

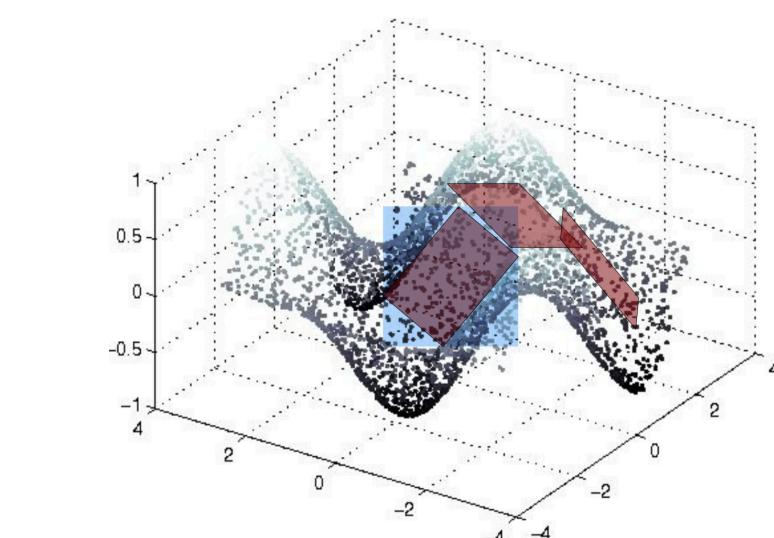
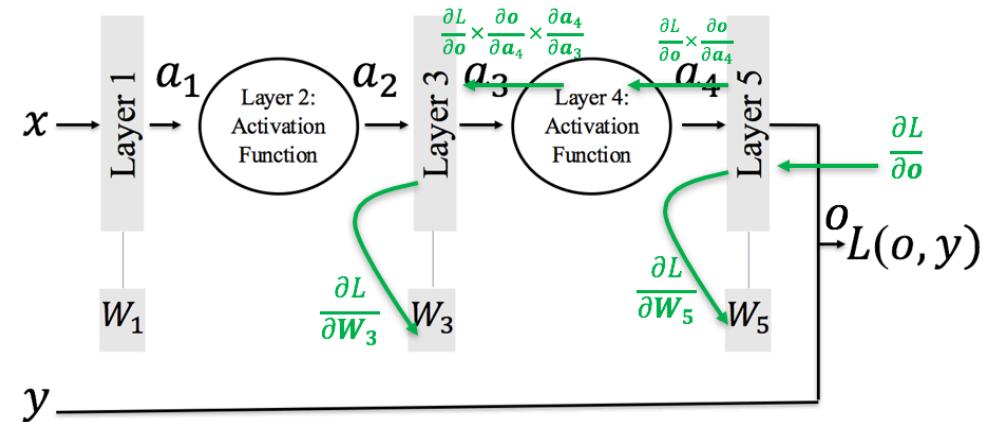
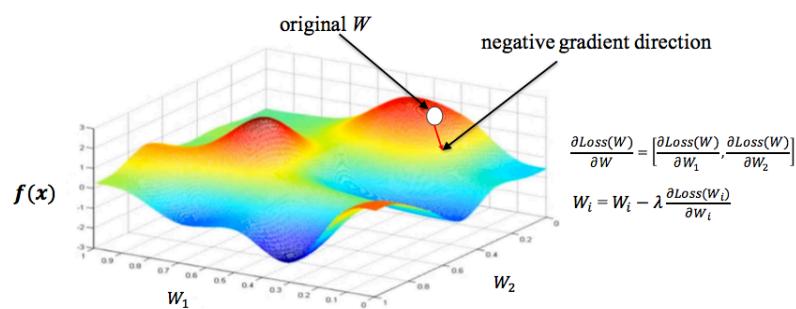
Deep Learning (for Computer Vision)

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

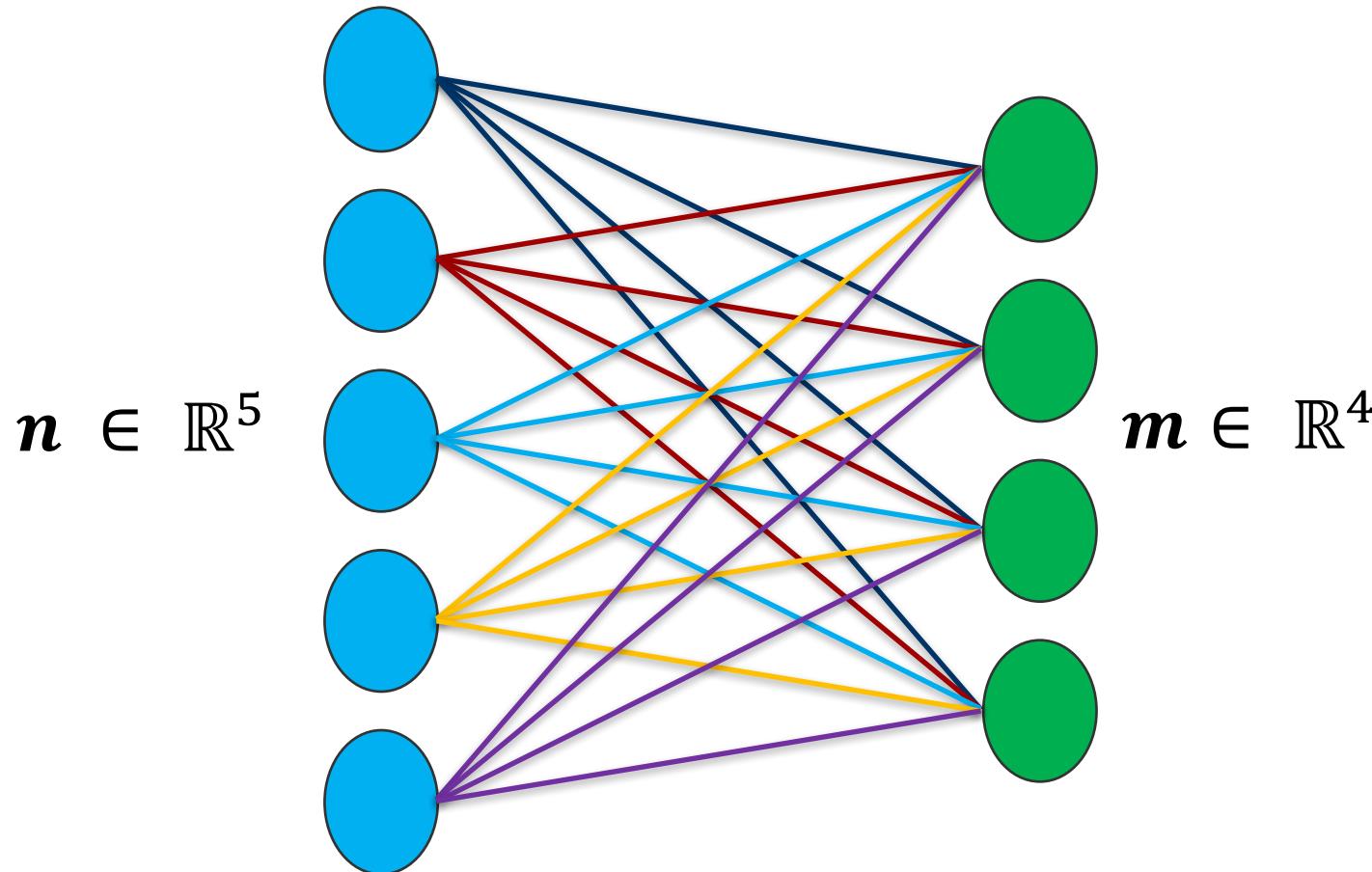
Sources

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

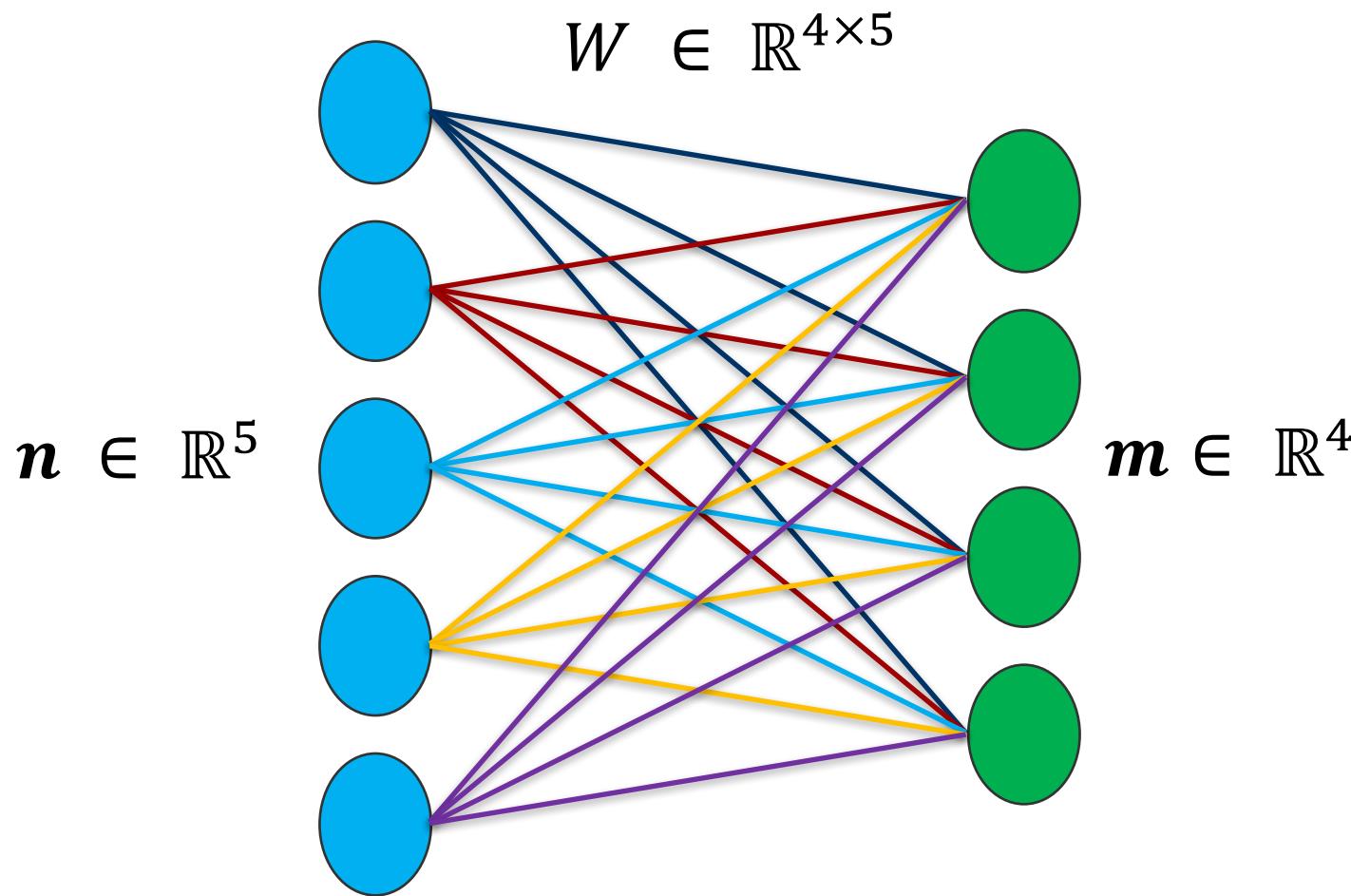
- <http://cs231n.stanford.edu/>
- <http://cs231n.github.io/convolutional-networks/>

Building Blocks: Fully Connected

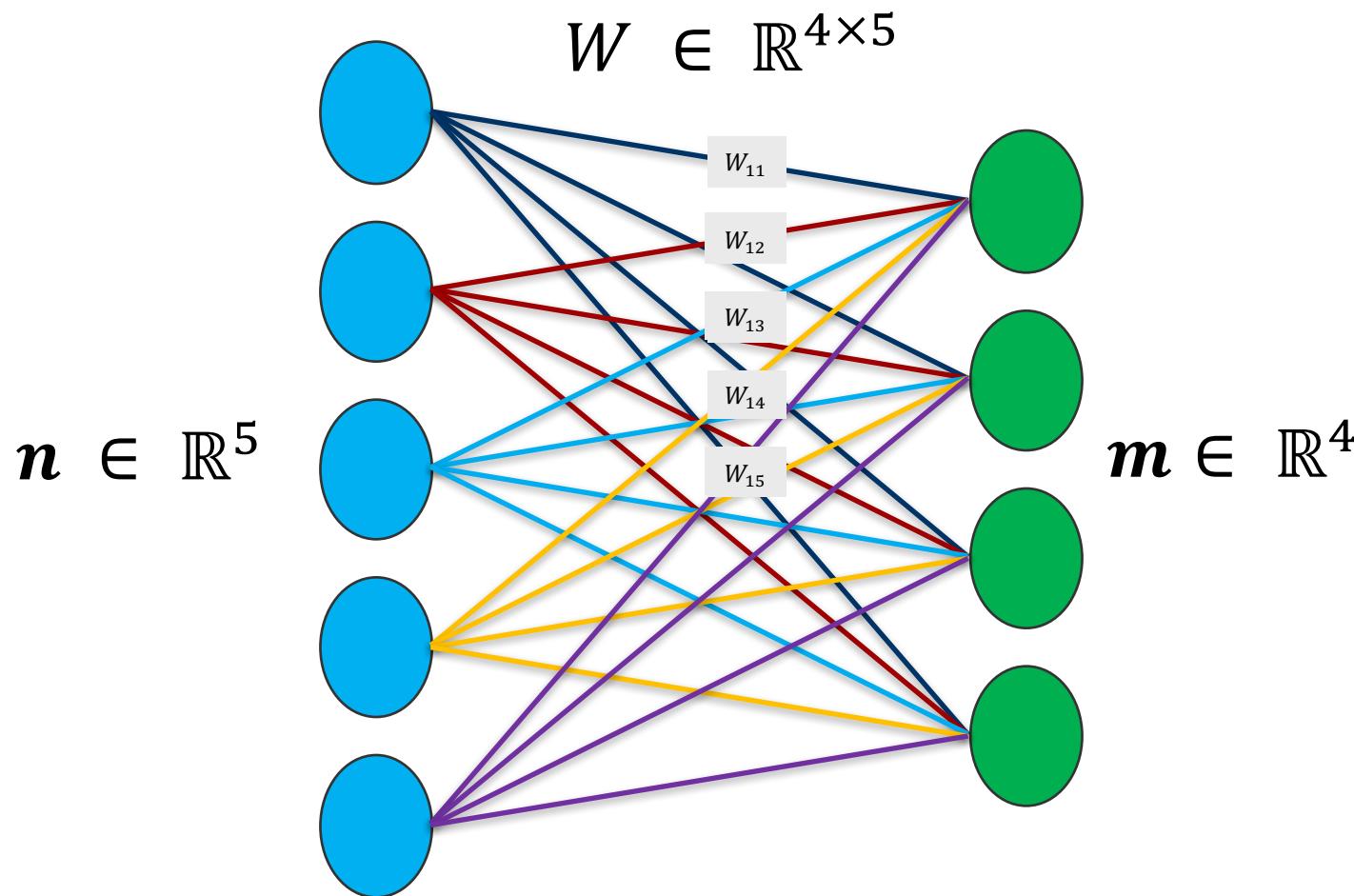
Building Blocks – Fully Connected



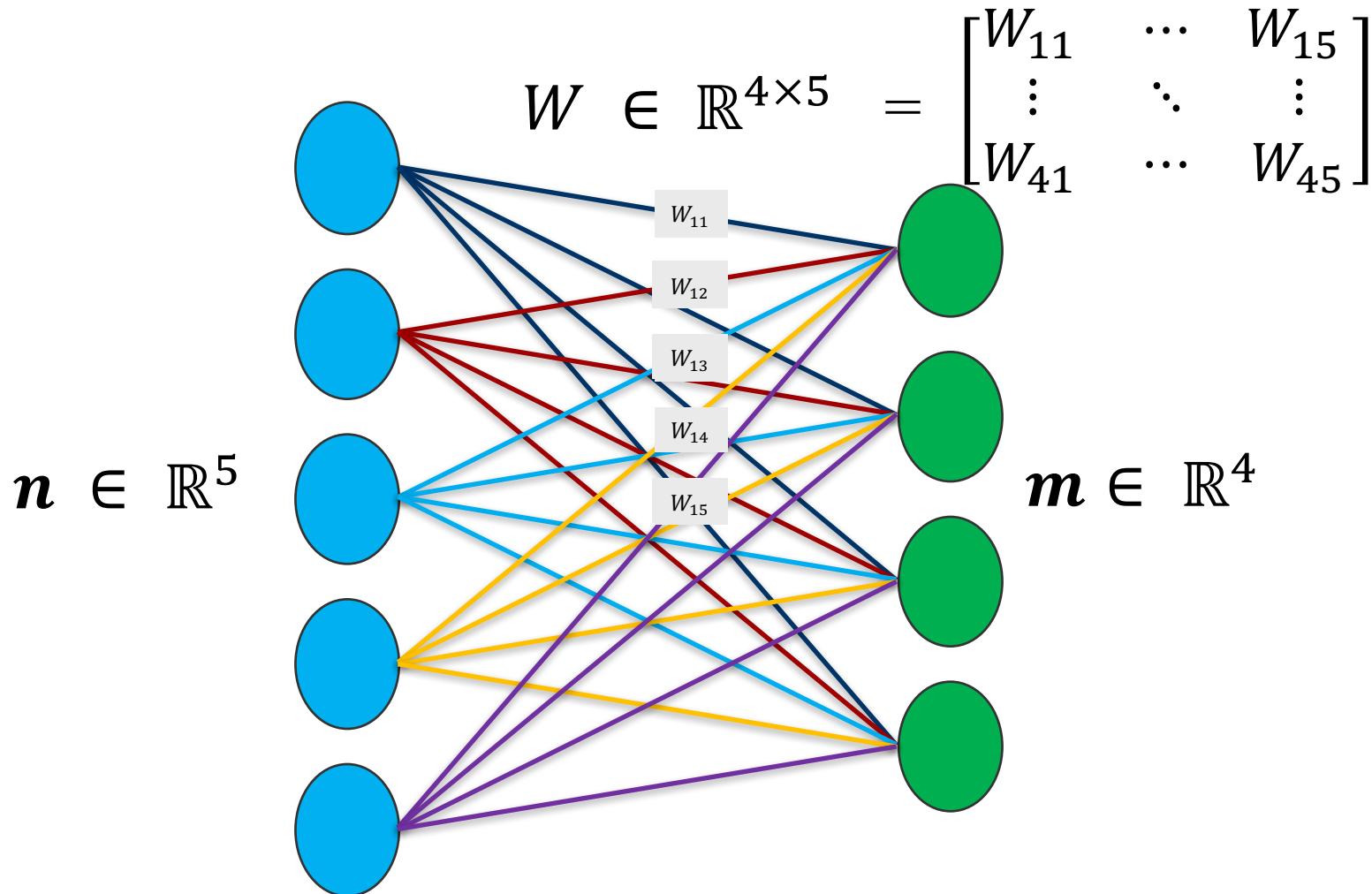
Building Blocks – Fully Connected



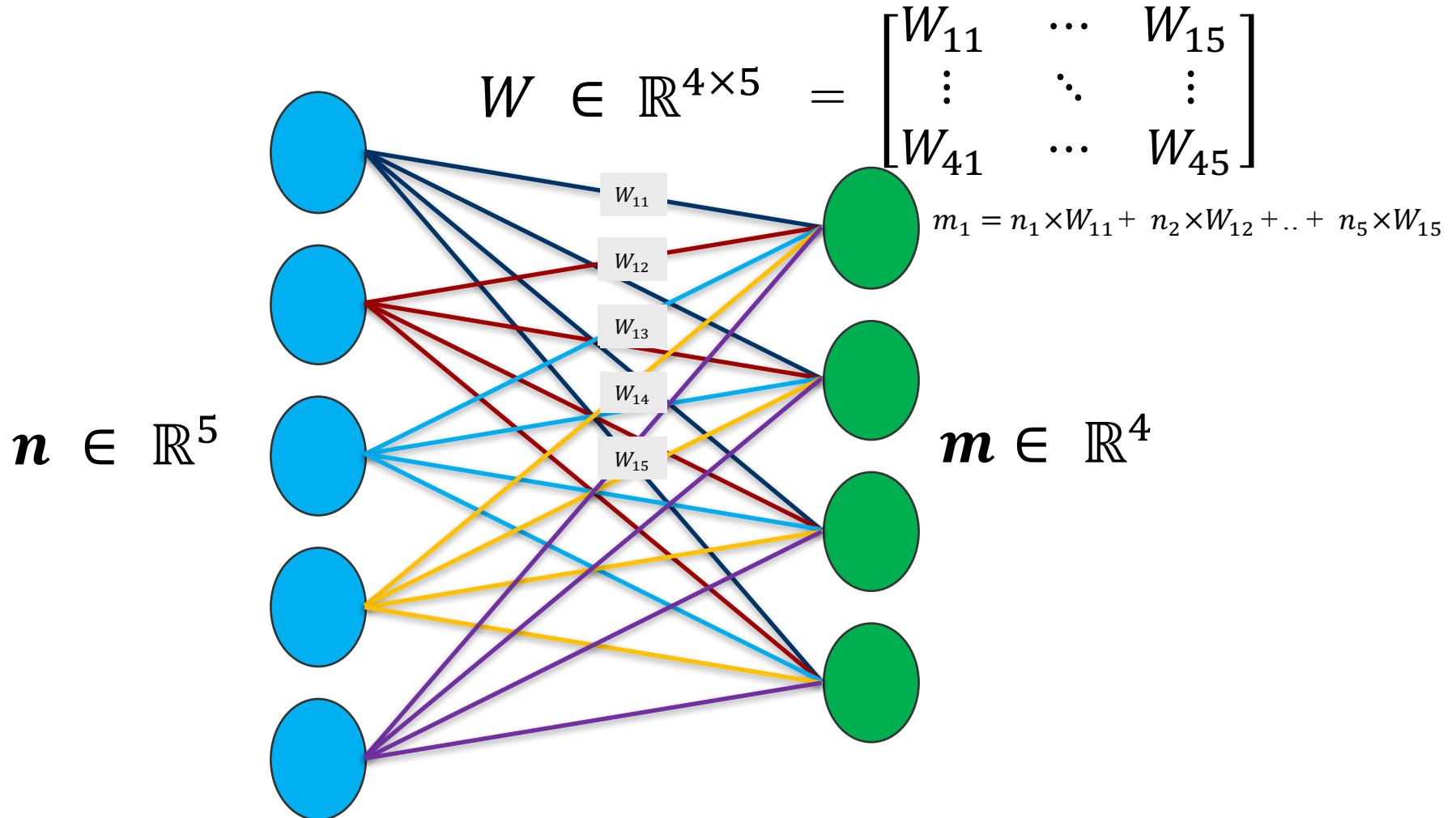
Building Blocks – Fully Connected



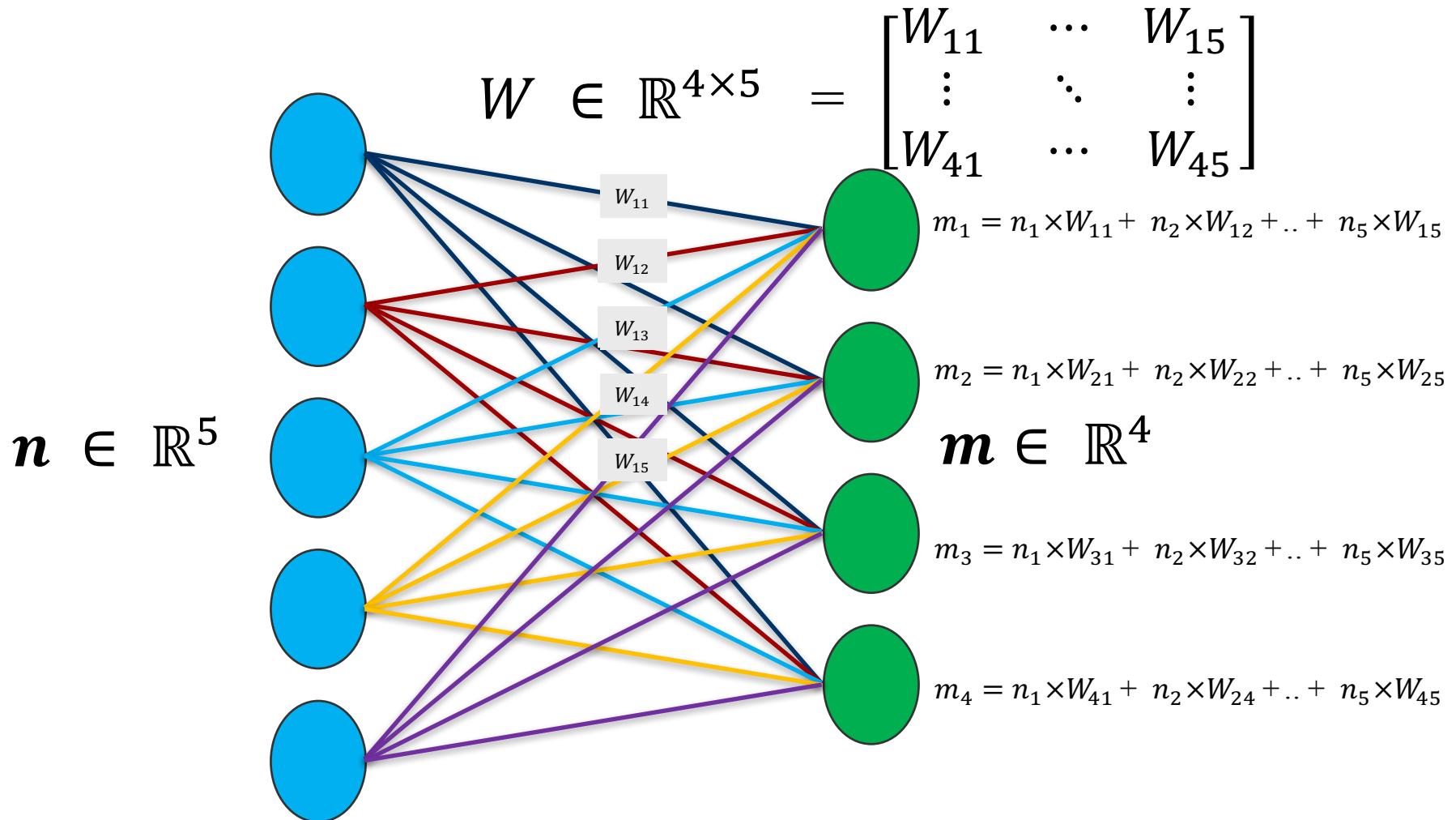
Building Blocks – Fully Connected



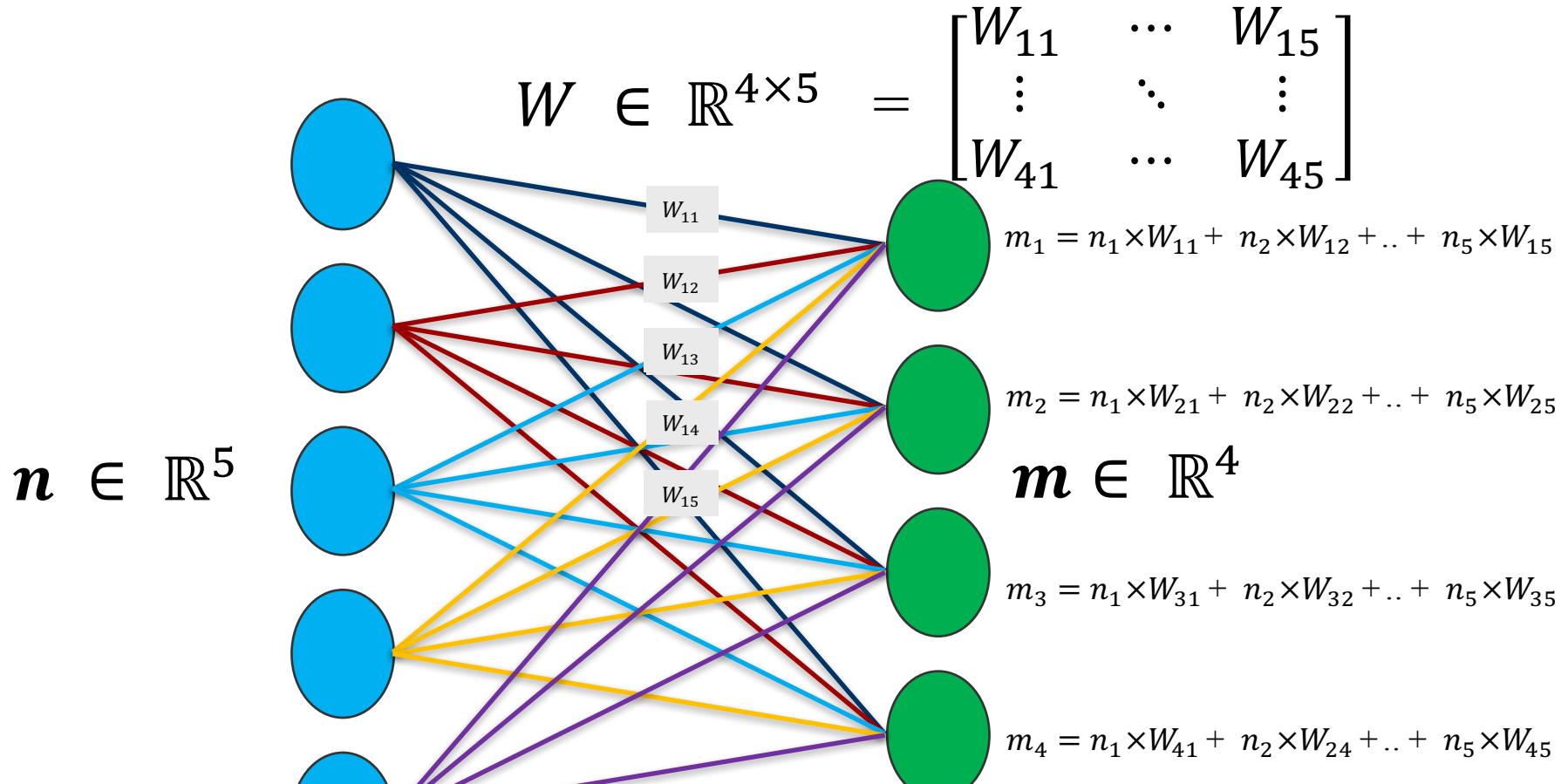
Building Blocks – Fully Connected



Building Blocks – Fully Connected

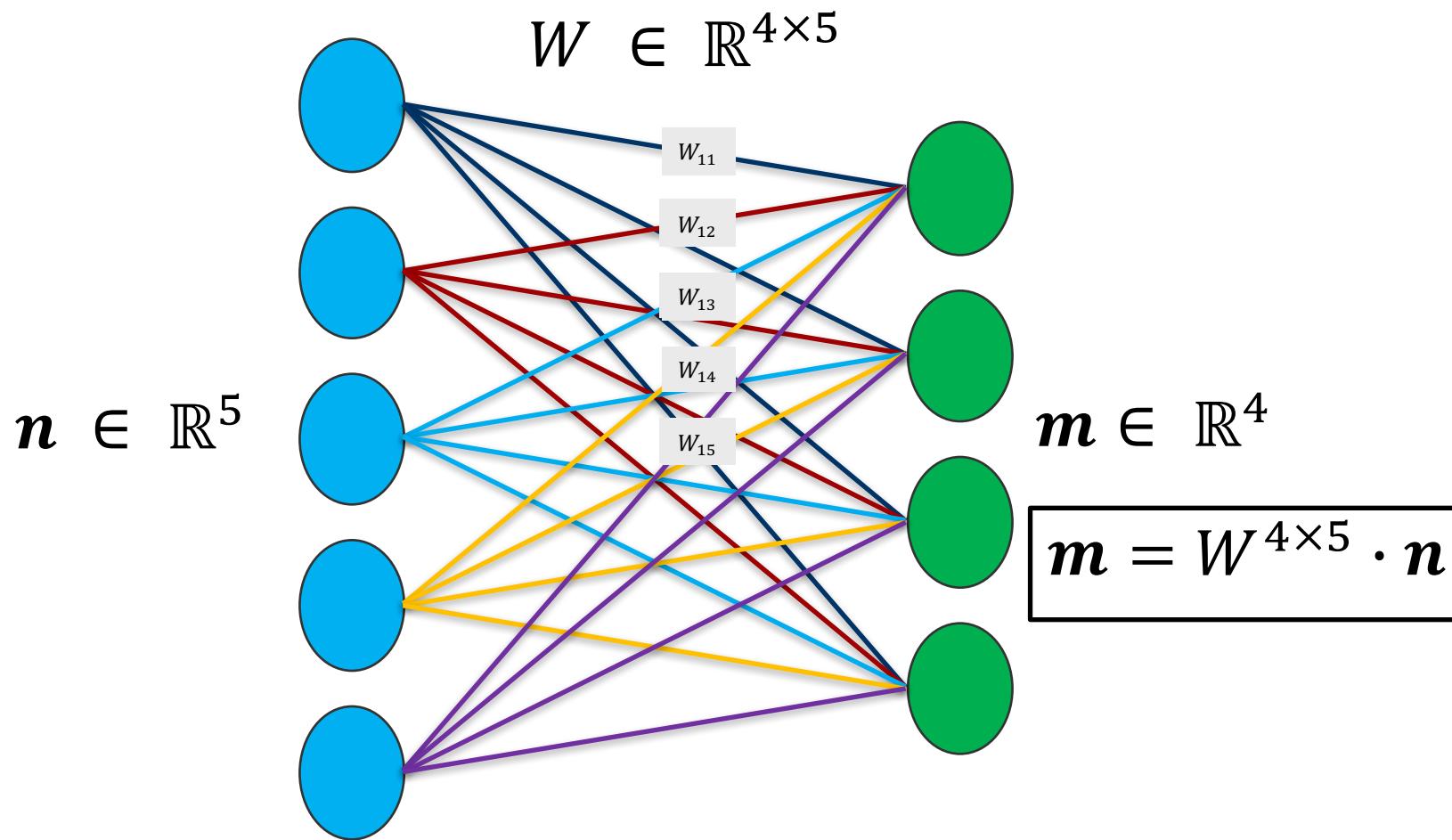


Building Blocks – Fully Connected

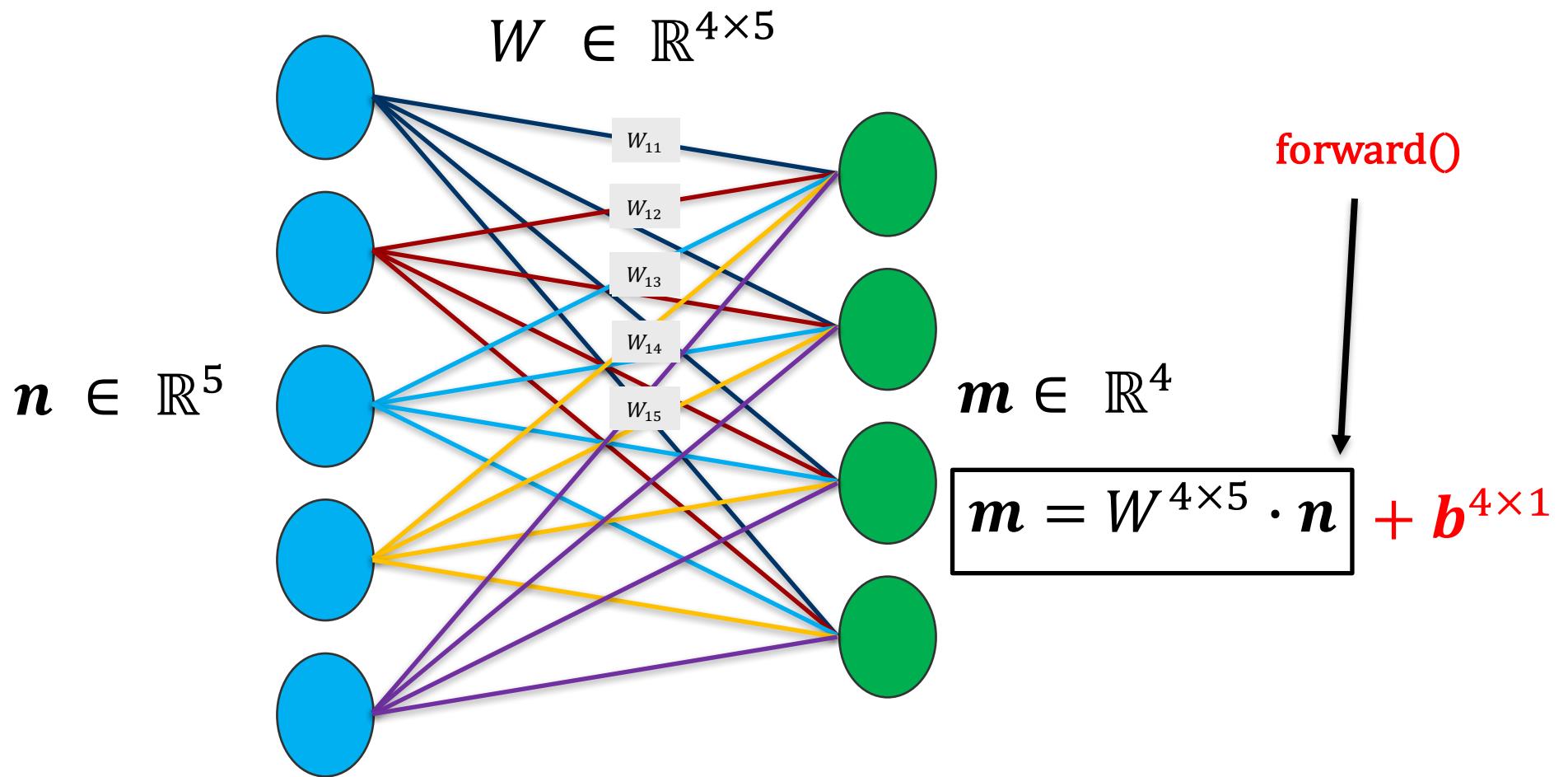


Why is it called fully connected?

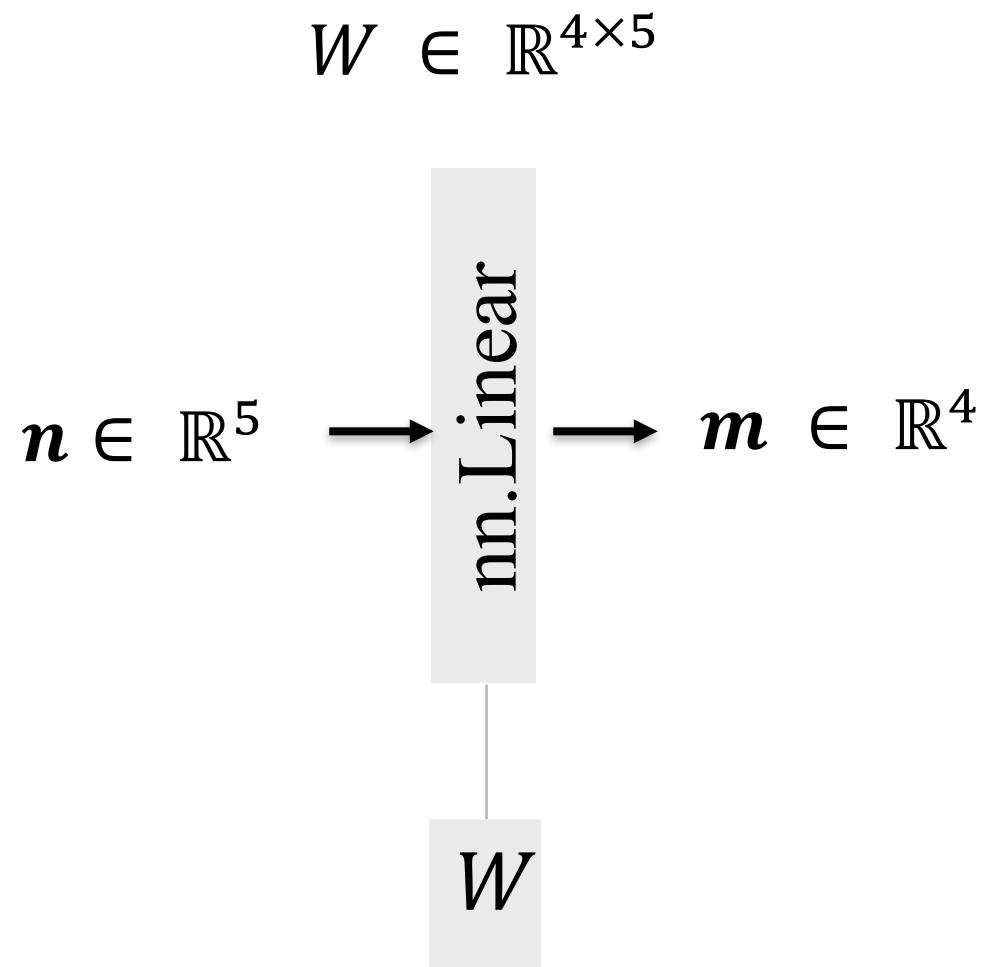
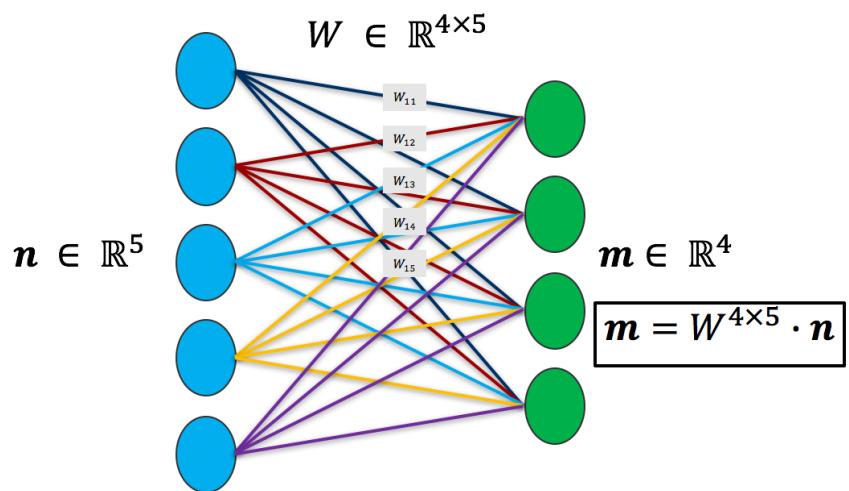
Building Blocks – Fully Connected



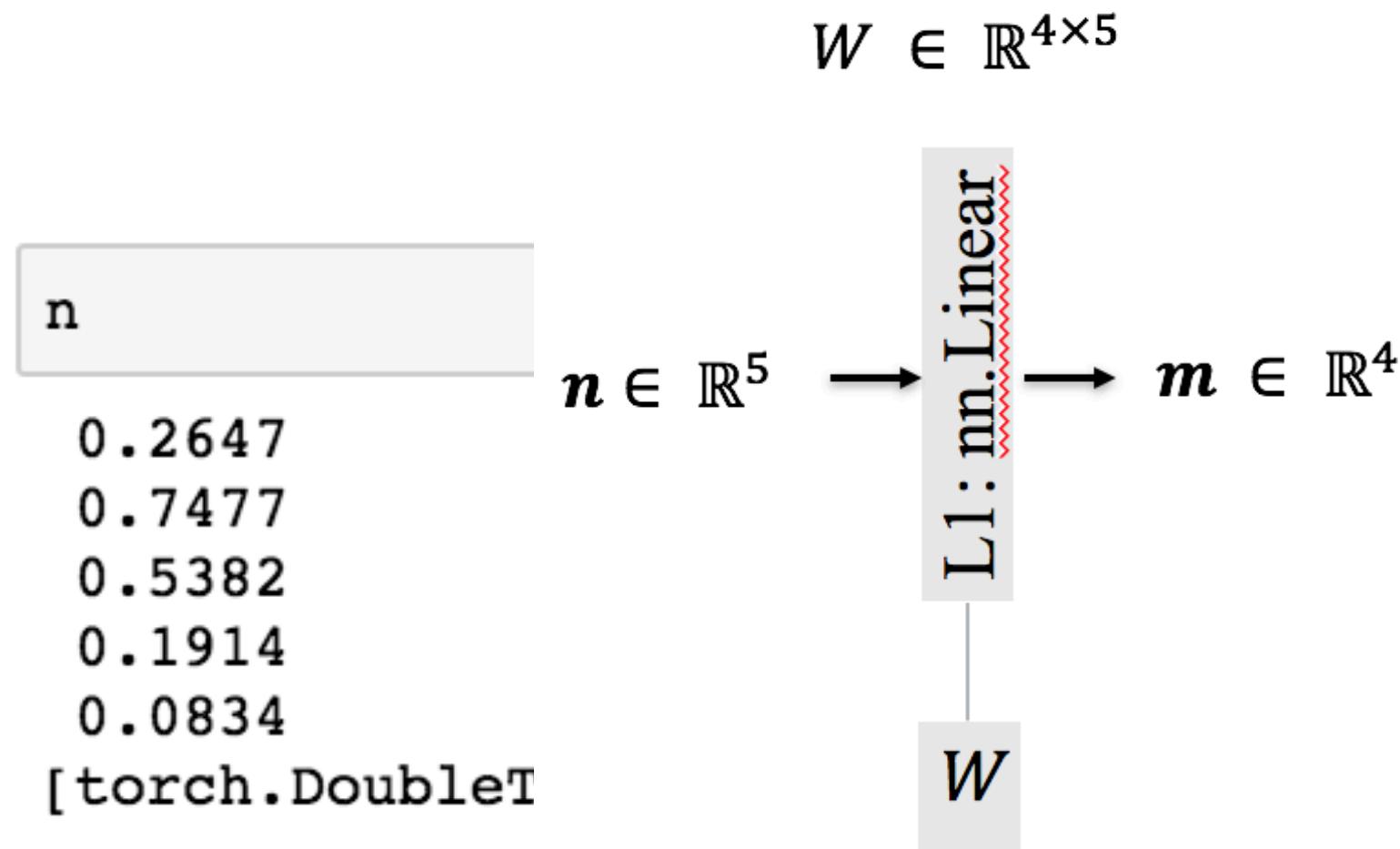
Building Blocks – Fully Connected



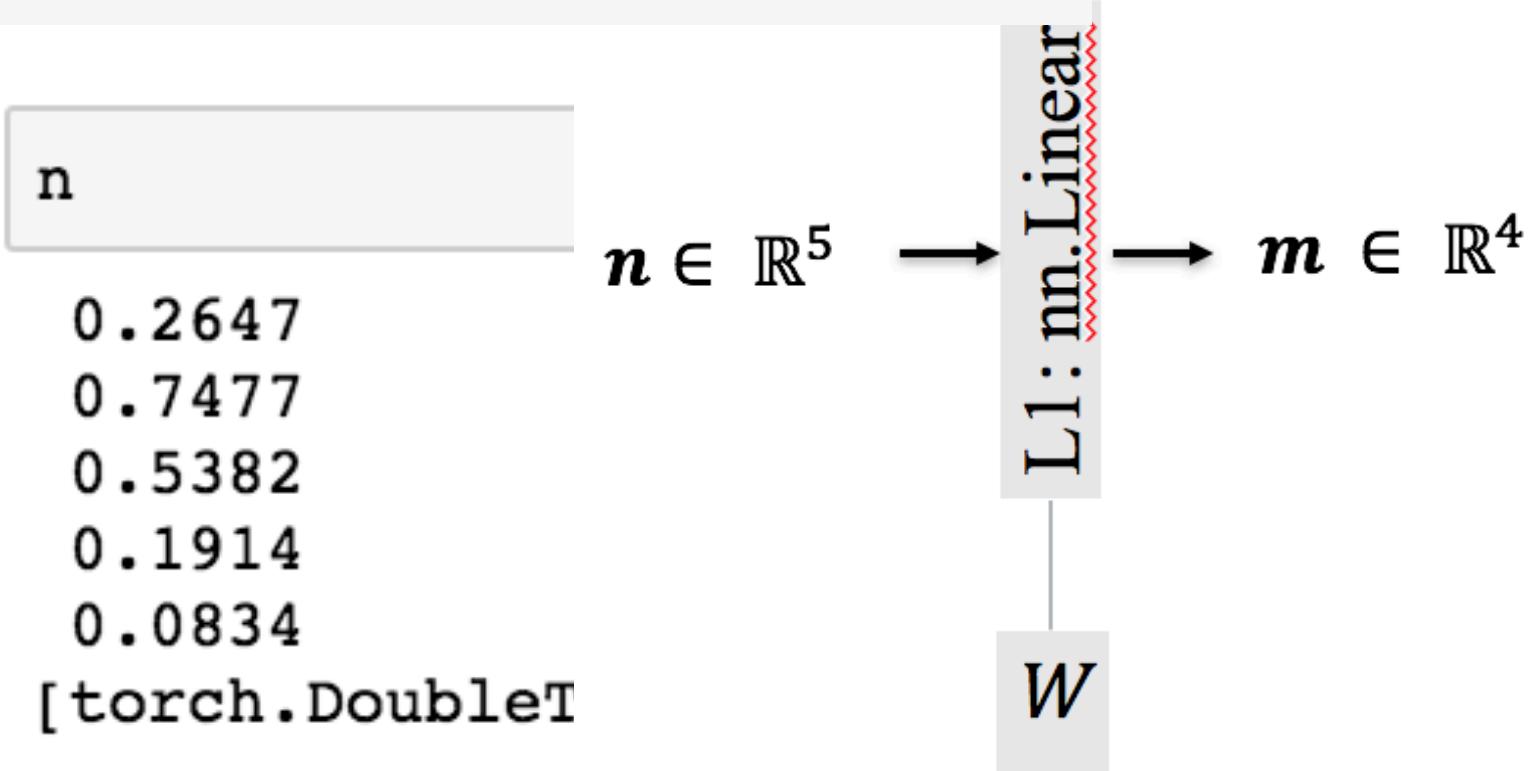
Building Blocks – Fully Connected



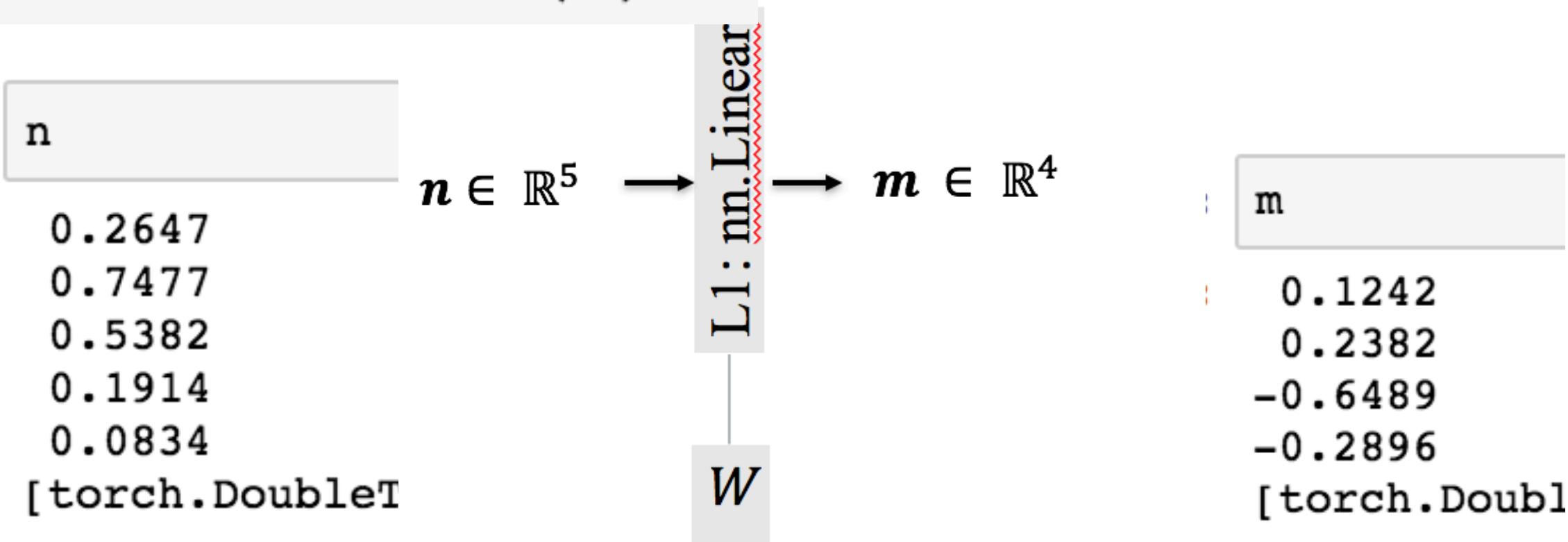
Building Blocks – Fully Connected – Forward: In Torch7



```
require 'nn';
n = torch.rand(5)
lin = nn.Linear(5,4)  $\mathbb{R}^{4 \times 5}$ 
m = lin:forward(n)
```



```
require 'nn';
n = torch.rand(5)
lin = nn.Linear(5,4)  $\mathbb{R}^{4 \times 5}$ 
m = lin:forward(n)
```



```

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

```

n
0.2647
0.7477
0.5382
0.1914
0.0834
[torch.DoubleT

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

L1: nn.Linear

W

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

```

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

```

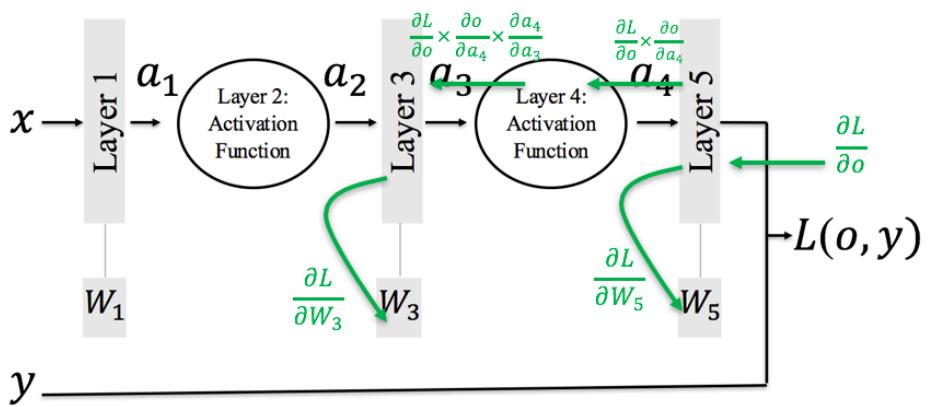
0.1242
0.2382
-0.6489
-0.2896

[torch.DoubleTensor of size 4

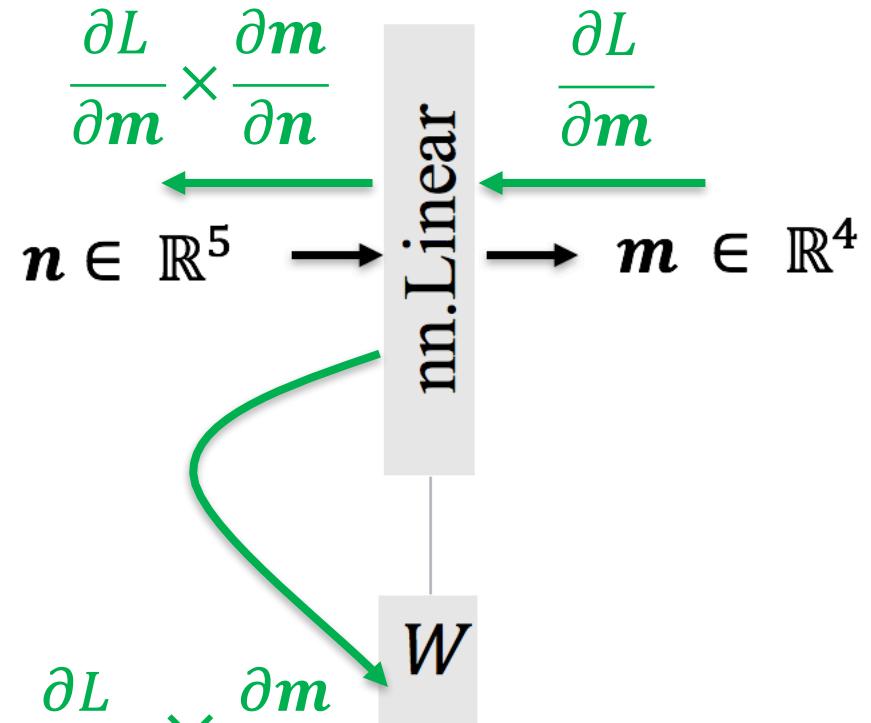
m

m
0.1242
0.2382
-0.6489
-0.2896
[torch.Doubl

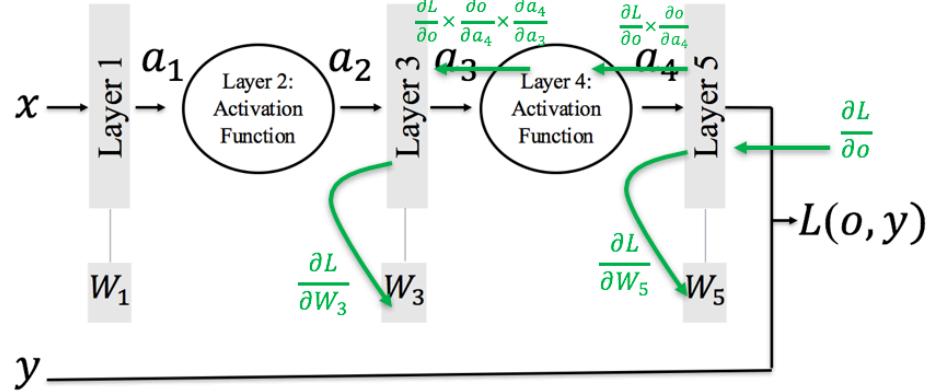
Building Blocks – Fully Connected - Backward



$$W \in \mathbb{R}^{4 \times 5}$$



Building Blocks – Fully Connected - Backward



Gradient (row vector 1x4)

$$W \in \mathbb{R}^{4 \times 5}$$

Jacobian (Matrix)
of size 4x5

$$\frac{\partial L}{\partial m} \times \frac{\partial m}{\partial n}$$

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

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

$$\frac{\partial L}{\partial m}$$

nn.Linear

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial m} \times \frac{\partial m}{\partial W}$$

Gradient (row vector 1x4)

Jacobian (Matrix) of size 4x20

Fully Connected - Backward

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

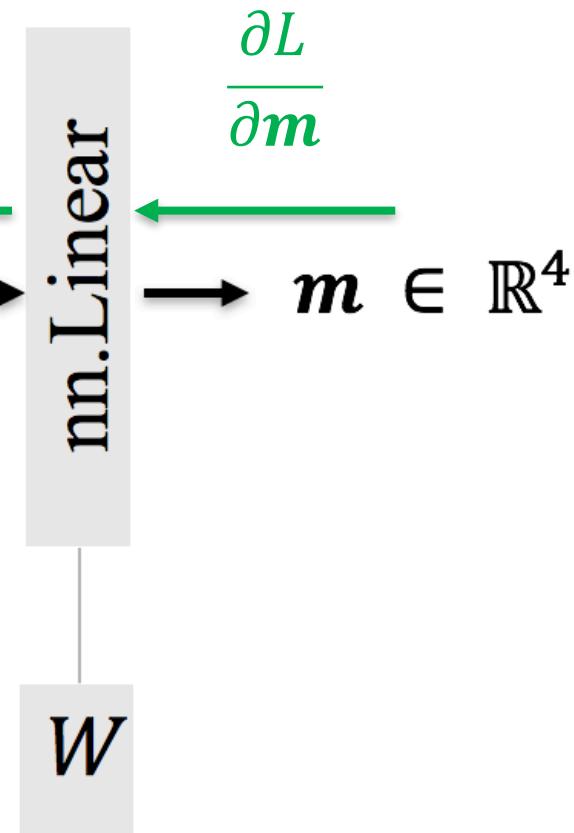
```
lin.gradInput
```

```
-0.1486
-0.3457
-0.0236
0.2292
0.3097
```

```
nextgrad.reshape(1,4)*lin.weight
```

```
-0.1486 -0.3457 -0.0236 0.2292 0.3097
[torch.DoubleTensor of size 1x5]
```

$$W \in \mathbb{R}^{4 \times 5}$$



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]
```

$$W \in \mathbb{R}^{4 \times 5}$$

$$\frac{\partial L}{\partial m}$$

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

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

nn.Linear

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial m} \times \frac{\partial m}{\partial W}$$

```
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)

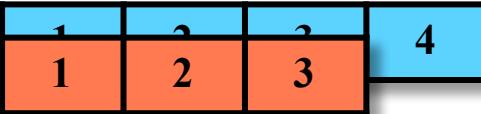
I:  A horizontal vector consisting of four light blue rectangular boxes. Each box contains a black number: 1, 2, 3, and 4 from left to right.

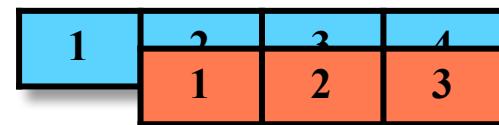
W:  A horizontal vector consisting of three orange rectangular boxes. Each box contains a black number: 1, 2, and 3 from left to right.

Building Blocks – Convolution (Discrete 1D)

I:  A horizontal vector of four blue boxes labeled 1, 2, 3, and 4.

W:  A horizontal vector of three orange boxes labeled 1, 2, and 3.

 A horizontal vector of four blue boxes labeled 1, 2, 3, and 4. The first three boxes are aligned with the input vector I, and the fourth box is aligned with the output vector.

 A horizontal vector of four orange boxes labeled 1, 1, 2, and 3. The first three boxes are aligned with the input vector I, and the fourth box is aligned with the output vector.

Slide

Building Blocks – Convolution (Discrete 1D)

I: 

W: 



O:  Dim = Dim(I) - Dim(W) + 1

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

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

Slide

Correlation

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

Building Blocks – Convolution (Discrete 1D)

$$I: \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

$$W: \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$W^{Flip}: \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = W: \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$$

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

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

$$O': \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \text{Dim} = \text{Dim}(I) - \text{Dim}(W) + 1$$

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

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

Slide

True
Convolution

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

Building Blocks – Convolution (Discrete 1D)

I:  [0, 1, 2, 3, 4, 0]

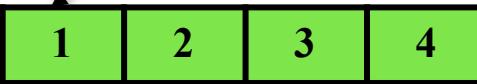
Half-padding (same size output)

W:  [1, 2, 3]

 [0, 1, 2, 3, 4, 0]
 [1, 2, 3]

...

 [0, 1, 2, 3, 4, 0]
 [1, 2, 3]

O:  [1, 2, 3, 4]

$$\text{Dim} = \text{Dim}(I) - \text{Dim}(W) + 1$$

Building Blocks – Convolution (Discrete 1D)

I: A horizontal array of six light blue boxes containing the values 0, 1, 2, 3, 4, and 0.

Half-padding, Stride = 2

W: A horizontal array of three orange-red boxes containing the values 1, 2, and 3.



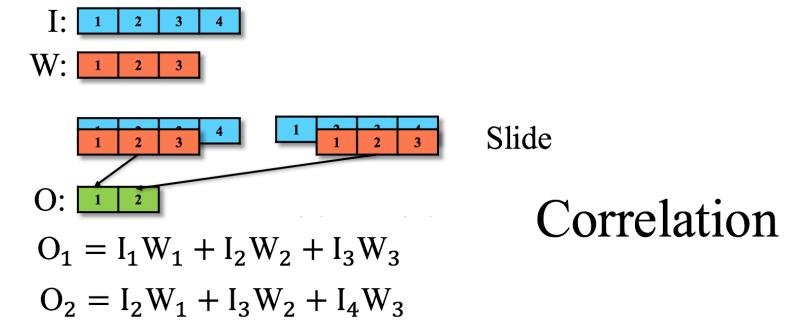
O: A horizontal array of two green boxes containing the values 1 and 2.



$$\text{Dim} = \left\lfloor \frac{\text{Dim}(I) - \text{Dim}(W)}{\text{Stride}=2} \right\rfloor + 1$$

Building Blocks – Convolution – Backward

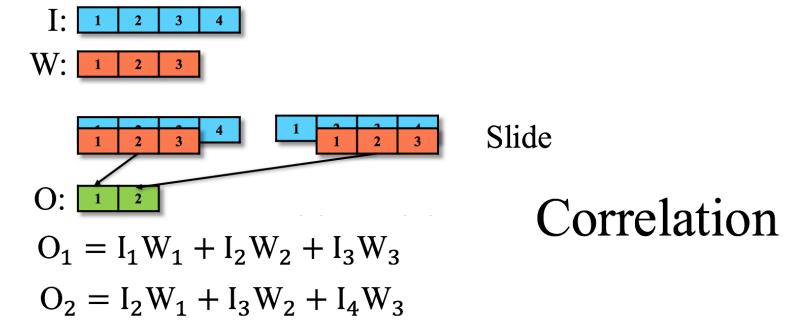
$$\frac{\partial O}{\partial I} = \begin{bmatrix} W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \end{bmatrix}$$



Building Blocks – Convolution – Backward

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

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

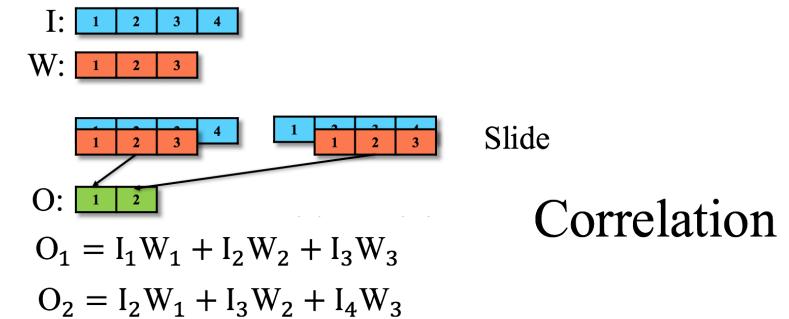


Building Blocks – Convolution – Backward

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

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

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



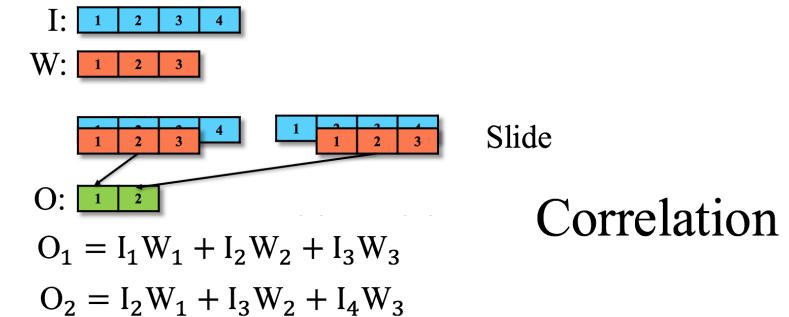
Building Blocks – Convolution – Backward

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

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

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

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



Building Blocks – Convolution – Backward

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

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

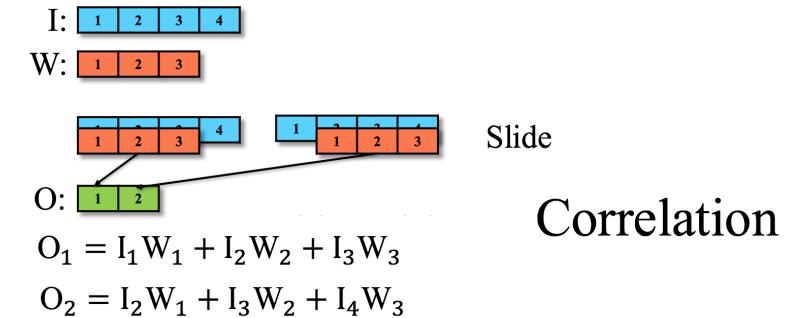
$$\frac{\partial L}{\partial \mathbf{O}} = [\partial LO_1 \quad \partial LO_2]$$

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

I: A horizontal row of four blue boxes containing the numbers 1, 2, 3, and 4.

LO: A horizontal row of two red boxes containing the numbers 1 and 2.

$\frac{\partial L}{\partial \mathbf{W}}$: Three horizontal rows of 4x2 matrices. The first row shows a window of size 2x2 with values [1, 2] overlapping with the input [1, 2]. The second row shows a window of size 2x2 with values [1, 2] overlapping with the input [2, 3]. The third row shows a window of size 2x2 with values [2, 1] overlapping with the input [3, 4].



Building Blocks – Convolution – Backward

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

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

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

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

I:

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

LO: 1 2

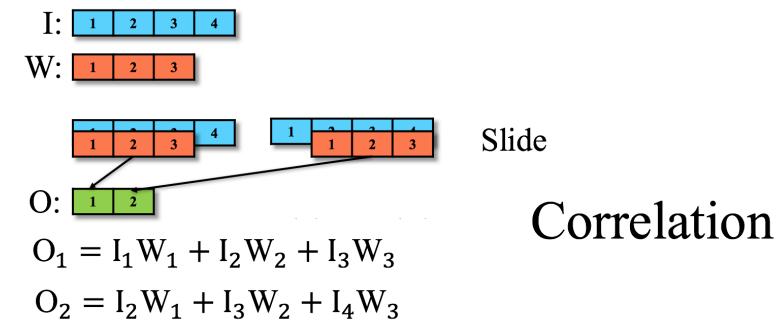
$\frac{\partial L}{\partial W}$:

1	2	3	4
1	2		

1	2	2	4
1	2		

1	2	3	4
		1	2

Slide



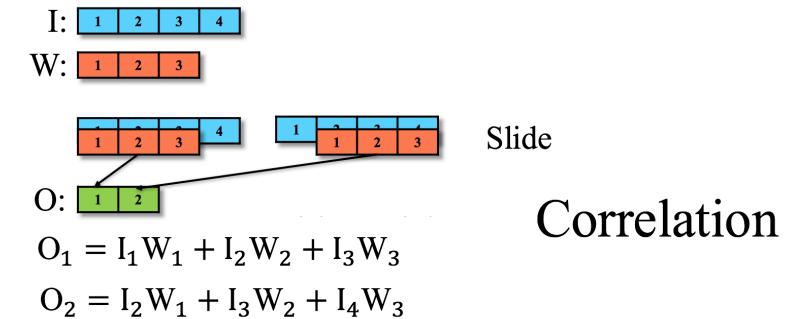
$$\frac{\partial L}{\partial W} = \text{Correlation}(I, LO)$$

Building Blocks – Convolution – Backward

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

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

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



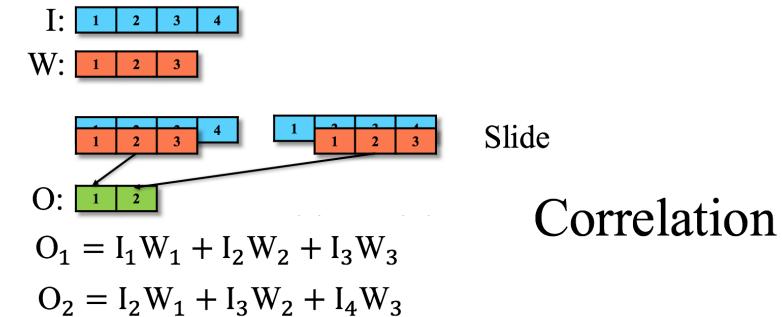
Building Blocks – Convolution – Backward

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

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

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

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



Building Blocks – Convolution – Backward

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

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

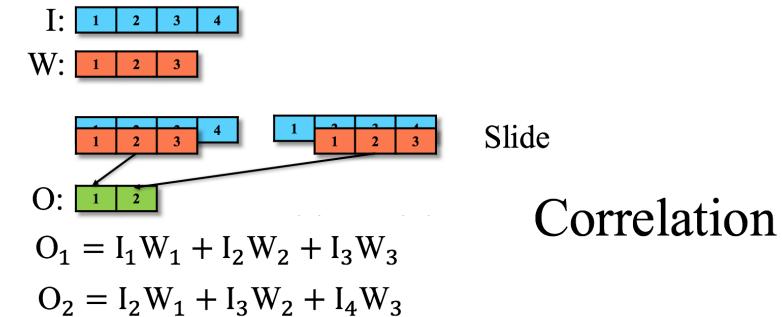
$$\frac{\partial L}{\partial \mathbf{O}} = [\partial LO_1 \quad \partial LO_2]$$

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

$$W_{pad}: \begin{bmatrix} 0 & 1 & 2 & 3 & 0 \end{bmatrix}$$

$$LO_{flip}: \begin{bmatrix} 2 & 1 \end{bmatrix}$$

$$\frac{\partial L}{\partial I}: \begin{bmatrix} 0 & 1 & 2 & 3 & 0 \end{bmatrix} \dots \begin{bmatrix} 0 & 1 & 2 & 3 & 0 \end{bmatrix} \quad \text{Slide}$$



Building Blocks – Convolution – Backward

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

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

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

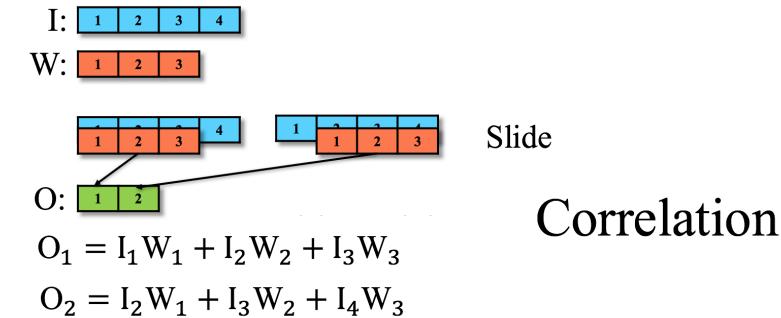
$$\frac{\partial L}{\partial \mathbf{I}} = \frac{\partial L}{\partial \mathbf{O}} \times \frac{\partial \mathbf{O}}{\partial \mathbf{I}} = [\partial LO_1 \times W_1 \quad \partial LO_1 \times W_2 + \partial LO_2 \times W_1 \quad \partial LO_1 \times W_3 + \partial LO_2 \times W_2 \quad \partial LO_2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

$$W_{pad}: \begin{bmatrix} 0 & 1 & 2 & 3 & 0 \end{bmatrix}$$

$$LO_{flip}: \begin{bmatrix} 2 & 1 \end{bmatrix}$$

$$\frac{\partial L}{\partial I}: \begin{bmatrix} 0 & 1 & 2 & 3 & 0 \\ 2 & 1 \end{bmatrix} \dots \begin{bmatrix} 0 & 1 & 2 & 3 & 0 \\ 2 & 1 \end{bmatrix}$$

Slide



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

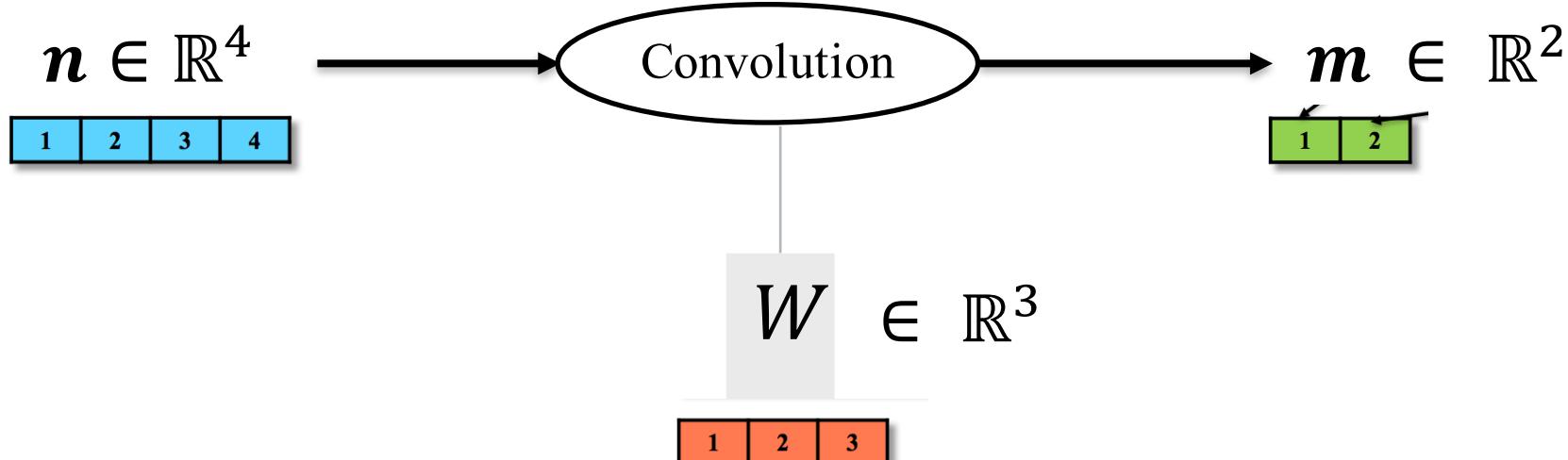
Building Blocks – Convolution – Forward

```
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]
```



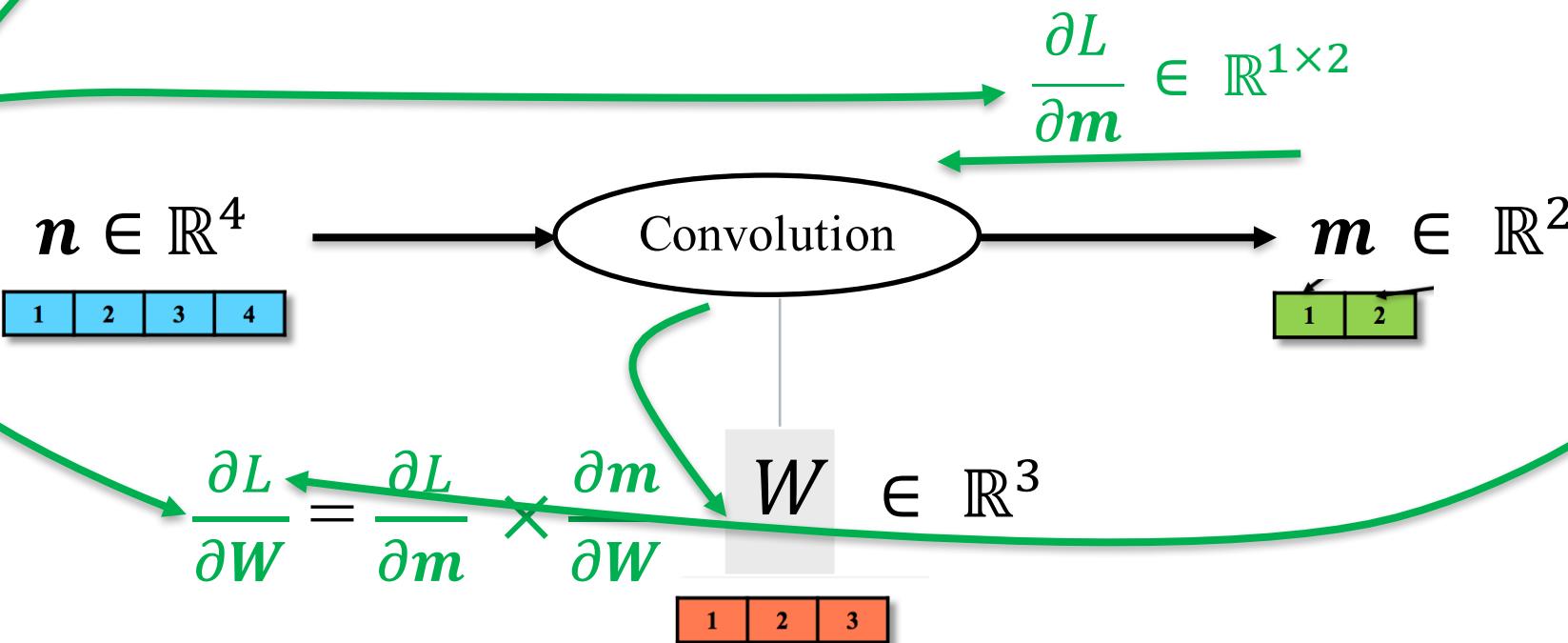
Building Blocks – Convolution – Backward

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

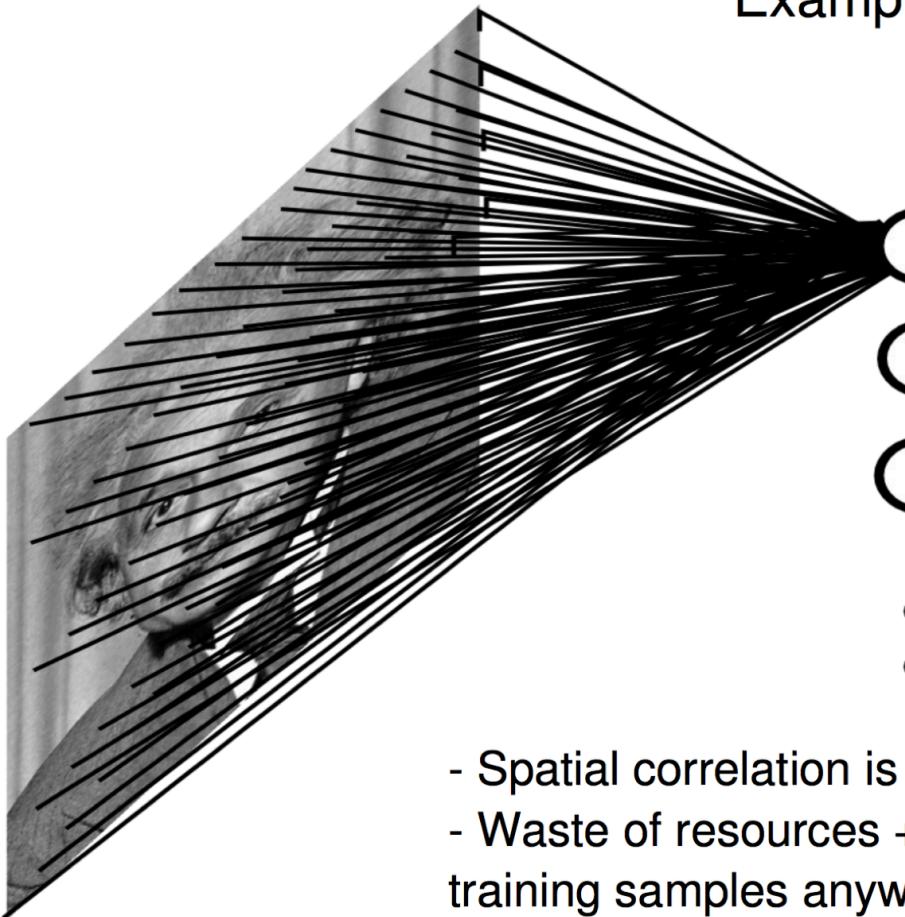
0.6464 0.8428 0.5257
[torch.DoubleTensor of size 1x3]

```
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]



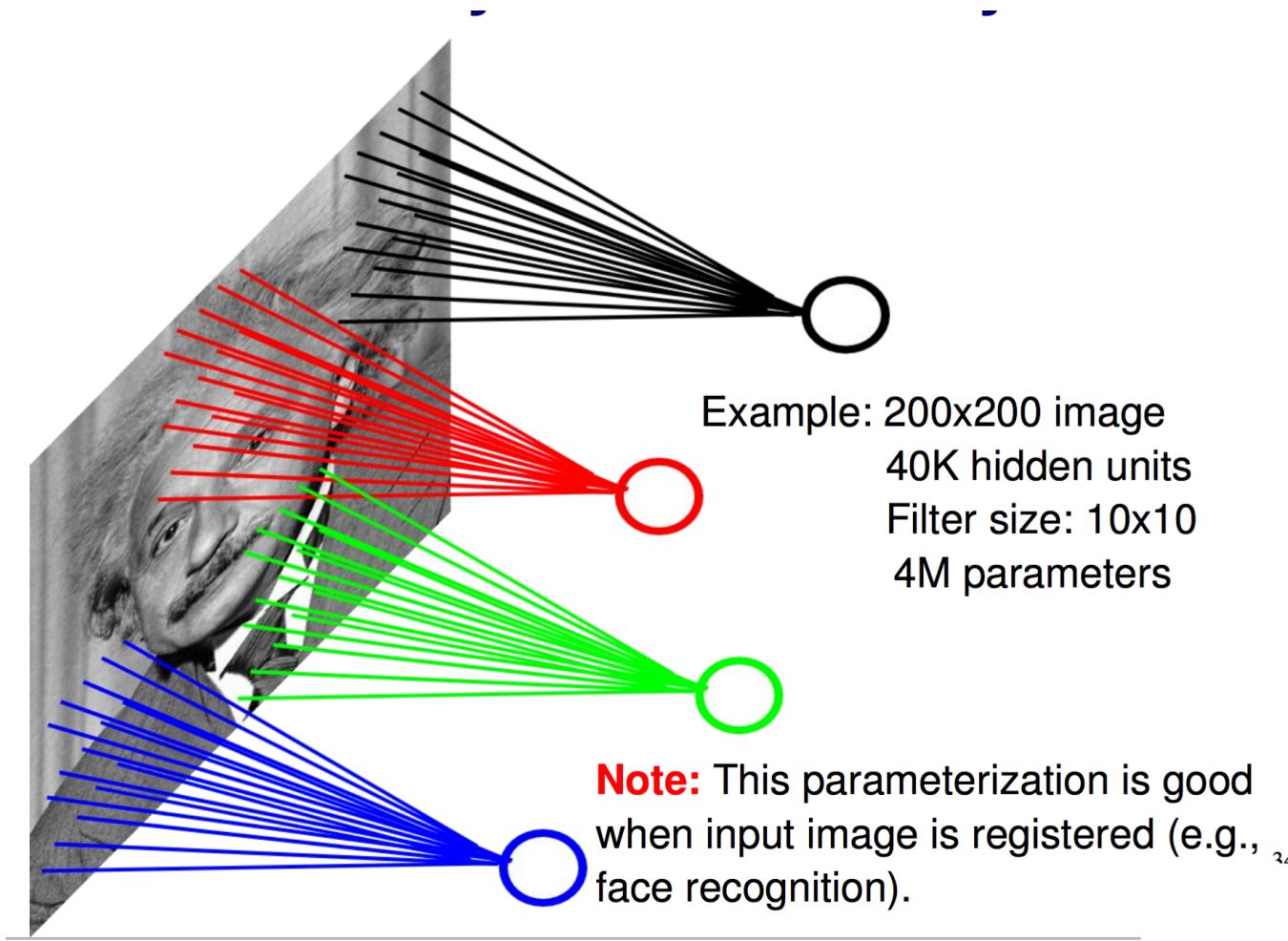
Building Blocks - Convolution



Example: 200x200 image
40K hidden units
→ **~2B parameters!!!**

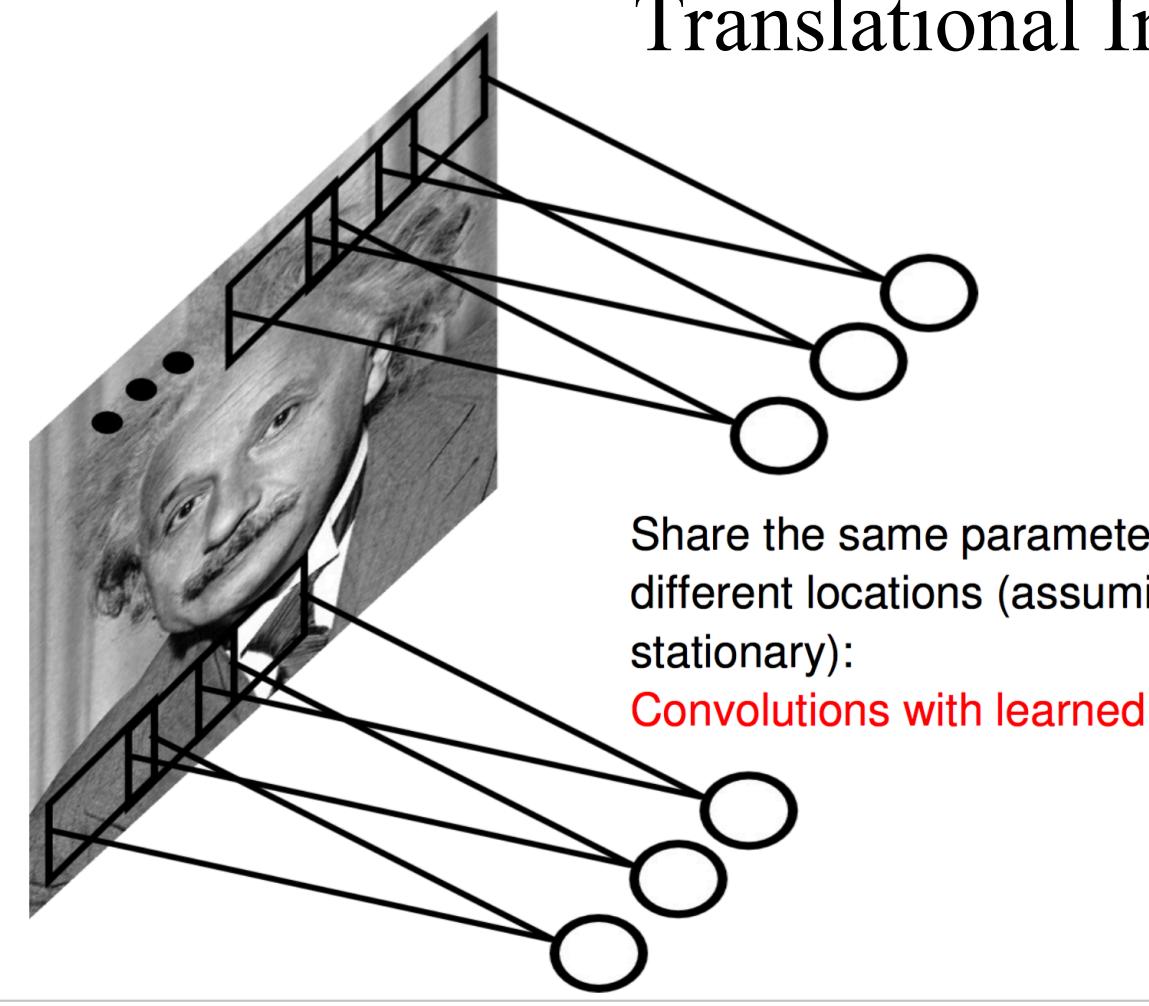
- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Building Blocks - Convolution

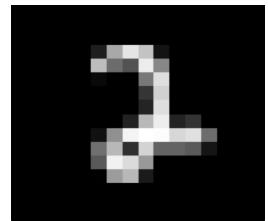


Building Blocks - Convolution

Translational Invariance

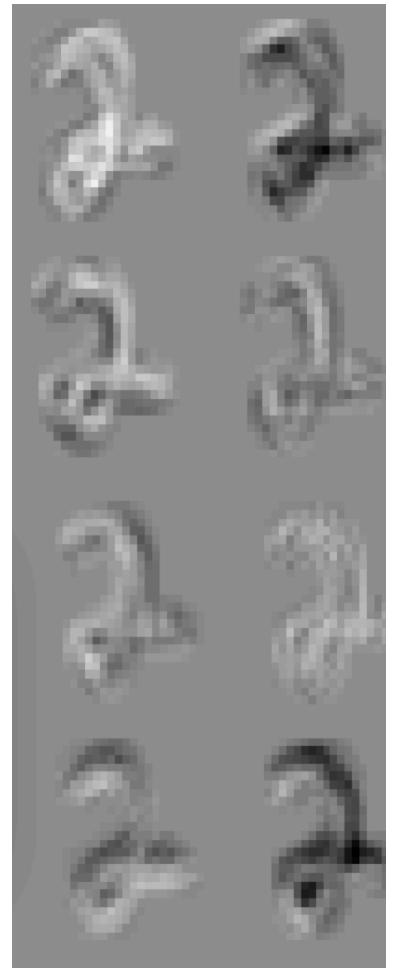
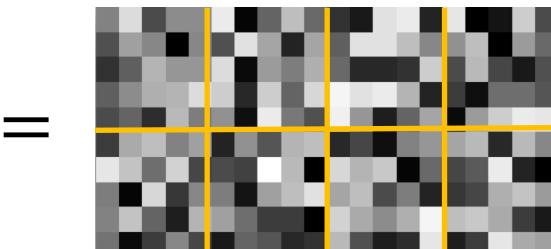


Building Blocks - Convolution

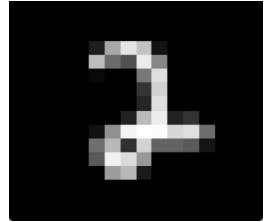


$$n \in \mathbb{R}^{16 \times 16} \rightarrow \text{Convolution} \rightarrow m \in \mathbb{R}^{8 \times 12 \times 12}$$

$$W \in \mathbb{R}^{1 \times 8 \times 5 \times 5}$$



Building Blocks - Convolution



$n \in \mathbb{R}^{16 \times 16}$

Convolution

$m \in \mathbb{R}^{8 \times 12 \times 12}$

W

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

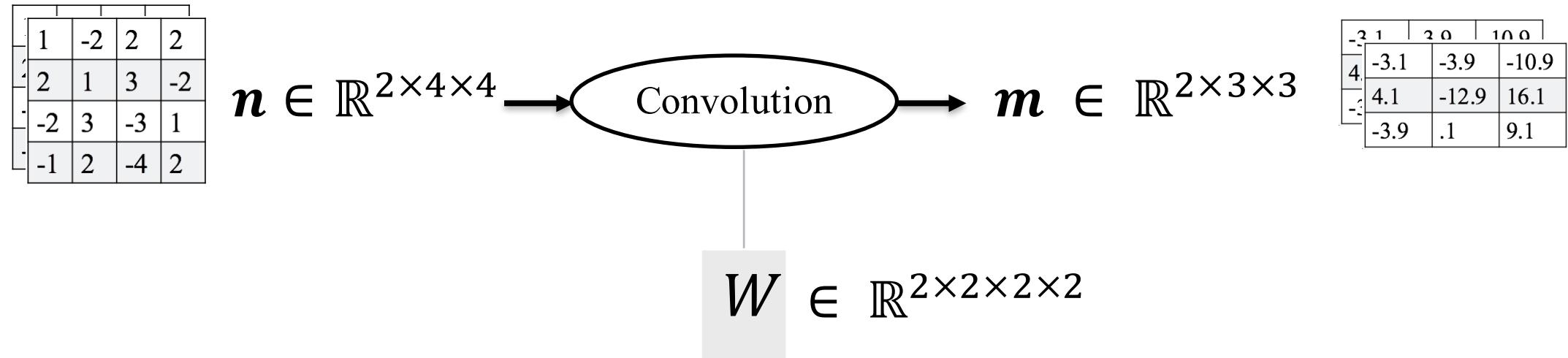
> =m:size()
8
12
12
[torch.LongStorage of size 3]
```

```
> =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, :, :] W[2, 1, :, :]

1	-2
-2	1

3	1
2	2

1	0
0	1

0	0
0	4

W[1, 2, :, :] W[2, 2, :, :]

0.1
0.2

Bias $\mathbf{b} = 2$
(nOutputPlane)

Image O = 2 x 3 x 3

O[1, :, :]

O[2, :, :]

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[1, 2, :, :]

1	0
0	1

W[2, 1, :, :]
W[2, 2, :, :]

3	1
2	2
0	0
0	4

Bias **b** = 2
(nOutputPlane)

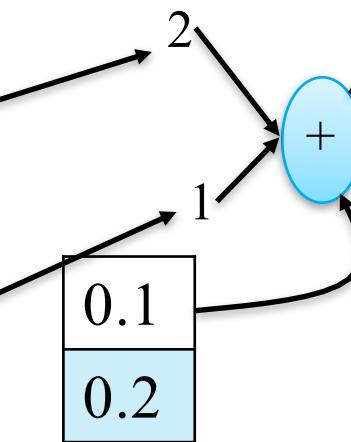


Image O = 2 x 3 x 3

O[1, :, :]

3.1		

O[2, :, :]

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[1, 2, :, :]

1	0
0	1

W[2, 1, :, :]

3	1
2	2

0	0
0	4

Bias **b** = 2
(nOutputPlane)

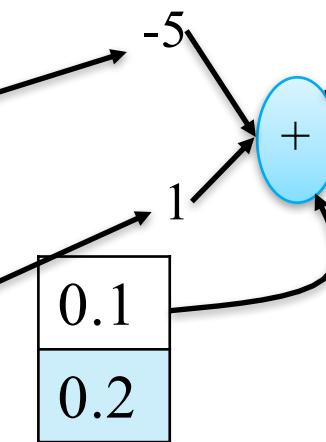


Image O = 2 x 3 x 3

O[1, :, :]

-3.1	-3.9	

O[2, :, :]

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[1, 2, :, :]

1	0
0	1

W[2, 1, :, :]

3	1
2	2
0	0

W[2, 2, :, :]

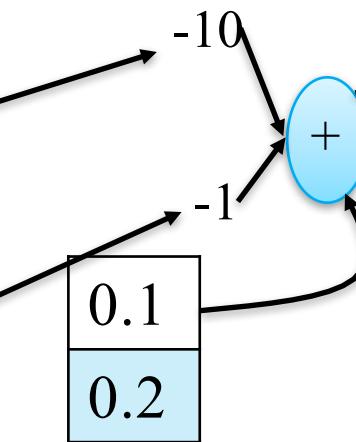
0	4
0	4

Image O = 2 x 3 x 3

O[1, :, :]

-3.1	-3.9	-10.9

O[2, :, :]



Bias **b** = 2
(nOutputPlane)

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)

1	-2
-2	1

W[1, 1, :, :]	W[2, 1, :, :]
3	1
2	2

1	0
0	1

W[1, 2, :, :]	W[2, 2, :, :]
0	0
0	4

Bias **b** = 2
(nOutputPlane)

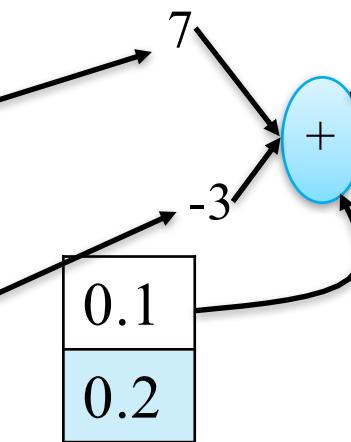


Image O = 2 x 3 x 3

O[1, :, :]

-3.1	-3.9	-10.9
4.1		

O[2, :, :]

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

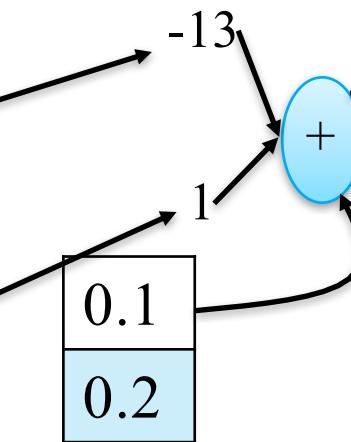
Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

1	-2
-2	1

W[1, 1, :, :]	W[2, 1, :, :]
3	1
2	2

1	0
0	1

W[1, 2, :, :]	W[2, 2, :, :]
0	0
0	4



Bias $\mathbf{b} = 2$
(nOutputPlane)

Image O = 2 x 3 x 3

O[1, :, :]

-3.1	-3.9	-10.9
4.1	-12.9	

O[2, :, :]

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

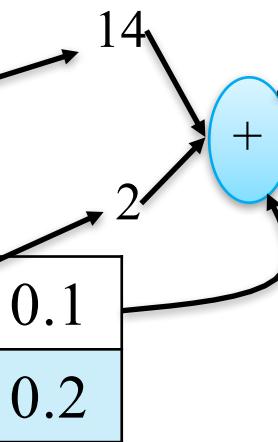
Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

1	-2
-2	1

W[1, 1, :, :]	W[2, 1, :, :]
3	1
2	2

1	0
0	1

W[1, 2, :, :]	W[2, 2, :, :]
0	0
0	4



Bias $\mathbf{b} = 2$
(nOutputPlane)

Image O = 2 x 3 x 3

O[1, :, :]

-3.1	-3.9	-10.9
4.1	-12.9	16.1

O[2, :, :]

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

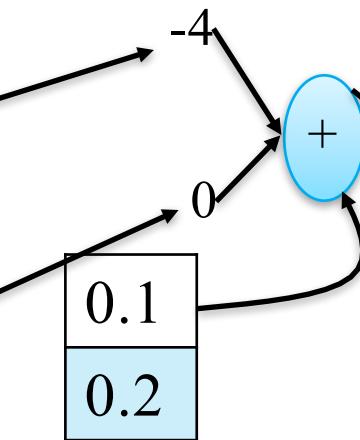


Image O = 2 x 3 x 3

O[1, :, :]

-3.1	-3.9	-10.9
4.1	-12.9	16.1
-3.9		

O[2, :, :]

Bias **b** = 2
(nOutputPlane)

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

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	

O[2, :, :]

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

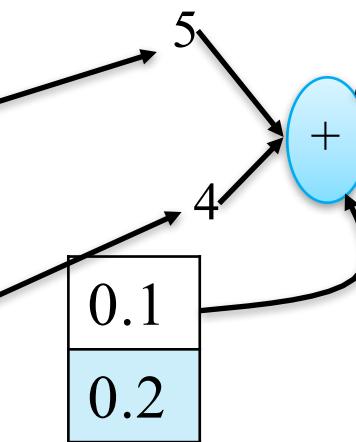


Image O = 2 x 3 x 3

O[1, :, :]

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

O[2, :, :]

Bias **b** = 2
(nOutputPlane)

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

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		

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

1	0
0	1

0	0
0	4

W[1, 2, :, :] W[2, 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	

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

1	0
0	1

0	0
0	4

W[1, 2, :, :]

W[2, 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

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

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

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

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	

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

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

Building Blocks - Convolution

Image I = 2 x 4 x 4

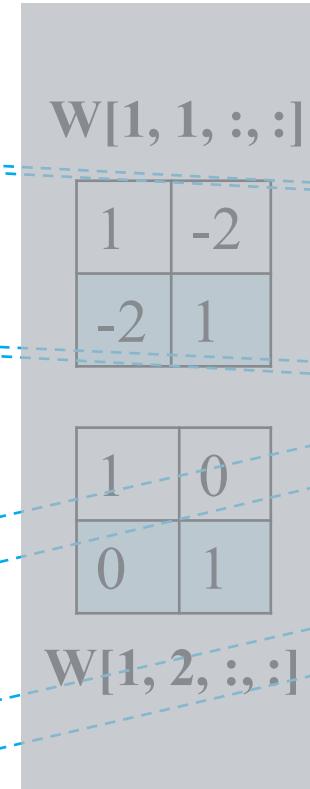
I[1, :, :]

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)



Bias $\mathbf{b} = 2$
(nOutputPlane)

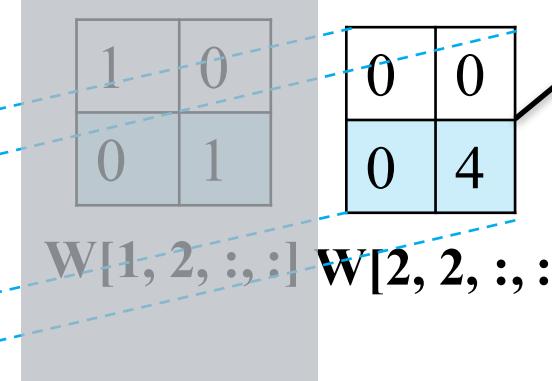


Image O = 2 x 3 x 3

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

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

Bias $\mathbf{b} = 2$
(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	2.2	

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

1	0
0	1

W[1, 2, :, :]

0	0
0	4

W[2, 2, :, :]

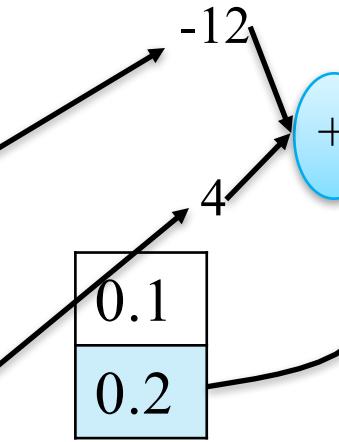


Image O = 2 x 3 x 3

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	2.2	-7.8

Building Blocks – Convolution – in Torch7

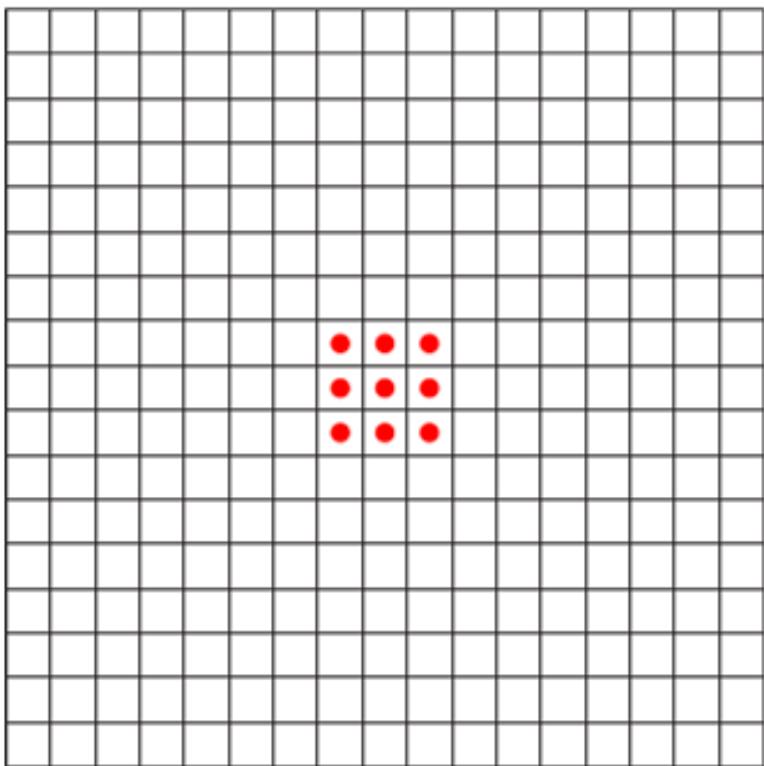
```
> I = torch.DoubleTensor({{{1,-2,2,2},{2,1,3,-2},{-2,3,-3,1},{-1,2,-4,2}}, {{3,0,0,0},{-2,-2,1,-1},{2,-1,3,1},{5,-2,0,1}}})  
> conv = nn.SpatialConvolutionMM(2,2,2,2)  
> conv.bias = torch.DoubleTensor({0.1, 0.2})  
> conv.weight = torch.DoubleTensor({{1,-2,-2,1,1,0,0,1},{3,1,2,2,0,0,0,4}})  
>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

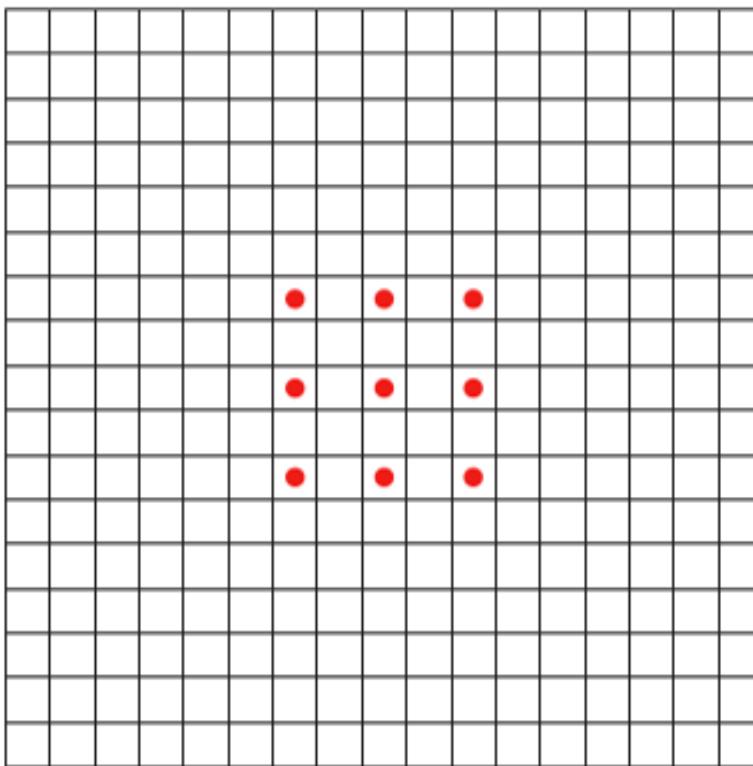
```
4  function SpatialConvolutionMM:__init__(nInputPlane, nOutputPlane, kW, kH, dW, dH, padW, padH)
5      parent.__init__(self)
6
7      dW = dW or 1
8      dH = dH or 1
9
10     self.nInputPlane = nInputPlane
11     self.nOutputPlane = nOutputPlane
12     self.kW = kW
13     self.kH = kH
14
15     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>

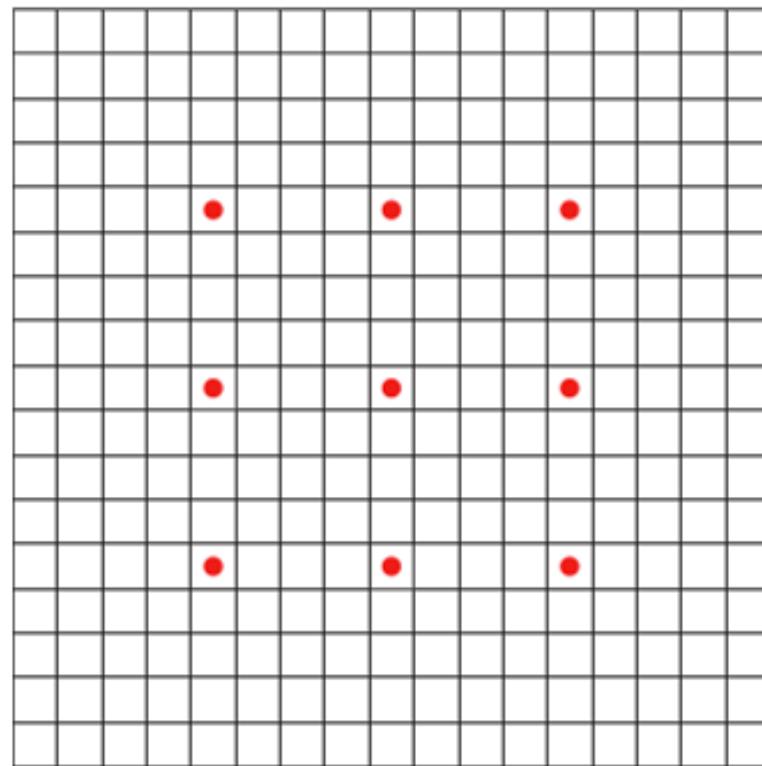
Aside: Dilated Convolution



(a)



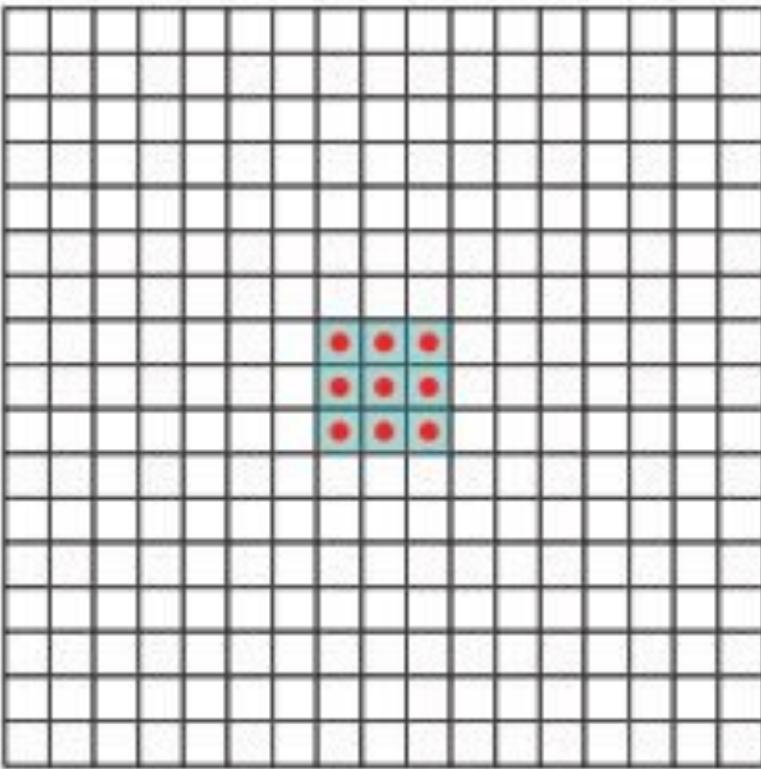
(b)



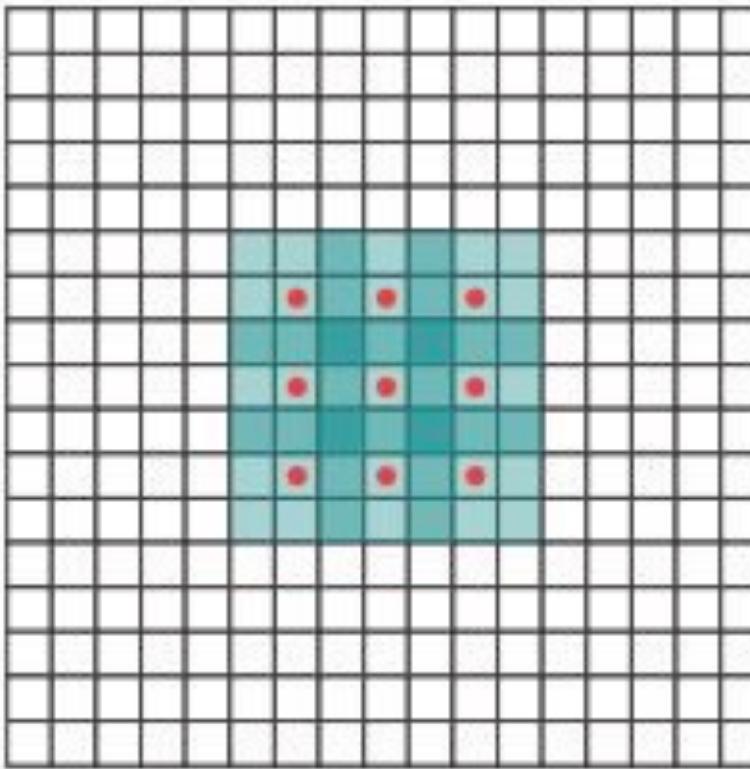
(c)

Aside: Dilated Convolution

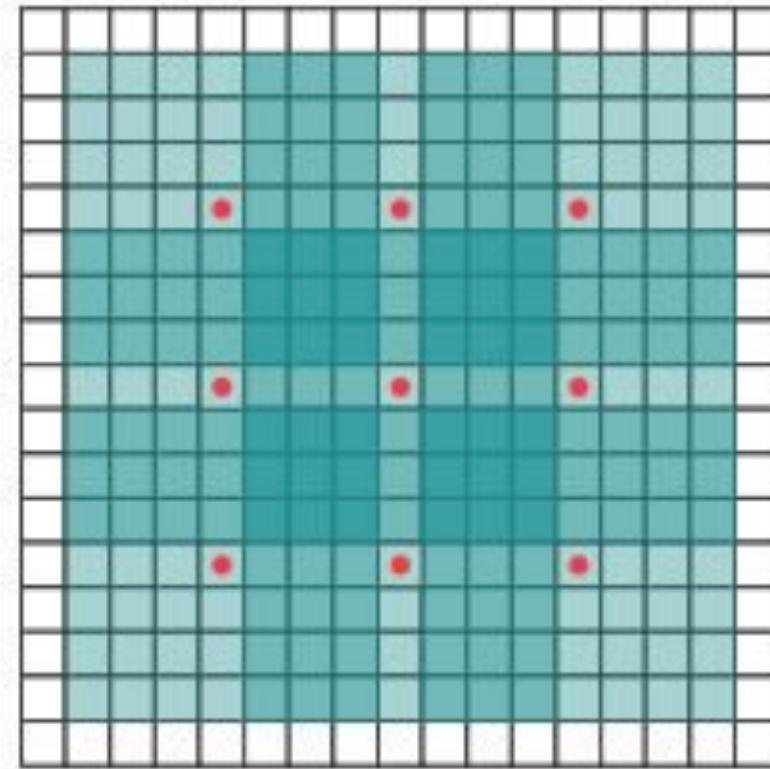
[nn.SpatialDilatedConvolution](#)



(a)



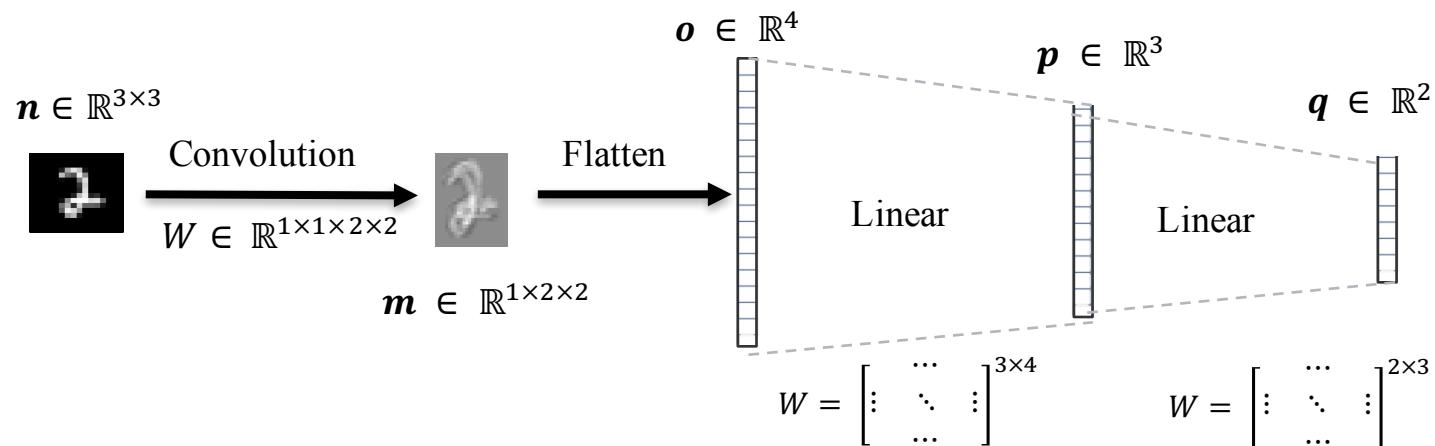
(b)



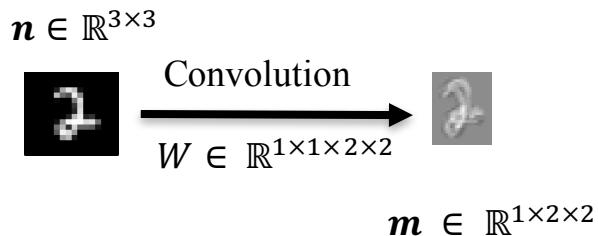
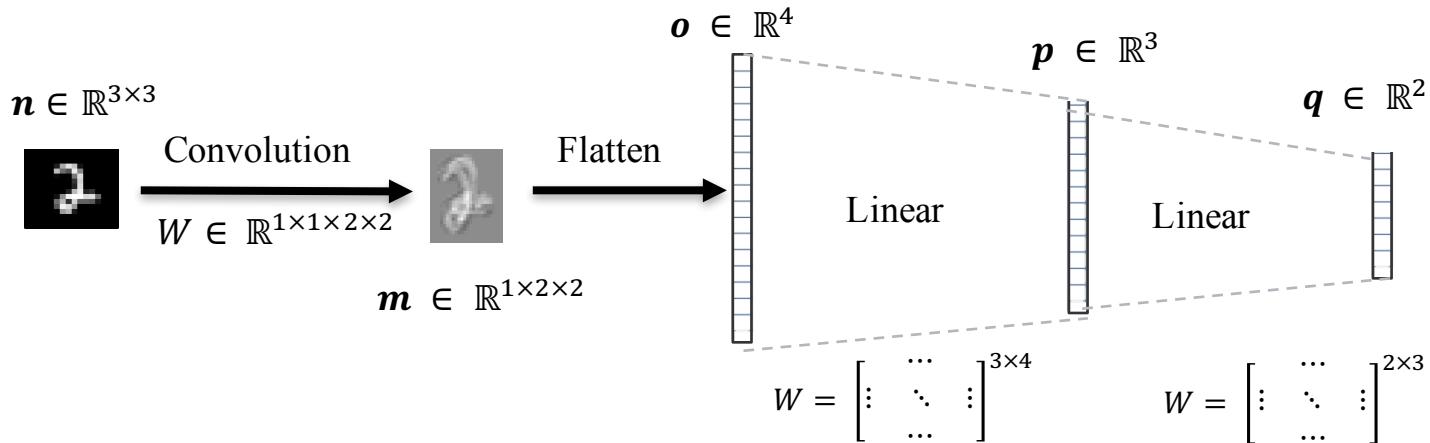
(c)

Multi-Scale Context Aggregation by Dilated Convolutions
[Fisher Yu, Vladlen Koltun](#)

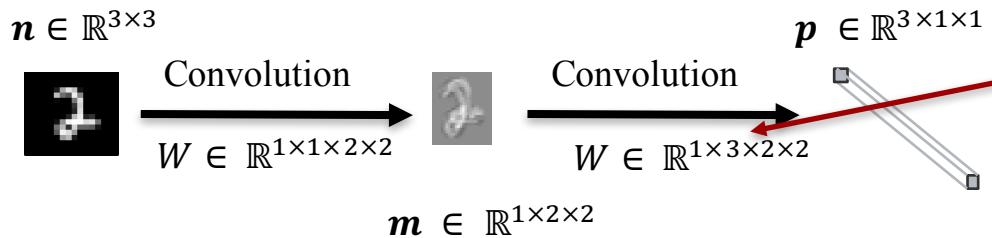
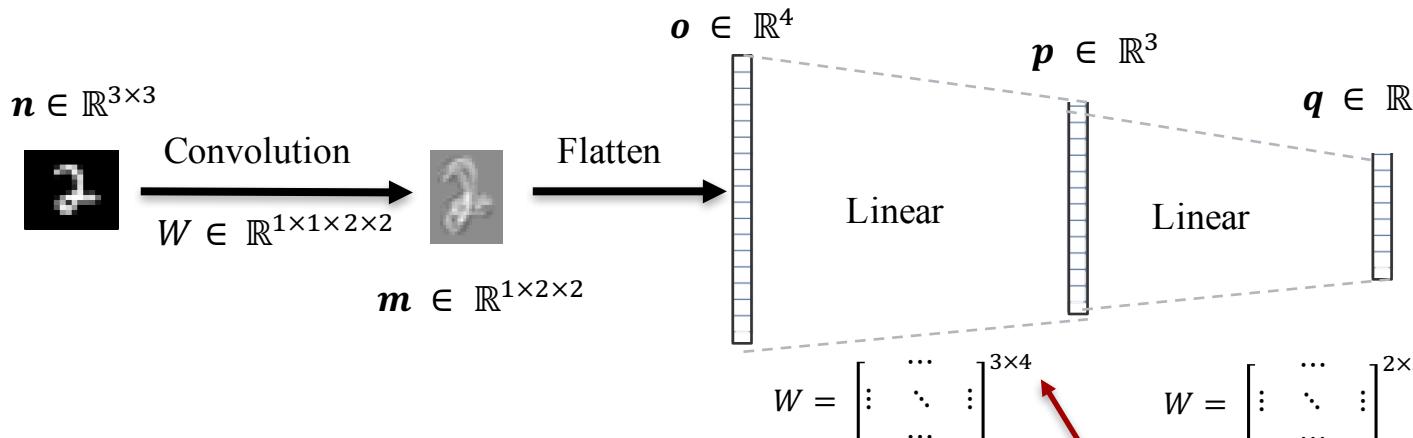
Everything is a Convolution!



Everything is a Convolution!

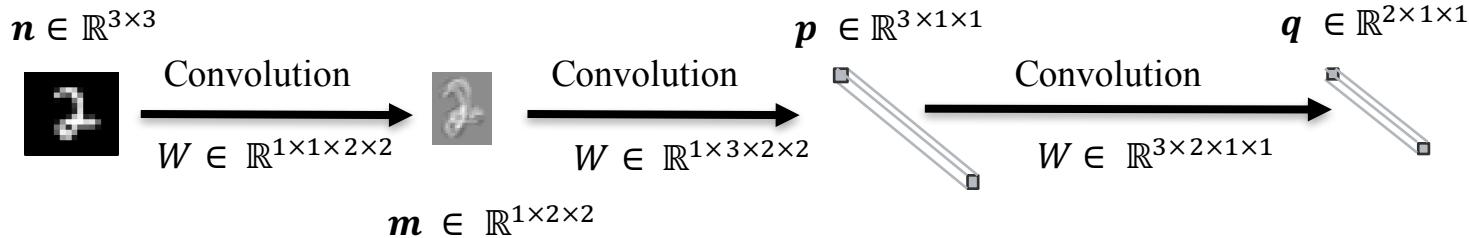
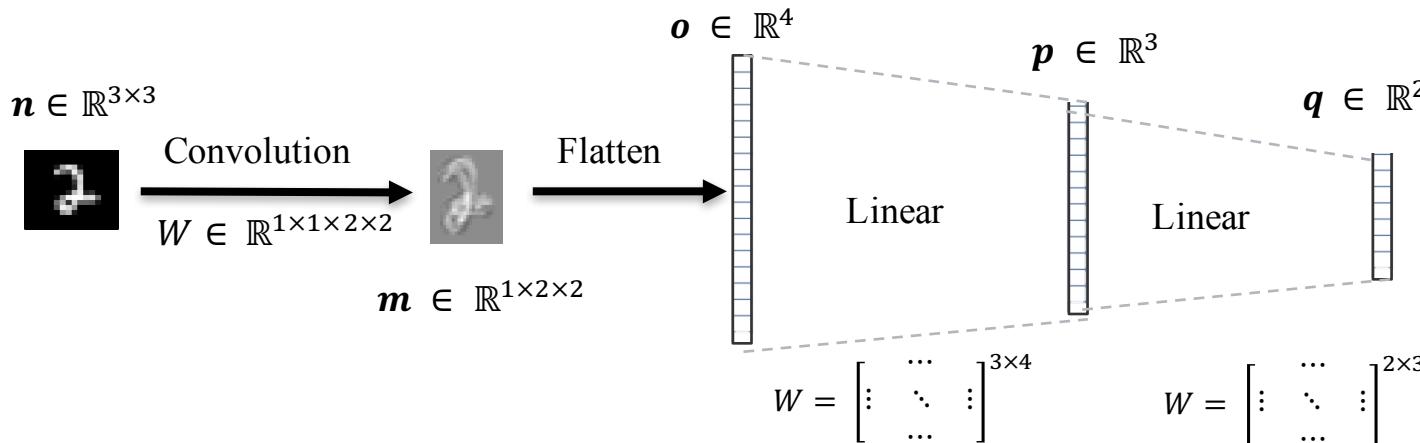


Everything is a Convolution!



Each output channel ($W[:, I, :, :]$) corresponds to a row I of the matrix

Everything is a Convolution!



Thank you!