CENG5030 Lab 02: Winograd

Bei Yu
Department of Computer Science & Engineering
Chinese University of Hong Kong
byu@cse.cuhk.edu.hk

November 2, 2023

Strassen Algorithm

Recap on Strassen Algorithm, to calculate matrix multiplication $\boldsymbol{M} \times \boldsymbol{N}$, we first split the matrices into:

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, \qquad N = \begin{pmatrix} E & F \\ G & H \end{pmatrix}.$$

Then, we calculate the intermediate matrices:

$$\begin{aligned}
S_1 &= (B - D)(G + H) \\
S_2 &= (A + D)(E + H) \\
S_3 &= (A - C)(E + F) \\
S_4 &= (A + B)H \\
S_5 &= A(F - H) \\
S_6 &= D(G - E) \\
S_7 &= (C + D)E.
\end{aligned}$$

Strassen Algorithm

Strassen Algorithm in a divide-and-conquer style. Split into submatrix:

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, \qquad N = \begin{pmatrix} E & F \\ G & H \end{pmatrix}.$$

```
int Divide  = (int)(N/2);

int A11[SIZE][SIZE], A12[SIZE][SIZE], A21[SIZE][SIZE], A22[SIZE][SIZE];
int B11[SIZE][SIZE], B12[SIZE][SIZE], B21[SIZE][SIZE], B22[SIZE][SIZE];
```

```
//dividing the matrices in 4 sub-matrices:
for (i = 0; i < Divide; i++)
{
    for (j = 0; j < Divide; j++)
    {
        A11[i][j] = A[i][j];
        A12[i][j] = A[i][j + Divide];
        A21[i][j] = A[i + Divide][j];
        A22[i][j] = A[i + Divide][j + Divide];

        B11[i][j] = B[i][j];
        B12[i][j] = B[i][j + Divide];
        B21[i][j] = B[i + Divide][j];
        B22[i][j] = B[i + Divide][j + Divide];
    }
}
```

Strassen Algorithm

Then we calculate the intermediate matrices:

$$
\begin{aligned}
S_1 &= (B-D)(G+H)\\
S_2 &= (A+D)(E+H)\\
S_3 &= (A-C)(E+F)\\
S_4 &= (A+B)H\\
S_5 &= A(F-H)\\
S_6 &= D(G-E)\\
S_7 &= (C+D)E.
\end{aligned}
$$

Let's look at an example of S_2:

```
MatrixAdd(A11, A22, AResult, Divide);    // a11 + a22
MatrixAdd(B11, B22, BResult, Divide);    // b11 + b22
StrassenAlgorithm(AResult, BResult, P1, Divide);  // p1 = (a11+a22) * (b11+b22)
```

Note that Strassen is recursive:

Algorithm 3 Strassen's Algorithm

function STRASSEN(M,N)
  if M is 1×1 **then**
    return $M_{11}N_{11}$
  end if
  Let $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ and $N = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$
  Set $S_1 = \text{STRASSEN}(B - D, G + H)$
  $\boxed{\text{Set } S_2 = \text{STRASSEN}(A + D, E + H)}$
  Set $S_3 = \text{STRASSEN}(A - C, E + F)$
  Set $S_4 = \text{STRASSEN}(A + B, H)$
  Set $S_5 = \text{STRASSEN}(A, F - H)$
  Set $S_6 = \text{STRASSEN}(D, G - E)$
  Set $S_7 = \text{STRASSEN}(C + D, E)$
  return $\begin{pmatrix} S_1 + S_2 - S_4 + S_6 & S_4 - S_5 \\ S_6 + S_7 & S_2 - S_3 + S_5 - S_7 \end{pmatrix}$
end function

```
MatrixAdd(A11, A22, AResult, Divide);    // a11 + a22
MatrixAdd(B11, B22, BResult, Divide);    // b11 + b22
StrassenAlgorithm(AResult, BResult, P1, Divide);  // p1 = (a11+a22) * (b11+b22)
```

Winograd

Now we move on Winograd Algorithm for convolution. Recap on the 2-D Winograd with 4×4 input feature D and 3×3 kernel K:

$$D = \begin{bmatrix} d_{00} & d_{01} & d_{02} & d_{03} \\ d_{10} & d_{11} & d_{12} & d_{13} \\ d_{20} & d_{21} & d_{22} & d_{23} \\ d_{30} & d_{31} & d_{32} & d_{33} \end{bmatrix}$$

$$K = \begin{bmatrix} k_{00} & k_{01} & k_{02} \\ k_{10} & k_{11} & k_{12} \\ k_{20} & k_{21} & k_{22} \end{bmatrix}$$

Winograd

After `im2col` operation, we can split

$$\left[\begin{array}{ccc|ccc|ccc} d_{00} & d_{01} & d_{02} & d_{10} & d_{11} & d_{12} & d_{20} & d_{21} & d_{22} \\ d_{01} & d_{02} & d_{03} & d_{11} & d_{12} & d_{13} & d_{21} & d_{22} & d_{23} \\ \hline d_{10} & d_{11} & d_{12} & d_{20} & d_{21} & d_{22} & d_{30} & d_{31} & d_{32} \\ d_{11} & d_{12} & d_{13} & d_{21} & d_{22} & d_{23} & d_{31} & d_{32} & d_{33} \end{array}\right] \left[\begin{array}{c} k_{00} \\ k_{01} \\ k_{02} \\ \hline k_{10} \\ k_{11} \\ k_{12} \\ \hline k_{20} \\ k_{21} \\ k_{22} \end{array}\right] = \left[\begin{array}{c} r_{00} \\ r_{01} \\ \hline r_{10} \\ r_{11} \end{array}\right]$$

$$\begin{bmatrix} D_{00} & D_{10} & D_{20} \\ D_{10} & D_{20} & D_{30} \end{bmatrix} \begin{bmatrix} \vec{k_0} \\ \vec{k_1} \\ \vec{k_2} \end{bmatrix} = \begin{bmatrix} \vec{r_0} \\ \vec{r_1} \end{bmatrix} = \begin{bmatrix} M_0 + M_1 + M_2 \\ M_1 - M_2 - M_3 \end{bmatrix}$$

$$\begin{aligned} M_0 &= (D_{00} - D_{20})\vec{k_0} \\ M_1 &= (D_{10} + D_{20})\frac{\vec{k_0} + \vec{k_1} + \vec{k_2}}{2} \\ M_2 &= (D_{20} - D_{10})\frac{\vec{k_0} - \vec{k_1} + \vec{k_2}}{2} \\ M_3 &= (D_{10} - D_{30})\vec{k_2} \end{aligned}$$

The formuation of this Winograd is:

$$Y = A^T[[GgG^T] \odot [B^T dB]]A \tag{1}$$

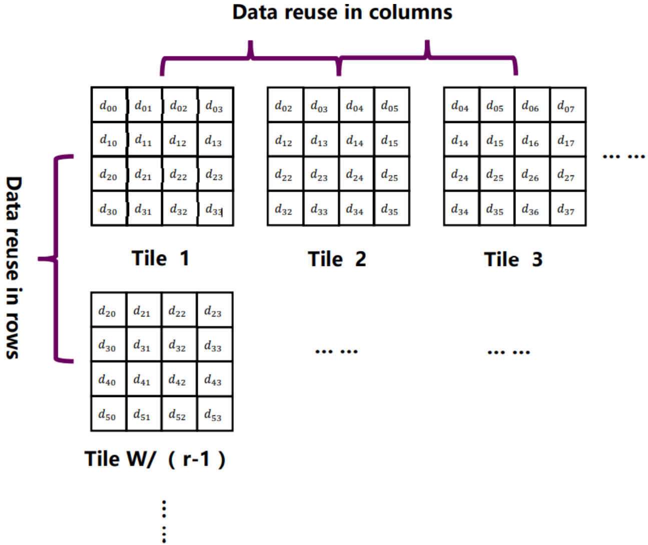
where the transforms for $F(3 \times 3, 2 \times 2)$ are:

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

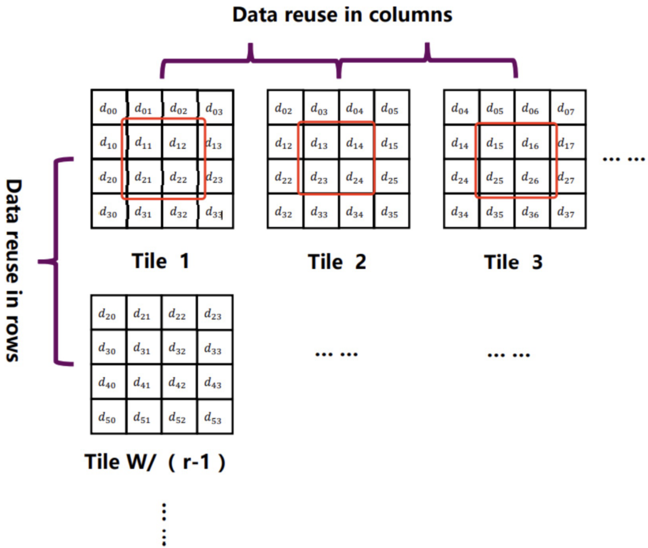
$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Winograd

Note that a feature map is usually larger than 4×4, we can handle it by splitting into 4×4 tiles with overlapping.



The overlapping step is set as 2 for 3×3 kernel size so the output of each tile can merge the original size.



THANK YOU!