MLPerf Tiny V1.0 Greenwaves Submission Fixed Point Quantization

As in TFLite every tensor in NNTool (Greenwaves tool for TFLite/ONNX model import, conversion to Gap9 model format and optional post training quantization) is quantized following the formula:

$$r = S * (q - Z)$$

Where:

- r are the real values
- q are the quantized values
- S is the Scaling factor
- Z is the zero-point

Quantization Spec

For MLPerf submission tiny V1.0, **statistics are taken from the TFLite graph provided for each Benchmark** (i.e. TFLite quantized graph where the statistics have been calculated from the calibration dataset provided with the TFLite converter tool, TOCO). These statistics define the scaling factor and zero-point using the following formulas, the formulas change if asymmetric quantization is possible or not on Gap9 hardware / software kernels:

r_{max}, r_{min}: taken from TFLite quantization provided. For the weights r_{min} and r_{max} are both 1D tensors collected along the output channel dimension. S and Z are, in this case, also calculated as 1D tensors i.e. The quantization is specific to each output channel.

q_{max}, q_{min}: defined by the number of bits you choose for that particular tensor, i.e. int8 → -128, 127. For weights this is always clipped to be totally symmetric, i.e. int8 → -127, 127

Asymmetric:

$$S = (r_{max} - r_{min}) / (2^{nbits} - 1)$$

calculate Z from r_{min} and r_{max} and choose the one which is closer to its quantized version (Z must always accurately represent zero in the fully quantized domain)

$Z_1 = abs(q_{min} - r_{min}/S)$

$Z_2 = abs(q_{max} - r_{max}/S)$

$if\ (Z_1 - round(Z_1)) < (Z_2 - round(Z_2)$:

$Z = Z_1$

$else$:

$Z = Z_2$

Symmetric:

$q_{range} = q_{max} - q_{min}$

$mid = ceil(q_{range}/2)$

$q_{min} = -mid$

$q_{max} = q_{range} - mid - 1$

$S_1 = abs(r_{max})/abs(q_{max})$

$S_2 = abs(r_{min})/abs(q_{min})$

$S = max(S_1, S_2)$

The selection of Asymmetric vs. Symmetric depends on the type of hardware it is targeted:

	NE16	**SQ8** (Software kernels on 8 cores of cluster)
Activation	Asymmetric - PerTensor *(8 or 16 bits)*	Asymmetric ONLY if the output is connected to one node which does not require Padding, otherwise Symmetric - PerTensor *(8 bits)*
Weights	Symmetric - PerChannel *(2-8 bits)*	Symmetric - PerChannel *(8 bits)*
Biases	Quantization depends on weights and activation scaling factor only (not the actual biases real values) $S_b = S_x S_w$ *(32 bits)*	

In case of the NE16 target the weights quantization can be done below 8 bits (2, 3, 4, 5, 6, 7, 8 bits are available), the scheme will be the same, only qmin/qmax change.

Filter-based Nodes (Convolution/MatMul)

Once $S,\ Z$ are collected for every tensor the quantization can be applied and the network execution of nodes will become:

$$r_y = S_y(Y - Z_y) = \sum r_x r_w + r_b = \sum S_x(X - Z_x) S_w(W - Z_w) + S_x S_y B$$

Which can be calculated in full-integer arithmetic with:

$$Y = \frac{S_x S_w}{S_y} \sum(XW - Z_x W - Z_w X + Z_x Z_w) + B$$

Values that can be calculated ahead of time are merged in the biases:

$$B' = B + \sum(-Z_x W + Z_x Z_w)$$

$$Y = \frac{S_x S_w}{S_y} \sum(XW - Z_w X) + B'$$

The conversion factor (named MulBias in NNTool+Autotier) $\frac{S_x S_w}{S_y}$ is a 1D tensor (S_w PerChannel) represented in a fixed point fashion:

$$M * 2^N = \frac{S_x S_w}{S_y}$$

With M and N represented as unsigned 8 bit integers.

Special Nodes

The same principles are applied to non-convolutional nodes as well with some differences. In the following few special cases are treated:

- **MatAdd/Sub**: one of the two inputs must be scaled to the other input range before sum. The one which is scaled between A and B is always the one with the greater scaling factor

  $$r_y = S_y(Y - Z_y) = r_a + r_b = S_a(A - Z_a) + S_b(B - Z_b)$$

$Y = \frac{S_a}{S_y}(A - Z_a + \frac{S_b}{S_a}(B - Z_b))$ if $S_a > S_b$

$Y = \frac{S_b}{S_y}(\frac{S_a}{S_b}(A - Z_a) + B - Z_b)$ if $S_a \leq S_b$

- **Concat/Split**: The inputs of a concat or outputs of a Split must have the same scaling factor. If it is not the case NNTool finds the greater S among the input or output tensors and propagates the quantization to previous or subsequent nodes. If it cannot find a solution, i.e. it gets stuck in a circular behavior, NNTool will insert quantization nodes (quantization node: node that selects a kernel that changes the Scaling Factor/Zero point/quantization type of the input tensor).
- **Softmax**: the output of the softmax nodes are always 16 bits symmetric and the quantization of inputs and outputs is always done with power-of-2 scaling factor, i.e. $S = 2^N$. This constraint is back-propagated to the output quantization of the softmax's input nodes as well.
- **Piecewise Operators:** Piecewise operators (binary and unary) that are compiled into custom kernels absorb the quantization into the compiled code. If quantization is selected the internal operations are done in a symmetrically scaled Q15 format.