Towards Universal 1-bit Weight Quantization of Neural Networks on Ultra-low Power Sensors

Minh Tri Lê*^, Etienne de Foras*
*TDK InvenSense; ^Inria Grenoble Rhône-Alpes

Introduction

Our tinyMLOps workflow supports:

- End-to-end model deployment on ultra-low power sensors (Arm M0+, M4) as low as ~10kB
- Standard NN layers and activations

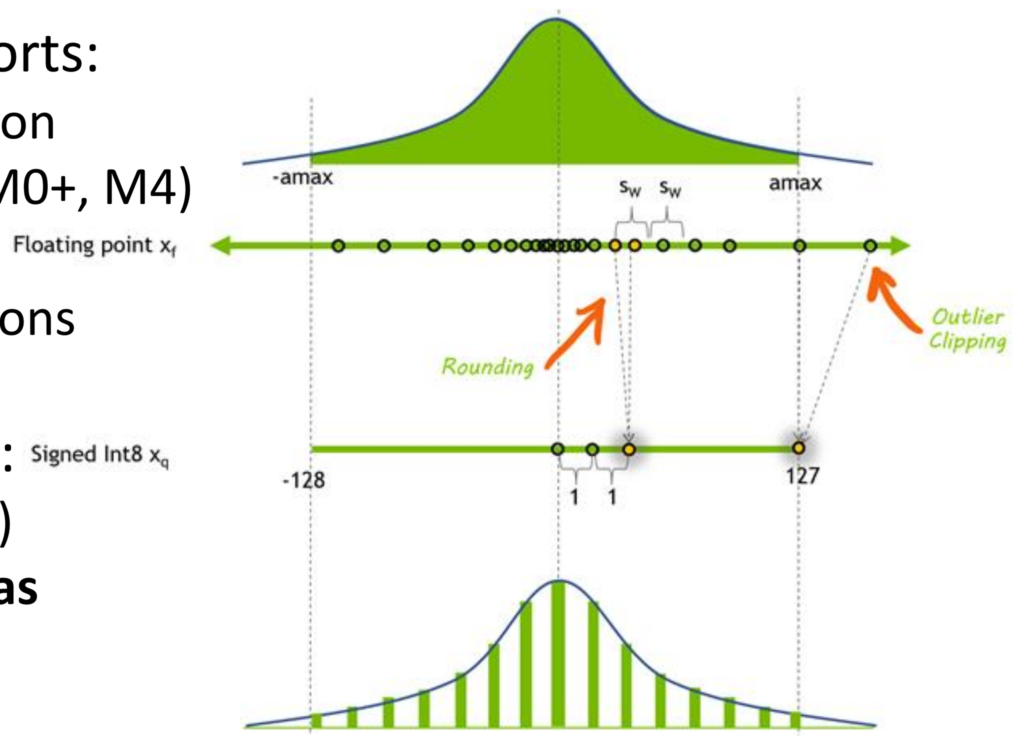

Fig.1 8-bit quantization of a floating-point tensor x_f to [-128, 127] [1]

Current quantization algorithm:

- Post-training quantization (PTQ)
- **8-bit integer weights, 32-bit bias**
- **Low performance loss (↘ 1-2% accuracy)**
- **➡ We can go lower!**
- **+** Model agnostic method
- **+** Very easy to integrate in a tinyML pipeline
- **-** Not optimized (sensitive to outliers)

Goal

- Aim for **1-bit weights:**
  - **Reduce model size by x8 (versus 8-bits model)**
  - **Faster and low-power inference**
- **Preserve acceptable performance** (accuracy, memory, latency…)
- **Hassle-free method**: few manual tweaks, seamless integration with our current tinyMLOps workflow
- **Scalable to standard NN layers** (Fully-connected, RNNs, CNN…) across many applications.

Universal 1-bit weight quantization

- PTQ of 1-bit weights is too destructive → Need weight optimization and 1-bit quantization during the training phase!
- Bias are kept in 32-bits like the activation
- **Model, framework & problem *agnostic* algorithm** → Scalable + flexible to any layer → **Allow hybrid $\{1;8\}$-bits quantization**.

We found **2 solutions** of our algorithm**: $\boldsymbol{S_1}, \boldsymbol{S_2}$.

Train model: Conjointly optimize weights for quantization and application with S_1 or S_2

Rounding strategy: Nearest, stochastic rounding[2], AdaRound[3]…

Deployment to low-power MCU: Full 1-bit quantization or Hybrid {1,8}-bits quantization

Test on MNIST

- We apply S_1 on all layers of a standard CNN.

→ Fast weight convergence towards $\{-1;\ +1\}$

- Accuracy loss <1%

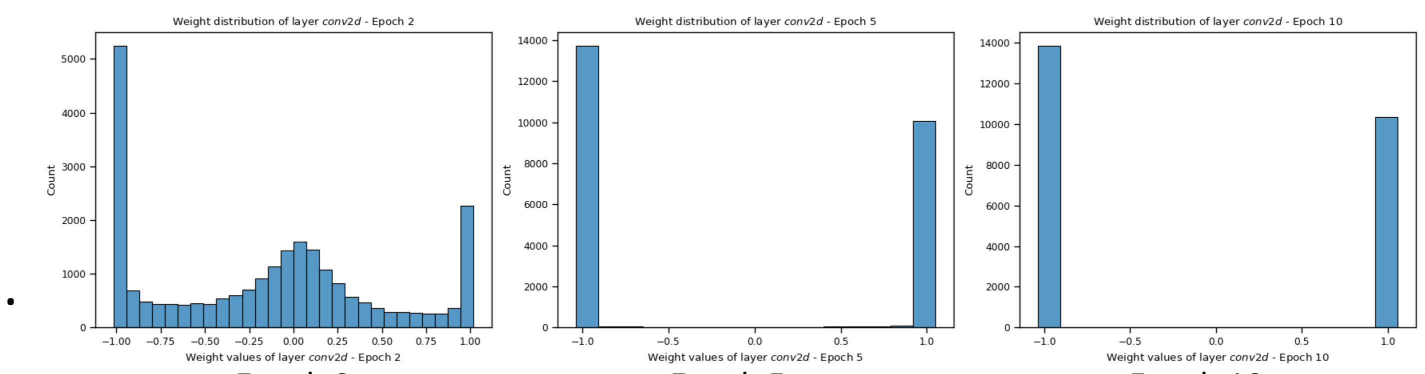

Fig. 2 Weight distribution of the first conv2d layer of a CNN (epoch {2; 5; 10})

***Under patent review*

Results on gesture recognition:

Where to apply our 1-bit quantization algorithm?

- Input, middle, output layer?
- Convolution, RNNs, Fully-connected?

Model architecture: CNN -> GRU -> FC

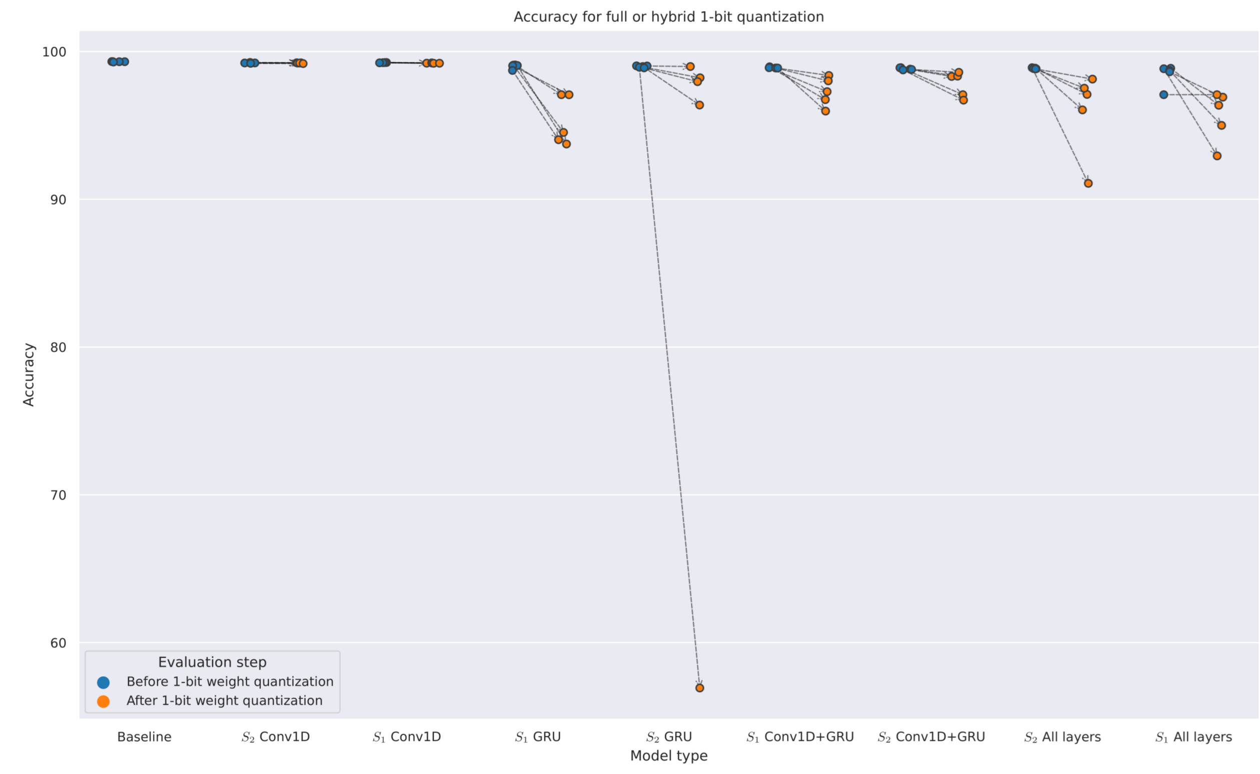

Fig. 3 Layer sensitivity to before/after binary rounding of full or hybrid quantized model (5x independent repetitions)

→ Binary convolution is less sensitive than binary GRU layers and that the output decision layer is also critical.

→ 1-bit performance is preserved for convolution only, else it is acceptable for some models.

→ Overall, $\boldsymbol{S_2}$ has less variance than $\boldsymbol{S_1}$ except when quantizing GRU only layer although $\boldsymbol{S_1}$ performs quite similarly.

The full binary quantized model is **45% smaller** than its int8 baseline. (Bias are kept in 32-bits)

Model type	Model size (bytes)
Baseline float32	1284
Baseline int8	504
Full binary baseline	276.5

Generalization to N-bits quantization:

We generalize our algorithm for N-bits quantization:

- Models are converging towards discrete weights

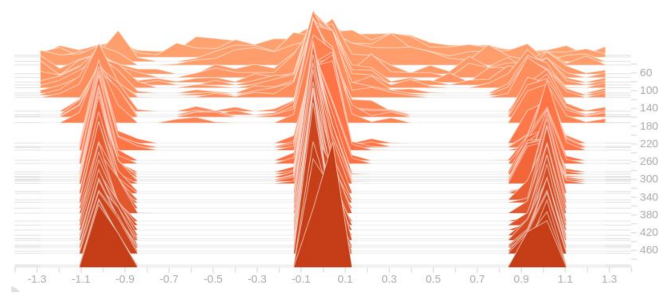

Fig. 4 2-bits quantization: GRU recurrent weight convergence. z-axis is epoch

Ternary quantization

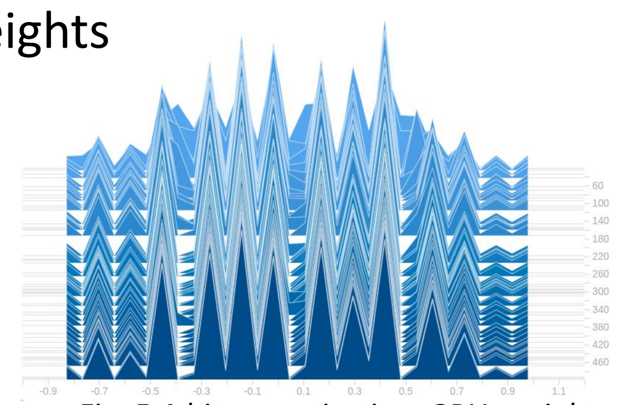

Fig. 5 4-bits quantization: GRU weights convergence. z-axis is epoch

4-bits quantization

Conclusion, future work, open challenges,

- Successfully improved our tinyML workflow by quantizing standard models down to 1-bit with a universal and hassle-free algorithm
- Enabled flexibility of per-layer hybrid quantization
- Obtained acceptable loss for 1-bit models on MNIST and gesture recognition
- Demonstrated potential for a N-bits generalization approach, and so N-bits hybrid quantization

Future work:

- Can we compensate 1-bit quantization performance loss by selecting larger baseline models? If so, which layers should we enlarge and how much?
- Comparing rounding strategies other than nearest.
- Running more extensive tests on the N-bits generalization and add hardware support for N-bits hybrid inference to leverage the power footprint gain.

References:

[1]
N. Zmora, H. Wu, and J. Rodge, "Achieving FP32 Accuracy for INT8 Inference Using Quantization Aware Training with NVIDIA TensorRT," *NVIDIA Technical Blog* 2021.

[2]
M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis, "Stochastic Rounding: Implementation, Error Analysis, and Applications," 2021.

[3]
M. Nagel, R. A. Amjad, M. van Baalen, C. Louizos, and T. Blankevoort, "Up or Down? Adaptive Rounding for Post-Training Quantization," 2020

[4]
A. Bulat, G. Tzimiropoulos, J. Kossaifi, and M. Pantic, "Improved training of binary networks for human pose estimation and image recognition." 2019.