Gradient Computation in Winograd Domain

1 Winograd Convolution

Assume we need to perform convolution on a $m \times m$ input tile $\boldsymbol{I}$ with a $n \times n$ weight kernel $\boldsymbol{W}$. The output tile $\boldsymbol{O}$ will have a size of $(m-n+1) \times (m-n+1)$.

With the Winograd transformation, this convolution operation can be performed by

$$\boldsymbol{O} = \boldsymbol{A}^\top \left[(\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top) \odot (\boldsymbol{B}^\top \boldsymbol{I} \boldsymbol{B}) \right] \boldsymbol{A} \tag{1}$$

where $\boldsymbol{A}$, $\boldsymbol{G}$ and $\boldsymbol{B}$ are matrices used for the Winograd transformation. $\odot$ is element-wise matrix multiplication.

As an example, for $m = 4$ and $n = 3$, we have

$$\boldsymbol{B}^\top = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{A}^\top = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix} \tag{2}$$

In this note, we only discuss the situation with an one-channel input tile and an one-channel weight kernel. It can be extended to a normal convolutional layer.

2 Winograd Layer

Deep Neural Networks (DNNs) have much internal redundancy. We can remove weight parameters from DNN models to reduce computation and model sizes. In this case, the weight kernels, e.g., $\boldsymbol{W}$, will become sparse.

In the Winograd convolution, the majority of the floating-point multiplication is performed in the element-wise multiplication. Therefore, to fully utilize the redundancy in DNN models, we need higher sparsity in $\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top$. However, the sparsity in $\boldsymbol{W}$ cannot be directly transformed into the sparsity in $\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top$.

Assume $\boldsymbol{Q} = \boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top$. Here $\boldsymbol{Q}$ has a size of $m \times m$ and $\boldsymbol{W}$ has a size of $n \times n$. Since we will have $m > n$, the mapping from $\boldsymbol{W}$ to $\boldsymbol{Q}$ is non-invertible.

One solution to increase the sparsity in $\boldsymbol{Q}$ is to directly consider $\boldsymbol{Q}$ as the parameters for a convolutional layer. Convolutional layers with weight parameters in the Winograd domain are named as Winograd layers. Then we can remove unimportant parameters directly from $\boldsymbol{Q}$.

The computation performed by a Winograd layer is

$$\boldsymbol{O} = \boldsymbol{A}^\top \left[\boldsymbol{Q} \odot (\boldsymbol{B}^\top \boldsymbol{I} \boldsymbol{B}) \right] \boldsymbol{A} \tag{3}$$

3 Explosion of Update Steps

With Winograd layers instead of original convolutional layers, we can still use stochastic gradient descent (SGD) algorithm to train the network. However, we found it requires a small learning rate (LR) and, therefore, the training converges extremely slow.

This is due to the explosion of update steps caused by the linear transformation $\boldsymbol{GWG}^\top$. Here we use a simple example to show the explosion problem of update steps.

Assume we have a parameter x and its gradient with respect to the loss (L) is

$$\frac{\partial L}{\partial x} = 1 \tag{4}$$

with $LR = 0.1$, x will be updated by

$$x := x + \Delta x \tag{5}$$

$$\Delta x = -LR \cdot \frac{\partial L}{\partial x} = -0.1 \tag{6}$$

Assume we use a linear transformation

$$y = 0.1x \tag{7}$$

to map x to y. If we are training parameter x, then we need to update y by

$$y := y + \Delta y \tag{8}$$

$$\Delta y = 0.1 \cdot \Delta x = -0.01 \tag{9}$$

However, assuming we are now directly performing training on y, the gradient of y with respect to the loss (L) is

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial x} \cdot \frac{\partial x}{\partial y} = 10 \tag{10}$$

In this case, y will be updated by

$$y := y + \Delta y' \tag{11}$$

$$\Delta y' = -LR \cdot \frac{\partial L}{\partial y} = -1 \tag{12}$$

We will have $\Delta y' = 100 \cdot \Delta y$. It means, if we directly train the parameter y instead of x, the updates on y will be extended by 100x, which is an explosion in update steps.

The linear transformation $\boldsymbol{Q} = \boldsymbol{GWG}^\top$ will have a similar effect. The direct training of $\boldsymbol{Q}$ will suffer from the explosion of the update steps, which requires LR to be extremely small. Also, it is difficult to find appropriate LR for each parameter in $\boldsymbol{Q}$.

4 Gradients in Winograd Domain

For parameter y, one solution for the explosion of update steps is to calculate Δy by

$$\Delta y = \Delta(0.1x) = 0.1 \cdot \Delta x = 0.1 \cdot (-LR) \cdot \frac{\partial L}{\partial x} \tag{13}$$

In this case, the effective gradient of y, $(\frac{\partial L}{\partial y})^*$, can be calculated by

$$(\frac{\partial L}{\partial y})^* = \frac{\Delta y}{-LR} = \frac{0.1 \cdot \Delta x}{-LR} = \frac{0.1 \cdot (-LR) \cdot \frac{\partial L}{\partial x}}{-LR} = 0.1 \cdot \frac{\partial L}{\partial x} \tag{14}$$

Similarly, for the Winograd layer parameters $\boldsymbol{Q}$, we need to have

$$\Delta \boldsymbol{Q} = \Delta(\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top) = \boldsymbol{G}(\Delta \boldsymbol{W})\boldsymbol{G}^\top \tag{15}$$

Therefore, we can calculate the effective gradients for $\boldsymbol{Q}$ by

$$(\frac{\partial L}{\partial \boldsymbol{Q}})^* = \frac{\Delta \boldsymbol{Q}}{-LR} = \boldsymbol{G}\frac{\Delta \boldsymbol{W}}{-LR}\boldsymbol{G}^\top = \boldsymbol{G}\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{G}^\top \tag{16}$$

Since $\boldsymbol{Q} = \boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top$, we will have

$$\frac{\partial L}{\partial \boldsymbol{W}} = \boldsymbol{G}^\top \frac{\partial L}{\partial \boldsymbol{Q}}\boldsymbol{G} \tag{17}$$

where $\frac{\partial L}{\partial \boldsymbol{Q}}$ is the original gradients of $\boldsymbol{Q}$ calculated through back-propagation.

Therefore, based on Equation 16 and 17, we will have

$$(\frac{\partial L}{\partial \boldsymbol{Q}})^* = \boldsymbol{G}\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{G}^\top = \boldsymbol{G}(\boldsymbol{G}^\top \frac{\partial L}{\partial \boldsymbol{Q}}\boldsymbol{G})\boldsymbol{G}^\top = (\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) \tag{18}$$

Then $\boldsymbol{Q}$ will be updated by

$$\boldsymbol{Q} := \boldsymbol{Q} - LR \cdot (\frac{\partial L}{\partial \boldsymbol{Q}})^* \tag{19}$$

5 Regularization

In this section, we will discuss how to perform regularization on $\boldsymbol{Q}$.

The loss function of DNN models usually includes two parts: the divergence between the DNN output and the ground-truth output, L^D, and the regularization term, L^R. The regularization term is used to prevent over-fitting. In this case, we have

$$L = L^D + L^R \tag{20}$$

Assume we are using the L2 regularization, then for the original weight kernel $\boldsymbol{W}$,

$$L^R = \frac{\lambda}{2}||\boldsymbol{W}||_2 \tag{21}$$

where λ is the regularization strength which is a scalar value.

Assume with the Winograd layers, we still use the same loss function as for the original convolutional layers. Then for Equation 18, we have

$$\begin{aligned}(\frac{\partial L}{\partial \boldsymbol{Q}})^* &= (\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) \\ &= (\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L^D}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) + (\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L^R}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top)\end{aligned} \tag{22}$$

$\frac{\partial L^D}{\partial \boldsymbol{Q}}$ can be calculated with the back-propagation. But

$$\frac{\partial L^R}{\partial \boldsymbol{Q}} = \frac{\partial}{\partial \boldsymbol{Q}}(\frac{\lambda}{2}||\boldsymbol{W}||_2) \tag{23}$$

is intractable since the linear transformation, $\boldsymbol{Q} = \boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top$, is non-invertible.

However, we can simplify the computation since

$$\begin{aligned}(\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L^R}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) &= \boldsymbol{G}(\boldsymbol{G}^\top\frac{\partial L^R}{\partial \boldsymbol{Q}}\boldsymbol{G})\boldsymbol{G}^\top \\ &= \boldsymbol{G}\frac{\partial L^R}{\partial \boldsymbol{W}}\boldsymbol{G}^\top \\ &= \boldsymbol{G}(\lambda \cdot \boldsymbol{W})\boldsymbol{G}^\top \\ &= \lambda \cdot (\boldsymbol{G}\boldsymbol{W}\boldsymbol{G}^\top) \\ &= \lambda \cdot \boldsymbol{Q}\end{aligned} \tag{24}$$

Then, Equation 22 can be simplified to

$$\begin{aligned}(\frac{\partial L}{\partial \boldsymbol{Q}})^* &= (\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L^D}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) + (\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L^R}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) \\ &= (\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L^D}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) + \lambda \cdot \boldsymbol{Q}\end{aligned} \tag{25}$$

The term, $\lambda \cdot \boldsymbol{Q}$, means, effectively, we are directly applying an L2 regularization

$$(L^R)^* = \frac{\lambda}{2}||\boldsymbol{Q}||_2 \tag{26}$$

on $\boldsymbol{Q}$, the parameters in the Winograd domain.

In conclusion, the update we need to perform in each step of SGD is

$$\boldsymbol{Q} := \boldsymbol{Q} - LR \cdot [(\boldsymbol{G}\boldsymbol{G}^\top)\frac{\partial L^D}{\partial \boldsymbol{Q}}(\boldsymbol{G}\boldsymbol{G}^\top) + \lambda \cdot \boldsymbol{Q}] \tag{27}$$

6 Evaluation and Results

We perform the training on two networks on the MNIST dataset: LeNet-5 and LeNet-5-3x3. LeNet-5 includes 2 convolution layers with the kernel size of 5 and 2 fully-connected layers. LeNet-5-3x3 is designed by ourselves. It includes 4 convolution layers with the kernel size of 3. The size of the output tiles in the Winograd convolution is fixed to 4×4, which means $m - n + 1 = 4$. All networks are trained from scratch. Table 1 shows the results.

Table 1: Training Accuracy

	Accuracy (%)	
	Original	Winograd
LeNet-5	99.44%	99.44%
LeNet-5-3x3	99.26%	99.24%