Convolutional Neural Networks

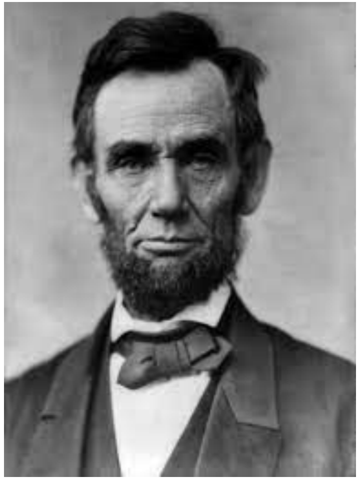
Forward Pass

Introduction to Deep Learning - 11-485/685/785 - Fall 2022

Abuzar Khan

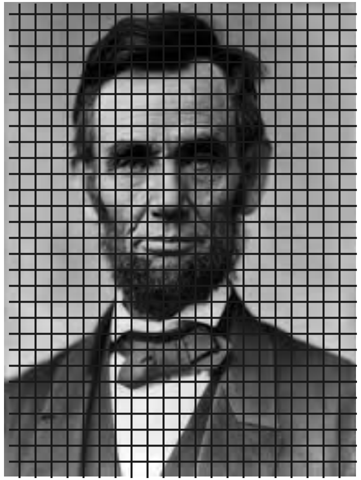
What is an image?

A thousand words.



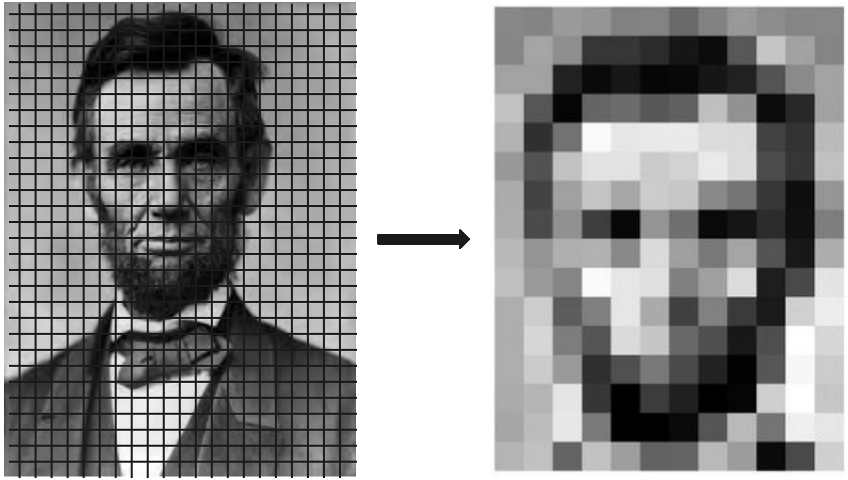
What is an image?

A thousand words.



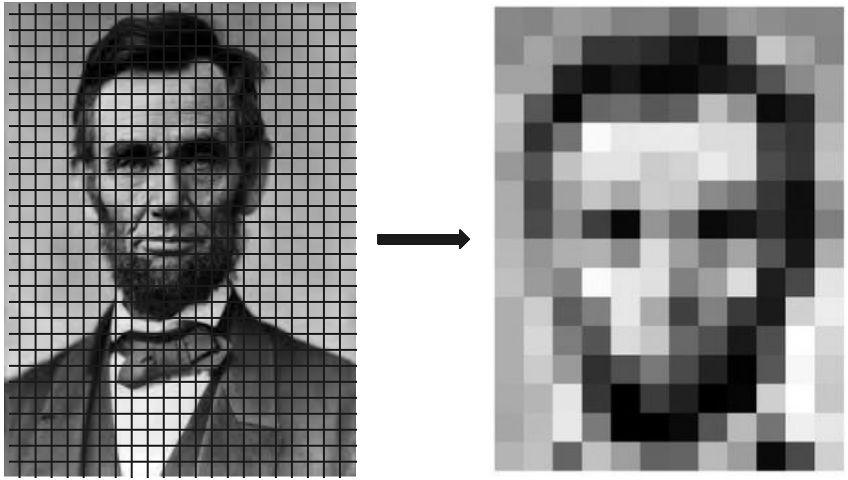
What is an image?

A thousand words.



What is an image?

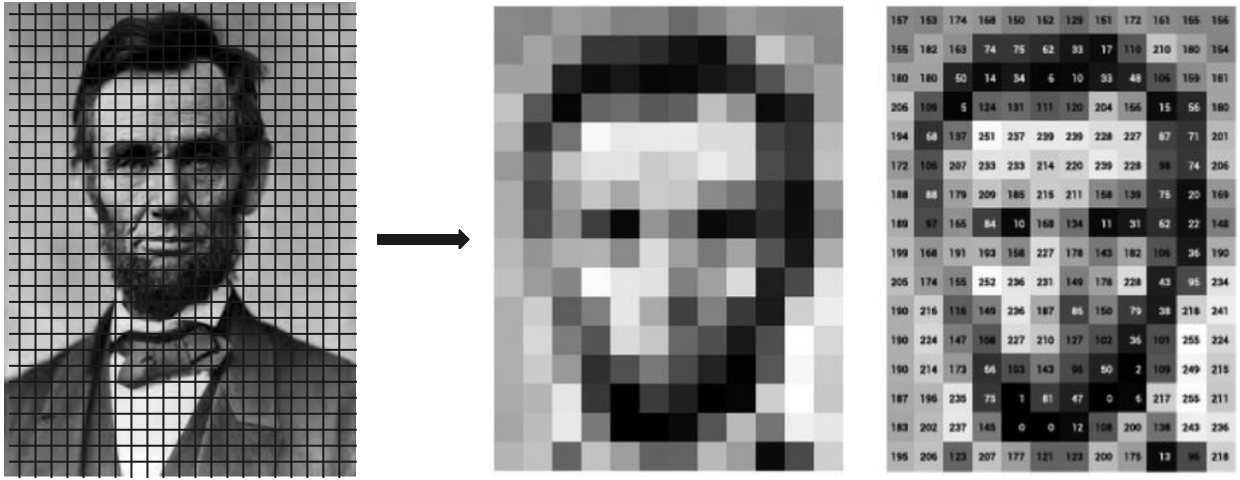
A thousand words.



157	153	174	168	150	162	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	85	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?

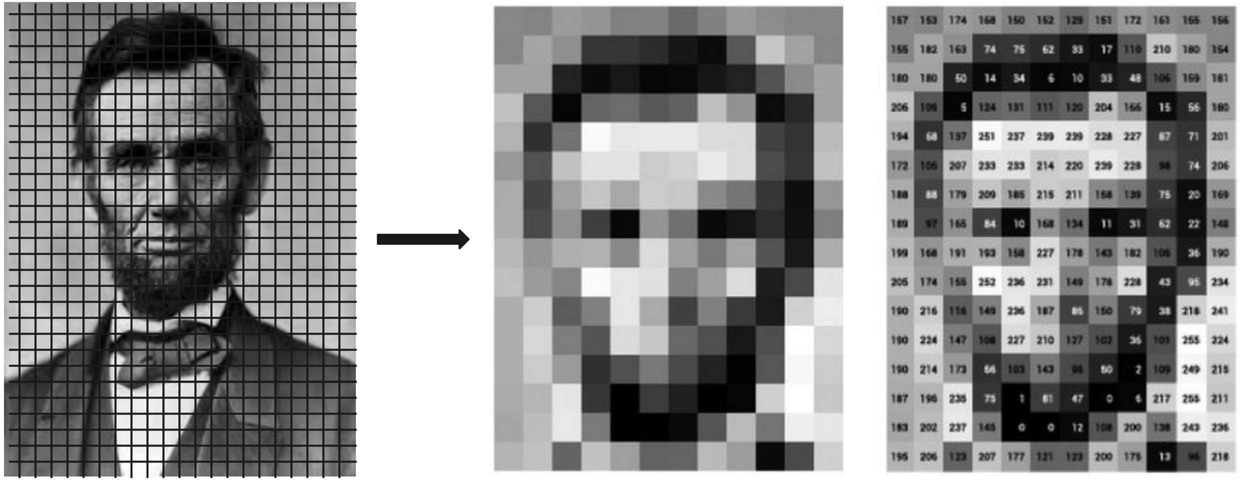
A thousand words.



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?

~~A thousand words.~~ 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

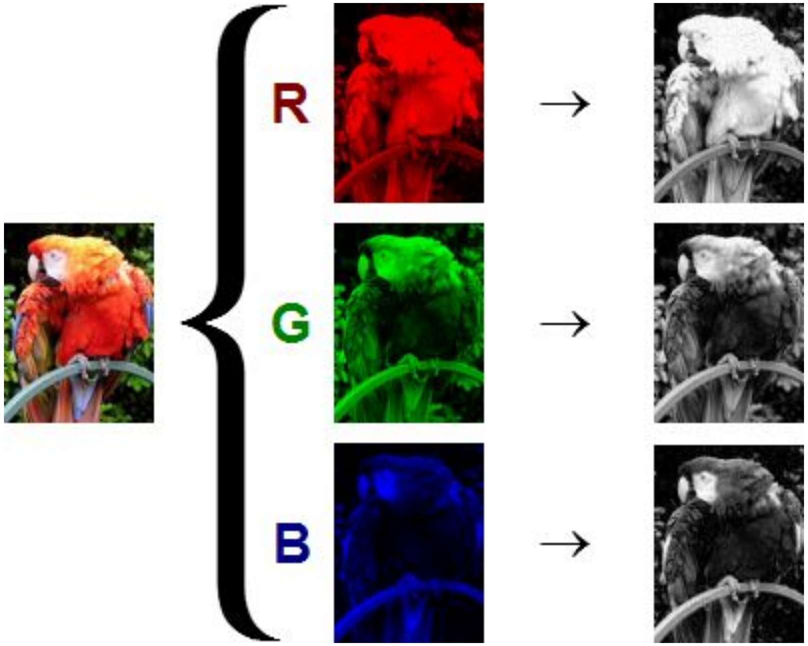
Beyond B/W

Colored Images:

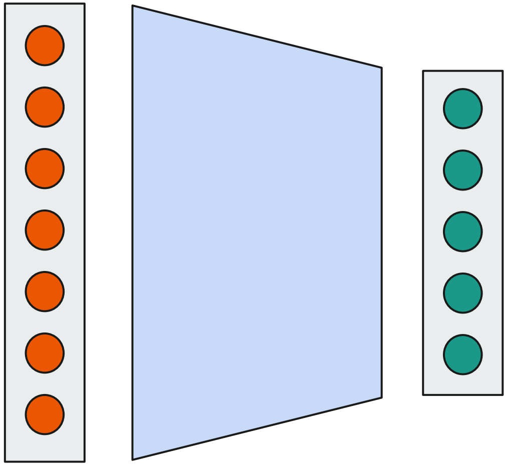
- Multi-channel
- R,G,B (an example)
- `I → (3,M,N)`

  `I[c][i][j] =`

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

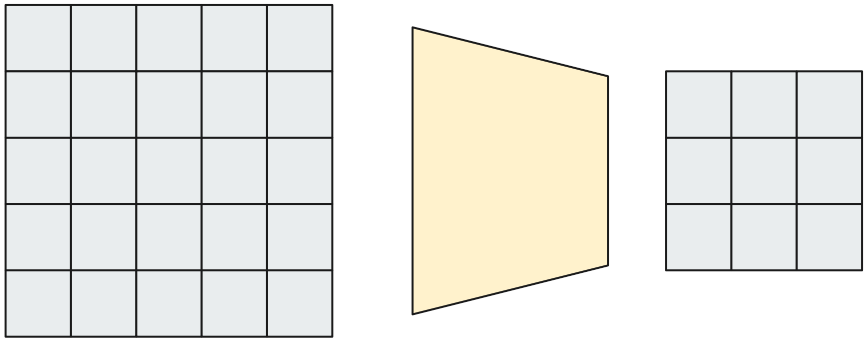


MLP



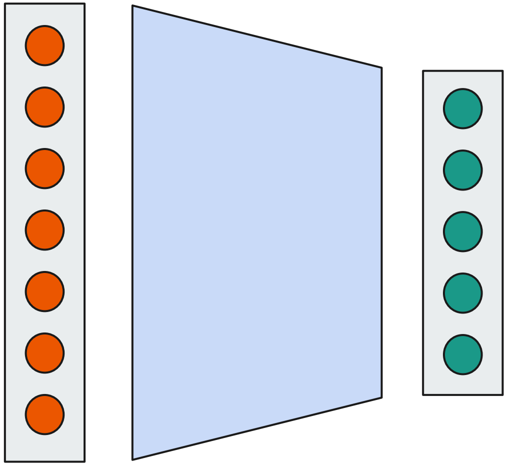
Vector to Vector

CNN

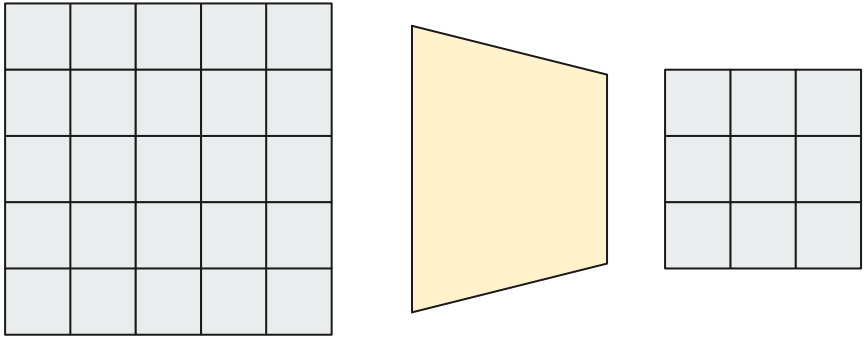


Feature map to Feature map

MLP Vs. CNN

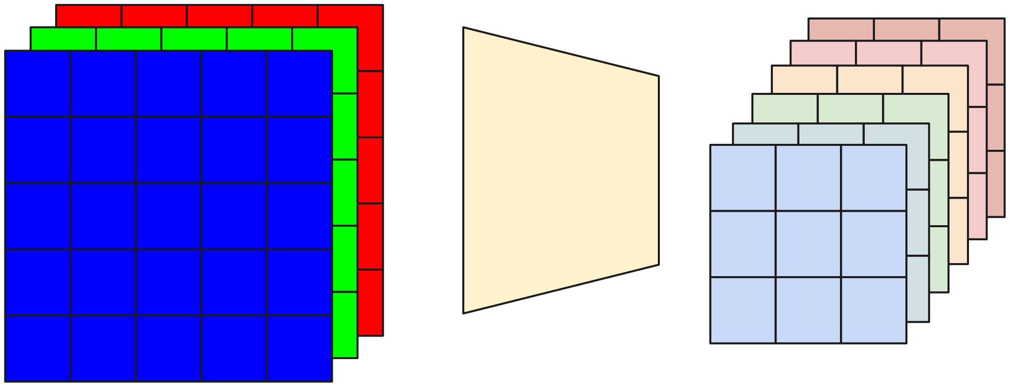


Vector to Vector



Feature map to Feature map

Multi-channel CNN



Components of a CNN

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix}$$

Input - **A**

$$\begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix}$$

Kernel - **W**

$$\begin{bmatrix} B_{1,1} \end{bmatrix}$$

Bias - **B**

Components of a CNN

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix}$$

Input - **A**

$$\begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix}$$

Kernel - **W**

$$\begin{bmatrix} B_{1,1} \end{bmatrix}$$

Bias - **B**

$$\begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} \\ Z_{2,1} & Z_{2,2} & Z_{2,3} \\ Z_{3,1} & Z_{3,2} & Z_{3,3} \end{bmatrix}$$

Output - **Z**

$$\mathbf{Z} = (\mathbf{A} \otimes \mathbf{W}) + \mathbf{B}$$

CNN Steps

Essentially element-wise (Hadamard) multiplications and summations

CNN Steps

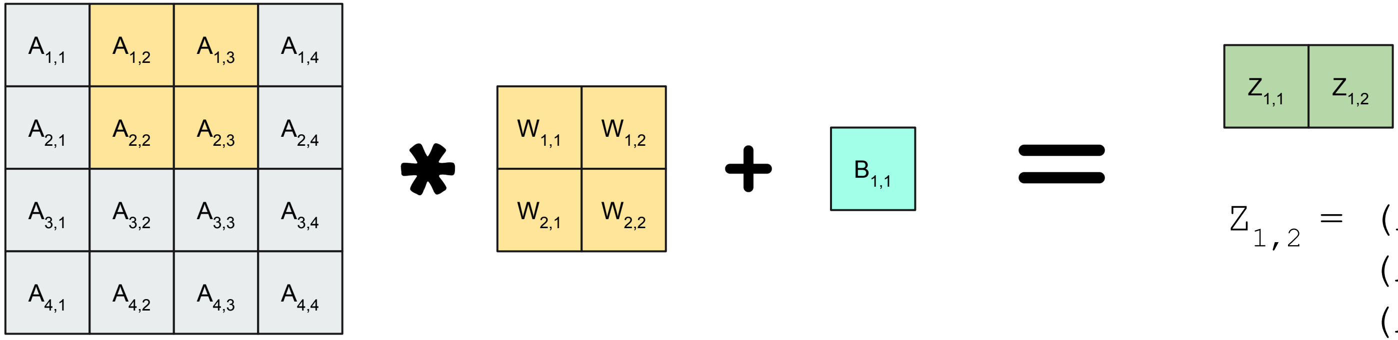
Essentially element-wise (Hadamard) multiplications and summations

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} * \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \end{bmatrix} = \begin{bmatrix} Z_{1,1} \end{bmatrix}$$

$$\begin{aligned} Z_{1,1} = \; & (A_{1,1} * W_{1,1}) + \\ & (A_{1,2} * W_{1,2}) + \\ & (A_{2,1} * W_{2,1}) + \\ & (A_{2,2} * W_{2,2}) + \\ & B \end{aligned}$$

CNN Steps

Essentially element-wise (Hadamard) multiplications and summations



$$
\begin{aligned}
Z_{1,2} = \; & (A_{1,2} * W_{1,1}) \; + \\
& (A_{1,3} * W_{1,2}) \; + \\
& (A_{2,2} * W_{2,1}) \; + \\
& (A_{2,3} * W_{2,2}) \; + \\
& B
\end{aligned}
$$

CNN Steps

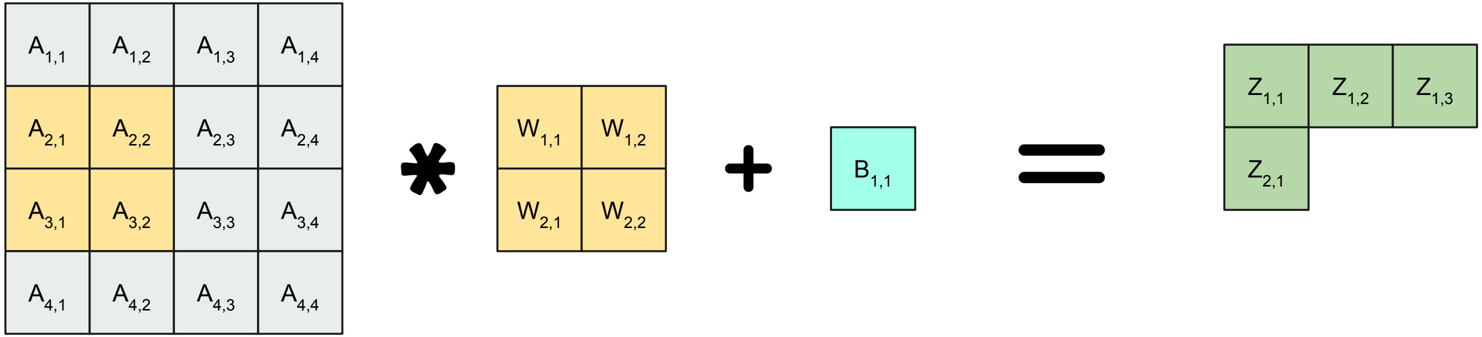
Essentially element-wise (Hadamard) multiplications and summations

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} * \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \end{bmatrix} = \begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} \end{bmatrix}$$

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

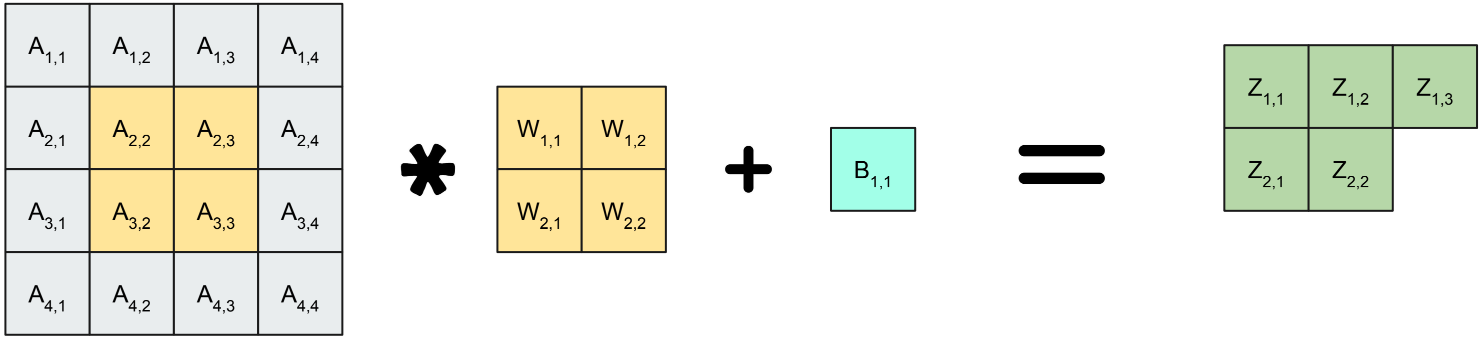
CNN Steps

Essentially element-wise (Hadamard) multiplications and summations



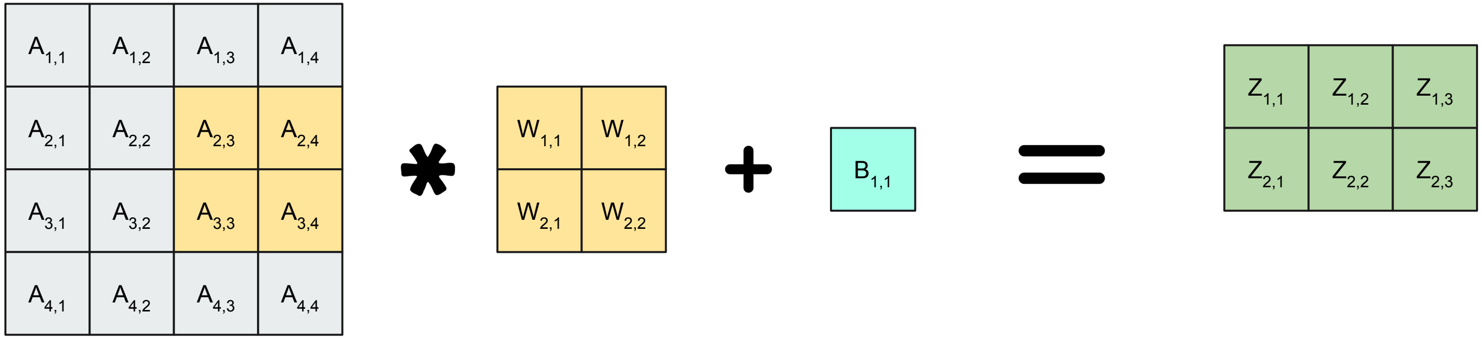
CNN Steps

Essentially element-wise (Hadamard) multiplications and summations



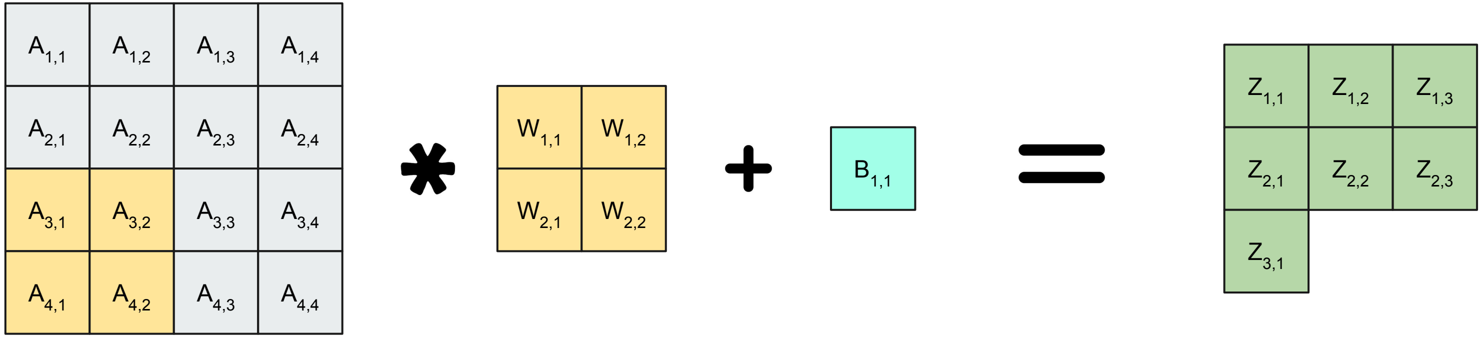
CNN Steps

Essentially element-wise (Hadamard) multiplications and summations



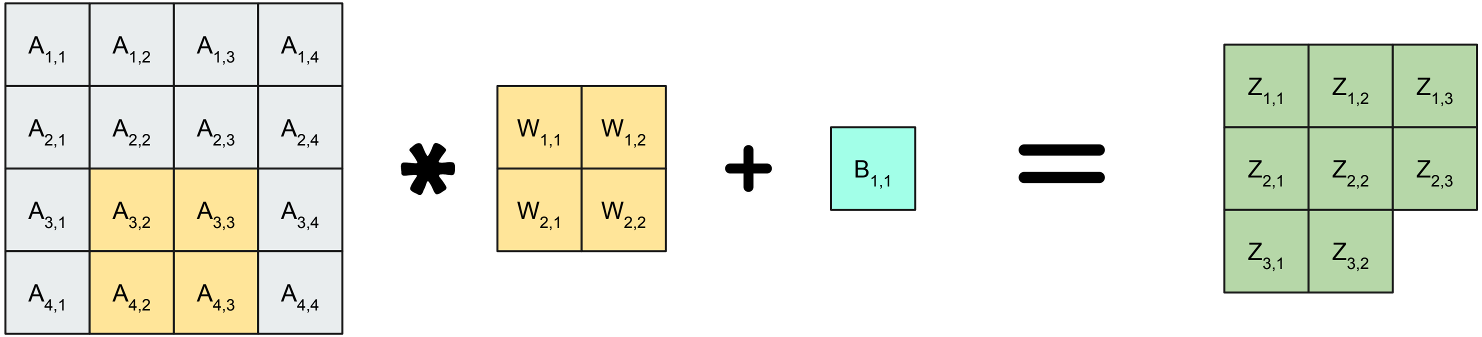
CNN Steps

Essentially element-wise (Hadamard) multiplications and summations



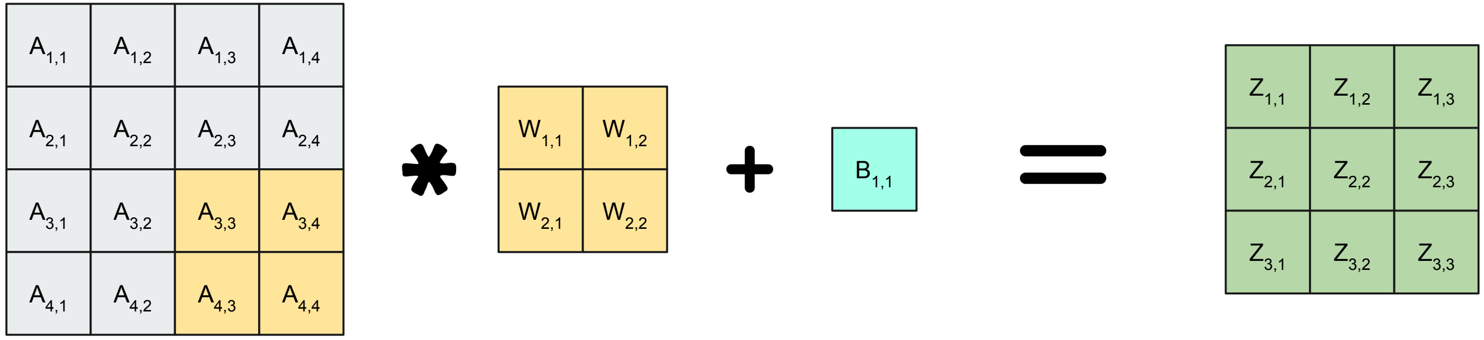
CNN Steps

Essentially element-wise (Hadamard) multiplications and summations



CNN Steps

Essentially element-wise (Hadamard) multiplications and summations



Output Size

$$
\begin{bmatrix}
A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\
A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\
A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4}
\end{bmatrix}
\longrightarrow
\begin{bmatrix}
Z_{1,1} & Z_{1,2} & Z_{1,3} \\
Z_{2,1} & Z_{2,2} & Z_{2,3} \\
Z_{3,1} & Z_{3,2} & Z_{3,3}
\end{bmatrix}
$$

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

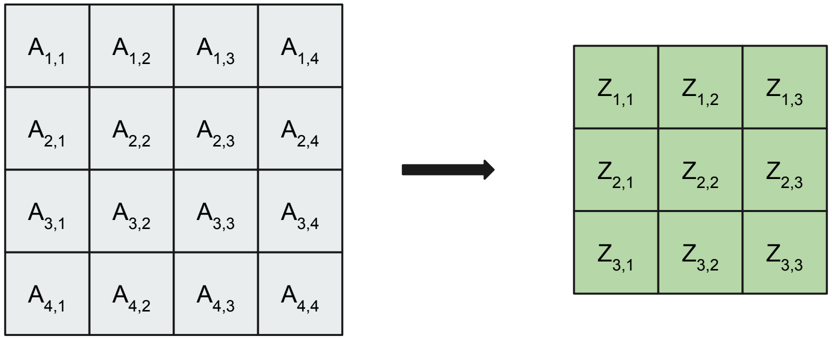
Same goes for Height.

Output Size

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} \rightarrow \begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} \\ Z_{2,1} & Z_{2,2} & Z_{2,3} \\ Z_{3,1} & Z_{3,2} & Z_{3,3} \end{bmatrix}$$

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

Output Size



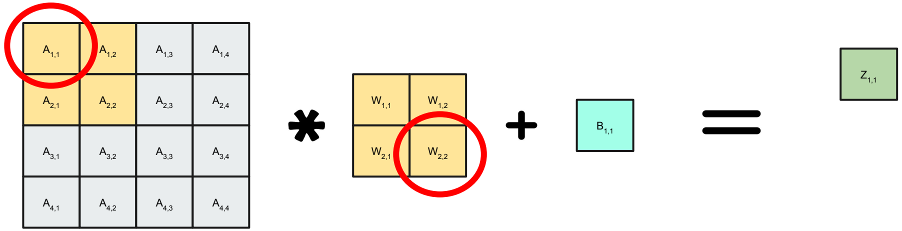
Output Width = $[(W_{in} - W_k + 2\mathbf{P})\ //\ (\mathbf{S})] + 1$

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

Padding

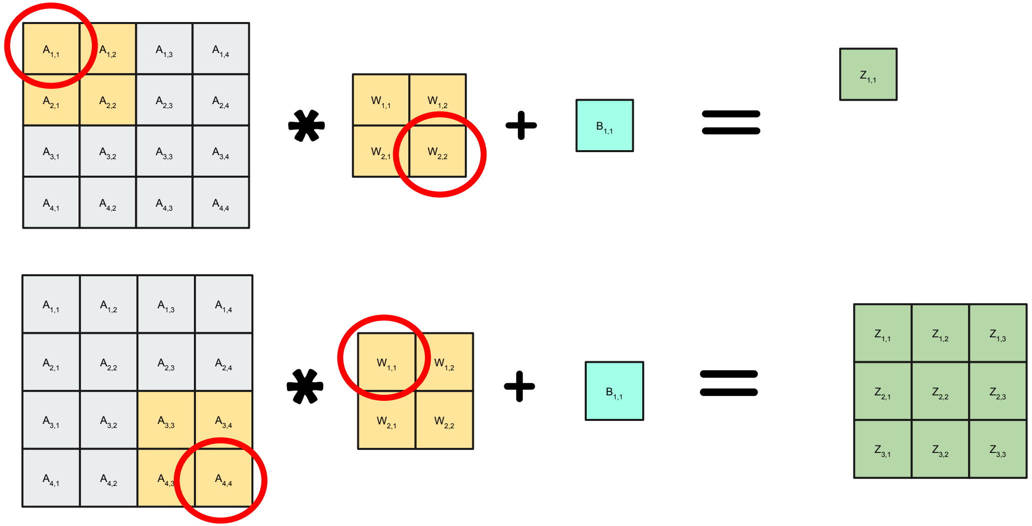
- Attaching zeros (usually) around inputs.
- Seen it before in HW1.
- Images can be padded to the left, right, top, and bottom.

Padding



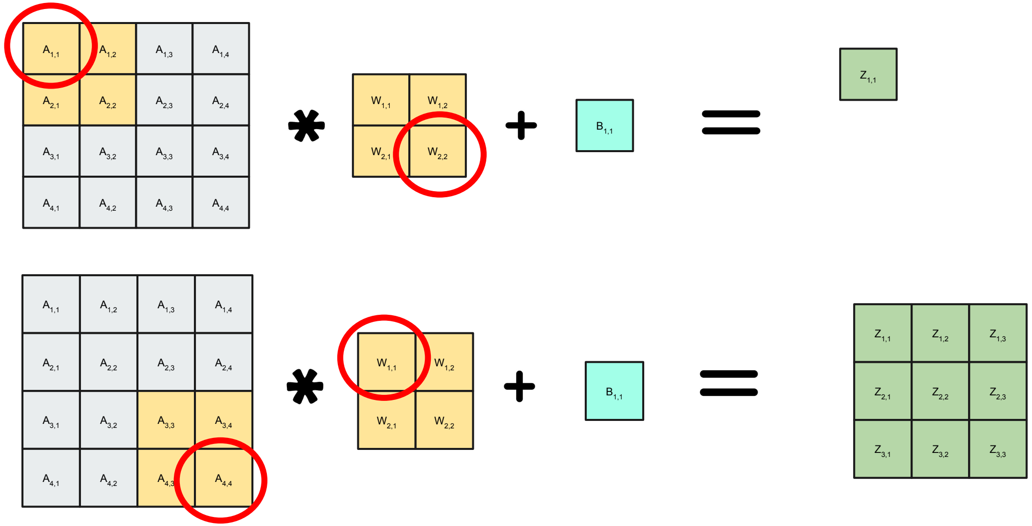
$$
\begin{aligned}
Z_{1,1} = \; & (A_{1,1} * W_{1,1}) \; + \\
& (A_{1,2} * W_{1,2}) \; + \\
& (A_{2,1} * W_{2,1}) \; + \\
& (A_{2,2} * W_{2,2}) \; + \\
& B
\end{aligned}
$$

Padding



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

Padding



Never Meet...

Padding

Increase output size

Preserve input size

More Kernel Interactions!

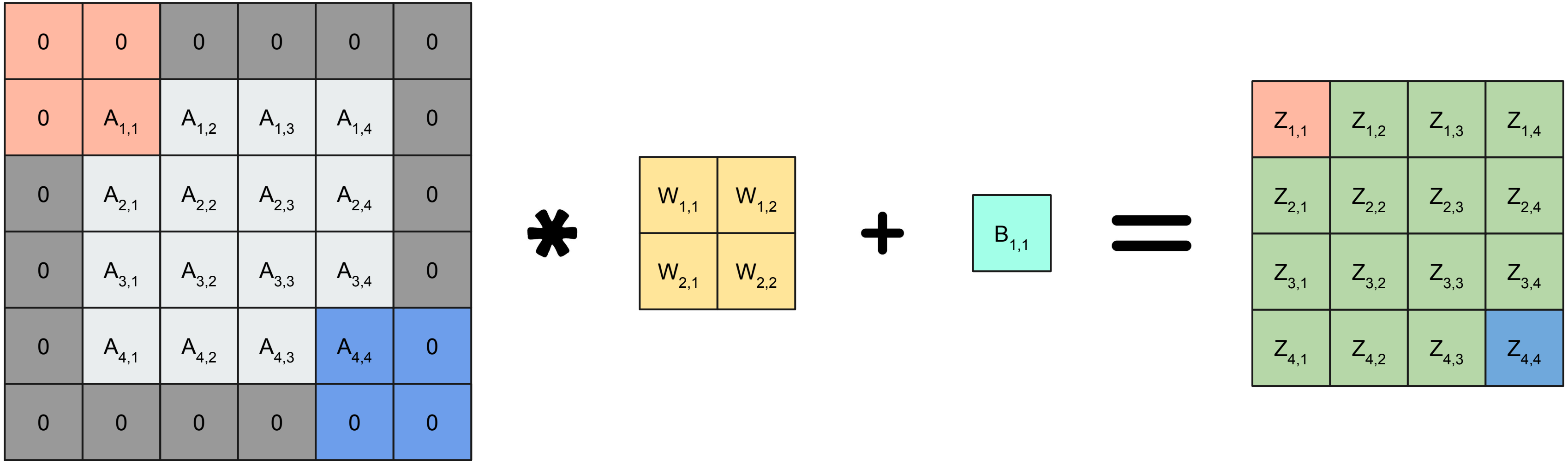
Padding

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} * \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \end{bmatrix} = \begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} \\ Z_{2,1} & Z_{2,2} & Z_{2,3} \\ Z_{3,1} & Z_{3,2} & Z_{3,3} \end{bmatrix}$$

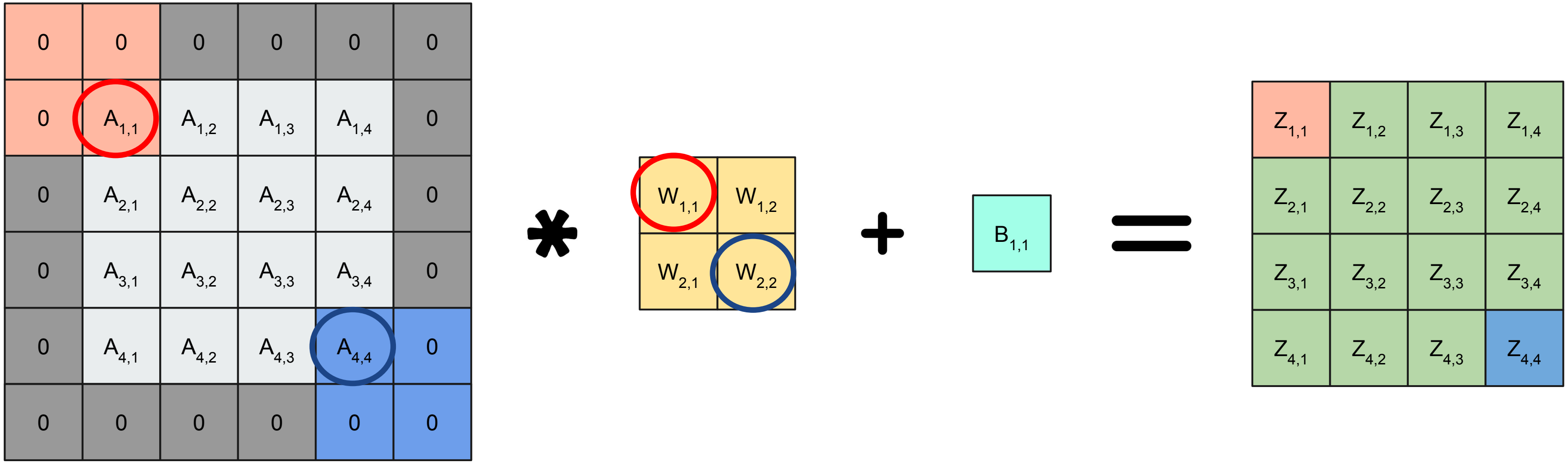
Padding

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & 0 \\ 0 & A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} & 0 \\ 0 & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} & 0 \\ 0 & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \end{bmatrix} = \begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} & Z_{1,4} \\ Z_{2,1} & Z_{2,2} & Z_{2,3} & Z_{2,4} \\ Z_{3,1} & Z_{3,2} & Z_{3,3} & Z_{3,4} \\ Z_{4,1} & Z_{4,2} & Z_{4,3} & Z_{4,4} \end{bmatrix}$$

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 $A_{1,1}$ $A_{1,2}$ $A_{1,3}$ $A_{1,4}$ 0
0 $A_{2,1}$ $A_{2,2}$ $A_{2,3}$ $A_{2,4}$ 0
0 $A_{3,1}$ $A_{3,2}$ $A_{3,3}$ $A_{3,4}$ 0
0 $A_{4,1}$ $A_{4,2}$ $A_{4,3}$ $A_{4,4}$ 0
0 0 0 0 0 0
*
$W_{1,1}$ $W_{1,2}$
$W_{2,1}$ $W_{2,2}$
+
$B_{1,1}$
=
$Z_{1,1}$ $Z_{1,2}$ $Z_{1,3}$ $Z_{1,4}$
$Z_{2,1}$ $Z_{2,2}$ $Z_{2,3}$ $Z_{2,4}$
$Z_{3,1}$ $Z_{3,2}$ $Z_{3,3}$ $Z_{3,4}$
$Z_{4,1}$ $Z_{4,2}$ $Z_{4,3}$ $Z_{4,4}$

Stride

Taking bigger steps!

Stride = 1

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

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} * \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \end{bmatrix} = \begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} \\ Z_{2,1} & Z_{2,2} & Z_{2,3} \\ Z_{3,1} & Z_{3,2} & Z_{3,3} \end{bmatrix}$$

Stride = 2

Start at the same place

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} * \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \end{bmatrix} = \begin{bmatrix} Z_{1,1} \end{bmatrix}$$

$$\begin{aligned} Z_{1,1} = \; & (A_{1,1} * W_{1,1}) \; + \\ & (A_{1,2} * W_{1,2}) \; + \\ & (A_{2,1} * W_{2,1}) \; + \\ & (A_{2,2} * W_{2,2}) \; + \\ & B \end{aligned}$$

Stride = 2

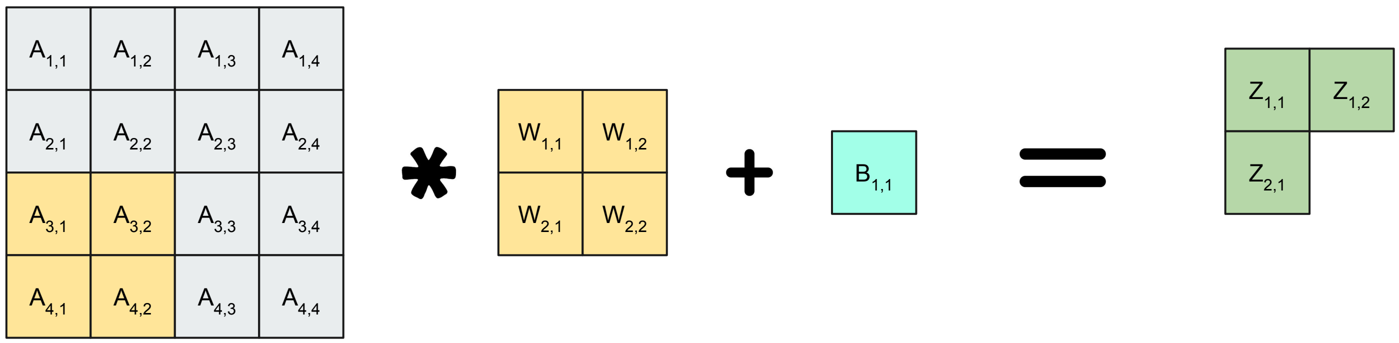
Move two elements to the right

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} * \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \end{bmatrix} = \begin{bmatrix} Z_{1,1} & Z_{1,2} \end{bmatrix}$$

$$\begin{aligned} Z_{1,2} = \; & (A_{1,3} * W_{1,1}) + \\ & (A_{1,4} * W_{1,2}) + \\ & (A_{2,3} * W_{2,1}) + \\ & (A_{2,4} * W_{2,2}) + \\ & B \end{aligned}$$

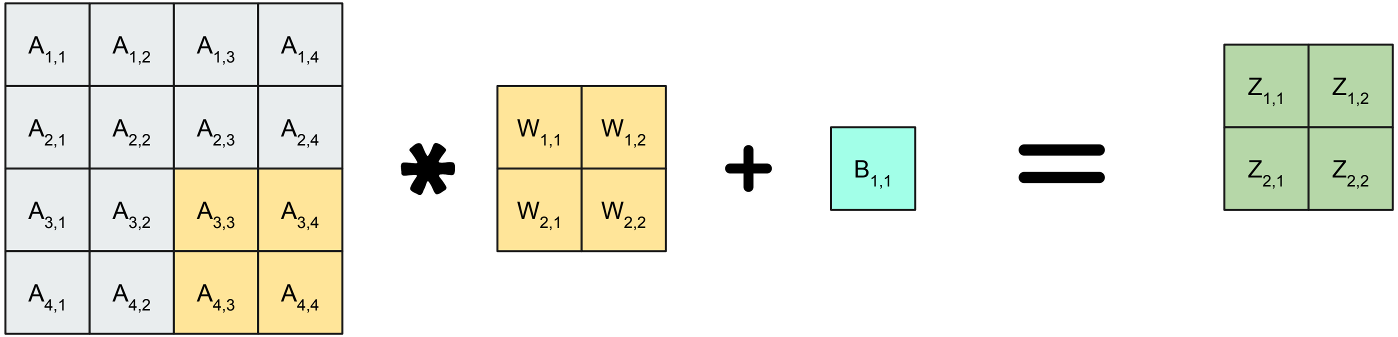
Stride = 2

Move two elements down.



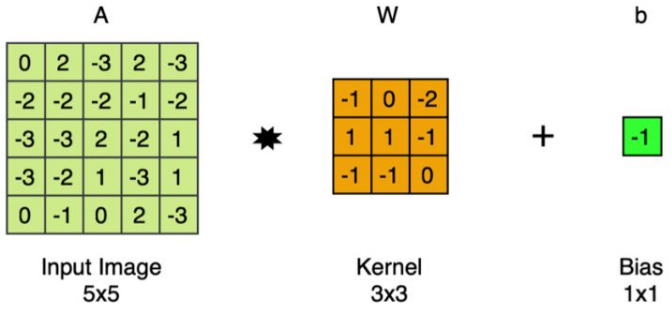
Stride = 2

Move two elements to the right.



[Courtesy Aparajit (Apa)]

Interpreting Stride > 1



[Courtesy Aparajit (Apa)]

Interpreting Stride > 1

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

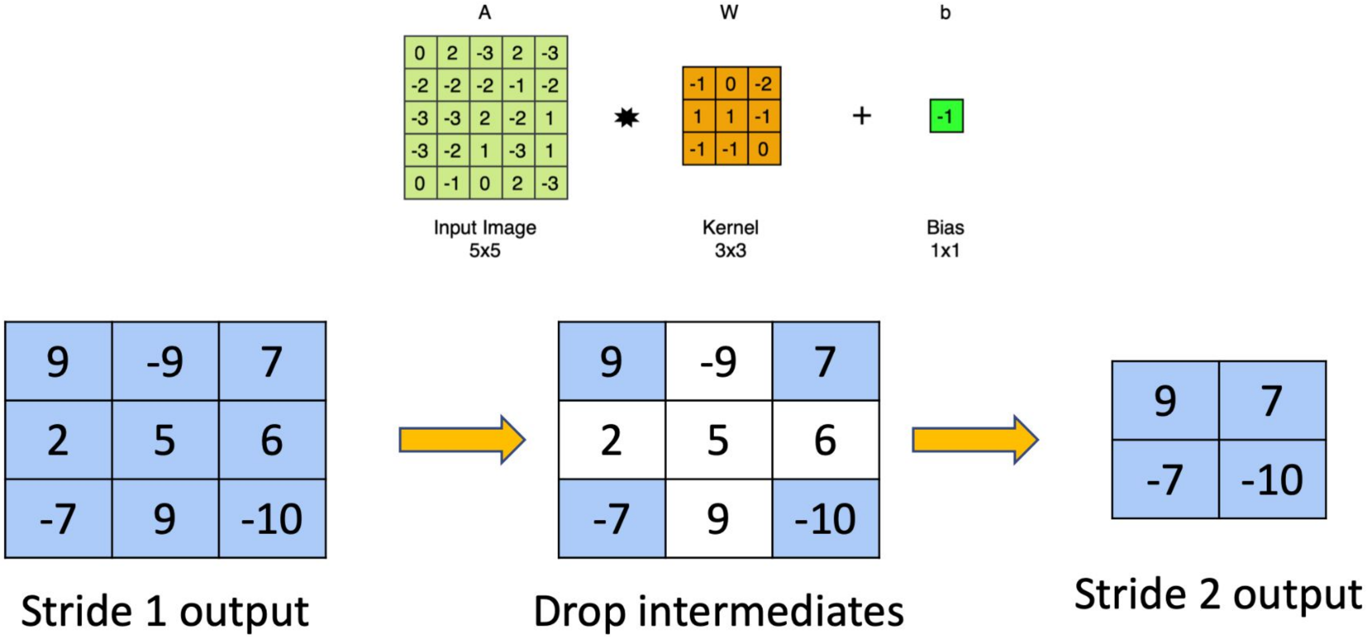
Stride 1 output

9	7
-7	-10

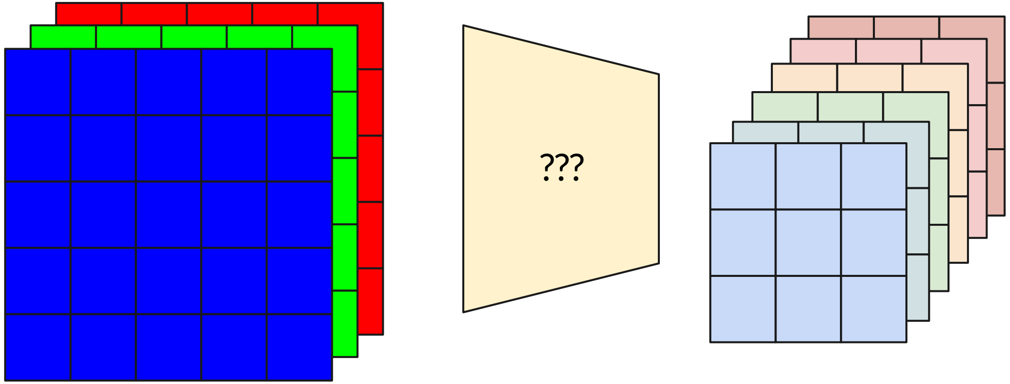
Stride 2 output

[Courtesy Aparajit (Apa)]

Interpreting Stride > 1



Multi-channel CNN



Multi-channel CNN

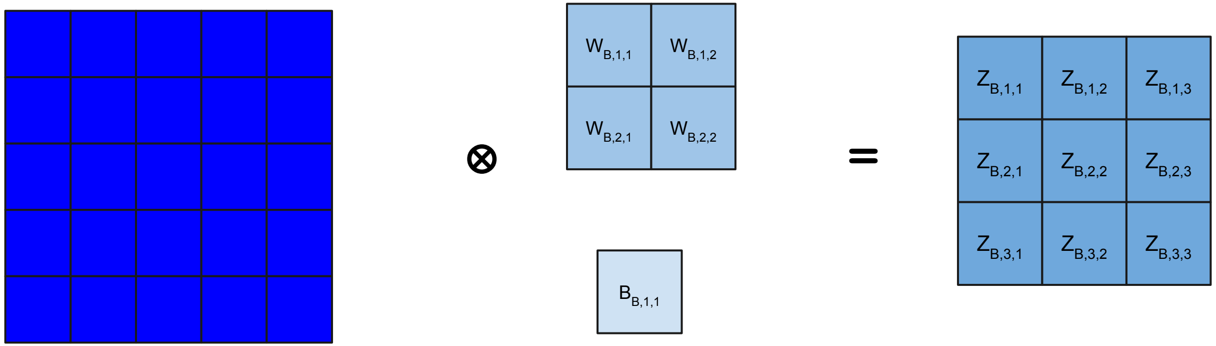
- Each kernel (or **filter**) has as many channels as the input does
- Channel **c of the kernel** convolves with channel **c** (corresponding) **of the input**.
- The number of output channels from the convolution = number of **filters**

C_{in} = Input channels

C_{kernel} = Kernel channels = C_{in}

K = Number of Kernels = C_{out} = Number of output channels

1 Filter with 3-channel input

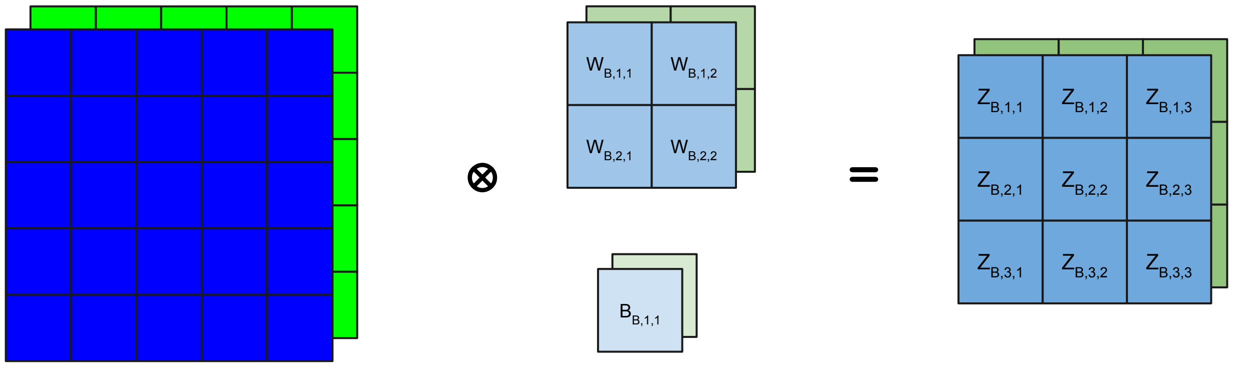


1 channel input

Kernel

1 almost-output map

1 Filter with 3-channel input

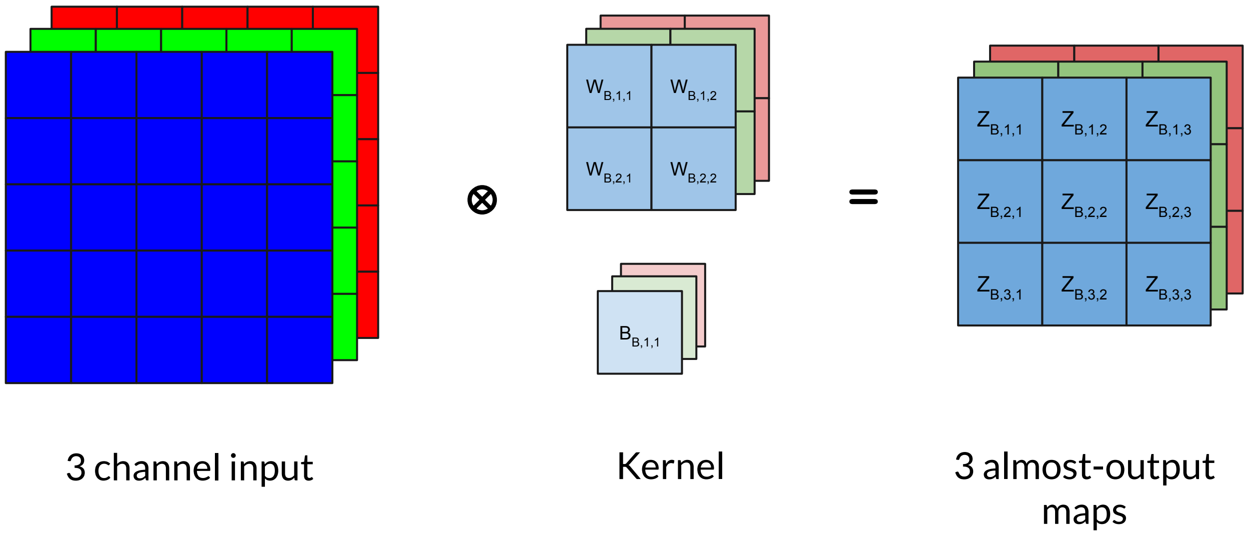


2 channel input

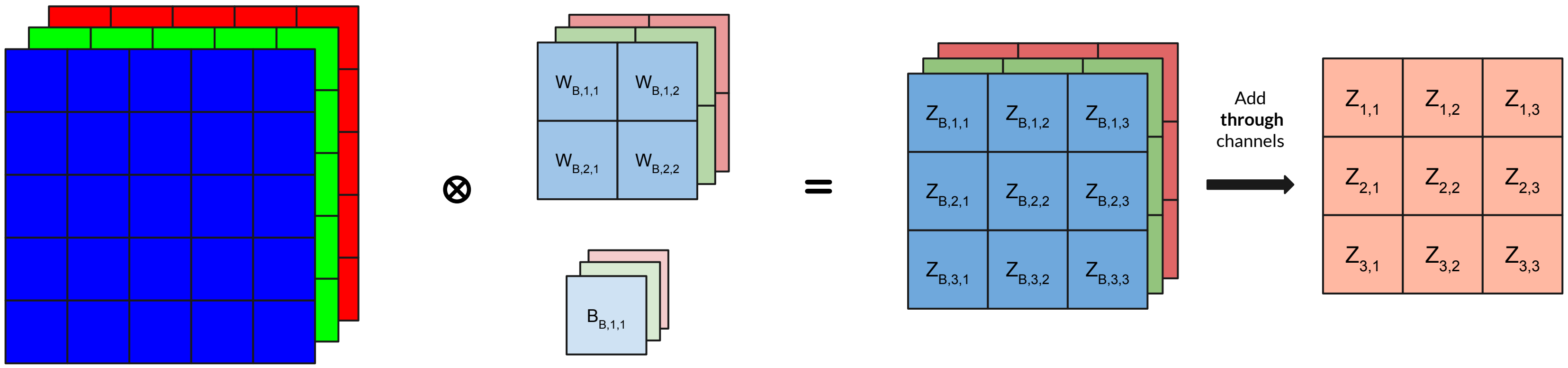
Kernel

2 almost-output maps

1 Filter with 3-channel input



1 Filter with 3-channel input



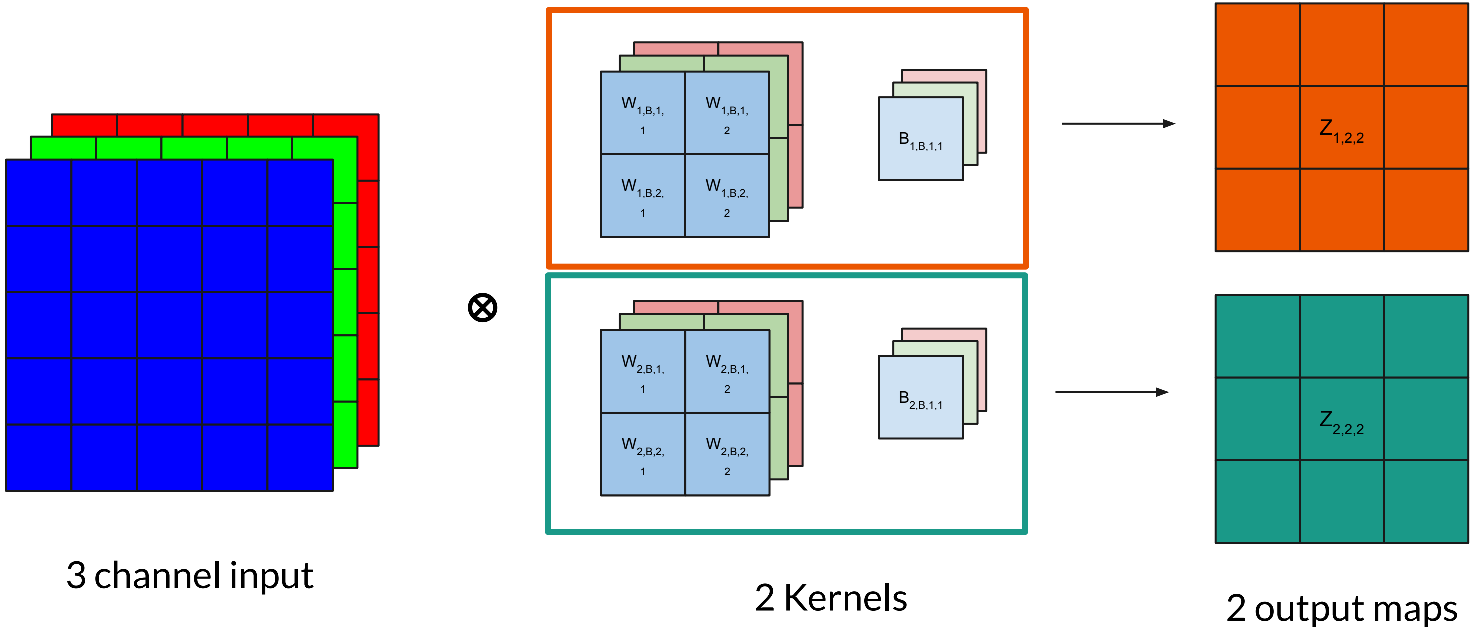
3 channel input

Kernel

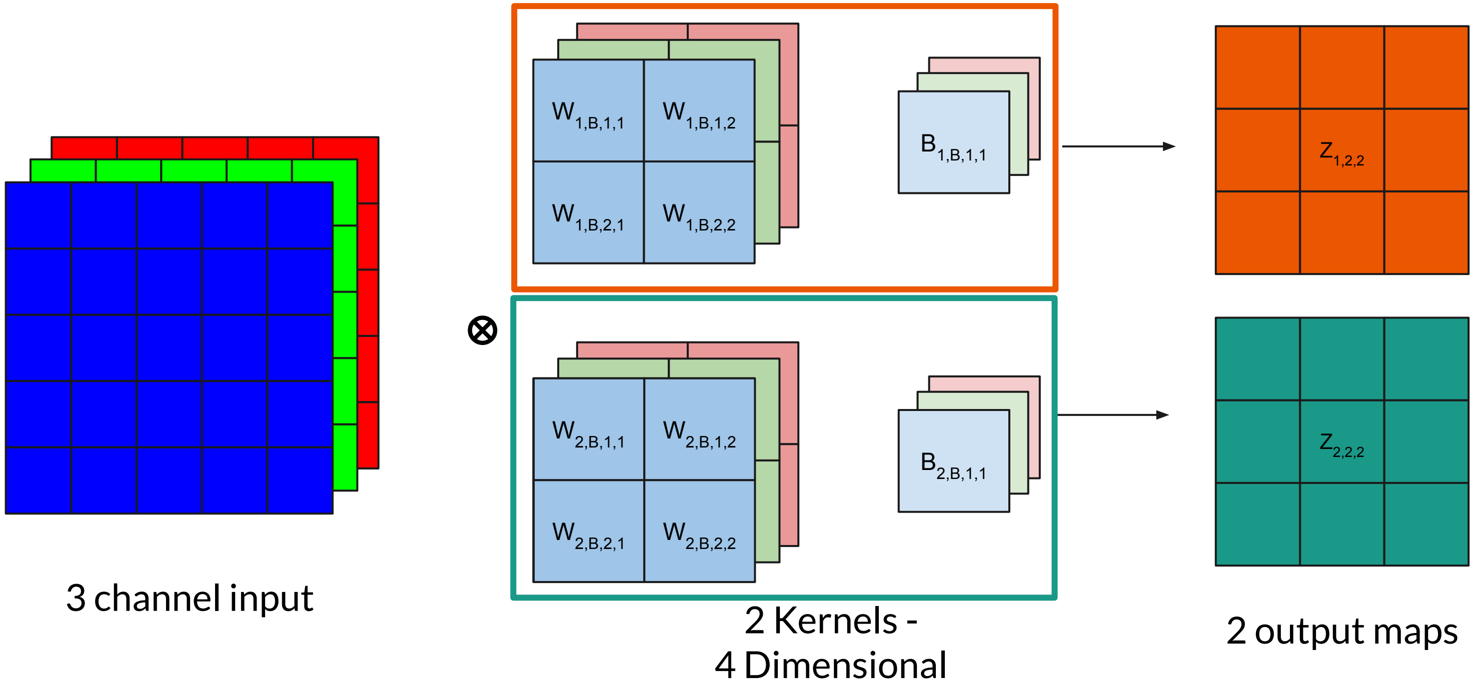
3 almost-output maps

1 output map

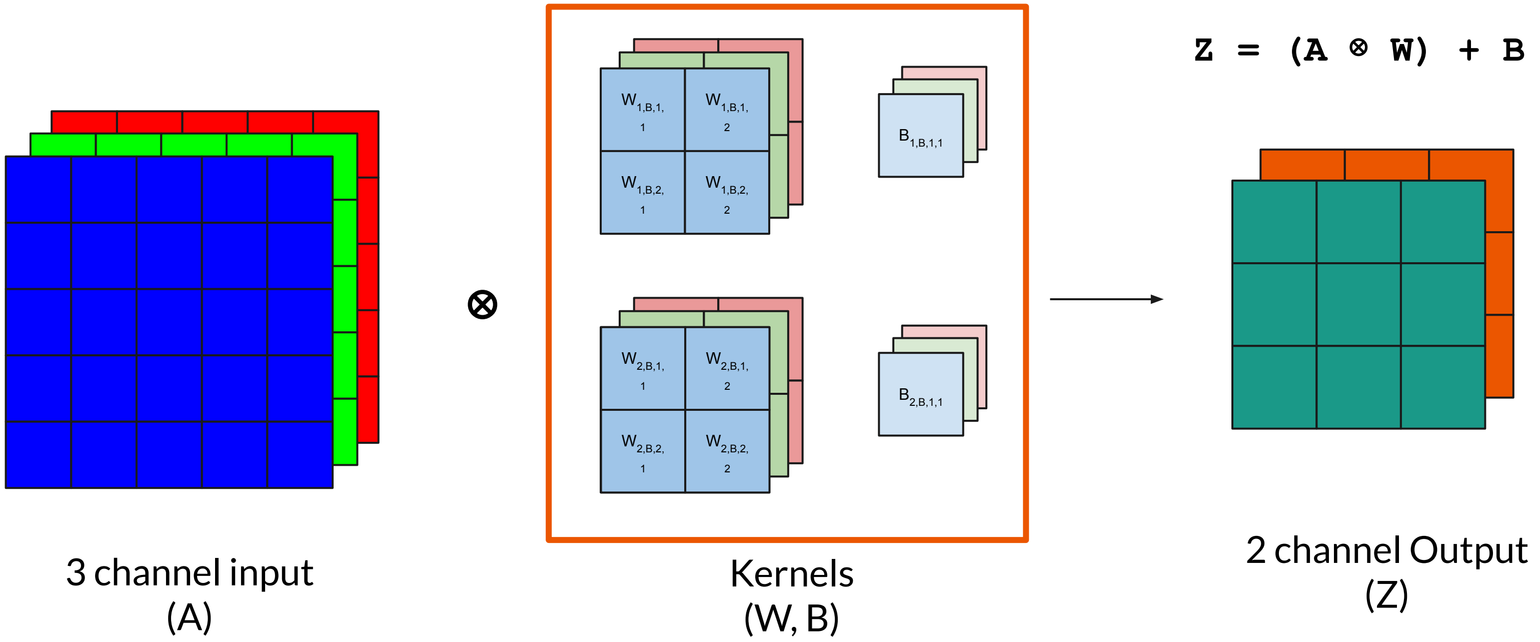
2 Filters with 3-channel input



2 Filters with 3-channel input



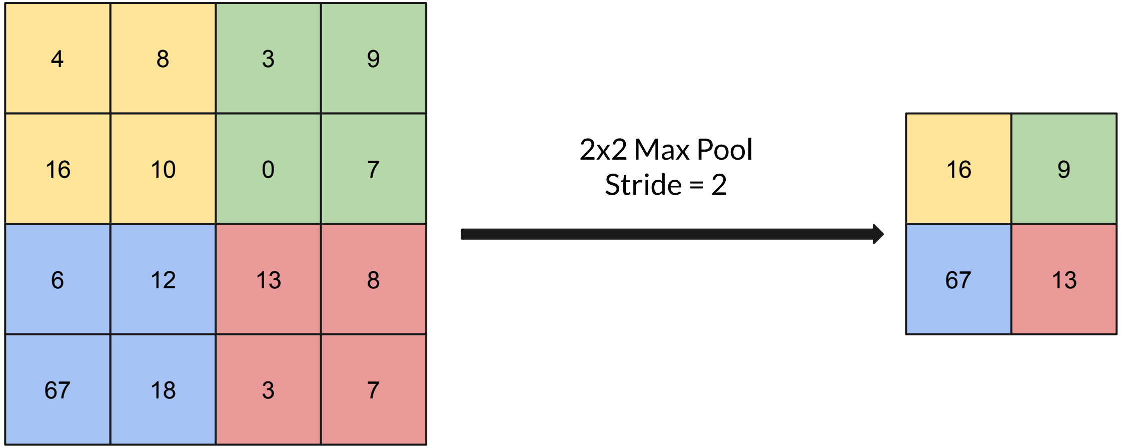
2 Filters with 3-channel input



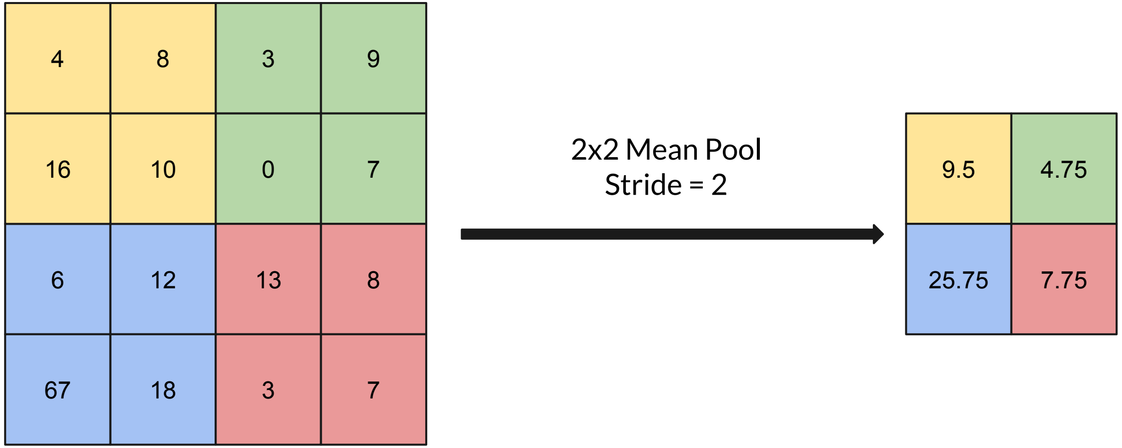
Pooling

- Usually follows convolutions
- Introduces Jitter Invariance
- Reduces feature-map size
- **Max**, **Mean**, Min

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

Pooling

- Usually follows convolutions
- Introduces Jitter Invariance
- Reduces feature-map size
- **Max**, **Mean**, Min
- What happens to the channels in pooling?

Pooling

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

Onto Backward...