Up or Down? Adaptive Rounding for Post-Training Quantization

A. Comparison among QUBO solvers

We compared optimizing task loss Hessian using the cross-entropy method vs QUBO solver from the publicly available package *qbsolv*[1]. We chose this qbsolv QUBO solver for comparison due to its ease of use for our needs as well its free availability for any researcher to reproduce our work. Table 1 presents the comparison between the two solvers. We see that cross-entropy method significantly outperforms the *qbsolv* QUBO solver. Furthermore the *qbsolv* QUBO solver has worse performance than rounding-to-nearest. We believe this is mainly due to the reason that the API does not allow us to provide a smart initialization (as we do for cross-entropy method). The performance of random rounding choices is significantly worse, on average, when compared to the rounding choices in the neighbourhood of rounding-to-nearest. Hence this initialization can provide a significant advantage in finding a better local minimum in this large problem space. We did not conduct an extensive search for better QUBO solvers as our own implementation of the cross-entropy method provided very good results with very little tweaking and allowed us to exploit GPU and memory resources more efficiently. Furthermore the choice of QUBO solver does not impact our final method AdaRound while clearly showing the gains that we can exploit via optimized rounding.

Rounding	First layer
Nearest	52.29
Cross-entropy Method	68.62±0.17
QUBO solver (qbsolv)	41.98±3.04

Table 1. Comparison between the cross-entropy method vs *qbsolv* QUBO solver. Only the first layer of Resnet18 is quantized to 4-bits and the results are reported in terms of ImageNet validation accuracy.

B. From Taylor expansion to local loss (conv. layer)

For a convolutional layer, defined as $\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)} * \mathbf{x}^{(\ell-1)}$, we have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\ell)}_{h_1,w_1,c_1^i,c_1^o}} = \sum_{i,j} \frac{\partial \mathbf{z}^{(\ell)}_{i,j,c_1^o}}{\partial \mathbf{W}^{(\ell)}_{h_1,w_1,c_1^i,c_1^o}} \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(\ell)}_{i,j,c_1^o}} \tag{1}$$

$$= \sum_{i,j} \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(\ell)}_{i,j,c_1^o}} \cdot \mathbf{x}^{(\ell-1)}_{i+h_1,j+w_1,c_1^i}, \tag{2}$$

where h_1 and w_1 denote the spatial dimensions, c_1^i denotes input channel dimension and c_1^o denotes output channel dimension. Additionally, we have assumed appropriate zero padding of $\mathbf{x}^{(\ell-1)}$. Differentiating (2) once again (possibly w.r.t. a different weight in the same layer), we get

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{W}^{(\ell)}_{h_1,w_1,c_1^i,c_1^o} \partial \mathbf{W}^{(\ell)}_{h_2,w_2,c_2^i,c_2^o}} = \sum_{i,j} \sum_{k,m} \mathbf{x}^{(\ell-1)}_{i+h_1,j+w_1,c_1^i} \mathbf{x}^{(\ell-1)}_{k+h_2,m+w_2,c_2^i} \cdot \frac{\partial^2 \mathcal{L}}{\partial \mathbf{z}^{(\ell)}_{i,j,c_1^o} \partial \mathbf{z}^{(\ell)}_{k,m,c_2^o}}. \tag{3}$$

In order to transform the Hessian QUBO optimization problem to a local loss based per-layer optimization problem, we assume that $\nabla^2_{\mathbf{z}^{(\ell)}} \mathcal{L}$ is a diagonal matrix that is independent of the data samples $(\mathbf{x}, \mathbf{y})$, i.e.,

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{z}^{(\ell)}_{i,j,c_1^o} \partial \mathbf{z}^{(\ell)}_{k,m,c_2^o}} = \begin{cases} \mathrm{c}_{c_1^o}, & \text{if } i = k, j = m, c_1^o = c_2^o \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

[1] https://docs.ocean.dwavesys.com/projects/qbsolv/

This assumption reduces (3) to

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{W}^{(\ell)}_{h_1,w_1,c_1^i,c_1^o} \partial \mathbf{W}^{(\ell)}_{h_2,w_2,c_2^i,c_2^o}} = \begin{cases} \mathbf{c}_{c_1^o} \sum_{i,j} \mathbf{x}^{(\ell-1)}_{i+h_1,j+w_1,c_1^i} \mathbf{x}^{(\ell-1)}_{i+h_2,j+w_2,c_2^i}, & \text{if } c_1^o = c_2^o \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Under the assumptions in (4) there are no interactions between weights in the same layer that affect two different output filters ($c_1^o \neq c_2^o$). We then reformulate the Hessian QUBO optimization

$$\mathbb{E}\left[\Delta \mathbf{w}^{(\ell),T} \mathbf{H}^{(\mathbf{w}^{(\ell)})} \Delta \mathbf{w}^{(\ell)}\right] \tag{6}$$

$$\overset{(a)}{=} \mathbb{E}\left[\sum_{c^o} \mathbf{c}_{c^o} \sum_{h_1,w_1,c_1^i} \sum_{h_2,w_2,c_2^i} \sum_{i,j} \Delta \mathbf{W}^{(\ell)}_{h_1,w_1,c_1^i,c^o} \Delta \mathbf{W}^{(\ell)}_{h_2,w_2,c_2^i,c^o} \mathbf{x}^{(\ell-1)}_{i+h_1,j+w_1,c_1^i} \mathbf{x}^{(\ell-1)}_{i+h_2,j+w_2,c_2^i}\right] \tag{7}$$

$$= \mathbb{E}\left[\sum_{c^o} \mathbf{c}_{c^o} \sum_{i,j} \left(\sum_{h,w,c^i} \Delta \mathbf{W}^{(\ell)}_{h,w,c^i,c^o} \mathbf{x}^{(\ell-1)}_{i+h,j+w,c^i}\right)^2\right] \tag{8}$$

$$= \mathbb{E}\left[\sum_{c^o} \mathbf{c}_{c^o} \left\| \Delta \mathbf{W}^{(\ell)}_{:,:,:,c^o} * \mathbf{x}^{(\ell-1)} \right\|_F^2\right], \tag{9}$$

where (a) follows from the assumption in (4). Hence the Hessian optimization problem, under the assumptions in (4), is the same as MSE optimization for the output feature map. Furthermore, it breaks down to an optimization problem for each individual output channel separately (each element in the summation in (9) is independent of the other elements in the summation for optimization purposes as they involve disjoint sets of variables).

$$\underset{\Delta \mathbf{w}^{(\ell)}}{\arg\min} \quad \mathbb{E}\left[\Delta \mathbf{w}^{(\ell),T} \mathbf{H}^{(\mathbf{w}^{(\ell)})} \Delta \mathbf{w}^{(\ell)}\right] = \underset{\Delta \mathbf{W}^{(\ell)}}{\arg\min}\, \mathbb{E}\left[\left\| \Delta \mathbf{W}^{(\ell)} * \mathbf{x}^{(\ell-1)} \right\|_F^2\right] \tag{10}$$

$$= \underset{\Delta \mathbf{W}^{(\ell)}_{:,:,:,c^o}}{\arg\min}\, \mathbb{E}\left[\left\| \Delta \mathbf{W}^{(\ell)}_{:,:,:,c^o} * \mathbf{x}^{(\ell-1)} \right\|_F^2\right] \qquad \forall c^o. \tag{11}$$