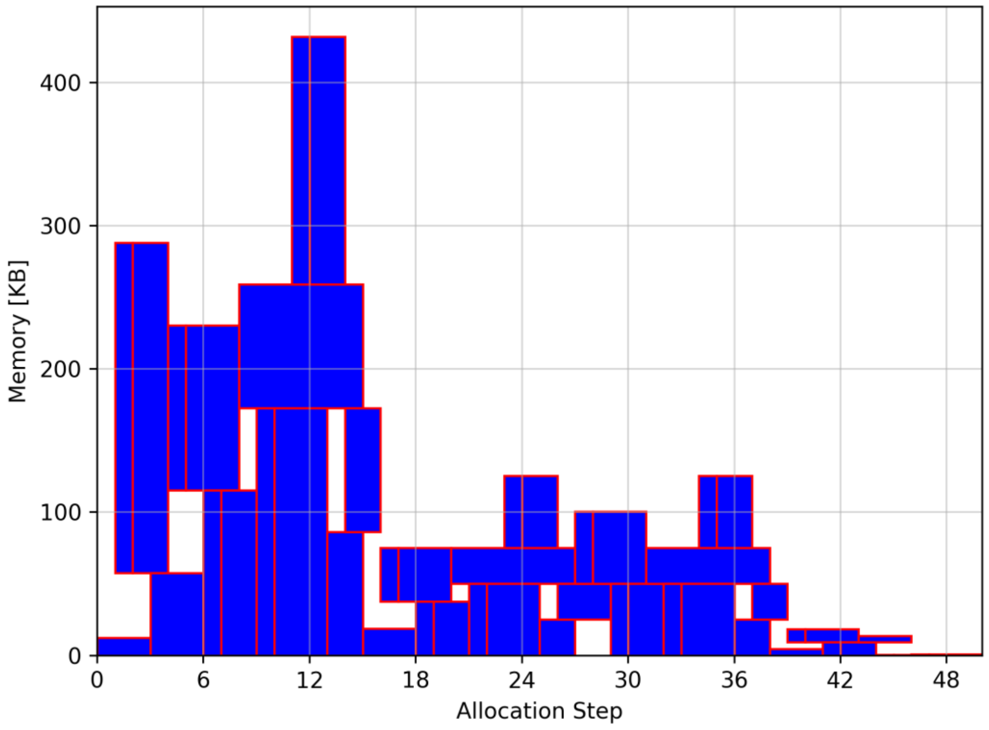
Deployment Pipeline [DWM+23, DHM+24a]



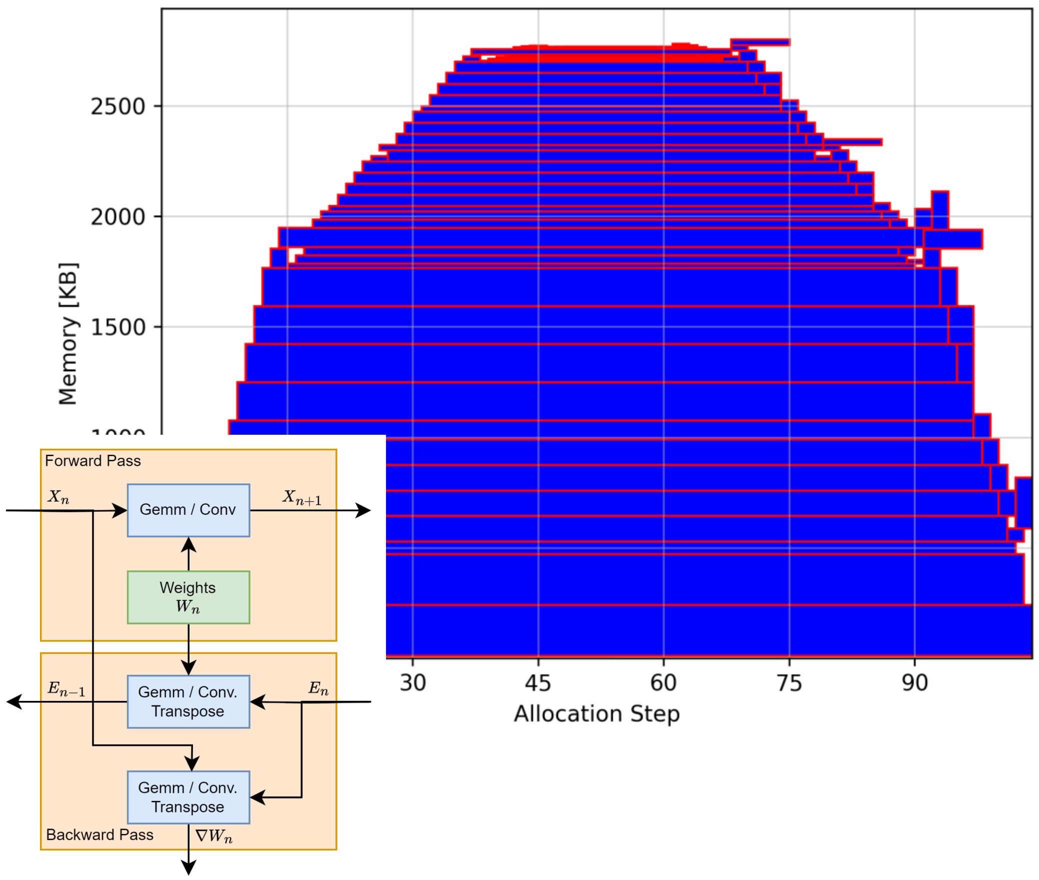
On-Device Training on Microcontrollers

Memory Allocation [DHM+24a]

Inference

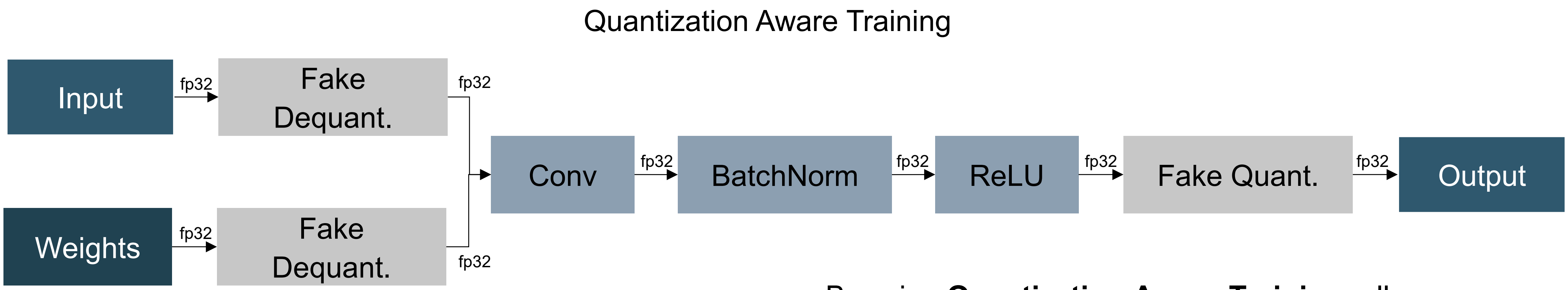


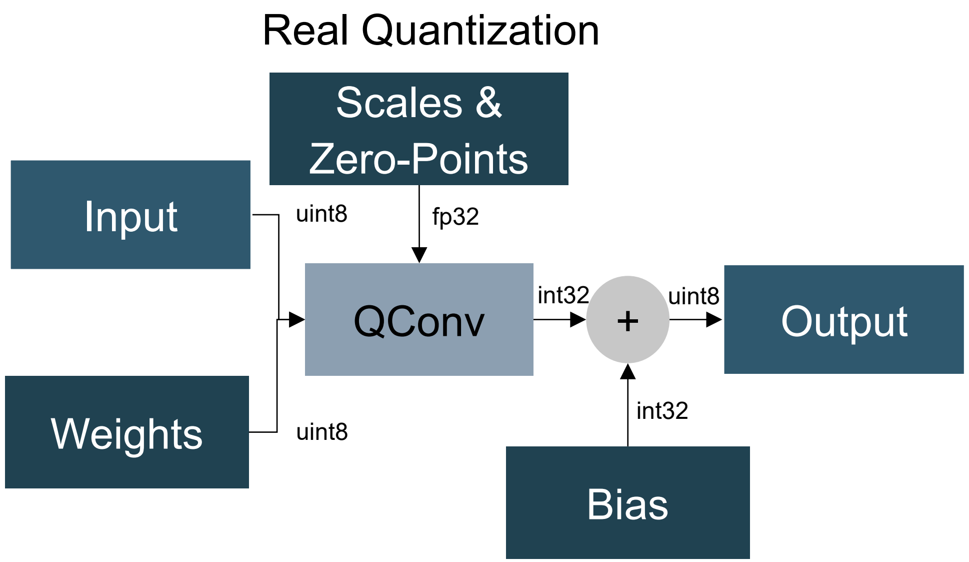
Training



Training Quantized Neural Networks

Quantized Backpropagation [DHM+24a]





- By using **Quantization Aware Training**, all memory and algorithmic optimizations used to enable deployment on MCUs would be lost
- **Real Quantization** fixes this issue, however …
  - … using quantized tensors for BP and SGD increase chance of training failure
  - … the scaling rates used for quantization must be adapted during training without any floating-point reference weights

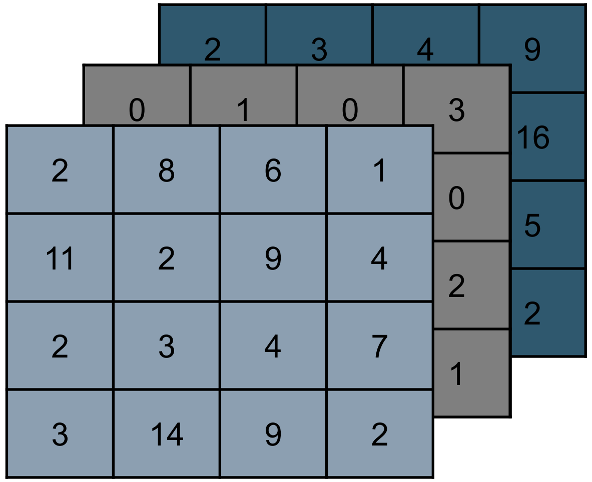
Training Quantized Neural Networks

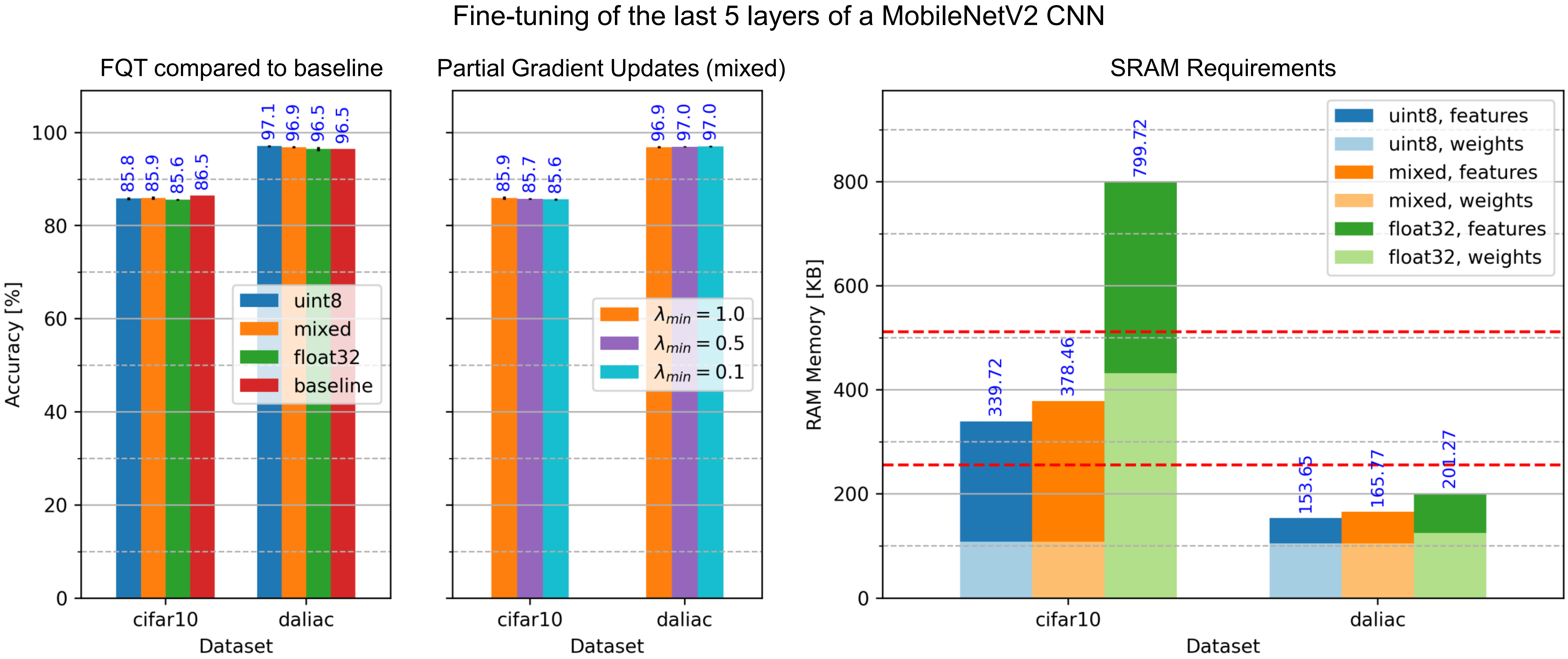
Partial Gradient Updates [DHM+24a]

- **Reduce computational overhead by only performing partial updates**
  - Prune structures from error tensors of backpropagation with a low magnitude
    - Reduces number of arithmetic operations in the subsequent operator
  - For each layer, use a dynamic update rate for each training sample

$$k = \lfloor min\{\lambda_{min} + |\varepsilon| * (\lambda_{max} - \lambda_{min}), 1\} * N \rfloor$$

    - with $0 \leq \lambda_{min} \leq \lambda_{max} \leq 1$ being hyperparameters controlling the aggressiveness of the algorithm in terminating error signals and ε being the maximum error observed so far

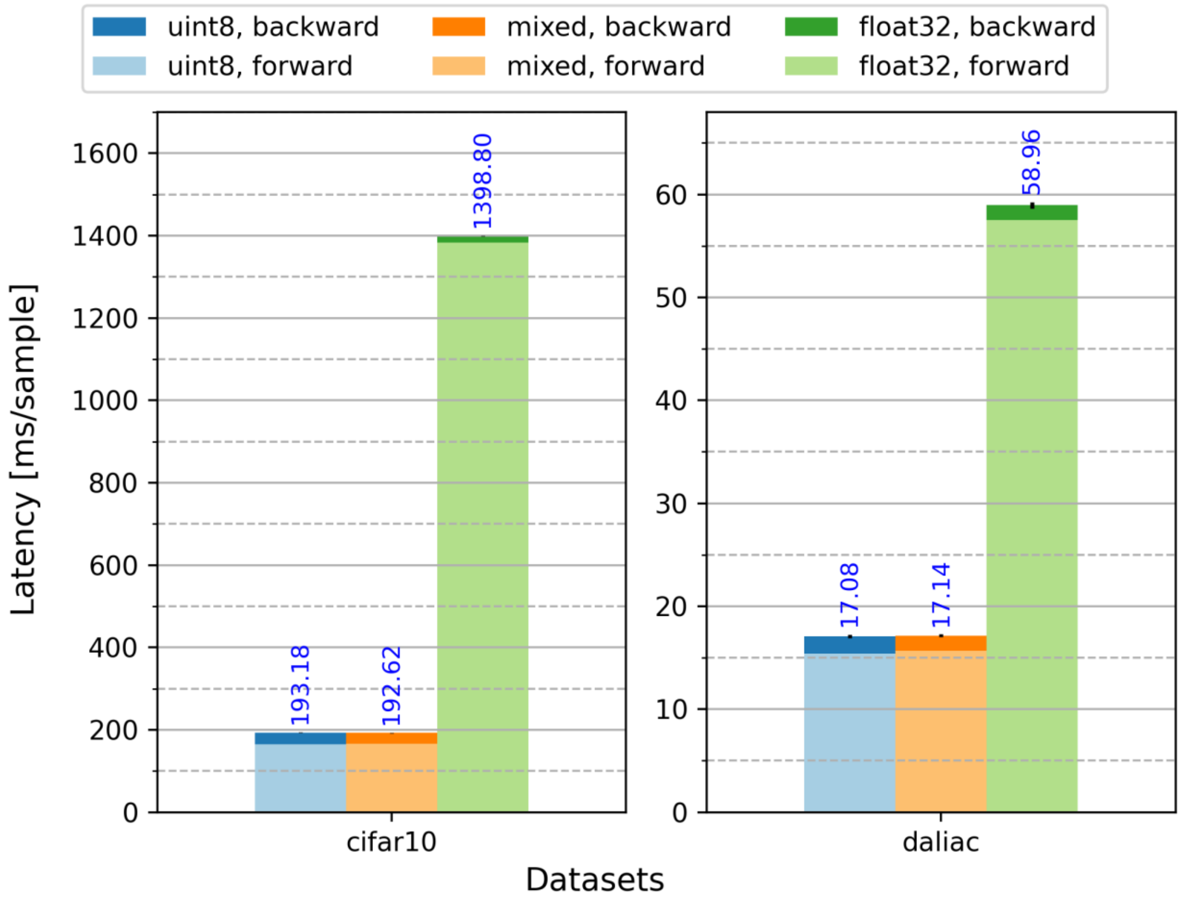


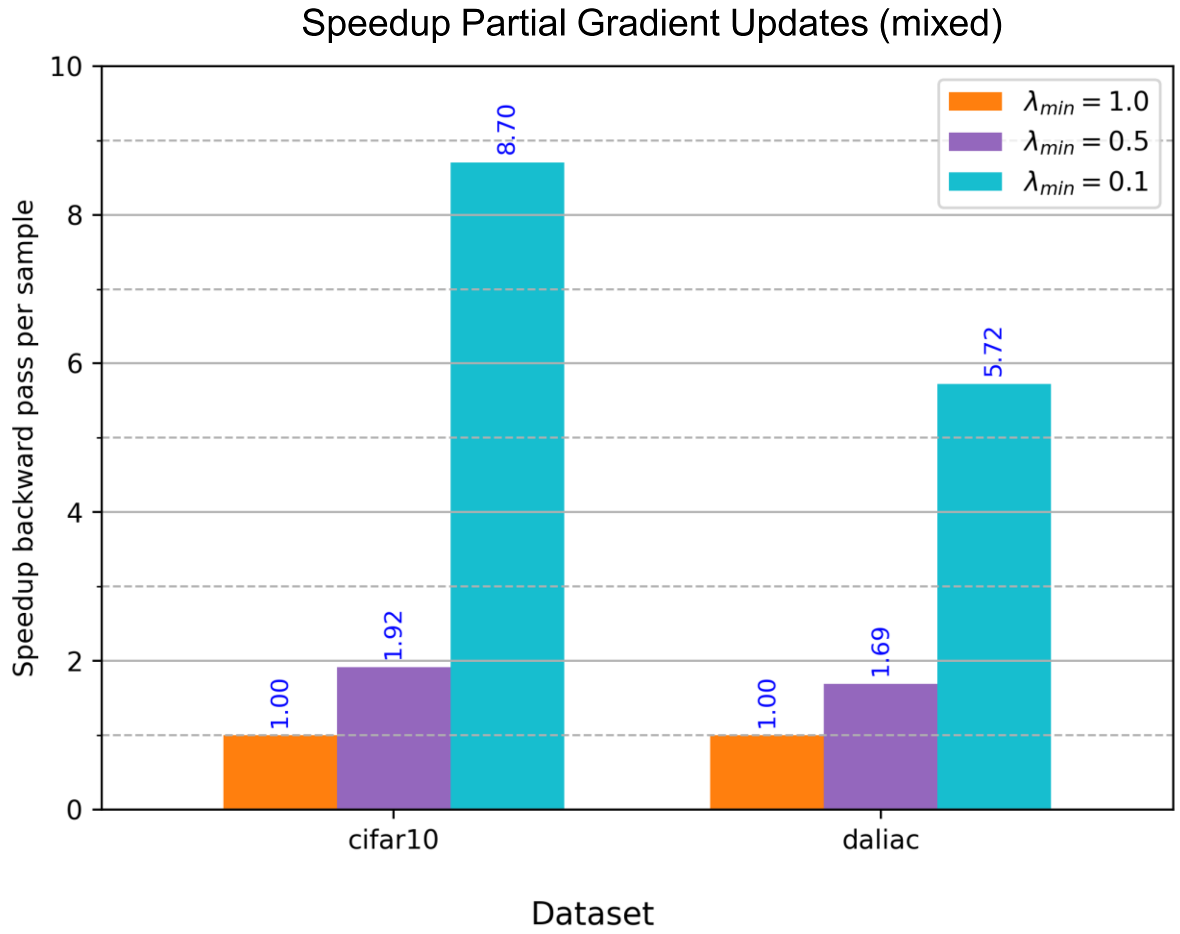


Average over 5 training runs

Fully Quantized Training – Latency [DHM+24a]

Fine-tuning of the last 5 layers of a MobileNetV2 CNN





MCU: IMXRT1062 @ 600 MHz (Teensy), 2 x 512 KB RAM, 7.75 MB Flash

Unsupervised On-Device Training

Unsupervised On-Device Training

Training Paradigms

Active Learning

- The learning algorithm can interactively query an oracle (a human)
- Useful in situations where data is abundant, but labeling is expensive
- Querying a human in an embedded environment might not be feasible

Weakly Supervised Learning

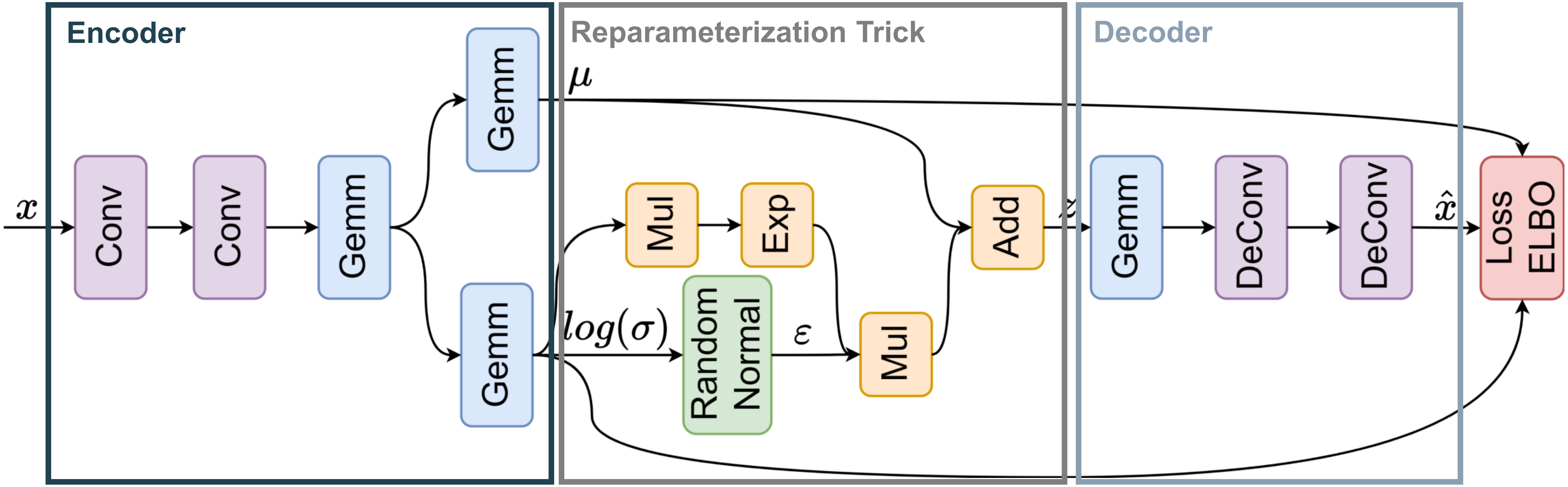
- Use a combination of labeled and unlabeled or imprecisely labeled data for training
- Automatically generate labels using functions or heuristics
- Usage of explicit or implicit prior knowledge to infer labels, usage of auxiliary information (captions, scribbles, …)
- Generative AI

Unsupervised/Self-Supervised Learning

- Learn patterns exclusively from unlabeled data
- Clustering algorithms, k-means, dimensionality reduction
- DNN uses the data itself to generate supervisory signals (autoassociative, contrastive)

Unsupervised On-Device Training

Variational Autoencoders[1] (VAEs)



Gaussian distributions are not differentiable as discrete functions→ use reparameterization trick for backpropagation

$$z_i = \mu_i + \sigma_i \epsilon_i \text{ with } \epsilon_i \sim \mathcal{N}(0,1)$$

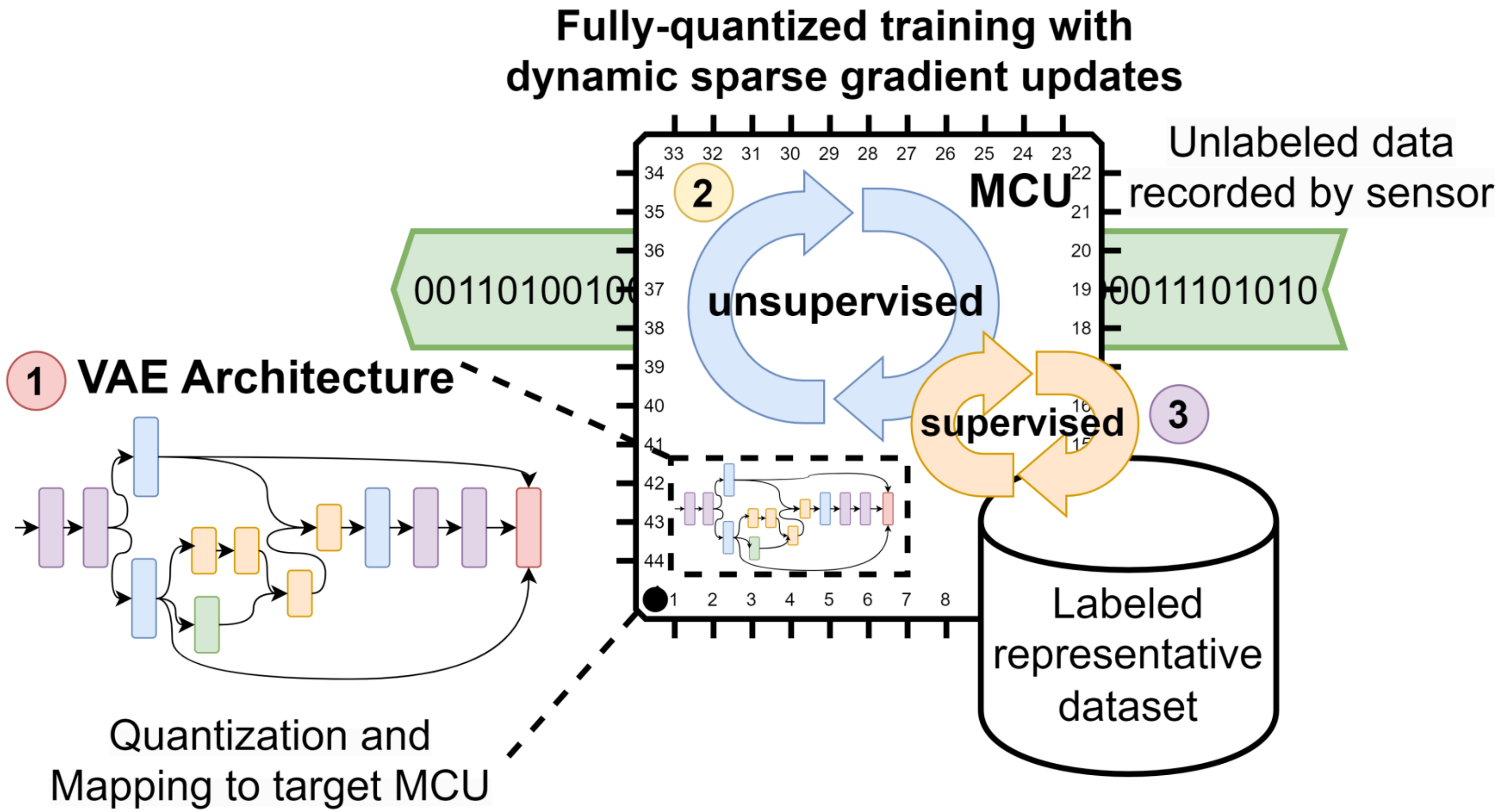
Evidence Lower Bound (ELBO) Loss

$$ELBO(q) = \mathbb{E}_q[\log p(x|z)] - KL(q(z)||p(z))$$

1. Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." 20 Dec. 2013

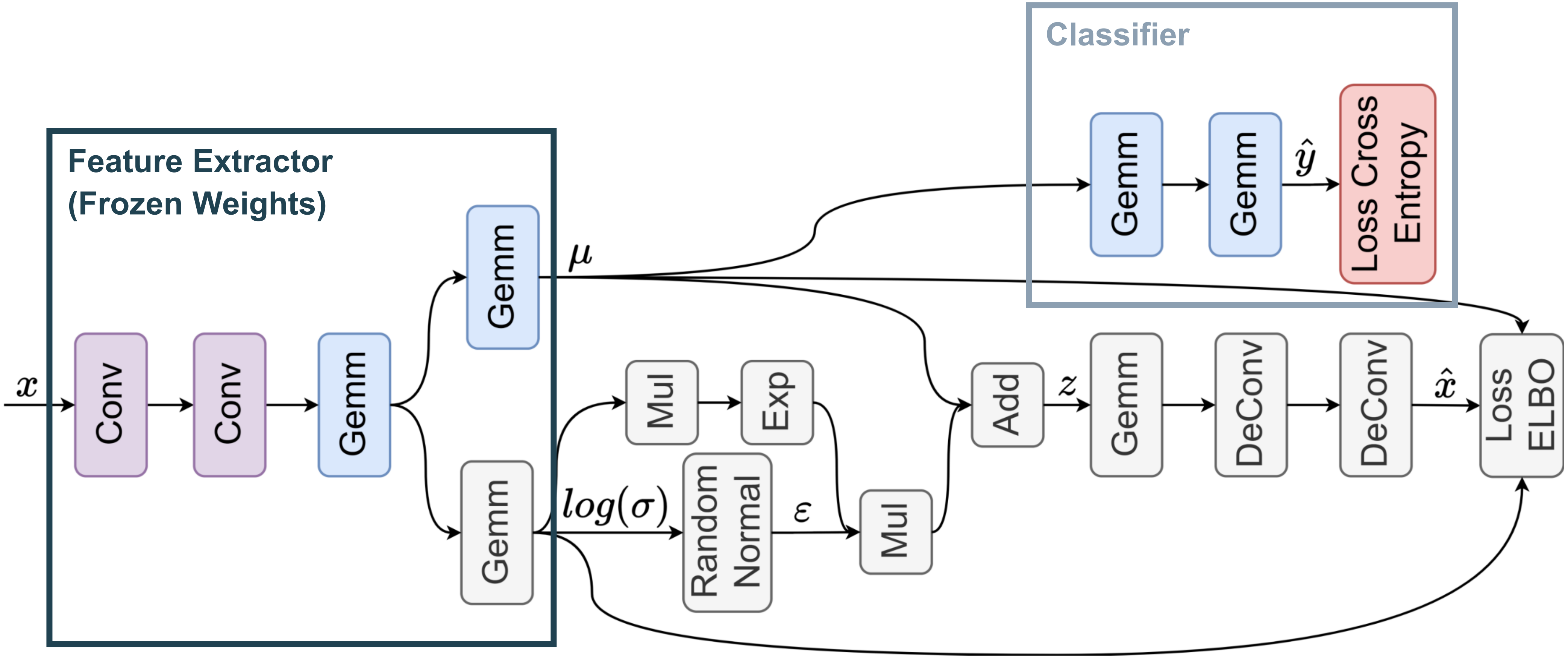
Unsupervised On-Device Training

Framework [DSP+25]



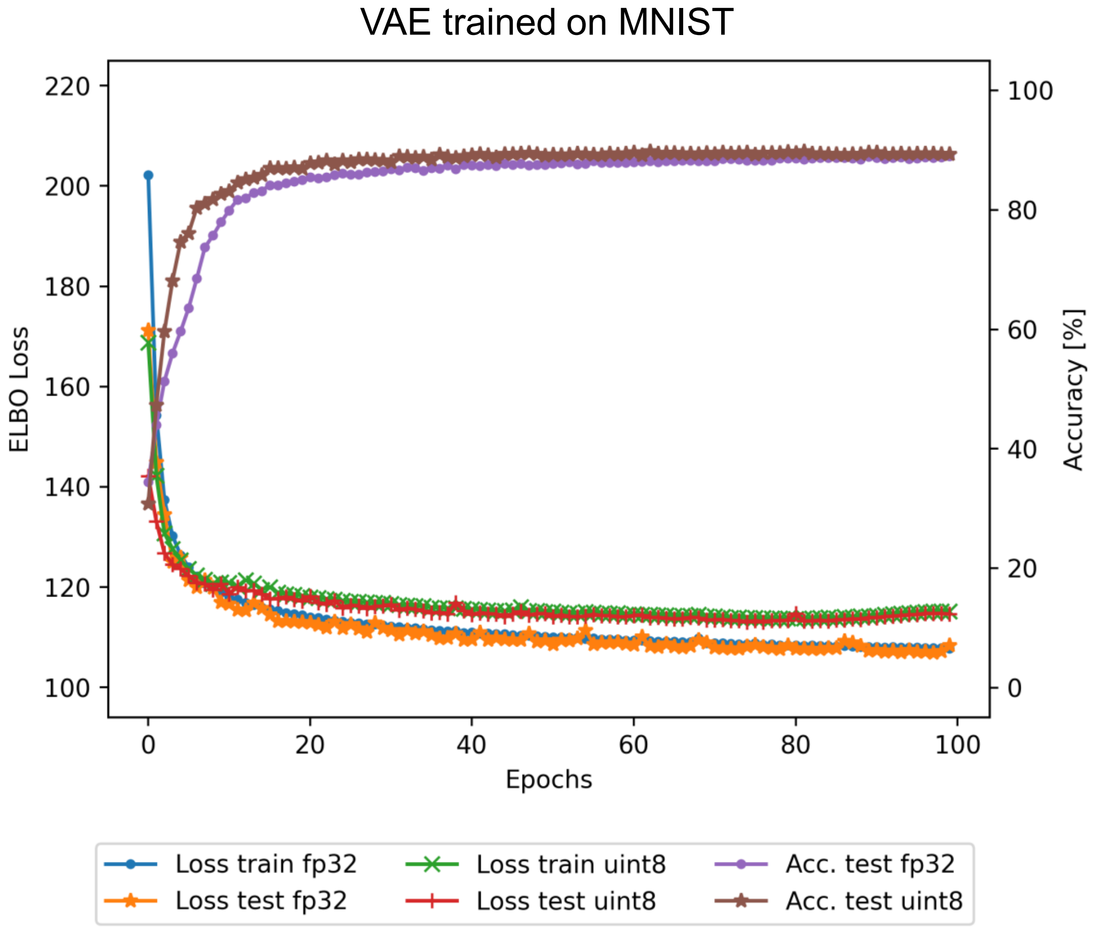
Variational Autoencoders

Supervised Training using the Encoder as a Feature Extractor [DSP+25]



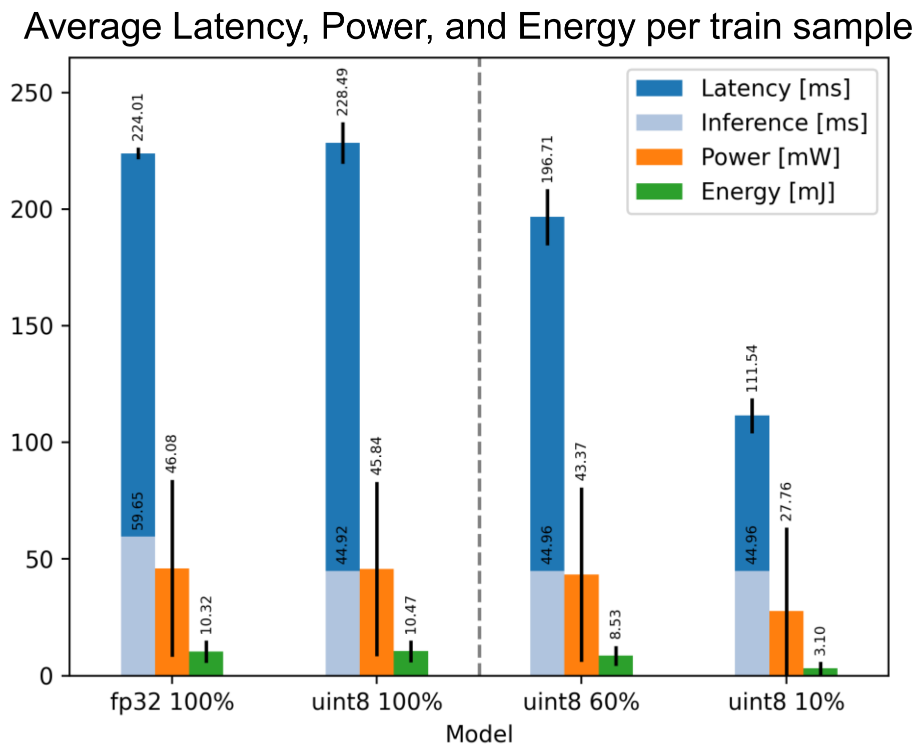
Evaluation

Unsupervised On-Device Training [DSP+25]



100 labeled samples per class used for training of classifier

Model	Activations	Weights
fp32	45.3 kB	484.1 kB
uint8	23.0 kB (↓49.2%)	221.0 kB (↓54.3%)



STM32 L4R5ZI Cortex-M4, 120MHz, Percentages denote amount of gradient updates

Conclusion

Conclusion

- **On-device training of deep neural networks** on MCUs comes with its own set of unique challenges. However, these challenges can be overcome using …
  - **Fully quantized training**
  - **Partial gradient updating**
  - Focus on fine-tuning of last layers
- To enable on-device training, existing mapping tools and runtimes need to be adapted to support training
  - i.e., backpropagation and stochastic gradient descent
- **Variation autoencoders** can be used to enable **unsupervised training** on-device, i.e., without labeled data

References

[DWM+23] **Mark Deutel**, Philipp Woller, Christopher Mutschler, and Jürgen Teich. "Energy-efficient Deployment of Deep Learning Applications on Cortex-M based Microcontrollers using Deep Compression". In: Proceedings of the 26th Workshop on Methods and Description Languages for Modelling and Verification of Circuits and Systems (MBMV), (pp. 1–12), 2023.

[DHM+24a] **Mark Deutel**, Frank Hannig, Christopher Mutschler, and Jürgen Teich. "On-device Training of Fully Quantized Deep Neural Networks on Cortex-M Microcontrollers". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 44(4), (pp. 1250–1261), 2024.

[DPS+25] **Mark Deutel**, Axel Plinge, Dominik Seuss, Christopher Mutschler, Frank Hannig, and Jürgen Teich. "Unsupervised Learning of Variational Autoencoders on Cortex-M Microcontrollers". To appear in: Proceedings of the IEEE 18th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2025.

Thank you for your attention!

Contact: mark.deutel@fau.de

Demo