TINYML DESIGN CONTEST - TEAM CDL-EML TU WIEN

Daniel Schnoell

Dominik Dallinger

Institute of Computer Technology at TU Wien,
Vienna, Austria,
e-mail: daniel.schnoell@tuwien.ac.at
dominik.dallinger@tuwien.ac.at
web: eml.ict.tuwien.ac.at

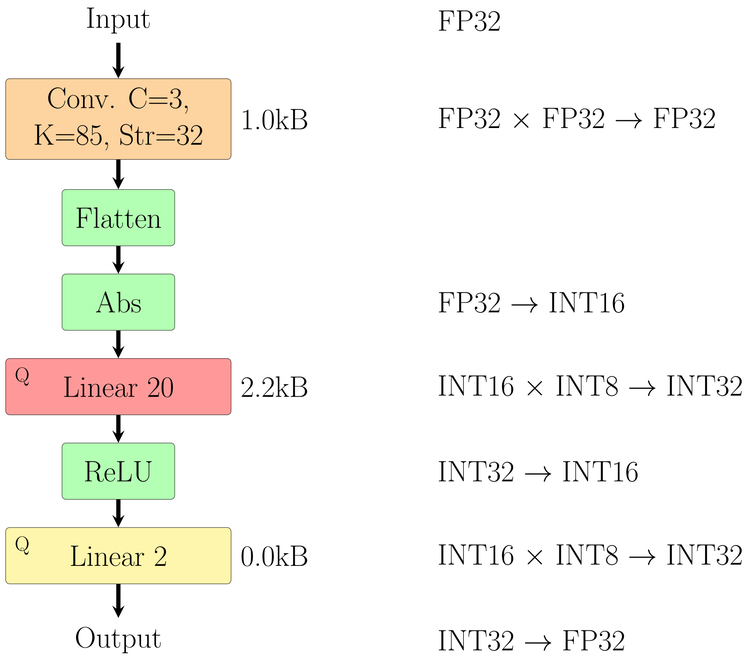
This work was supported in part by the Austrian Federal Ministry for Digital and Economic Affairs, in part by the National Foundation for Research, Technology and Development, and in part by the Christian Doppler Research Association.

TECHNISCHE UNIVERSITÄT WIEN

Institut für Computertechnik
Institute of Computer Technology

Christian Doppler Forschungsgesellschaft

MODEL – MIXED PRECISION



The Model with Layer sizes and data types used. Color based on size.

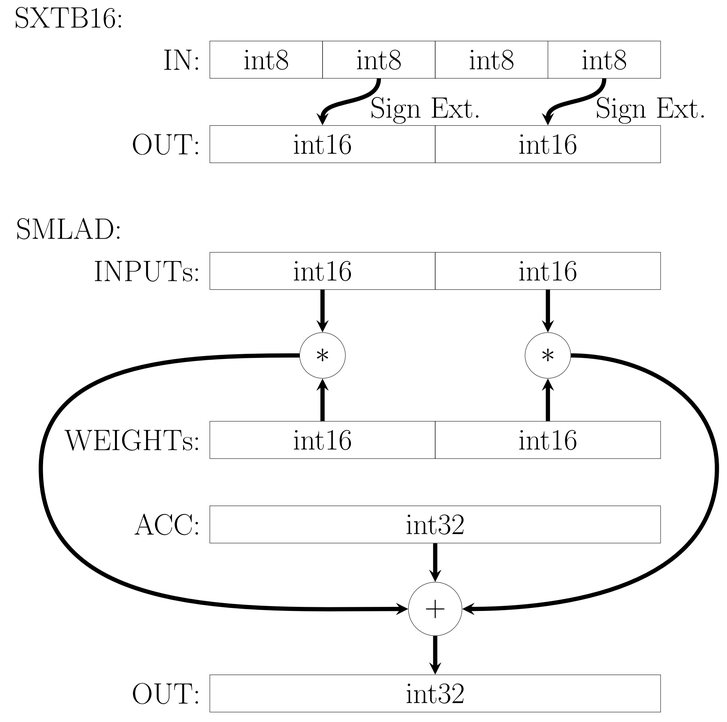
MODEL PROGRESSION

We enhanced the previous year's winning *Gatech EIC Lab Team* model [1] by altering its architecture. Specifically, we replaced the initial activation functon with an absolute value and removed one linear layer for improved performance. During our experimentation, we found that utilizing signal energy rather than raw input data was more effective, which evolved to using the absolute function as the first non-linearity.

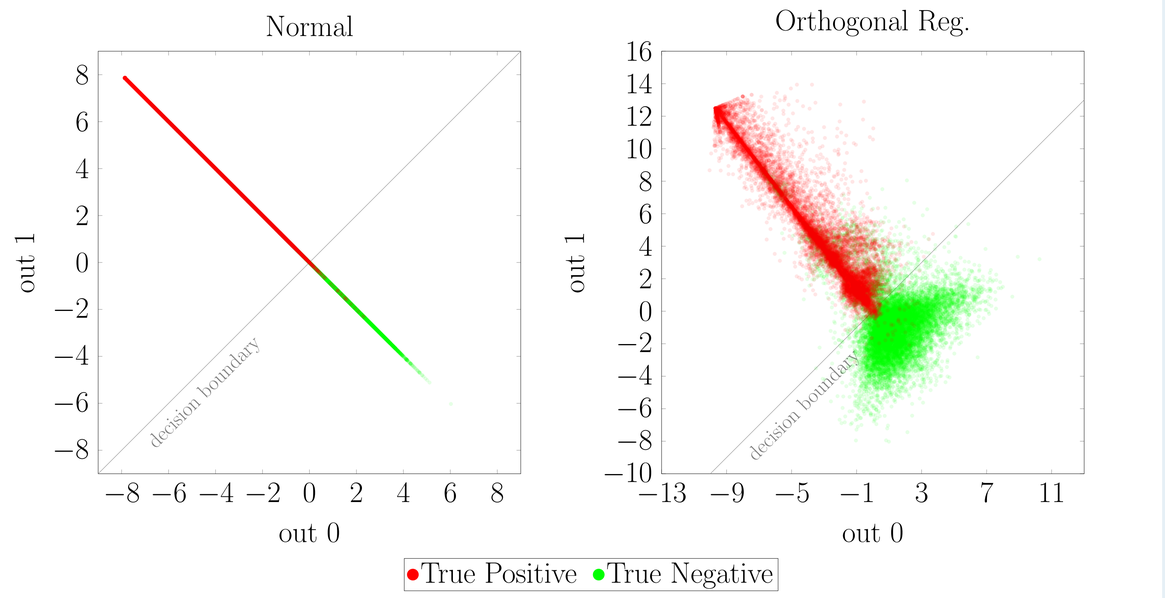
Given that the last two linear layers comprised roughly 90% of the network's size, we used our Github repo. [2] to quantize them to 8-bits, while leaving the first layer in full precision to preserve accuracy. This quantization strategy aligns with established practices.

In the final phase, we implemented the network in C++ and manually incorporated SIMD instructions for the quantized layers, as illustrated on the right side. Leveraging the ARM Cortex-M4 processor's support for 16×16 bit Multiply Accumulate operations (SMLAD), we extended the 8-bit weights with sign extension (SXTB16).

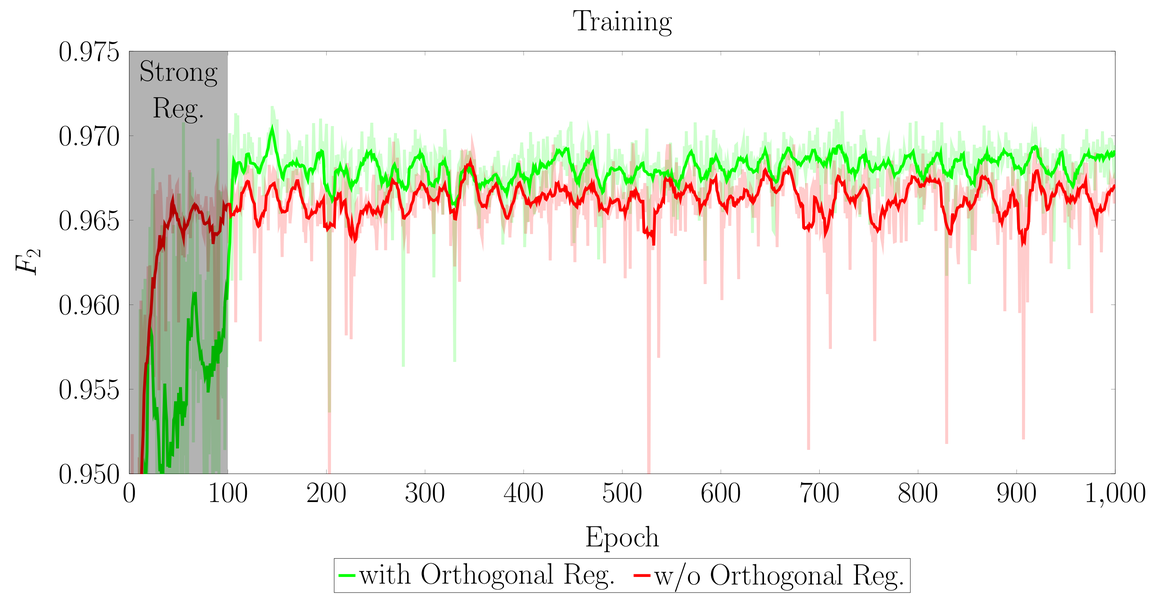
SIMD



ORTHOGONAL REG. – OUTPUT COMPARISON



ORTHOGONAL REGULARIZATION – TRAINING



ORTHOGONAL REG.

To combat overfitting and boost network generalization, we adopted orthogonal regularization [3] to diversify weight values. This technique promotes independent and non-redundant neural network weight vectors, reducing overfitting through a penalty term in the loss function.

$$A = W^T W \qquad \text{loss} = \lambda \left\| \frac{A}{\sqrt{\text{diag}(A)\text{diag}(A)^T}} - \mathbb{1} \right\|_F$$

In a two-stage process, we first "pretrained" the network with orthogonal regularization for 100 epochs and then limited its use to the final linear layer. This approach led to incremental improvements in both the F-score and generalization. Our output graph vividly demonstrates how this method enhanced the last layer's ability to distinguish output values, affirming its effectiveness.

C++ – IMPLEMENTATION

We discovered that the CMSIS library [4] in MDK5 for STM microcontrollers was outdated for our specific needs. Consequently, we had to implement the assembly function, '__SXTB16_RORn,' utilizing the 'SXTB16' assembly instruction with the "ror" option.

In addition to this, we had to address weight reordering in two critical scenarios. Firstly, we needed to reorder the weight matrices for the first linear layers as our convolution transposes the feature map. Secondly, we restructured the weights of all linear layers to facilitate efficient loading using SIMD instructions.

STATS

Flash: 12.83KB →Code: 9458, RO-data: 3446, RW-data: 236, ZI-data: 10492
Latency: 1.52ms→Conv.: 1.17ms, Linear1: 0.293ms, Linear2: 0.009ms

REFERENCES

[1] *Gatech eic lab team github repository*, `https://github.com/GATECH-EIC/TinyML-Contest-Solution`, Accessed: 2023-10-29.

[2] *Fast qat for power of two rescaler github repository*, `https://github.com/embedded-machine-learning/FastQATforPOTRescaler`, Accessed: 2023-10-29.

[3] N. Bansal, X. Chen, and Z. Wang, *Can we gain more from orthogonality regularizations in training deep cnns?* 2018. arXiv: `1810.09102 [cs.LG]`.

[4] *Cmsis-core (cortex-m)*, `https://www.keil.com/pack/doc/CMSIS/Core/html/index.html`, Accessed: 2023-10-29.