A Winograd transformation matrices

Depending on the particular choice of Winograd domain (i.e., polynomial domain), transformation matrices A, B, and G in the Winograd algorithm can be different. In the paper, we present that the most popular interpolation points for F(2,3) are $[0, +1, -1]$ and then these transformation matrices can be constructed as follows:

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}, \quad B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \quad G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

For F(4,3) and F(6,3), we choose the same transformation matrices as BQW [1]. For F(4,3), the Winograd transformation matrices are as follows:

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 2 & -2 & 0 \\ 0 & 1 & 1 & 4 & 4 & 0 \\ 0 & 1 & -1 & 8 & -8 & 1 \end{bmatrix}, \tag{2}$$

$$B^T = \begin{bmatrix} 4 & 0 & -5 & 0 & 1 & 0 \\ 0 & -4 & -4 & 1 & 1 & 0 \\ 0 & 4 & -4 & -1 & 1 & 0 \\ 0 & -2 & -1 & 2 & 1 & 0 \\ 0 & 2 & -1 & -2 & 1 & 0 \\ 0 & 4 & 0 & -5 & 0 & 1 \end{bmatrix}, \tag{3}$$

$$G = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} \\ -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6} \\ \frac{1}{24} & \frac{1}{12} & -\frac{1}{6} \\ \frac{1}{24} & -\frac{1}{12} & -\frac{1}{6} \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

For F(6,3), the Winograd transformation matrices are as follows:

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 2 & -2 & 0 \\ 0 & 1 & 4 & 4 & 0 & \\ 0 & 1 & -1 & 8 & -8 & 1 \end{bmatrix}, \tag{5}$$

$$B^T = \begin{bmatrix} 1 & 0 & -\frac{21}{4} & 0 & \frac{21}{4} & 0 & -1 & 0 \\ 0 & 1 & 1 & -\frac{17}{4} & -\frac{17}{4} & 1 & 1 & 0 \\ 0 & -1 & 1 & \frac{17}{4} & -\frac{17}{4} & -1 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{4} & -\frac{5}{2} & -\frac{5}{4} & 2 & 1 & 0 \\ 0 & -\frac{1}{2} & \frac{1}{4} & \frac{5}{2} & -\frac{5}{4} & -2 & 1 & 0 \\ 0 & 2 & 4 & -\frac{5}{2} & -5 & \frac{1}{2} & 1 & 0 \\ 0 & -2 & 4 & \frac{5}{2} & -5 & -\frac{1}{2} & 1 & 0 \\ 0 & -1 & 0 & \frac{21}{4} & 0 & -\frac{21}{4} & 0 & 1 \end{bmatrix}, \tag{6}$$

$$G = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{2}{9} & -\frac{2}{9} & -\frac{2}{9} \\ -\frac{2}{9} & \frac{2}{9} & -\frac{2}{9} \\ \frac{1}{90} & \frac{1}{45} & -\frac{2}{45} \\ \frac{1}{90} & -\frac{1}{45} & \frac{2}{45} \\ \frac{32}{45} & \frac{16}{45} & \frac{8}{45} \\ \frac{32}{45} & -\frac{16}{45} & \frac{8}{45} \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

B Derivatives of transformation matrices

In the paper, in order to align these transformation procedures after quantization, we propose to adjust transformation matrices via an optimization procedure as follows:

$$\underset{A,B,G}{argmin} \quad \mathbb{E}_{X\sim\mathcal{D}}\left[\Sigma_f^{C_o}||A^T(\Sigma_c^{C_i}Q(B^TX_cB)\odot Q(GW_{f,c}G^T))A-Y_f||^2\right] \tag{8}$$

By using the straight-through estimator [2] to approximate the gradient through the round function as a pass-through operation, we can obtain the derivatives of A, B and G. In this paper, we directly present the derivative of B. Here, a more comprehensive derivation is provided as follows:

$$\frac{\partial\mathcal{L}}{\partial B_{ij}} = \Sigma_f^{C_o}\ \text{tr}\left\{\frac{\partial\mathcal{L}}{\partial O_f^T}\cdot\frac{\partial O_f}{\partial B_{ij}}\right\} \tag{9}$$

$$= \Sigma_f^{C_o}\ \text{tr}\left\{\frac{\partial\mathcal{L}}{\partial O_f^T}\cdot\left[\Sigma_c^{C_i}(\delta_{ji}X_cB)\odot Q(V_{f,c})+(B^TX_c\delta_{i,j})\odot Q(V_{f,c})\right]\right\} \tag{10}$$

$$= \Sigma_f^{C_o}\ \Sigma_c^{C_i}\ \text{tr}\left\{\frac{\partial\mathcal{L}}{\partial O_f^T}\cdot[(\delta_{ji}X_cB)\odot Q(V_{f,c})]+\frac{\partial\mathcal{L}}{\partial O_f^T}\cdot\left[(B^TX_c\delta_{ij})\odot Q(V_{f,c})\right]\right\} \tag{11}$$

$$= \Sigma_f^{C_o}\ \Sigma_c^{C_i}\ \text{tr}\left\{(\delta_{ji}X_cB)^T\cdot\left[\frac{\partial\mathcal{L}}{\partial O_f}\odot Q(V_{f,c})\right]+(B^TX_c\delta_{ij})^T\cdot\left[\frac{\partial\mathcal{L}}{\partial O_f}\odot Q(V_{f,c})\right]\right\} \tag{12}$$

$$= \Sigma_f^{C_o}\ \Sigma_c^{C_i}\ \left[X_cB\cdot(\frac{\partial L}{\partial O_f}\odot Q(V_{f,c}))^T\right]_{ij}+\left[X_c^TB\cdot(\frac{\partial L}{\partial O_f}\odot Q(V_{f,c}))\right]_{ij} \tag{13}$$

We have obtained the derivative of B_{ij}, and now we can provide the expression for the derivative of B:

$$\frac{\partial\mathcal{L}}{\partial B} = \Sigma_f^{C_o}\ \Sigma_c^{C_i}\ X_cB(\frac{\partial L}{\partial O_f}\odot Q(V_{f,c}))^T+X_c^TB(\frac{\partial L}{\partial O_f}\odot Q(V_{f,c})) \tag{14}$$

The derivatives of A, G and O_f can be computed in a similar manner:

$$\frac{\partial\mathcal{L}}{\partial A} = \Sigma_f^{C_o}\ O_f^TA(A^TO_fA-Y_f)+O_fA(A^TO_fA-Y_f)^T \tag{15}$$

$$\frac{\partial\mathcal{L}}{\partial G} = \Sigma_f^{C_o}\ \Sigma_c^{C_i}\ (\frac{\partial L}{\partial O_f}\odot Q(U_c))GW_{f,c}^T+(\frac{\partial L}{\partial O_f}\odot Q(U_c))^TGW_{f,c} \tag{16}$$

$$\frac{\partial\mathcal{L}}{\partial O_f} = 2A(A^TO_fA-Y)A^T \tag{17}$$

C Optimal quantization scale for Guassion varibles

In Theorem 1, in order to demonstrate that the optimal per-pixel scale S can be factorized into vectors, we rely on the conclusion that the optimal scale s^* to minimize the mean-square error of quantization of Gaussian variables $z\sim\mathcal{N}(0,\sigma^2)$ is proportional to σ, i.e., $s^*=K\sigma$, where K is a constant. Here, we will provide a proof of it.

Theorem C.1. *Assuming $z\sim\mathcal{N}(0,\sigma^2)$, the optimal scale s^* to minimize the mean-square error of quantization of z is proportional to the standard deviation σ, i.e., $s^*=K\sigma$, where K is a constant.*

Proof. Because $z\sim\mathcal{N}(0,\sigma^2)$, z can be reparameterized as $z=\sigma\cdot u$, where $u\sim\mathcal{N}(0,1)$.

$$
\begin{aligned}
\mathbf{E}\left[(Q(z)-z)^2\right] &= \int_{-\infty}^{\infty} p_z(z)(Q(z)-z)^2 dz && (18)\\
&= \int_{-\infty}^{\infty} p_u(u)(Q(\sigma u)-\sigma u)^2 du && (19)\\
&= \int_{-\infty}^{\infty} p_u(u)(clip(\left\lfloor \frac{\sigma u}{s} \right\rceil, -q_{min}, q_{max}) \cdot s - \sigma u)^2 du && (20)\\
&= \sigma^2 \int_{-\infty}^{\infty} p_u(u)(clip(\left\lfloor \frac{u}{s/\sigma} \right\rceil, -q_{min}, q_{max}) \cdot \frac{s}{\sigma} - u)^2 du && (21)\\
&= \sigma^2 h(\frac{s}{\sigma}) && (22)
\end{aligned}
$$

Eq. (18) can be treated as a function of s/σ when solving for s with σ as a known value. Assuming K minimizes function $h(x)$, i.e., $K = \underset{x}{argmin}\ h(x)$, we have:

$$
s^* = \underset{s}{argmin}\ \mathbf{E}\left[(Q(z)-z)^2\right] = \underset{s}{argmin}\ \sigma^2 h(\frac{s}{\sigma}) = K \cdot \sigma \tag{23}
$$

□

D Experiments on other architectures

In Section 5, we compare our methods to previous work BQW[1] on the ResNet model family with comprehensive experiment settings, including various bit widths, tile sizes, and datasets. Here, we present a similar analysis for two other popular architectures VGG and Squeezenet using the Cifar-10 dataset. The results are shown in Table 1 and Table 2. These results align with our analysis in Section 5. Our PTQ-Aware Winograd (PAW) method outperforms the strong baseline introduced in Section 5 and our FSQ method is well-compatible with PAW.

Table 1: PTQ results of VGG11 on CIFAR-10.

Model	Tile	Bits	Partial Quantization		Full Quantization	
			Baseline	**PAW**	FSQ	**FSQ+PAW**
VGG-11 (92.02%)	F(4,3)	6	89.13	91.56	86.59	91.55
		8	92.02	92.28	90.82	91.83
	F(6,3)	6	75.10	89.94	68.98	90.34
		8	91.27	91.88	88.44	91.63

Table 2: PTQ results of SqueezeNet on CIFAR-10.

Model	Tile	Bits	Partial Quantization		Full Quantization	
			Baseline	**PAW**	FSQ	**FSQ+PAW**
SqueezeNet (92.62%)	F(4,3)	6	89.69	91.98	88.66	91.78
		8	92.61	92.68	92.01	92.80
	F(6,3)	6	80.50	90.67	76.48	91.26
		8	92.37	92.61	90.54	92.42

References

[1] Vladimir Chikin and Vladimir Kryzhanovskiy. Channel balancing for accurate quantization of winograd convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 12497–12506. IEEE, 2022.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.