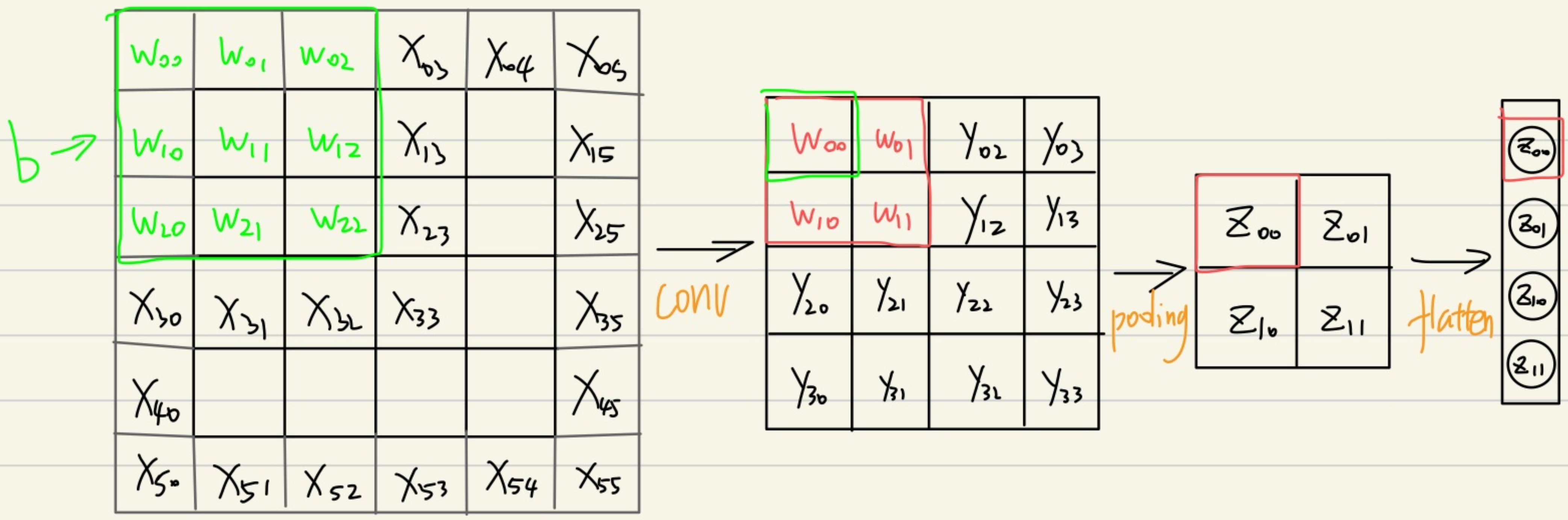


Convolution $\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a^2+b^2+c^2+d^2$ Hadamard product $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \circ \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a^2 & b^2 \\ c^2 & d^2 \end{bmatrix}$

Known: gradient $\left\{ \underline{\underline{z}} = \begin{bmatrix} z_{00} & z_{01} \\ z_{10} & z_{11} \end{bmatrix} \right\}$ $w_{00} = w_{01} = w_{10} = w_{11} = \frac{1}{4}$

Ave Pooling:

1

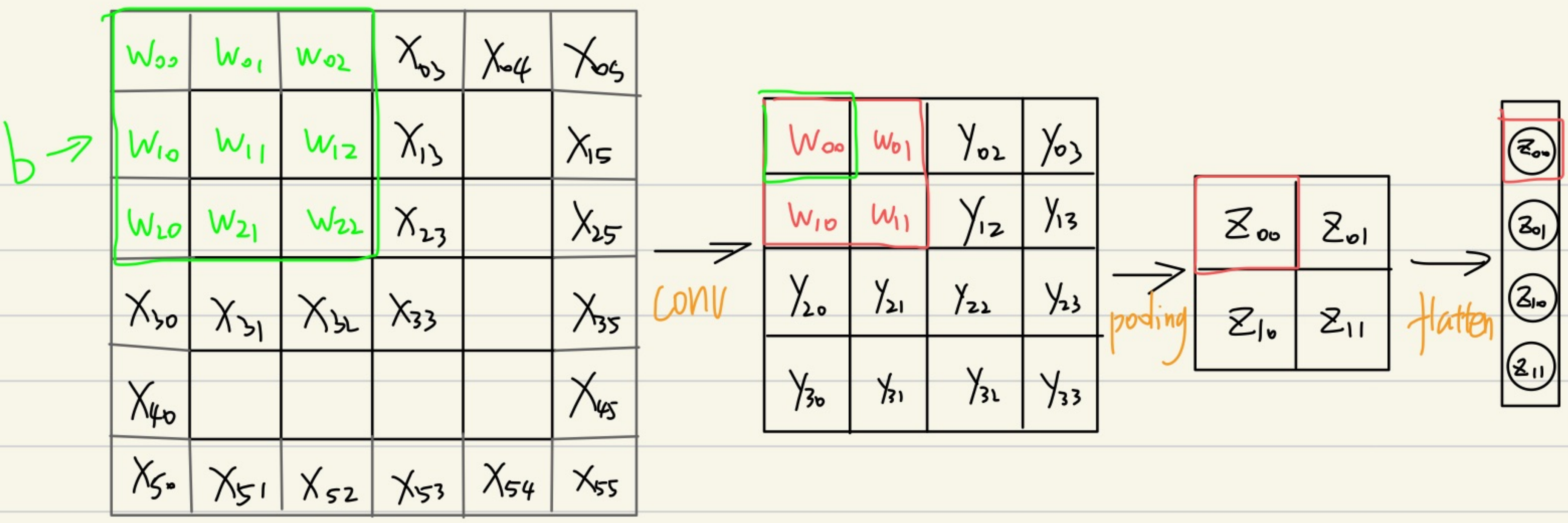
$$z_{ij} = \sum_{m=0}^{1} \sum_{n=0}^{1} y_{(2i+m)(2j+n)} \cdot w_{mn} = \frac{1}{4} \sum_{m=0}^{1} \sum_{n=0}^{1} y_{(2i+m)(2j+n)}$$

$$\text{gradient } y_{ij} = \text{gradient } z_{(i//2)(j//2)} \cdot \nabla_{y_{ij}} z_{(i//2)(j//2)}$$

$$= \text{gradient } z_{(i//2)(j//2)} \cdot w_{(i\%2)(j\%2)}$$

$$= \frac{1}{4} \text{gradient } z_{(i//2)(j//2)}$$

$$\text{gradiant } \underline{\underline{y}} = \frac{1}{4} \cdot \left[\begin{array}{cc|cc} \text{grad } z_{00} & \text{grad } z_{00} & \text{grad } z_{01} & \text{grad } z_{01} \\ \text{grad } z_{00} & \text{grad } z_{00} & \text{grad } z_{01} & \text{grad } z_{01} \\ \hline \text{grad } z_{10} & \text{grad } z_{10} & \text{grad } z_{11} & \text{grad } z_{11} \\ \text{grad } z_{10} & \text{grad } z_{10} & \text{grad } z_{11} & \text{grad } z_{11} \end{array}\right]$$



② convolution [stride=1, kernel_size=3]

$$Y_{ij} = X_{ij} \cdot W_{00} + X_{i(j+1)} W_{01} + X_{i(j+2)} W_{02} + X_{(i+1)j} \cdot W_{10} + X_{(i+1)(j+1)} W_{11} + X_{(i+1)(j+2)} W_{12}$$

$$+ X_{(i+2)j} \cdot W_{20} + X_{(i+2)(j+1)} W_{21} + X_{(i+2)(j+2)} W_{22} + b$$

$$= \sum_{m=0}^{2} \sum_{n=0}^{2} X_{(i+m)(j+n)} \cdot W_{mn} + b$$

$$\text{gradient } W_{ij} = \sum_{m=0}^{3} \sum_{n=0}^{3} \text{gradient } Y_{mn} \cdot \nabla_{W_{ij}} Y_{mn} = \sum_{m=0}^{3} \sum_{n=0}^{3} \text{gradient } Y_{mn} \cdot X_{(m+i)(n+j)}$$

$$= \text{gradient } \underline{\underline{Y}} * \underline{\underline{X}}_{(i+0,j+0) \to (i+3,j+3)}$$

$$\text{gradient } X_{ij} = \sum_{m=0}^{2} \sum_{n=0}^{2} \text{gradient } Y_{(i-m)(j-n)} \cdot \nabla_{X_{ij}} Y_{(i-m)(j-n)} = \sum_{m=0}^{2} \sum_{n=0}^{2} \text{gradient } Y_{(i-m)(j-n)} \cdot W_{mn}$$

$$= \text{gradient } \underline{\underline{Y}}_{(i-0)(j-0) \to (i-2)(j-2)} * \underline{\underline{W}} = \text{gradient } \underline{\underline{Y}}_{(i-2)(j-2) \to (i,j)} * \text{Rot}180^\circ(\underline{\underline{W}})$$

$$\text{gradient } b = \sum_{m=0}^{3} \sum_{n=0}^{3} \text{gradient } Y_{mn} \cdot \nabla_b Y_{mn} = \sum_{m=0}^{3} \sum_{n=0}^{3} \text{gradient } Y_{mn} \cdot 1$$

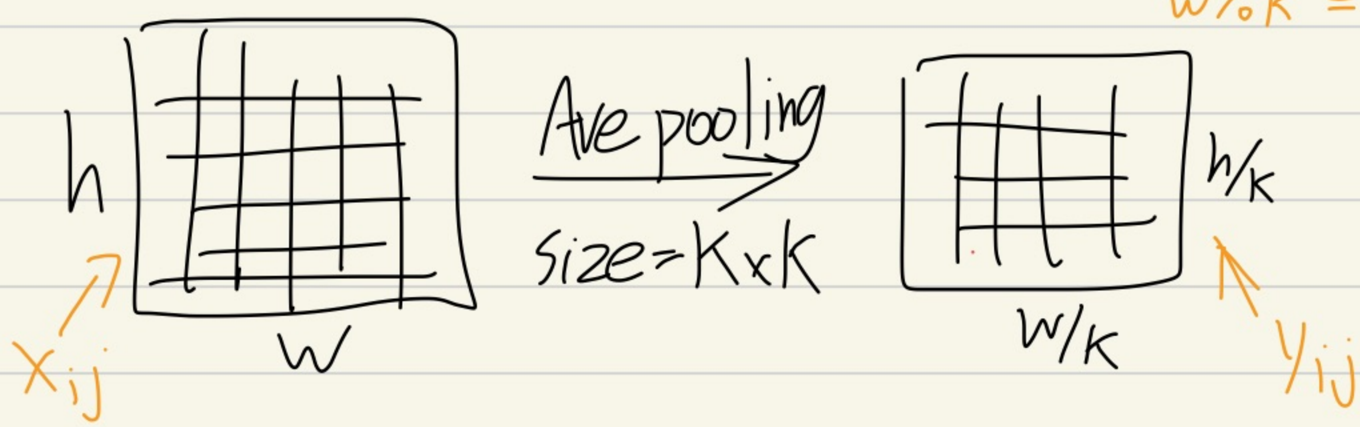
$$= \text{sum}(\text{gradient } \underline{\underline{Y}})$$

Summary

Average pooling

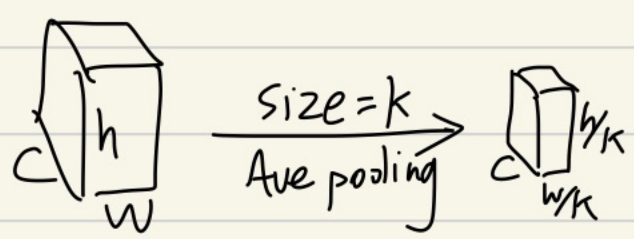
: Basically a conv with kernel size = stride = k, $w_{ij} = \frac{1}{k^2}$ $\forall i,j$
(only for channel = 1)

suppose $h \% k = 0$
$w \% k = 0$



$$\text{gradient } x_{ij} = \frac{1}{k^2} \text{ gradient } y_{(i//k)(j//k)}$$

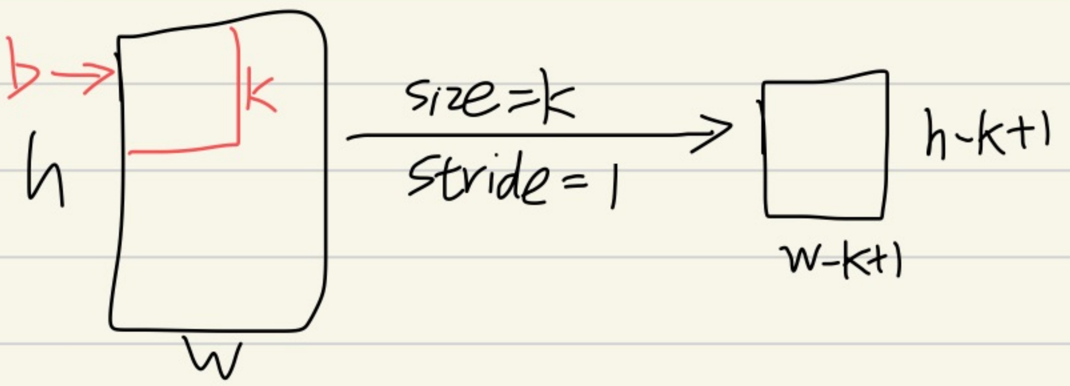
if channel = c

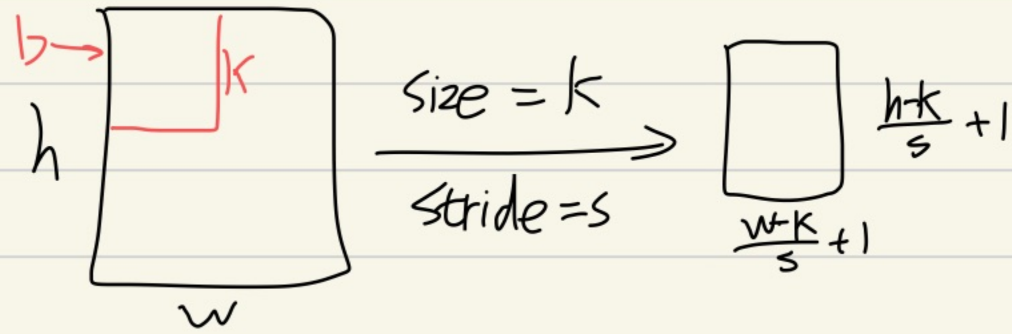


Forward: $$y_{ijc} = \frac{1}{k^2} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} x_{(k \cdot i+m)(k \cdot j+n)c}$$

Backward: $$\text{gradient } x_{ijc} = \frac{1}{k^2} \text{ gradient } y_{(i//k)(j//k)c}$$

Convolution





Suppose $\frac{w-k}{s}$ & $\frac{h-k}{s}$ are integers and $s < k$

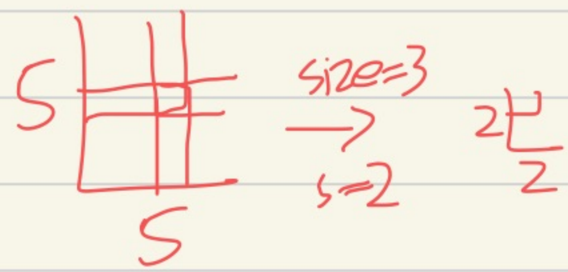
Forward $y_{ij} = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} X_{(s\cdot i+m)(s\cdot j+n)} \cdot W_{mn} + b$

Backward $\text{grad } \underline{\underline{W}} = \text{Rot}180^\circ(\text{Inserted Grad } \underline{Y} * \underline{\underline{X}})$

$\text{grad } \underline{\underline{X}} = \text{Inserted Grad } \underline{Y} * \text{Rot}180^\circ(\underline{\underline{W}})$

$\text{grad } b = \text{sum}(\text{gradient } \underline{Y})$

Ex:



(5x5 to 2x2 with kernel size = 3 stride = 2)

add zeros

Inserted Grad $\underline{Y}$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \text{grad}y_{00} & 0 & \text{grad}y_{01} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \text{grad}y_{10} & 0 & \text{grad}y_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{7\times 7}$$

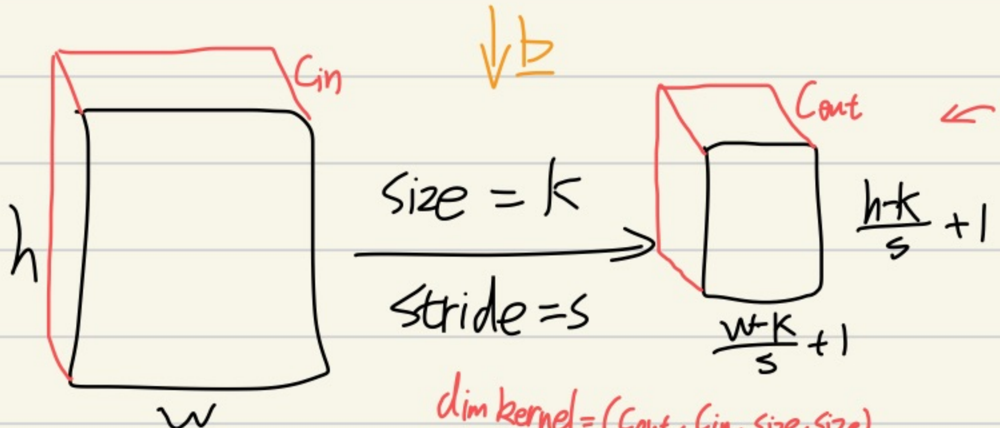
Generally, $\underbrace{h \times w}_{\underline{X}}$ to $\underbrace{\left(\frac{h-k}{s}+1\right) \times \left(\frac{w-k}{s}+1\right)}_{\underline{Y}}$

$$
\underbrace{\begin{bmatrix}
k-1\left\{\overbrace{\begin{matrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{matrix}}^{k-1}\right. & & & & \\
& grad\,Y_{00} & \overbrace{0 \cdots 0}^{s-1} & grad\,Y_{01} & \cdots & grad\,Y_{0,\frac{w-k}{s}} \\
\vdots & s-1\left\{\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right. & \begin{matrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{matrix} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} & \cdots & \\
& grad\,Y_{10} & 0 \cdots 0 & grad\,Y_{11} & & \\
\vdots & \vdots & \vdots & & & \\
& grad\,Y_{\frac{h-k}{s},0} & & & & \\
k-1\left\{\begin{matrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{matrix}\right. & & & & &
\end{bmatrix}}_{}
\leftarrow \text{Inserted Grad } \underline{Y}
$$

$$\left[\frac{h-k}{s}+1+(s-1)\cdot\frac{h-k}{s}+2(k-1)\right] \times \left[\frac{w-k}{s}+1+(s-1)\cdot\frac{w-k}{s}+2(k-1)\right]$$

eg. $(h+k-1)\times(w+k-1)$

Convolution [channel = Cin, Cout]



← dim Y = $\left(C_{out}, \frac{h-k}{s}+1, \frac{w-k}{s}+1\right)$

dim kernel = $(C_{out}, C_{in}, size, size)$

dim X = (C_{in}, h, w)

Suppose $\frac{w-k}{s}$ & $\frac{h-k}{s}$ are integers and $s < k$

Forward $$Y_{qij} = \sum_{p=0}^{C_{in}-1} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} X_{p(s\cdot i+m)(s\cdot j+n)} \cdot W_{qpmn} + b_q \quad [0 \le q \le C_{out}-1]$$

Backward $$\text{grad } \underline{\underline{W}}_{qp} = \text{Rot180°}\left(\text{Inserted Grad } \underline{Y}_q * \underline{\underline{X}}_p\right)$$

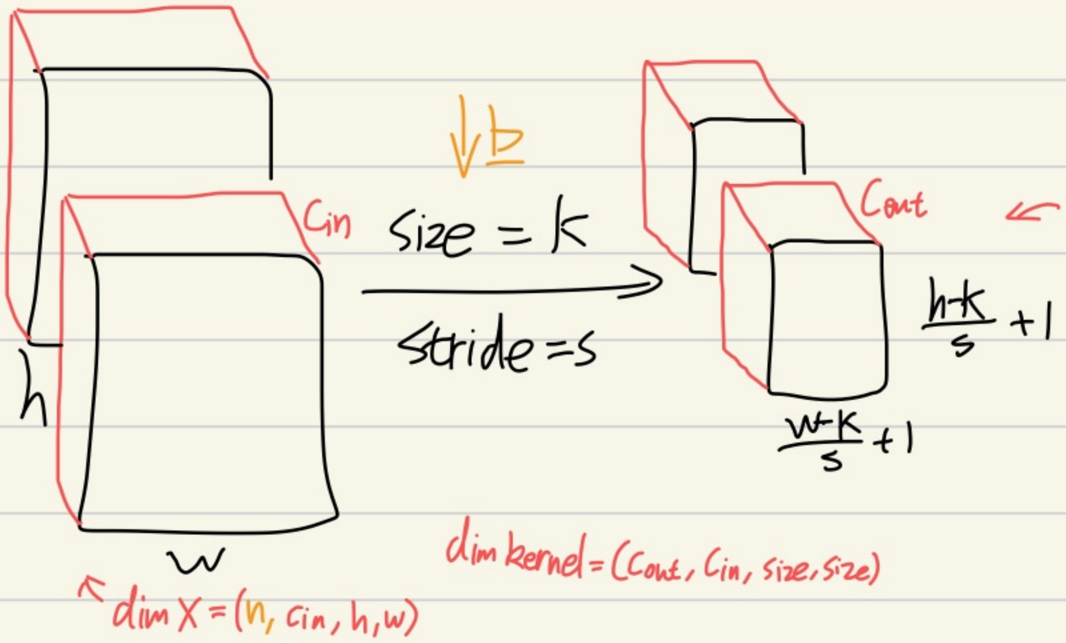
↳ need 2 loops to calculate $\underline{\underline{W}}$, but in numpy, some tricks can be used to accelerate computation.

$$\text{grad } \underline{\underline{X}} = \sum_{q=0}^{C_{out}-1} \underbrace{\text{Inserted Grad } \underline{Y}_q * \text{Rot180°}(\underline{\underline{W}}_q)}_{\text{layer-wise 2D conv}}$$

$$\underline{\text{grad } b_q = \text{sum}(\text{gradient } \underline{Y}_q)}$$

with numpy, $\underline{b}$ = np.sum(gradient $\underline{\underline{Y}}$, axis=(1,2)) is faster than loop

Convolution [batch = n, channel = Cin, Cout]



dim Y = $\left(n, C_{out}, \frac{h-k}{s}+1, \frac{w-k}{s}+1\right)$

Suppose $\frac{w-k}{s}$ & $\frac{h-k}{s}$ are integers and $s < K$

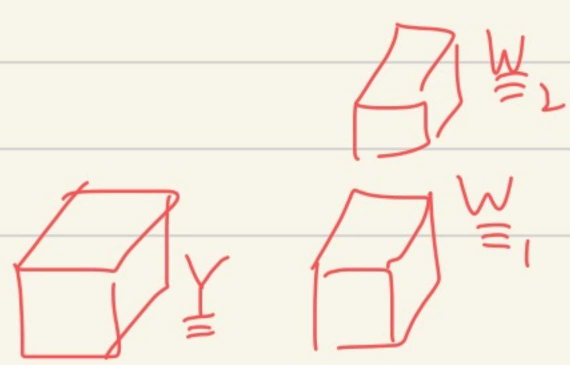
Forward $\quad Y_{nqij} = \sum_{p=0}^{c_{in}-1} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \underset{\text{batch}}{X_n}\, p\, (s\cdot i+m)\, \underset{\text{kernel width}}{(s\cdot j+n)} \cdot W_{qpmn} + b_q \quad [0 \le q \le C_{out}-1]$

Backward $\quad$ grad $\underline{\underline{W}}_{qp} = \sum_n \mathrm{Rot}180^\circ(\text{Inserted Grad } \underline{Y}_{nq} * \underline{\underline{X}}_{np})$

grad $\underline{\underline{X}}_n = \sum_{q=0}^{C_{out}-1} \text{Inserted Grad } \underline{\underline{Y}}_{nq} * \mathrm{Rot}180^\circ(\underline{\underline{W}}_q)$

grad $b_q = \sum_n \mathrm{sum}(\text{gradient } \underline{Y}_{nq})$

the idea is pretty simple, but if loops are used to calculate them, trianing time would be unacceptable.



Master Dimension

Dimension's Secret