Dual Precision Quantization for Efficient and Accurate Deep Neural Networks Inference

Supplementary Material

6. Group Aware Reordering (GAR) - more details

In this section, we extend the description of the GAR method. We consider the W4A16 scheme, i.e., when weights are quantized and stored in 4 bits, and computation is performed in BF16. The same rationale also applies to the W4A8 scheme discussed in this paper. We begin by revisiting the quantization and dequantization process under the GPTQ algorithm when using per-group scaling and zero-point. We then explain the rationale behind activation reordering, followed by a discussion on why unconstrained reordering introduces inference-time overhead. Finally, we present the proposed GAR method and explain how it improves accuracy without incurring any inference overhead.

6.1. Scales and Zero-Points Computation

We start by describing the computation of per-group scales and zero-points in the GPTQ quantization algorithm [12]. GPTQ operates iteratively, as described in Sec. 3.1. While GPTQ is applied independently to all weights in the column, we describe it for some row i for clarity. The algorithm begins by computing the per-group scale and zero-point, $s^{i,1}$ and $z^{i,1}$, for converting weights of the first group from BF16 to INT4. These values are computed once at the beginning of the group quantization phase, using Eq. (1).

Once $s^{i,1}$ and $z^{i,1}$ are computed, the first weight in the group, $w^{i,1}$ is quantized to INT4 and then dequantized to BF16 using Eq. (14). The quantization error with respect to the original BF16 weight is then calculated and distributed to the remaining unprocessed weights in the same row (i.e., from left to right), following Eq. (12). This process is then repeated for the second group, with a new set of parameters $s^{i,2}$, $z^{i,2}$, and so on for each subsequent group. At the end of the quantization process, each group of weights has its own scale and zero-point, as shown in Fig. 5, and the tensor is stored in its 4-bit representation, denoted by W_4.

At inference time, the stored 4-bit integer weights are loaded, and each group is dequantized to its 16-bit representation using the formula:

$$\hat{W}_{16}^{i,g} = (W_4^{i,g} - z^{i,g}) \cdot s^{i,g}. \tag{14}$$

Since each scale and zero-point corresponds to a group of *consecutive* weights in the weight tensor, the dequantization for an entire group can be efficiently performed using a single multiplication and subtraction operation, leveraging vectorized computation.

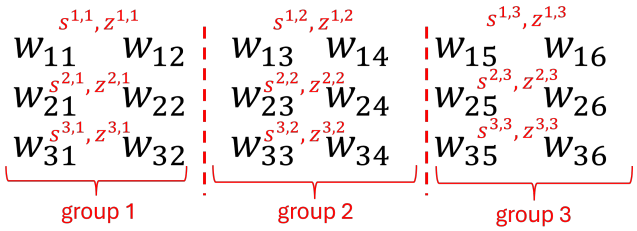

Figure 5. An illustration of per-group scale and zero-point. In this example, the weight tensor is divided into three groups, each containing two weights (i.e., a group size of 2).

6.2. Activation Reordering

We next describe the rationale behind activation reordering, its effect on the scaling factors and zero-points, and why this scheme results in inference-time overhead.

The rationale for activation reordering can be explained as follows. When using an iterative error compensation method, as described above and implemented in [12] and in this paper, it is beneficial to begin by quantizing the most "important" weights. This is because the quantization error from the weights at the beginning of the row is propagated to later weights, which reduces the error for the earlier-positioned weights. Furthermore, the weights that appear at the end of the row accumulate more error due to previous updates, resulting in higher quantization error. Thus, weights that are quantized earlier will typically incur less quantization error.

To determine which weights are the most "important", we consider the diagonal of the Hessian matrix. These diagonal elements reflect the relative significance of different input features, as they correspond to the second-order sensitivity of the loss with respect to each weight, as explained intuitively next. Recall that for a linear layer with input X, the Hessian under our objective in Eq. (5) is given by $X^T X$. In this case, the diagonal elements of the Hessian reflect the squared magnitudes of the input features. Consequently, weights associated with larger diagonal values contribute more significantly to the output. Modifying such weights has a stronger effect on the model's behavior, making them more critical to quantize accurately.

Combining the above observations that (1) earlier-quantized weights incur lower error and (2) the most important weights are those associated with the largest Hessian diagonal entries, it is advantageous to reorder the weights with respect to the Hessian diagonal before the iterative

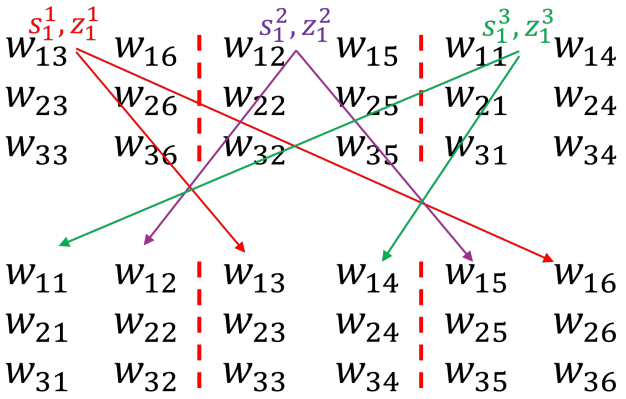

Figure 6. Computing scales and zero-points under full activation reordering scheme. As can be seen in this example in the top matrix, the third and sixth columns have been reordered to the first and second columns. Thus during inference, their scale and zero point need to be fetched from these groups, which are not consecutive in memory, and require indexing.

quantization process. Specifically, this is done by permuting the Hessian in descending order with respect to its diagonal elements, and then reordering the weight tensor accordingly. This is precisely the activation reordering technique proposed in [12]. Empirically, this technique has been shown to enhance the accuracy of quantized models and is widely adopted in practice.

However, despite its accuracy benefits, the reordering technique introduces additional overhead during inference (i.e., increased latency) as discussed next.

When applying the activation reordering technique, we first permute the weight tensor and then run the iterative algorithm as described at the beginning of this section. Specifically, the scale factors and zero-points are computed for groups of the *permuted* weight tensor. At the end of the process, the weight tensor is re-permuted to its original order to ensure correct multiplication during inference.

However, after re-permutation, the scale factors and zero-points are no longer aligned with the weight groups. That is, each weight in the tensor may now have a different scale and zero-point, as illustrated in Fig. 6. As a result, the dequantization in Eq. (14) can no longer be efficiently applied using a single scalar subtraction and multiplication per group. Instead, each weight must be individually processed with its corresponding zero-point and scale factor, increasing the number of operations required during inference and thereby increasing latency.

6.3. Group Aware Reordering

The proposed GAR method addresses the issue of inference overhead while still allowing the weight tensor to be reordered based on weight importance, subject to certain constraints. Specifically, as described in Sec. 3.2, GAR allows permutations in two ways: (i) within a group, and/or (ii) by rearranging entire groups, as illustrated in the top panel of Fig. 7.

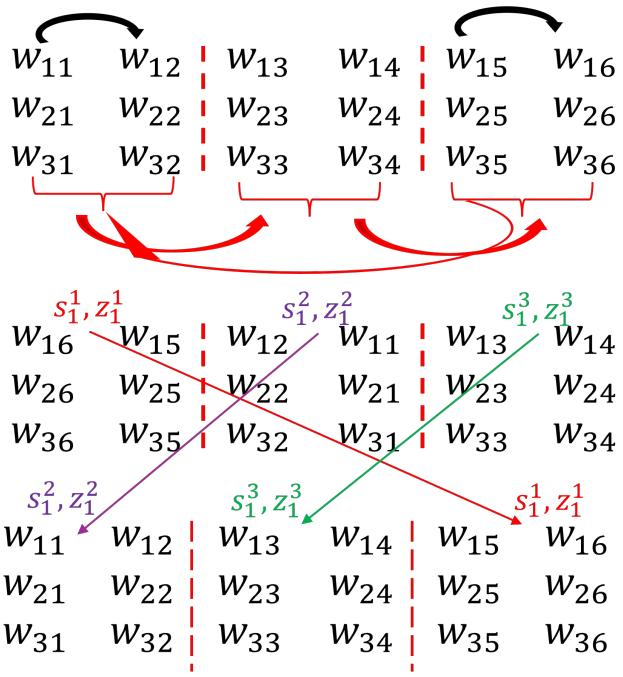

Figure 7. Computing scales and zero-points under GAR method.

This constrained permutation ensures that, after computing the scales and zero-points for the permuted tensor, each original group of weights shares the same scale and zero-point, as shown in the middle panel of Fig. 7. As a result, when the weight tensor is re-permuted back to its original order, the associated scales and zero-points can be re-permuted accordingly (see the bottom panel of Fig. 7). This guarantees that each group of weights retains its own scale and zero-point, allowing Eq. (14) to be efficiently implemented during inference, as in the original scheme without activation reordering.

Although our method imposes constraints on the activation order, prior studies have shown that most layers are dominated by a small number of highly important output activations [24, 41]. Therefore, even though our scheme only allows a partial ordering of activations, it effectively preserves the most significant weights. As a result, we expect only a small accuracy gap compared to the full reordering scheme. This hypothesis is supported by our results in Table 4. Under both W4A16 and W4A8 quantization settings, the GAR method delivers a significant accuracy improvement over the baseline without reordering, while incurring only a minor accuracy drop relative to the full reordering approach.

Finally, we note that GAR requires a ranking criterion

to determine which group should be quantized first. In our experiments, we ranked groups based on the maximum value of the Hessian diagonal within each group, that is, we compared the maximum diagonal entry of each group of weights. Alternative criteria are possible, for example, ranking groups by the average of the top 10% largest elements before sorting. We leave the exploration of other ranking strategies for future work.