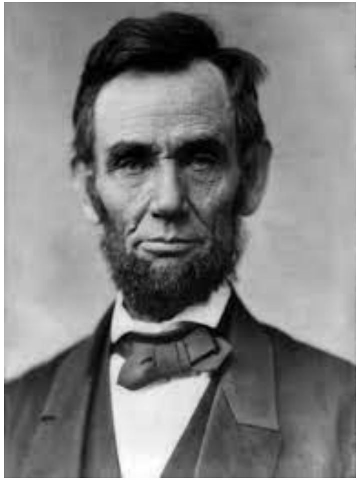
Recitation 5

CNN: Basics and Backprop

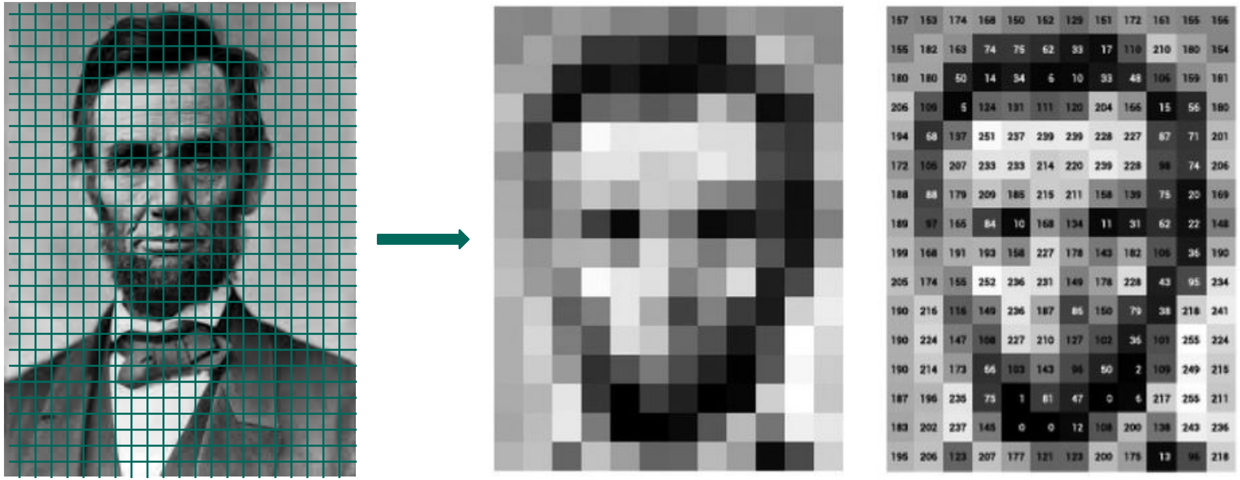
What is an image?

A visual representation



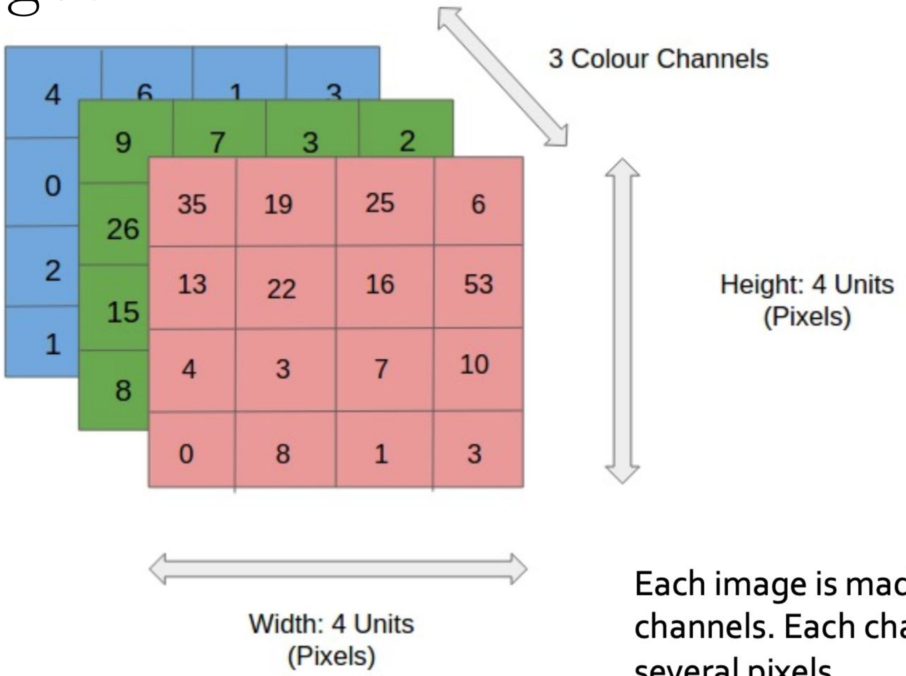
What is an image? : For a computer!

~~A visual representation~~. A Matrix **`I`** of dimensions **`(M,N)`** with **`I[i][j] = intensity(pixel(i,j))`**



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

What is an image?



I → (3,M,N)

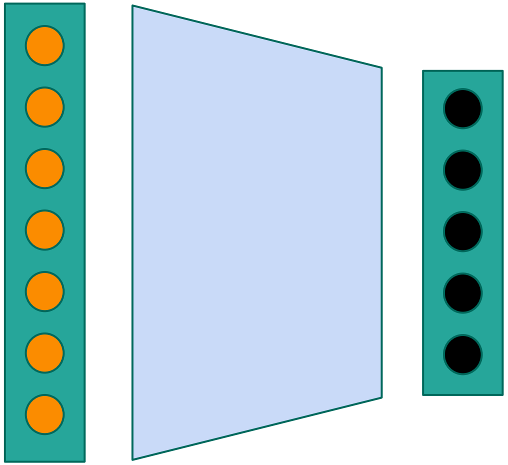
I[c][i][j] =

Intensity at **pixel(i,j)** for channel **c**

Each image is made up of a set of channels. Each channel comprises of several pixels
3 for a colored image, 2 for B&W.

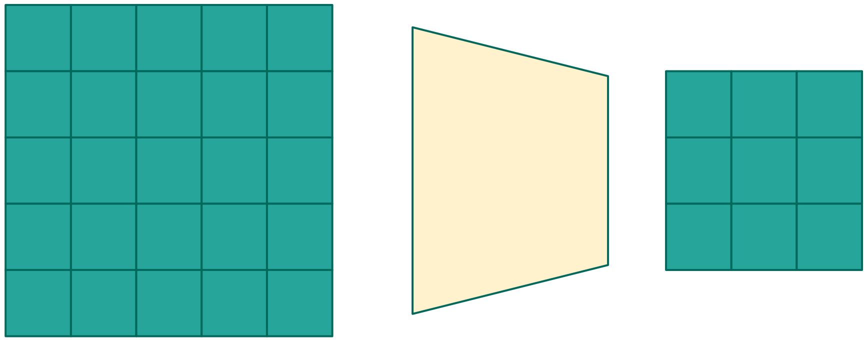
The number of channels you encounter could even increase!

MLP



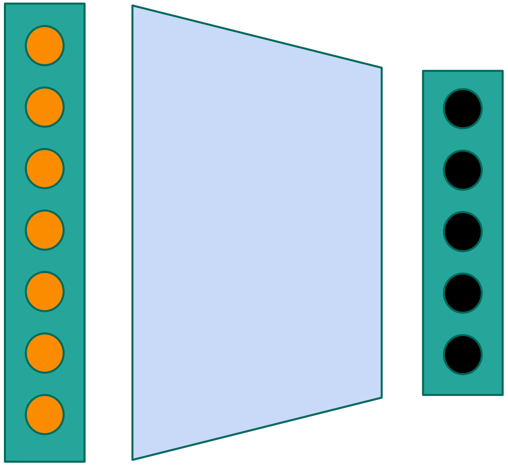
Vector to Vector

CNN

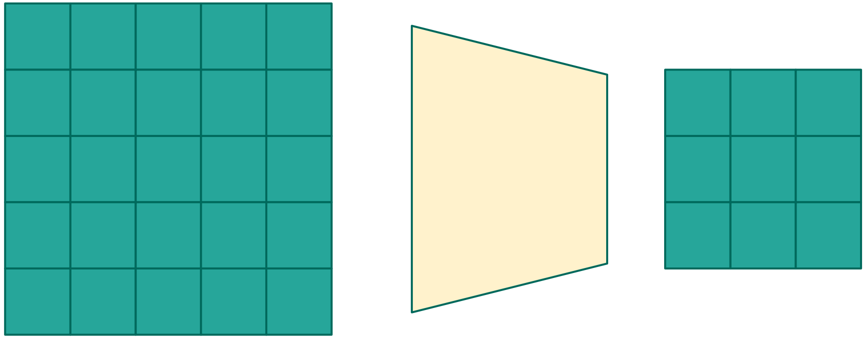


Feature map to Feature map

MLP Vs. CNN



Vector to Vector



Feature map to Feature map

Building Blocks of a CNN

- Convolution Layer
- Pooling Layer

Hyperparameters

- Filter/Kernel size
- **Stride**
- # of filters, # of layers
- **Pooling** size & type
- **Padding** Type

Convolution

Essentially element-wise (Hadamard) multiplications and summations

Input - **A**

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

*

Kernel - **W**

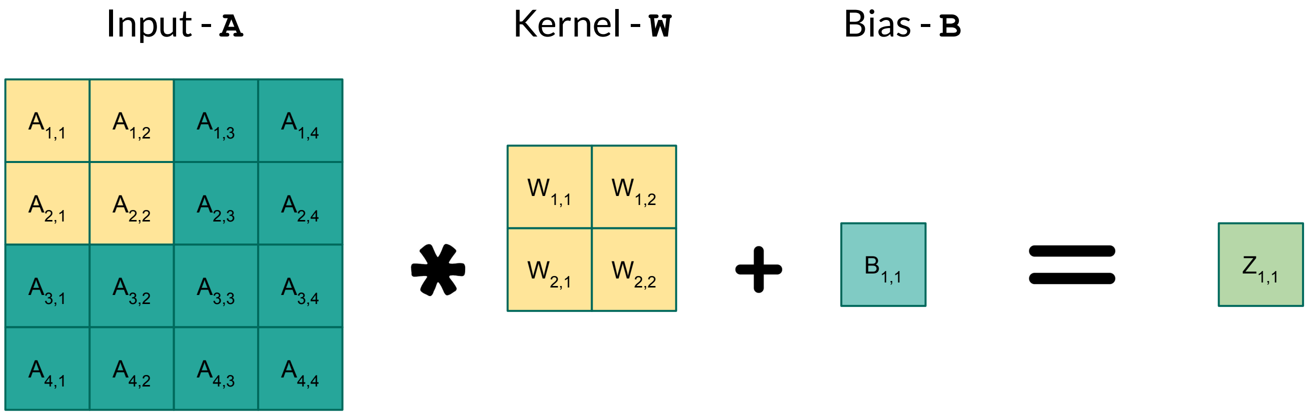
-1	0	1
-2	0	2
-1	0	1

=

0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0

Convolution

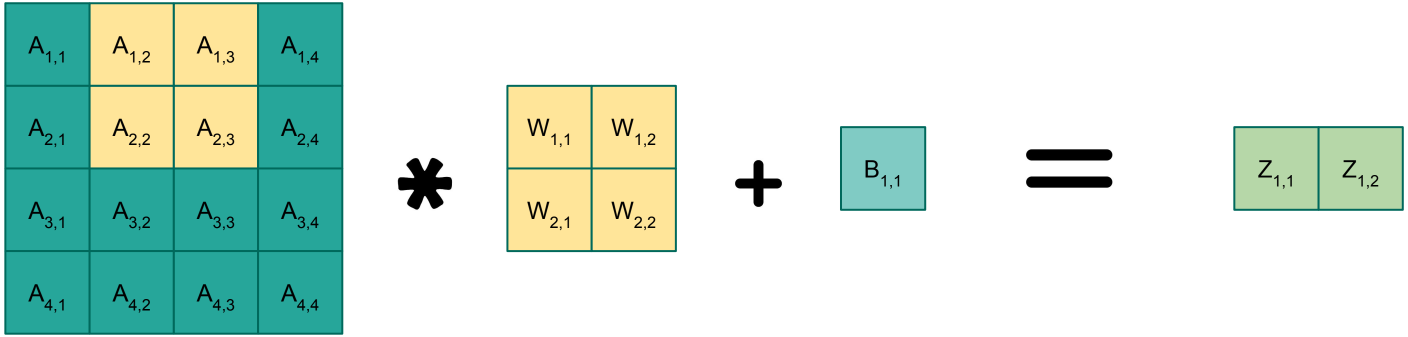
Essentially element-wise (Hadamard) multiplications and summations



$$Z_{1,1} = (A_{1,1} * W_{1,1}) + (A_{1,2} * W_{1,2}) + (A_{2,1} * W_{2,1}) + (A_{2,2} * W_{2,2}) + B$$

Convolution

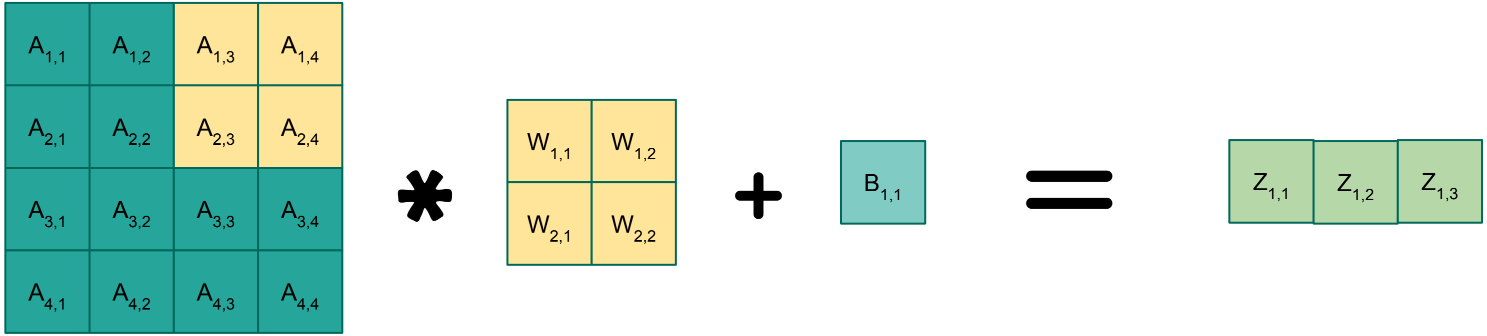
Essentially element-wise (Hadamard) multiplications and summations



$$Z_{1,2} = (A_{1,2} * W_{1,1}) + (A_{1,3} * W_{1,2}) + (A_{2,2} * W_{2,1}) + (A_{2,3} * W_{2,2}) + B$$

Convolution

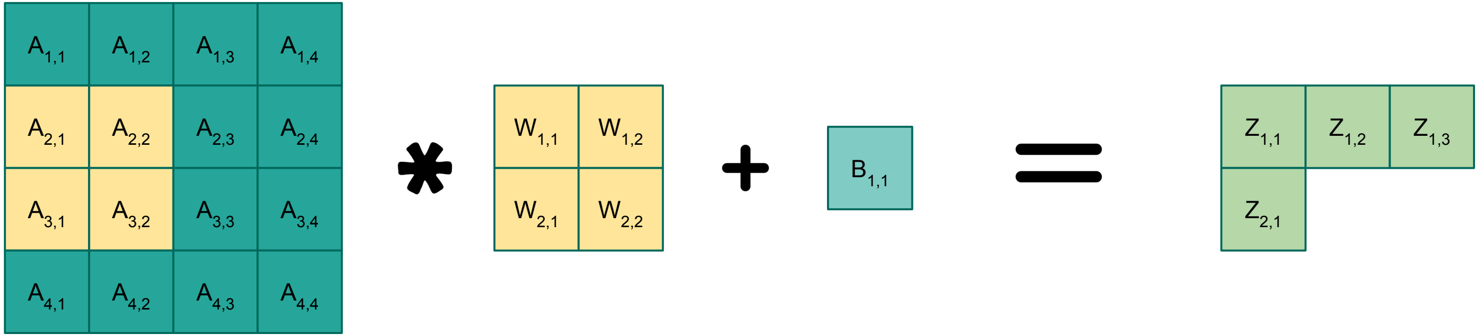
Essentially element-wise (Hadamard) multiplications and summations



$$Z_{1,3} = (A_{1,3} * W_{1,1}) + (A_{1,4} * W_{1,2}) + (A_{2,3} * W_{2,1}) + (A_{2,4} * W_{2,2}) + B$$

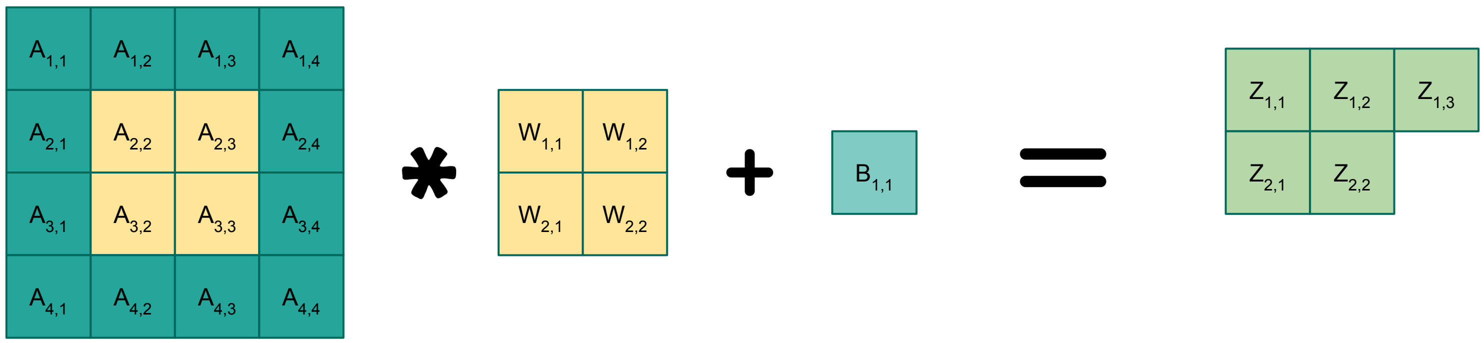
Convolution

Essentially element-wise (Hadamard) multiplications and summations



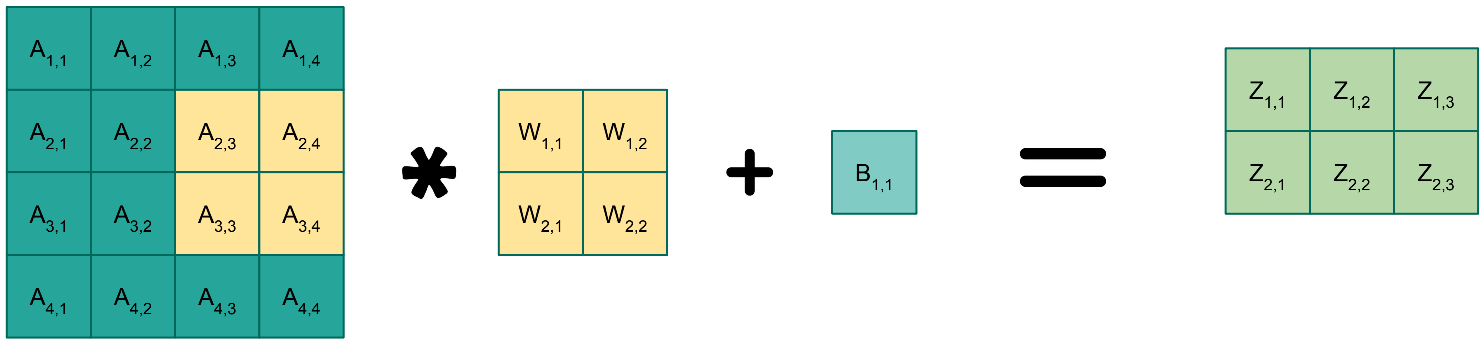
Convolution

Essentially element-wise (Hadamard) multiplications and summations



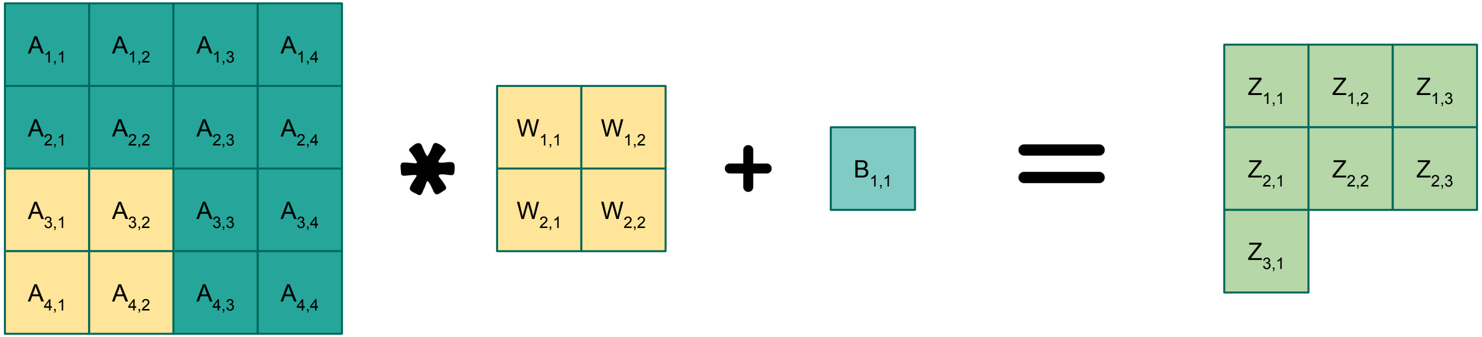
Convolution

Essentially element-wise (Hadamard) multiplications and summations



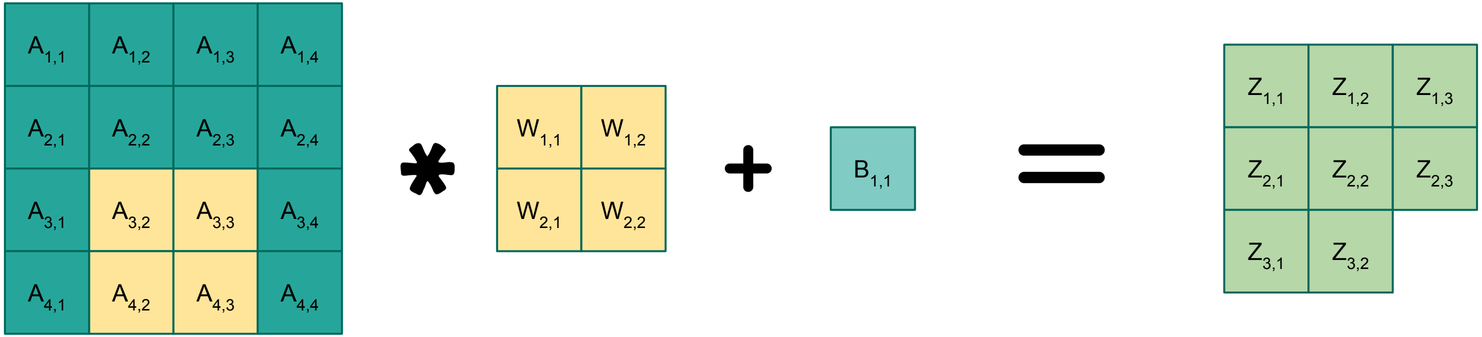
Convolution

Essentially element-wise (Hadamard) multiplications and summations



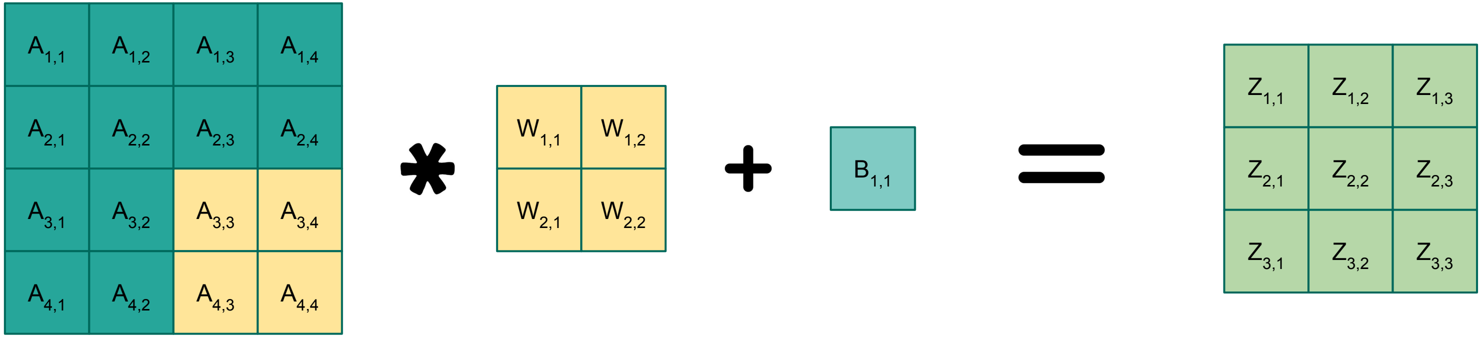
Convolution

Essentially element-wise (Hadamard) multiplications and summations

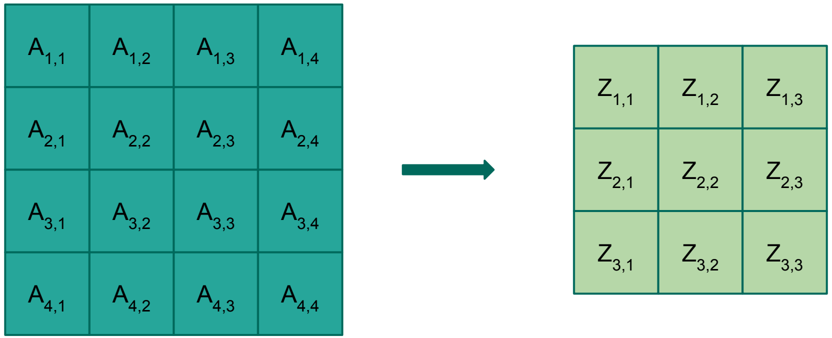


Convolution

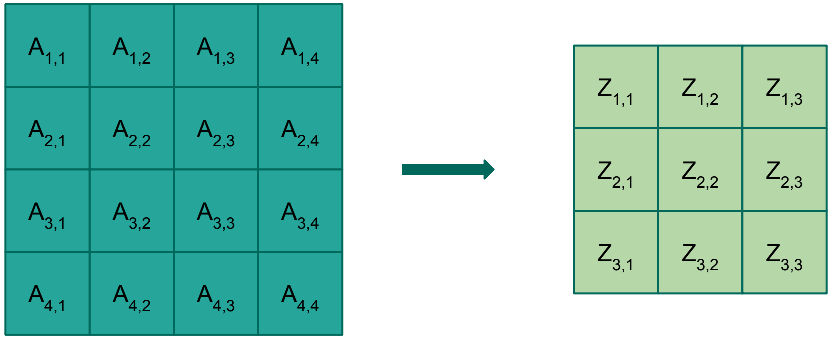
Essentially element-wise (Hadamard) multiplications and summations



Output Size


$A_{1,1}$ $A_{1,2}$ $A_{1,3}$ $A_{1,4}$
$A_{2,1}$ $A_{2,2}$ $A_{2,3}$ $A_{2,4}$
$A_{3,1}$ $A_{3,2}$ $A_{3,3}$ $A_{3,4}$
$A_{4,1}$ $A_{4,2}$ $A_{4,3}$ $A_{4,4}$
$Z_{1,1}$ $Z_{1,2}$ $Z_{1,3}$
$Z_{2,1}$ $Z_{2,2}$ $Z_{2,3}$
$Z_{3,1}$ $Z_{3,2}$ $Z_{3,3}$

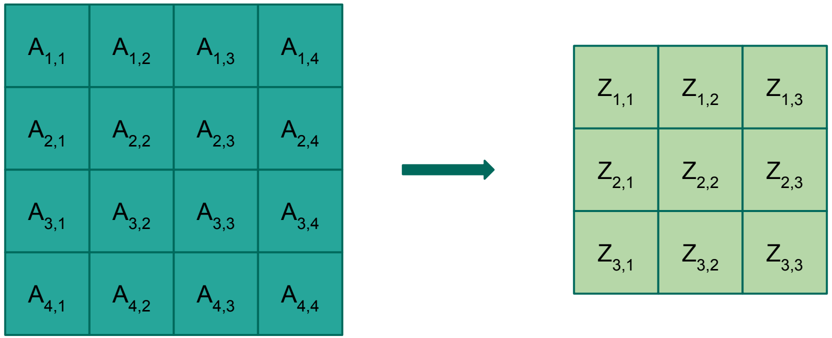
Output Size



Output Width =
$[(W_{in} - W_k + 2P) \,//\, (S)] + 1$

Same goes for Height.

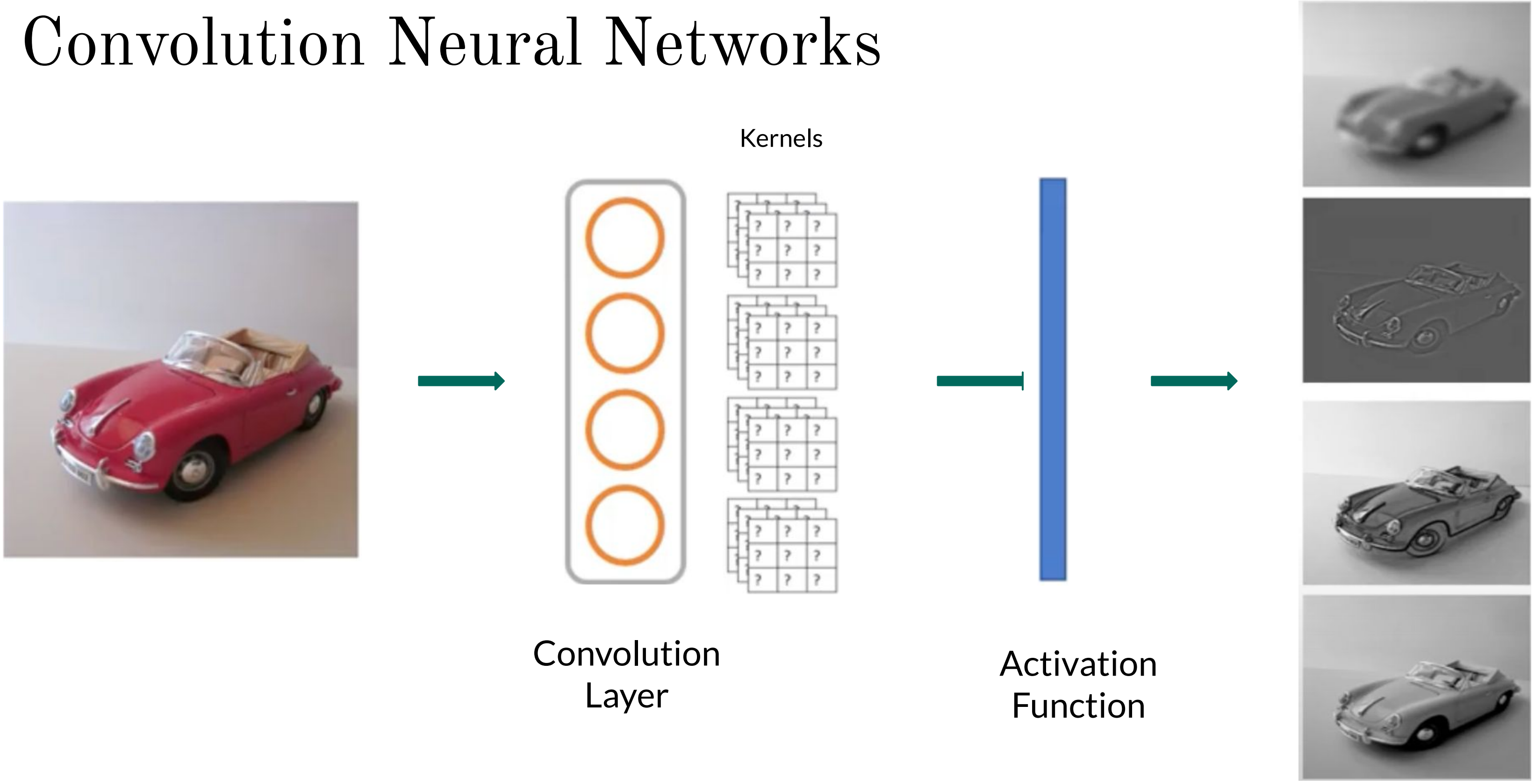
Output Size



Output Width =
$[(W_{in} - W_k + 2\mathbf{P}) // (\mathbf{S})] + 1$

P: Padding (here - 0)
S: Stride (here - 1)

Convolution Neural Networks

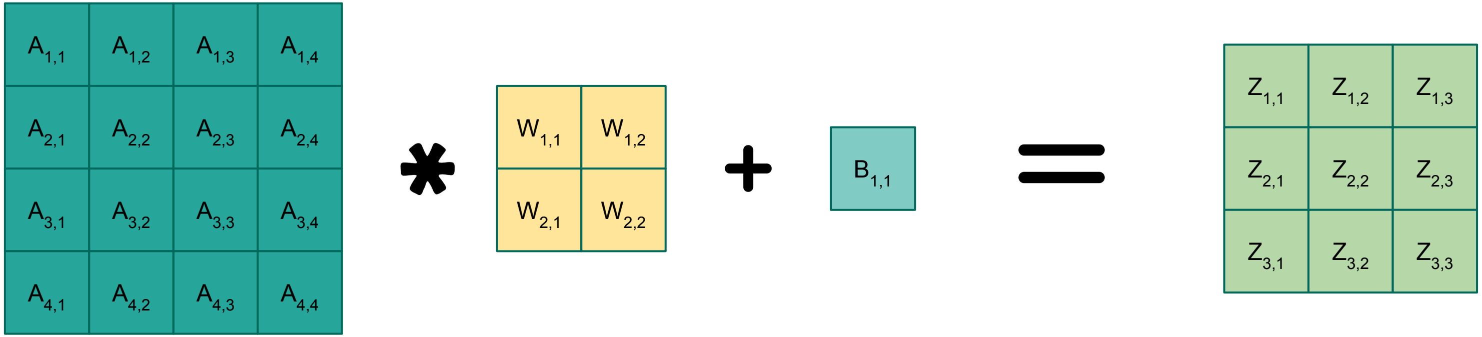


Stride

Taking bigger steps!

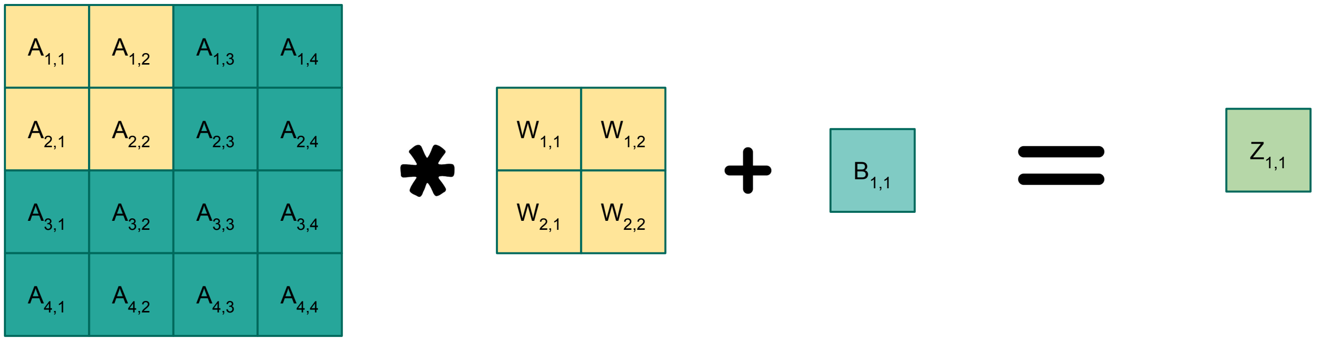
Stride = 1

What we did before - The kernel “moves” one pixel (or element) at a time.



Stride = 2

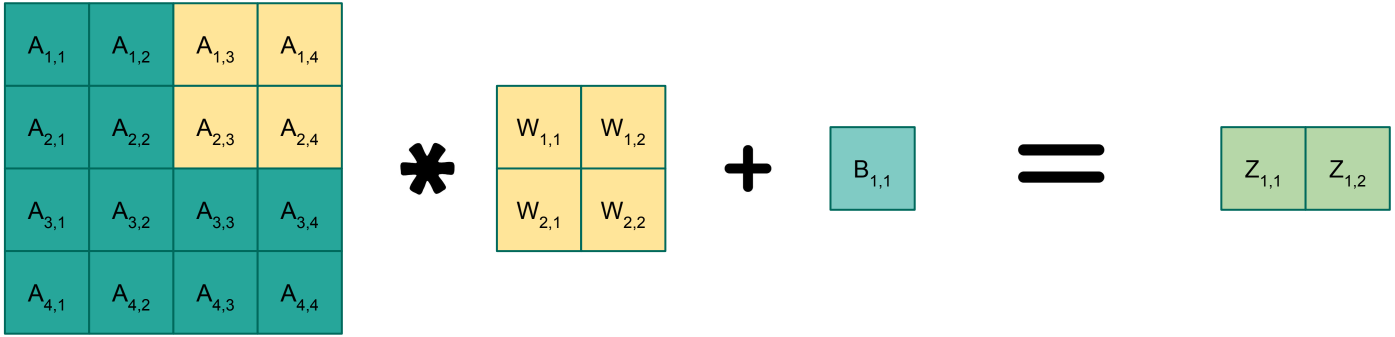
Start at the same place



$$Z_{1,1} = (A_{1,1}*W_{1,1}) + (A_{1,2}*W_{1,2}) + (A_{2,1}*W_{2,1}) + (A_{2,2}*W_{2,2}) + B$$

Stride = 2

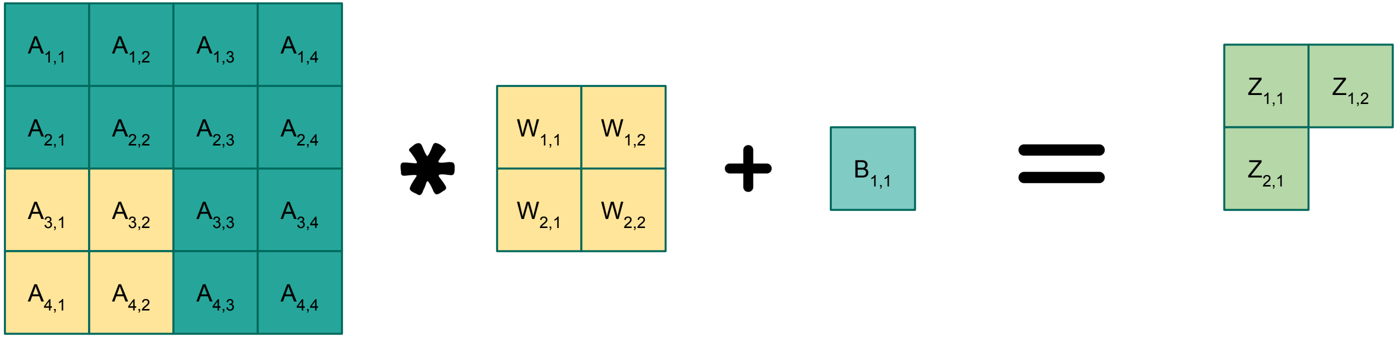
Move two elements to the right



$$Z_{1,2} = (A_{1,3} * W_{1,1}) + (A_{1,4} * W_{1,2}) + (A_{2,3} * W_{2,1}) + (A_{2,4} * W_{2,2}) + B$$

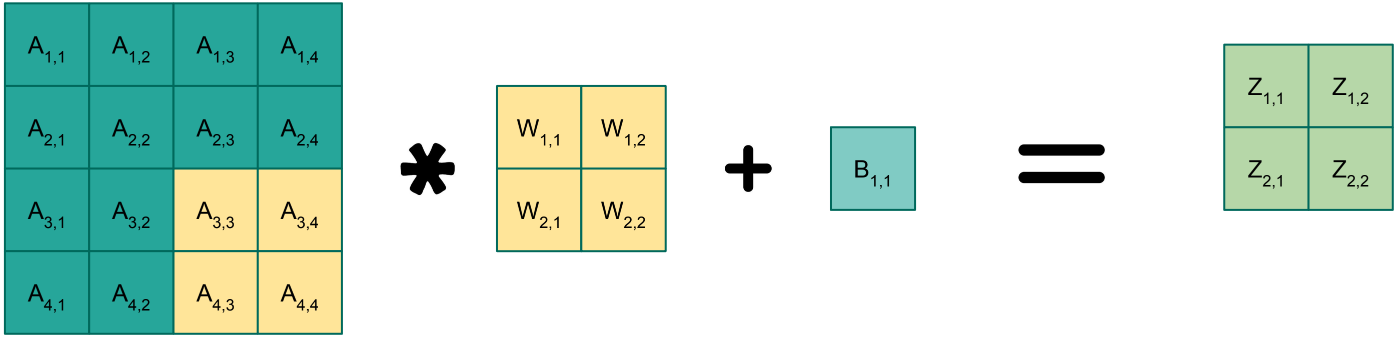
Stride = 2

Move two elements down.



Stride = 2

Move two elements to the right.



Interpreting Stride > 1

Think about how it is related to Upsampling(and Downsampling.

Will learn more in HW2

A

0	2	-3	2	-3
-2	-2	-2	-1	-2
-3	-3	2	-2	1
-3	-2	1	-3	1
0	-1	0	2	-3

Input Image
5x5

$*$

W

-1	0	-2
1	1	-1
-1	-1	0

Kernel
3x3

$+$

b

-1

Bias
1x1

9	-9	7
2	5	6
-7	9	-10

Stride 1 output

→

9	-9	7
2	5	6
-7	9	-10

Drop intermediates

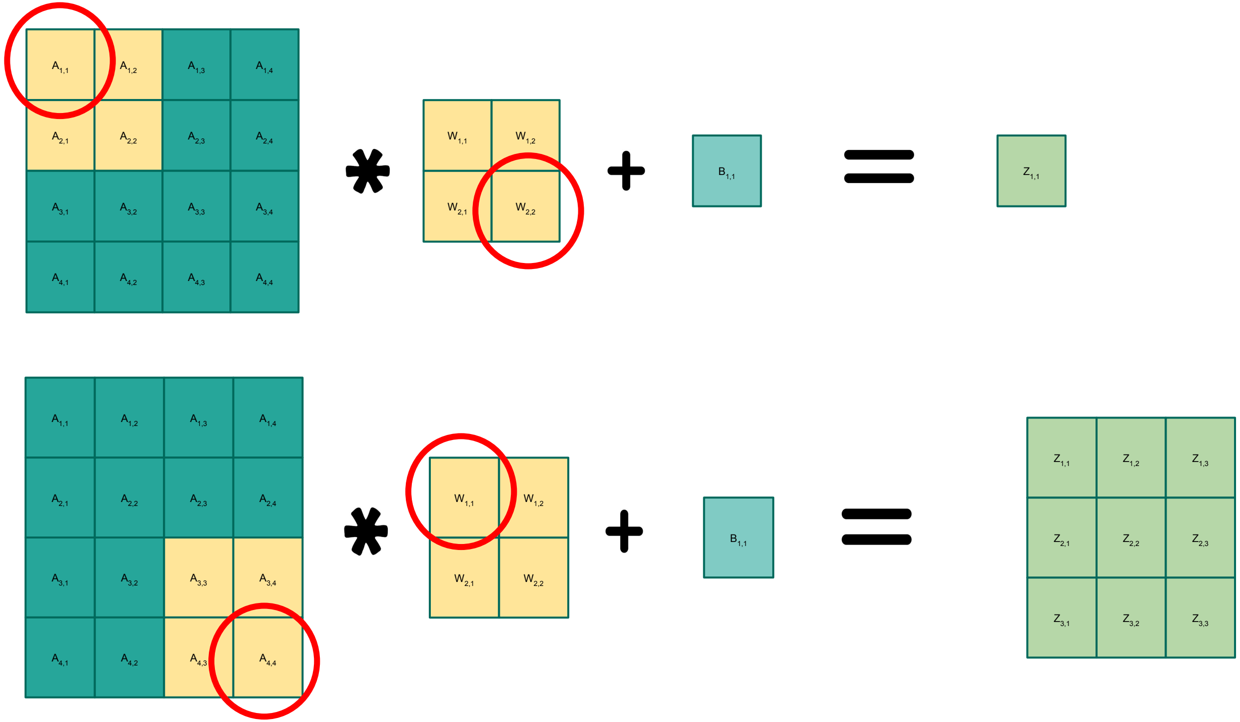
→

9	7
-7	-10

Stride 2 output

Padding

Padding



A1,1 A1,2 A1,3 A1,4
A2,1 A2,2 A2,3 A2,4
A3,1 A3,2 A3,3 A3,4
A4,1 A4,2 A4,3 A4,4
W1,1 W1,2
W2,1 W2,2
B1,1
Z1,1
A1,1 A1,2 A1,3 A1,4
A2,1 A2,2 A2,3 A2,4
A3,1 A3,2 A3,3 A3,4
A4,1 A4,2 A4,3 A4,4
W1,1 W1,2
W2,1 W2,2
B1,1
Z1,1 Z1,2 Z1,3
Z2,1 Z2,2 Z2,3
Z3,1 Z3,2 Z3,3

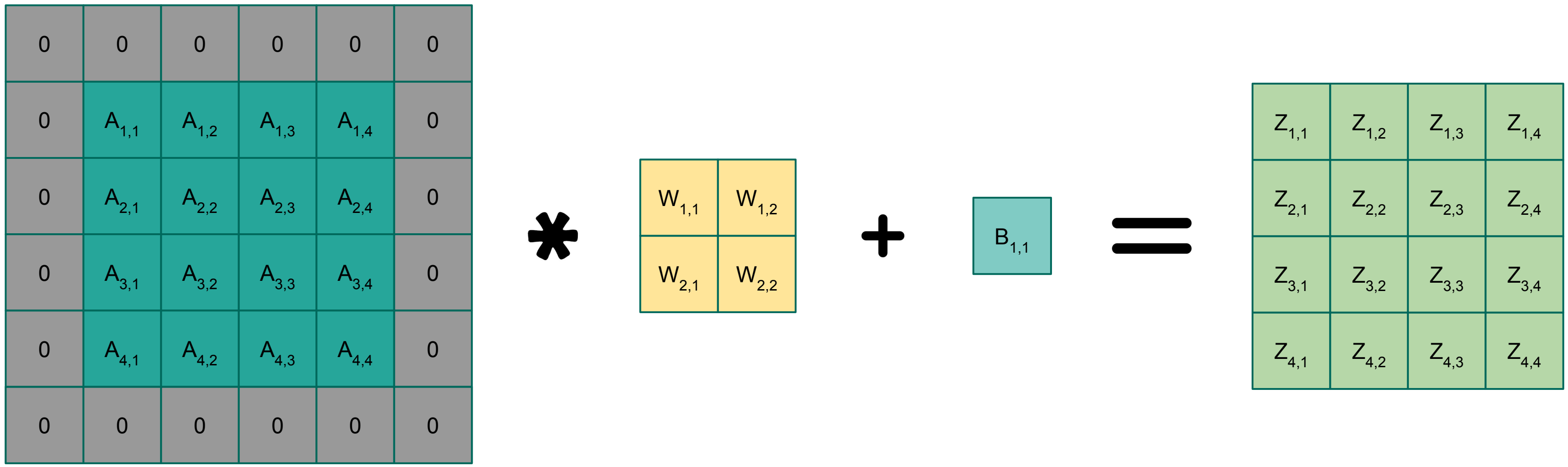
Padding

Increase output size

Preserve input size

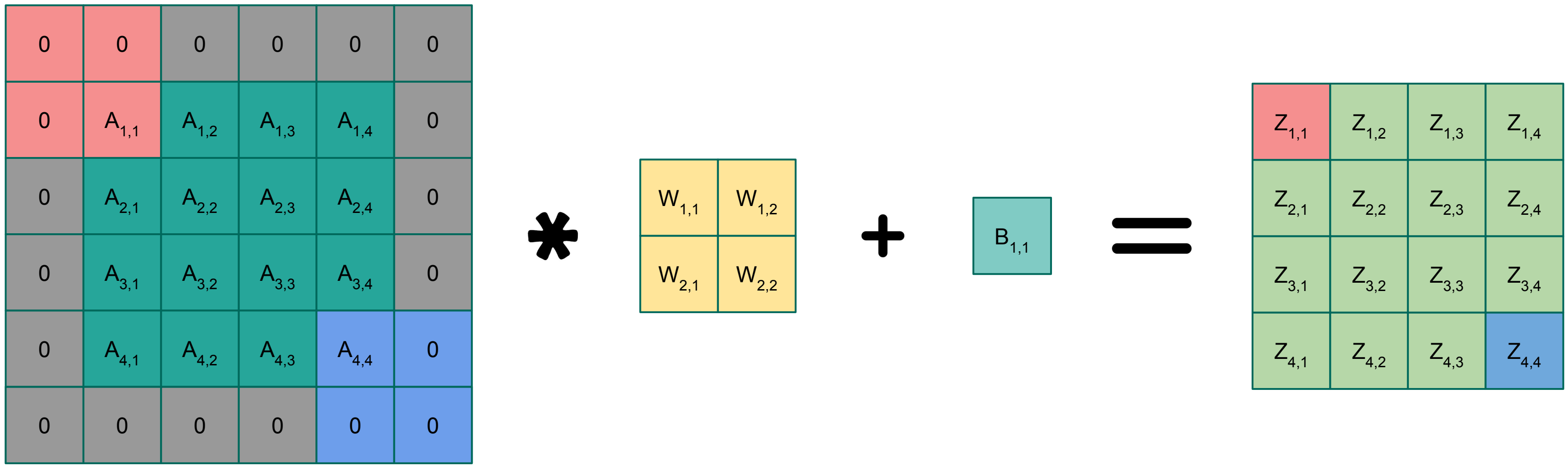
More Kernel Interactions!

Padding

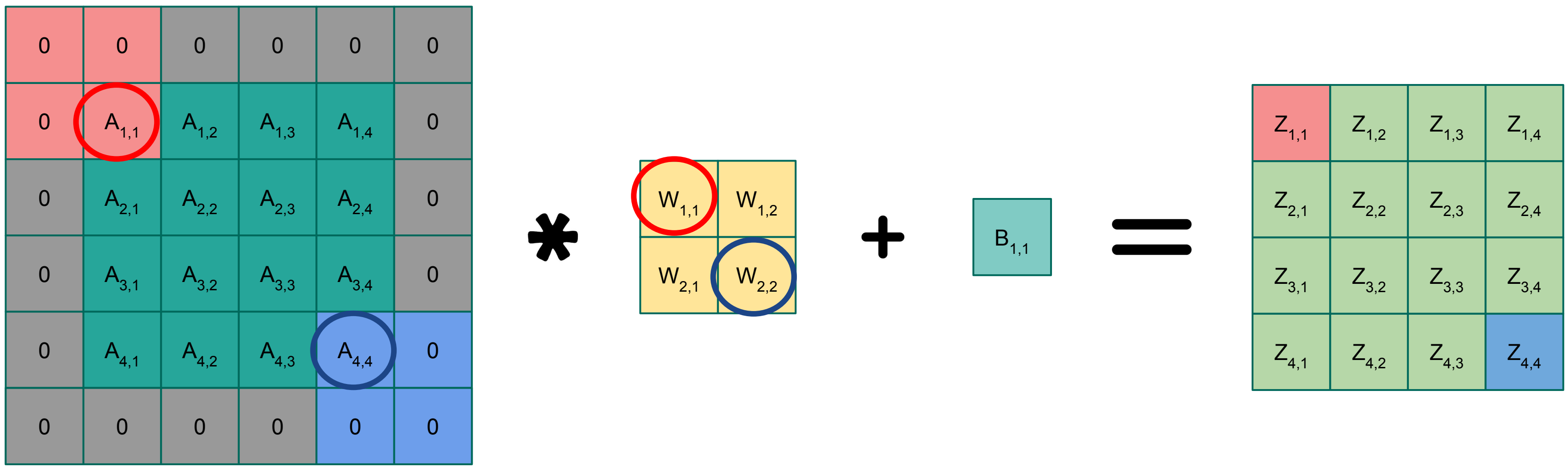


0 0 0 0 0 0
0 A1,1 A1,2 A1,3 A1,4 0
0 A2,1 A2,2 A2,3 A2,4 0
0 A3,1 A3,2 A3,3 A3,4 0
0 A4,1 A4,2 A4,3 A4,4 0
0 0 0 0 0 0
*
W1,1 W1,2
W2,1 W2,2
+
B1,1
=
Z1,1 Z1,2 Z1,3 Z1,4
Z2,1 Z2,2 Z2,3 Z2,4
Z3,1 Z3,2 Z3,3 Z3,4
Z4,1 Z4,2 Z4,3 Z4,4

Padding

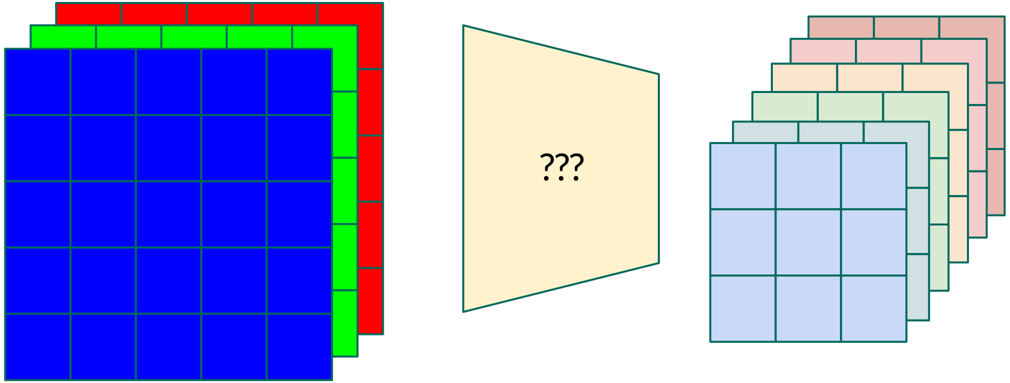

0 0 0 0 0 0
0 A1,1 A1,2 A1,3 A1,4 0
0 A2,1 A2,2 A2,3 A2,4 0
0 A3,1 A3,2 A3,3 A3,4 0
0 A4,1 A4,2 A4,3 A4,4 0
0 0 0 0 0 0
*
W1,1 W1,2
W2,1 W2,2
+
B1,1
=
Z1,1 Z1,2 Z1,3 Z1,4
Z2,1 Z2,2 Z2,3 Z2,4
Z3,1 Z3,2 Z3,3 Z3,4
Z4,1 Z4,2 Z4,3 Z4,4

Padding



0 0 0 0 0 0
0 A1,1 A1,2 A1,3 A1,4 0
0 A2,1 A2,2 A2,3 A2,4 0
0 A3,1 A3,2 A3,3 A3,4 0
0 A4,1 A4,2 A4,3 A4,4 0
0 0 0 0 0 0
*
W1,1 W1,2
W2,1 W2,2
+
B1,1
=
Z1,1 Z1,2 Z1,3 Z1,4
Z2,1 Z2,2 Z2,3 Z2,4
Z3,1 Z3,2 Z3,3 Z3,4
Z4,1 Z4,2 Z4,3 Z4,4

Multi-channel CNN

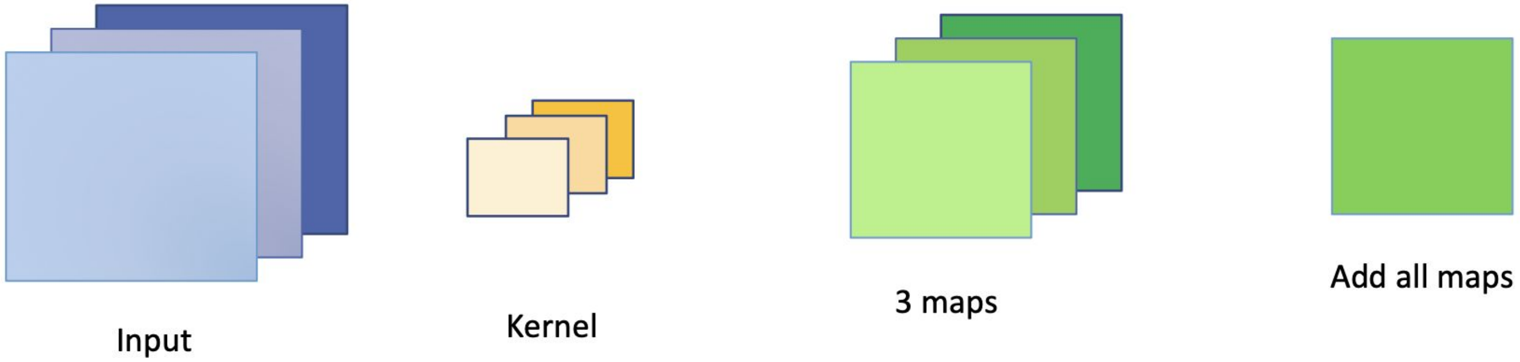


Multi-channel CNN

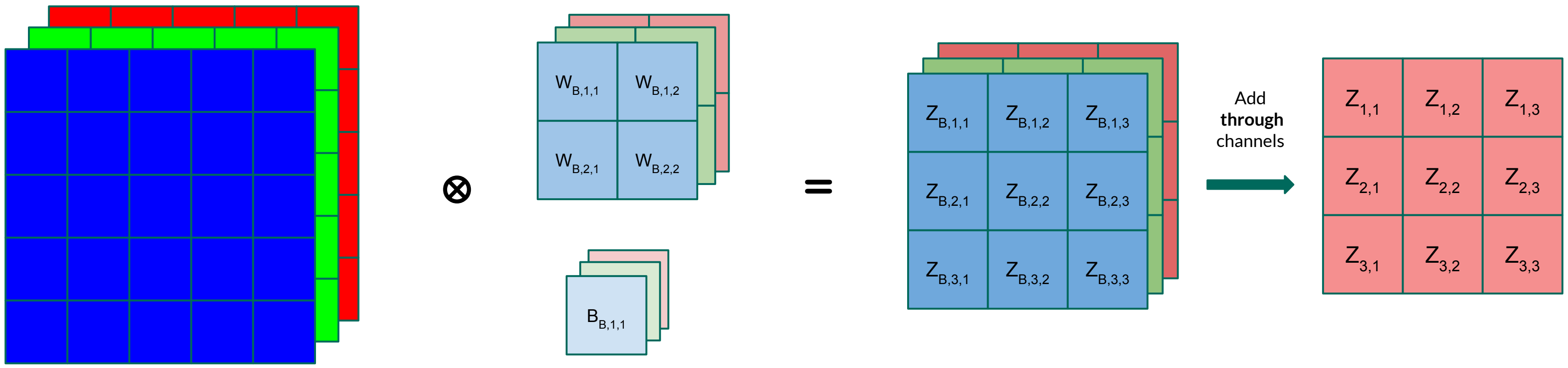
- Each kernel (or **filter**) has as many channels as the input does.

  `[kernel channels = Input channels]`

- Channel **`c` of the kernel** convolves with channel **`c`** (corresponding) **of the input**.
- The number of output channels from the convolution = number of **filters**



1 Filter with 3-channel input



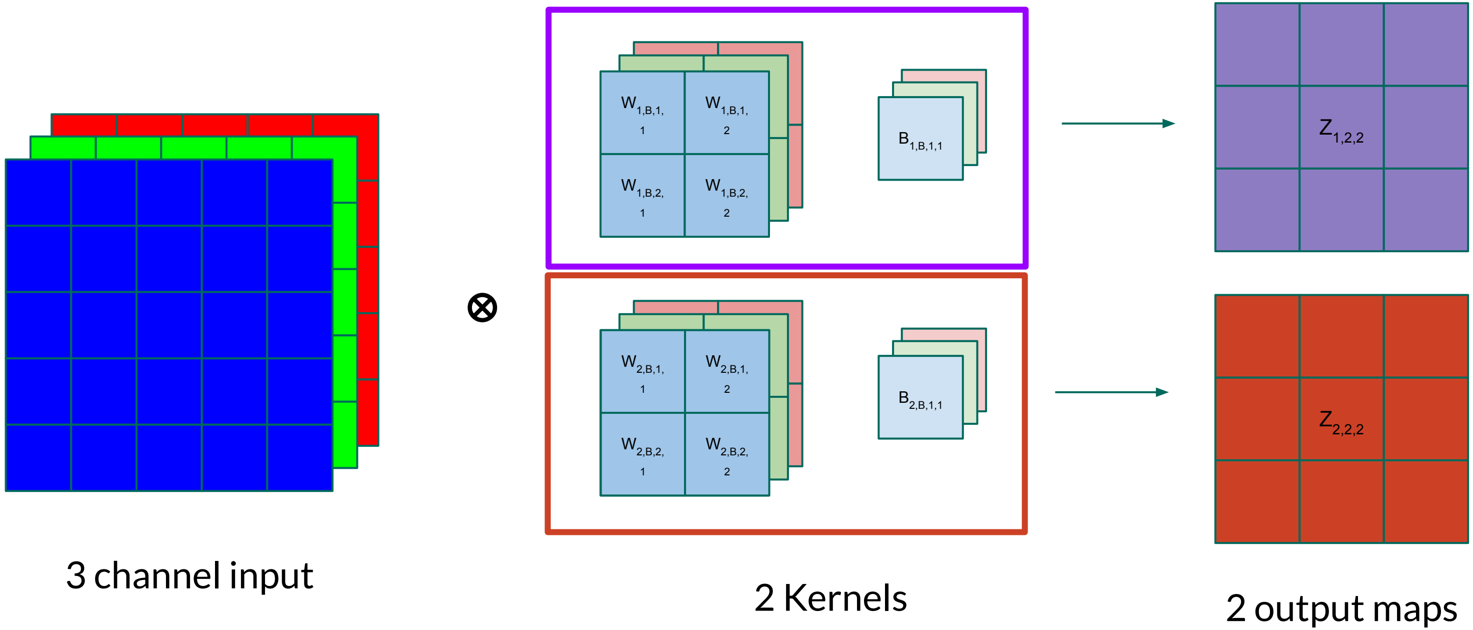
3 channel input

Kernel

3 almost-output maps

1 output map

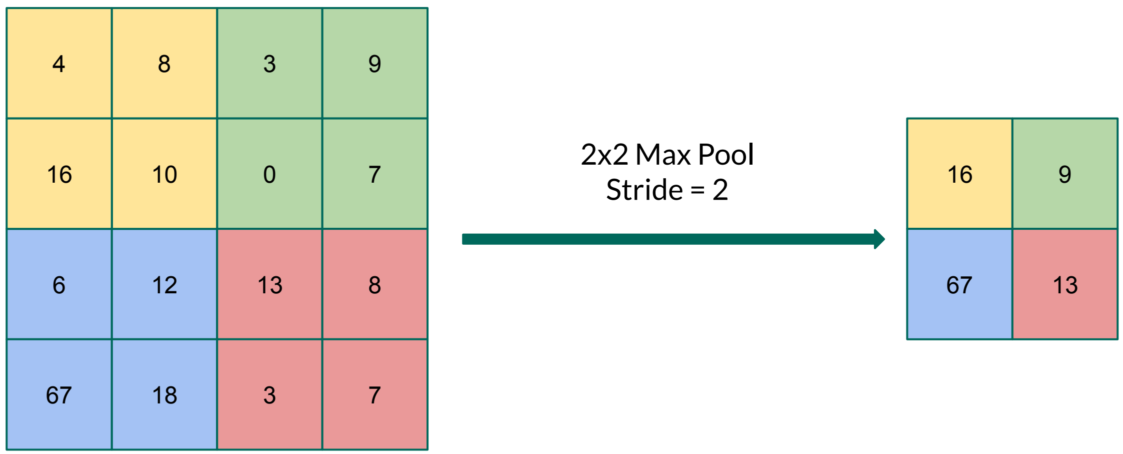
2 Filters with 3-channel input



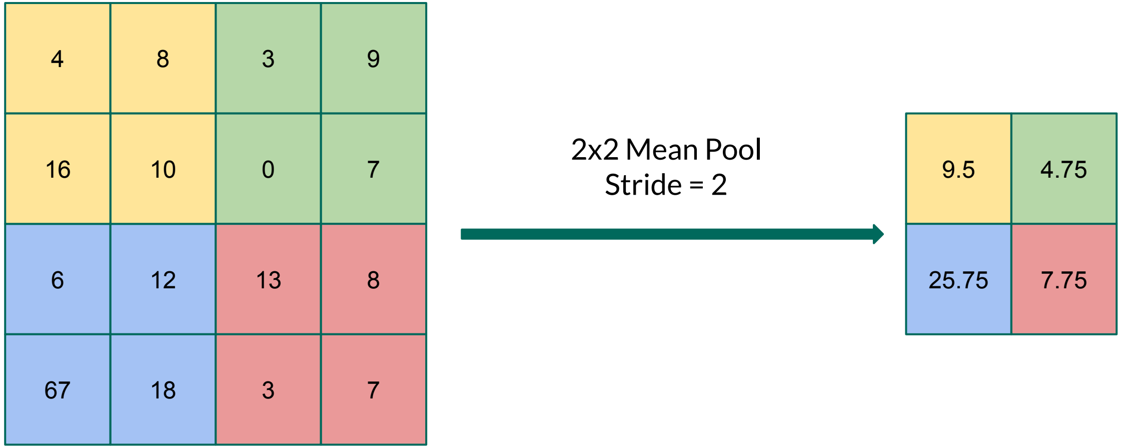
Pooling

- Usually follows convolutions
- Introduces Jitter Invariance
- Reduces memory footprint by reducing the feature-map size
- **Max**, **Mean**, Min
- **Pooling preserves number of channels**

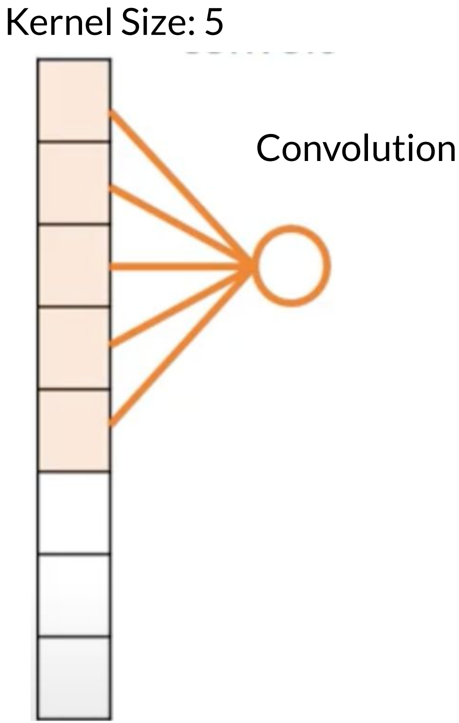
Pooling

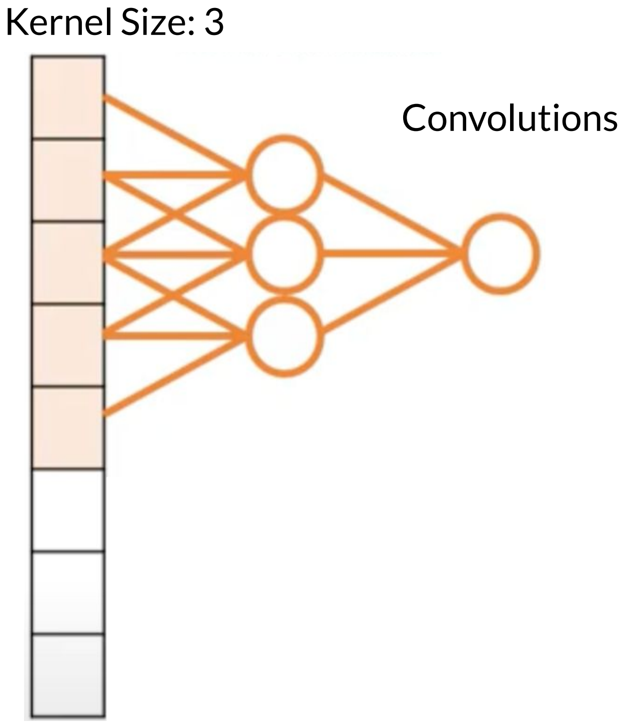


Pooling

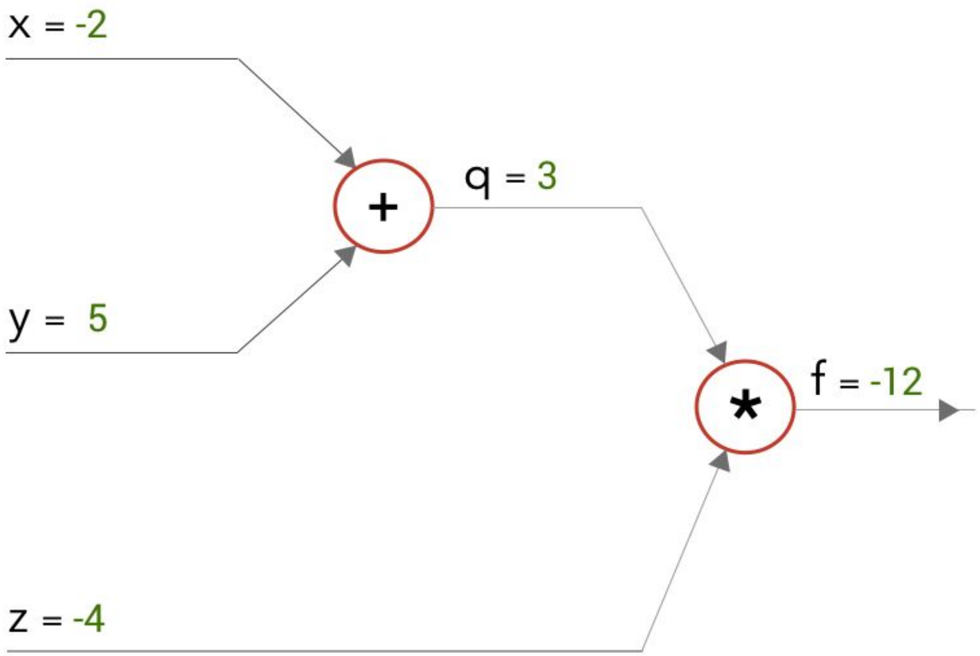

4
8
3
9
16
10
0
7
6
12
13
8
67
18
3
7
2x2 Mean Pool
Stride = 2
9.5
4.75
25.75
7.75

Kernel Size



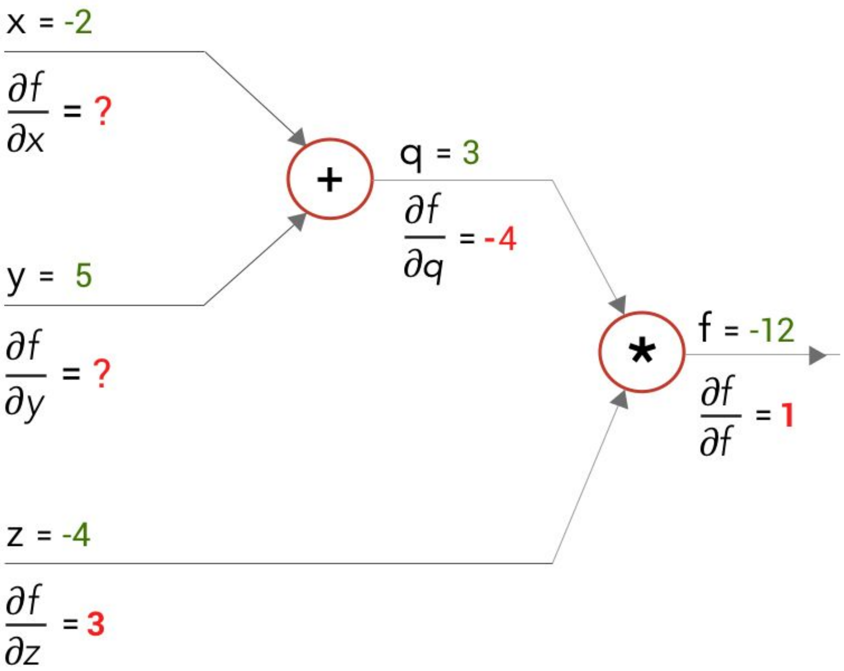


Backpropagation: Chain Rule Refresher



All visual credits to: https://pavisj.medium.com/convolutions-and-backpropagations-46026a8f5d2c

Backpropagation: Chain Rule Refresher



$f = q * z$

$\frac{\partial f}{\partial q} = z \mid z = -4$

$\frac{\partial f}{\partial z} = q \mid q = 3$

$q = x + y$

$\frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$

Backpropagation: Chain Rule Refresher

$$\frac{\partial q}{\partial x} = 1 \qquad \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = -4$$

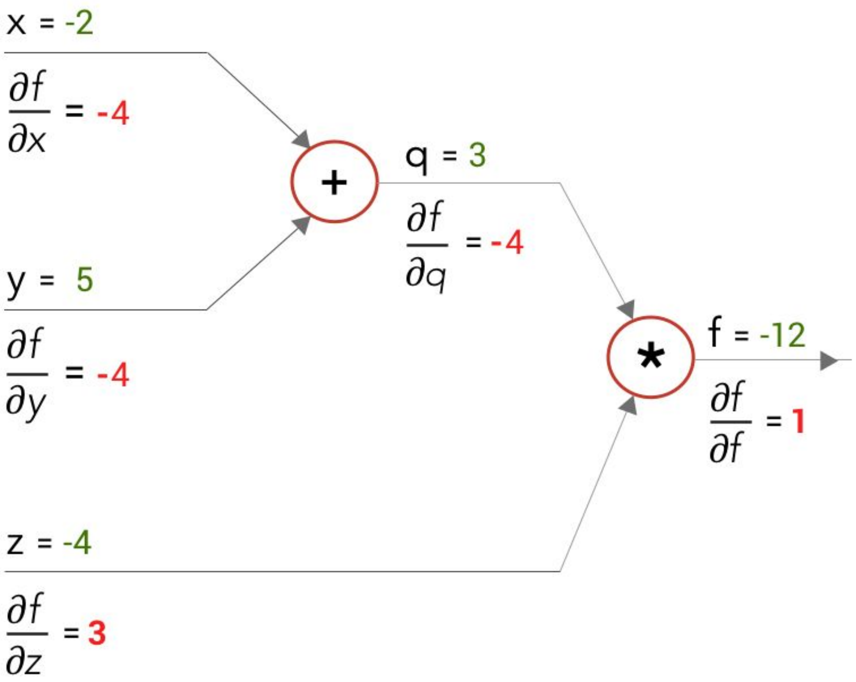
$$\frac{\partial f}{\partial x} = ?$$

$$\frac{\partial f}{\partial y} = ?$$

Using chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial x}$$

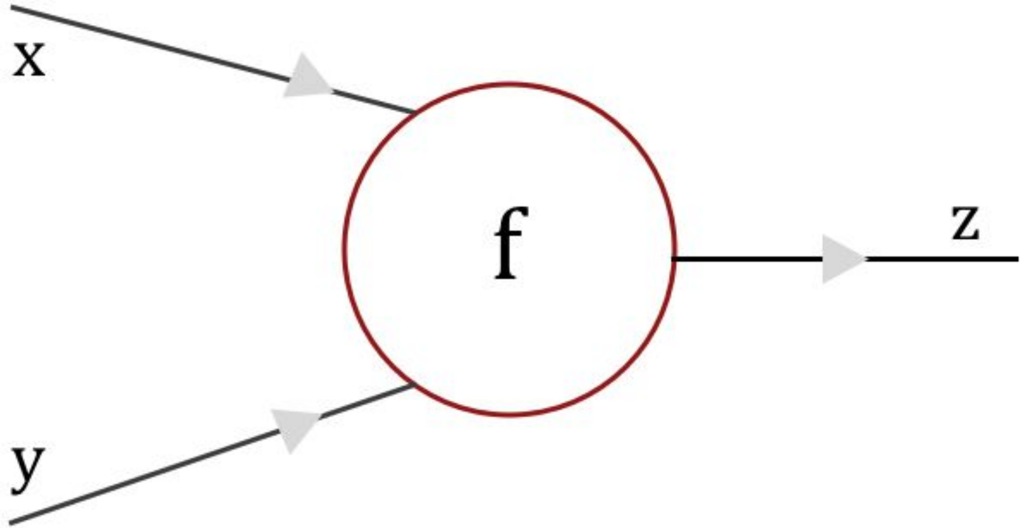
Backpropagation: Chain Rule Refresher



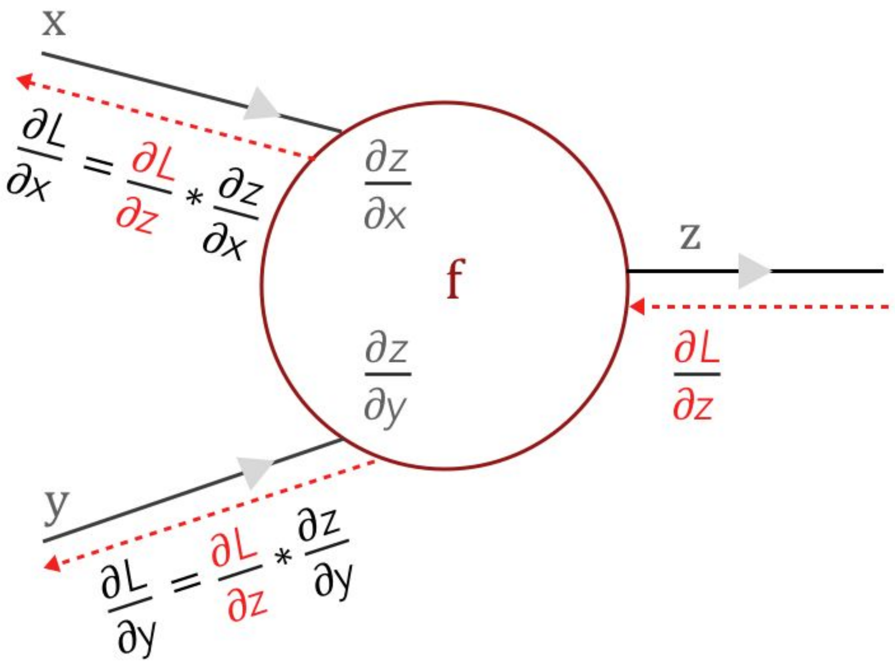
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial x} = -4 * 1 = -4$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} * \frac{\partial q}{\partial y} = -4 * 1 = -4$$

Backpropagation: Chain Rule in Convolutional Layer



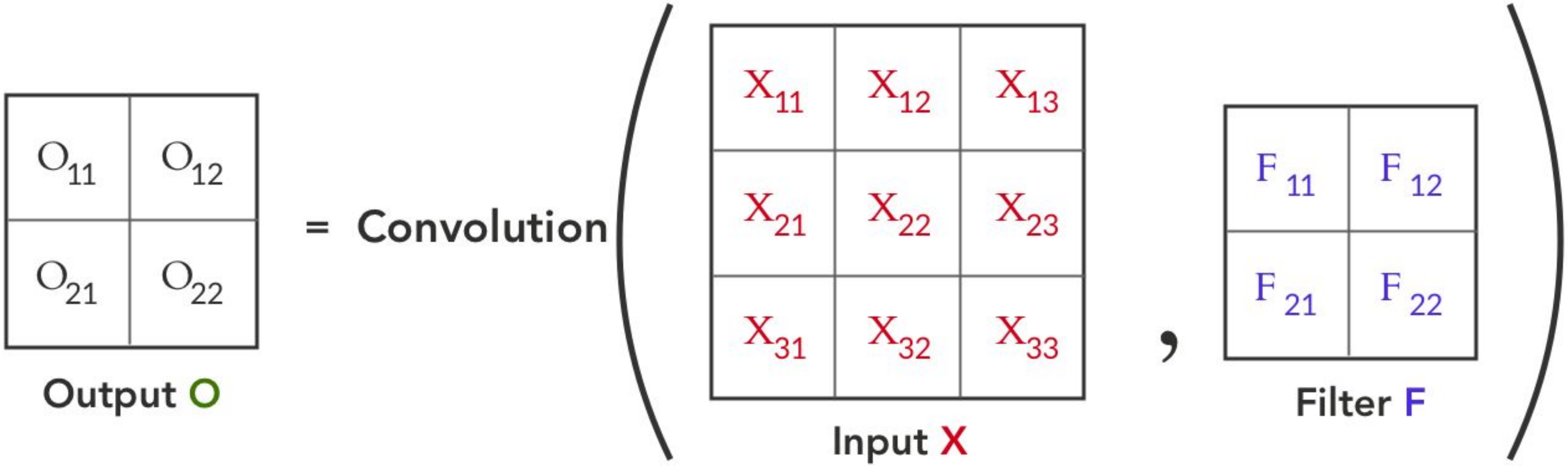
Backpropagation: Chain Rule in Convolutional Layer



$\frac{\partial z}{\partial x}$ & $\frac{\partial z}{\partial y}$ *are local gradients*

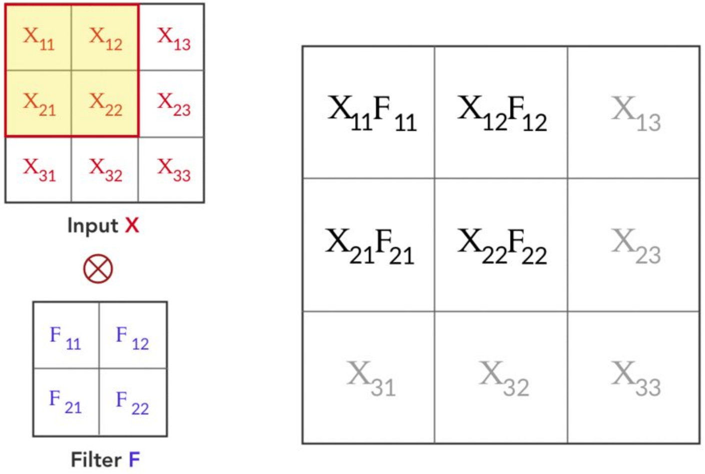
$\frac{\partial L}{\partial z}$ *is the loss from the previous layer which has to be backpropagated to other layers*

Backpropagation: Chain Rule in Convolutional Layer

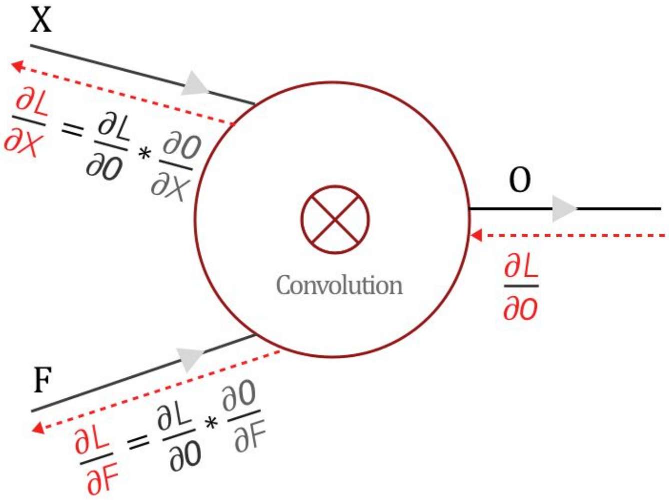


*This slide contains an animation, so it might not show up in the pdf file

Backpropagation: Chain Rule in Convolutional Layer



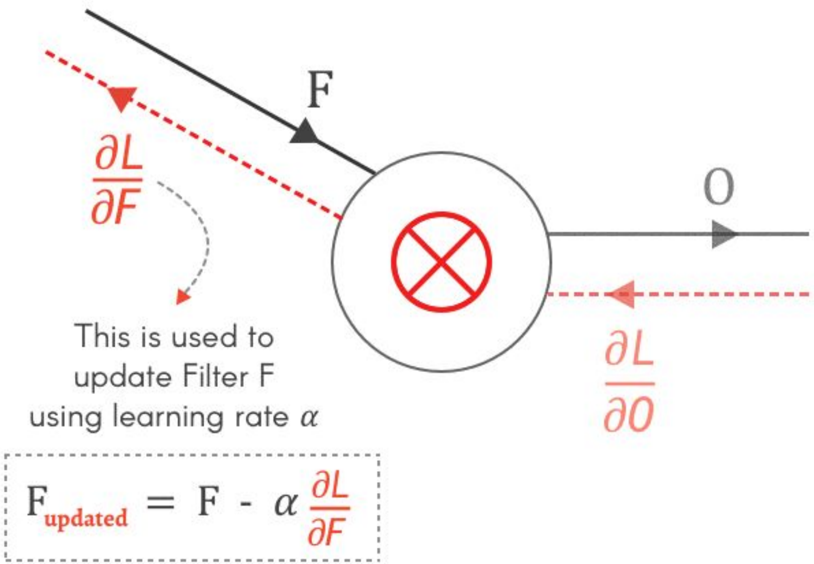
$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

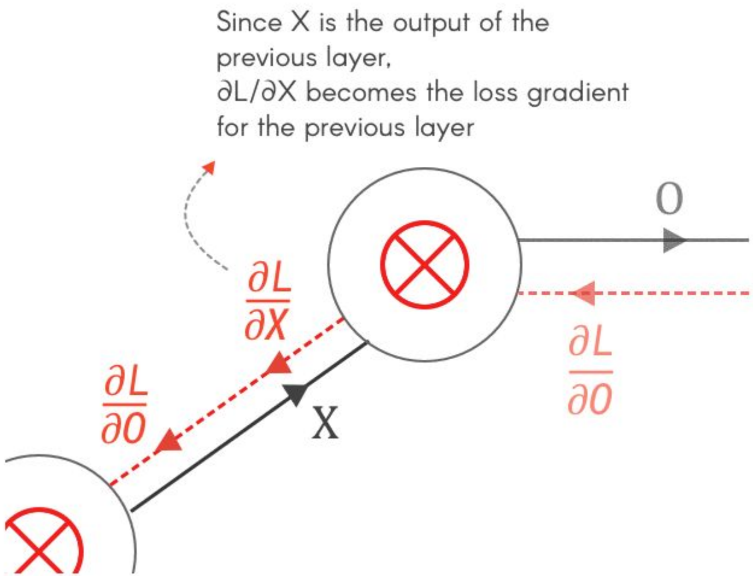


$\frac{\partial O}{\partial X}$ & $\frac{\partial O}{\partial F}$ *are local gradients*

$\frac{\partial L}{\partial z}$ *is the loss from the previous layer which has to be backpropagated to other layers*

Backpropagation: Finding Gradients for X and F





Backpropagation: Finding $\partial L/\partial F$

Step 1: Finding the local gradient - $\partial O/\partial F$

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Finding derivatives with respect to F_{11}, F_{12}, F_{21} and F_{22}

$$\frac{\partial O_{11}}{\partial F_{11}} = X_{11} \quad \frac{\partial O_{11}}{\partial F_{12}} = X_{12} \quad \frac{\partial O_{11}}{\partial F_{21}} = X_{21} \quad \frac{\partial O_{11}}{\partial F_{22}} = X_{22}$$

Similarly, we can find the local gradients for O_{12}, O_{21} and O_{22}

Backpropagation: Finding $\partial L/\partial F$

Step 2: Using the Chain rule

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial O} * \frac{\partial O}{\partial F}$$

Gradient to update Filter F

Loss Gradient from previous layer

Local Gradients

For every element of F

$$\frac{\partial L}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial F_i}$$

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{11}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{11}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{11}}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{12}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{12}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{12}}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{21}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{21}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{21}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{21}}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{22}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{22}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{22}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{22}}$$

Backpropagation: Finding $\partial L / \partial F$

Step 2: Using the Chain rule

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial O} * \frac{\partial O}{\partial F}$$

Gradient to update Filter F

Loss Gradient from previous layer

Local Gradients

For every element of F

$$\frac{\partial L}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial F_i}$$

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

Backpropagation: Finding *∂L/∂F*

∂L/∂F is nothing but the convolution between Input X and Loss Gradient from the next layer ∂L/∂O

$$\begin{bmatrix} \frac{\partial L}{\partial F_{11}} & \frac{\partial L}{\partial F_{12}} \\ \frac{\partial L}{\partial F_{21}} & \frac{\partial L}{\partial F_{22}} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} , \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} \right)$$

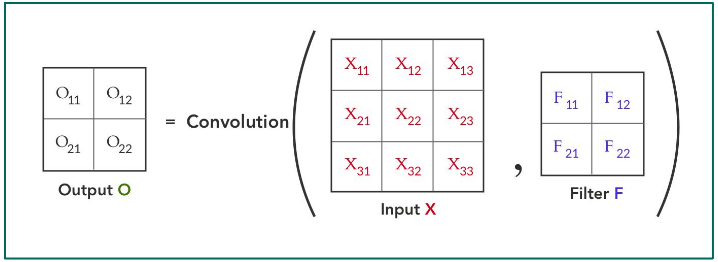
where

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} = \text{Input X}$$

$$\begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} = \frac{\partial L}{\partial O}$$

Loss gradient from previous layer

Backpropagation: Finding $\partial L / \partial O$



Step 1: Finding the local gradient - $\boldsymbol{\partial O / \partial X}$

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Differentiating with respect to X_{11}, X_{12}, X_{21} *and* X_{22}

$$\frac{\partial O_{11}}{\partial X_{11}} = F_{11} \quad \frac{\partial O_{11}}{\partial X_{12}} = F_{12} \quad \frac{\partial O_{11}}{\partial X_{21}} = F_{21} \quad \frac{\partial O_{11}}{\partial X_{22}} = F_{22}$$

Similarly, we can find local gradients for O_{12}, O_{21} *and* O_{22}

Backpropagation: Finding $\partial L/\partial O$

$$\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \right)$$

Output O — Input X — Filter F

Step 2: Using the Chain rule

For every element of X_i

$$\frac{\partial L}{\partial X_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial X_i}$$

$$\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial O_{11}} * F_{11}$$

$$\frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial O_{11}} * F_{12} + \frac{\partial L}{\partial O_{12}} * F_{11}$$

$$\frac{\partial L}{\partial X_{13}} = \frac{\partial L}{\partial O_{12}} * F_{12}$$

$$\frac{\partial L}{\partial X_{21}} = \frac{\partial L}{\partial O_{11}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{11}$$

$$\frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial O_{11}} * F_{22} + \frac{\partial L}{\partial O_{12}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{12} + \frac{\partial L}{\partial O_{22}} * F_{11}$$

$$\frac{\partial L}{\partial X_{23}} = \frac{\partial L}{\partial O_{12}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{12}$$

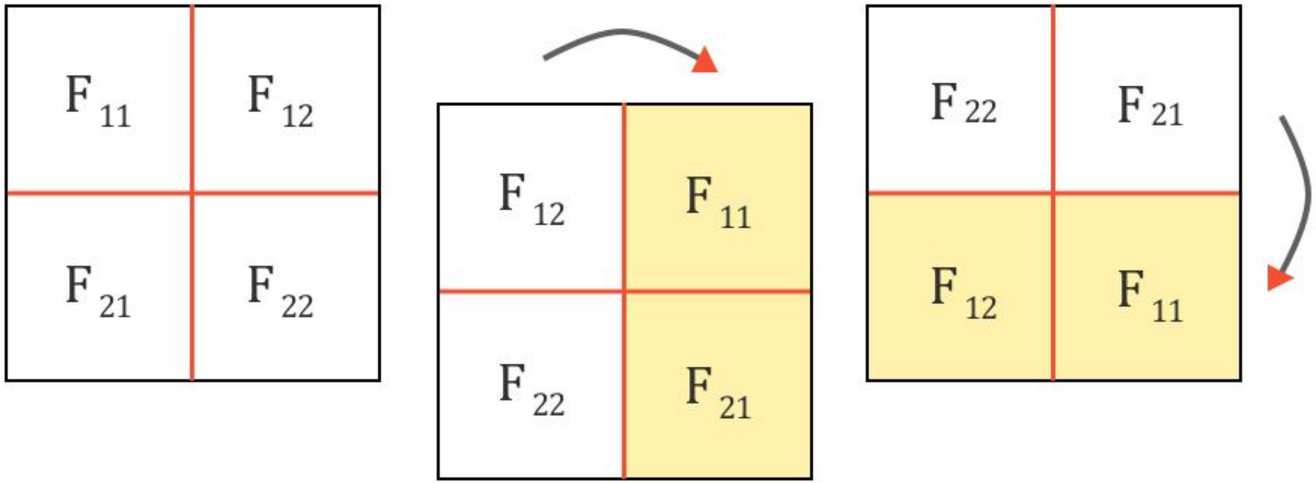
$$\frac{\partial L}{\partial X_{31}} = \frac{\partial L}{\partial O_{21}} * F_{21}$$

$$\frac{\partial L}{\partial X_{32}} = \frac{\partial L}{\partial O_{21}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{21}$$

$$\frac{\partial L}{\partial X_{33}} = \frac{\partial L}{\partial O_{22}} * F_{22}$$

Backpropagation: $\partial L / \partial X$ as a 'Full Convolution'

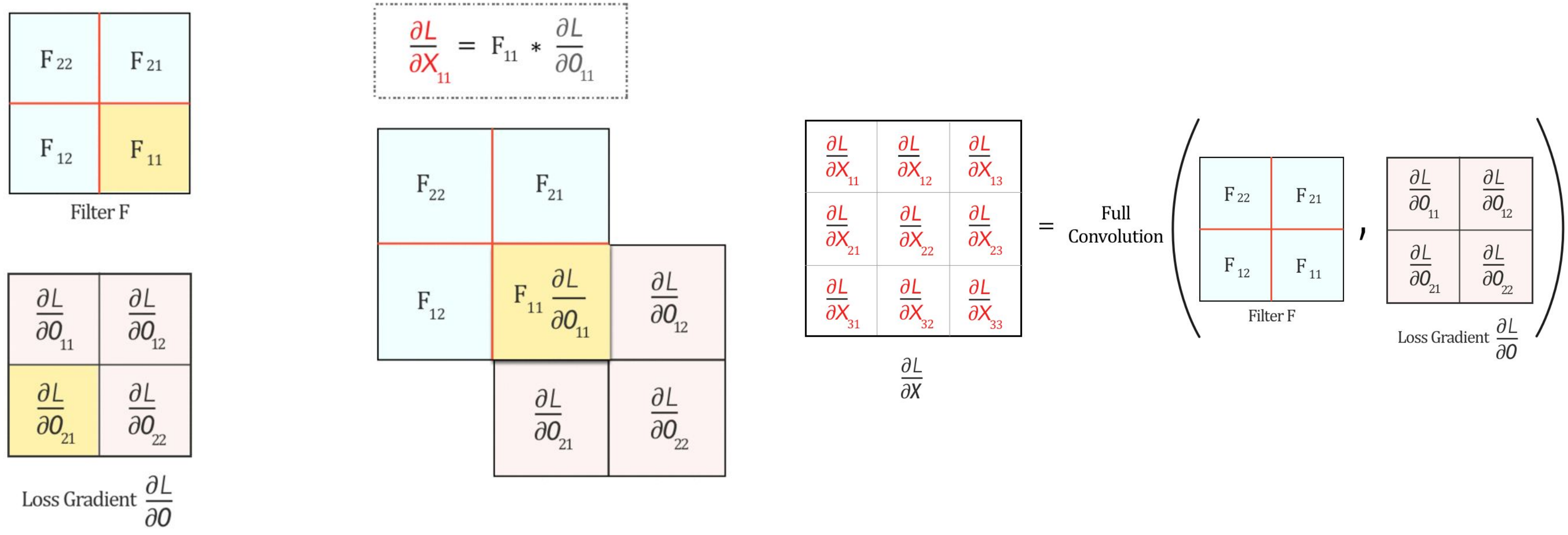
Step 1: Rotate the Filter **F** by 180 degrees - flipping it first vertically and then horizontally



*This slide contains an animation, so it might not show up in the pdf file

Backpropagation: $\partial L/\partial X$ as a ‘Full Convolution’

Step 2: Full convolution between flipped filter F and $\partial L/\partial O$



Backpropagation: Conclusion

Backpropagation in a Convolutional Layer of a CNN

Finding the gradients:

$$\frac{\partial L}{\partial F} = \text{Convolution}\left(\text{Input } X, \text{ Loss gradient } \frac{\partial L}{\partial O}\right)$$

$$\frac{\partial L}{\partial X} = \begin{matrix}\text{Full}\\ \text{Convolution}\end{matrix}\left(\begin{matrix}180^{\circ}\text{rotated}\\ \text{Filter } F\end{matrix}, \begin{matrix}\text{Loss}\\ \text{Gradient}\end{matrix}\ \frac{\partial L}{\partial O}\right)$$