Friedrich-Alexander-Universität
Technische Fakultät

Fraunhofer IIS

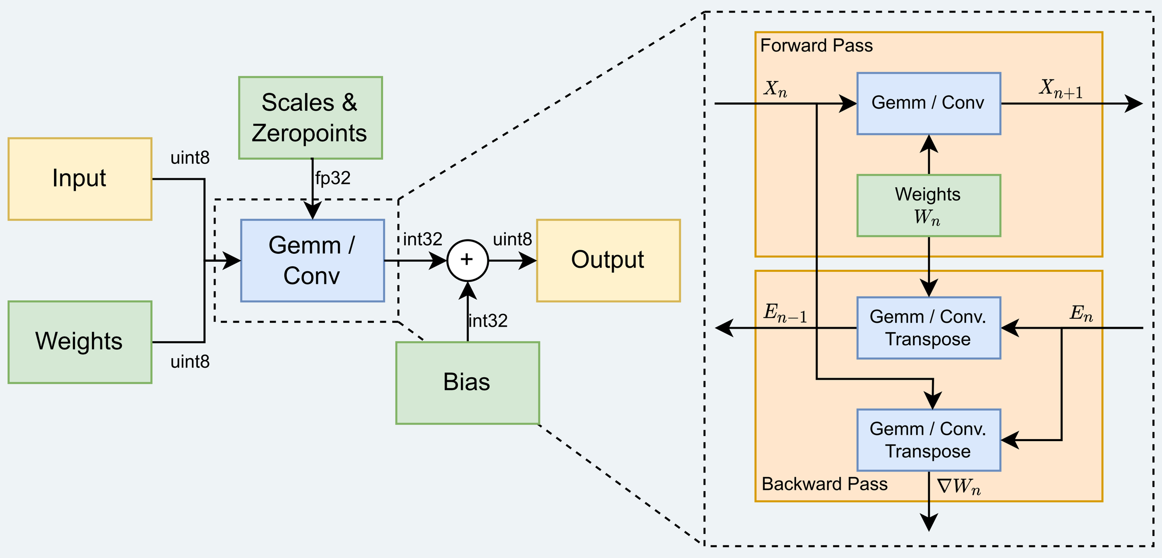
On-Device Training of Fully Quantized Deep Neural Networks on Cortex-M Microcontrollers

Mark Deutel[1,2], Frank Hannig[1], Christopher Mutschler[2], Jürgen Teich[1]
[1]Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) [2]Fraunhofer Institute for Integrated Circuits IIS

Problem Statement

- On-device training of deep neural networks (DNNs) on microcontroller units (MCUs) is challenging due to the computational and memory overheads introduced by *Backpropagation* (BP) and *Stochastic Gradient Descent*
- Training with quantized data types results in *reduced training stability* and *diminished loss convergence*
- Contribution: An extension of our DNN inference framework for MCUs [1] to support memory and computional efficient on-device training of DNNs through **fully-quantized training**, **partial gradient updating**, and **gradient standardization**

Fully-Quantized Training



- The backward pass that implements BP is derived from the forward pass during offline code generation
- Both forward and backward pass are fully quantized. For example, each element $e_{n-1} \in E_{n-1}$ can be computed from $e_n \in E_n$ and $w_n \in W_n$ as

$$e_{n-1} = \left\lfloor \frac{s_{w_n} s_{e_n}}{s_{e_{n-1}}} \sum (w_n - z_{w_n})(e_n - z_{e_n}) \right\rceil + z_{e_{n-1}}$$

- Weight tensors W update their quantization parameters based on the range $[f_{min}, f_{max}]$ in floating-point space

$$s_{w_{i+1}} = \frac{f_{max} - f_{min}}{255} \qquad z_{w_{i+1}} = \left\lfloor -\frac{f_{min}}{s_{w_{i+1}}} \right\rceil$$

- Error tensors E use a quantization range between $[-1, 1]$
- The quantization ranges for the activation tensors X are pre-determined from the training data set.

Dynamic Sparse Gradients

- The computational complexity of BP can be reduced by computing gradients only for structures of neurons that were highly activated during the forward pass
- The gradient update rate k is calculated from two hyperparameters $0 <= \lambda_{min} <= \lambda_{max} <= 1$, and $|\varepsilon|$, which is the difference between the loss of the current sample and the maximum loss observed over the entire training

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

- Based on the gradient update rate and for each layer, only the top-k structures are trained

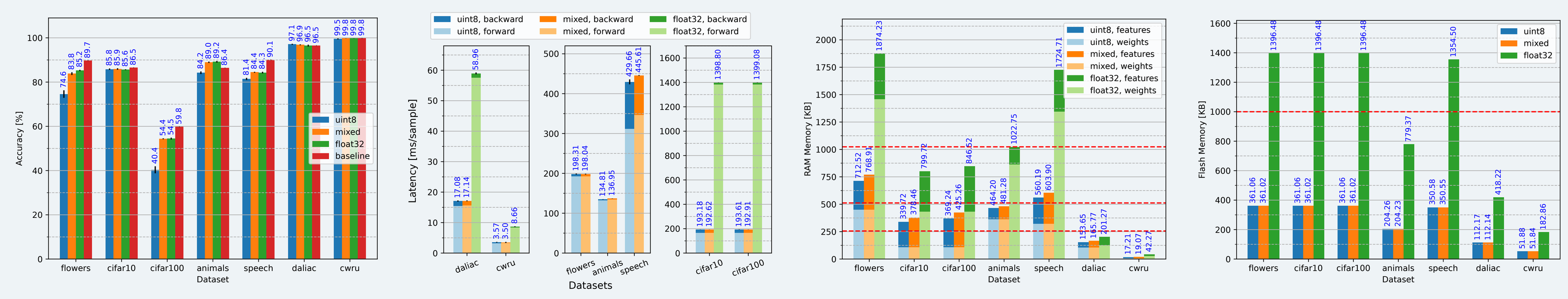
Gradient Standardization

The stability of fully quantized training can be significantly improved by standardizing the gradients ∇W while updating the weights W with a learning rate l and using the mean $\mu_{\nabla W}$ and standard deviation $\sigma_{\nabla W}$ of ∇W, similar to [2]

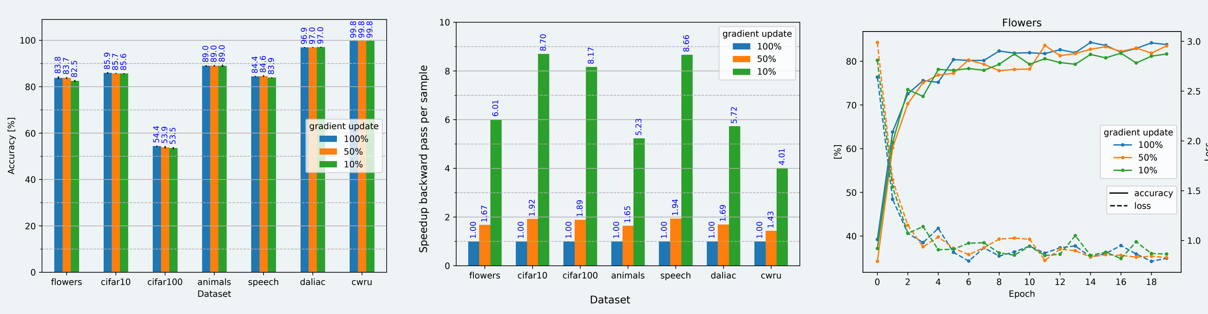
$$w_{i+1} = \frac{1}{s_{w_{i+1}}} \left[(w_i - z_{w_i})\, s_{w_i} - \ell \frac{\nabla w_i - \mu_{\nabla W}}{\sigma_{\nabla W}} \right] + z_{w_{i+1}}$$

Evaluation

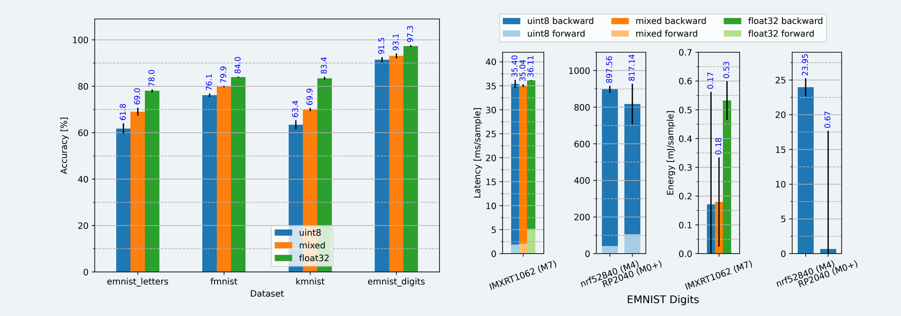
- On-device transfer learning compared to floating-point and mixed training using our MBedNet DNN architecture



- Performance on the test datasets after 20 training epochs for three gradient update rates ($\lambda_{min} \in 0.1, 0.5, 1.0$)

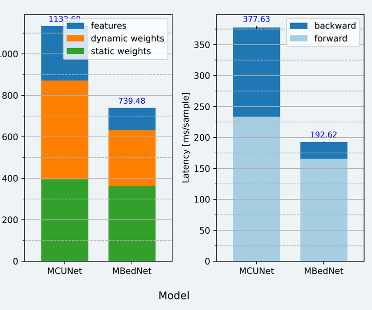


- Complete on-device training of a smaller CNN (4 trainable layers) for different datasets of the MNIST family



- Comparison between our optimizer and SGD+M+QAS [3] for updating the last two blocks of MCUNet-5FPS.

Precision	Optimizer	Accuracy (%) (MCUNet: 23M MACs, 0.48M Param)								Avg. Acc.
		Cars	CF10	CF100	CUB	Flowers	Food	Pets	VWW	
fp32	SGD-M [3]	56.7	86.0	63.4	56.2	88.8	67.1	79.5	88.7	73.3
int8	SGD-M [3]	31.2	75.4	64.5	55.1	84.5	52.5	79.5	88.7	64.9
int8	SGD+M+QAS [3]	**55.2**	86.9	64.6	57.8	**89.1**	64.4	**80.9**	89.3	73.5
uint8	**ours**	54.5	**89.5**	**65.2**	**58.5**	85.8	**66.6**	79.8	**89.3**	**73.7**



- Our optimizer provides a tradeoff between memory, latency, and accuracy that enables on-device, MCU-based re-training of DNNs equivalent to regular DNN training
- The retaining performance of our approach is comparable to other implementations such as SGD+M+QAS, while our DNN MBedNet provides a better memory and latency tradeoff than MCUNet-5FPs

References

[1] M. Deutel et al. "Energy-efficient Deployment of Deep Learning Applications on Cortex-M based Microcontrollers using Deep Compression". In: *MBMV 2023; 26th Workshop*. VDE. 2023, pp. 1–12.
[2] G. Hinton, N. Srivastava, and K. Swersky. *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.* (Date last accessed 04-June-2024). URL: http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf.
[3] J. Lin et al. "On-device training under 256kb memory". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 22941–22954.