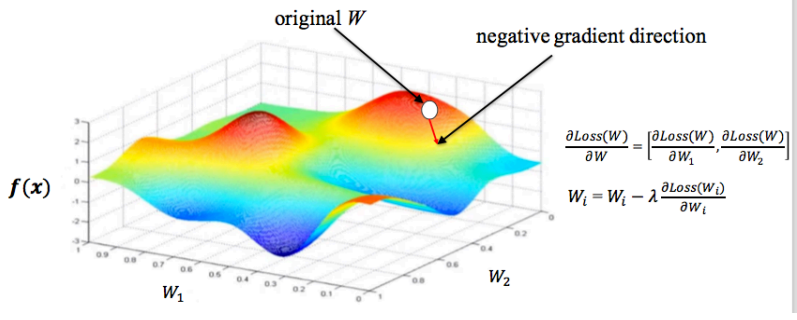
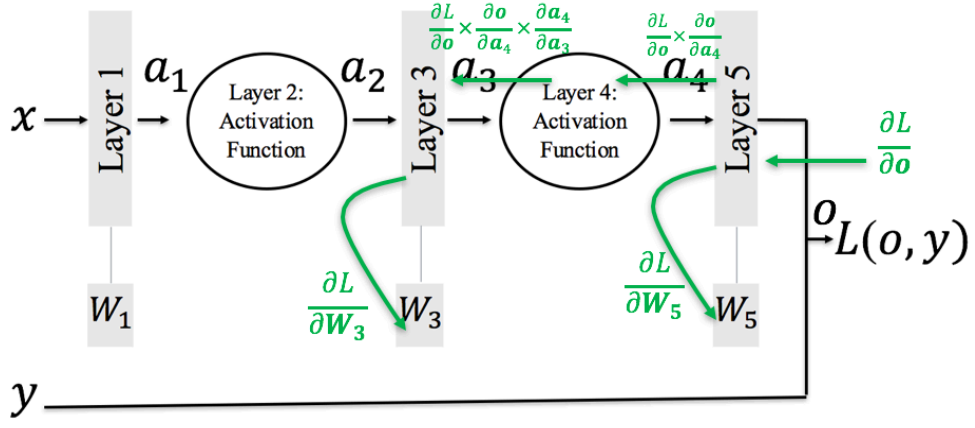
Deep Learning (for Computer Vision)

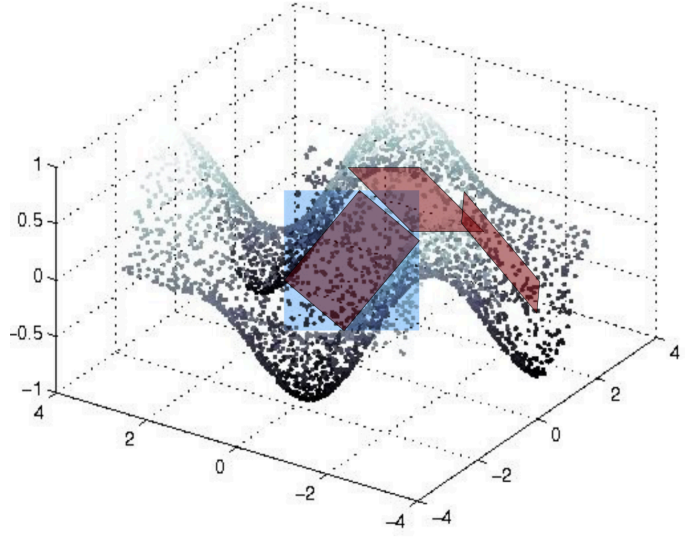
Arjun Jain | 31 March 2017

Recap

- Data-driven paradigm, CIFAR-10 dataset
- NN classifier, Linear Classifier
- Loss functions, Optimization
- Feed forward networks
- Back-propagation and chain rule
- Activation Layers
  - Sigmoid
  - ReLU
  - …







Agenda this Week

- Building blocks
  - Linear
  - Convolution
  - Max Pooling
  - Cross Entropy
  - Dropout
  - Batch Normalization
- Weight initializations, data preprocessing
- Choosing hyperparameters, baby sitting the learning process

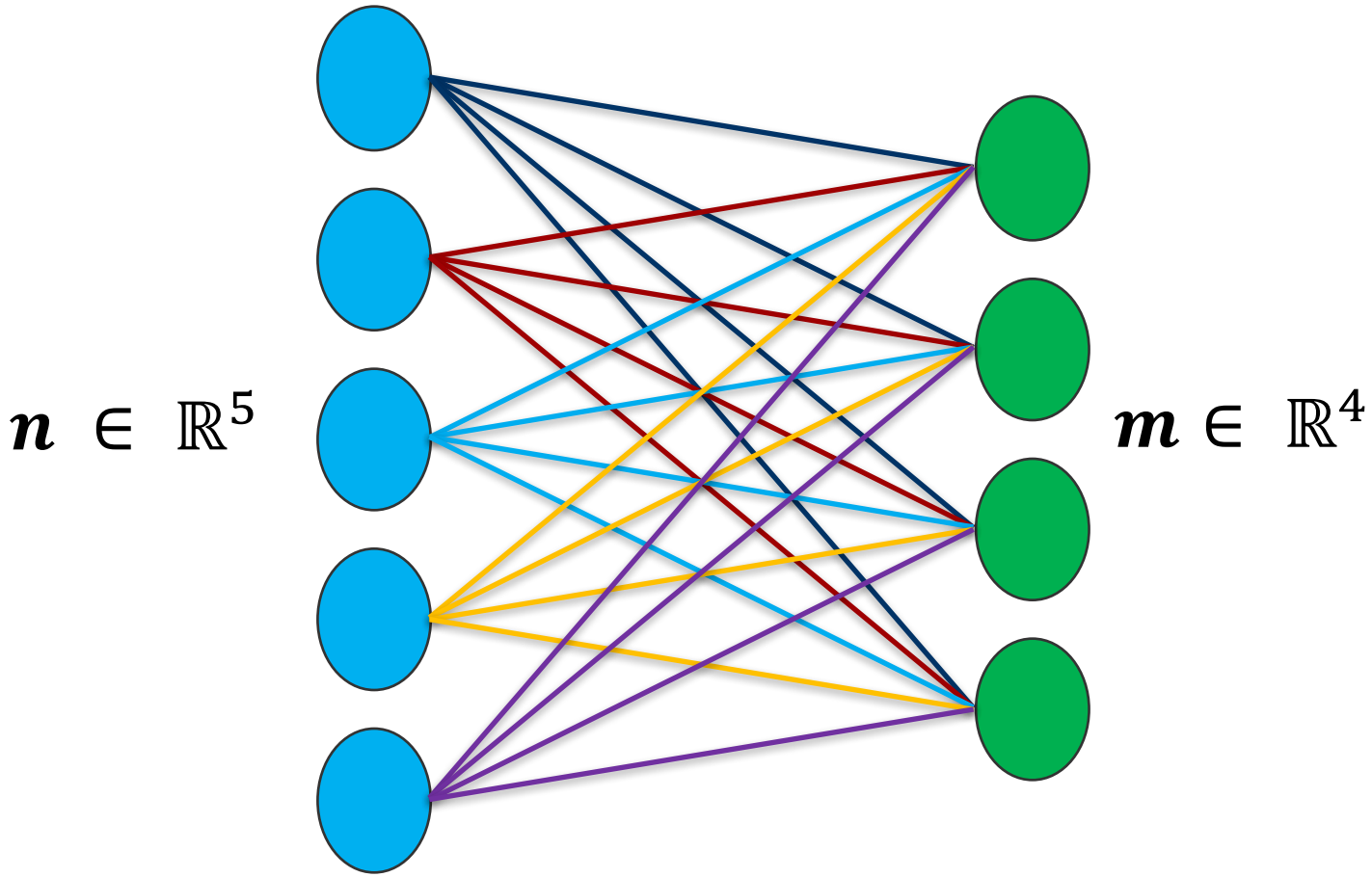
Sources

A lot of the material has been shamelessly and gratefully collected from:

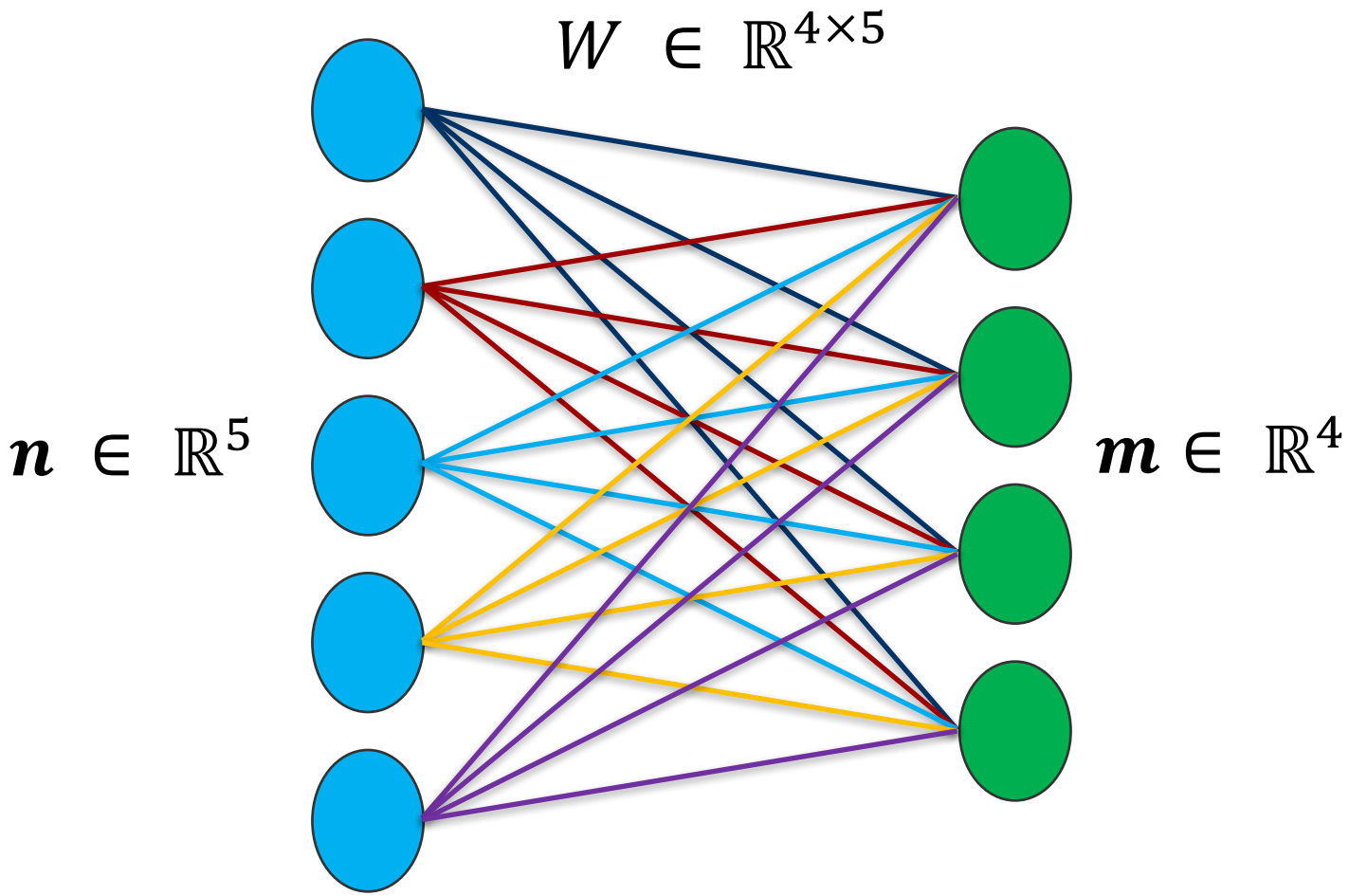
- **http://cs231n.stanford.edu/**
- http://cs231n.github.io/convolutional-networks/

Building Blocks:
Fully Connected

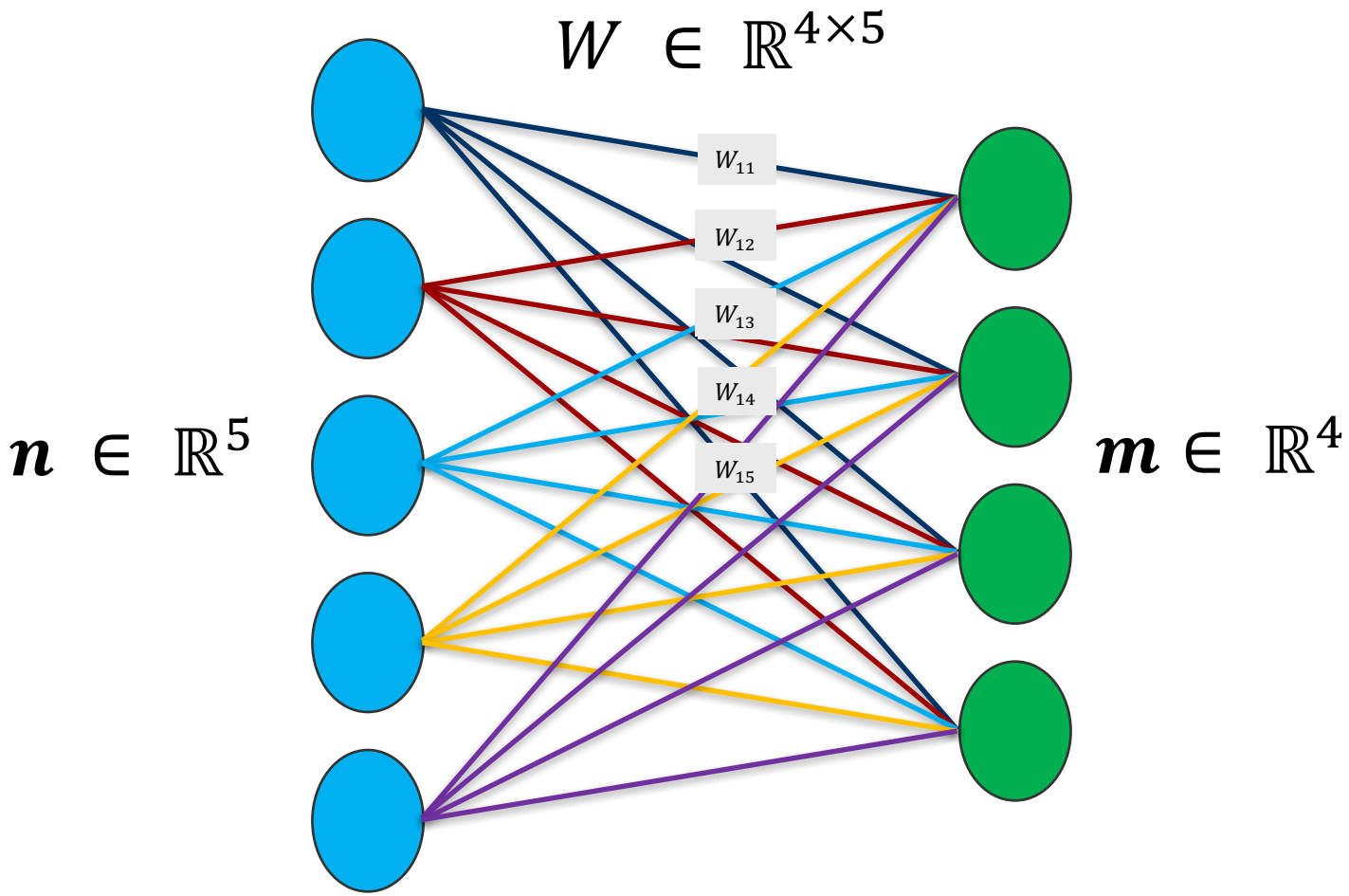
Building Blocks – Fully Connected



Building Blocks – Fully Connected

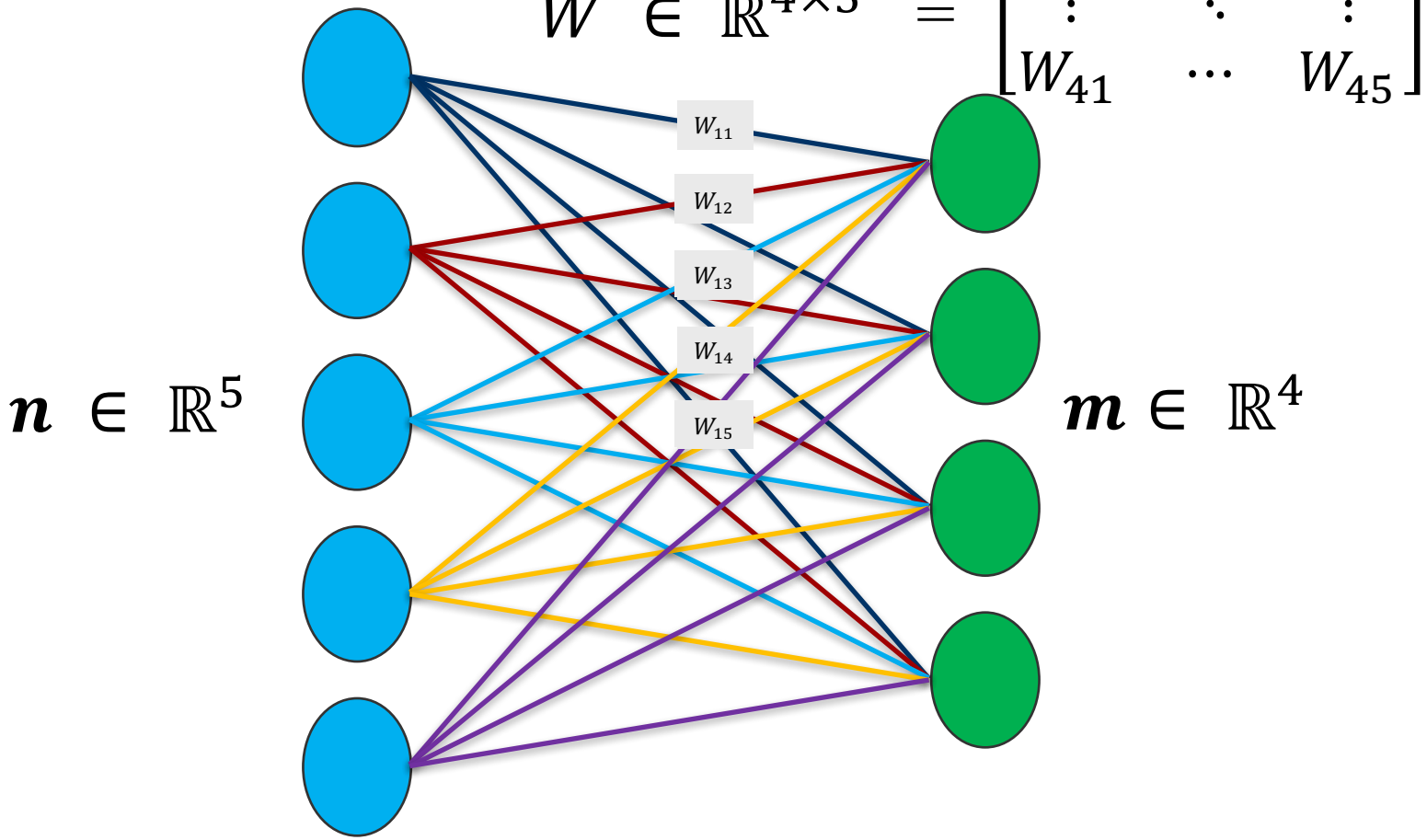


Building Blocks – Fully Connected

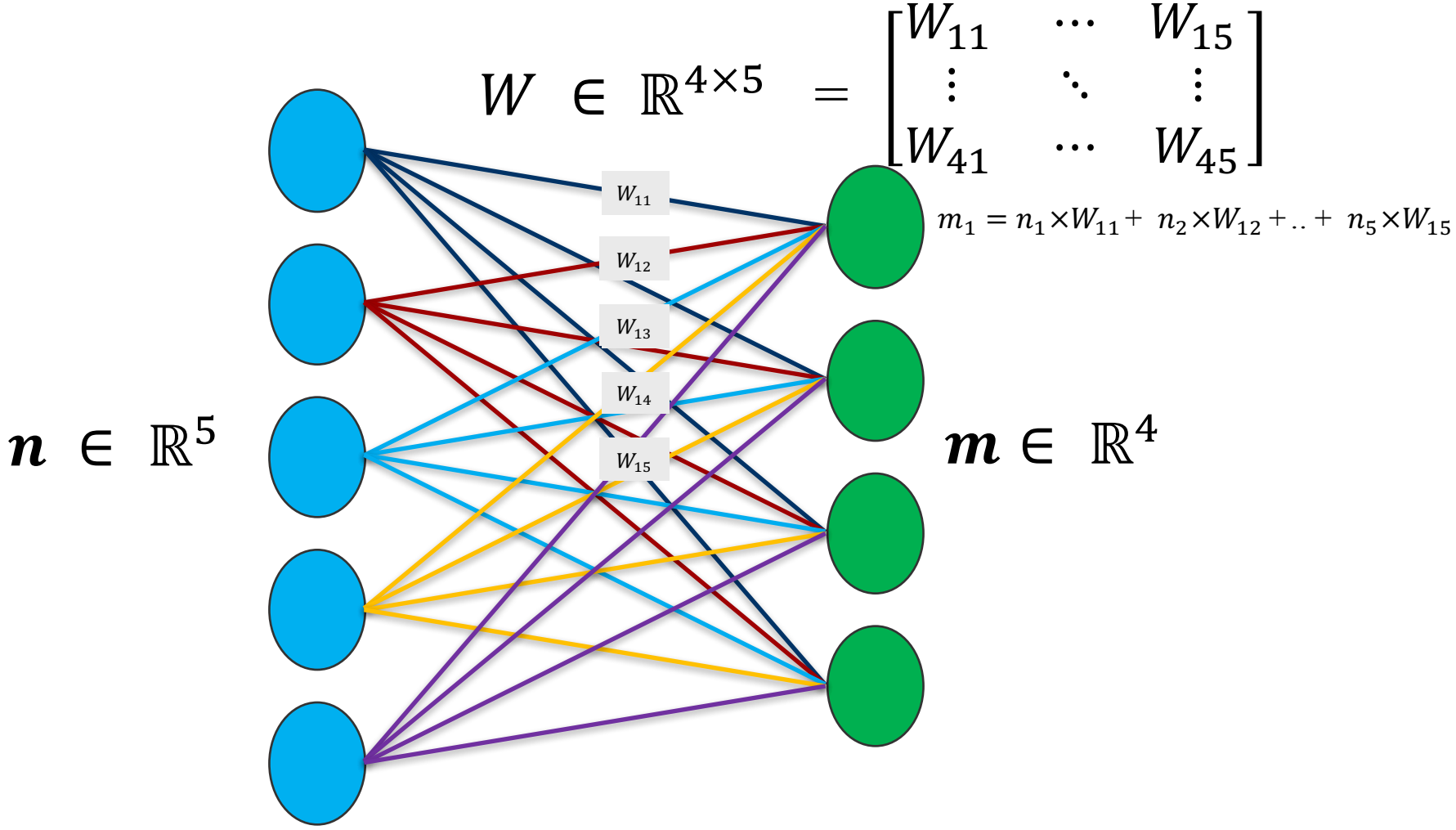


Building Blocks – Fully Connected

$$W \in \mathbb{R}^{4\times5} = \begin{bmatrix} W_{11} & \cdots & W_{15} \\ \vdots & \ddots & \vdots \\ W_{41} & \cdots & W_{45} \end{bmatrix}$$

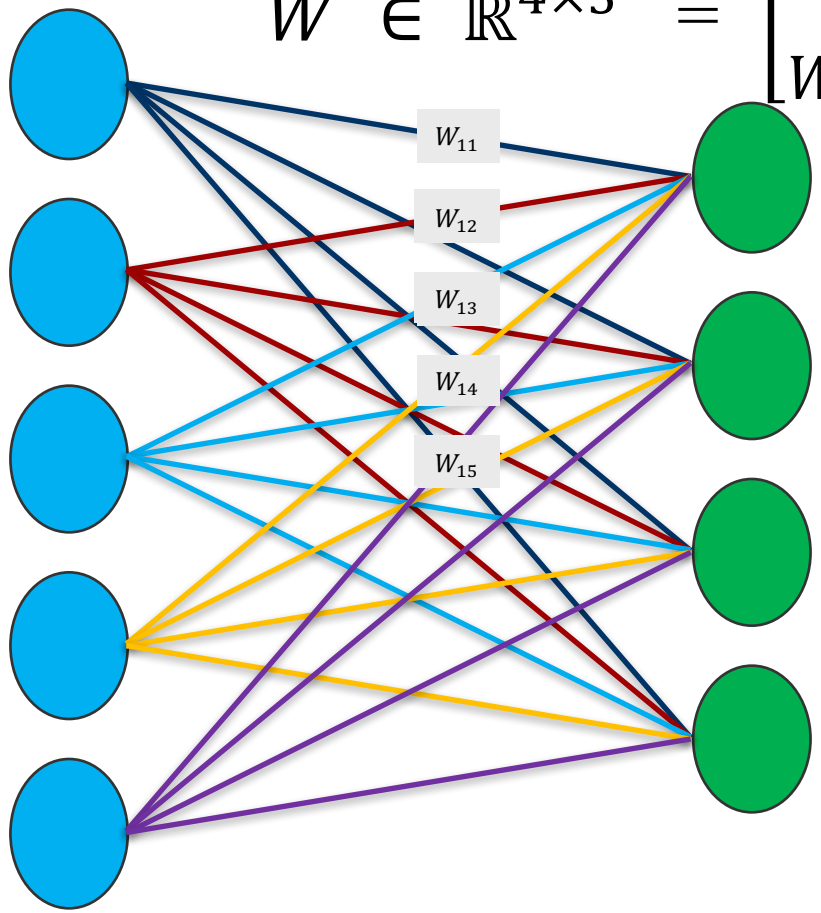


Building Blocks – Fully Connected



Building Blocks – Fully Connected

$$W \in \mathbb{R}^{4\times5} = \begin{bmatrix} W_{11} & \cdots & W_{15} \\ \vdots & \ddots & \vdots \\ W_{41} & \cdots & W_{45} \end{bmatrix}$$



$\boldsymbol{n} \in \mathbb{R}^5$

$m_1 = n_1 \times W_{11} + n_2 \times W_{12} + .. + n_5 \times W_{15}$

$m_2 = n_1 \times W_{21} + n_2 \times W_{22} + .. + n_5 \times W_{25}$

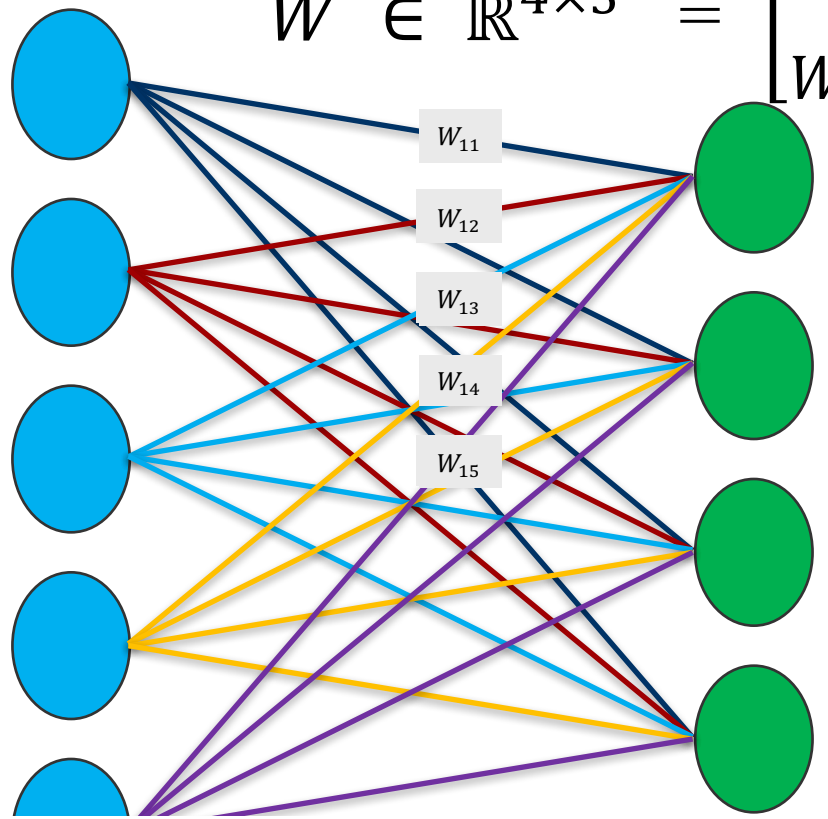
$\boldsymbol{m} \in \mathbb{R}^4$

$m_3 = n_1 \times W_{31} + n_2 \times W_{32} + .. + n_5 \times W_{35}$

$m_4 = n_1 \times W_{41} + n_2 \times W_{24} + .. + n_5 \times W_{45}$

Building Blocks – Fully Connected

$$W \in \mathbb{R}^{4\times5} = \begin{bmatrix} W_{11} & \cdots & W_{15} \\ \vdots & \ddots & \vdots \\ W_{41} & \cdots & W_{45} \end{bmatrix}$$



$\boldsymbol{n} \in \mathbb{R}^5$

$m_1 = n_1 \times W_{11} + n_2 \times W_{12} + .. + n_5 \times W_{15}$

$m_2 = n_1 \times W_{21} + n_2 \times W_{22} + .. + n_5 \times W_{25}$

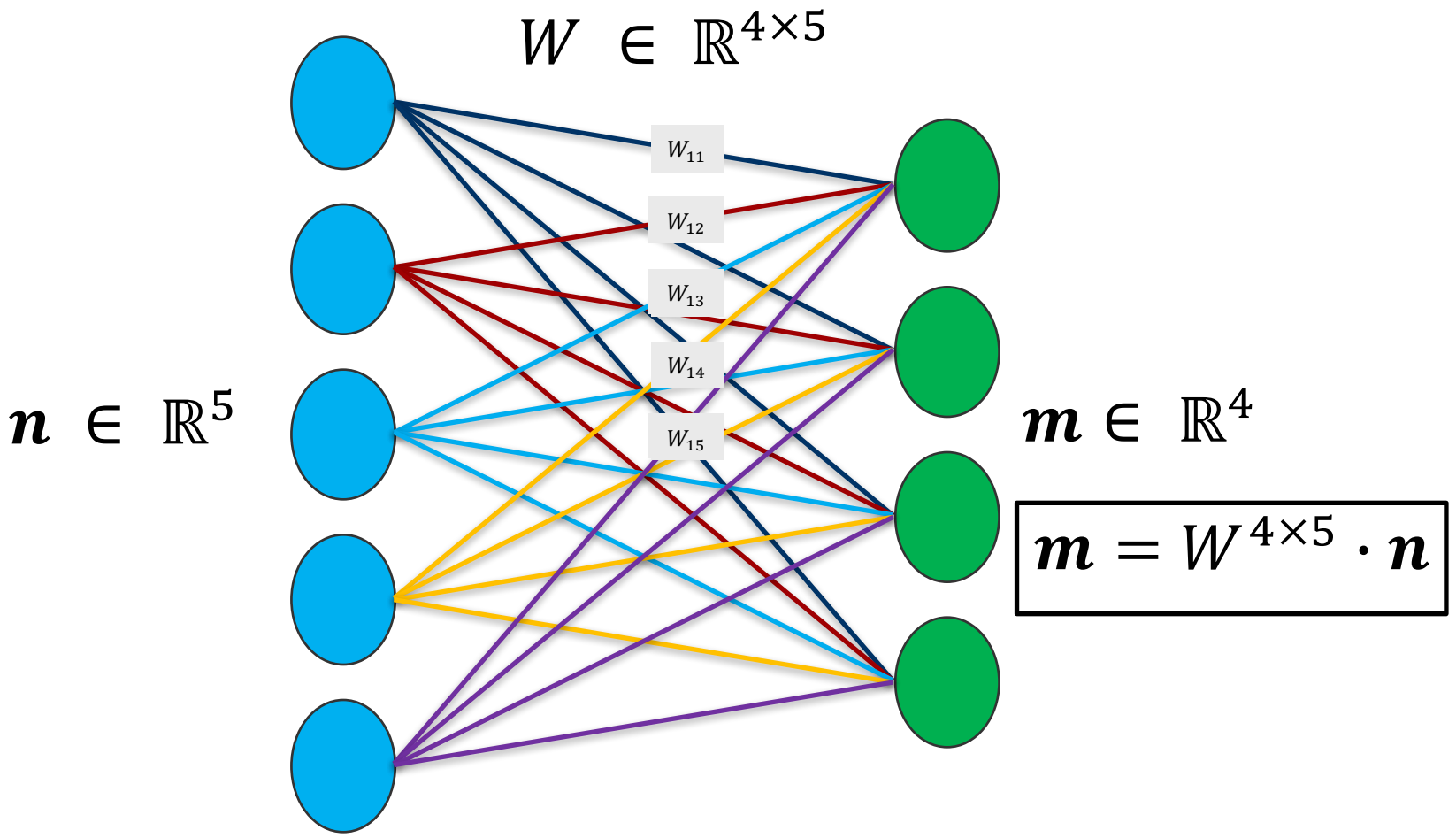
$\boldsymbol{m} \in \mathbb{R}^4$

$m_3 = n_1 \times W_{31} + n_2 \times W_{32} + .. + n_5 \times W_{35}$

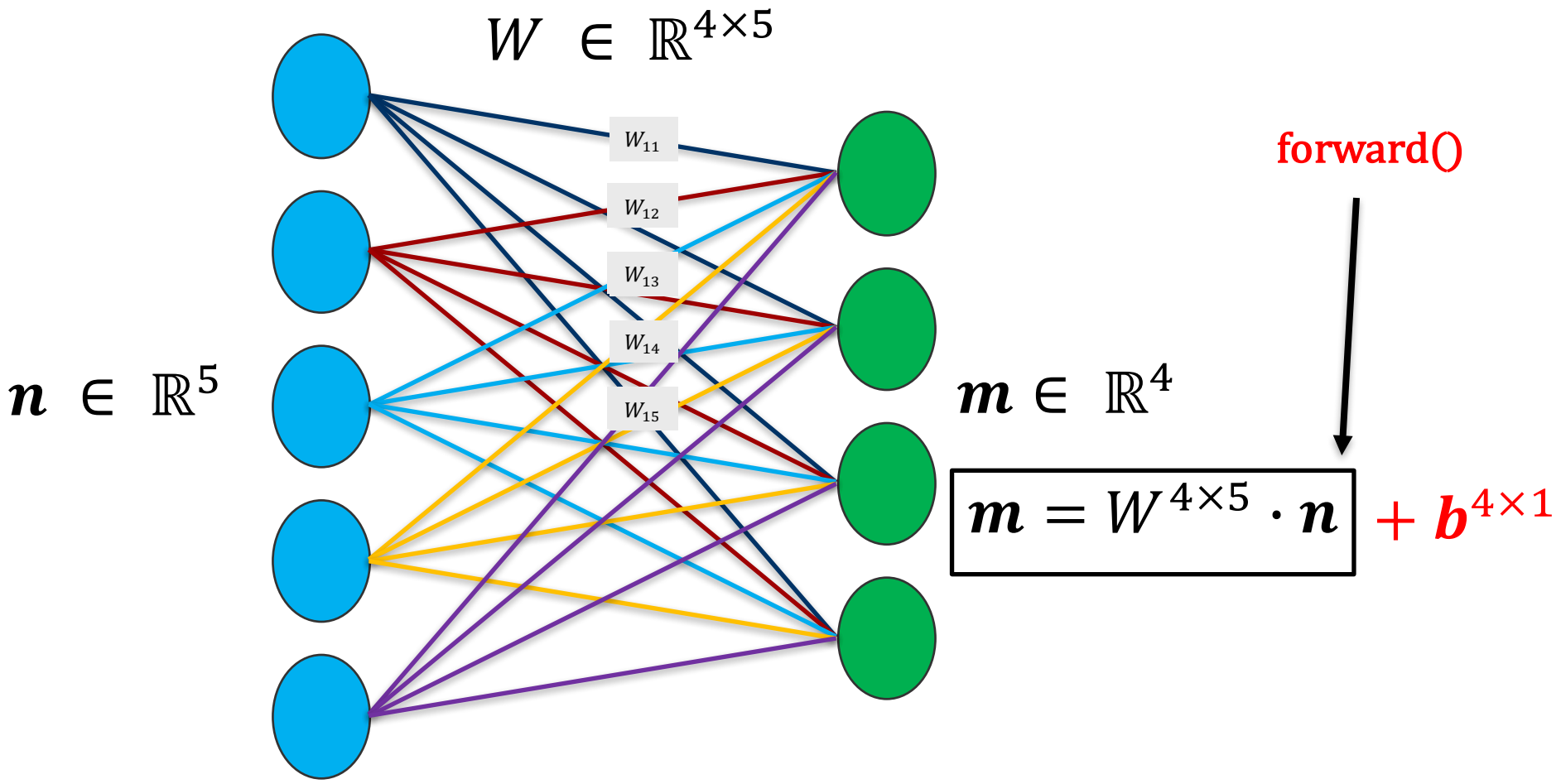
$m_4 = n_1 \times W_{41} + n_2 \times W_{24} + .. + n_5 \times W_{45}$

Why is it called fully connected?

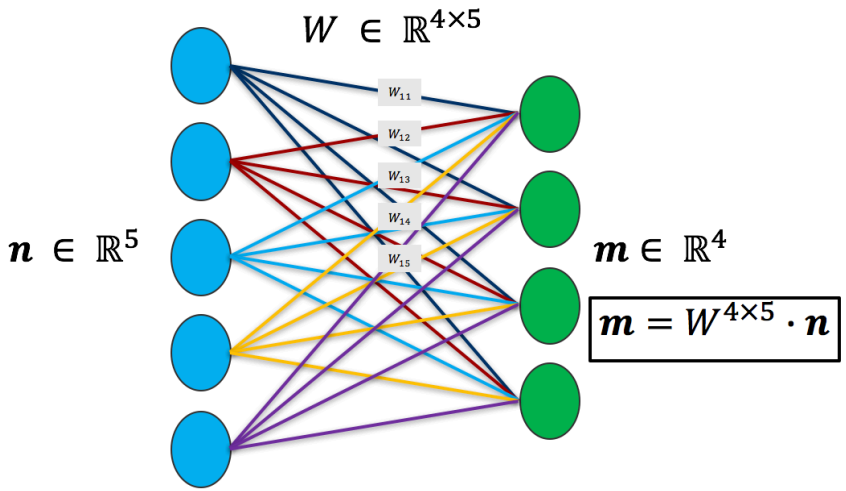
Building Blocks – Fully Connected

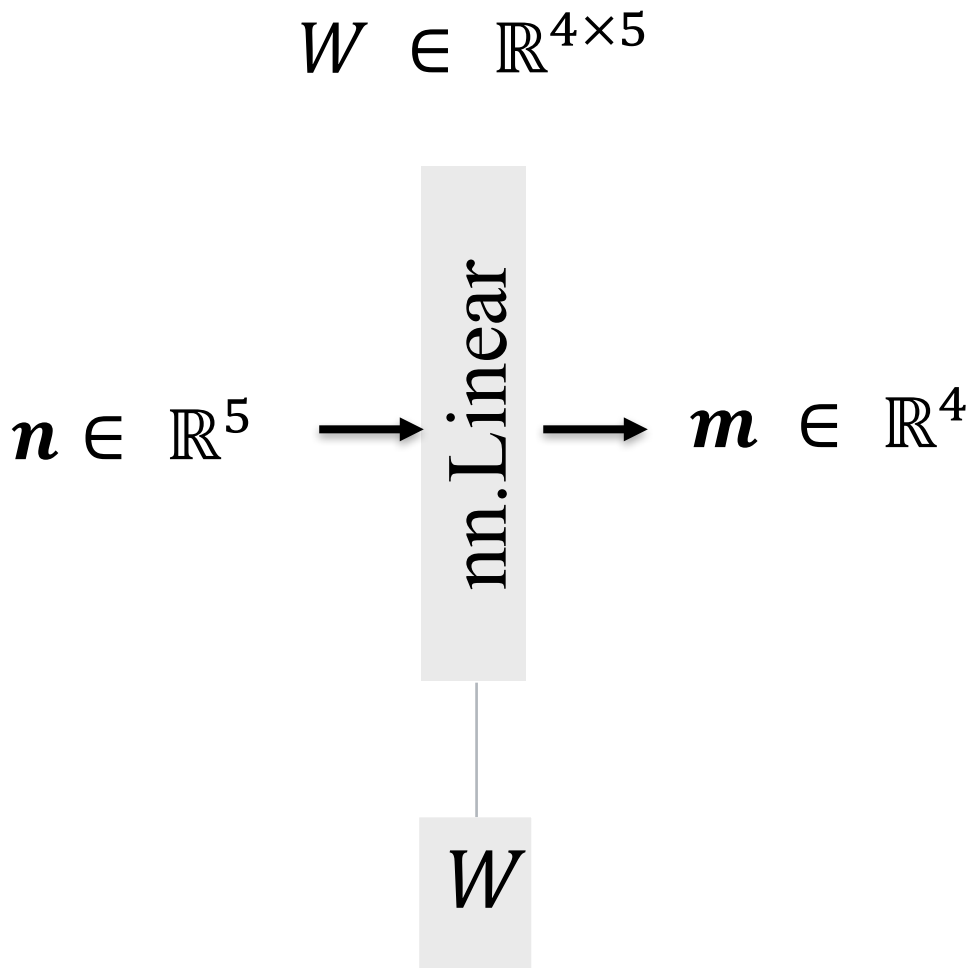


Building Blocks – Fully Connected



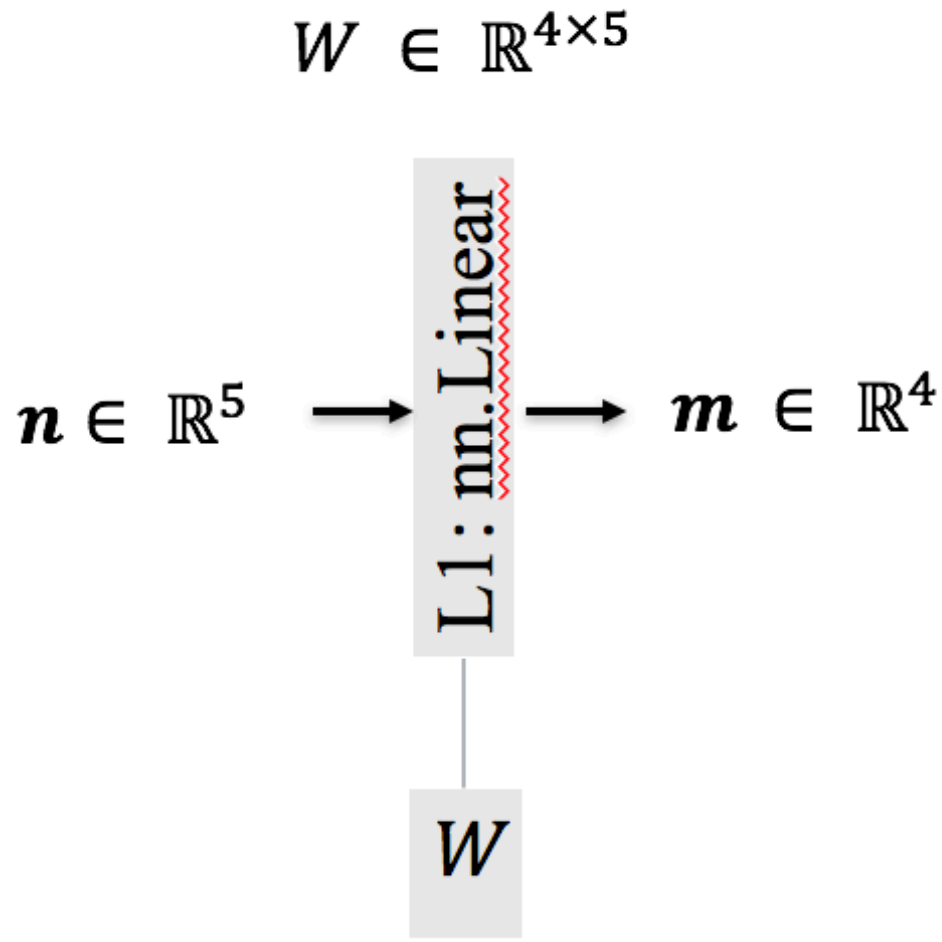
Building Blocks – Fully Connected





Building Blocks – Fully Connected – Forward: In Torch7

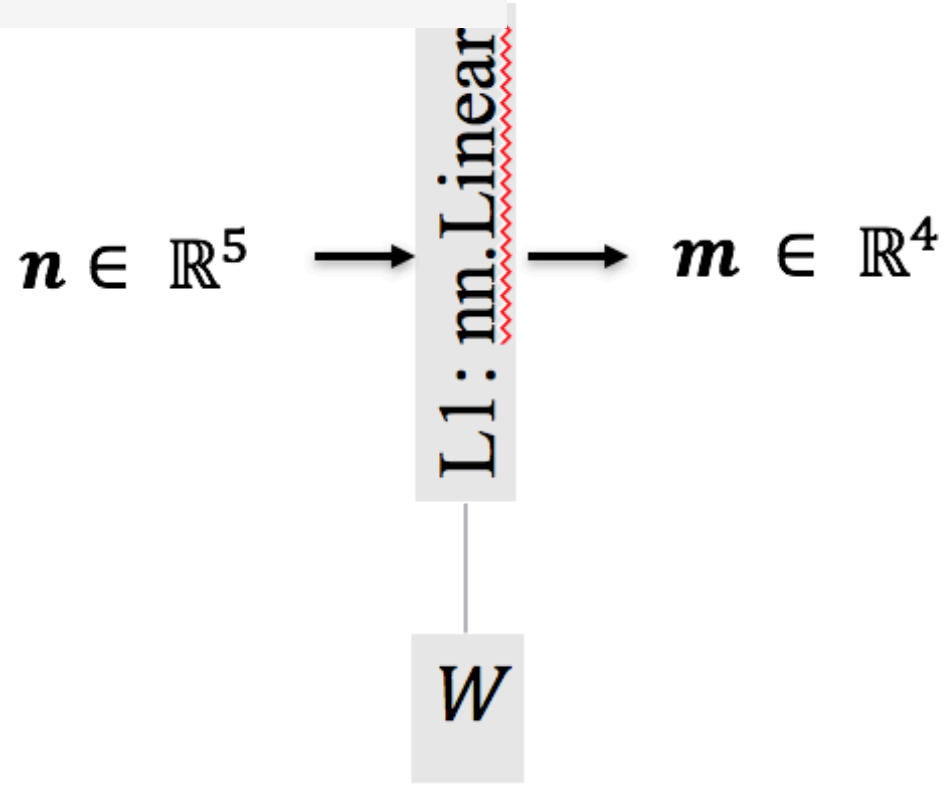
```
n

 0.2647
 0.7477
 0.5382
 0.1914
 0.0834
[torch.DoubleT
```



```
require 'nn';
n = torch.rand(5)
lin = nn.Linear(5,4)
m = lin:forward(n)
```

$\mathbb{R}^{4\times5}$

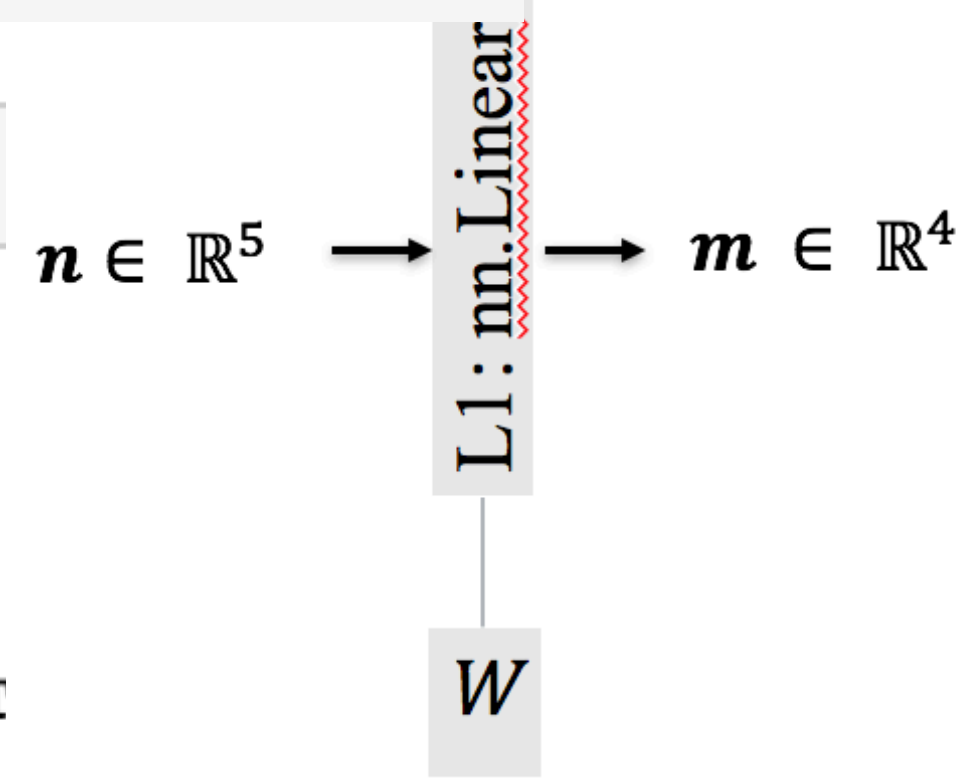
```
n

 0.2647
 0.7477
 0.5382
 0.1914
 0.0834
[torch.DoubleT
```



```
require 'nn';
n = torch.rand(5)
lin = nn.Linear(5,4)
m = lin:forward(n)
```

$\mathbb{R}^{4\times5}$

```
n

 0.2647
 0.7477
 0.5382
 0.1914
 0.0834
[torch.DoubleT
```




```
m

 0.1242
 0.2382
-0.6489
-0.2896
[torch.Doubl
```

```
require 'nn';
n = torch.rand(5)
lin = nn.Linear(5,4)
m = lin:forward(n)
```

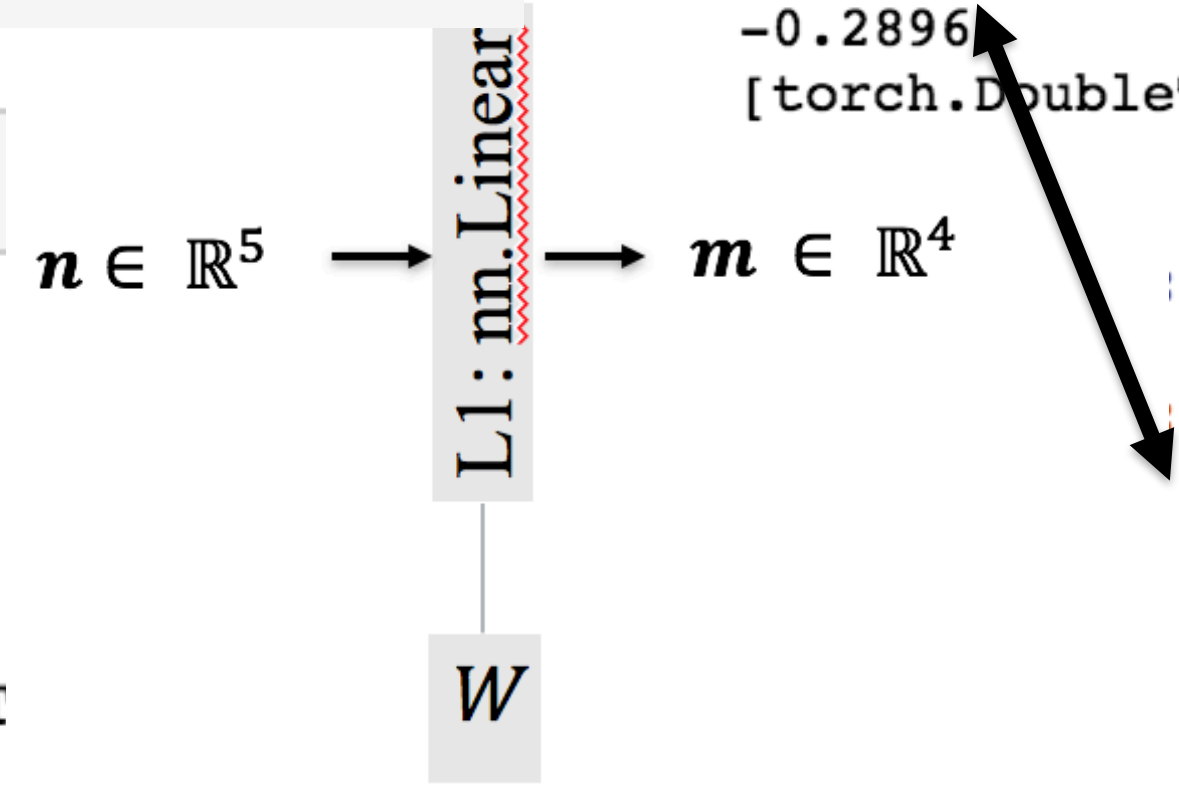
```
m_ = lin.weight*n + lin.bias
print(m_)
```

```
 0.1242
 0.2382
-0.6489
-0.2896
[torch.DoubleTensor of size 4
```

$\mathbb{R}^{4\times5}$

```
n
```

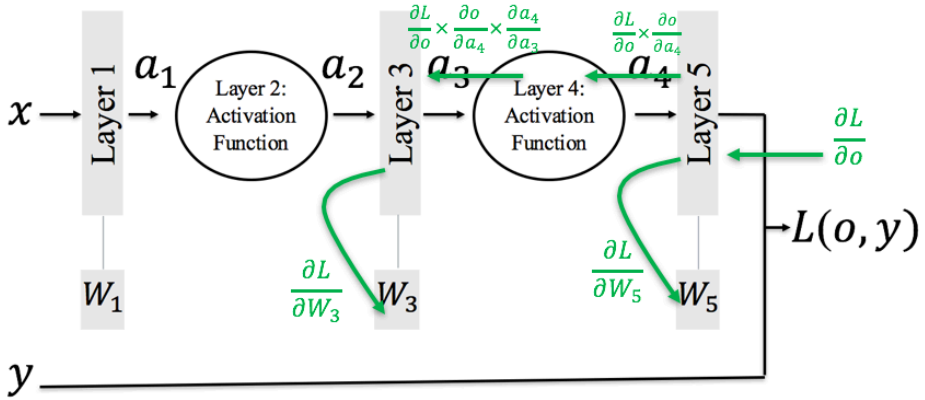
```
 0.2647
 0.7477
 0.5382
 0.1914
 0.0834
[torch.DoubleT
```

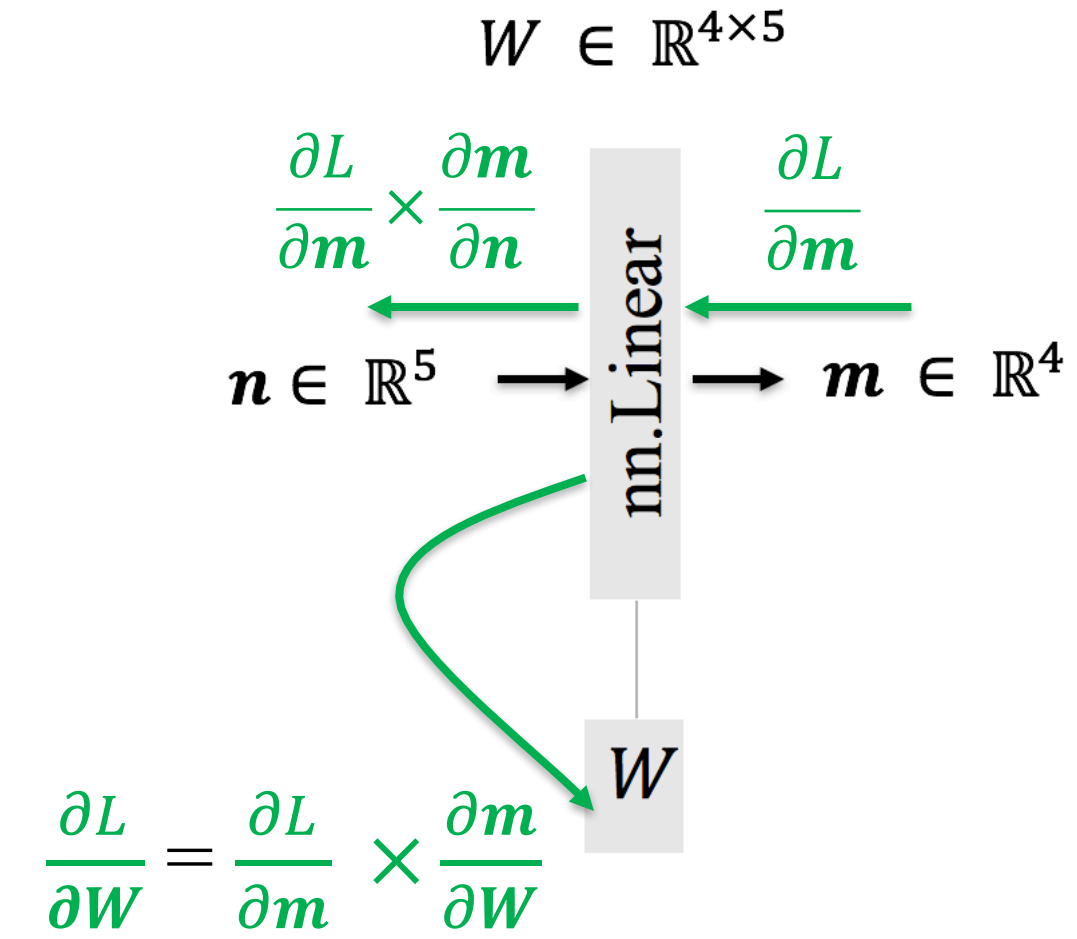



```
m
```

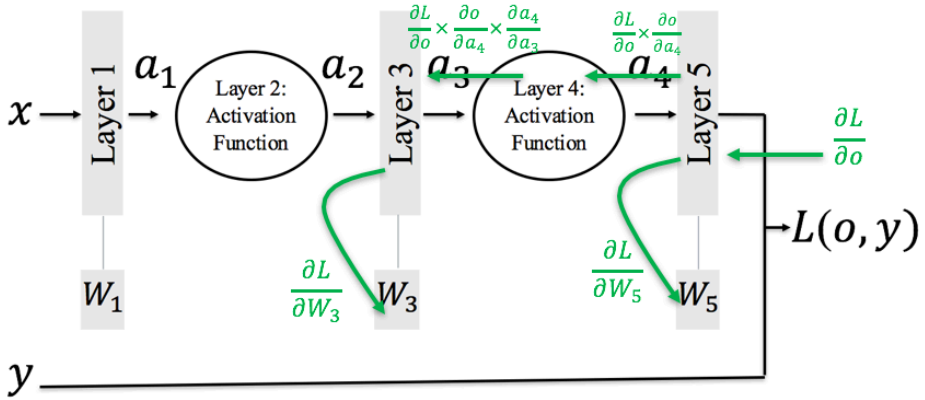
```
 0.1242
 0.2382
-0.6489
-0.2896
[torch.Doubl
```

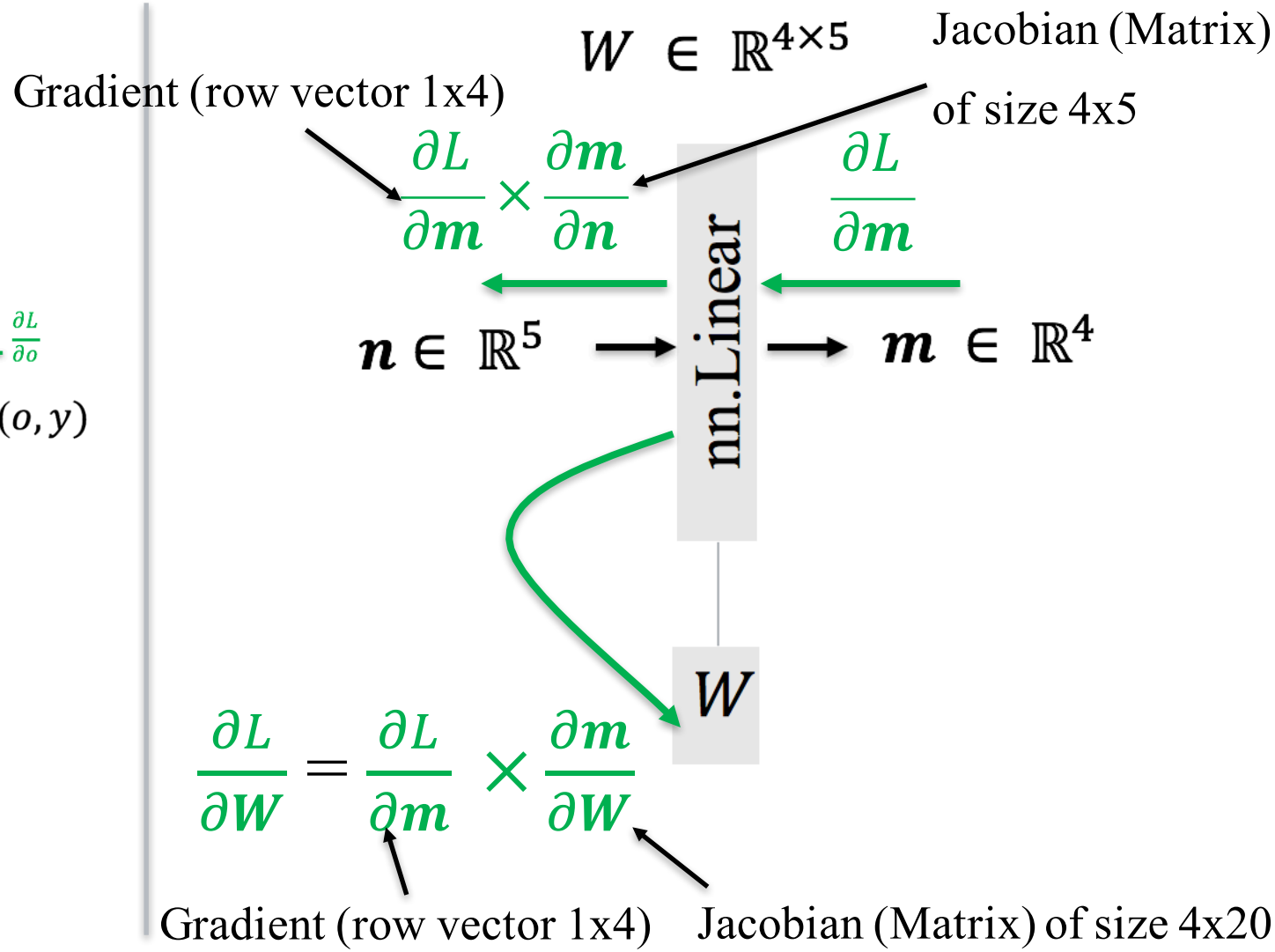
Building Blocks – Fully Connected - Backward



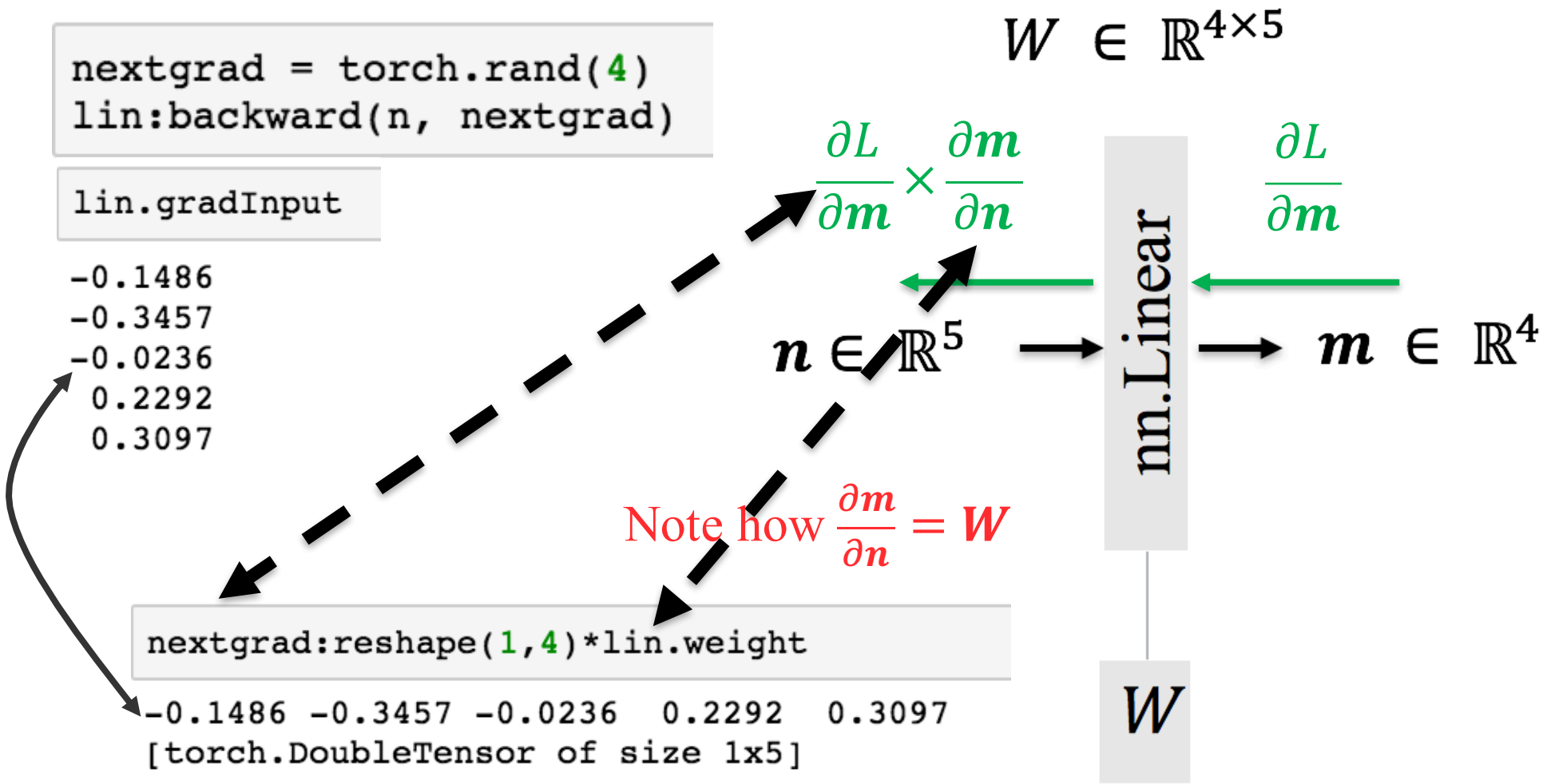


Building Blocks – Fully Connected - Backward





Fully Connected - Backward



Fully Connected - Backward

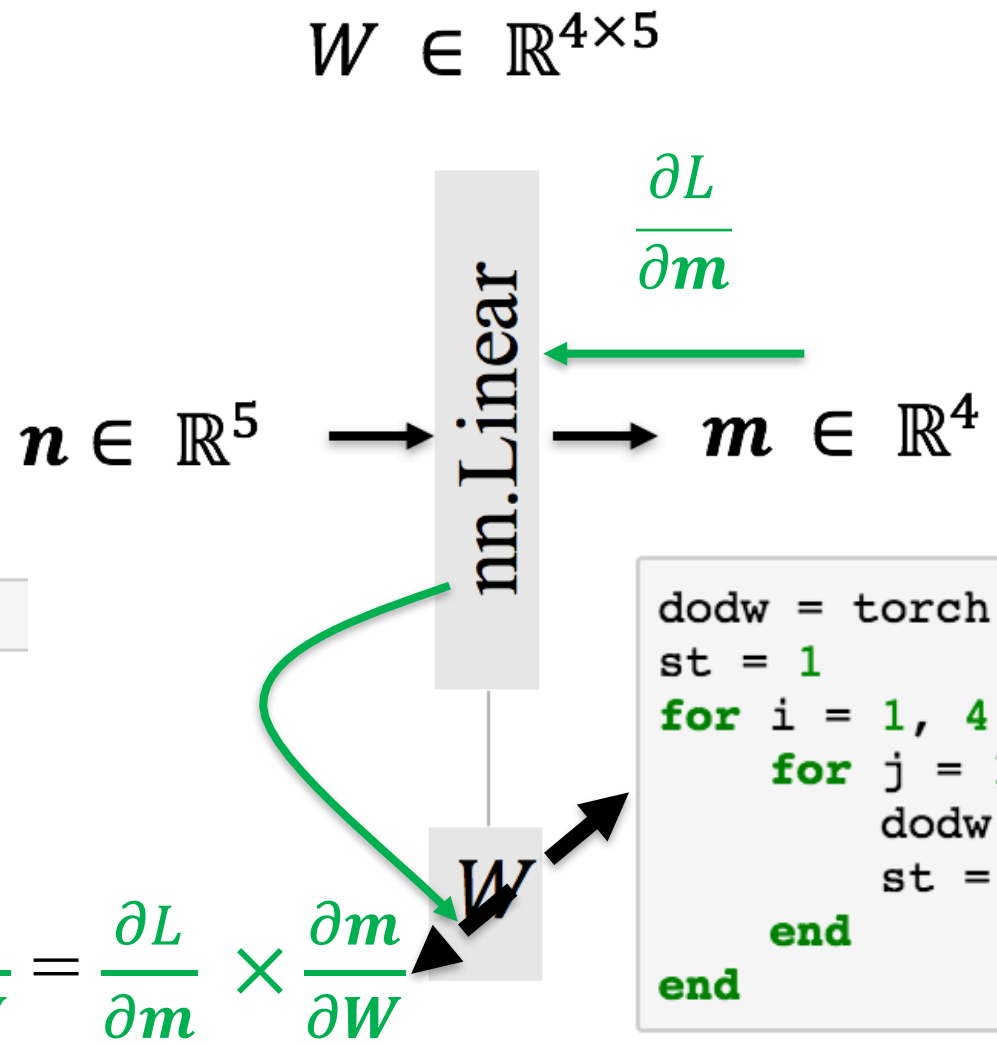
```
nextgrad = torch.rand(4)
lin:backward(n, nextgrad)
```

```
lin.gradWeight
```

```
 0.2135  0.6033  0.4343  0.1544  0.0673
 0.0556  0.1572  0.1132  0.0402  0.0175
 0.0850  0.2402  0.1729  0.0615  0.0268
 0.1717  0.4850  0.3491  0.1242  0.0541
[torch.DoubleTensor of size 4x5]
```

```
(nextgrad:reshape(1,4) * dodw):reshape(4,5)
```

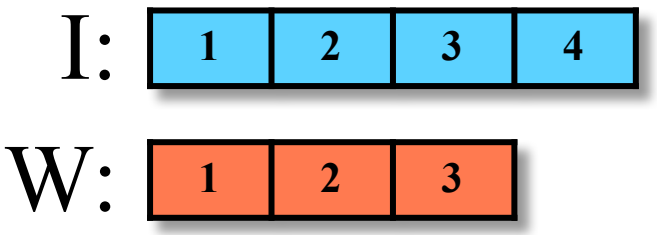
```
 0.2135  0.6033  0.4343  0.1544  0.0673
 0.0556  0.1572  0.1132  0.0402  0.0175
 0.0850  0.2402  0.1729  0.0615  0.0268
 0.1717  0.4850  0.3491  0.1242  0.0541
[torch.DoubleTensor of size 4x5]
```



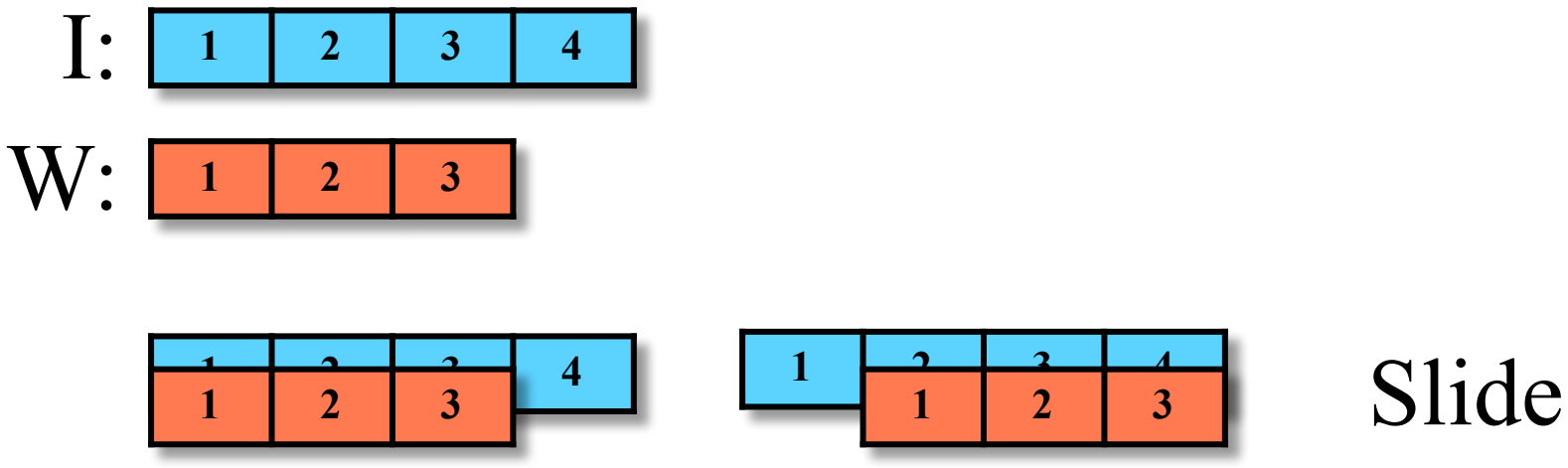

```
dodw = torch.Tensor(4,20)
st = 1
for i = 1, 4 do
    for j = 1, 5 do
        dodw[i][st]=n[j]
        st = st + 1
    end
end
```

Building Blocks:
Convolution

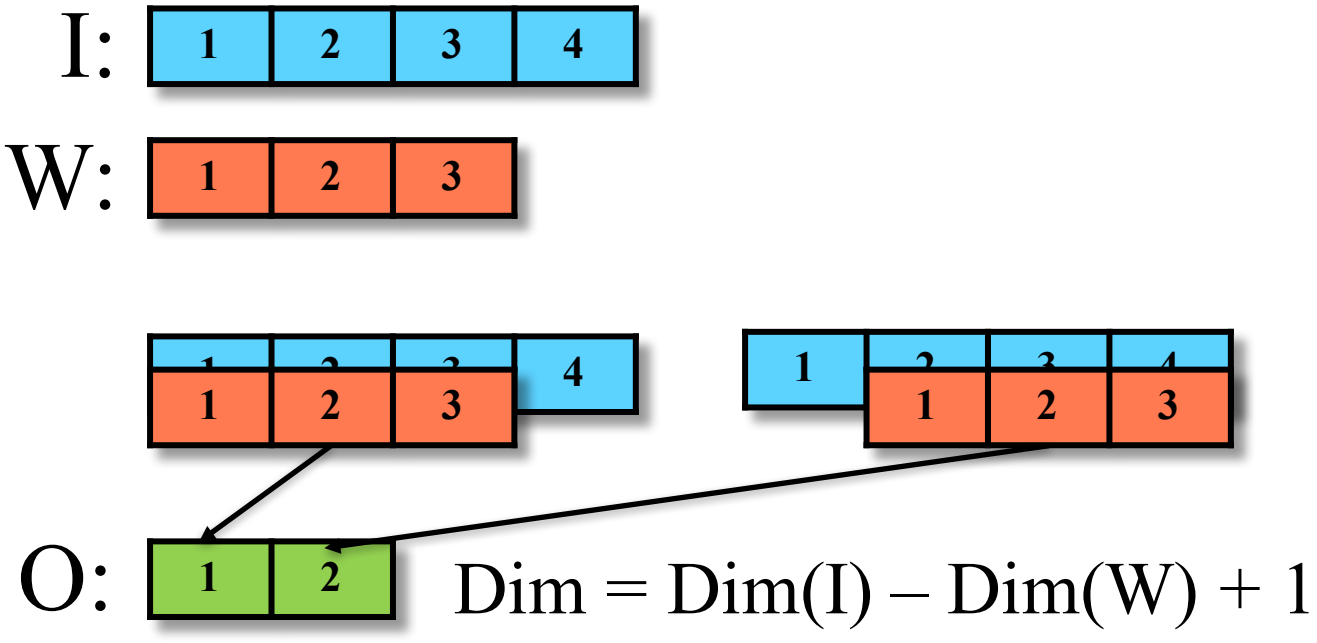
Building Blocks – Convolution (Discrete 1D)



Building Blocks – Convolution (Discrete 1D)



Building Blocks – Convolution (Discrete 1D)



Slide

Dim = Dim(I) – Dim(W) + 1

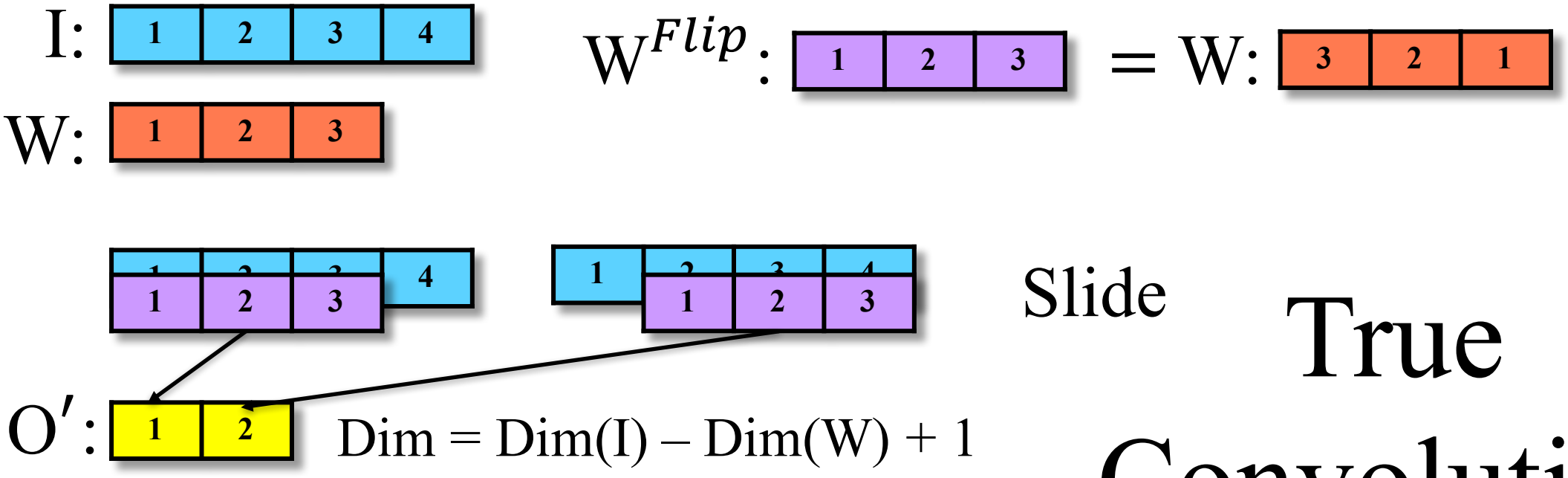
$$O_1 = I_1 W_1 + I_2 W_2 + I_3 W_3$$

$$O_2 = I_2 W_1 + I_3 W_2 + I_4 W_3$$

Correlation

$$O_i = \sum_{j=1}^{Dim(W)} I_{j+i-1} W_j$$

Building Blocks – Convolution (Discrete 1D)



Slide

Dim = Dim(I) – Dim(W) + 1

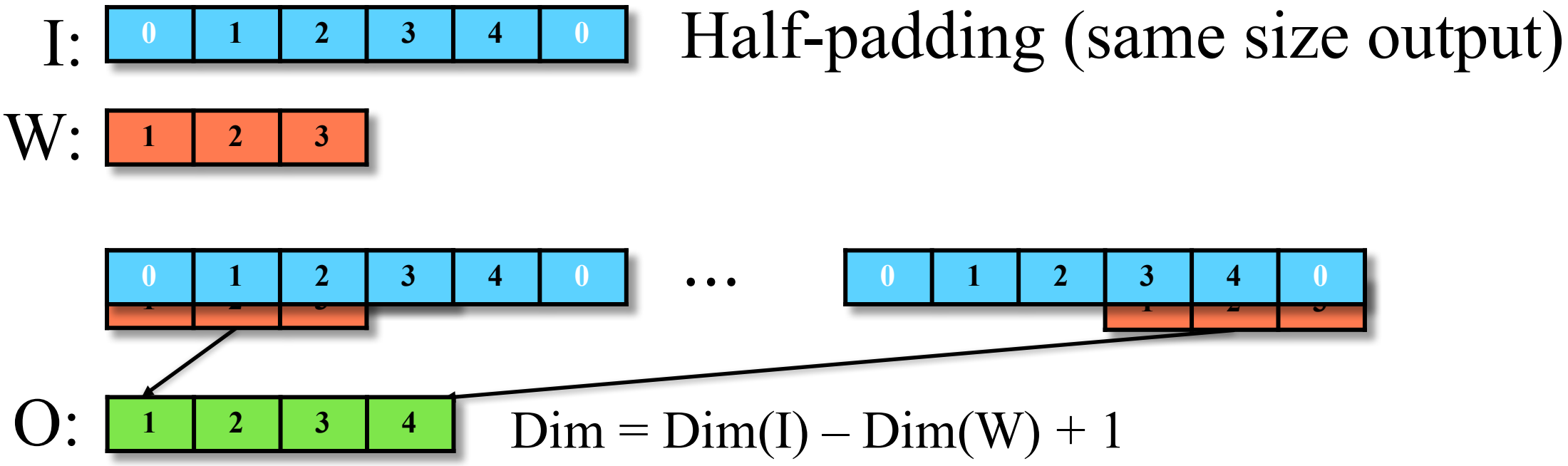
$$O'_1 = I_1 W_1^{Flip} + I_2 W_2^{Flip} + I_3 W_3^{Flip}$$

$$O'_2 = I_2 W_1^{Flip} + I_3 W_2^{Flip} + I_4 W_3^{Flip}$$

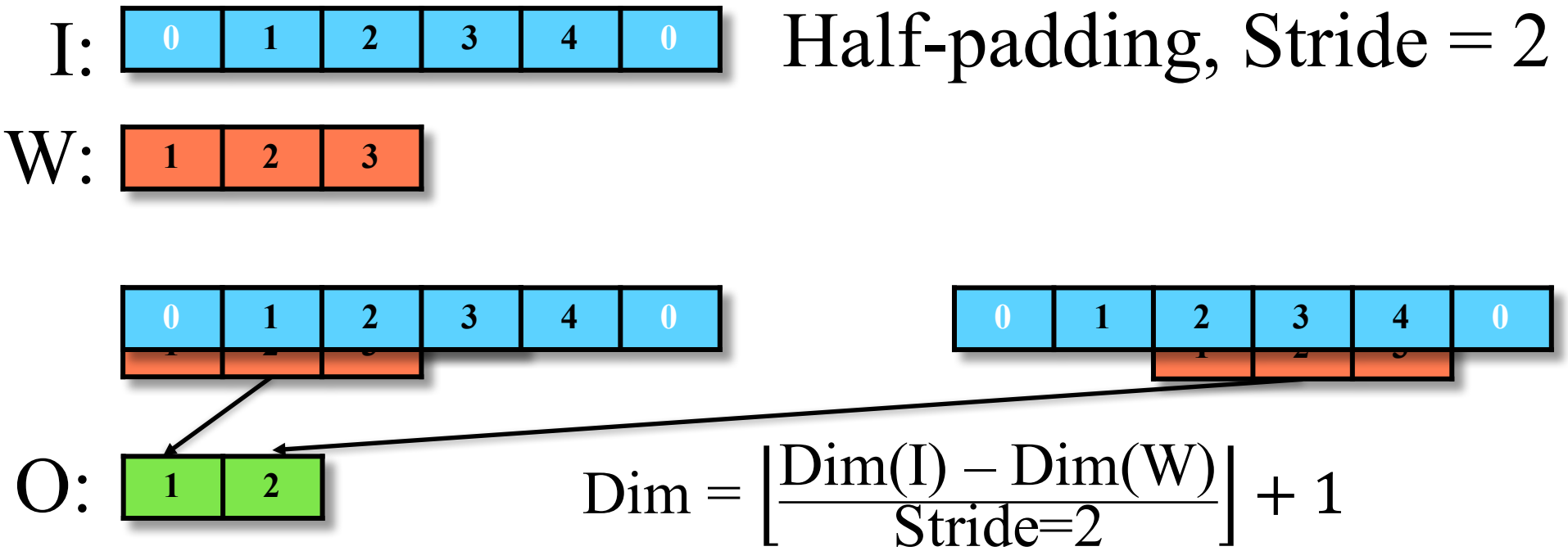
True Convolution

$$O_i = \sum_{j=1}^{Dim(W)} I_{j+i-1} W_{Dim(W)-j+1}$$

Building Blocks – Convolution (Discrete 1D)

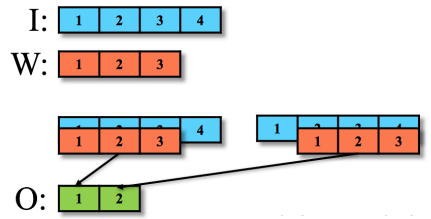


Building Blocks – Convolution (Discrete 1D)



Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$



$O_1 = I_1W_1 + I_2W_2 + I_3W_3$

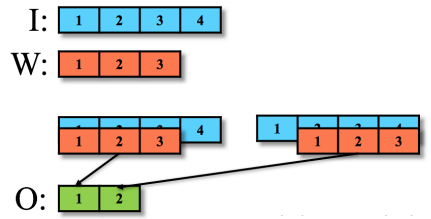
$O_2 = I_2W_1 + I_3W_2 + I_4W_3$

Correlation

Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$



$O_1 = I_1W_1 + I_2W_2 + I_3W_3$

$O_2 = I_2W_1 + I_3W_2 + I_4W_3$

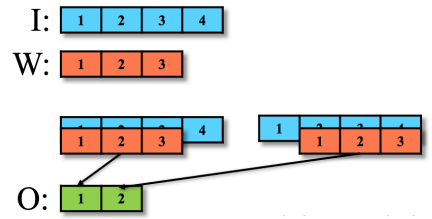
Correlation

Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial L O_1 \quad \partial L O_2]$$



$O_1 = I_1 W_1 + I_2 W_2 + I_3 W_3$

$O_2 = I_2 W_1 + I_3 W_2 + I_4 W_3$

Correlation

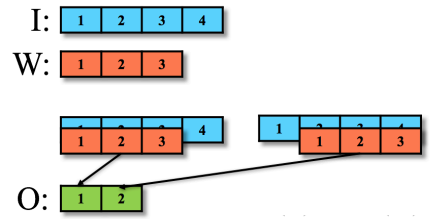
Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial LO_1 \quad \partial LO_2]$$

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{O}} \times \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = [\partial LO_1 \times I_1 + \partial LO_2 \times I_2 \quad \partial LO_1 \times I_2 + \partial LO_2 \times I_3 \quad \partial LO_1 \times I_3 + \partial LO_2 \times I_4]$$



$O_1 = I_1W_1 + I_2W_2 + I_3W_3$

$O_2 = I_2W_1 + I_3W_2 + I_4W_3$

Correlation

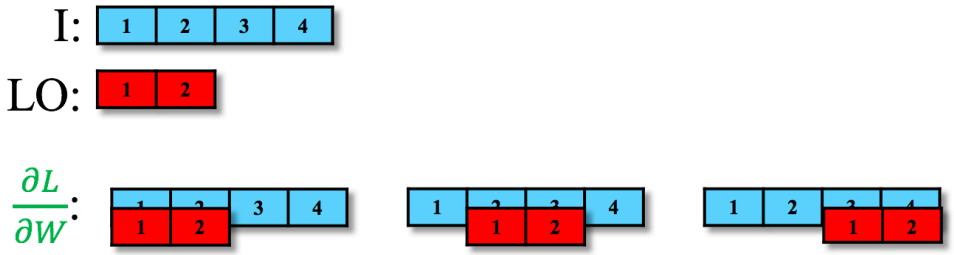
Building Blocks – Convolution – Backward

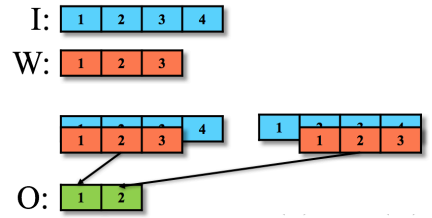
$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial LO_1 \quad \partial LO_2]$$

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{O}} \times \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = [\partial LO_1 \times I_1 + \partial LO_2 \times I_2 \quad \partial LO_1 \times I_2 + \partial LO_2 \times I_3 \quad \partial LO_1 \times I_3 + \partial LO_2 \times I_4]$$





$O_1 = I_1W_1 + I_2W_2 + I_3W_3$

$O_2 = I_2W_1 + I_3W_2 + I_4W_3$

Correlation

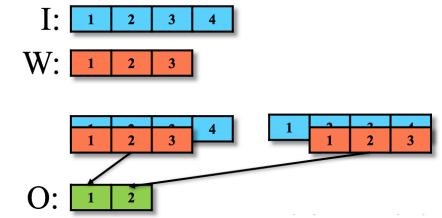
Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial LO_1 \quad \partial LO_2]$$

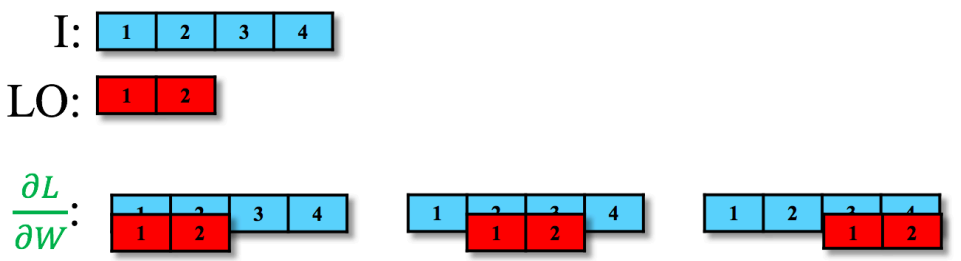
$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{O}} \times \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = [\partial LO_1 \times I_1 + \partial LO_2 \times I_2 \quad \partial LO_1 \times I_2 + \partial LO_2 \times I_3 \quad \partial LO_1 \times I_3 + \partial LO_2 \times I_4]$$



$O_1 = I_1W_1 + I_2W_2 + I_3W_3$

$O_2 = I_2W_1 + I_3W_2 + I_4W_3$

Correlation



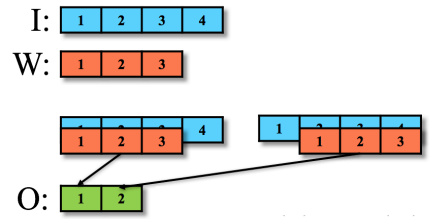
$$\frac{\partial L}{\partial \boldsymbol{W}} = \text{Correlation(I, LO)}$$

Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial L O_1 \quad \partial L O_2]$$



$O_1 = I_1 W_1 + I_2 W_2 + I_3 W_3$

$O_2 = I_2 W_1 + I_3 W_2 + I_4 W_3$

Correlation

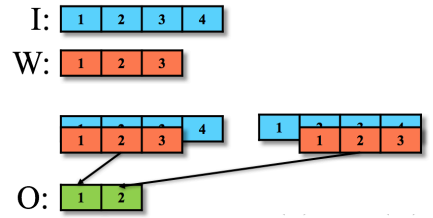
Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial LO_1 \quad \partial LO_2]$$

$$\frac{\partial L}{\partial \boldsymbol{I}} = \frac{\partial L}{\partial \boldsymbol{O}} \times \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = [\partial LO_1 \times W_1 \quad \partial LO_1 \times W_2 + \partial LO_2 \times W_1 \quad \partial LO_1 \times W_3 + \partial LO_2 \times W_2 \quad \partial LO_2 \times W_3]$$



$O_1 = I_1W_1 + I_2W_2 + I_3W_3$

$O_2 = I_2W_1 + I_3W_2 + I_4W_3$

Correlation

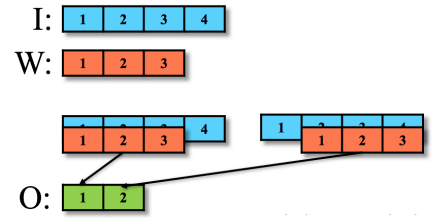
Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial LO_1 \quad \partial LO_2]$$

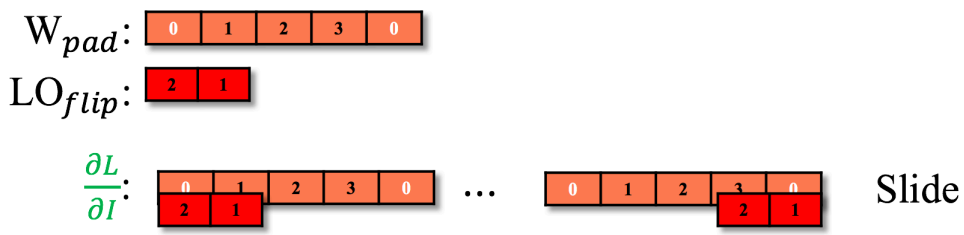
$$\frac{\partial L}{\partial \boldsymbol{I}} = \frac{\partial L}{\partial \boldsymbol{O}} \times \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = [\partial LO_1 \times W_1 \quad \partial LO_1 \times W_2 + \partial LO_2 \times W_1 \quad \partial LO_1 \times W_3 + \partial LO_2 \times W_2 \quad \partial LO_2 \times W_3]$$



$O_1 = I_1W_1 + I_2W_2 + I_3W_3$

$O_2 = I_2W_1 + I_3W_2 + I_4W_3$

Correlation



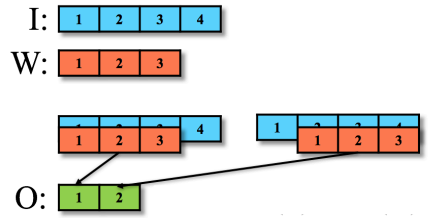
Building Blocks – Convolution – Backward

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$

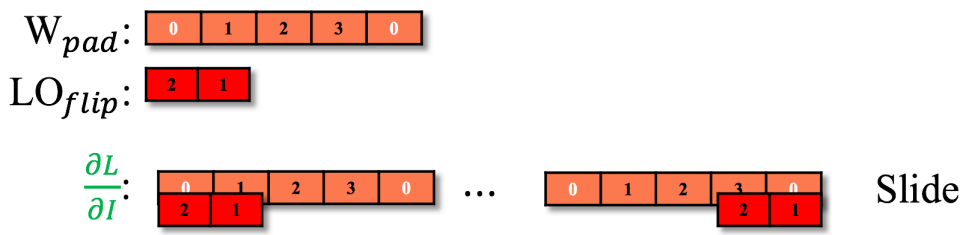
$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{W}} = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_3 & I_4 \end{bmatrix}$$

$$\frac{\partial L}{\partial \boldsymbol{O}} = [\partial LO_1 \quad \partial LO_2]$$

$$\frac{\partial L}{\partial \boldsymbol{I}} = \frac{\partial L}{\partial \boldsymbol{O}} \times \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{I}} = [\partial LO_1 \times W_1 \quad \partial LO_1 \times W_2 + \partial LO_2 \times W_1 \quad \partial LO_1 \times W_3 + \partial LO_2 \times W_2 \quad \partial LO_2 \times W_3]$$



Correlation



$$\frac{\partial L}{\partial \boldsymbol{I}} = \text{Correlation}(W_{pad}, \text{LO}_{flip})$$

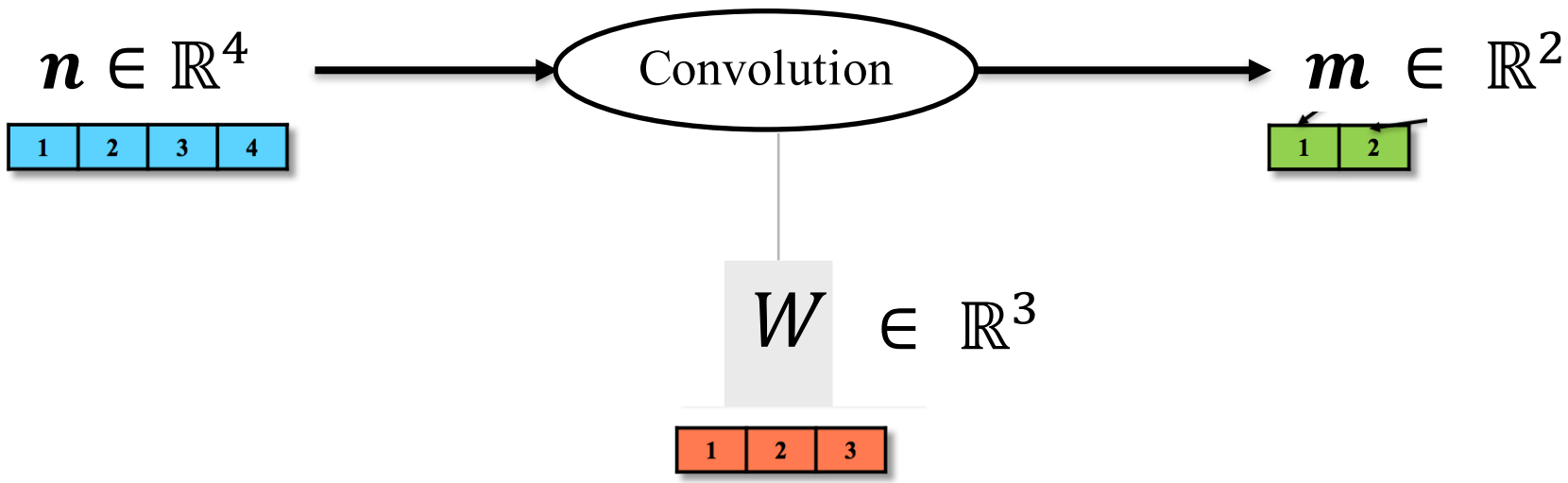
Building Blocks – Convolution – Forward

```
require 'nn';
n = torch.rand(4):reshape(1,1,4)
print(n)
```

```
(1,.,.) =
  0.3347  0.5901  0.7132  0.3187
[torch.DoubleTensor of size 1x1x4]
```

```
conv = nn.SpatialConvolutionMM(1,1,3,1)
conv.bias:fill(0)
m = conv:forward(n)
print(m)
```

```
(1,.,.) =
  0.1897  0.1130
[torch.DoubleTensor of size 1x1x2]
```



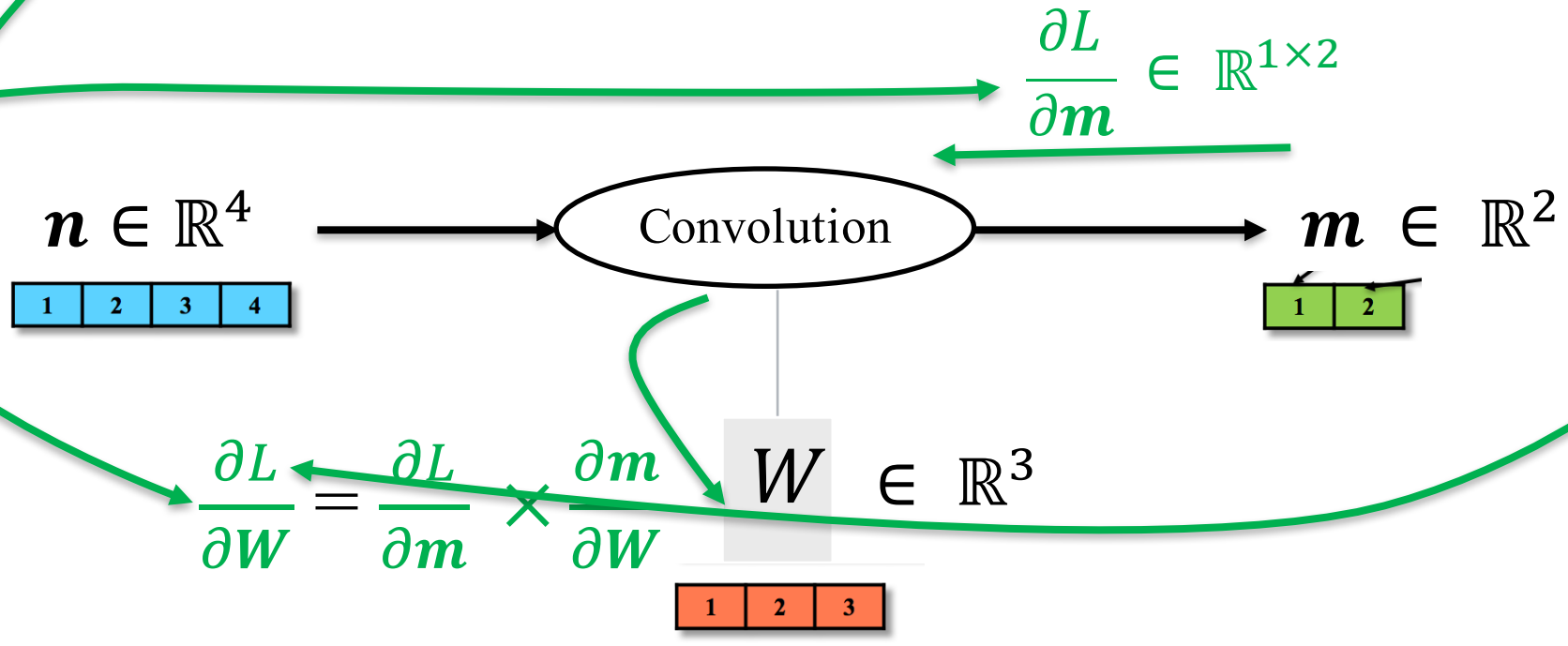
Building Blocks – Convolution – Backward

```
nextgrad=torch.rand(2):reshape(1,1,2)
conv:backward(n, nextgrad)
print(conv.gradWeight)
```

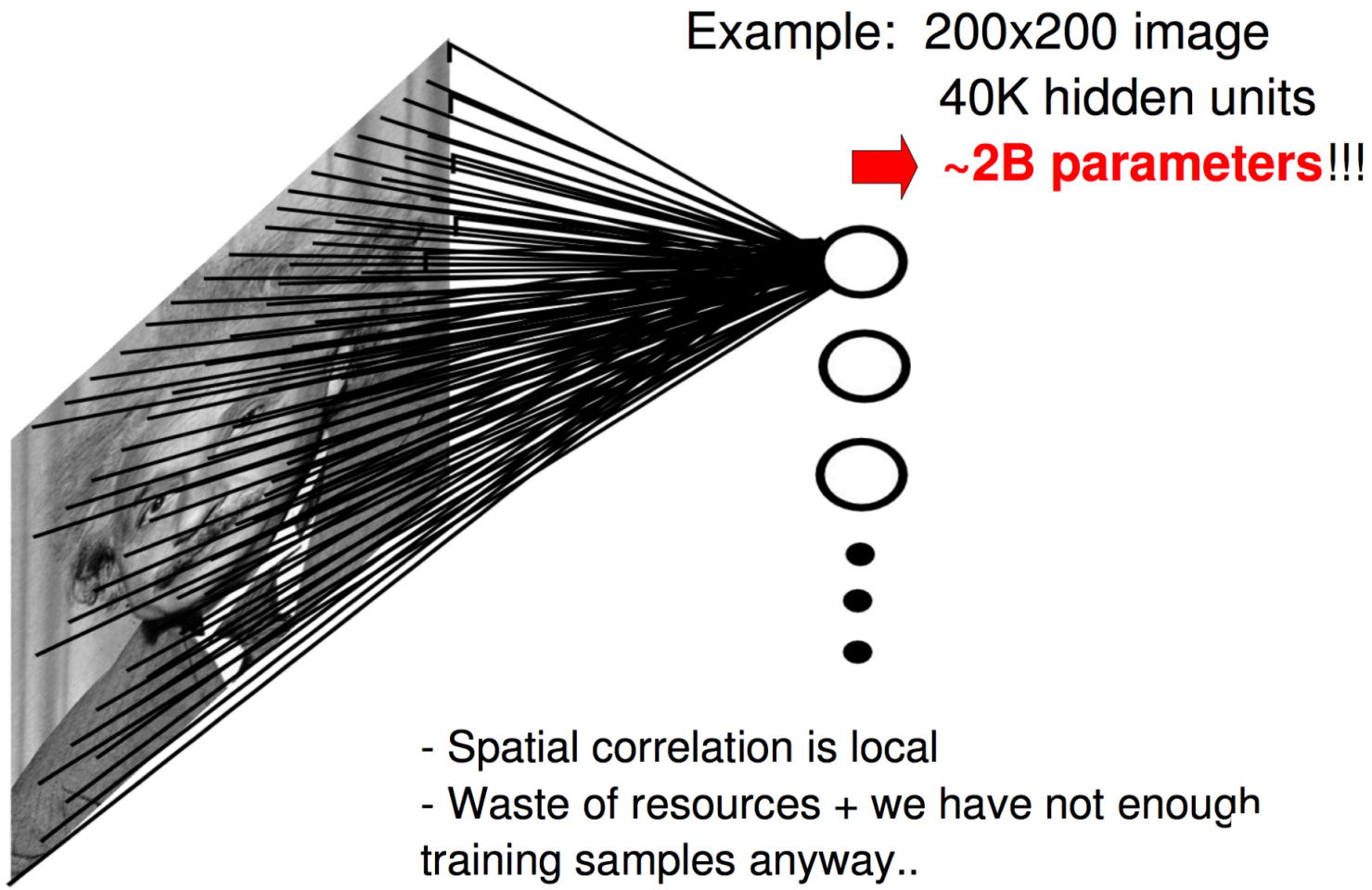
```
 0.6464   0.8428   0.5257
[torch.DoubleTensor of size 1x3]
```

```
convback = nn.SpatialConvolutionMM(1,1,2,1)
convback.bias:fill(0)
convback.weight:copy(nextgrad:reshape(1,2))
gradWeight = convback:forward(n)
print(gradWeight)
```

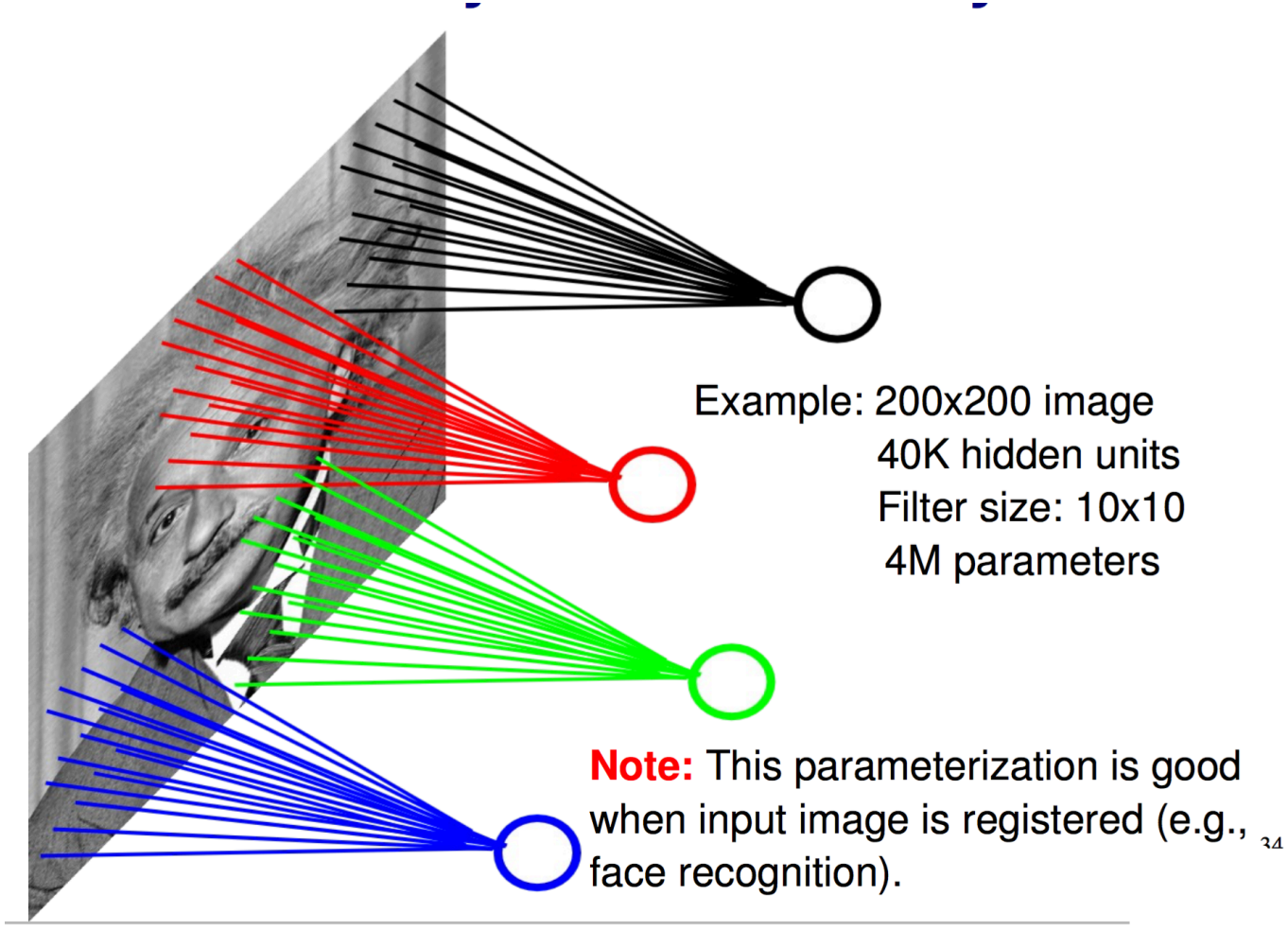
```
(1,.,.) =
  0.6464   0.8428   0.5257
[torch.DoubleTensor of size 1x1x3]
```



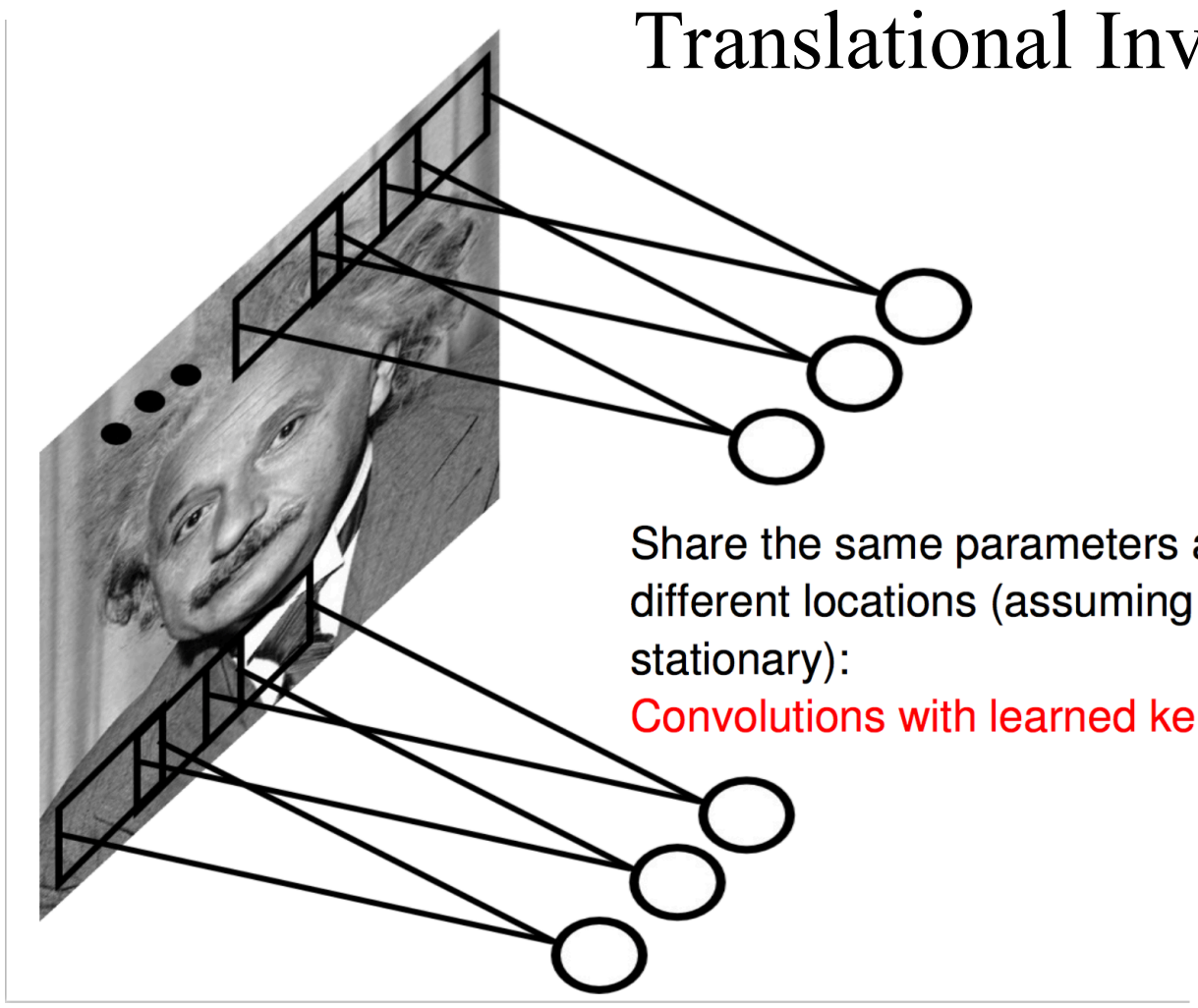
Building Blocks - Convolution



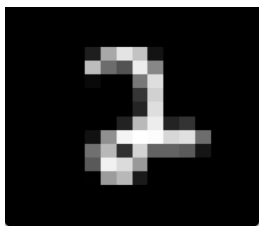
Building Blocks - Convolution



Building Blocks - Convolution



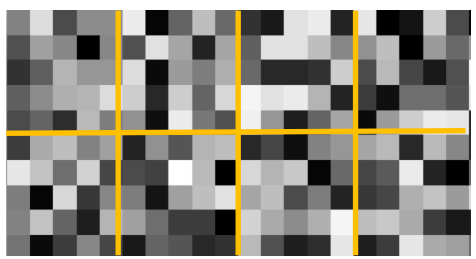
Building Blocks - Convolution



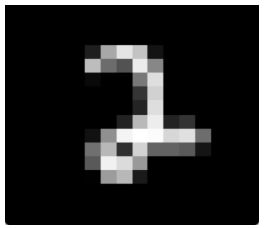
$\boldsymbol{n} \in \mathbb{R}^{16\times16}$ → Convolution → $\boldsymbol{m} \in \mathbb{R}^{8\times12\times12}$

$W \in \mathbb{R}^{1\times8\times5\times5}$

$=$



Building Blocks - Convolution



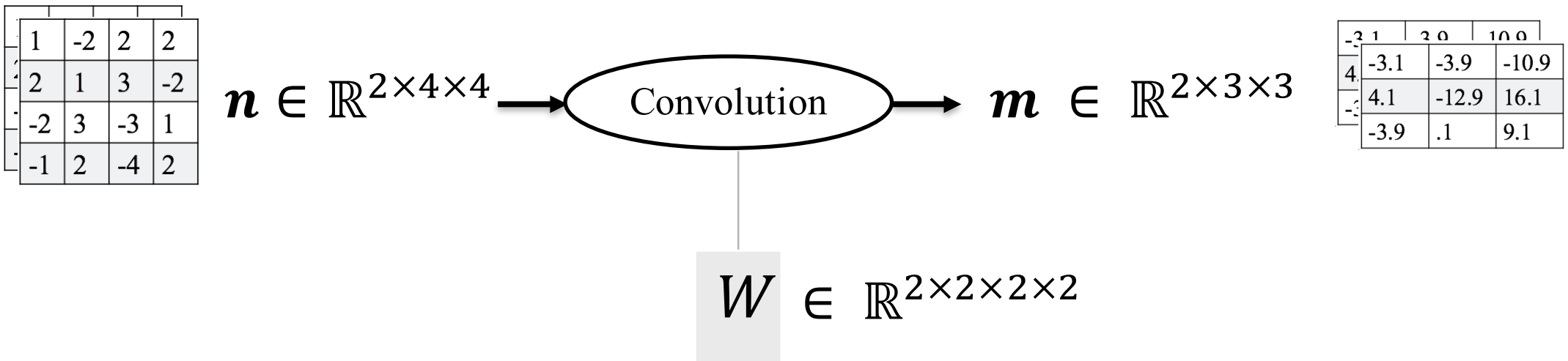
$$\boldsymbol{n} \in \mathbb{R}^{16\times16} \rightarrow \text{Convolution} \rightarrow \boldsymbol{m} \in \mathbb{R}^{8\times12\times12}$$

$$W$$

```
> conv = nn.SpatialConvolutionMM(1, 8, 5, 5)
> m = conv:forward(n)
> =n:size()
  1
 16
 16
[torch.LongStorage of size 3]

> =m:size()
  8
 12
 12
[torch.LongStorage of size 3]
```

```
> =conv.weight:size()
  8
 25
[torch.LongStorage of size 2]

> =conv.bias:size()
 8
[torch.LongStorage of size 1]
```

Building Blocks - Convolution



Building Blocks - Convolution

Image I = 2 x 4 x 4

I[1, :, :]

1	-2	2	2
2	1	3	-2
-2	3	-3	1
-1	2	-4	2

I[2, :, :]

3	0	0	0
-2	-2	1	-1
2	-1	3	1
5	-2	0	1

Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

W[1, 1, :, :]

1	-2
-2	1

W[2, 1, :, :]

3	1
2	2

W[1, 2, :, :]

1	0
0	1

W[2, 2, :, :]

0	0
0	4

0.1
0.2

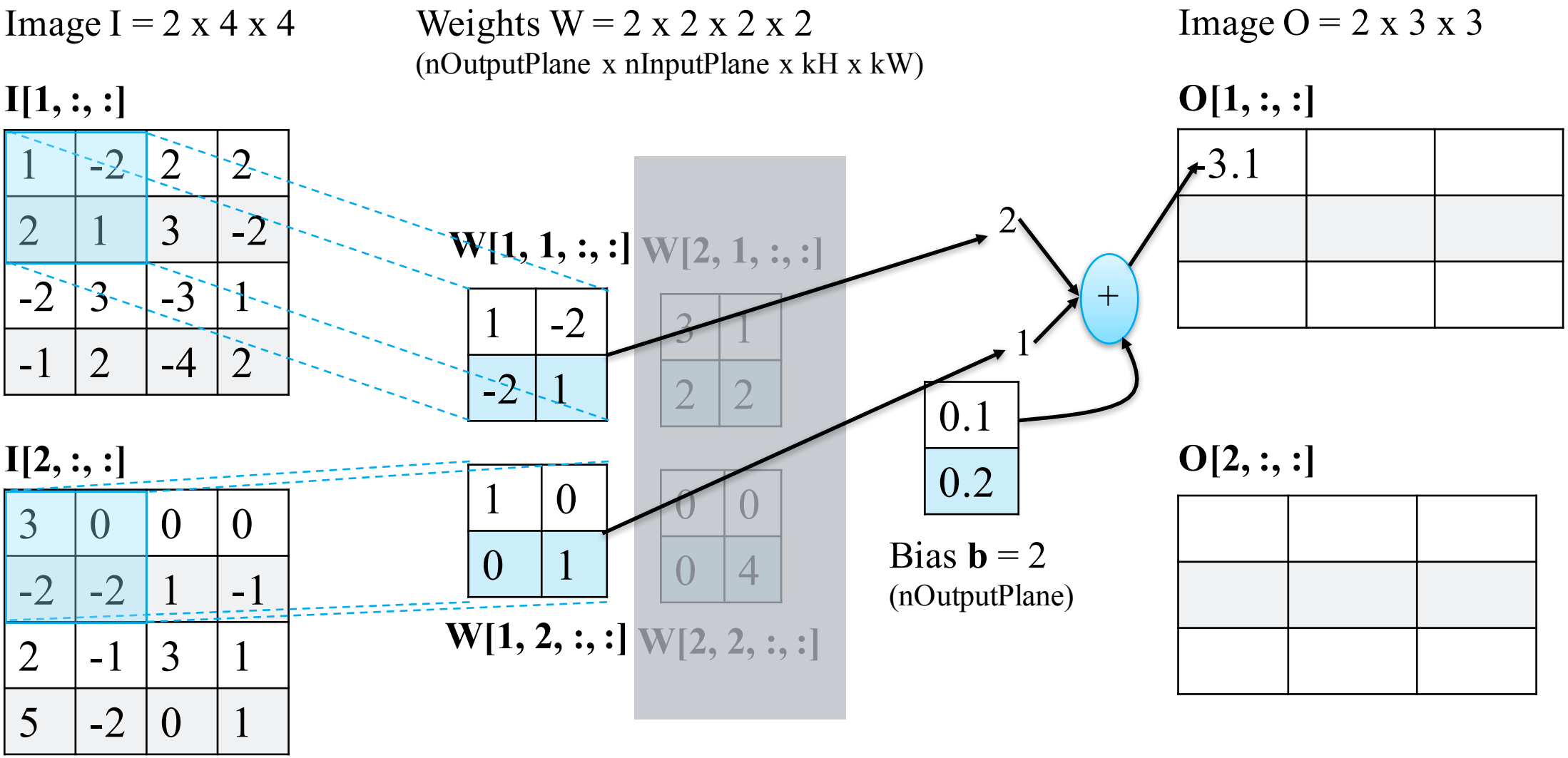
Bias **b** = 2
(nOutputPlane)

Image O = 2 x 3 x 3

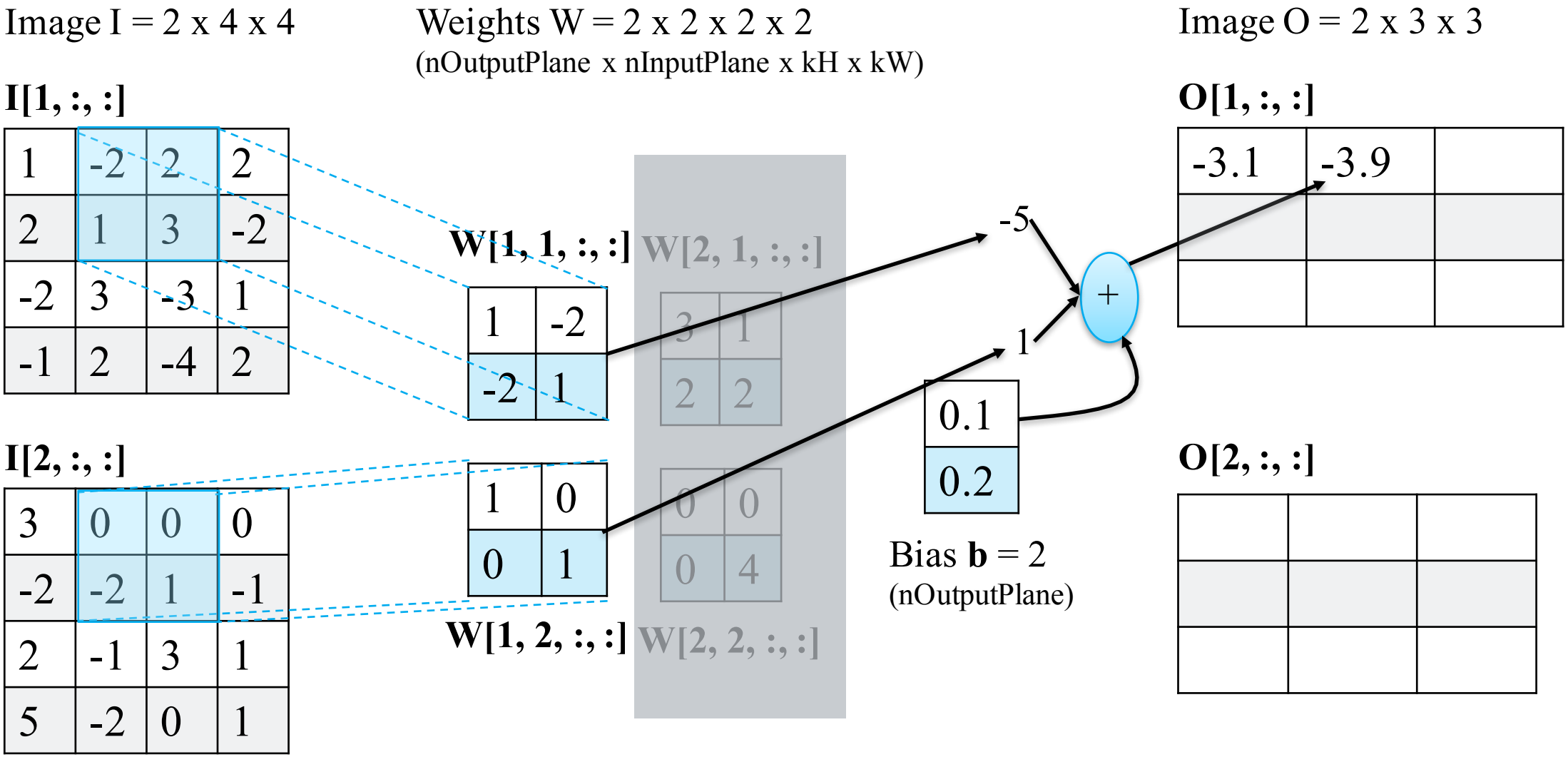
O[1, :, :]

O[2, :, :]

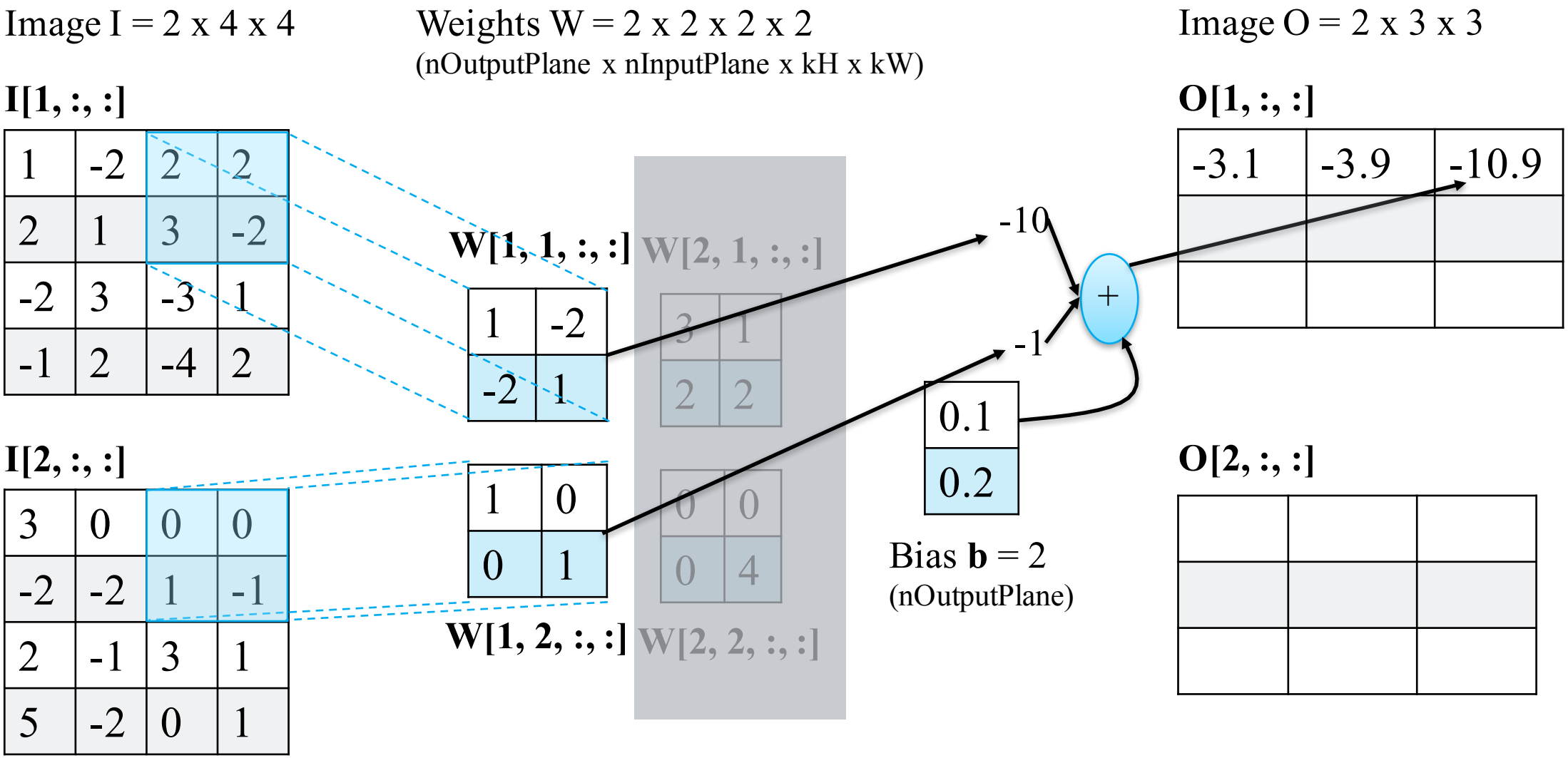
Building Blocks - Convolution



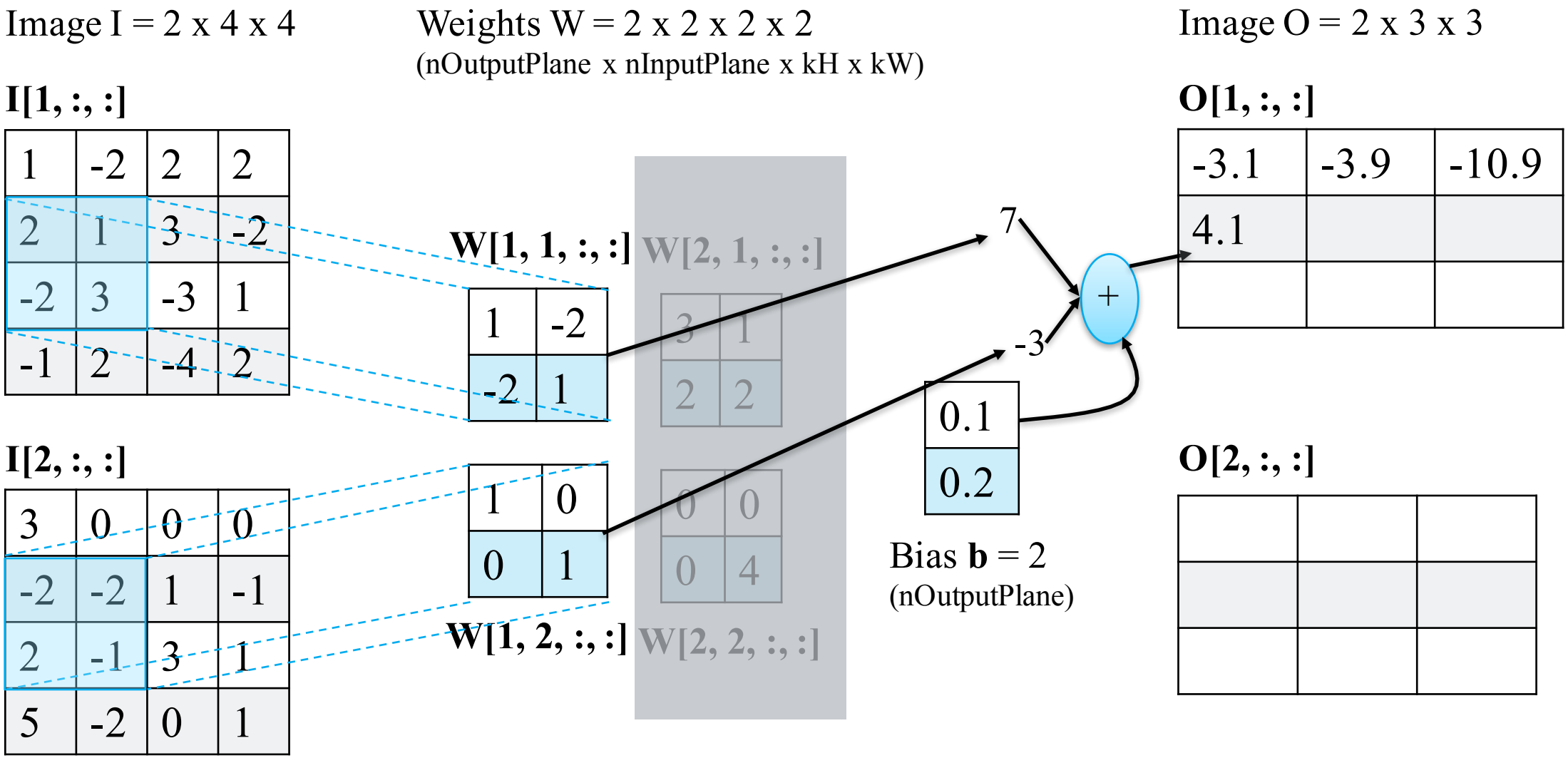
Building Blocks - Convolution



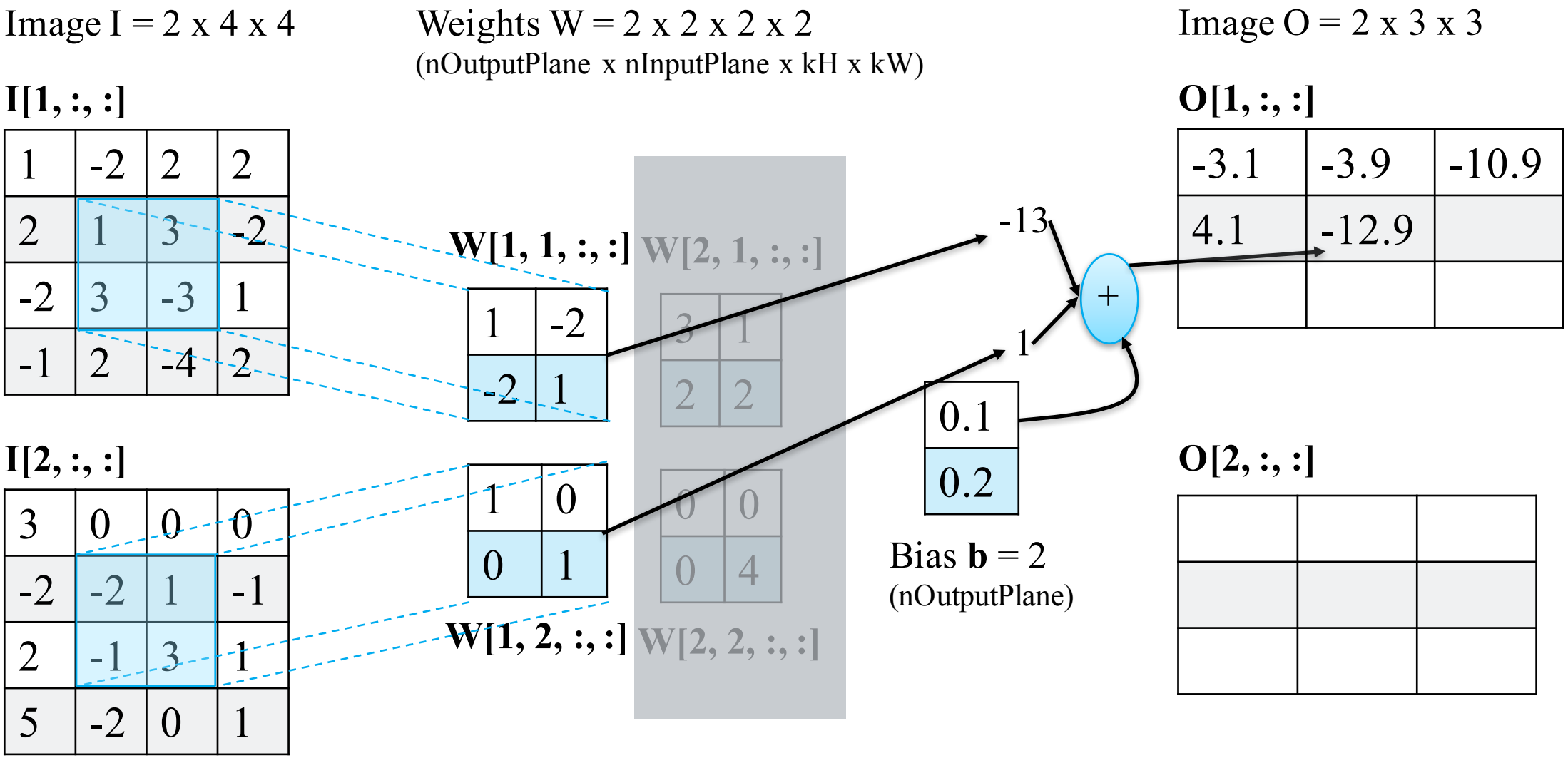
Building Blocks - Convolution



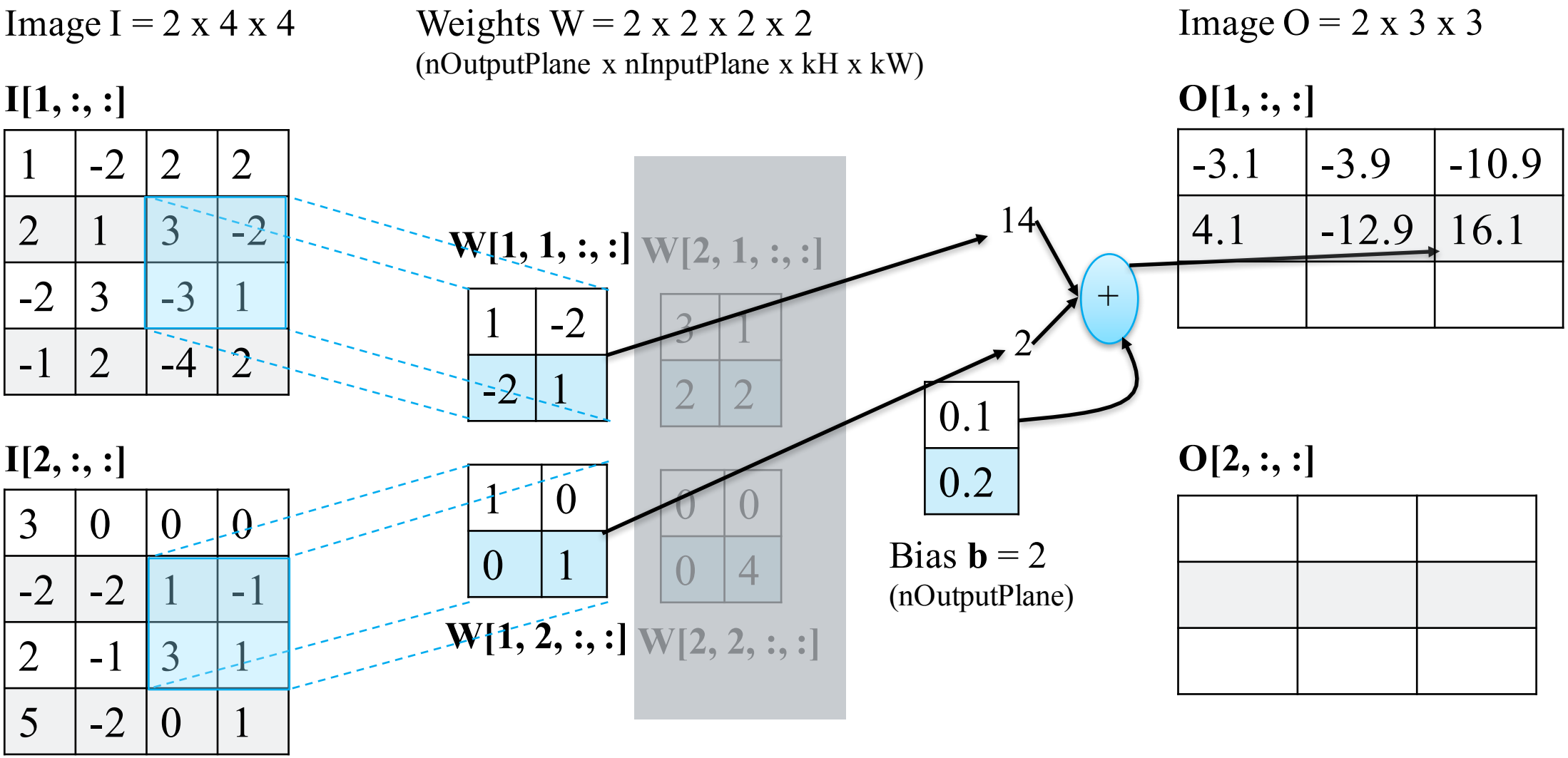
Building Blocks - Convolution



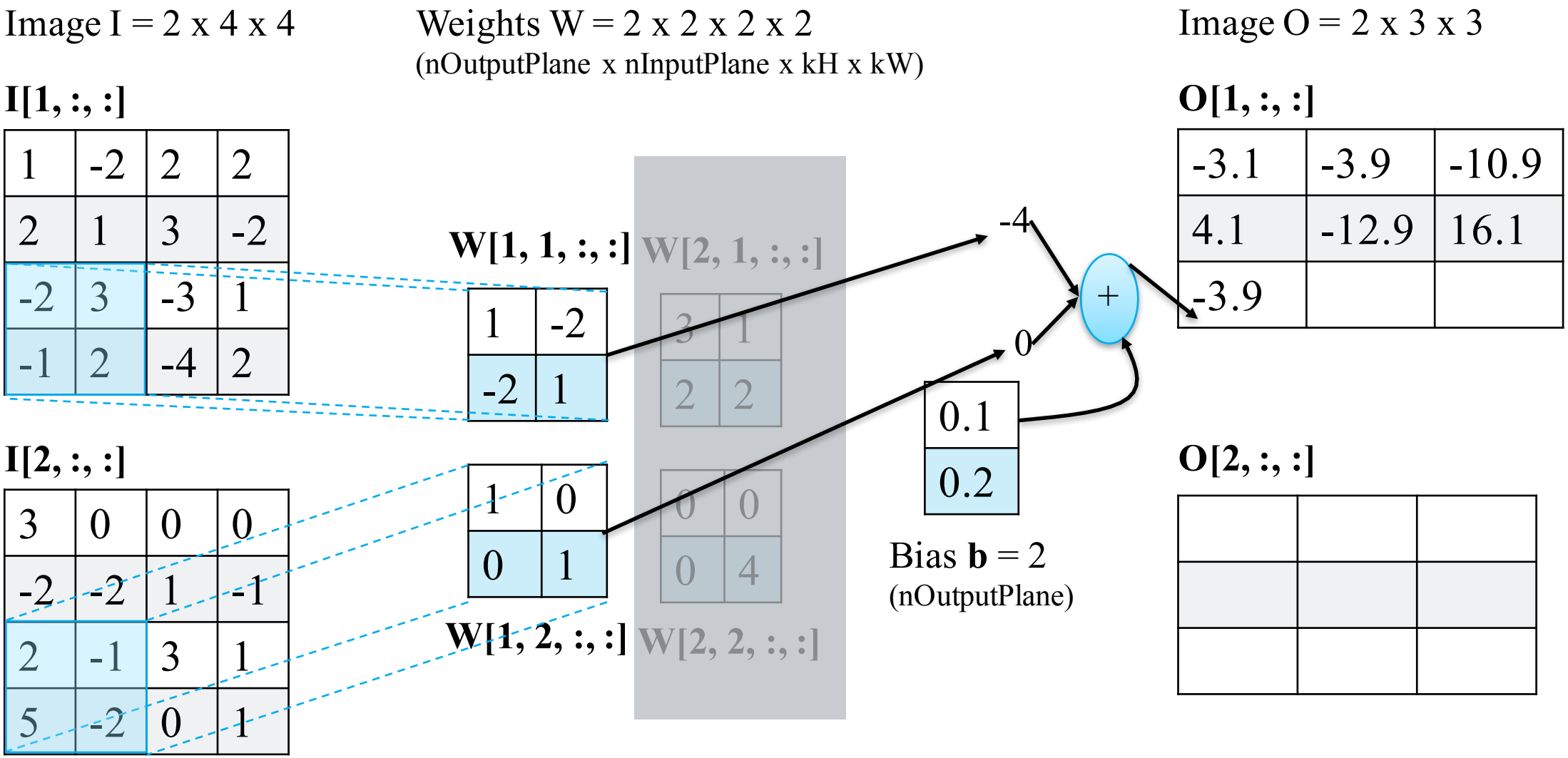
Building Blocks - Convolution



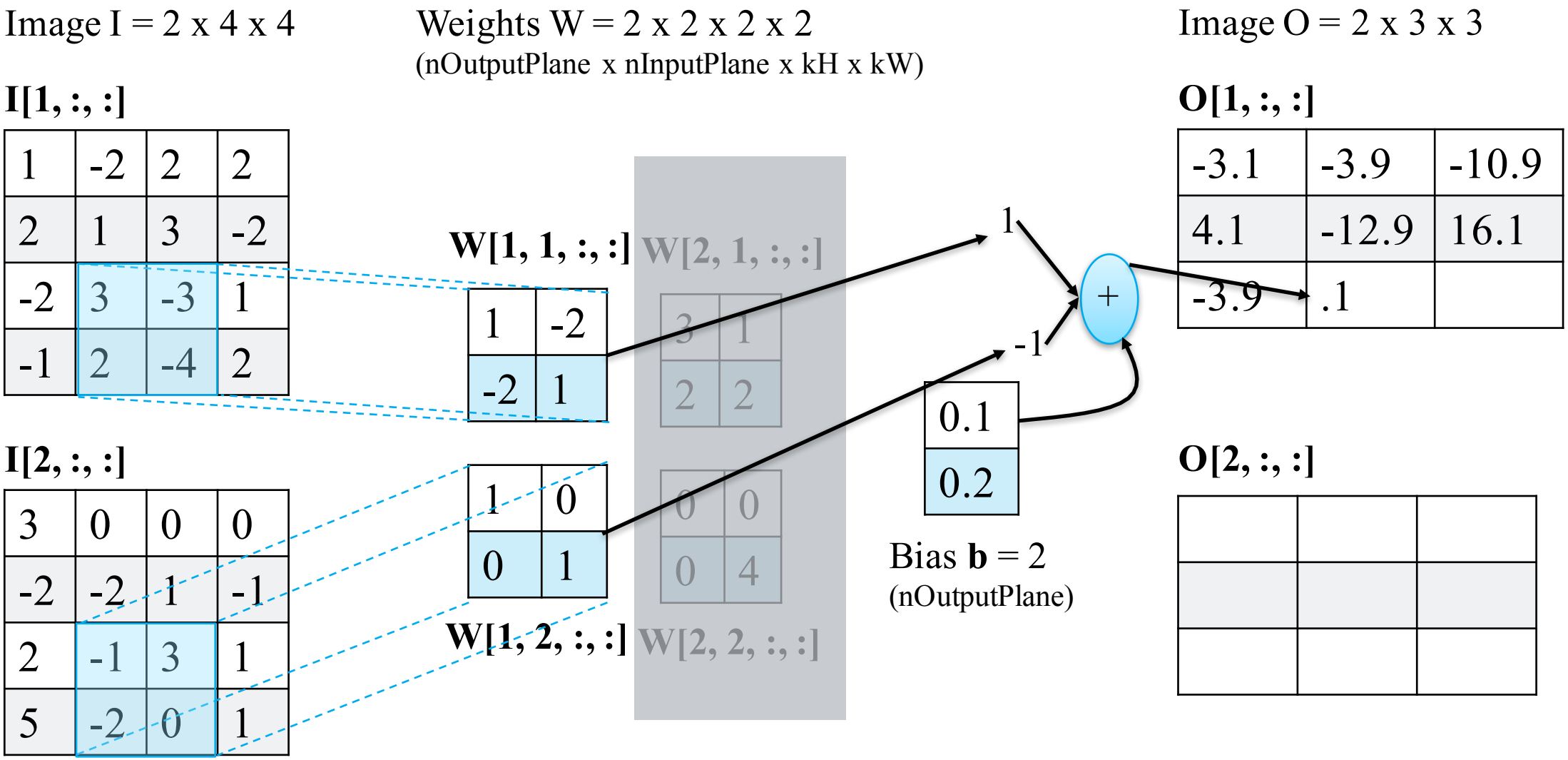
Building Blocks - Convolution



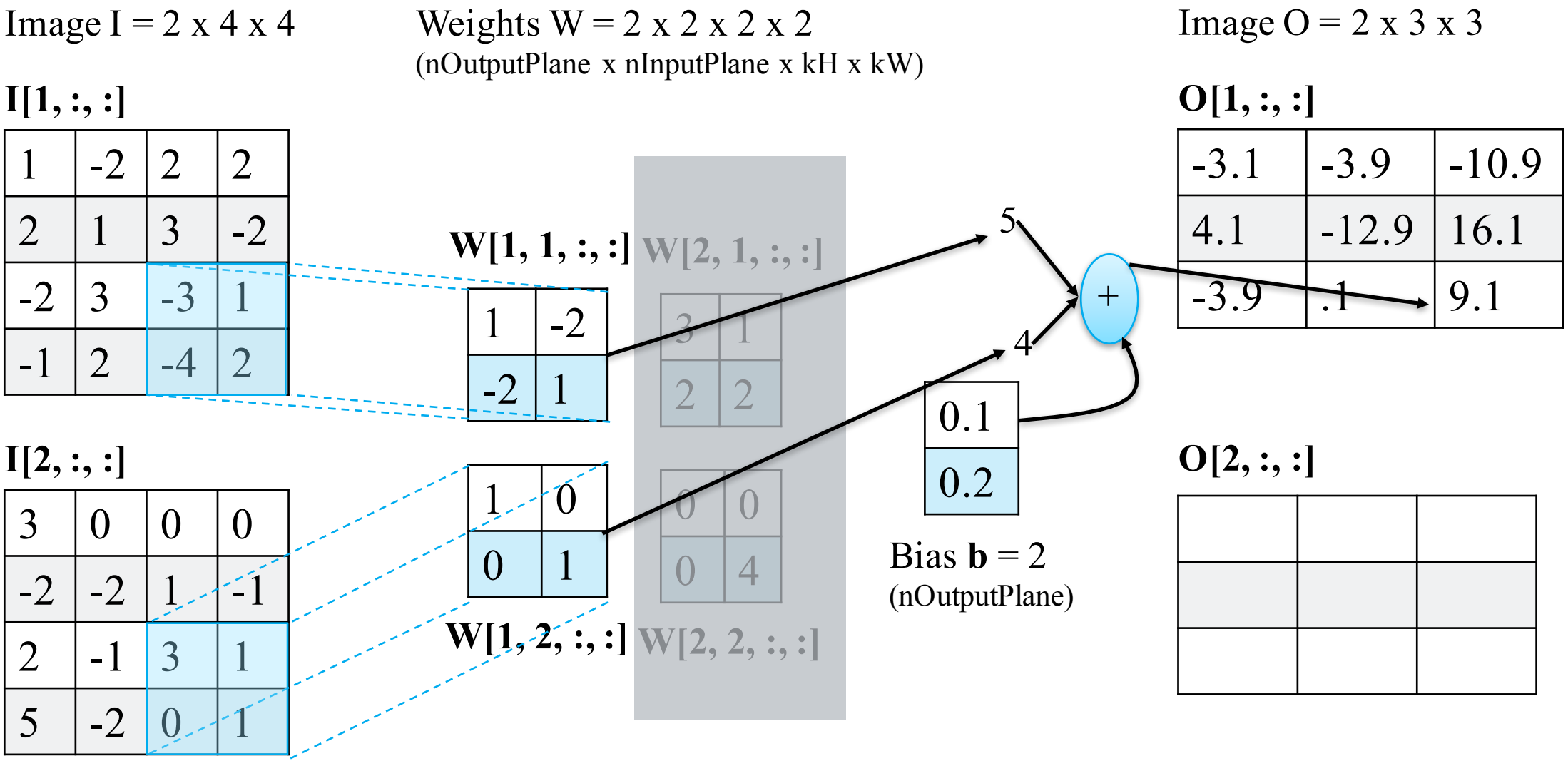
Building Blocks - Convolution



Building Blocks - Convolution



Building Blocks - Convolution



Building Blocks - Convolution

Image I = 2 x 4 x 4

I[1, :, :]

1	-2	2	2
2	1	3	-2
-2	3	-3	1
-1	2	-4	2

I[2, :, :]

3	0	0	0
-2	-2	1	-1
2	-1	3	1
5	-2	0	1

Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

W[1, 1, :, :]

1	-2
-2	1

W[2, 1, :, :]

3	1
2	2

W[1, 2, :, :]

1	0
0	1

W[2, 2, :, :]

0	0
0	4

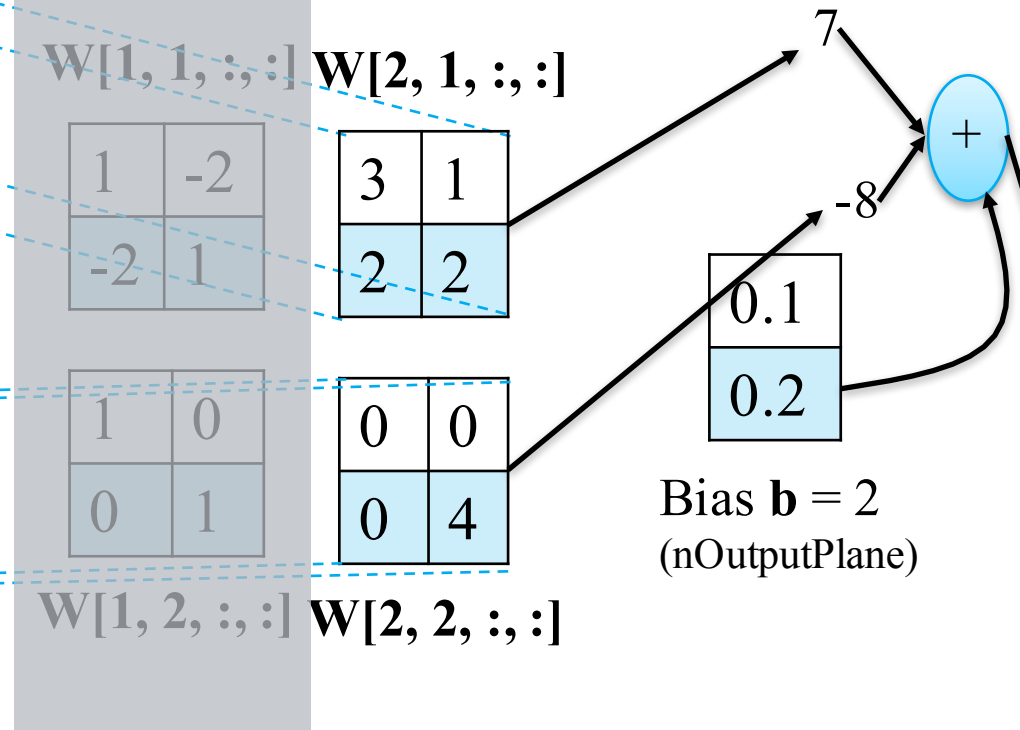


Image O = 2 x 3 x 3

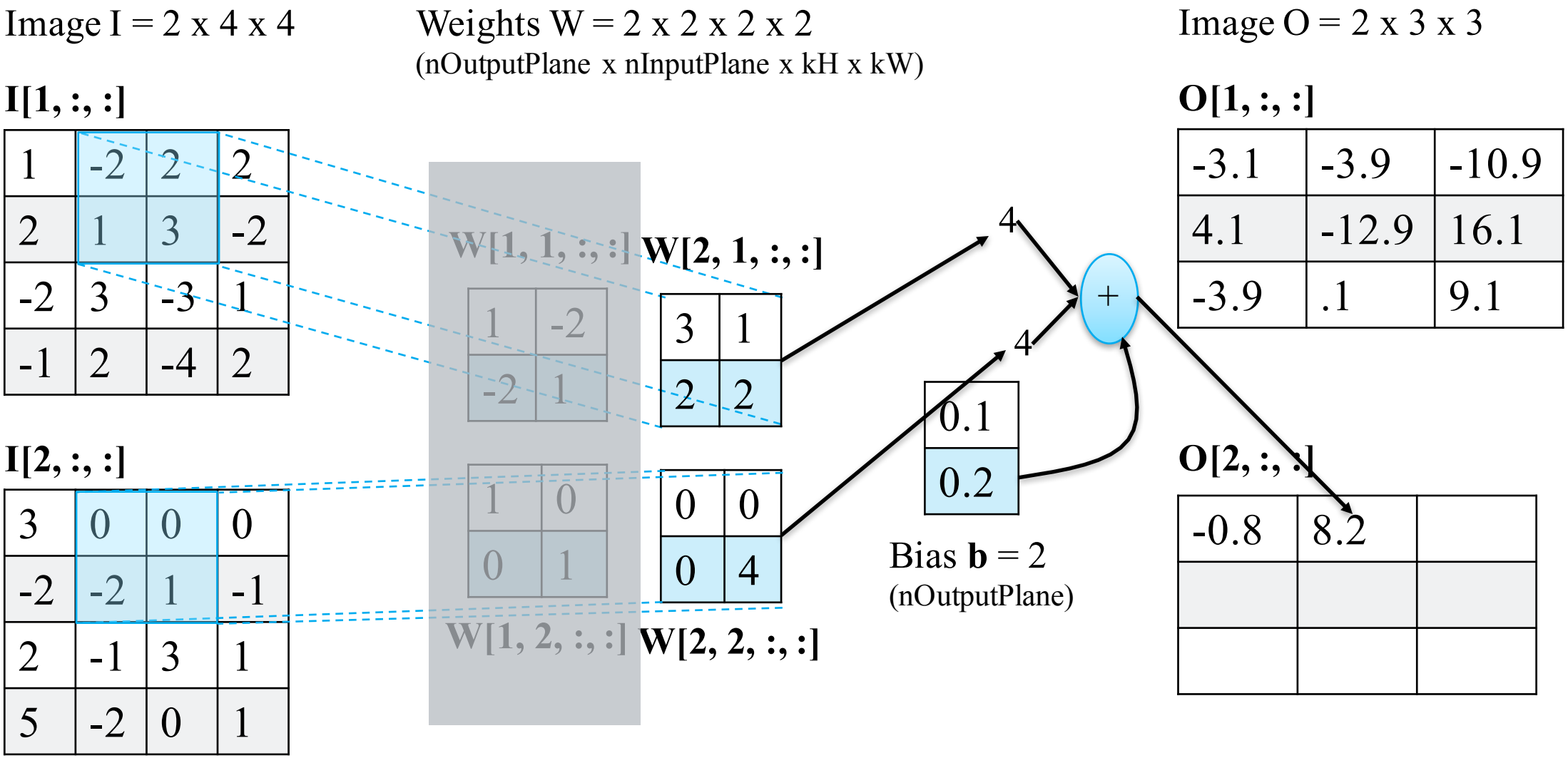
O[1, :, :]

-3.1	-3.9	-10.9
4.1	-12.9	16.1
-3.9	.1	9.1

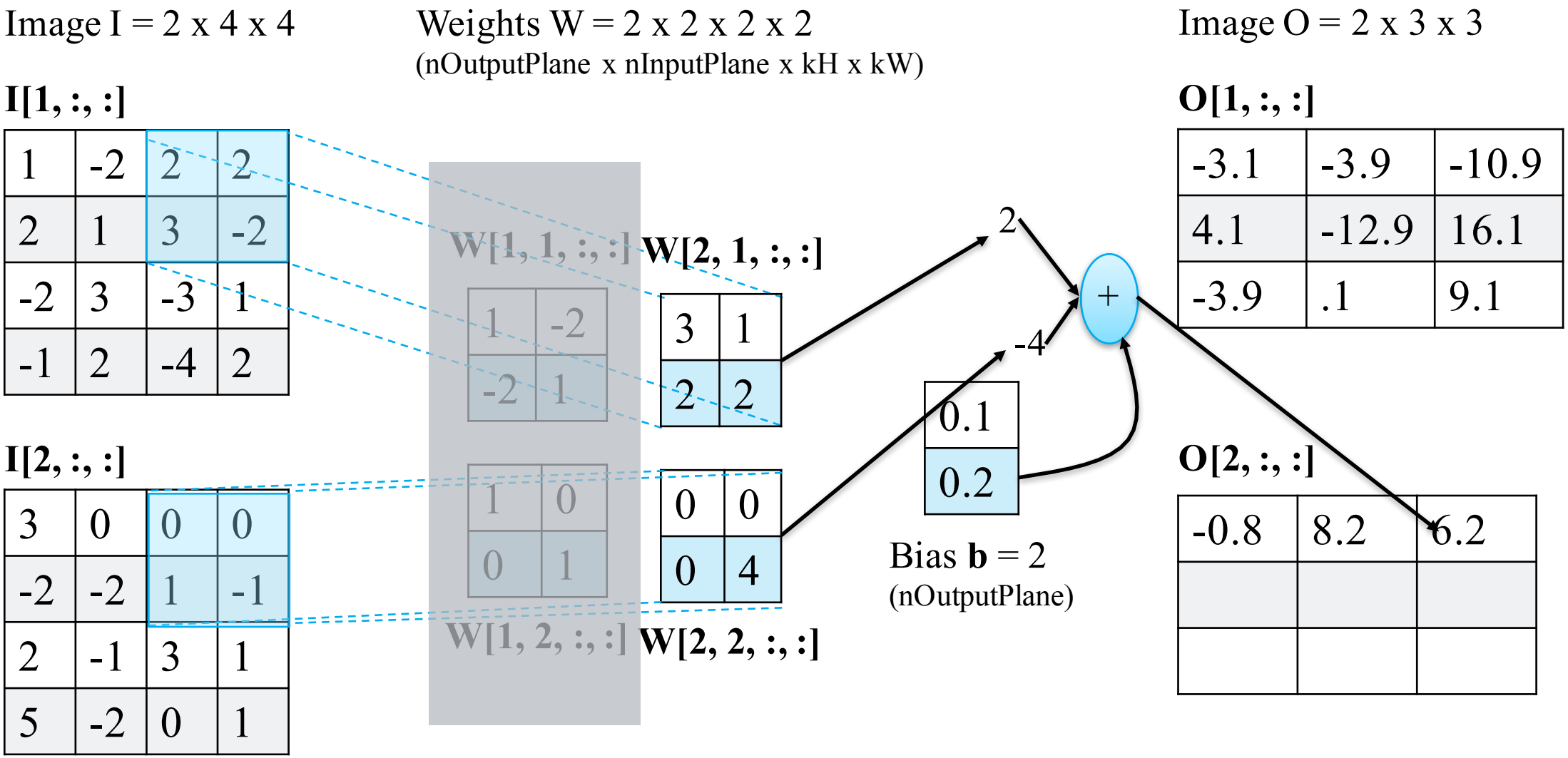
O[2, :, :]

-0.8		

Building Blocks - Convolution



Building Blocks - Convolution



Building Blocks - Convolution

Image I = 2 x 4 x 4

I[1, :, :]

1	-2	2	2
2	1	3	-2
-2	3	-3	1
-1	2	-4	2

I[2, :, :]

3	0	0	0
-2	-2	1	-1
2	-1	3	1
5	-2	0	1

Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

W[1, 1, :, :]

1	-2
-2	1

W[2, 1, :, :]

3	1
2	2

W[1, 2, :, :]

1	0
0	1

W[2, 2, :, :]

0	0
0	4

9

-4

0.1
0.2

Bias **b** = 2
(nOutputPlane)

+

Image O = 2 x 3 x 3

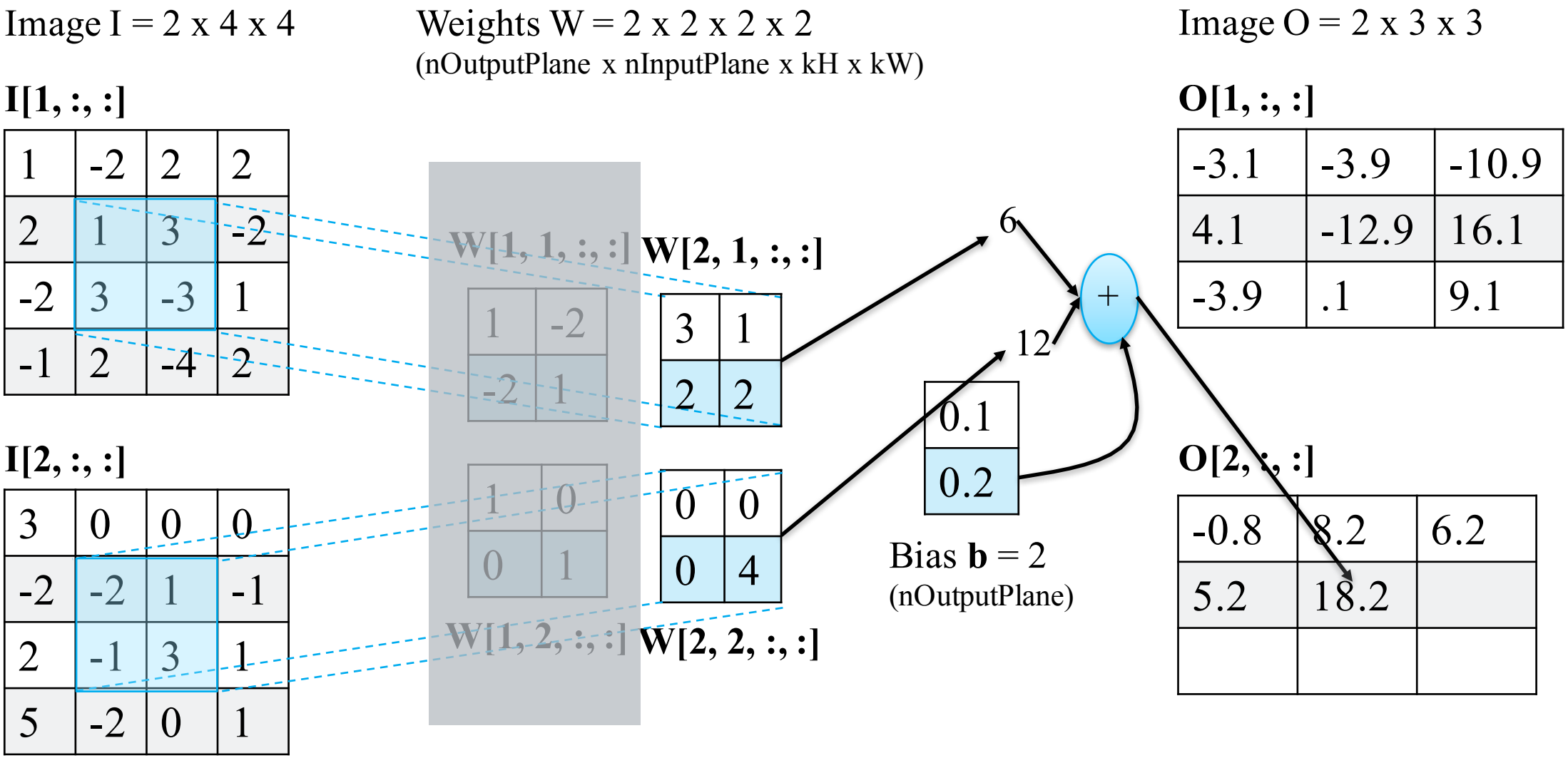
O[1, :, :]

-3.1	-3.9	-10.9
4.1	-12.9	16.1
-3.9	.1	9.1

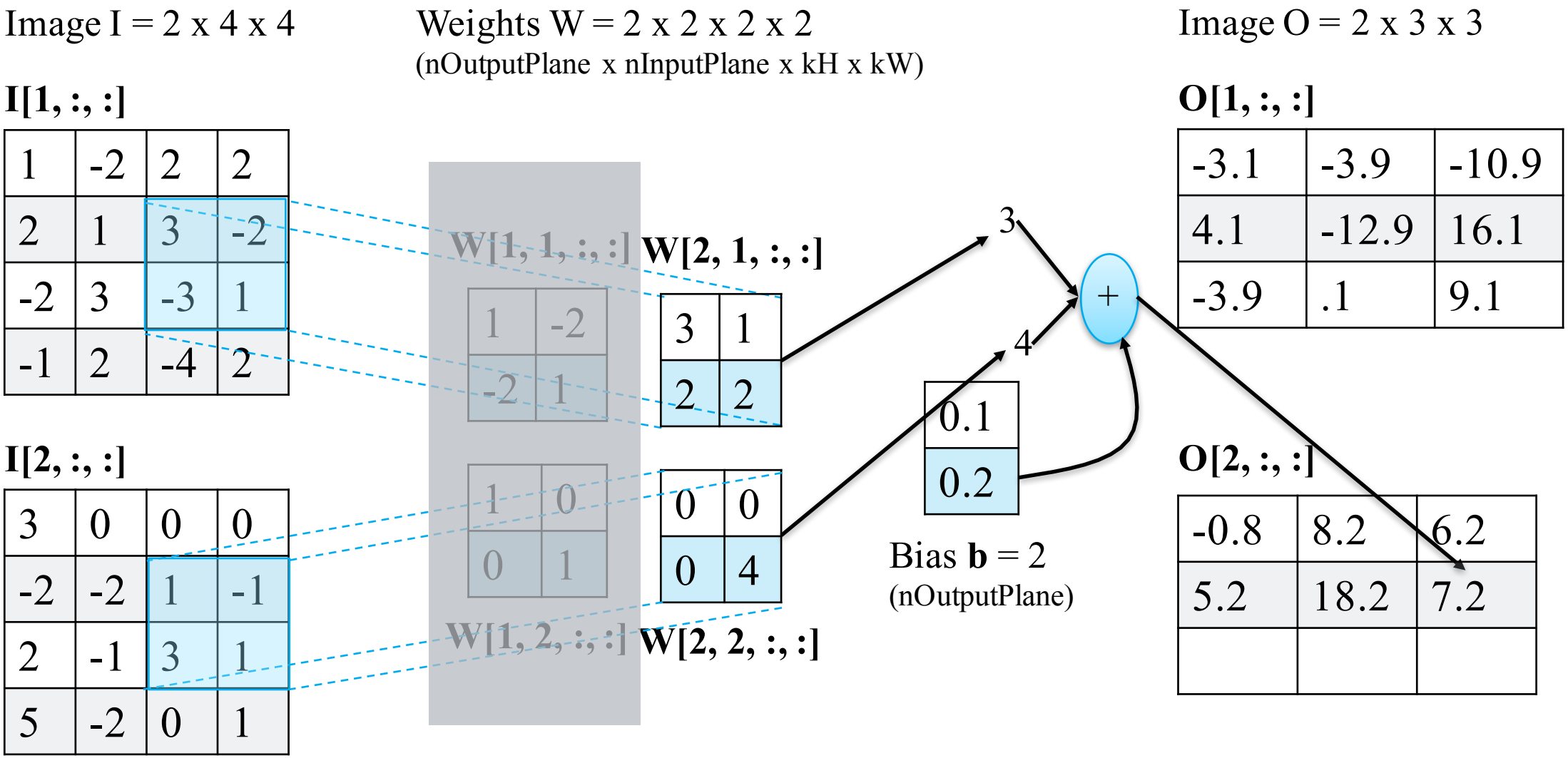
O[2, :, :]

-0.8	8.2	6.2
5.2		

Building Blocks - Convolution



Building Blocks - Convolution



Building Blocks - Convolution

Image I = 2 x 4 x 4

I[1, :, :]

1	-2	2	2
2	1	3	-2
-2	3	-3	1
-1	2	-4	2

I[2, :, :]

3	0	0	0
-2	-2	1	-1
2	-1	3	1
5	-2	0	1

Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

W[1, 1, :, :]

1	-2
-2	1

W[2, 1, :, :]

3	1
2	2

W[1, 2, :, :]

1	0
0	1

W[2, 2, :, :]

0	0
0	4

-1

-8

+

0.1
0.2

Bias **b** = 2
(nOutputPlane)

Image O = 2 x 3 x 3

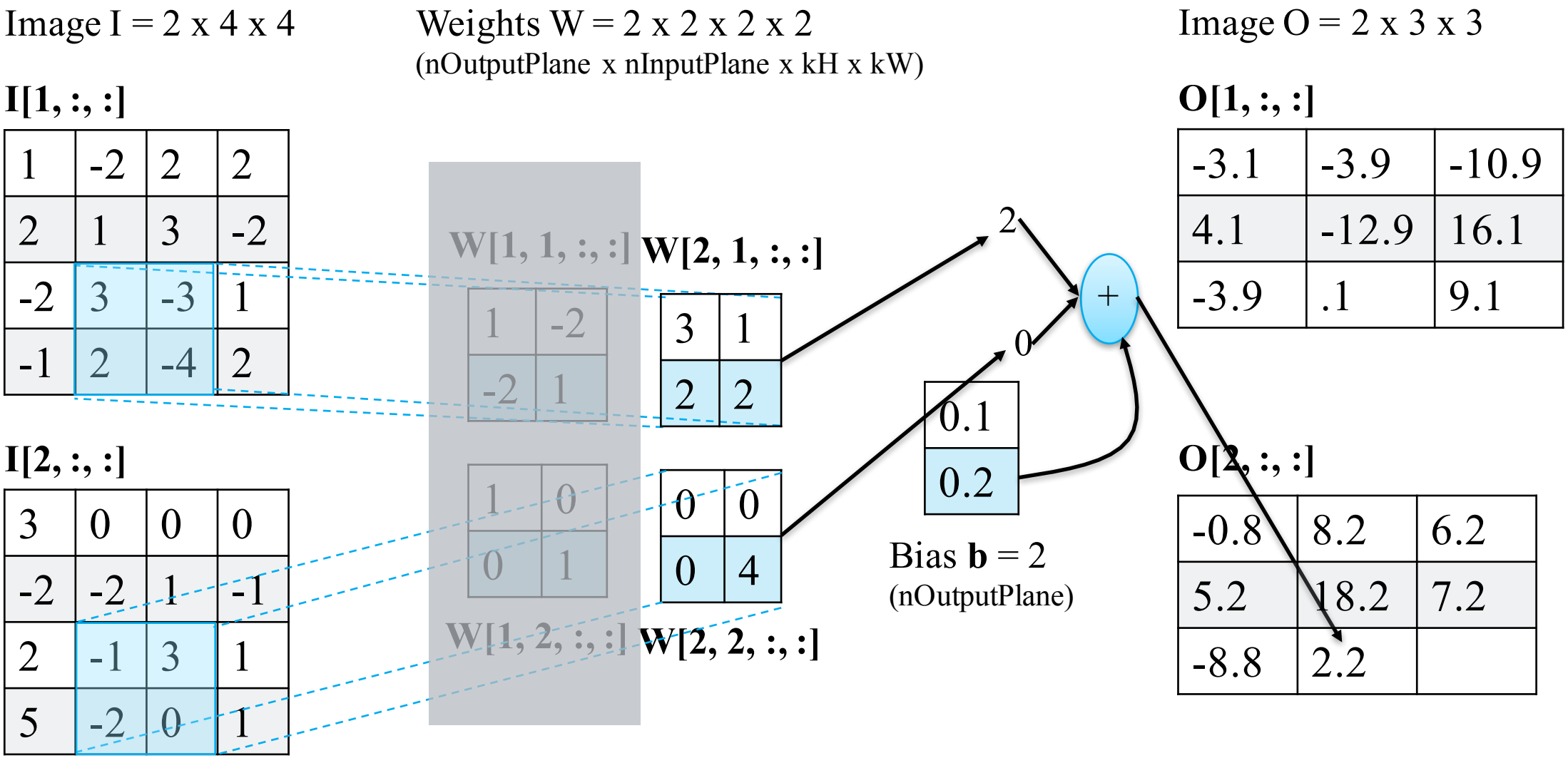
O[1, :, :]

-3.1	-3.9	-10.9
4.1	-12.9	16.1
-3.9	.1	9.1

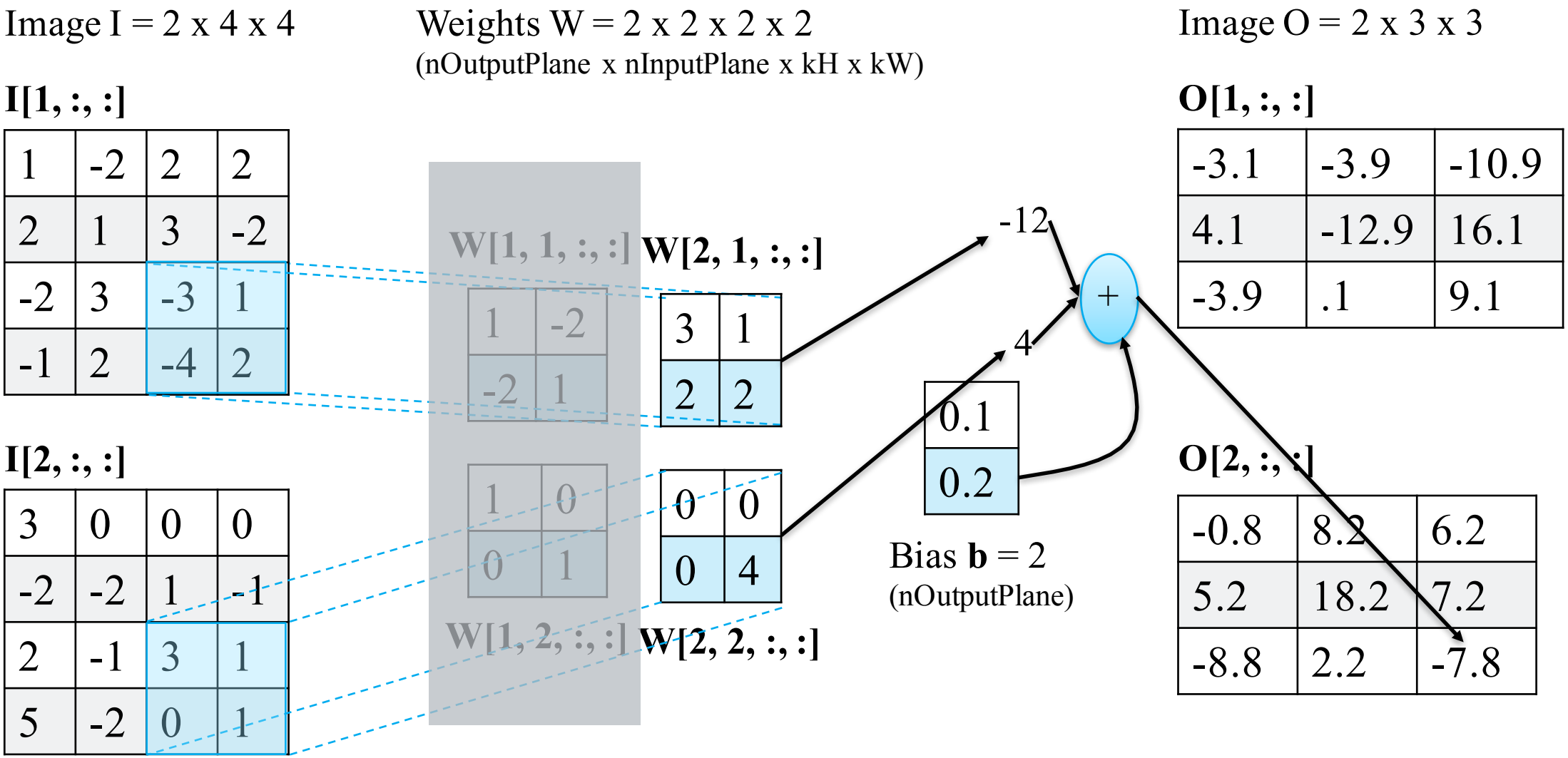
O[2, :, :]

-0.8	8.2	6.2
5.2	18.2	7.2
-8.8		

Building Blocks - Convolution



Building Blocks - Convolution



Building Blocks – Convolution – in Torch7

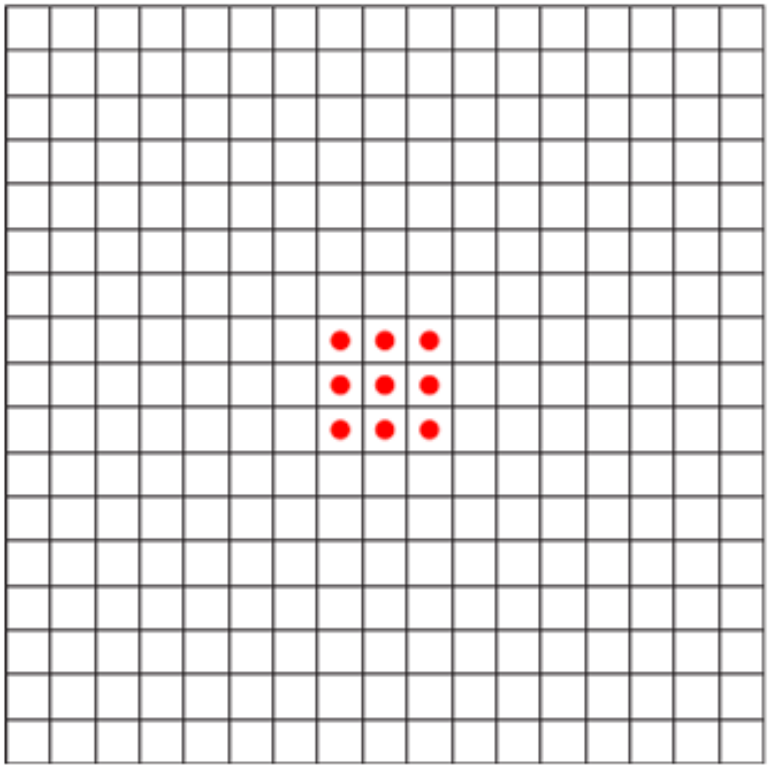
```
> I = torch.DoubleTensor({{{1,-2,2,2},{2,1,3,-2},{-2,3,-3,1},{-1,2,-4,2}}, {{3,0,0,0},{-2,-2,1,-1},{2,-1,3,1},{5,-2,0,1}}})
> conv = nn.SpatialConvolutionMM(2,2,2,2)
> conv.bias = torch.DoubleTensor({0.1, 0.2})
> conv.weight = torch.DoubleTensor({{1,-2,-2,1,1,0,0,1},{3,1,2,2,0,0,0,4}})
> O = conv:forward(I)
> =I
(1,.,.) =
  1 -2  2  2
  2  1  3 -2
 -2  3 -3  1
 -1  2 -4  2

(2,.,.) =
  3  0  0  0
 -2 -2  1 -1
  2 -1  3  1
  5 -2  0  1
[torch.DoubleTensor of size 2x4x4]

> =O
(1,.,.) =
   3.1000  -3.9000 -10.9000
   4.1000 -12.9000  16.1000
  -3.9000   0.1000   9.1000

(2,.,.) =
  -0.8000   8.2000   6.2000
   5.2000  18.2000   7.2000
  -8.8000   2.2000  -7.8000
[torch.DoubleTensor of size 2x3x3]
```

Building Blocks – Convolution – in Torch7

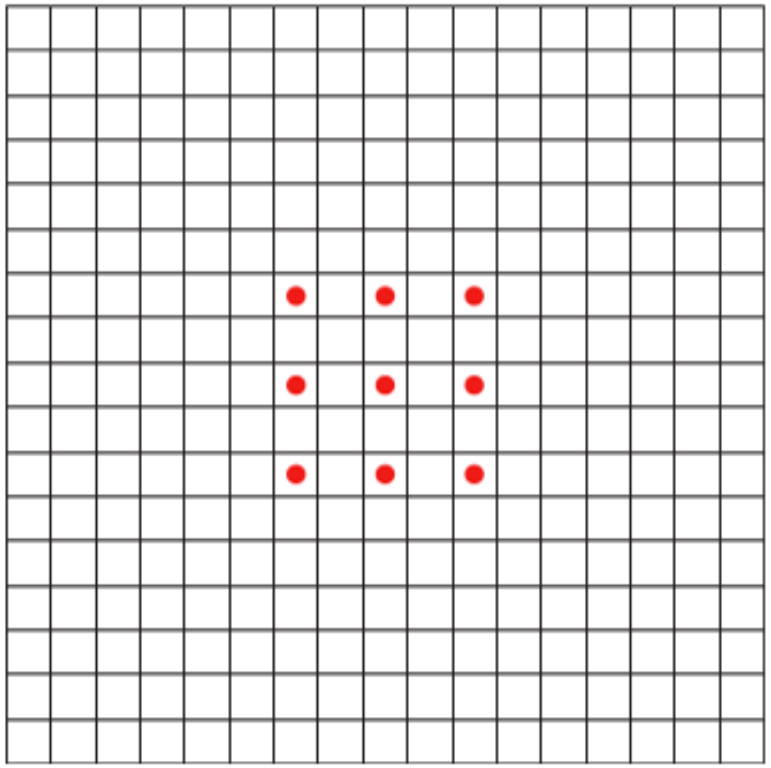
```
4  function SpatialConvolutionMM:__init(nInputPlane, nOutputPlane, kW, kH, dW, dH, padW, padH)
5     parent.__init(self)
6
7     dW = dW or 1
8     dH = dH or 1
9
10    self.nInputPlane = nInputPlane
11    self.nOutputPlane = nOutputPlane
12    self.kW = kW
13    self.kH = kH
14
15    self.dW = dW
16    self.dH = dH
17    self.padW = padW or 0
18    self.padH = padH or self.padW
19
20    self.weight = torch.Tensor(nOutputPlane, nInputPlane*kH*kW)
21    self.bias = torch.Tensor(nOutputPlane)
22    self.gradWeight = torch.Tensor(nOutputPlane, nInputPlane*kH*kW)
23    self.gradBias = torch.Tensor(nOutputPlane)
24
25    self:reset()
26 end
```

https://github.com/torch/nn/blob/master/SpatialConvolutionMM.lua

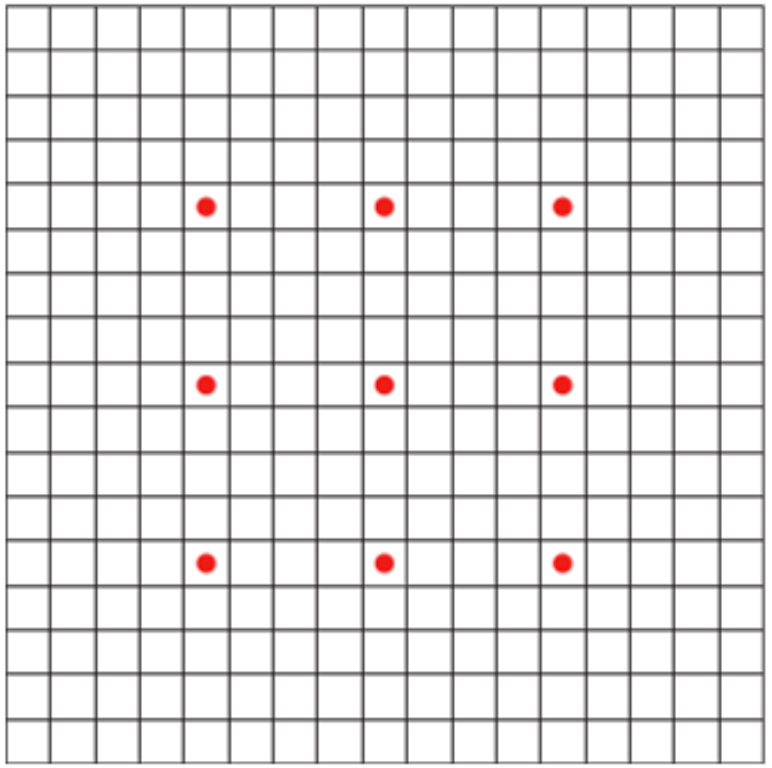
Aside: Dilated Convolution



(a)



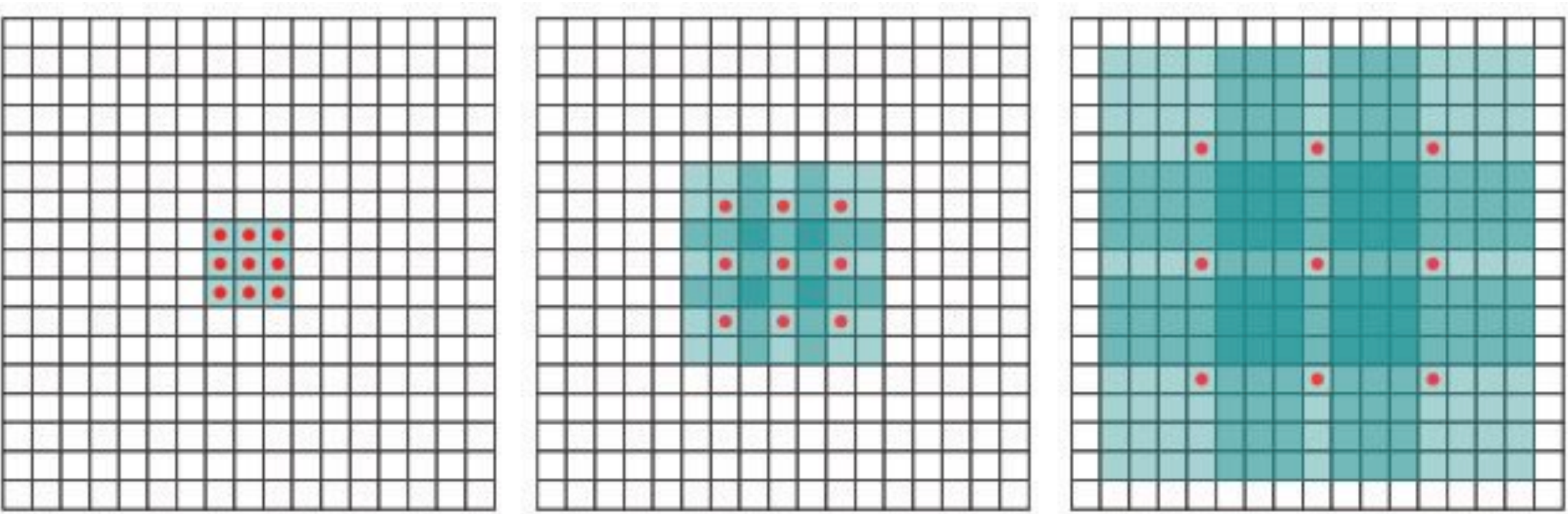
(b)



(c)

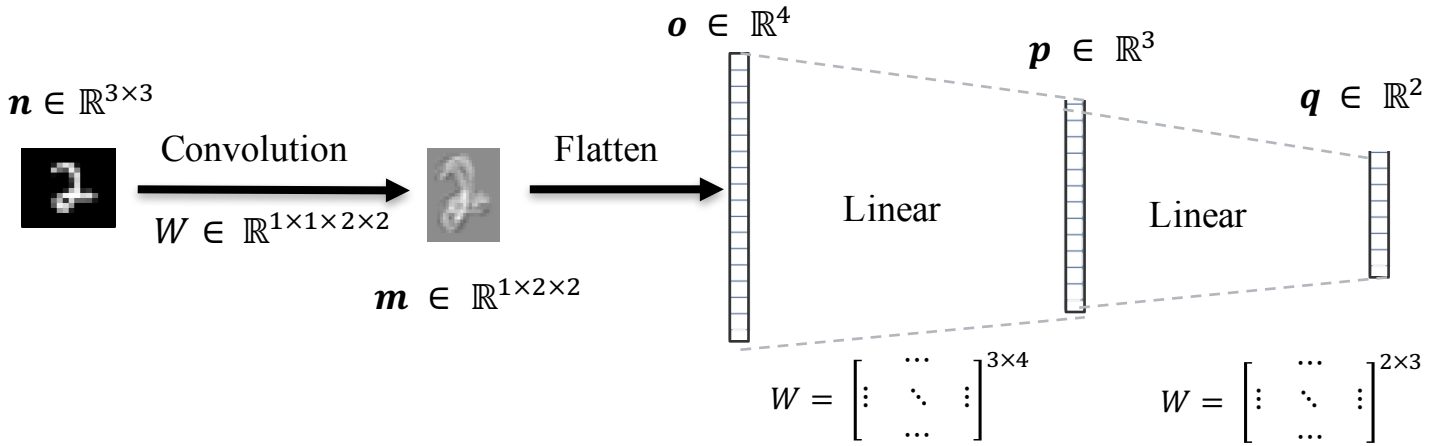
Aside: Dilated Convolution

nn.SpatialDilatedConvolution

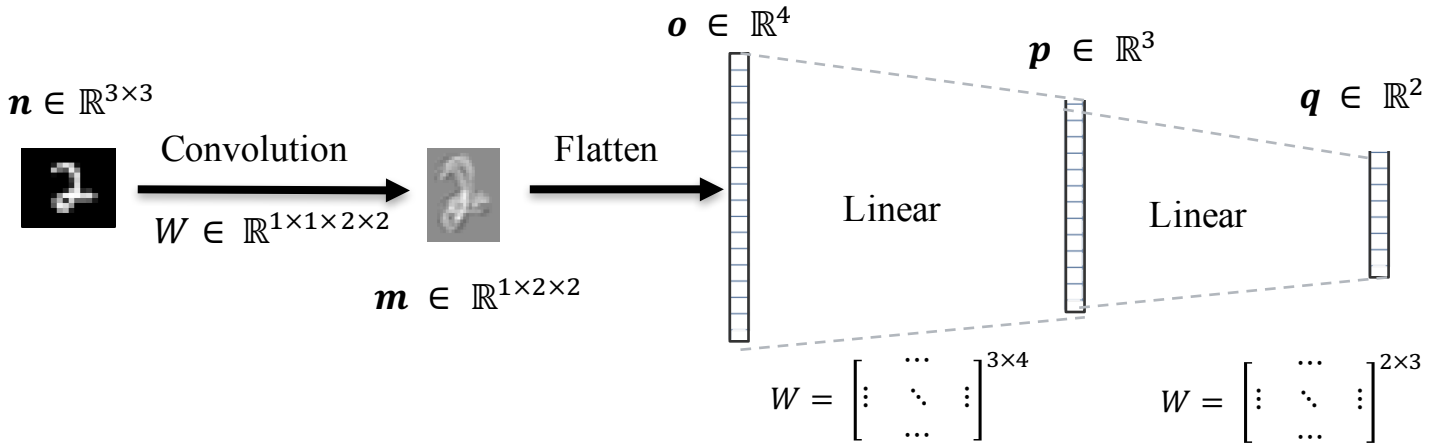


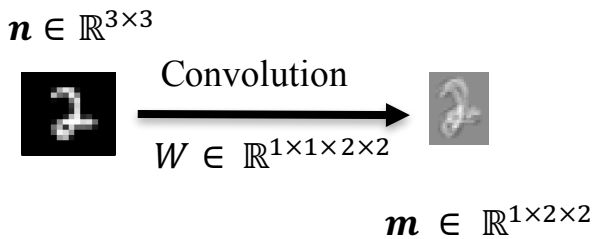
Multi-Scale Context Aggregation by Dilated Convolutions
Fisher Yu, Vladlen Koltun

Everything is a Convolution!

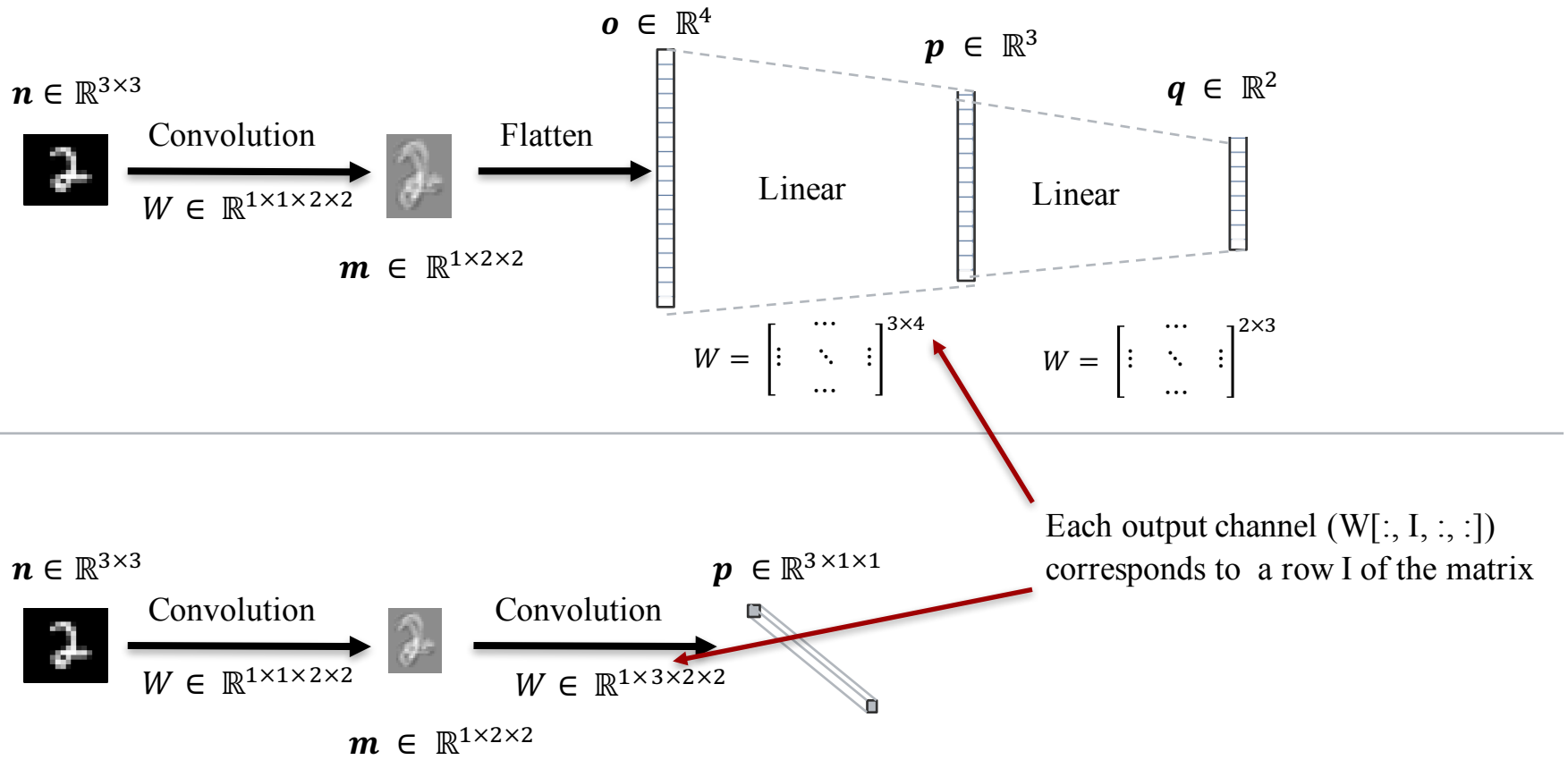


Everything is a Convolution!

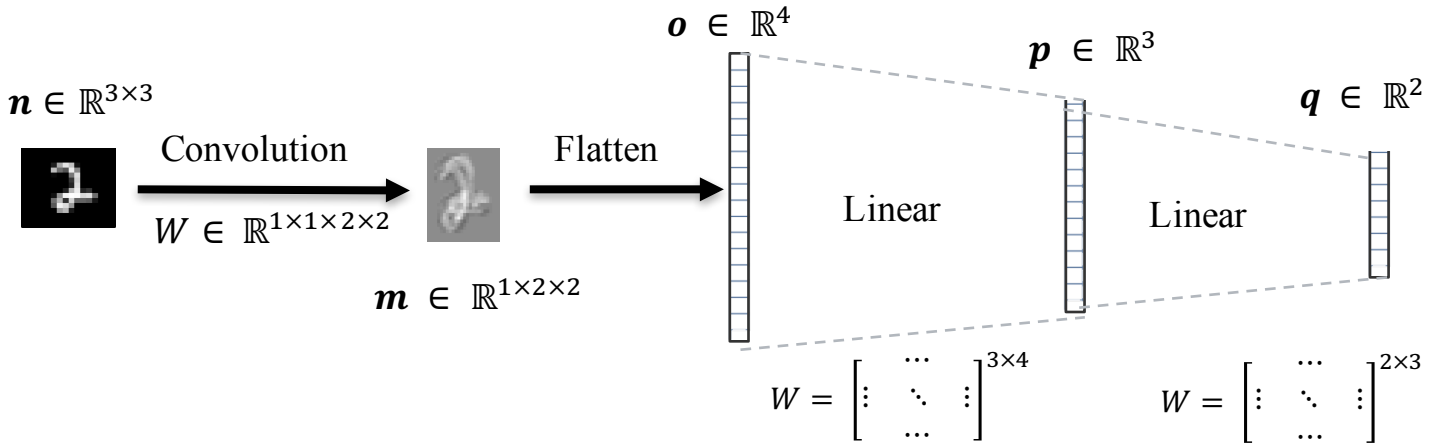


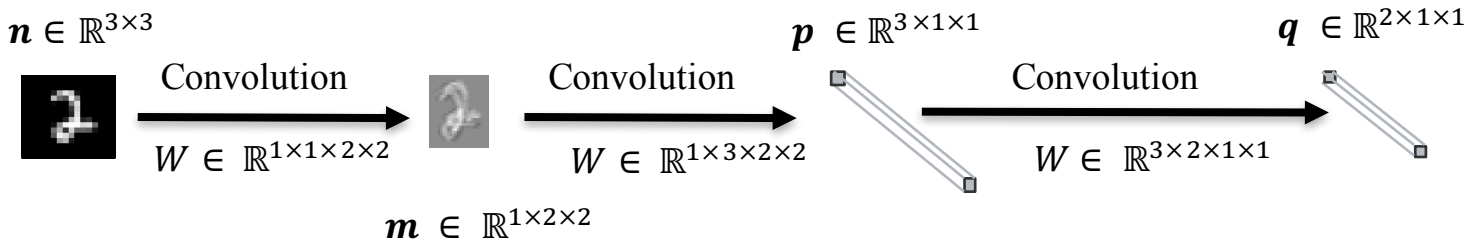


Everything is a Convolution!



Everything is a Convolution!





Thank you!