

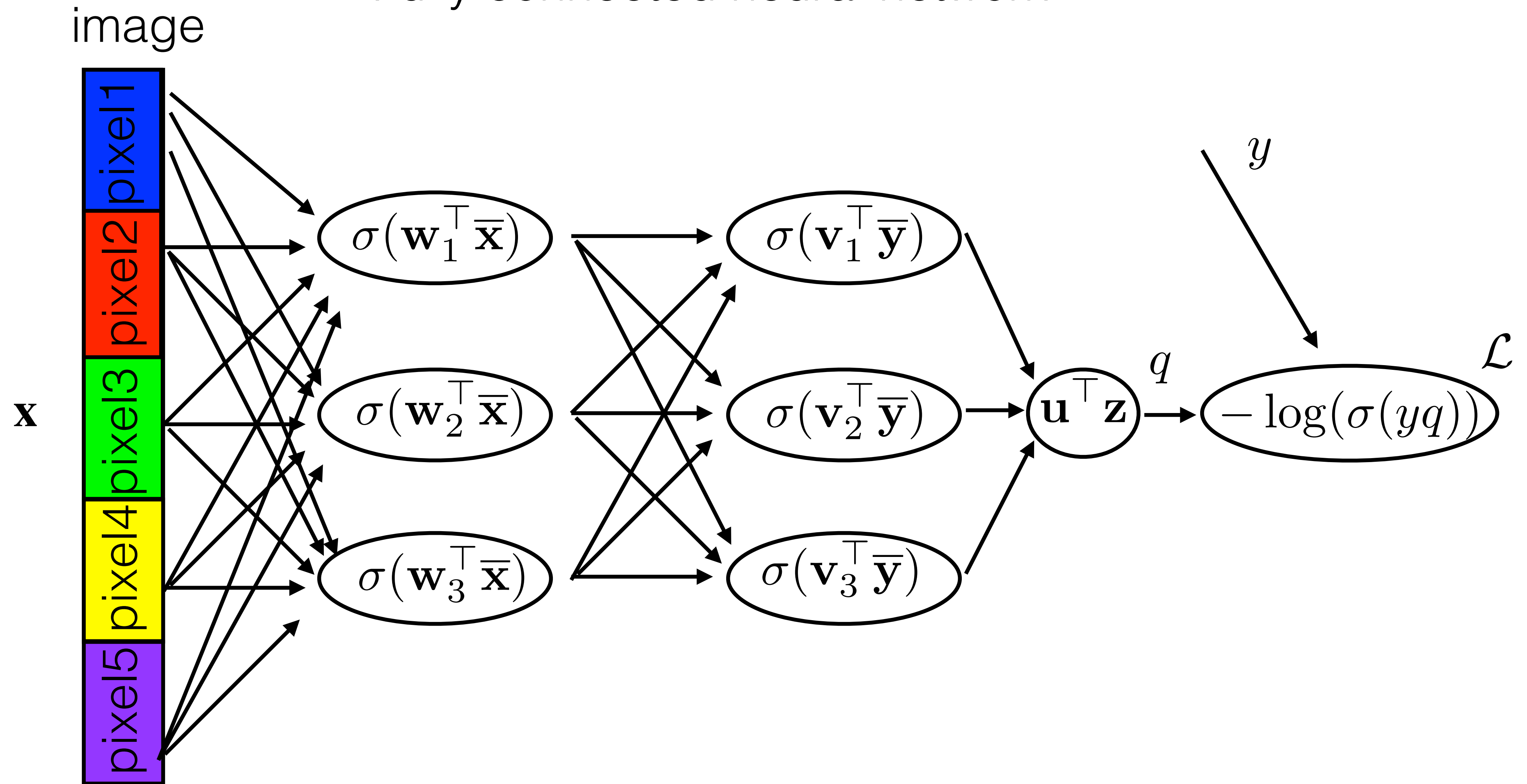
The story of the cat's brain surgery

cortex, convolution layer, its vector-Jacobian product, feature maps, low-dimensional encoding, and fun with pre-trained convnet

Karel Zimmermann
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics



Fully connected neural network



Learning prone to overfitting, the structure is too general, the resulting function is wild

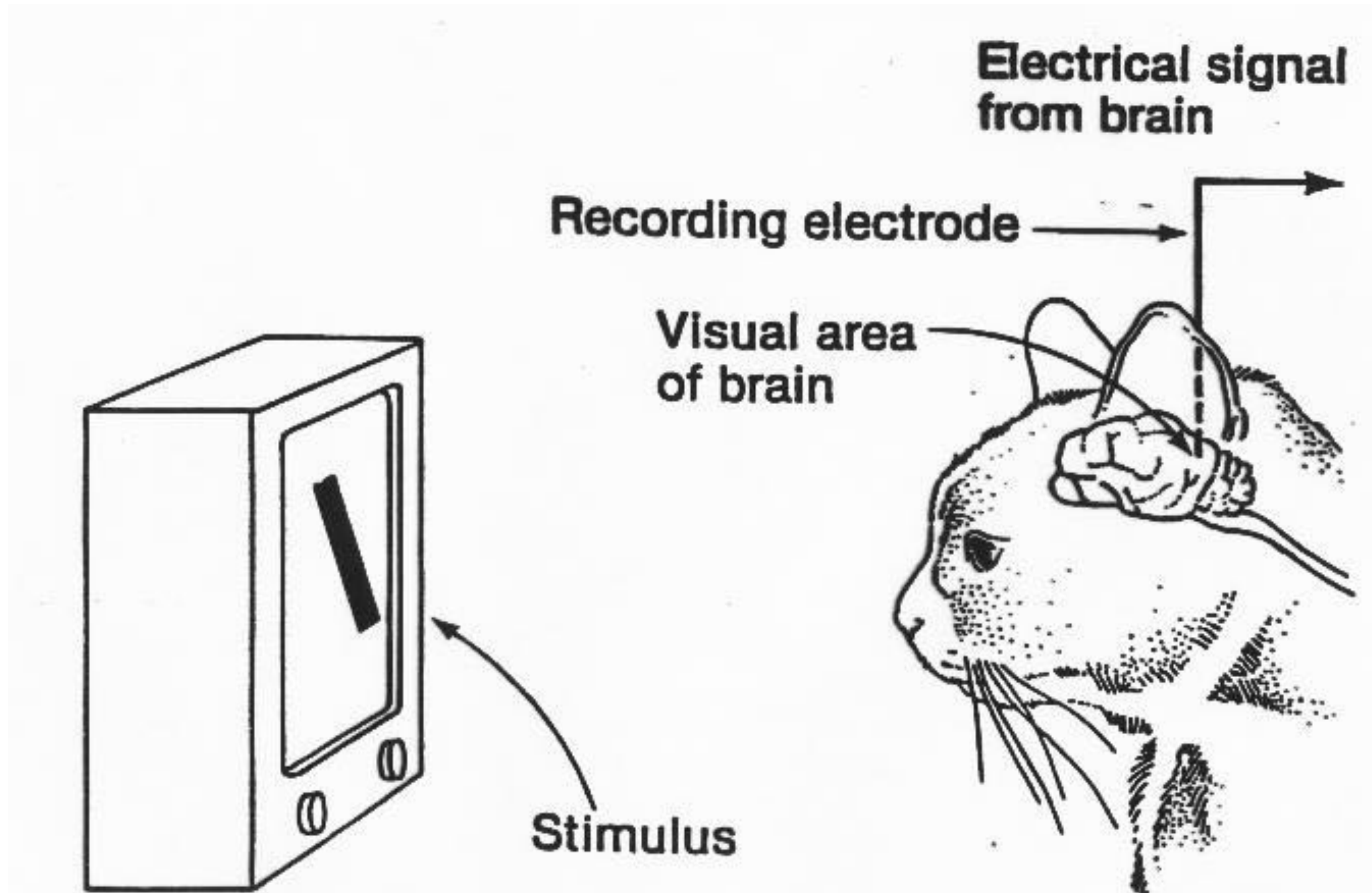
The Tungsten Electrode [Hubel-Science-1957]



<http://braintour.harvard.edu/archives/portfolio-items/hubel-and-wiesel>

- Device capable to record signal from a single neuron

[Hubel and Wiesel 1959]



- Experiment with anaesthetised paralysed cat

[Hubel and Wiesel 1960]

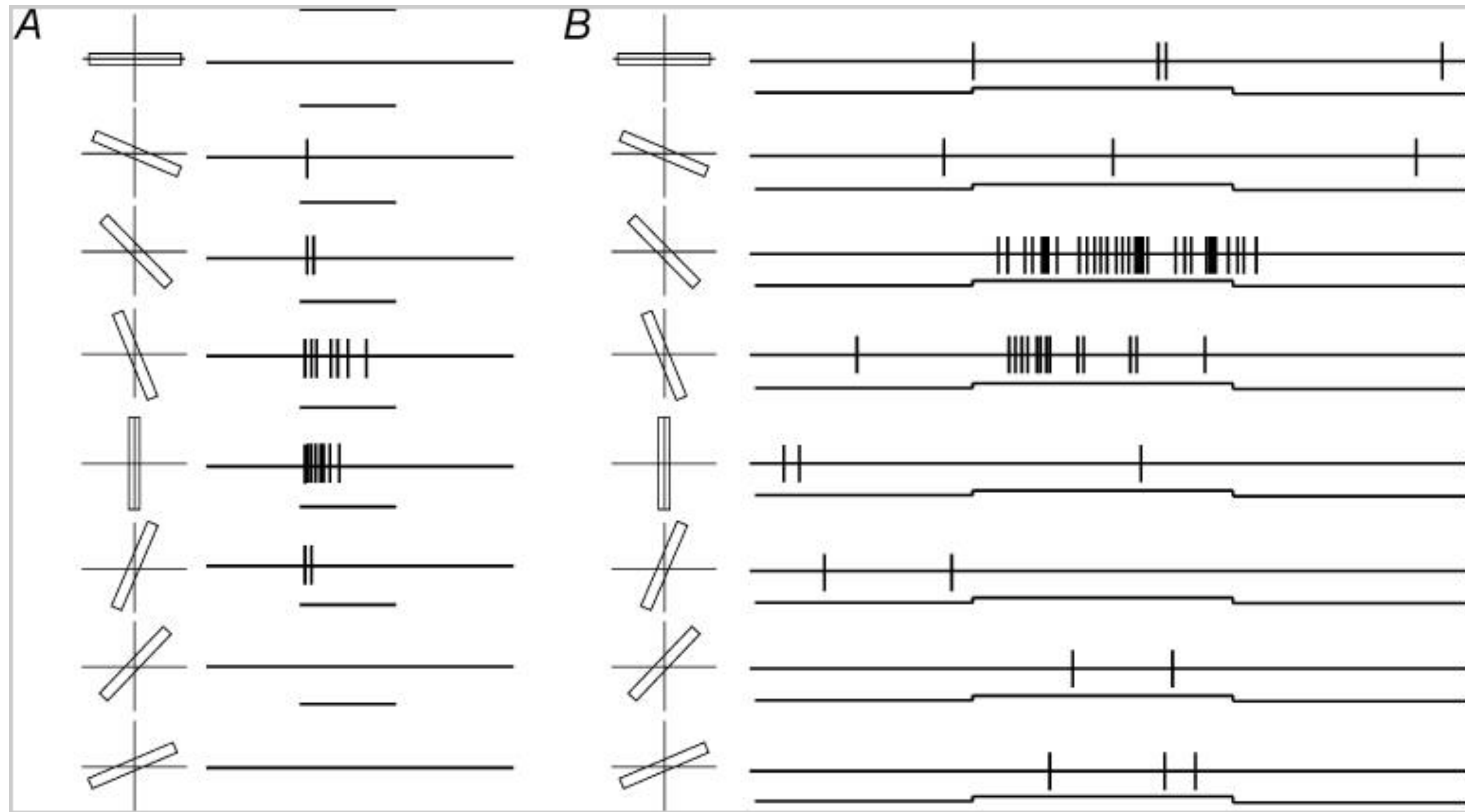


- Edge sensitivity
- Topographical mapping (nearby neurons process information from nearby visual fields)
- Translation invariance (the same edge is detected at all positions)

[Hubel and Wiesel 1960]

paralysed cat

awake monkey



<https://knowingneurons.com/2014/10/29/hubel-and-wiesel-the-neural-basis-of-visual-perception/>

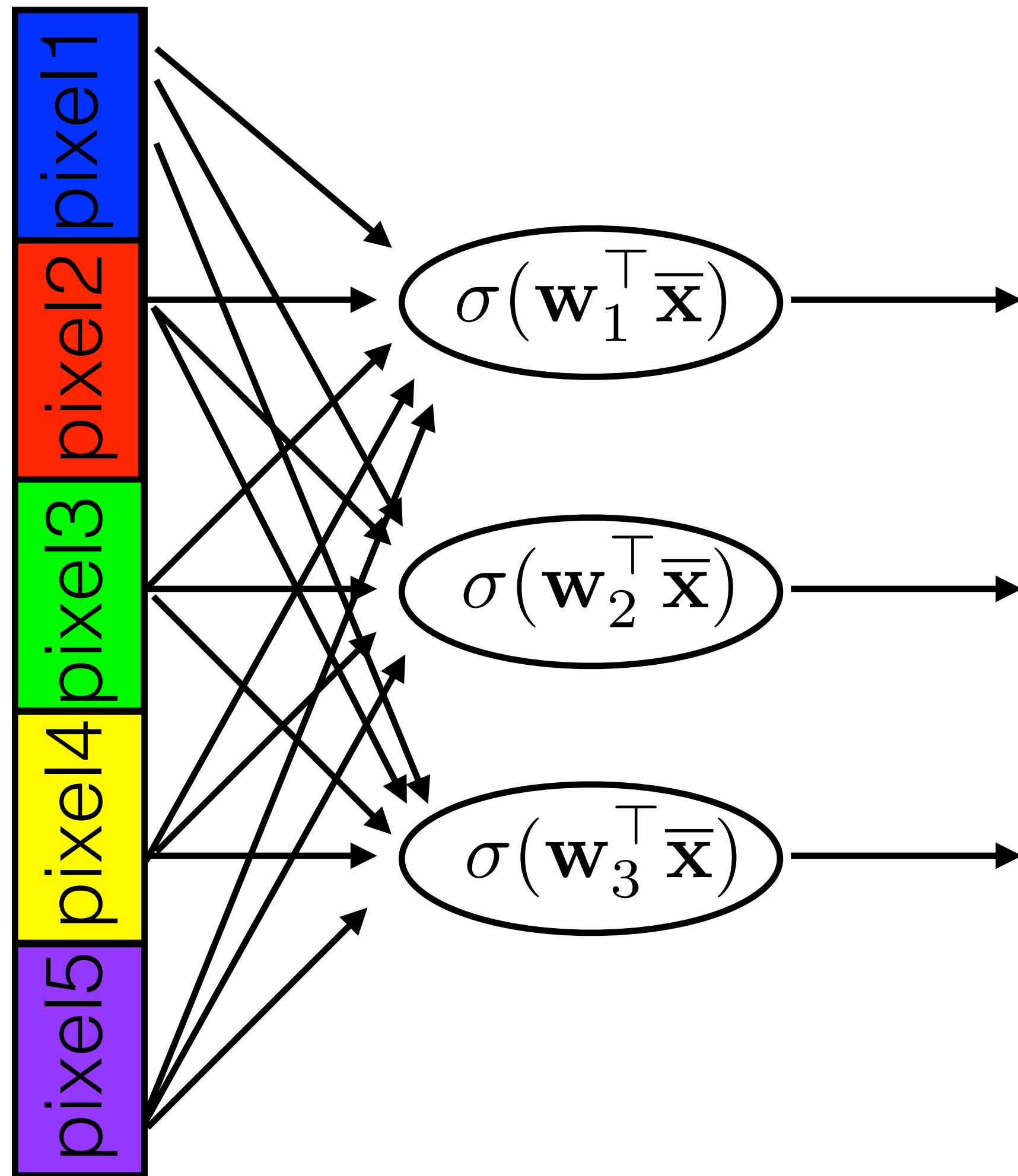
Hubel and Wiesel experiments in 1950s and 1960s



- Nobel Prize in Physiology and Medicine in 1981
- Dr. Hubel: “There has been a myth that the brain cannot understand itself. It is compared to a man trying to lift himself by his own bootstraps. We feel that is nonsense. The brain can be studied just as the kidney can.”

1. **Topographical map:** nearby neurons process information from nearby visual fields

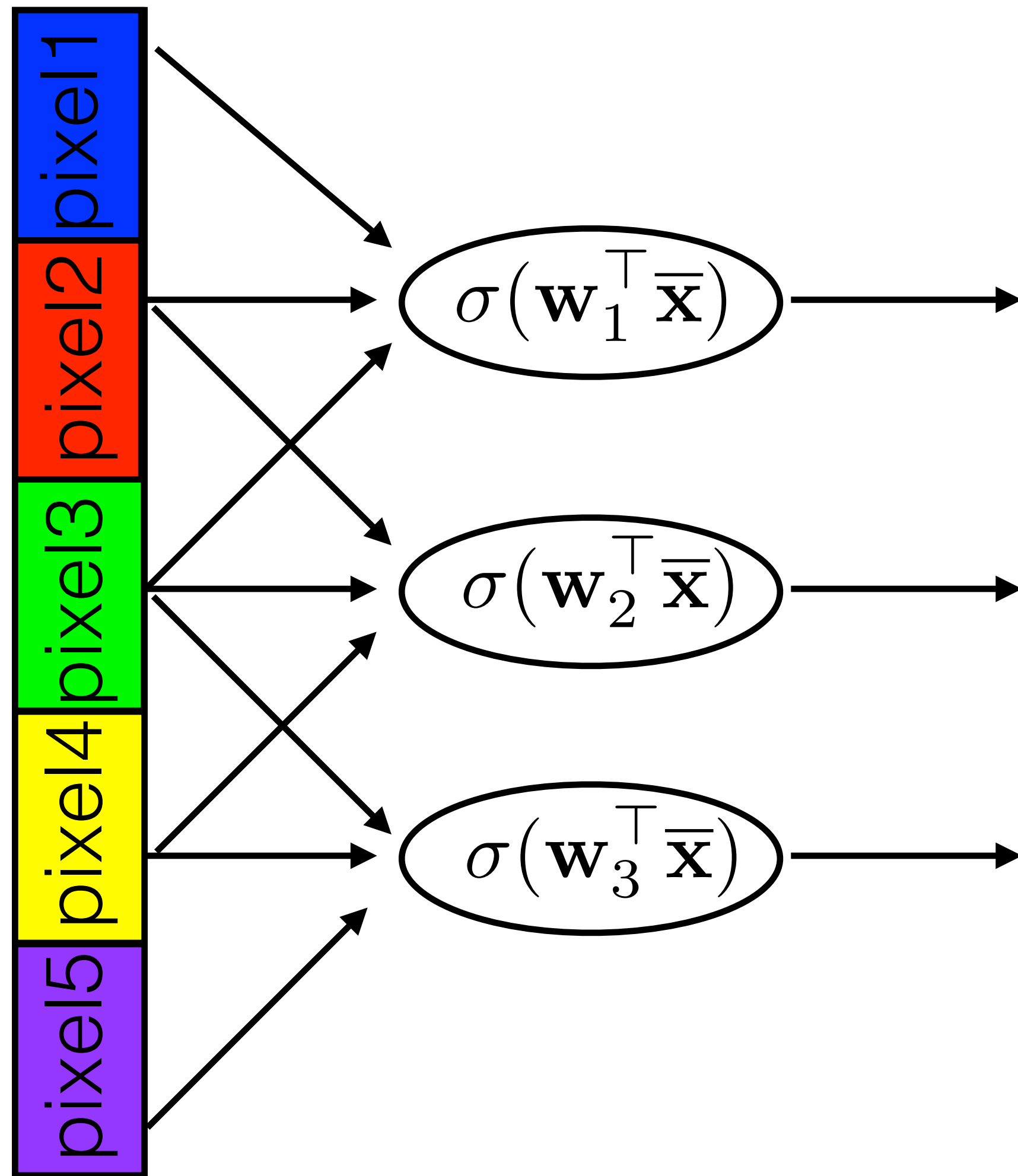
image



- Processing of visual information in cortex is not fully connected.

1. **Topographical map:** nearby neurons process information from nearby visual fields

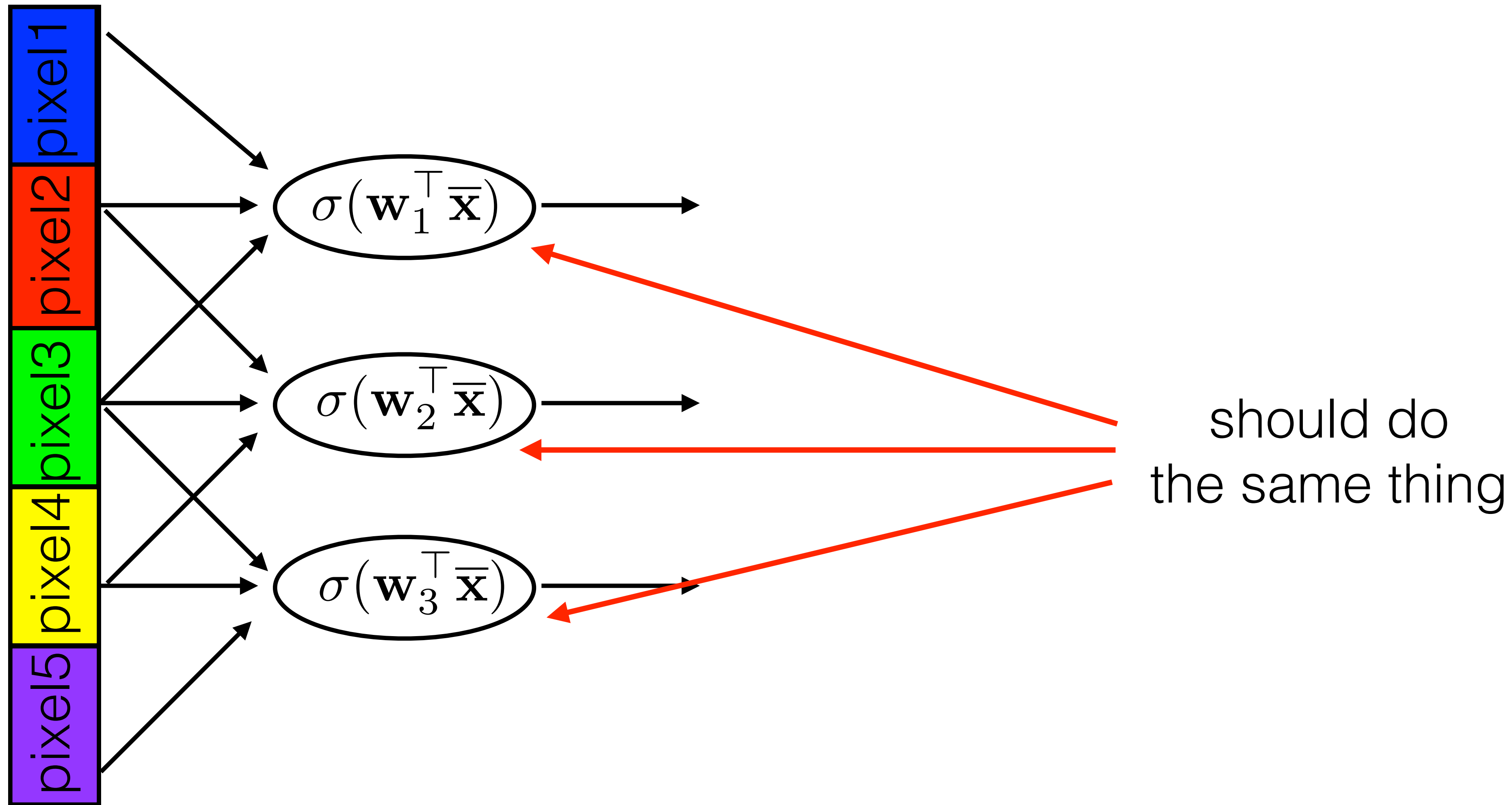
image



- Processing of visual information in cortex is not fully connected.

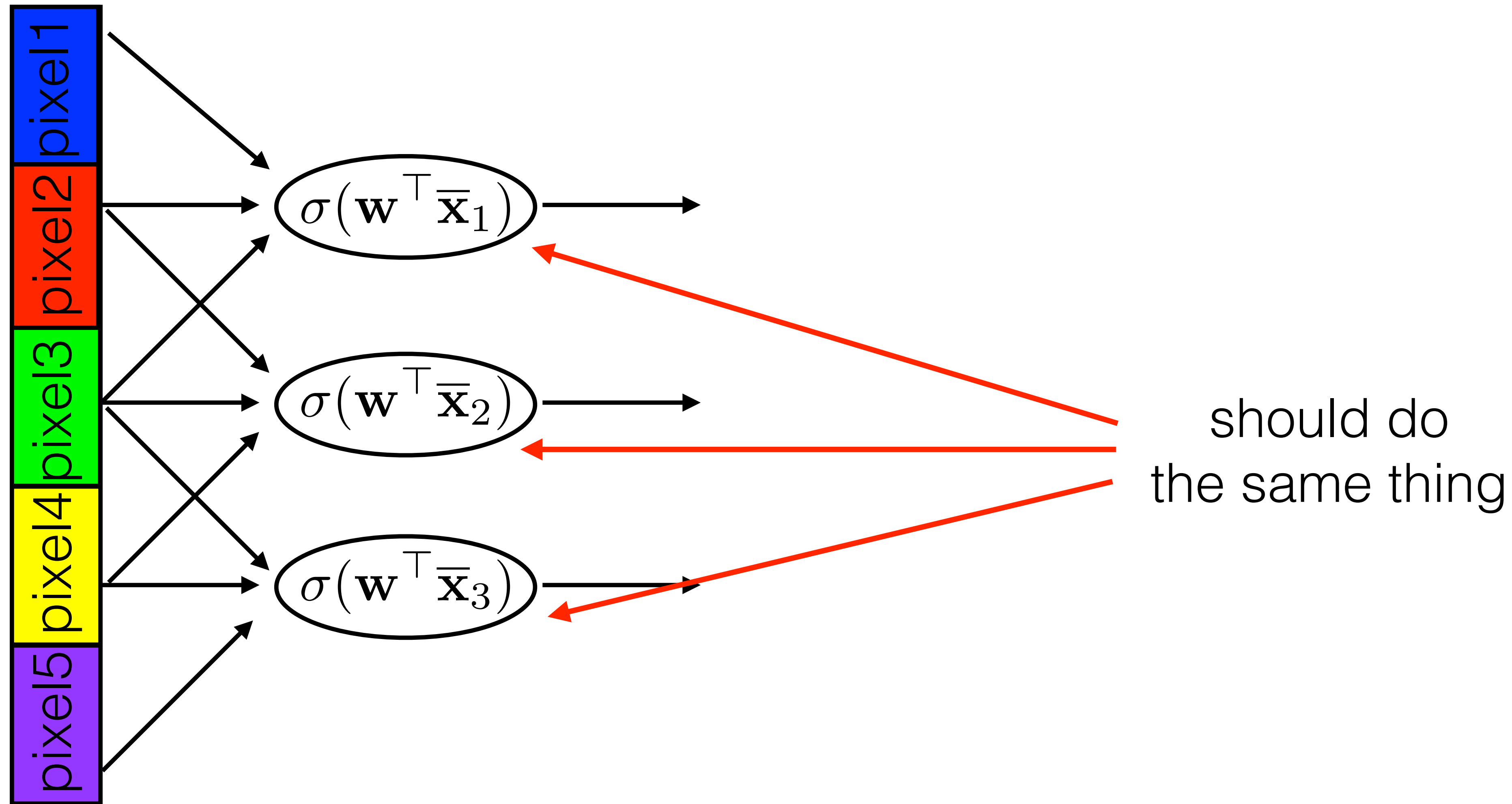
2. **Translation invariance:** the same edge is detected at all positions

image



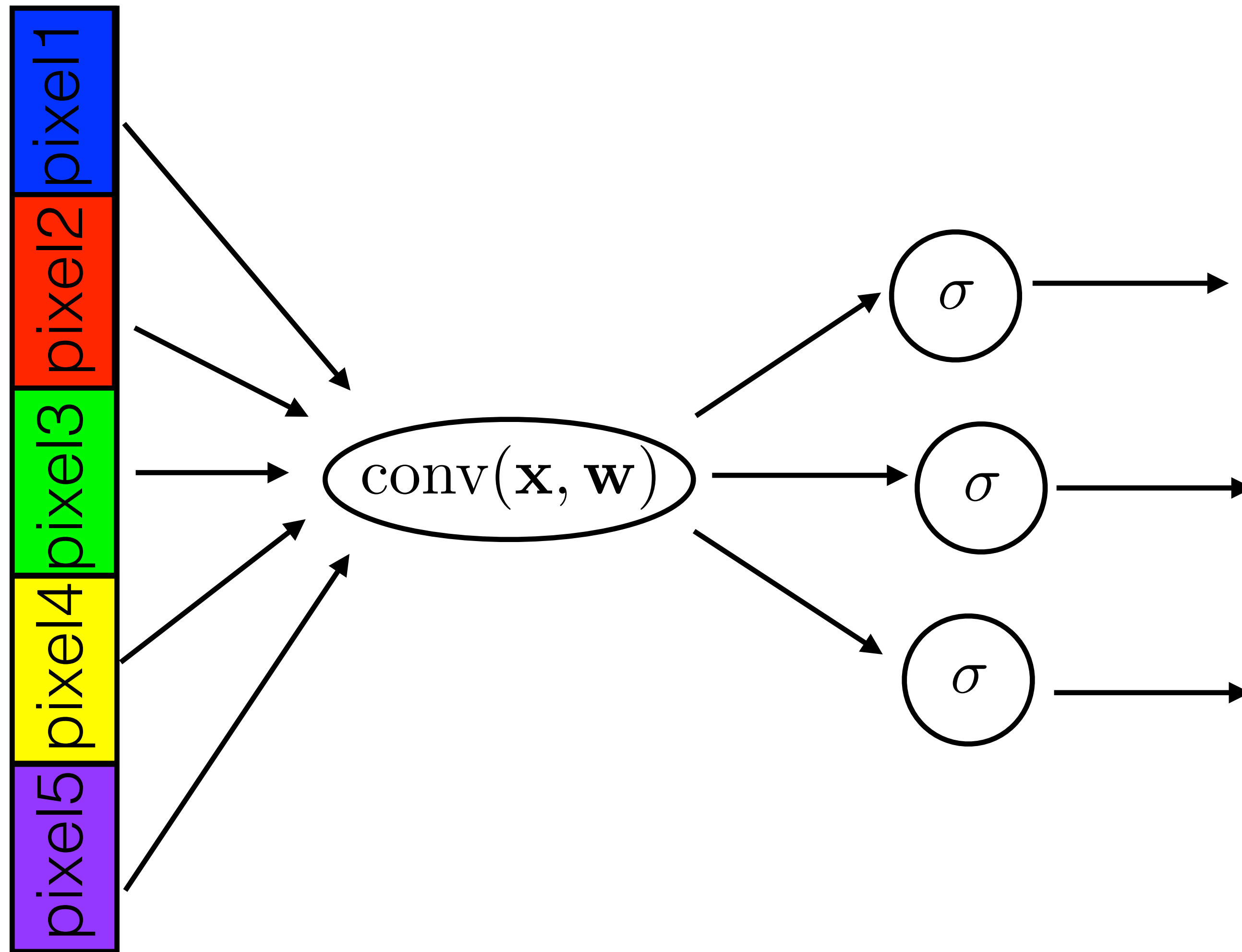
2. **Translation invariance:** the same edge is detected at all positions

image



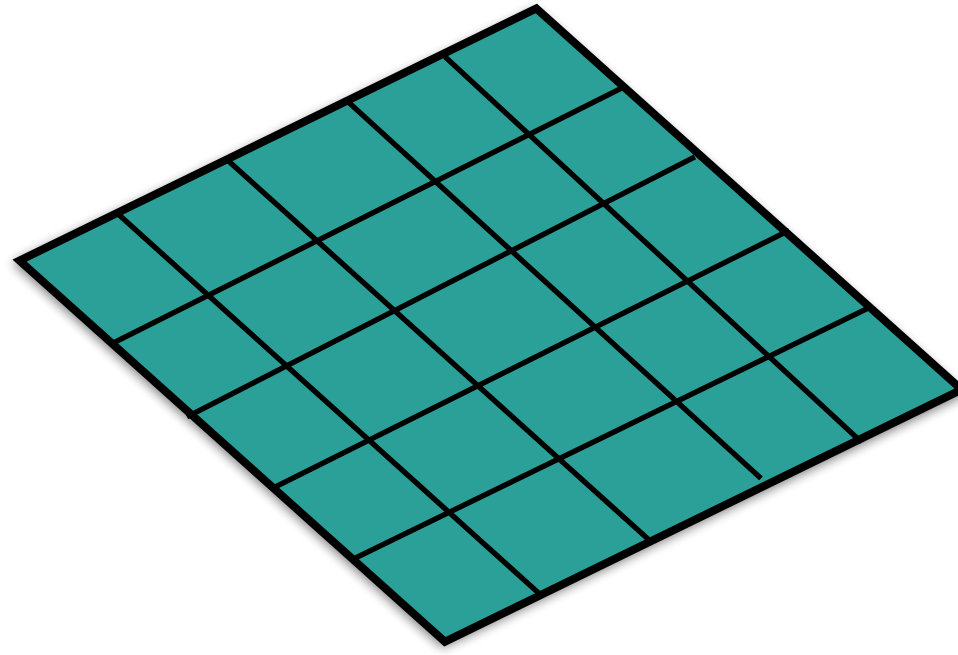
2. **Translation invariance:** the same edge is detected at all positions

image

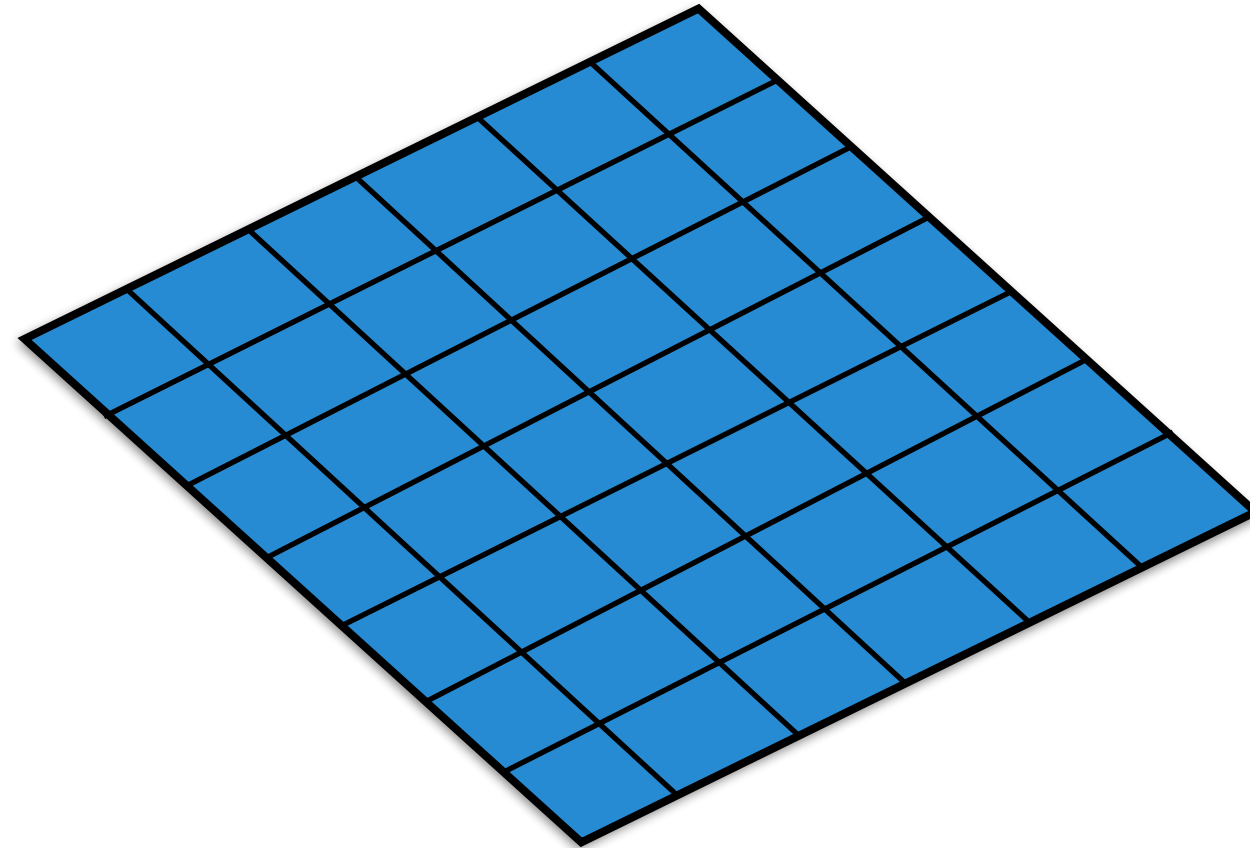


Fully Connected layer on images

output
(fcnn layer with
5x5 neurons)



input
(7x7 image)



Fully-connected layer

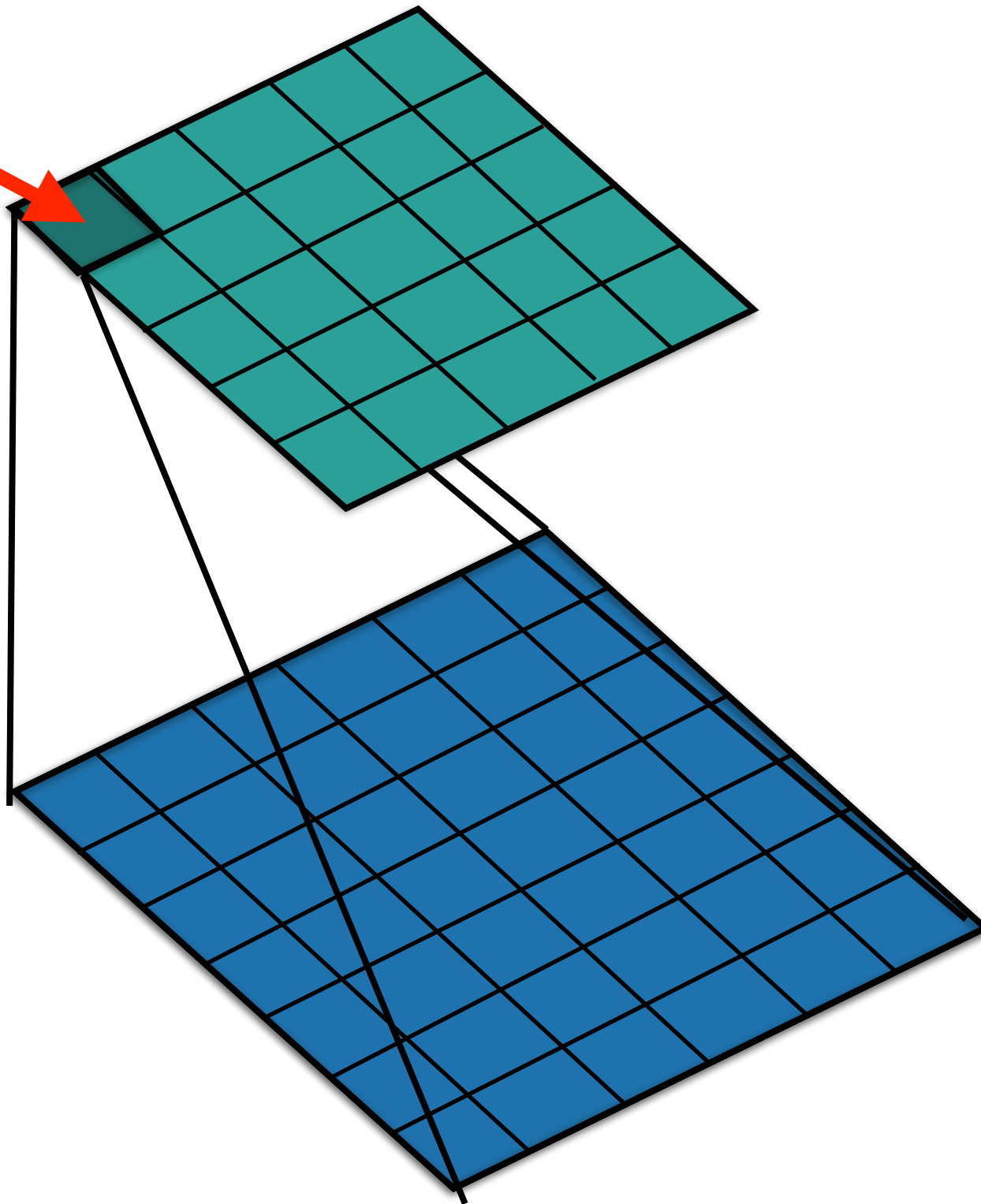
Fully Connected layer on images

$$\sigma(\sum \mathbf{w}_1 \cdot \mathbf{X})$$

output
(fcnn layer with
5x5 neurons)

input
(7x7 image)

Fully-connected layer



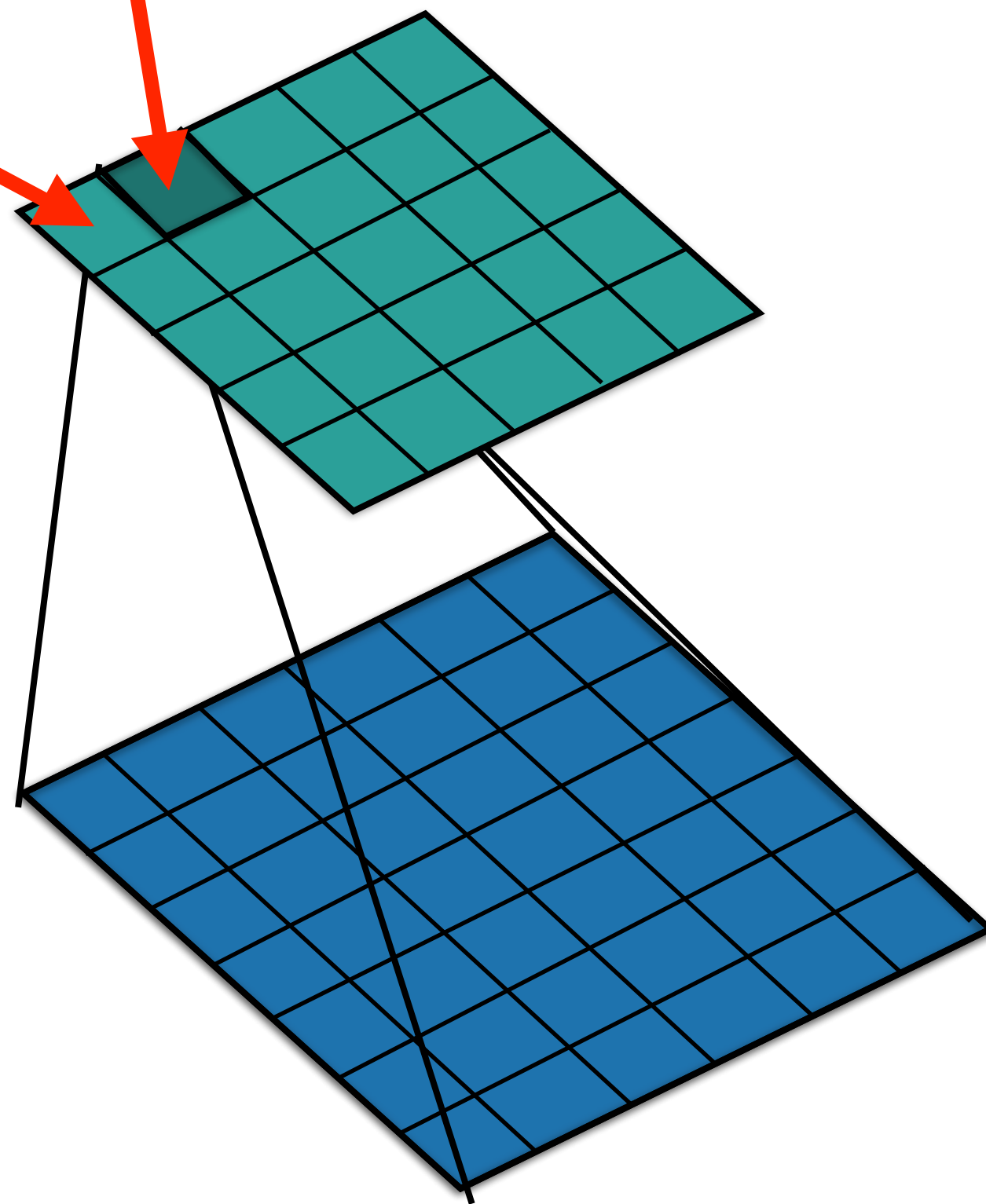
Fully Connected layer on images

$$\sigma(\sum \mathbf{w}_1 \cdot \mathbf{X}) \quad \sigma(\sum \mathbf{w}_2 \cdot \mathbf{X})$$

output
(fcnn layer with
5x5 neurons)

input
(7x7 image)

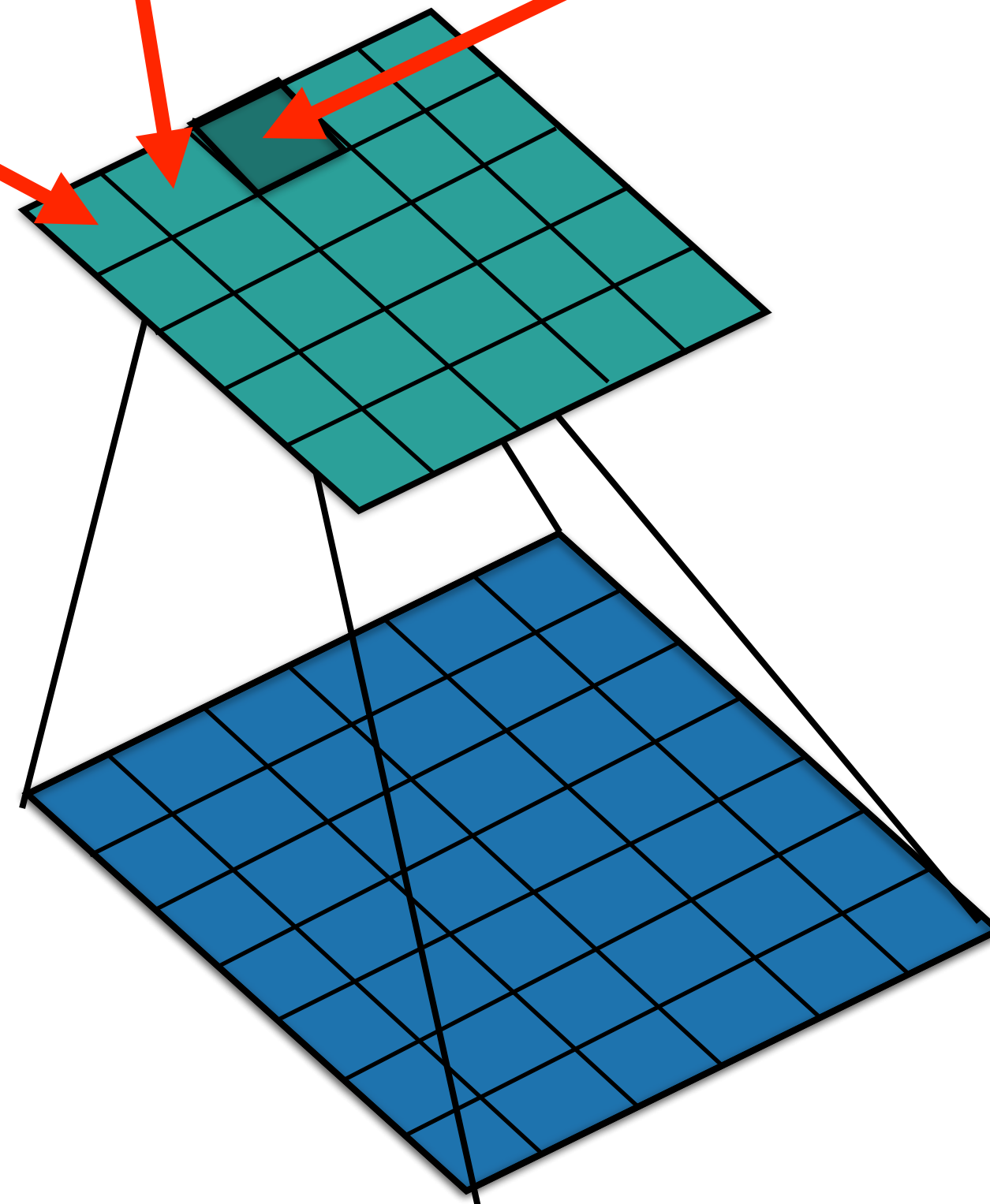
Fully-connected layer



Fully Connected layer on images

$$\sigma(\sum \mathbf{w}_1 \cdot \mathbf{X}) \quad \sigma(\sum \mathbf{w}_2 \cdot \mathbf{X}) \quad \sigma(\sum \mathbf{w}_3 \cdot \mathbf{X})$$

output
(fcnn layer with
5x5 neurons)



input
(7x7 image)

Fully-connected layer

How many weights do I have in total?

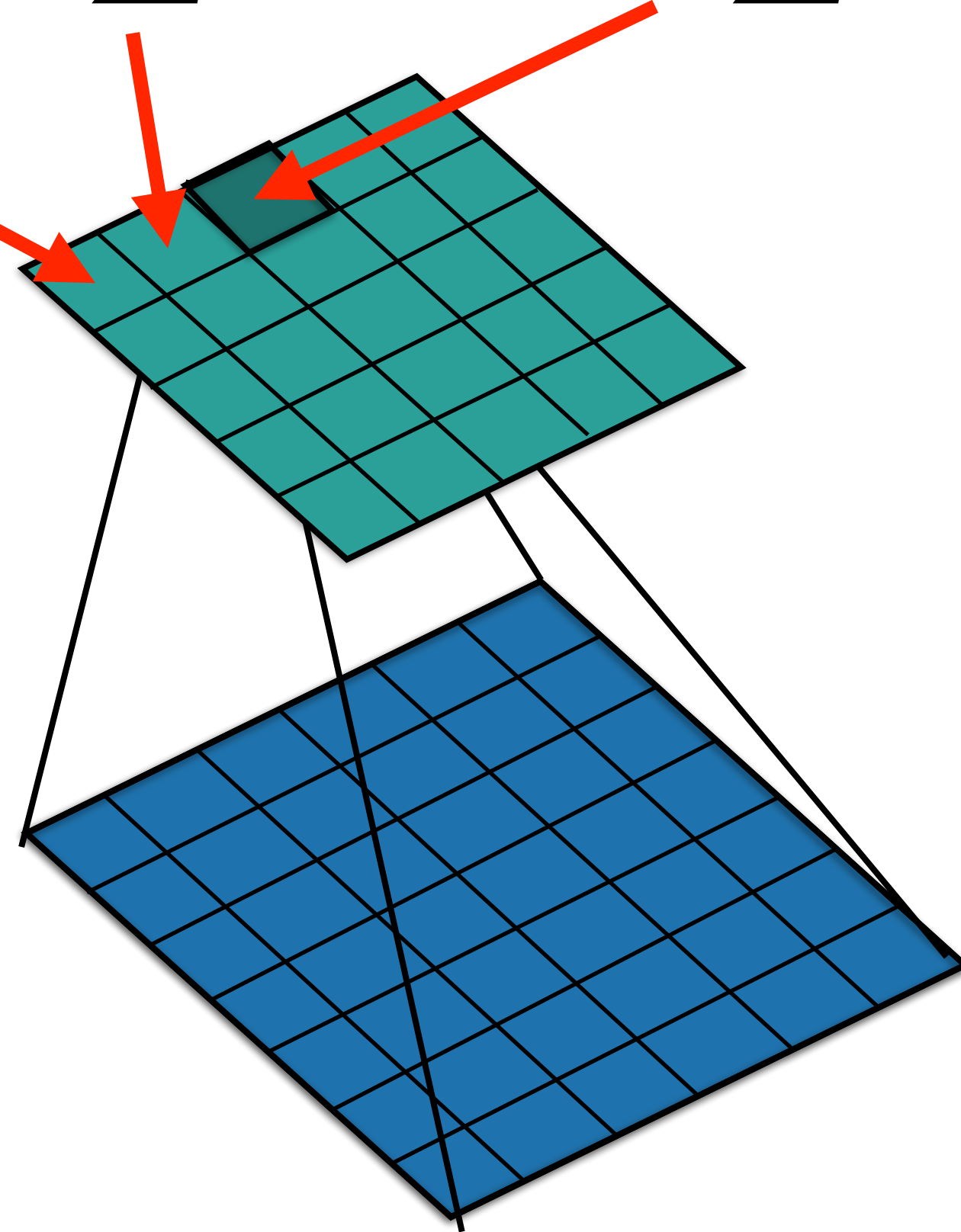
$$25 \times (7 \times 7 + 1) = 1250$$

Fully Connected layer on images

$$\sigma(\sum \mathbf{w}_1 \cdot \mathbf{X}) \quad \sigma(\sum \mathbf{w}_2 \cdot \mathbf{X}) \quad \sigma(\sum \mathbf{w}_3 \cdot \mathbf{X})$$

output
(fcnn layer with
5x5 neurons)

input
(7x7 image)

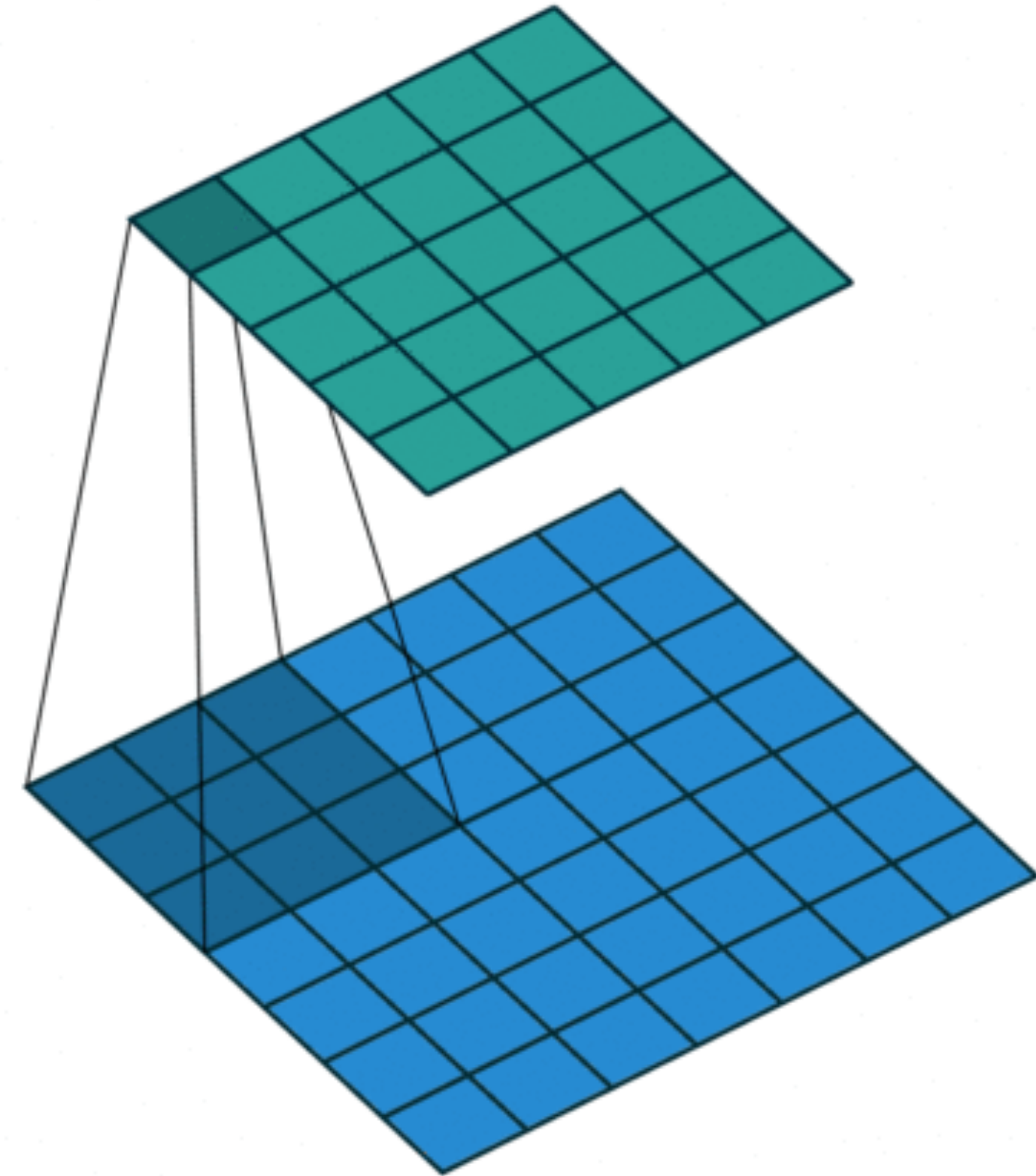


Fully-connected layer

How many weights do I have in total?

$$25 \times (7 \times 7 + 1) = 1250$$

$$\sigma(\text{conv}(\mathbf{X}, \mathbf{w}))$$

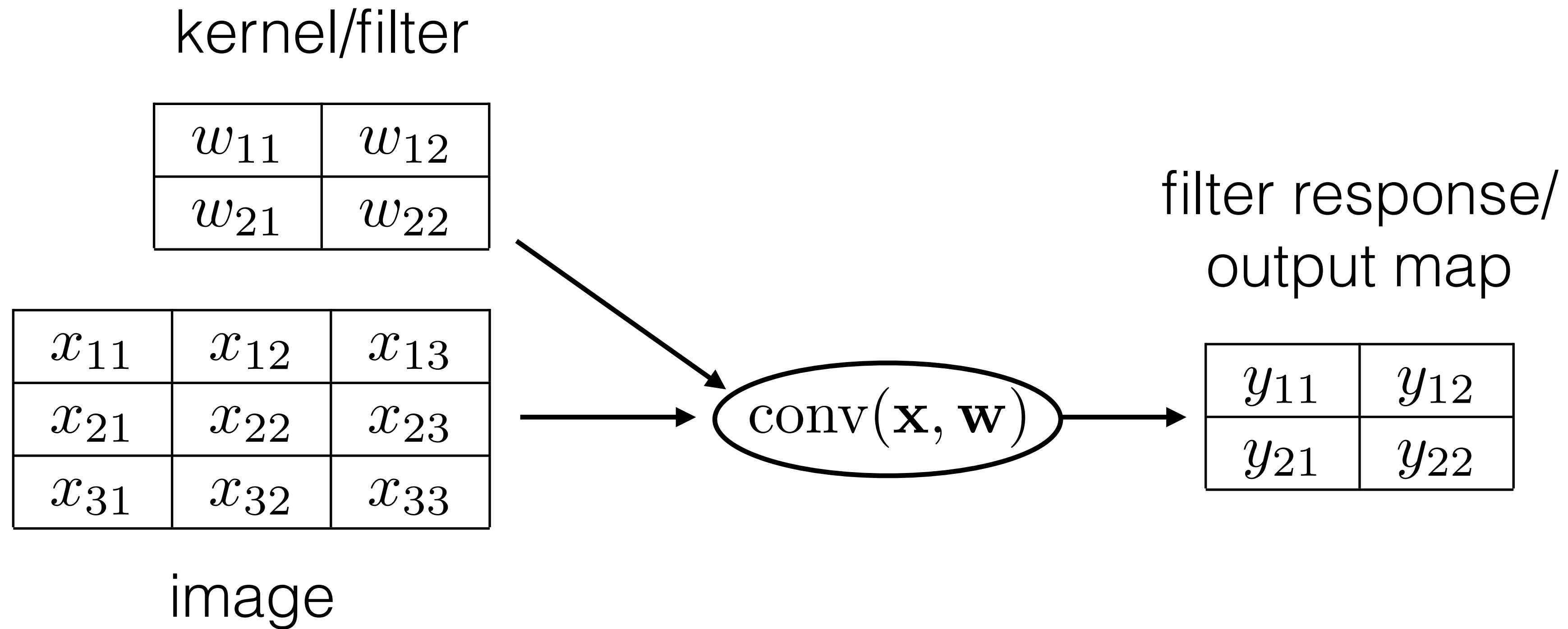


Convolutional layer

How many weights do I have in total?

$$3 \times 3 + 1 = 10$$

Convolution forward pass $\mathbf{y} = \text{conv}(\mathbf{x}, \mathbf{w})$



Convolution forward pass $\mathbf{y} = \text{conv}(\mathbf{x}, \mathbf{w})$

Local linear classifier run in double-for-loop over rows and columns

$$\begin{array}{|c|c|} \hline y_{11} & y_{12} \\ \hline y_{21} & y_{22} \\ \hline \end{array} = \text{conv} \left(\begin{array}{|c|c|c|} \hline x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \hline x_{31} & x_{32} & x_{33} \\ \hline \end{array}, \begin{array}{|c|c|} \hline w_{11} & w_{12} \\ \hline w_{21} & w_{22} \\ \hline \end{array} \right)$$

$$y_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$

$$y_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$$

$$y_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{21}x_{31} + w_{22}x_{32}$$

$$y_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{21}x_{32} + w_{22}x_{33}$$

Convolution forward pass $\mathbf{y} = \text{conv}(\mathbf{x}, \mathbf{w})$

Local linear classifier run in double-for-loop over rows and columns

$$\begin{array}{|c|c|} \hline y_{11} & y_{12} \\ \hline y_{21} & y_{22} \\ \hline \end{array} = \text{conv} \left(\begin{array}{|c|c|c|} \hline x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \hline x_{31} & x_{32} & x_{33} \\ \hline \end{array}, \begin{array}{|c|c|} \hline w_{11} & w_{12} \\ \hline w_{21} & w_{22} \\ \hline \end{array} \right)$$

$$y_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$

$$y_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$$

$$y_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{21}x_{31} + w_{22}x_{32}$$

$$y_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{21}x_{32} + w_{22}x_{33}$$

Convolution forward pass $\mathbf{y} = \text{conv}(\mathbf{x}, \mathbf{w})$

Local linear classifier run in double-for-loop over rows and columns

$$\begin{array}{|c|c|} \hline y_{11} & y_{12} \\ \hline y_{21} & y_{22} \\ \hline \end{array} = \text{conv} \left(\begin{array}{|c|c|c|} \hline x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \hline x_{31} & x_{32} & x_{33} \\ \hline \end{array}, \begin{array}{|c|c|} \hline w_{11} & w_{12} \\ \hline w_{21} & w_{22} \\ \hline \end{array} \right)$$

$$y_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$

$$y_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$$

$$y_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{21}x_{31} + w_{22}x_{32}$$

$$y_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{21}x_{32} + w_{22}x_{33}$$

Convolution forward pass $\mathbf{y} = \text{conv}(\mathbf{x}, \mathbf{w})$

Local linear classifier run in double-for-loop over rows and columns

$$\begin{array}{|c|c|} \hline y_{11} & y_{12} \\ \hline y_{21} & y_{22} \\ \hline \end{array} = \text{conv} \left(\begin{array}{|c|c|c|} \hline x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \hline x_{31} & x_{32} & x_{33} \\ \hline \end{array}, \begin{array}{|c|c|} \hline w_{11} & w_{12} \\ \hline w_{21} & w_{22} \\ \hline \end{array} \right)$$

$$y_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$

$$y_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$$

$$y_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{21}x_{31} + w_{22}x_{32}$$

$$y_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{21}x_{32} + w_{22}x_{33}$$

Convolution forward pass $\mathbf{y} = \text{conv}(\mathbf{x}, \mathbf{w})$

Local linear classifier run in double-for-loop over rows and columns

$$\begin{array}{|c|c|} \hline y_{11} & y_{12} \\ \hline y_{21} & y_{22} \\ \hline \end{array} = \text{conv} \left(\begin{array}{|c|c|c|} \hline x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \hline x_{31} & x_{32} & x_{33} \\ \hline \end{array}, \begin{array}{|c|c|} \hline w_{11} & w_{12} \\ \hline w_{21} & w_{22} \\ \hline \end{array} \right)$$

$$y_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$

$$y_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$$

$$y_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{21}x_{31} + w_{22}x_{32}$$

$$y_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{21}x_{32} + w_{22}x_{33}$$

Convolution layer properties - output size

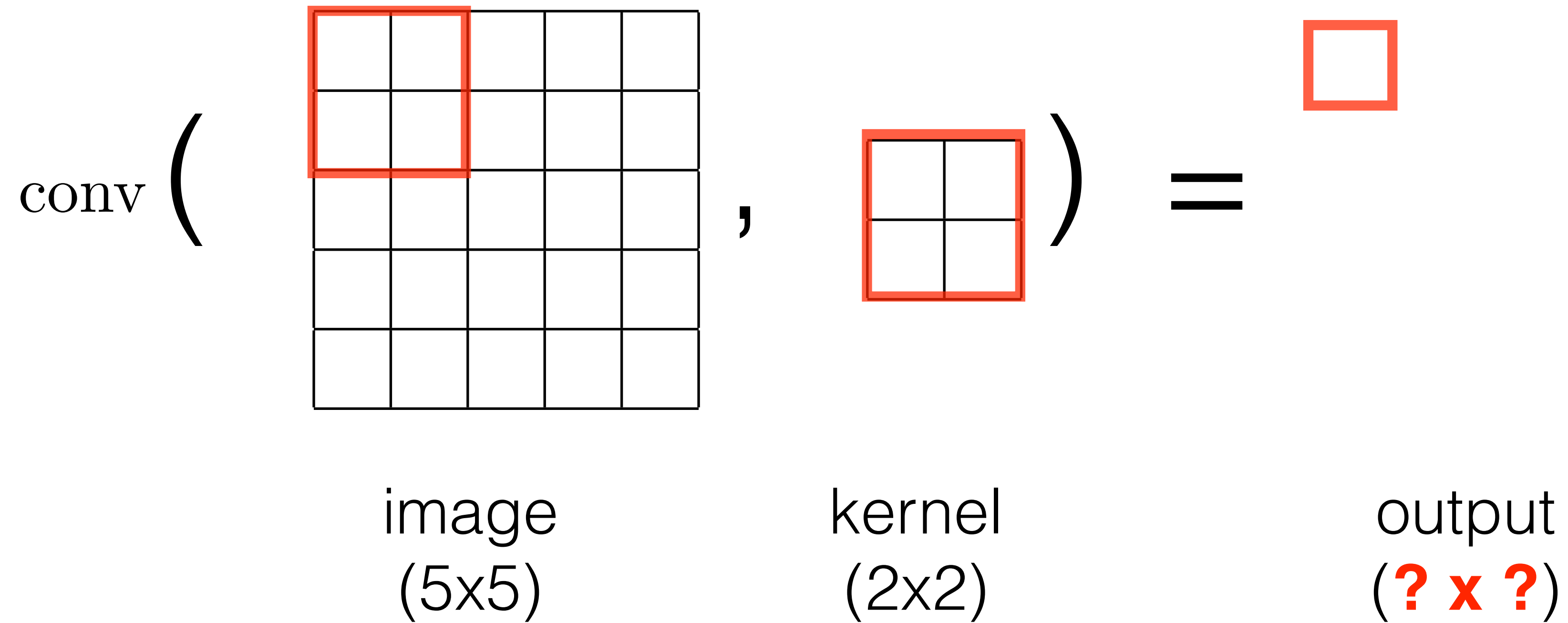
$$\text{conv} \left(\begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array} , \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline \end{array} \right) =$$

image
(5x5)

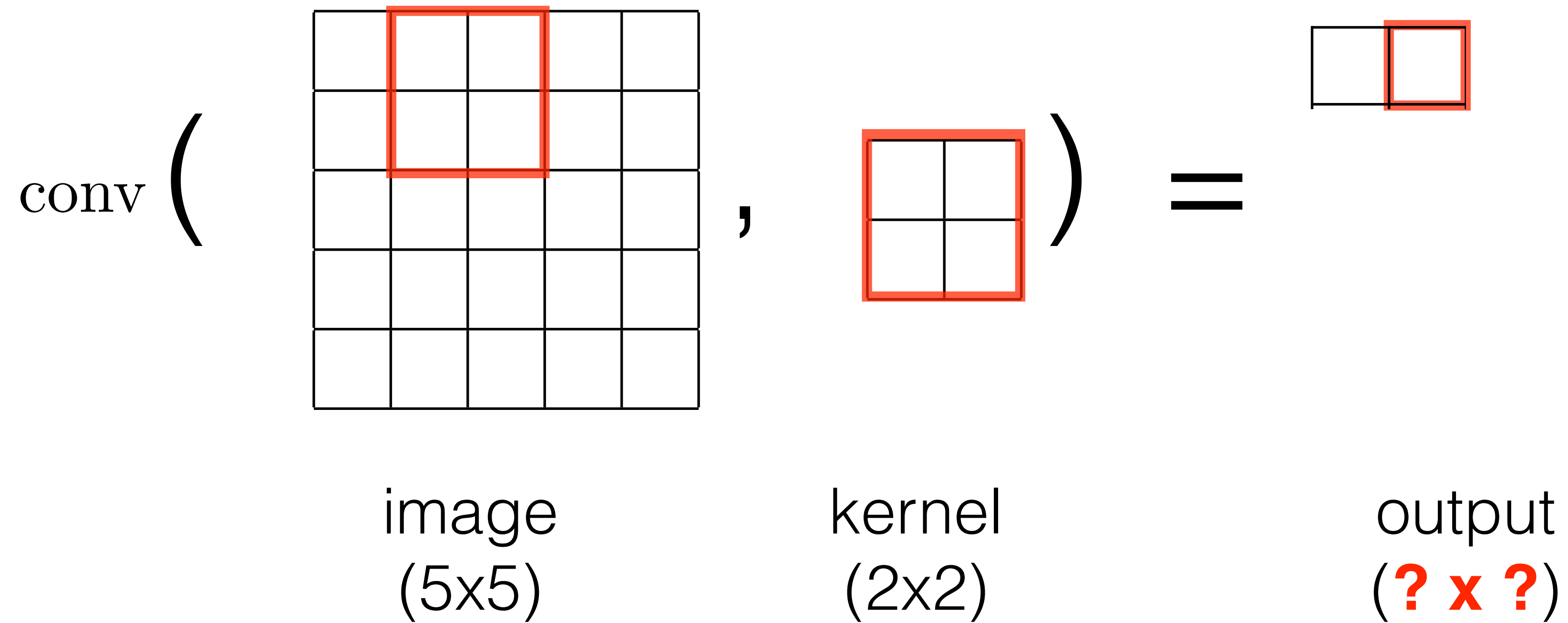
kernel
(2x2)

output
(**?** x **?**)

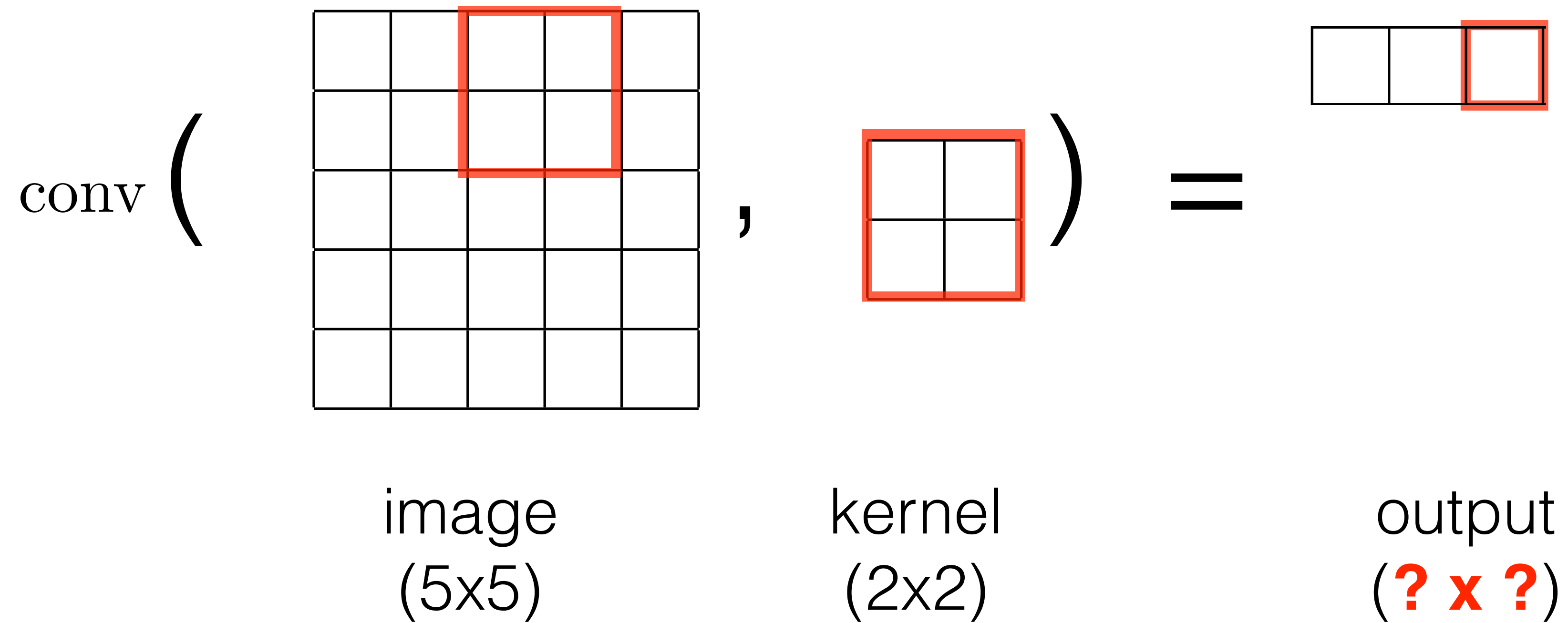
Convolution layer properties - output size



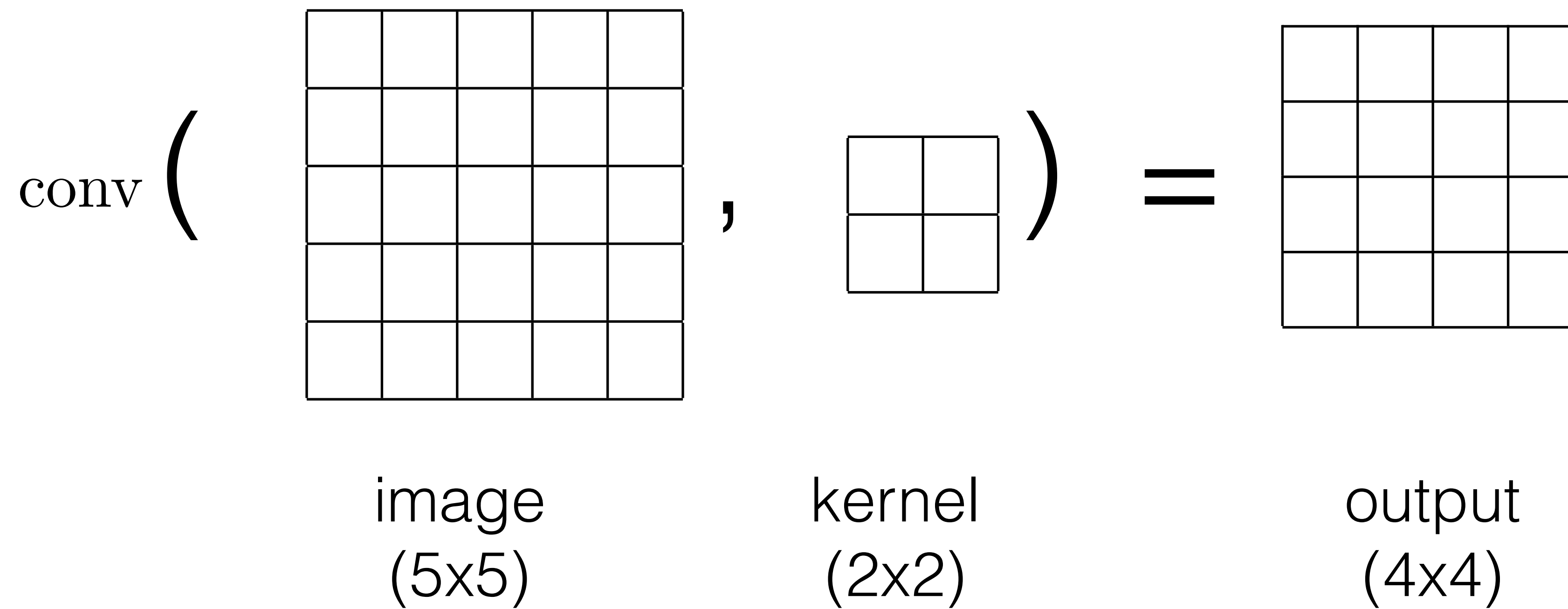
Convolution layer properties - output size



Convolution layer properties - output size

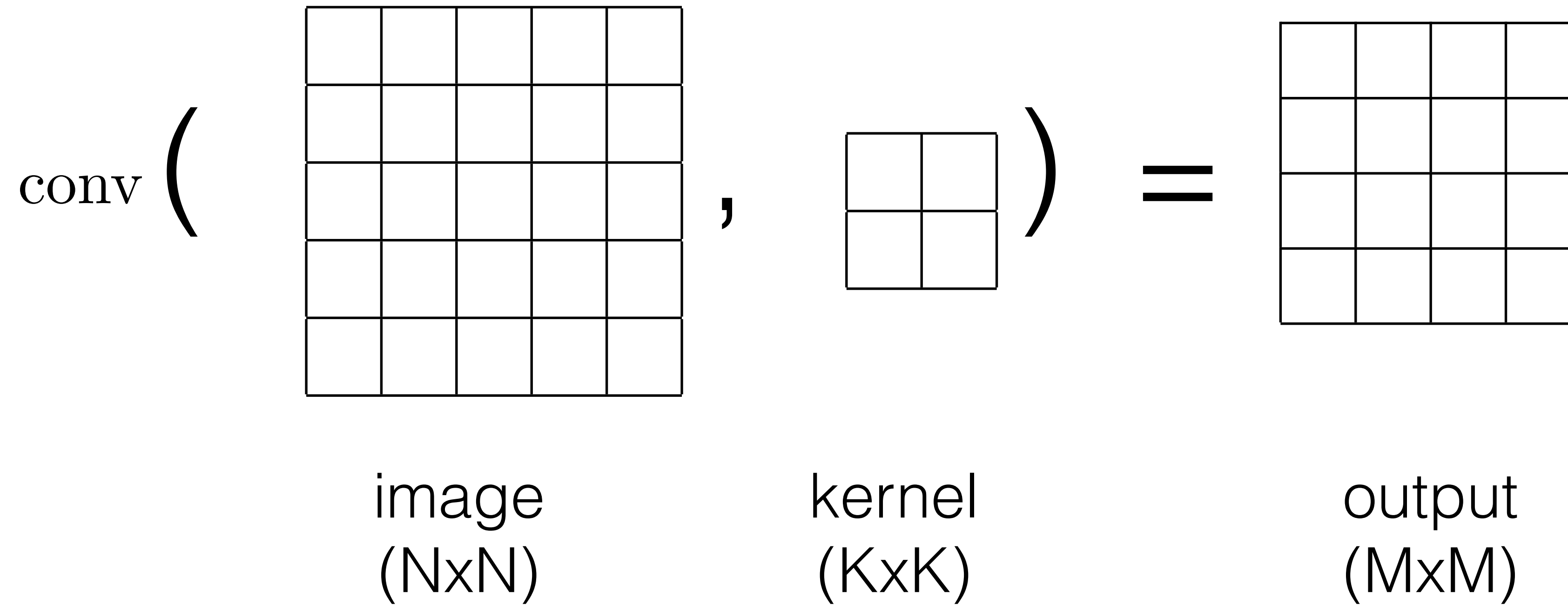


Convolution layer properties - output size



Convolution layer properties - output size

$$M = N - K + 1$$



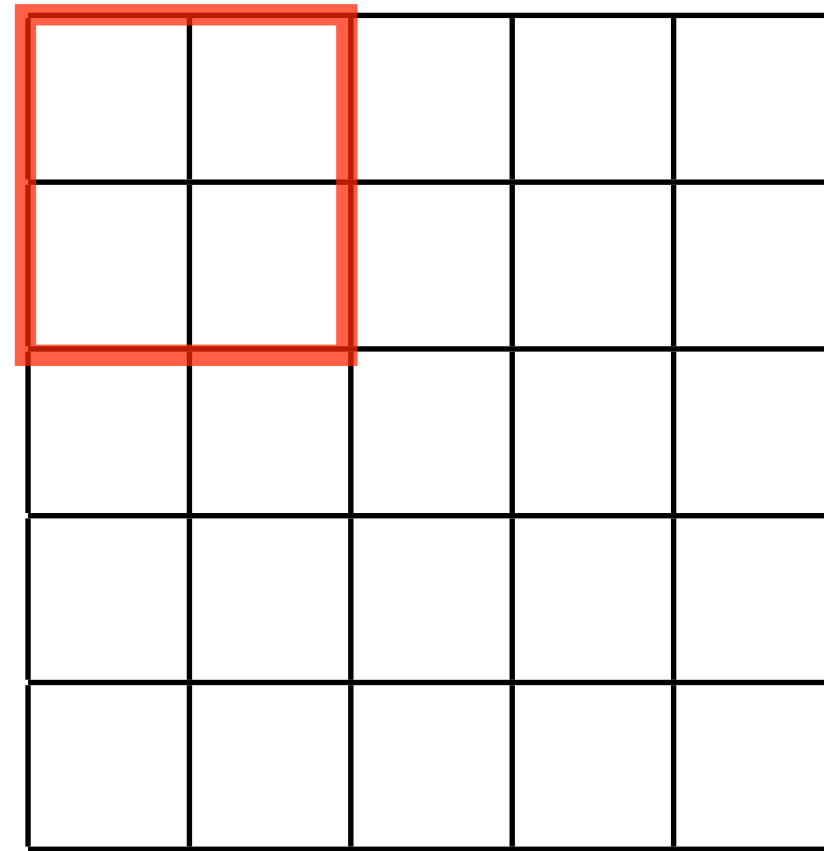
Convolution layer properties - stride

stride = 1

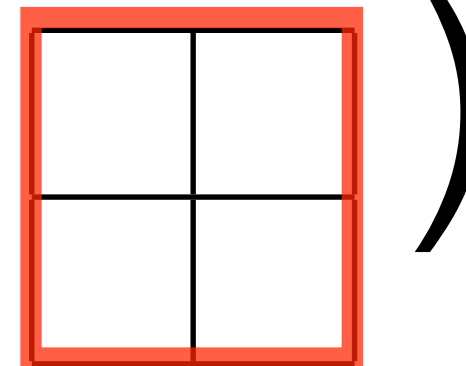
kernel moves by 1 pixel



conv (



,



)

=

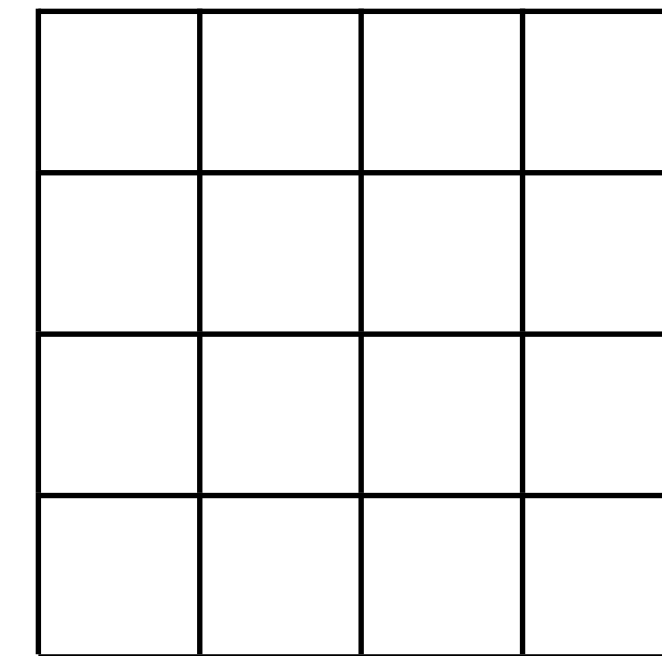


image
(5x5)

kernel
(2x2)

output
(4x4)

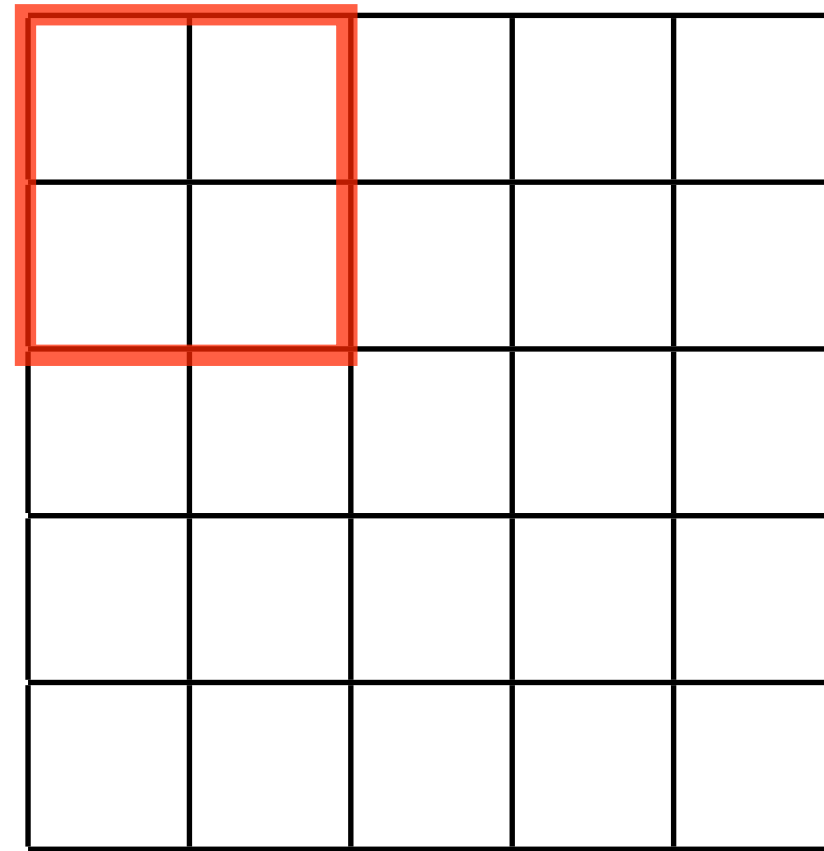
Convolution layer properties - stride

stride = 3

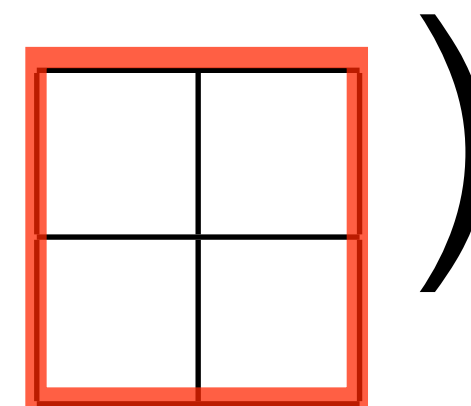
kernel moves by 3 pixels



conv (



,



)

=



image
(5x5)

kernel
(2x2)

output
(? x ?)

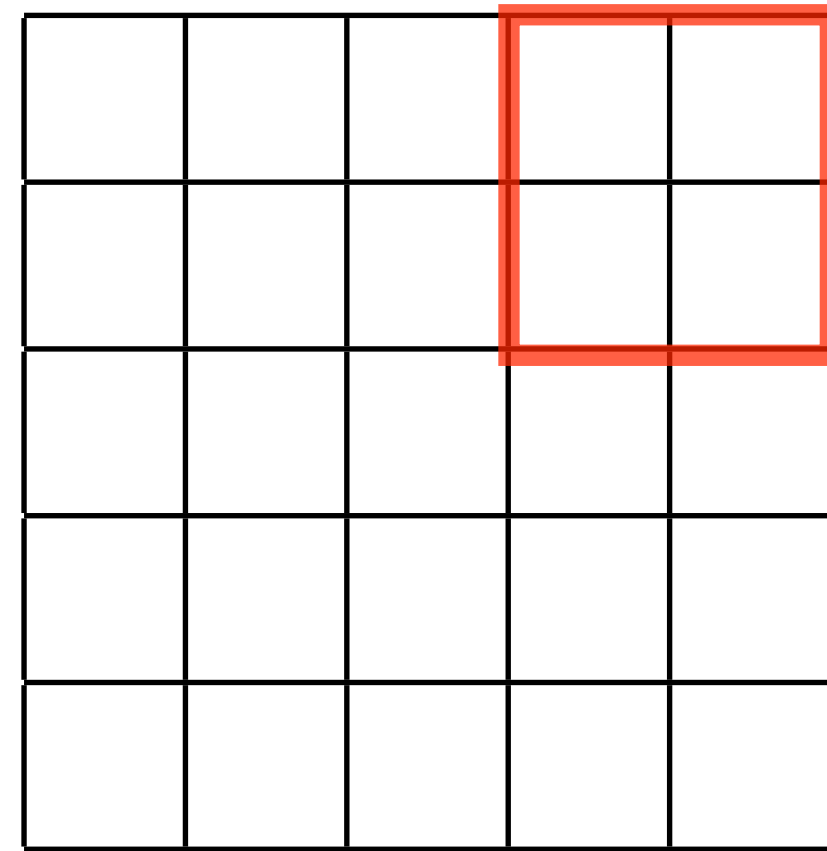
Convolution layer properties - stride

stride = 3

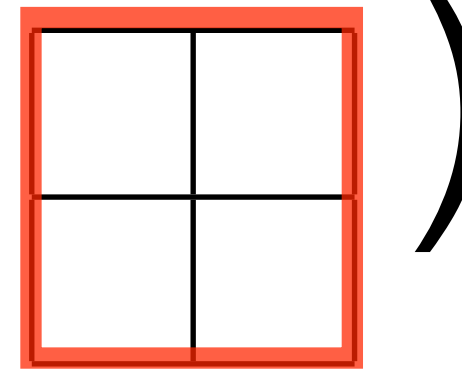
kernel moves by 3 pixels



conv (



,



)

=

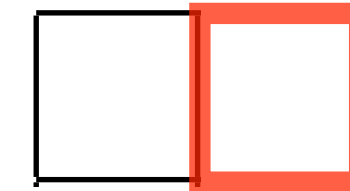


image
(5x5)

kernel
(2x2)

output
(**?** **x** **?**)

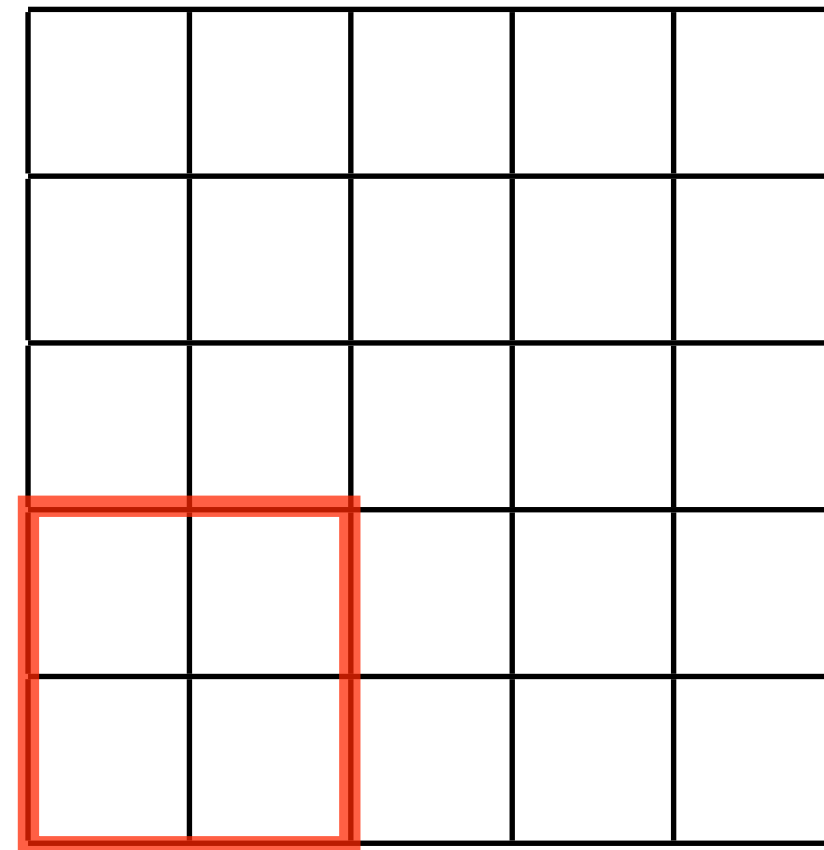
Convolution layer properties - stride

stride = 3

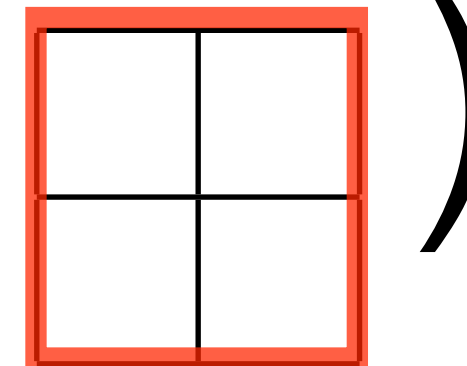
kernel moves by 3 pixels



conv (



,



)

=

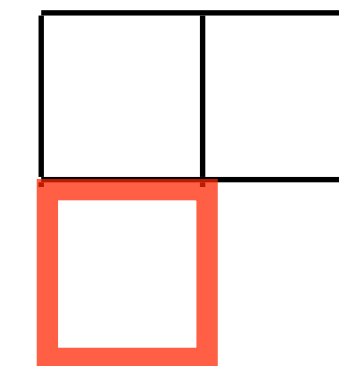


image
(5x5)

kernel
(2x2)

output
(? x ?)

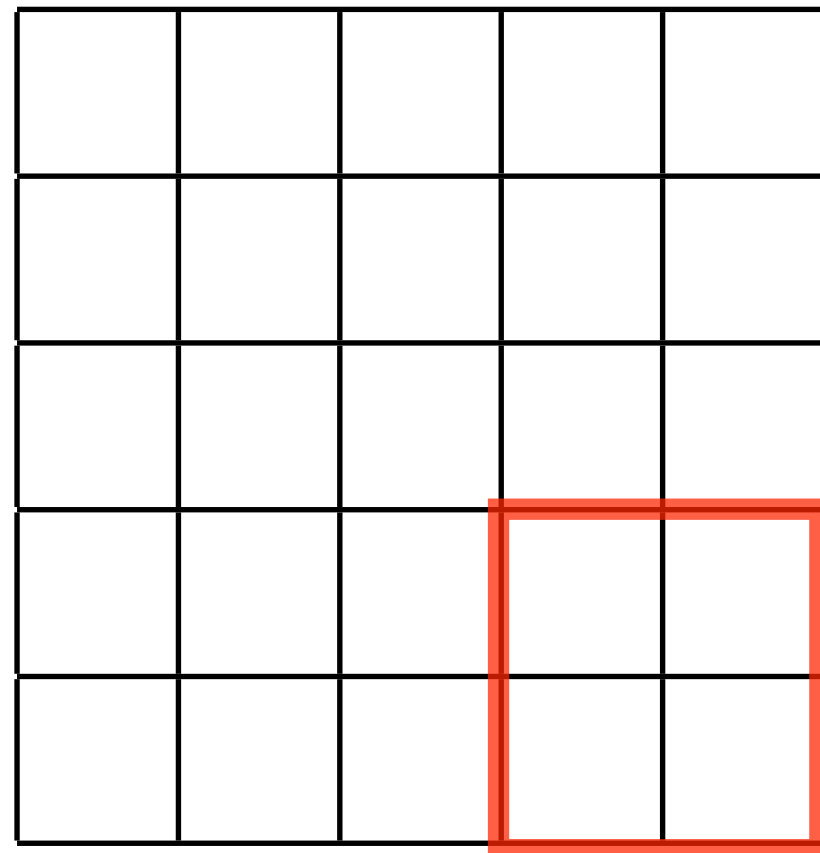
Convolution layer properties - stride

stride = 3

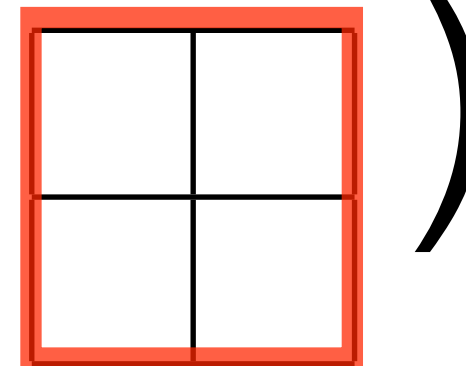
kernel moves by 3 pixels



conv (



,



)

=

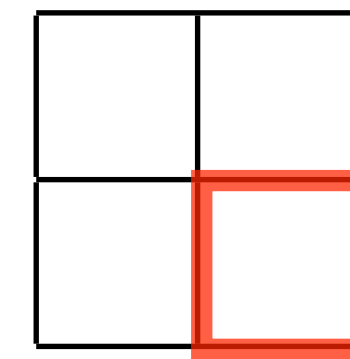


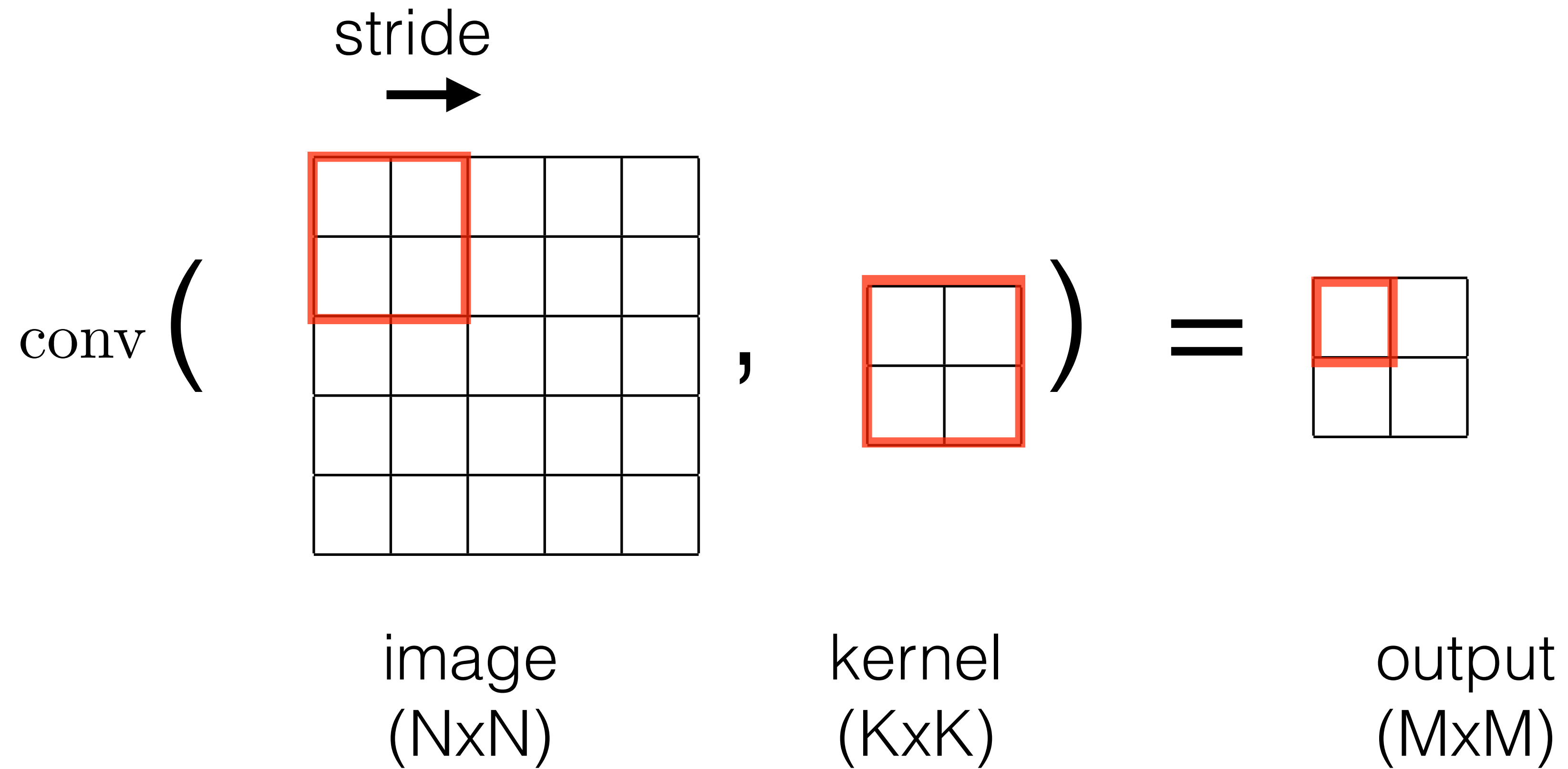
image
(5x5)

kernel
(2x2)

output
(2x2)

Convolution layer properties - stride

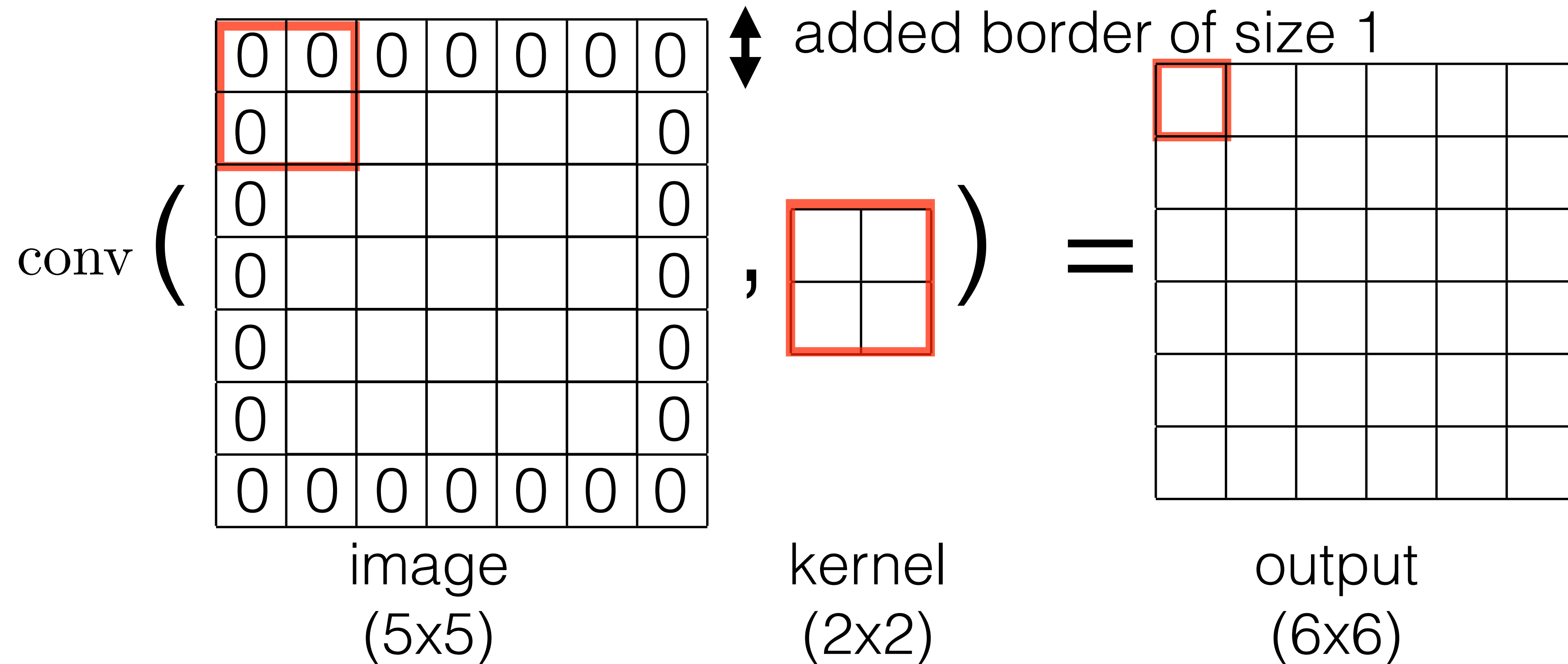
$$M = \text{floor}((N-K) / \text{stride} + 1)$$



e.g. $M = (5-2) / 3 + 1 = 2$

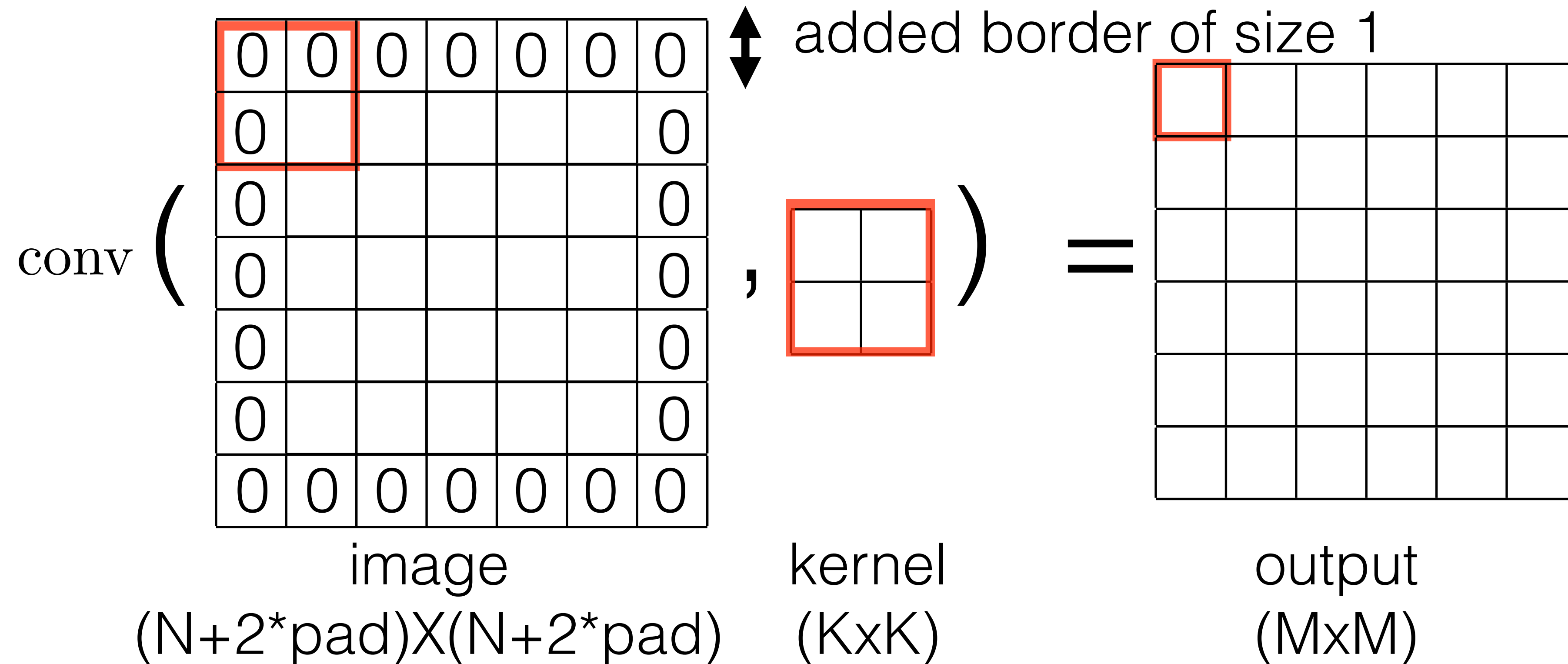
Convolution layer properties - pad

pad = 1



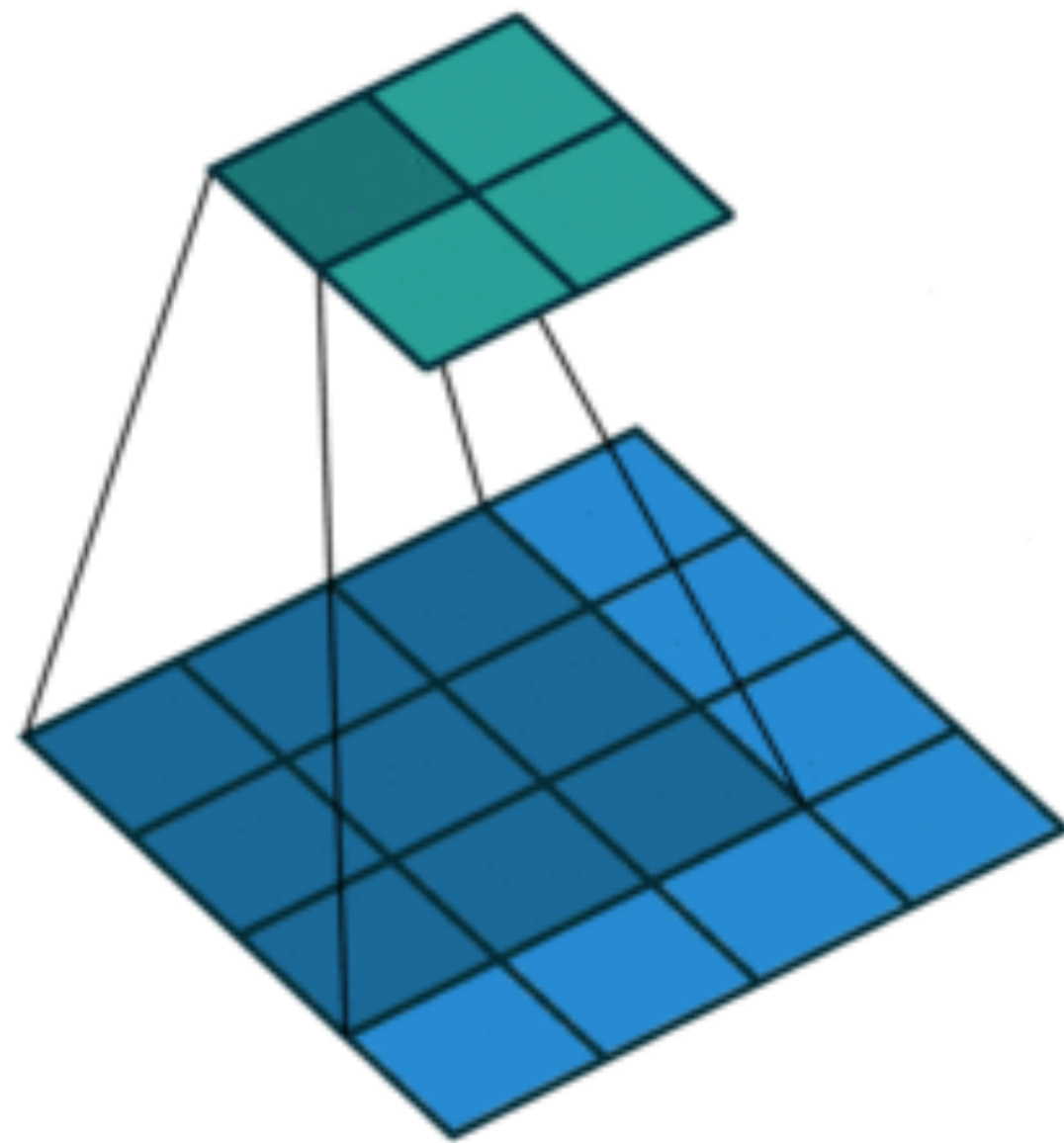
Convolution layer properties - pad

$$M = \text{floor}((N+2*\text{pad}-K) / \text{stride} + 1)$$

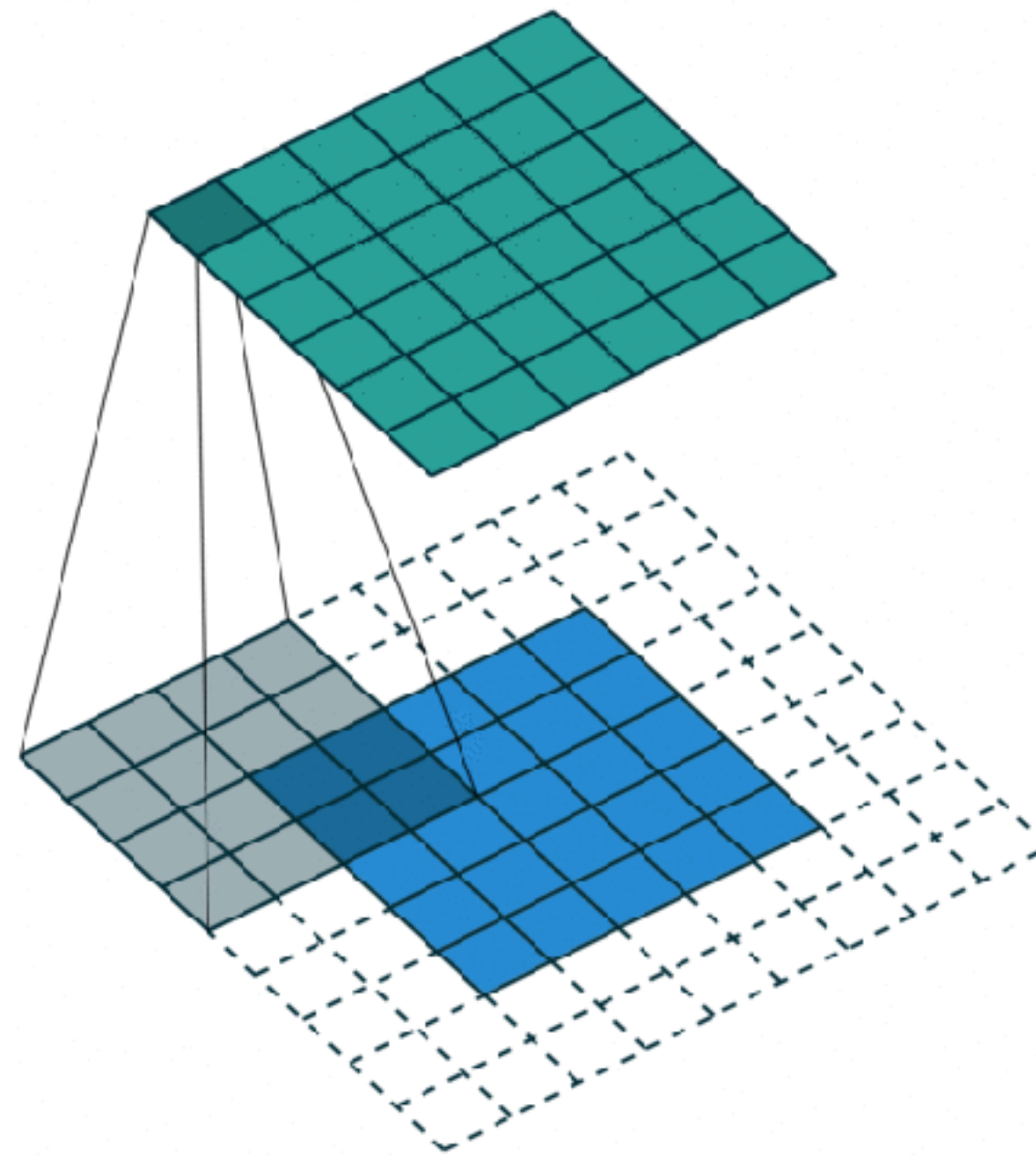


Convolution + padding + stride

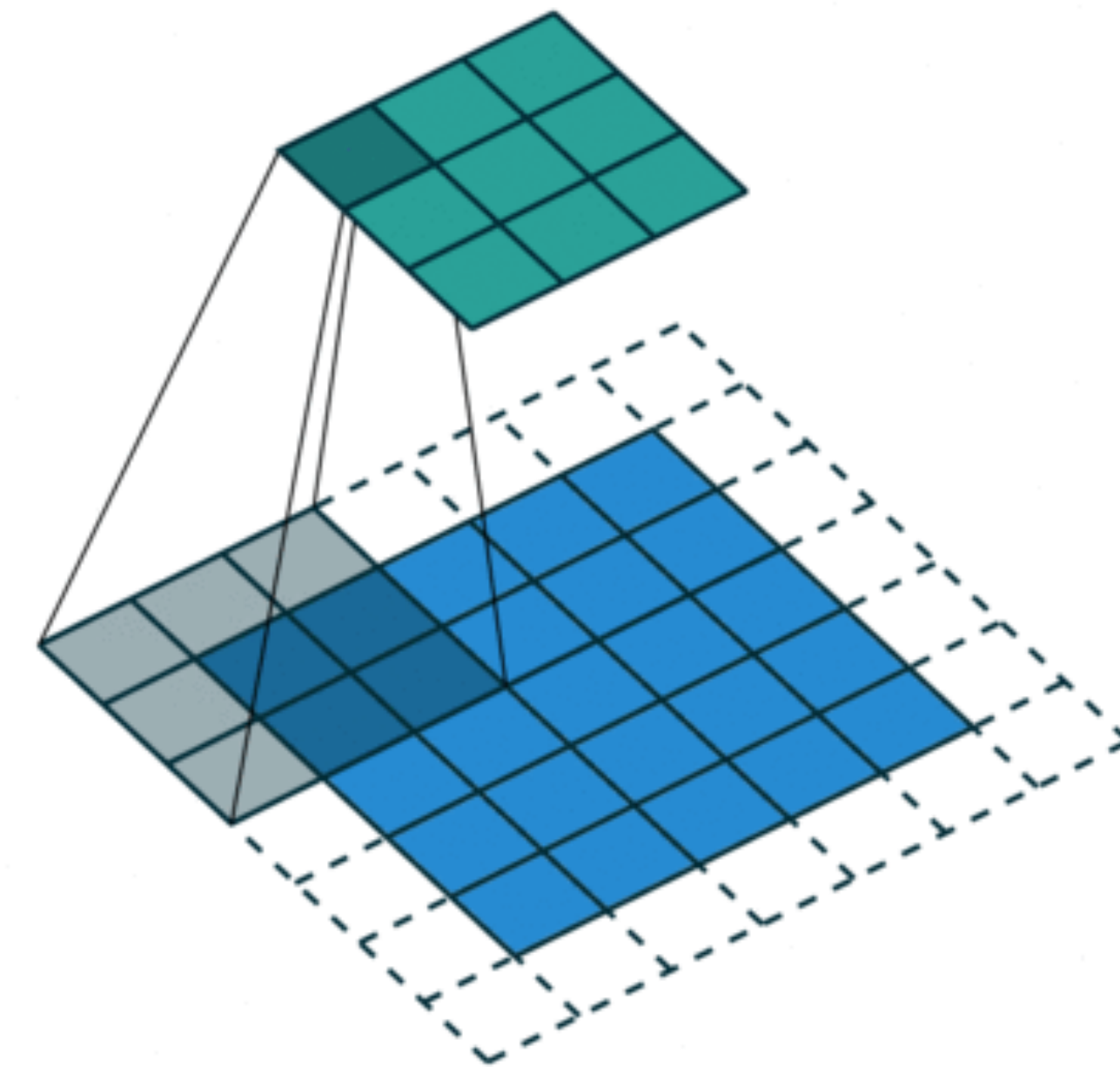
convolution

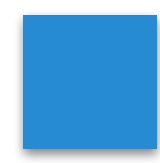


padding



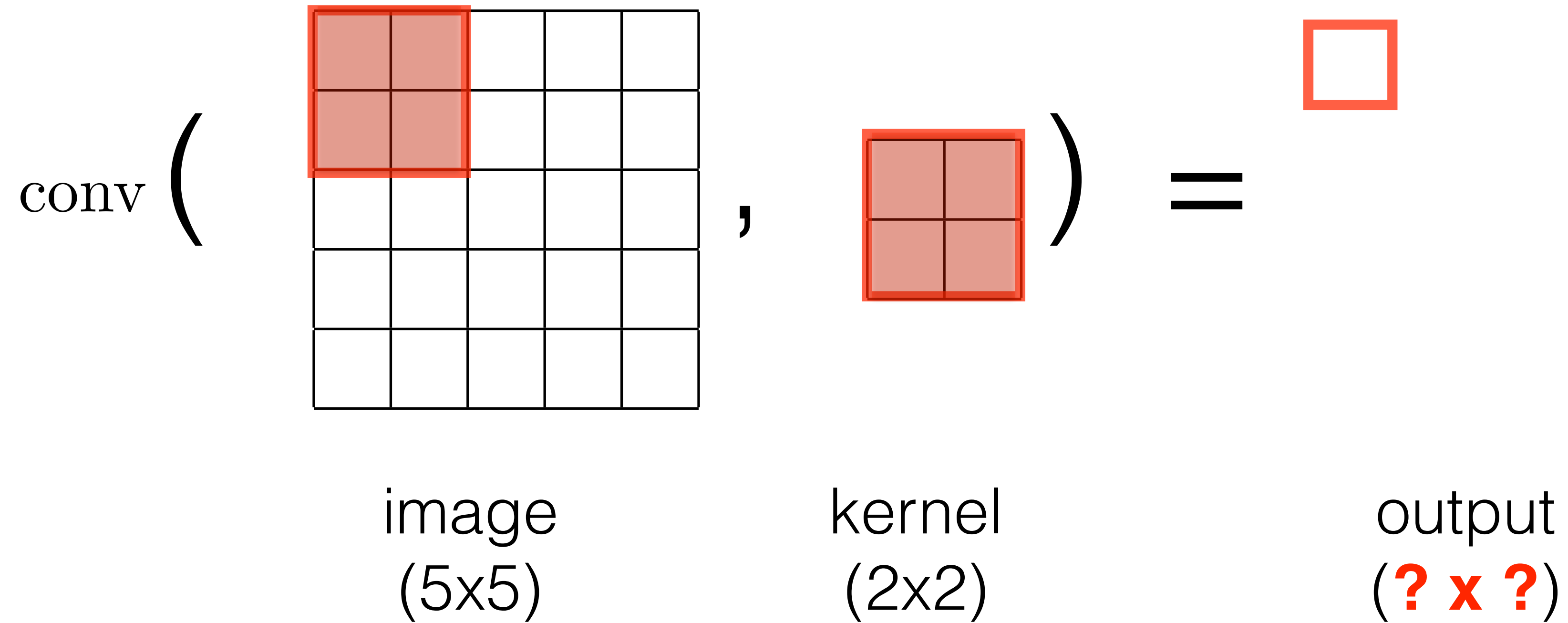
padding+stride



 input
 output

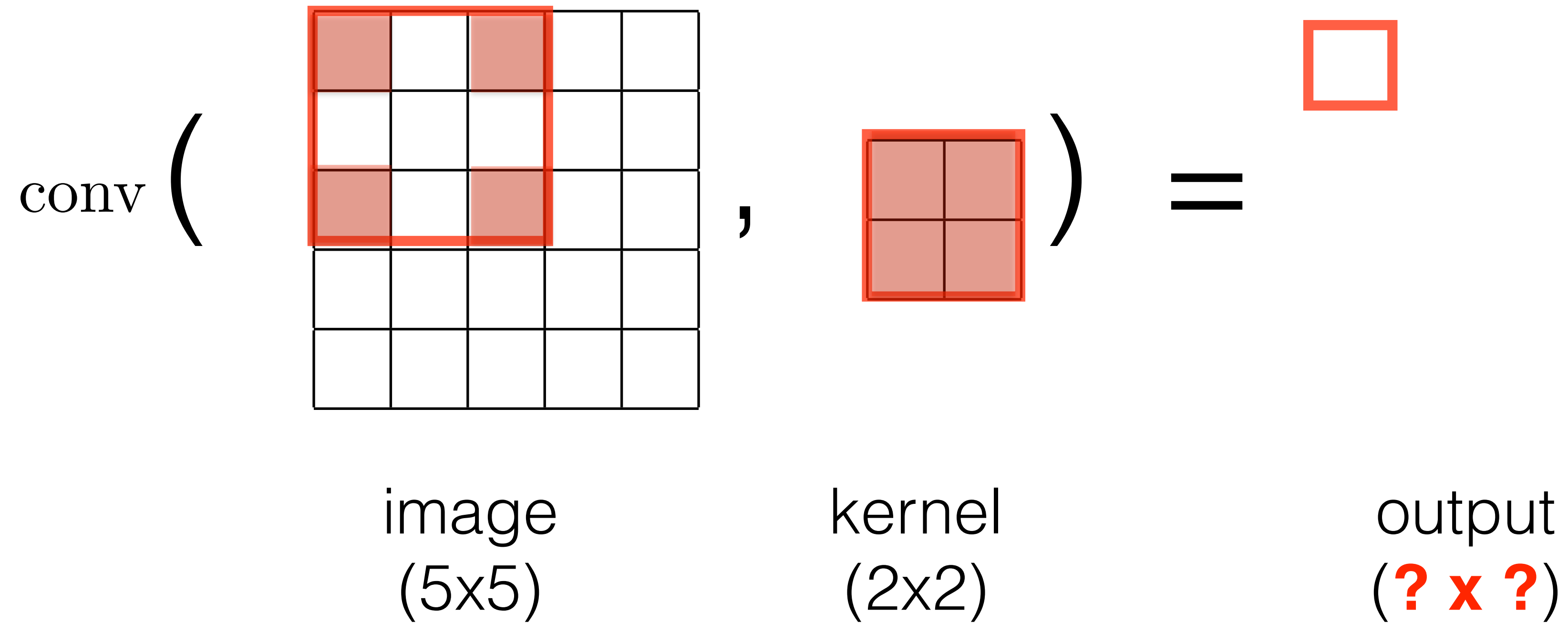
Convolution layer

Dilatation rate = 1



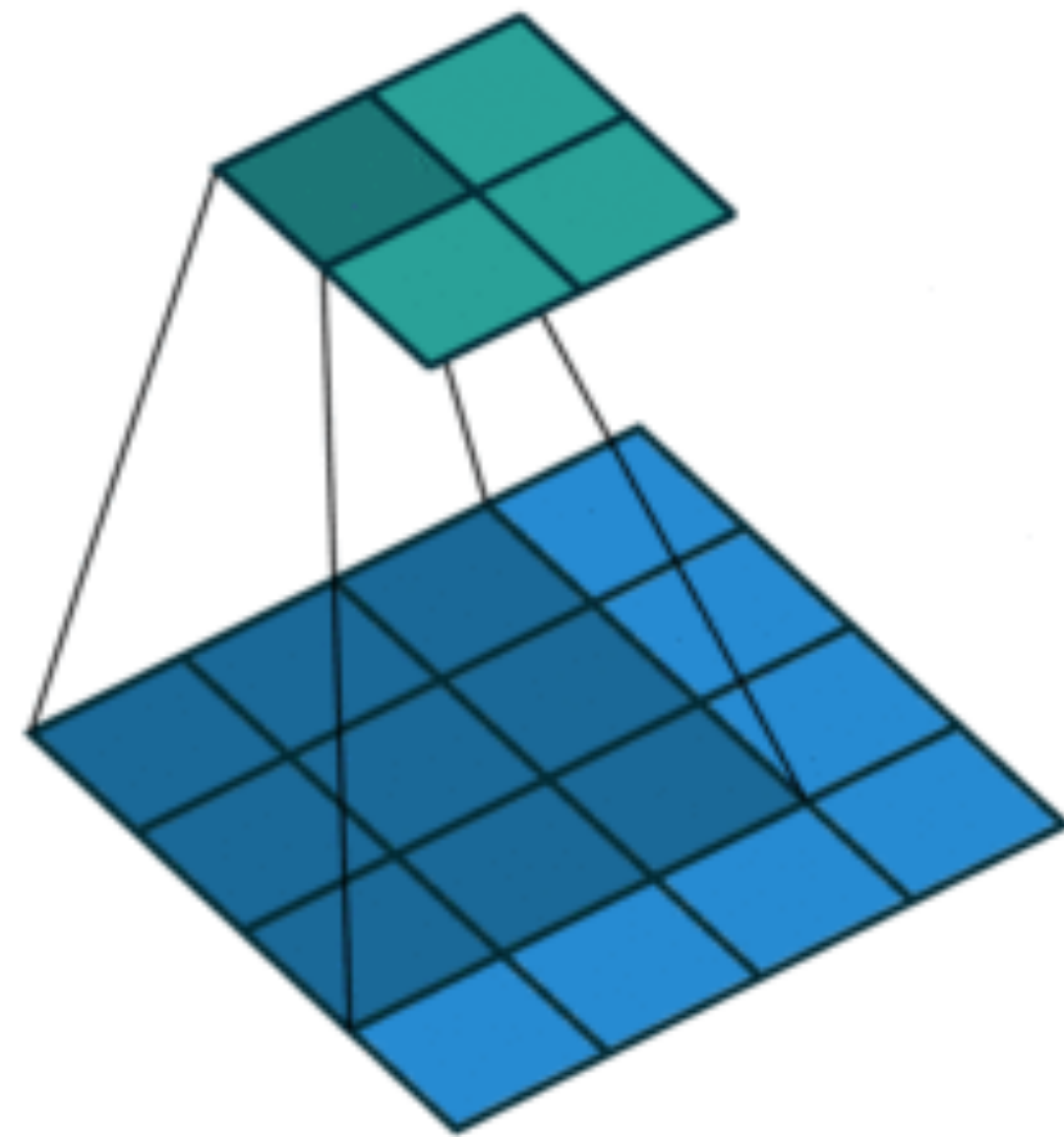
Dilated convolution layer

Dilatation rate = 2

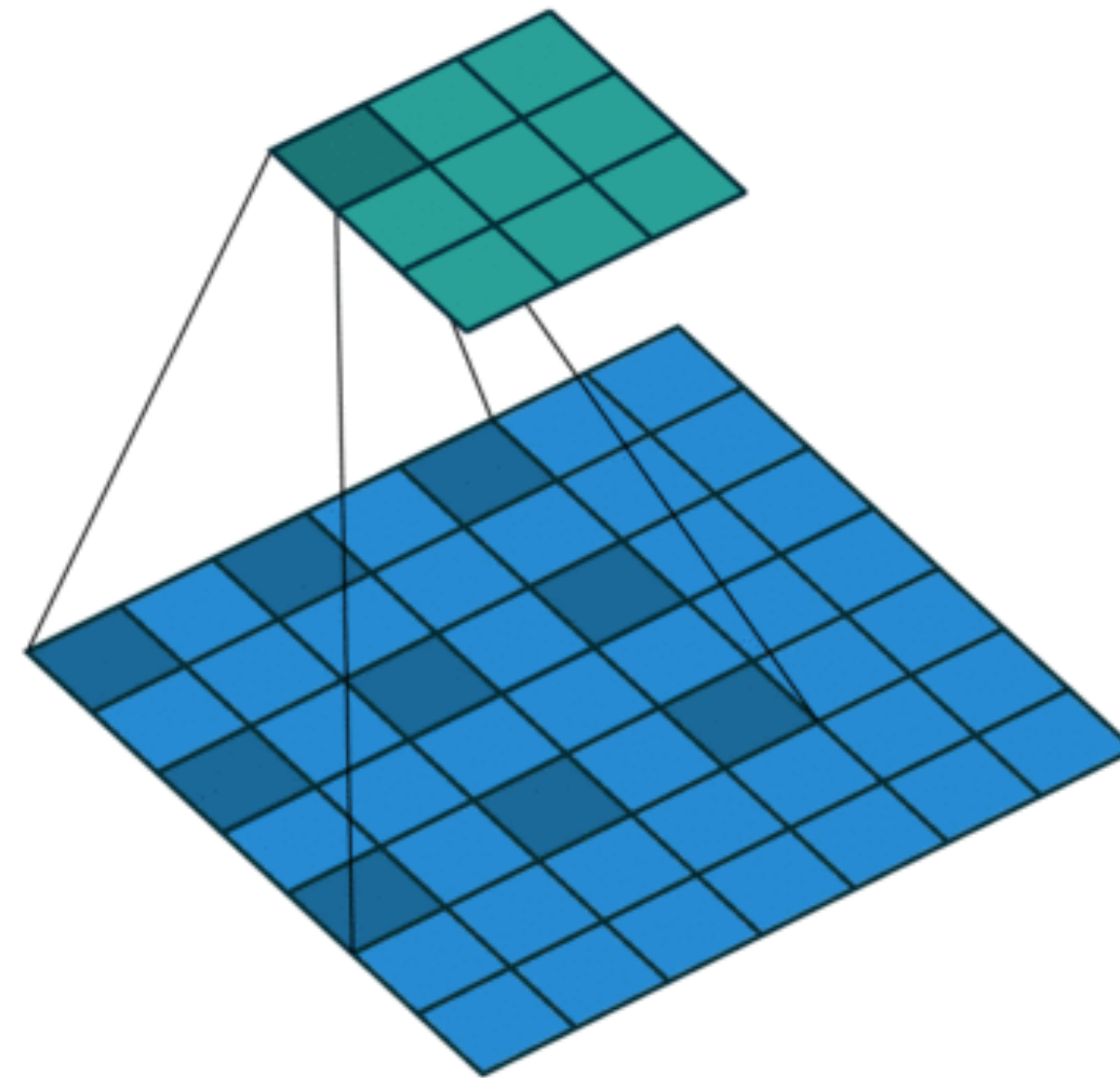


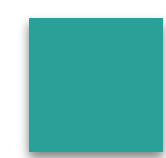
Dilated convolution layer

dilatation rate = 1



dilatation rate = 2

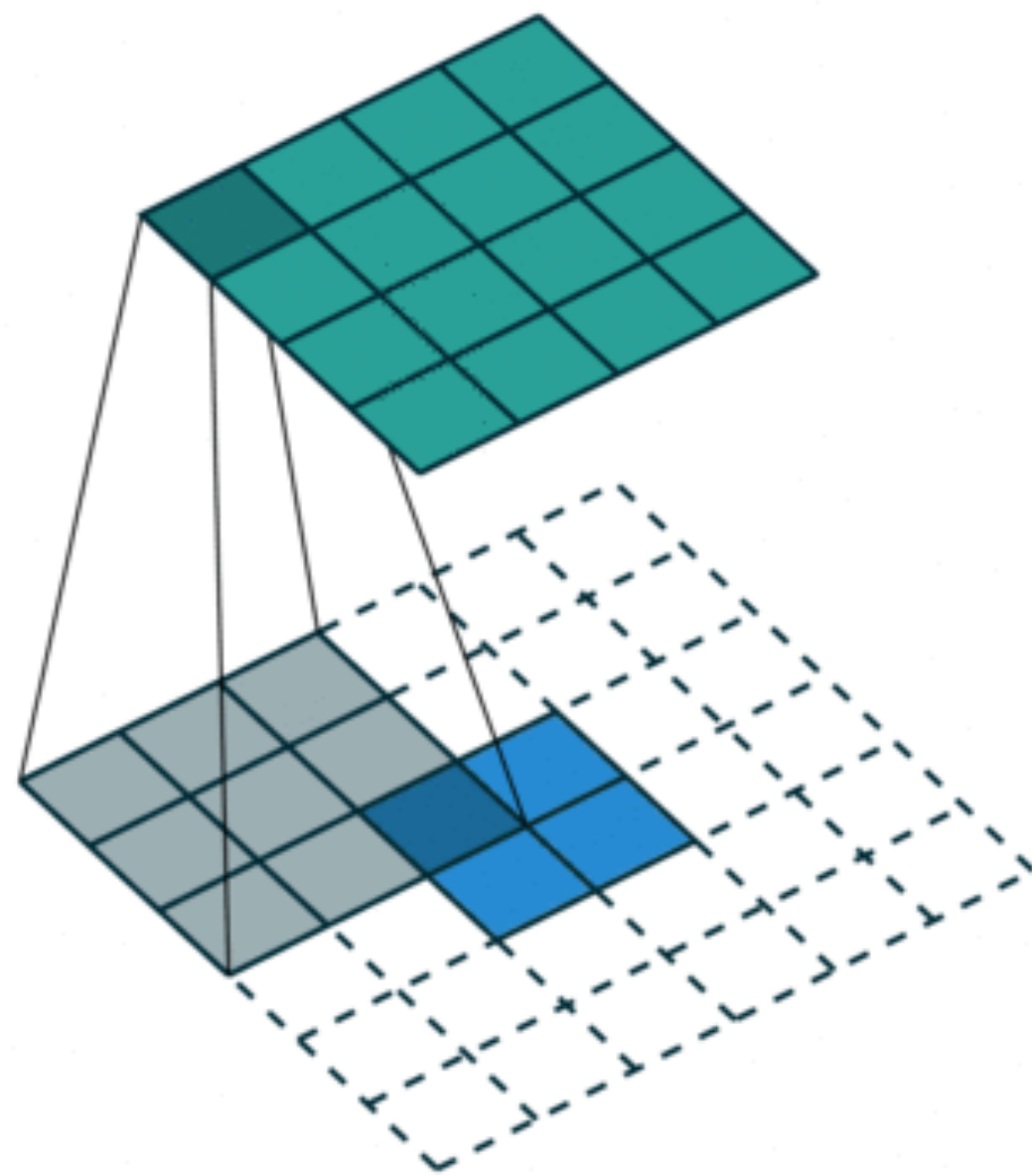


 input
 output

Transposed convolution

Upsampling / smart interpolation

no padding
no stride

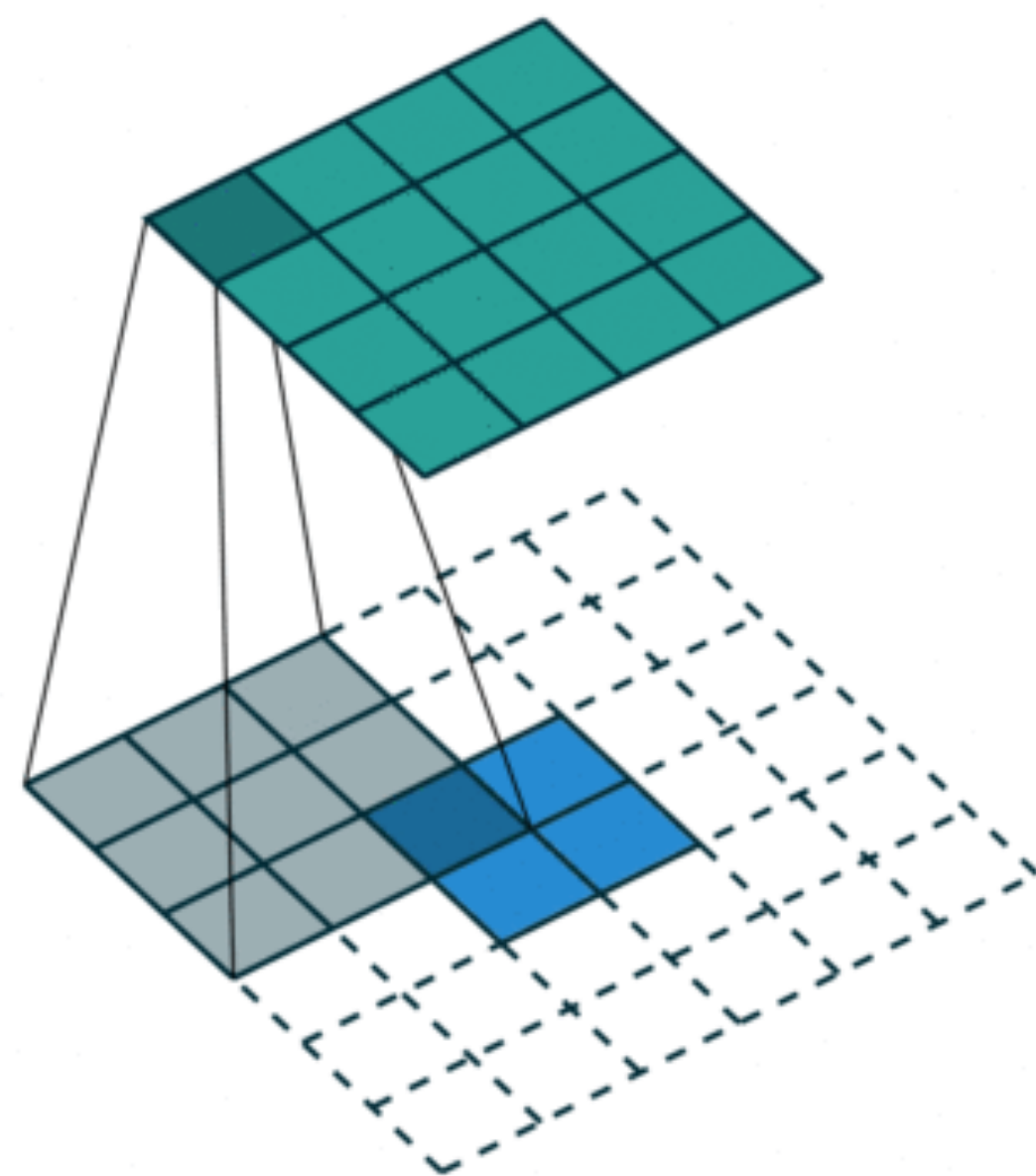


input
output

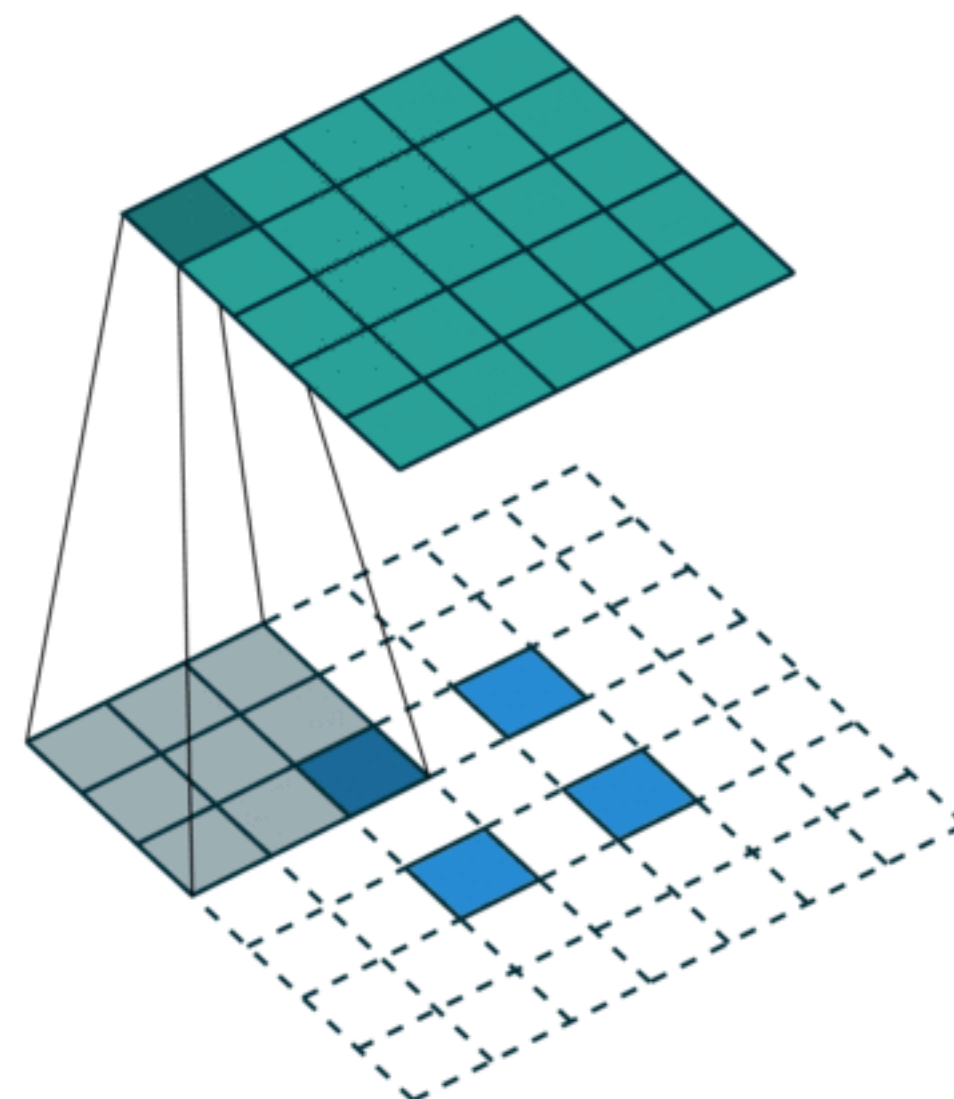
Transposed convolution

Upsampling / smart interpolation

no padding
no stride



no padding
stride

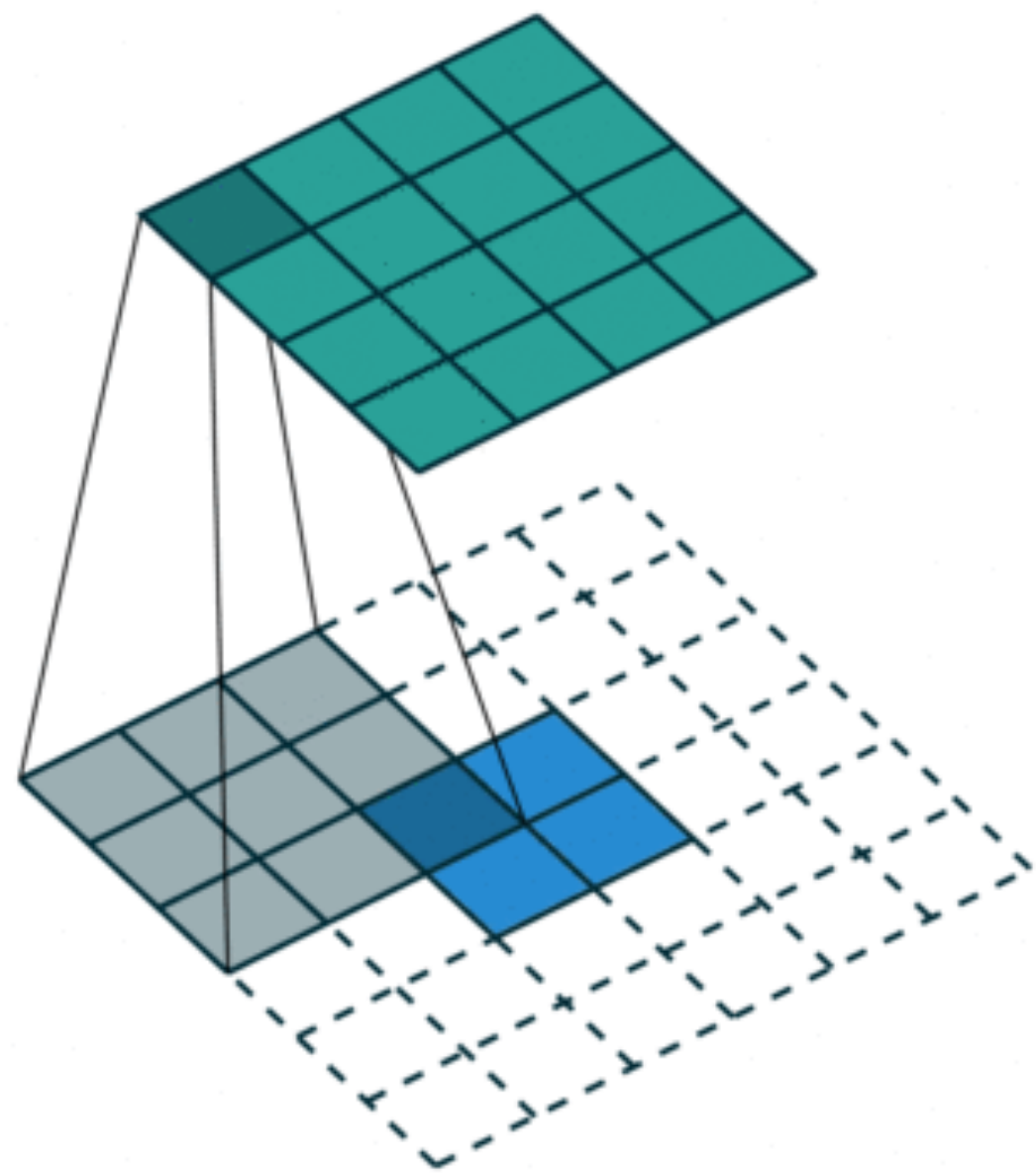


input
output

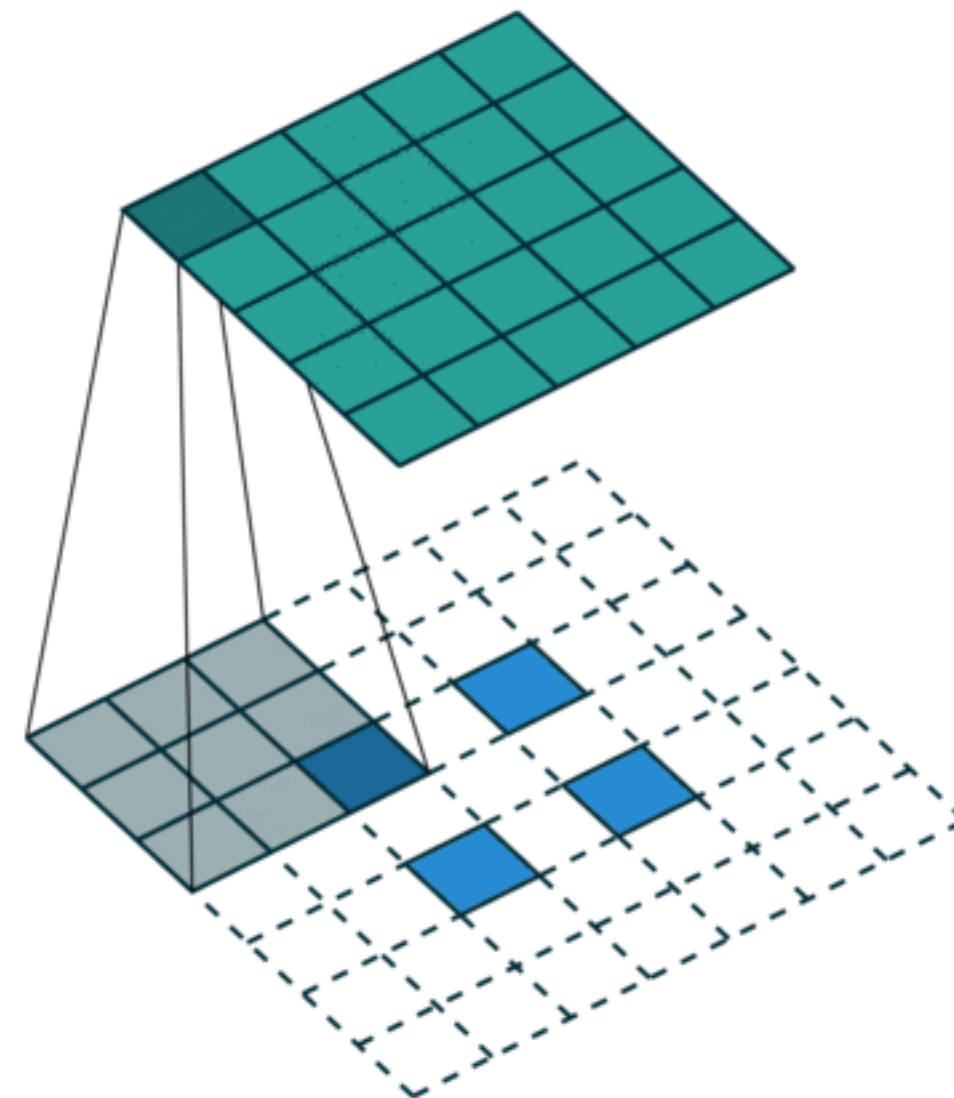
Transposed convolution

Upsampling / smart interpolation

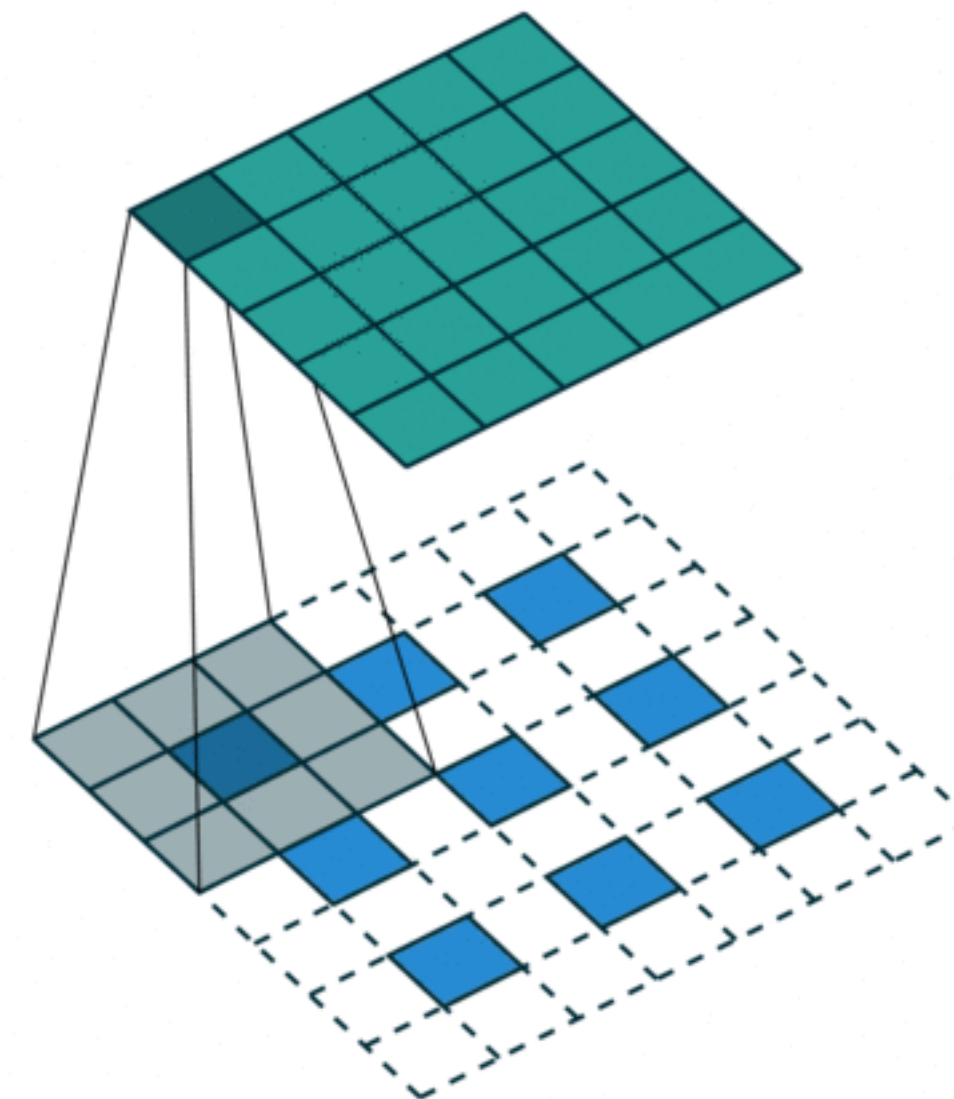
no padding
no stride



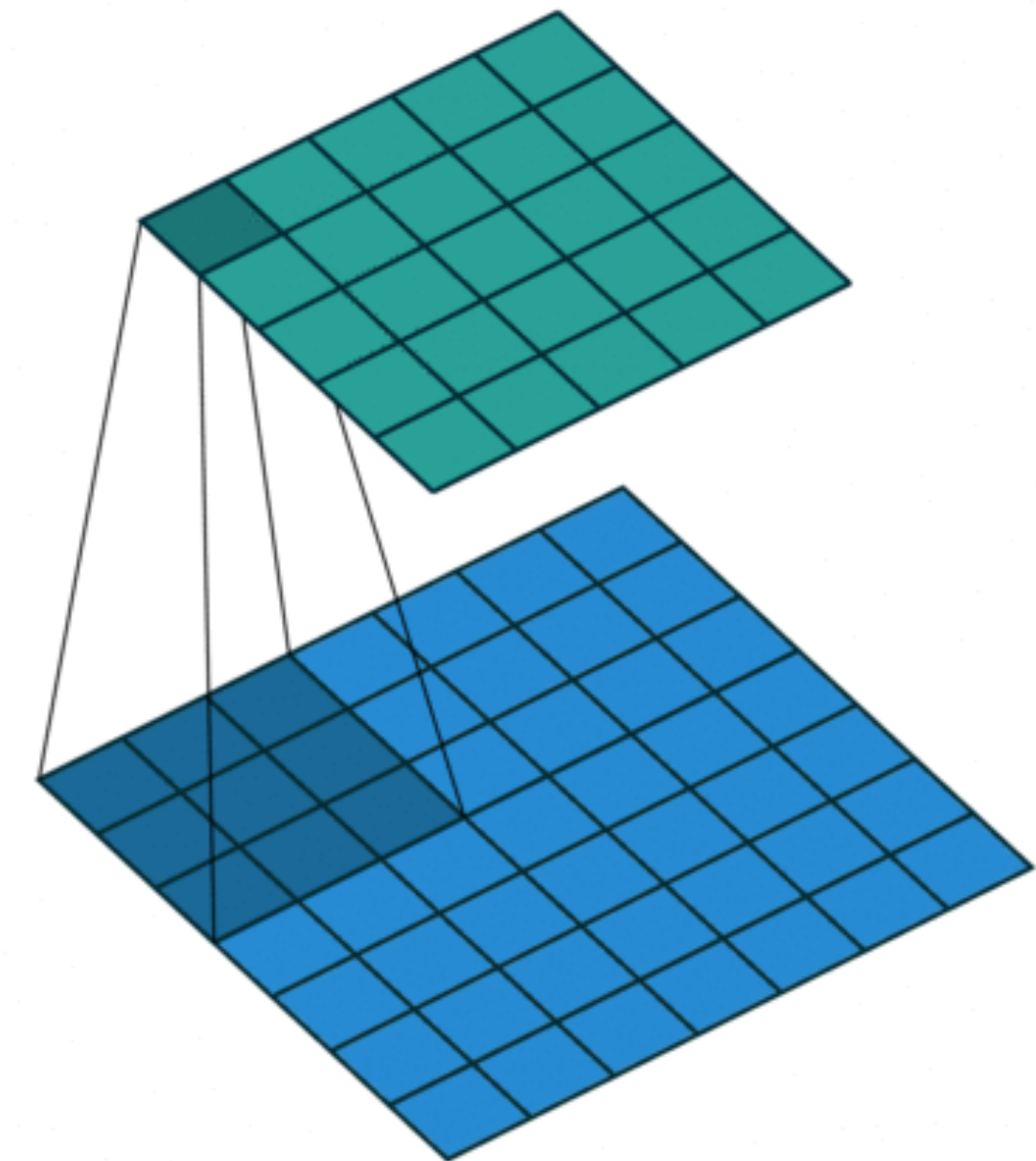
no padding
stride



padding
stride

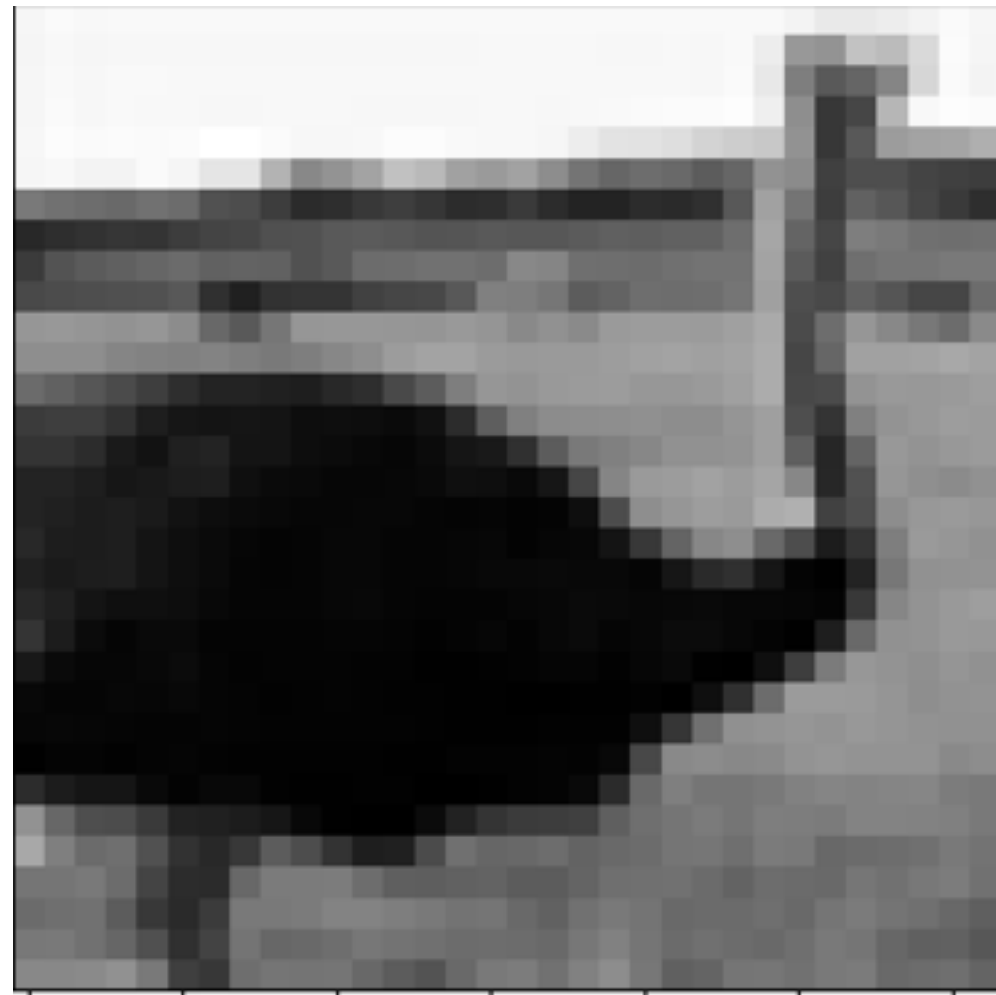


full padding
no stride

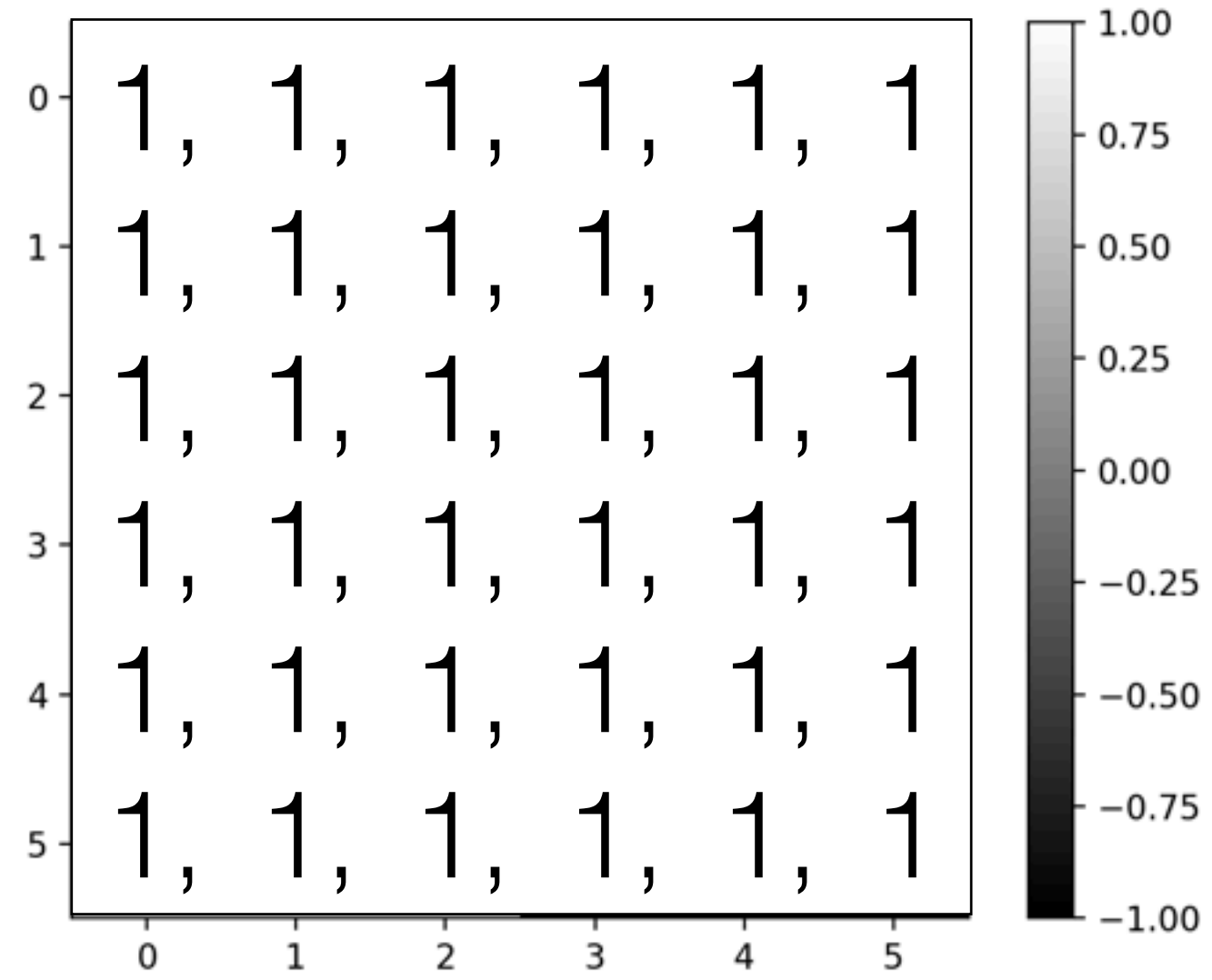


input
output

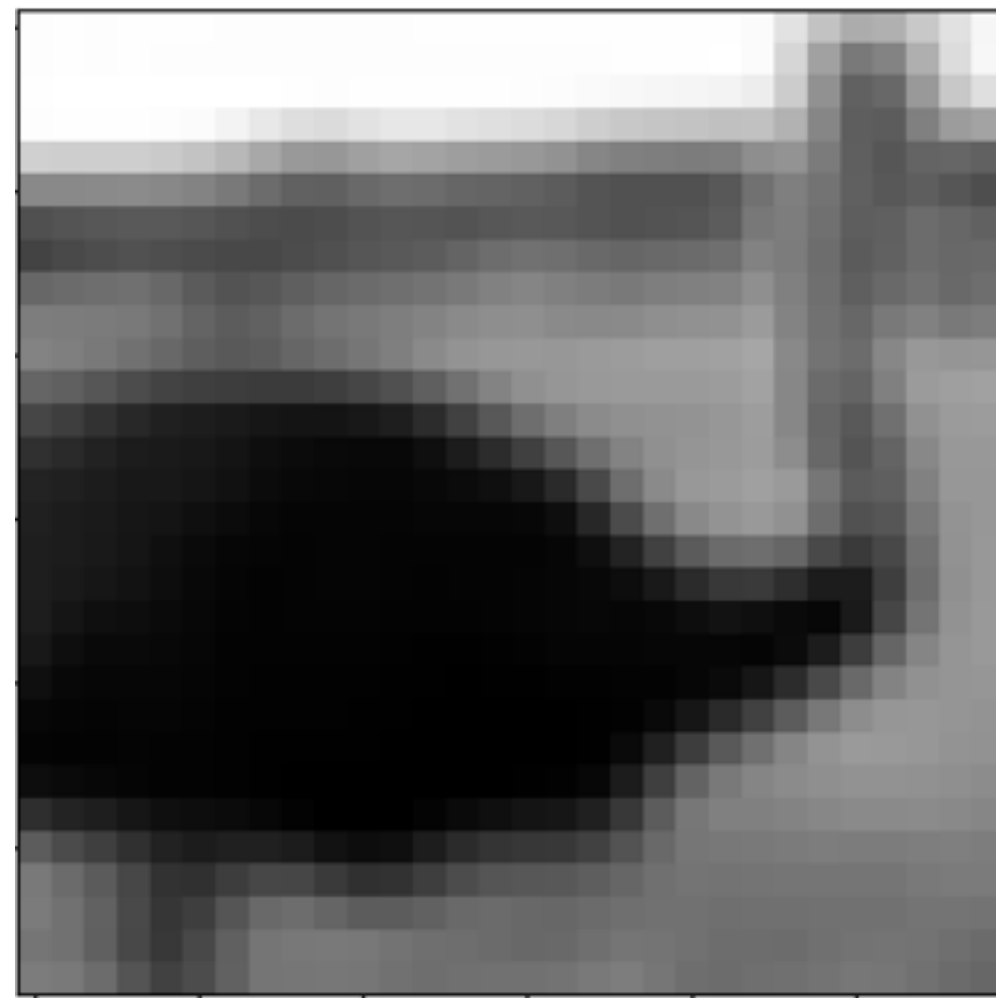
Input image



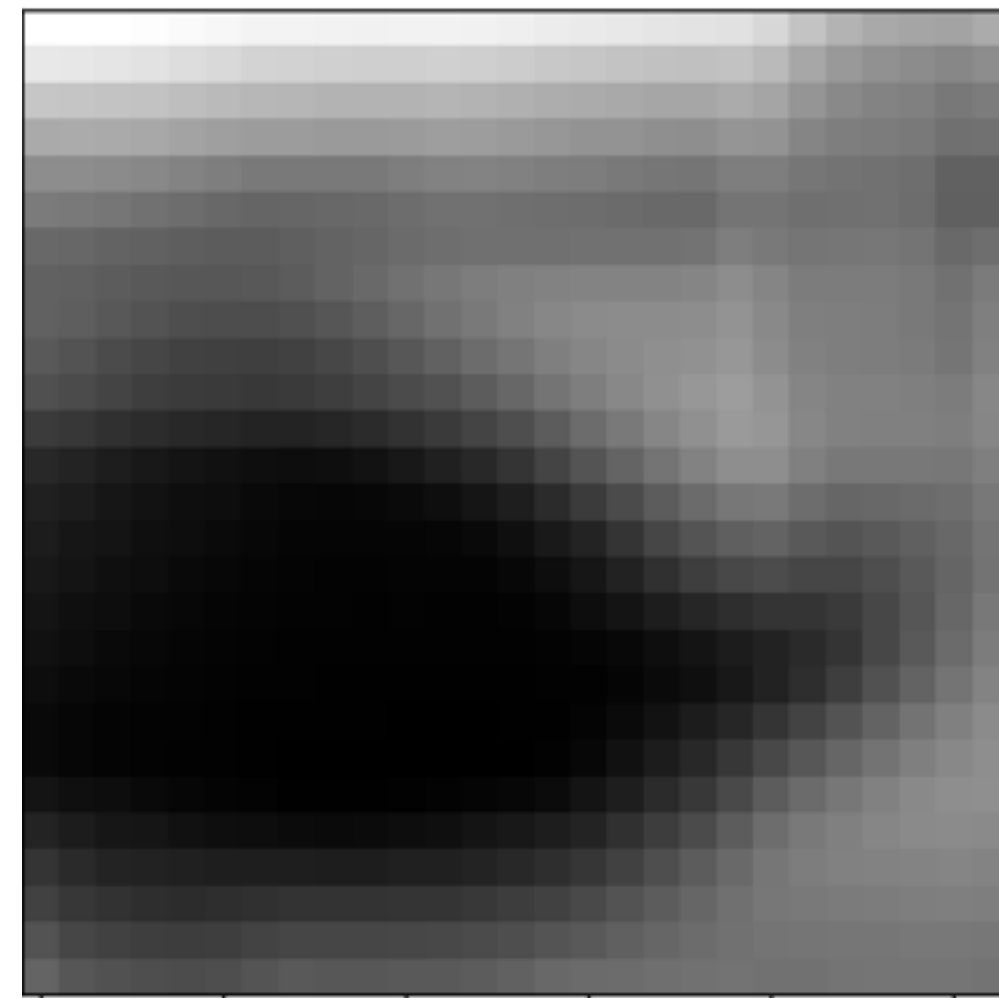
Input kernel



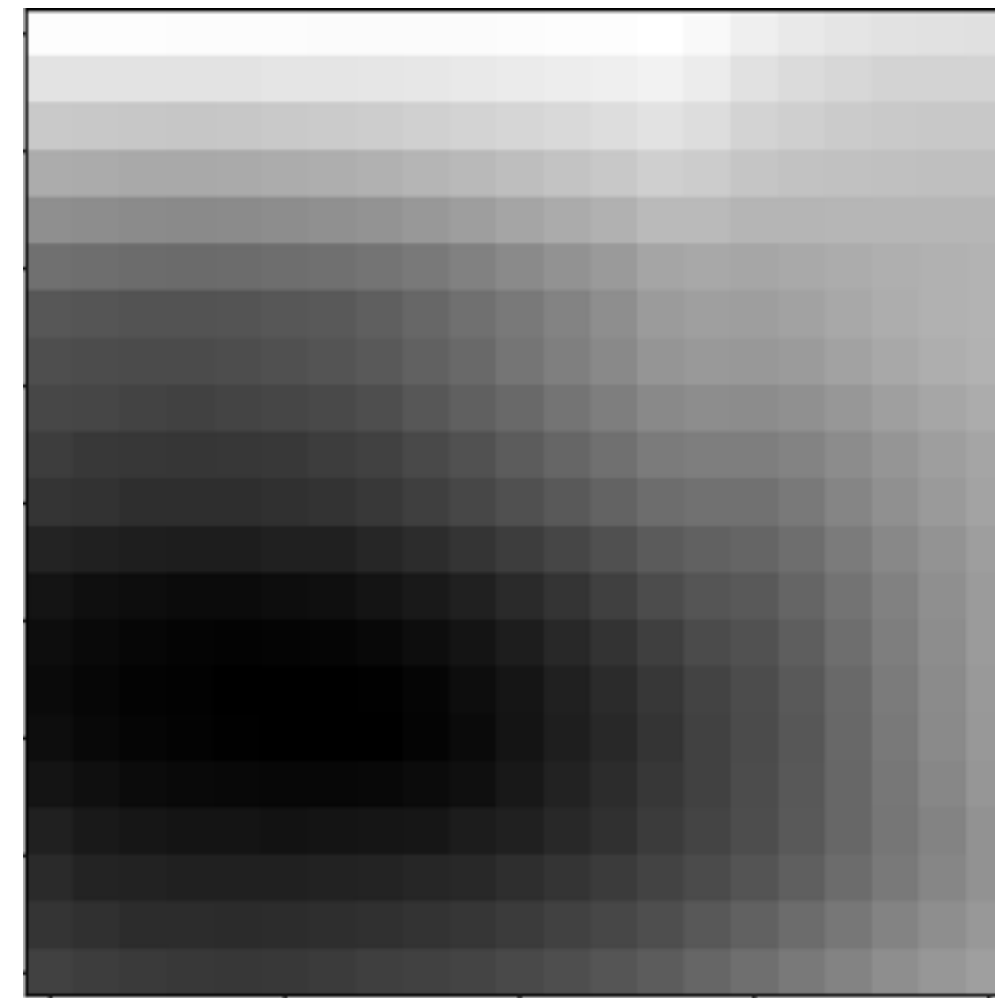
Output



```
nn.Conv2d(1, 1, 3)
```

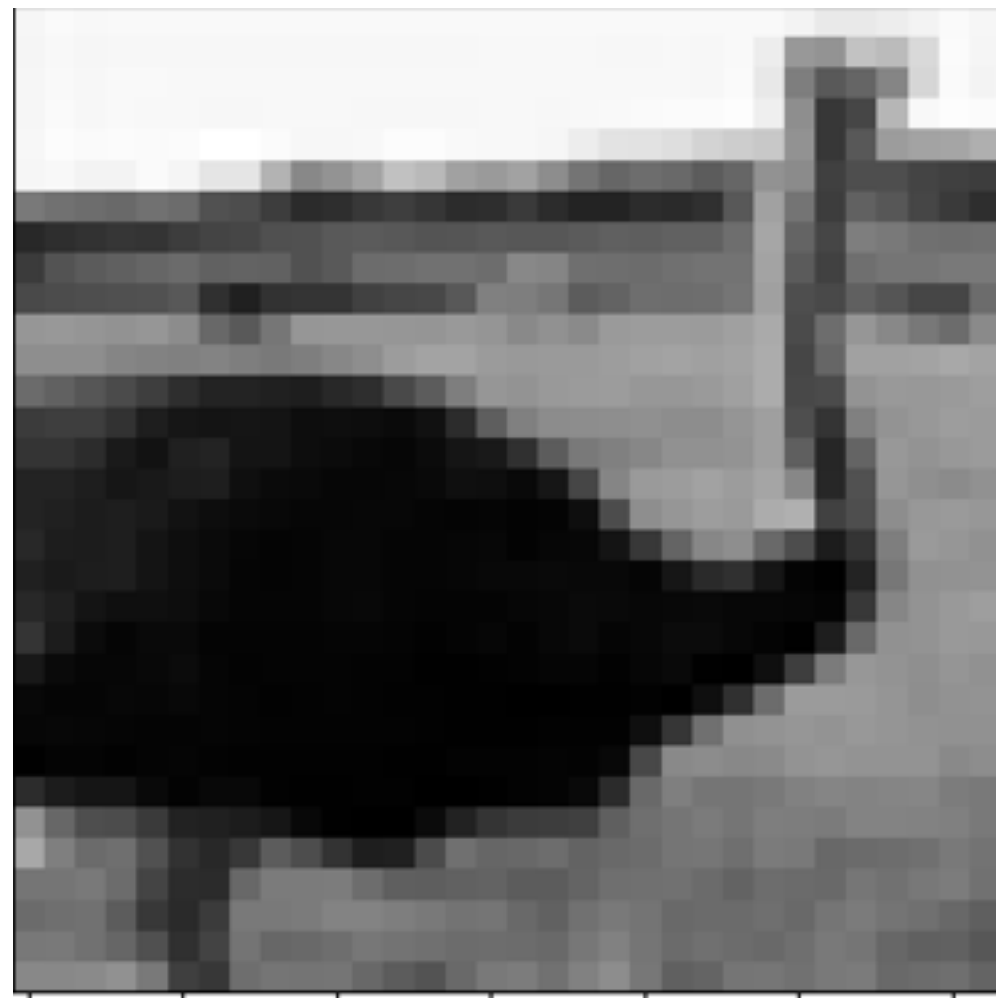


```
nn.Conv2d(1, 1, 6)
```

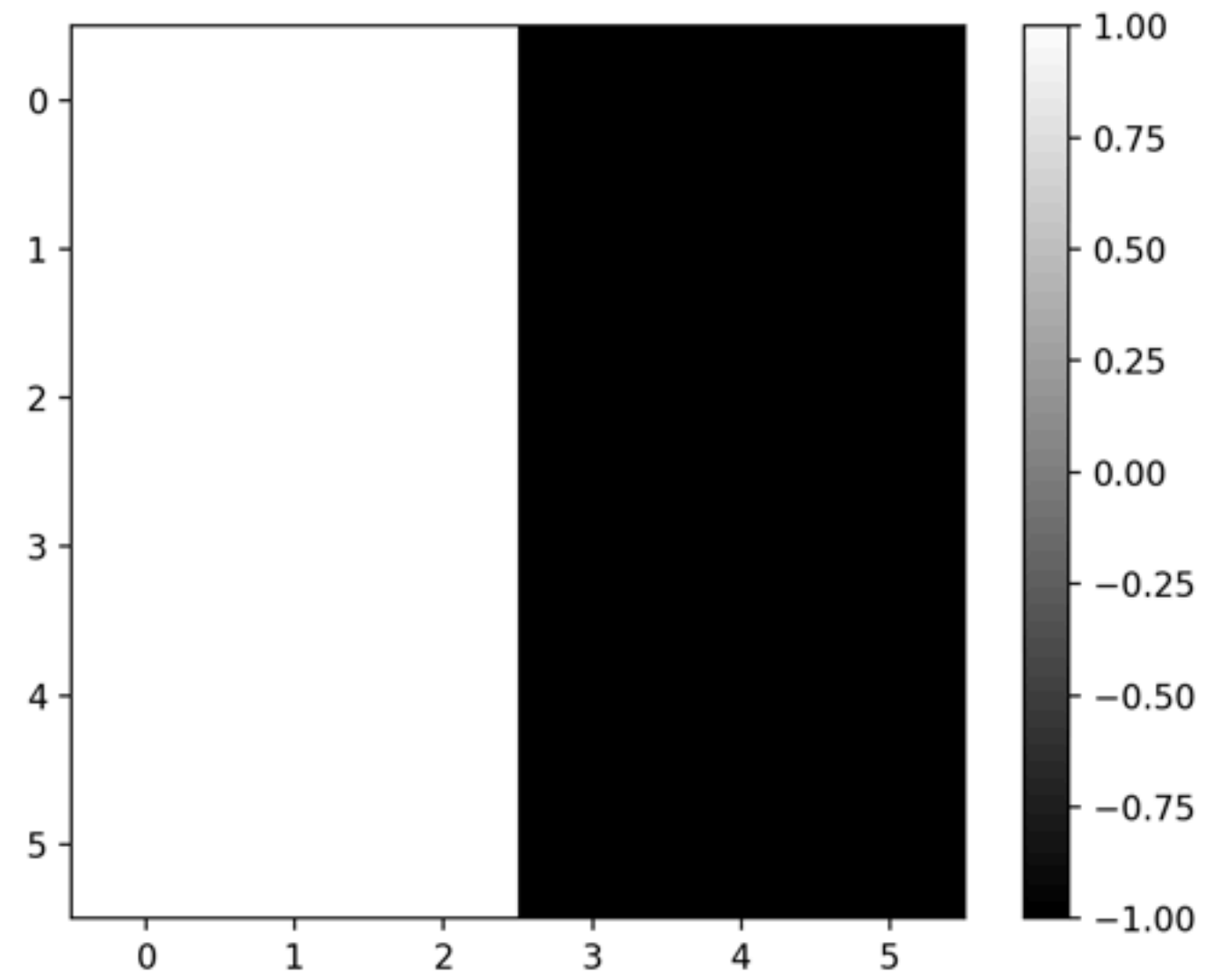


```
nn.Conv2d(1, 1, 12)
```

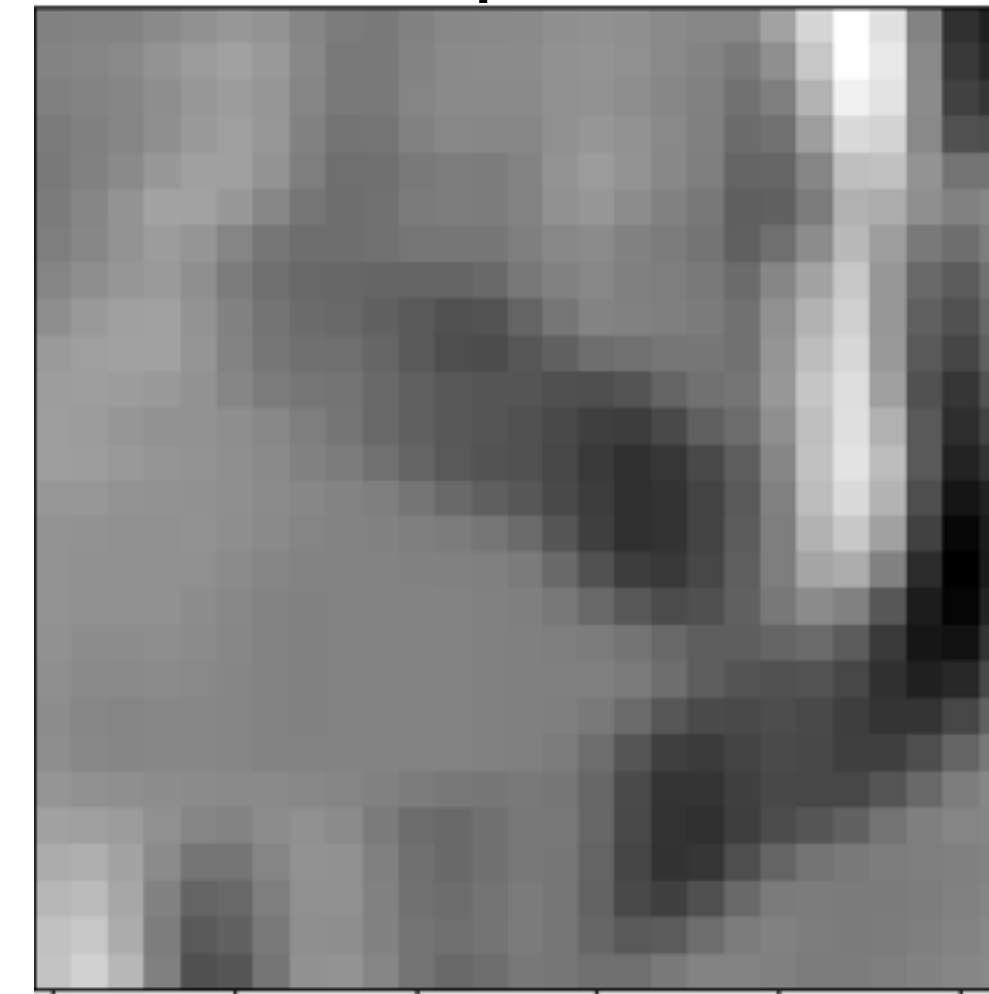

Input image



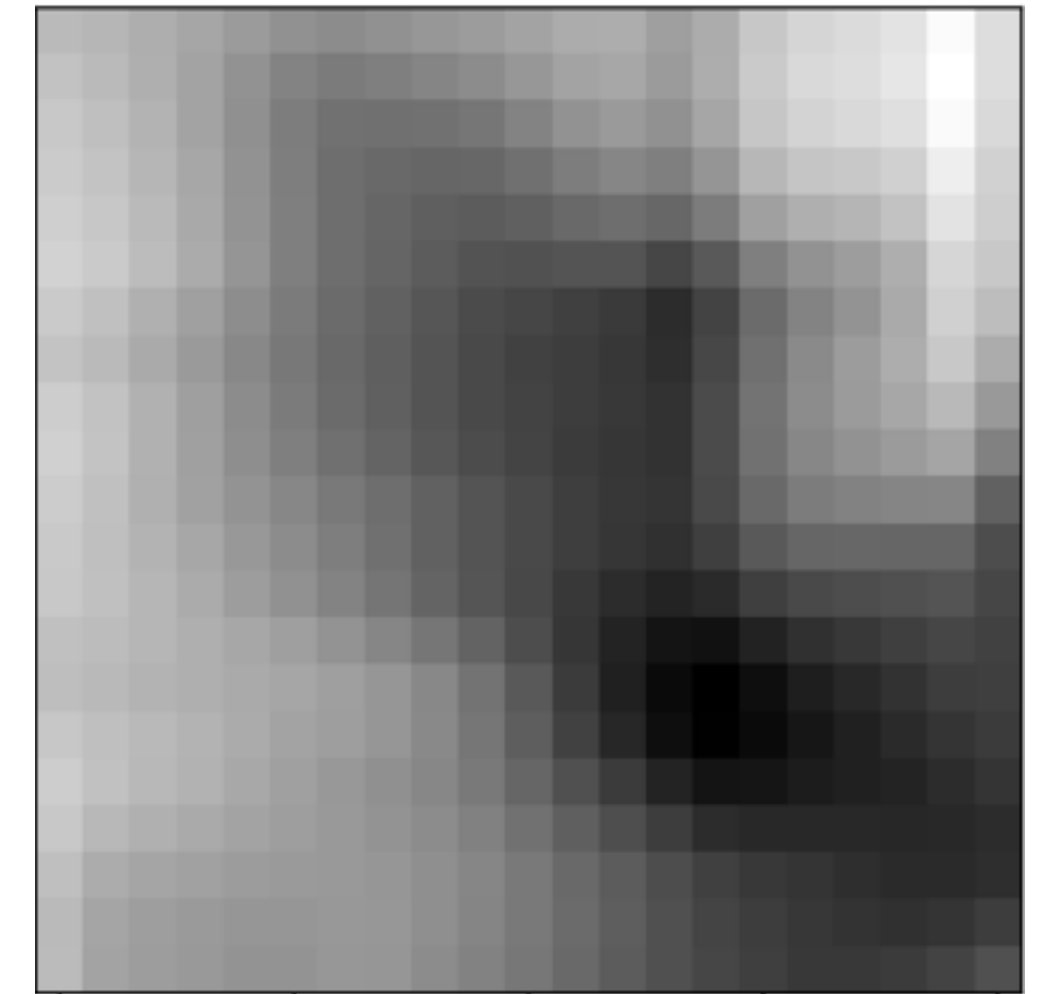
Input kernel



Output

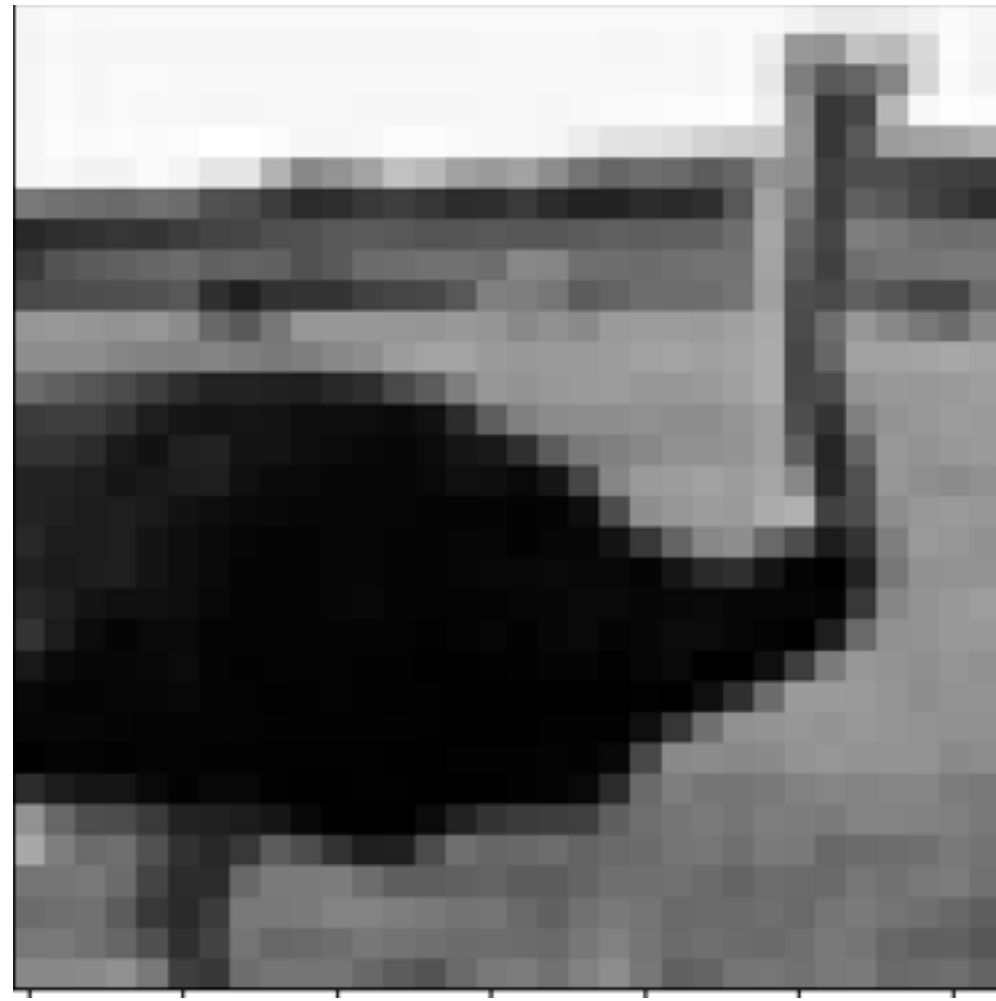


```
nn.Conv2d(1, 1, 6)
```

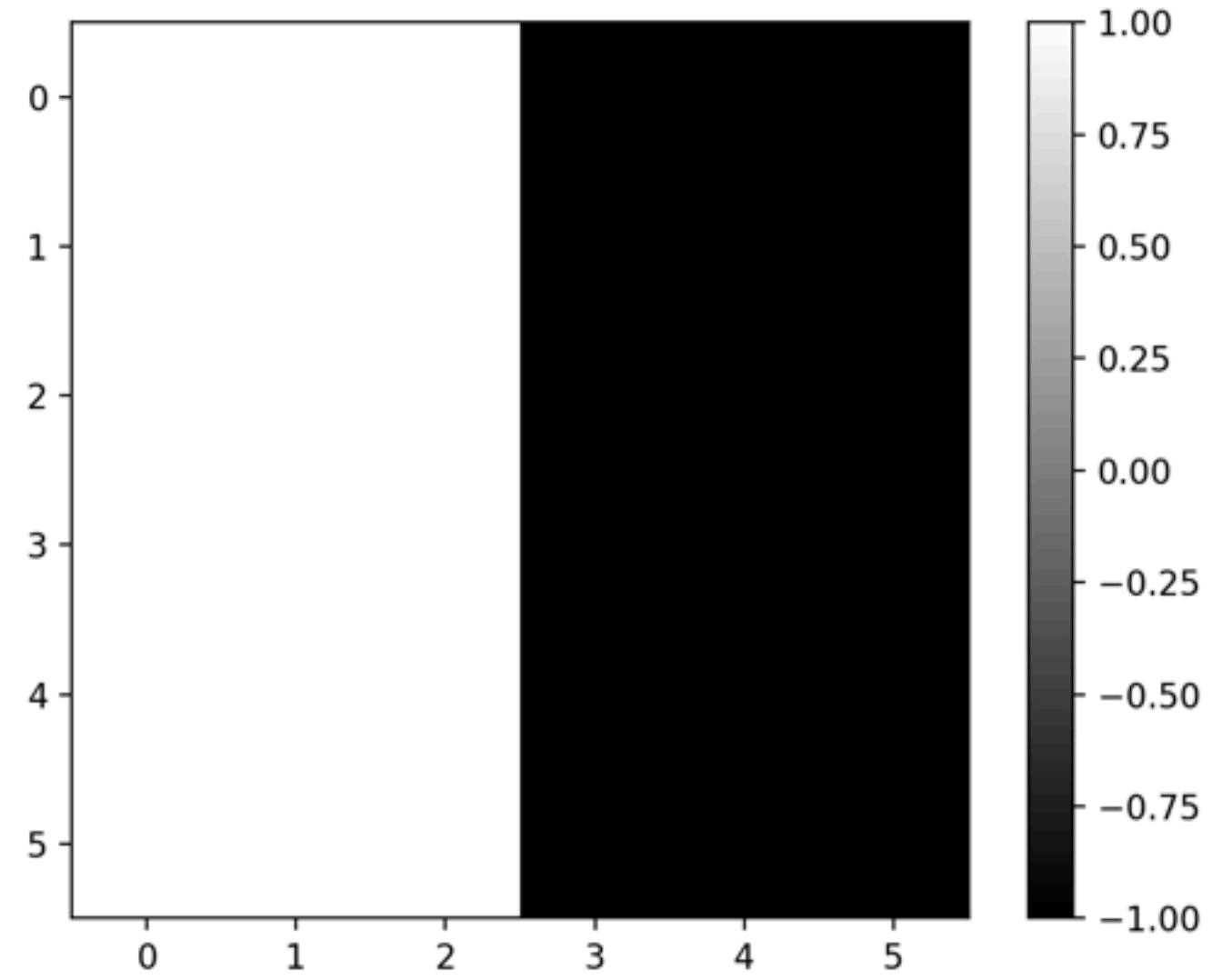


```
nn.Conv2d(1, 1, 12)
```

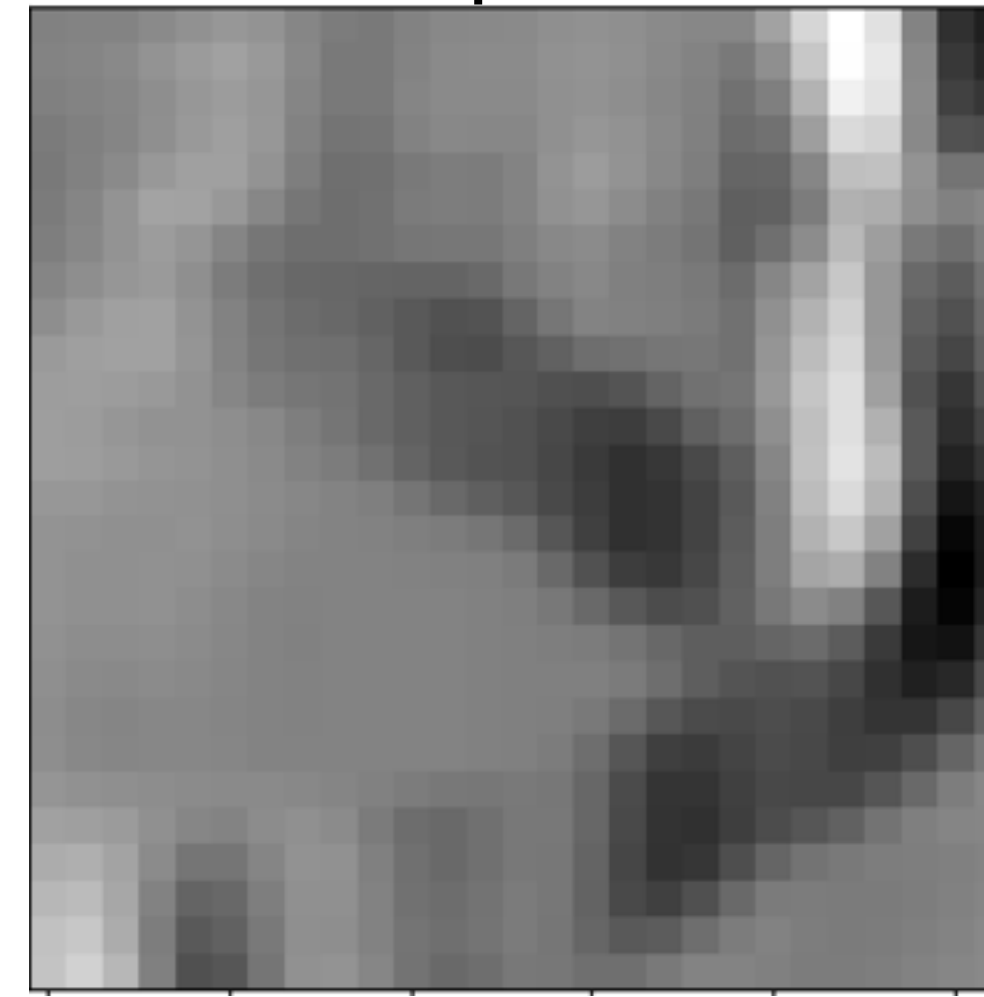
Input image



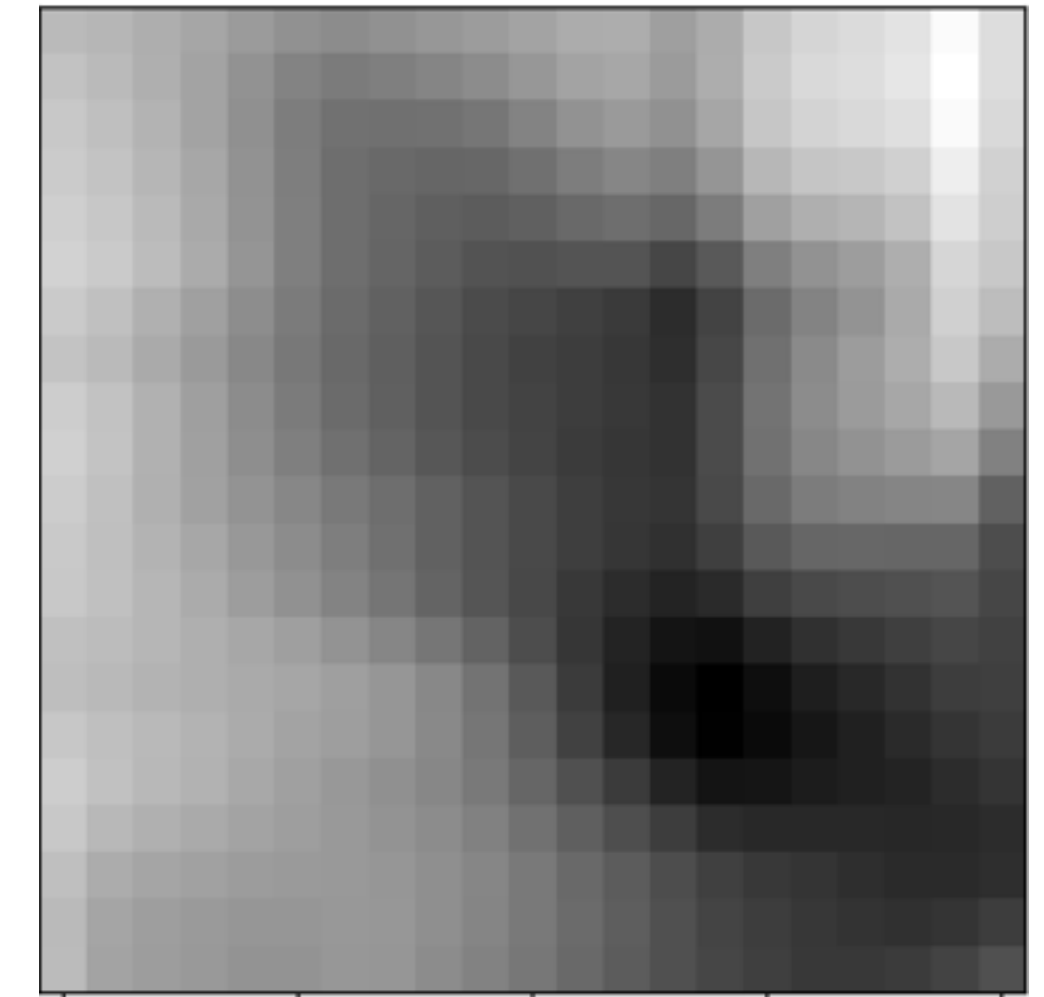
Input kernel



Output

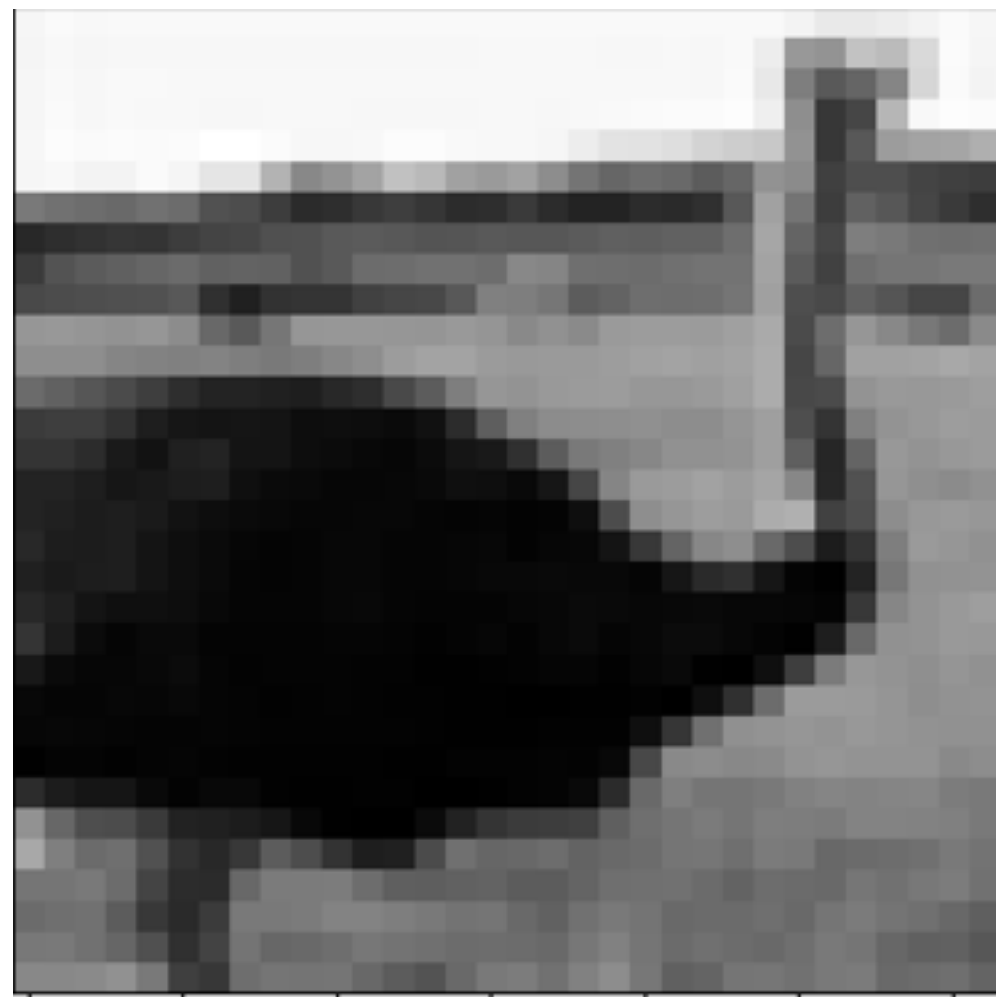


```
nn.Conv2d(1, 1, 6)
```

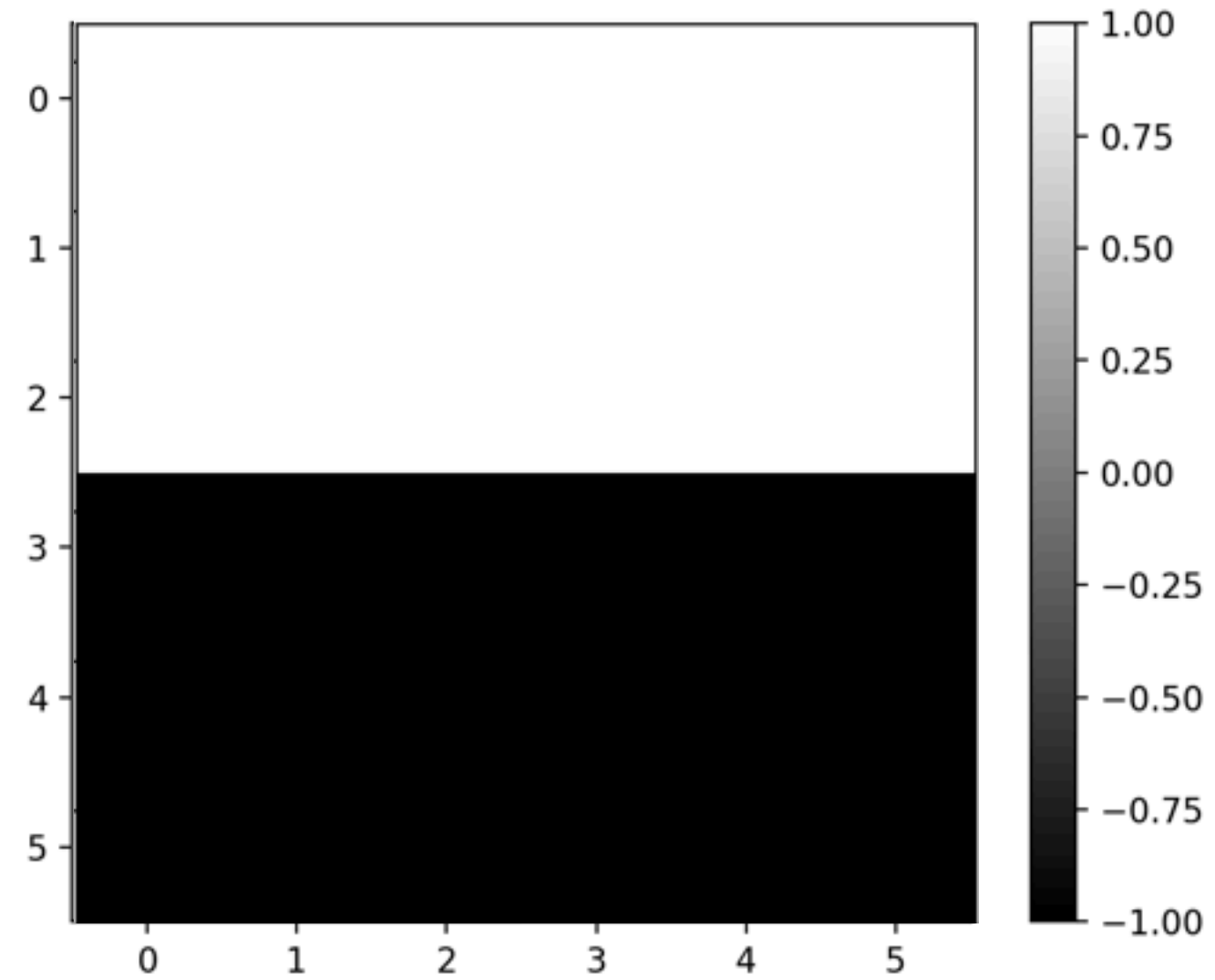


```
nn.Conv2d(1, 1, 12)
```

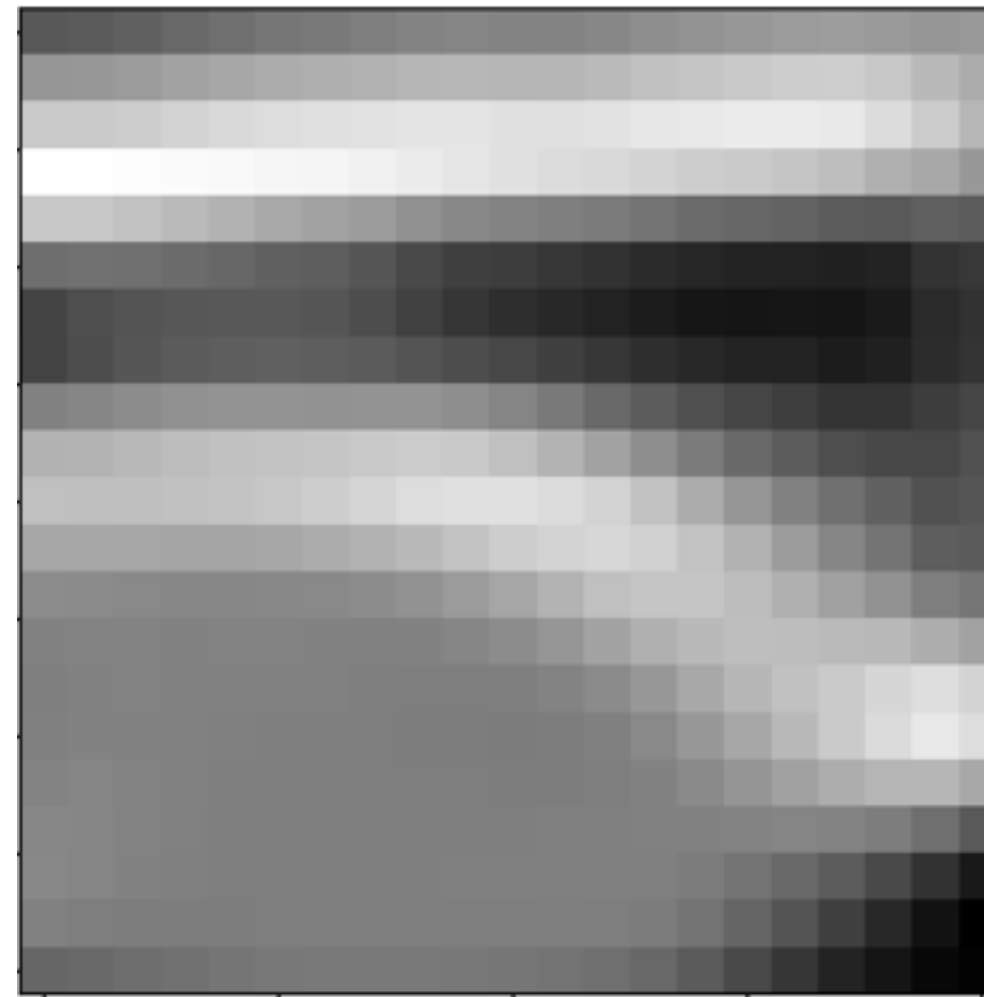
Input image



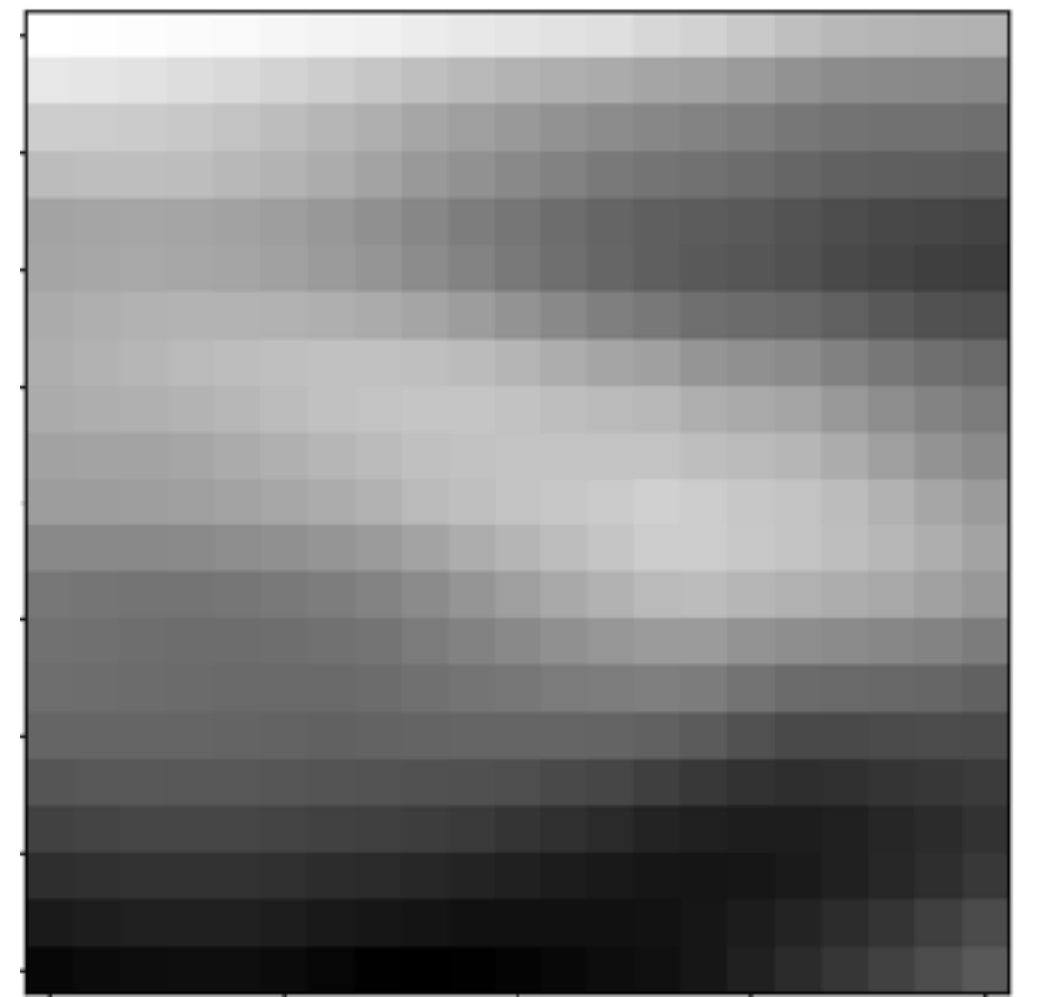
Input kernel



Output

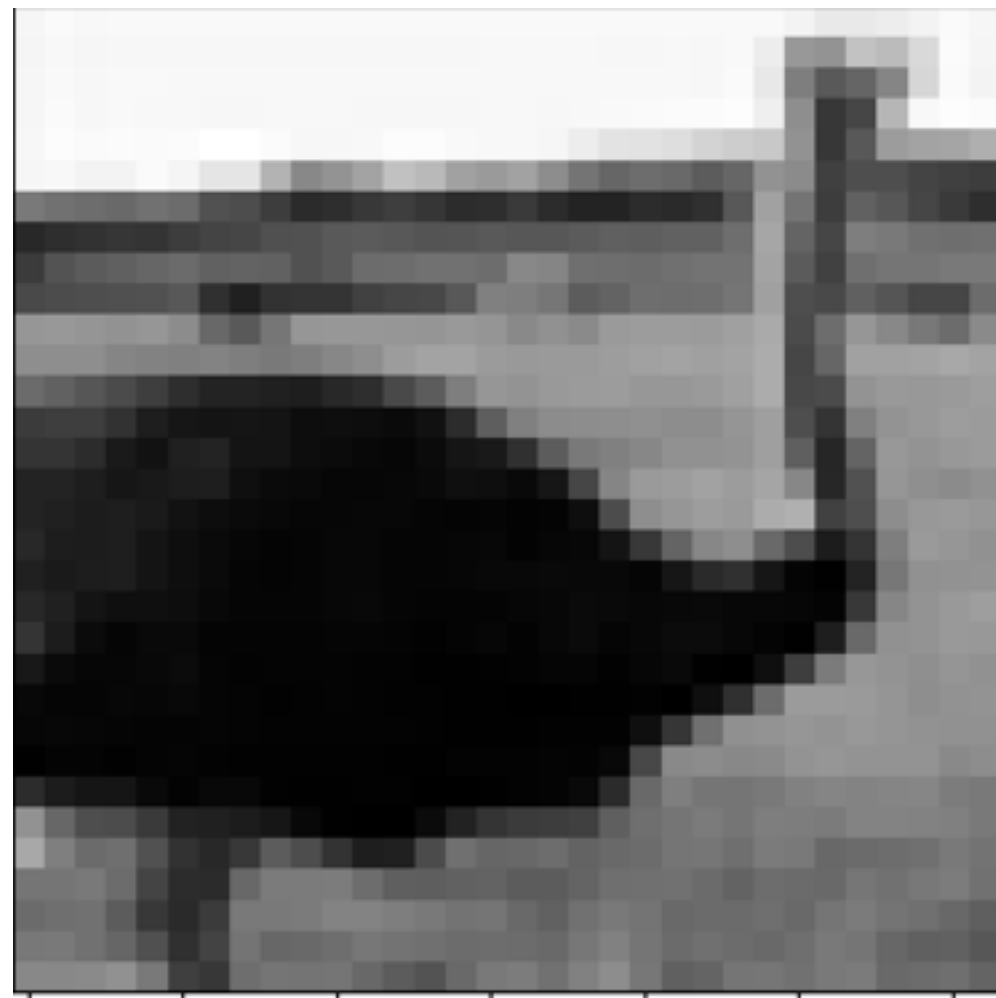


```
nn.Conv2d(1, 1, 6)
```

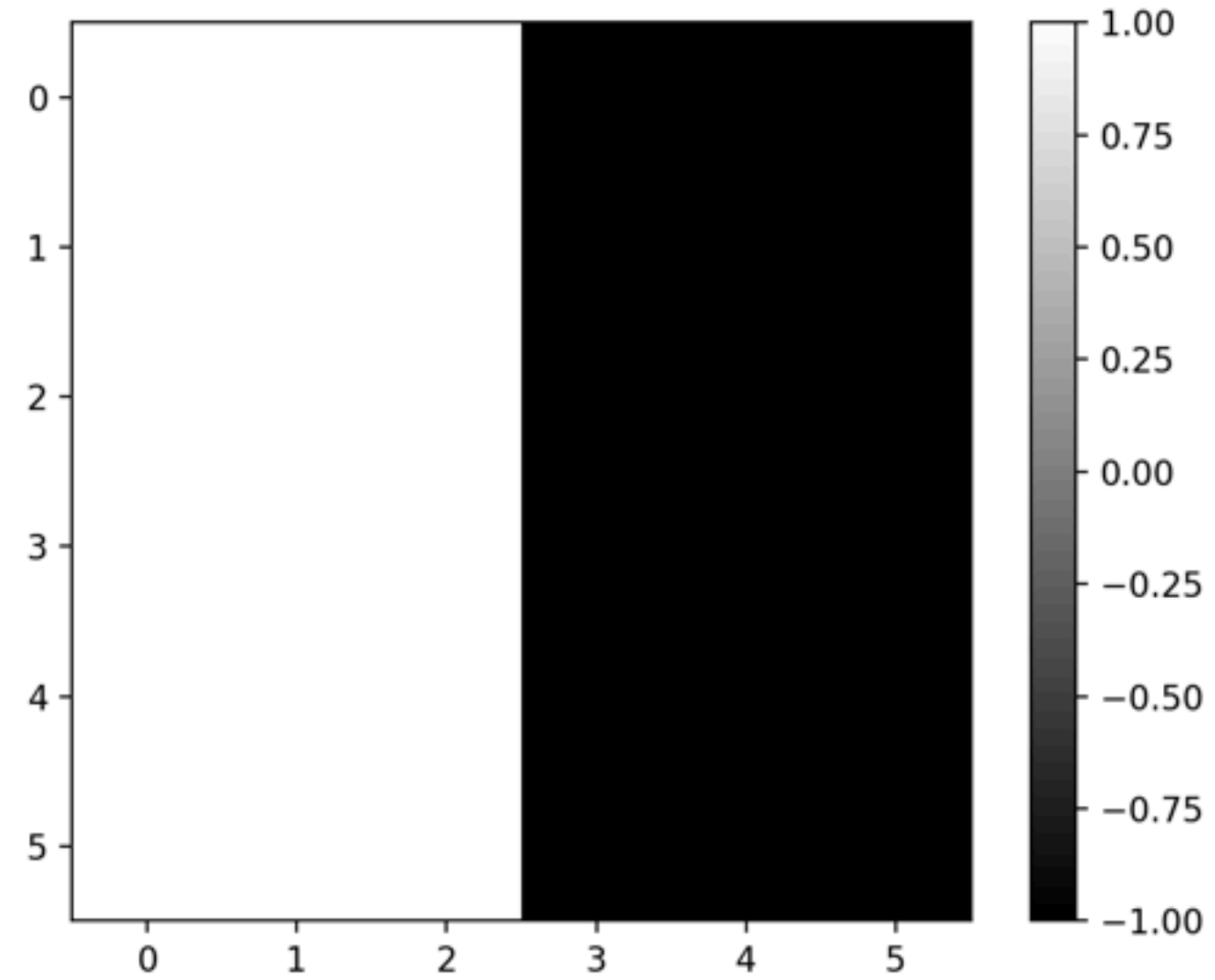


```
nn.Conv2d(1, 1, 12)
```

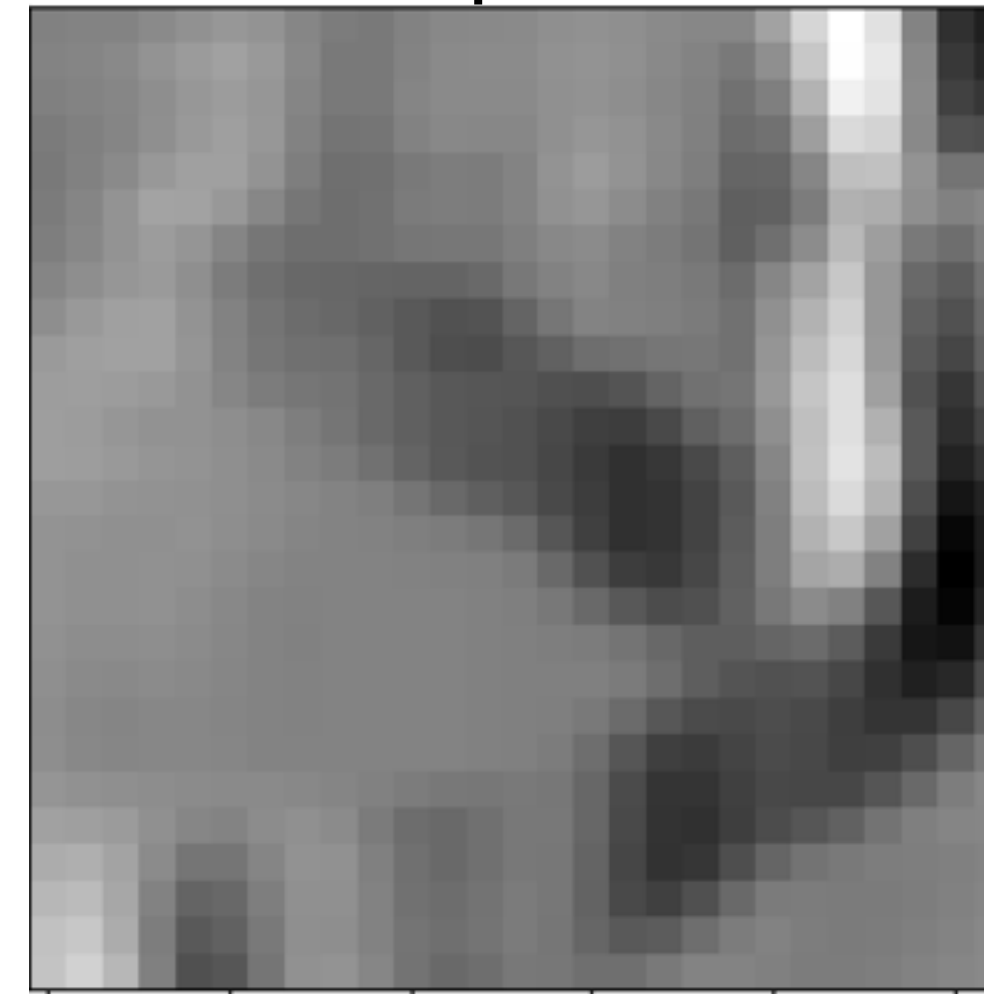
Input image



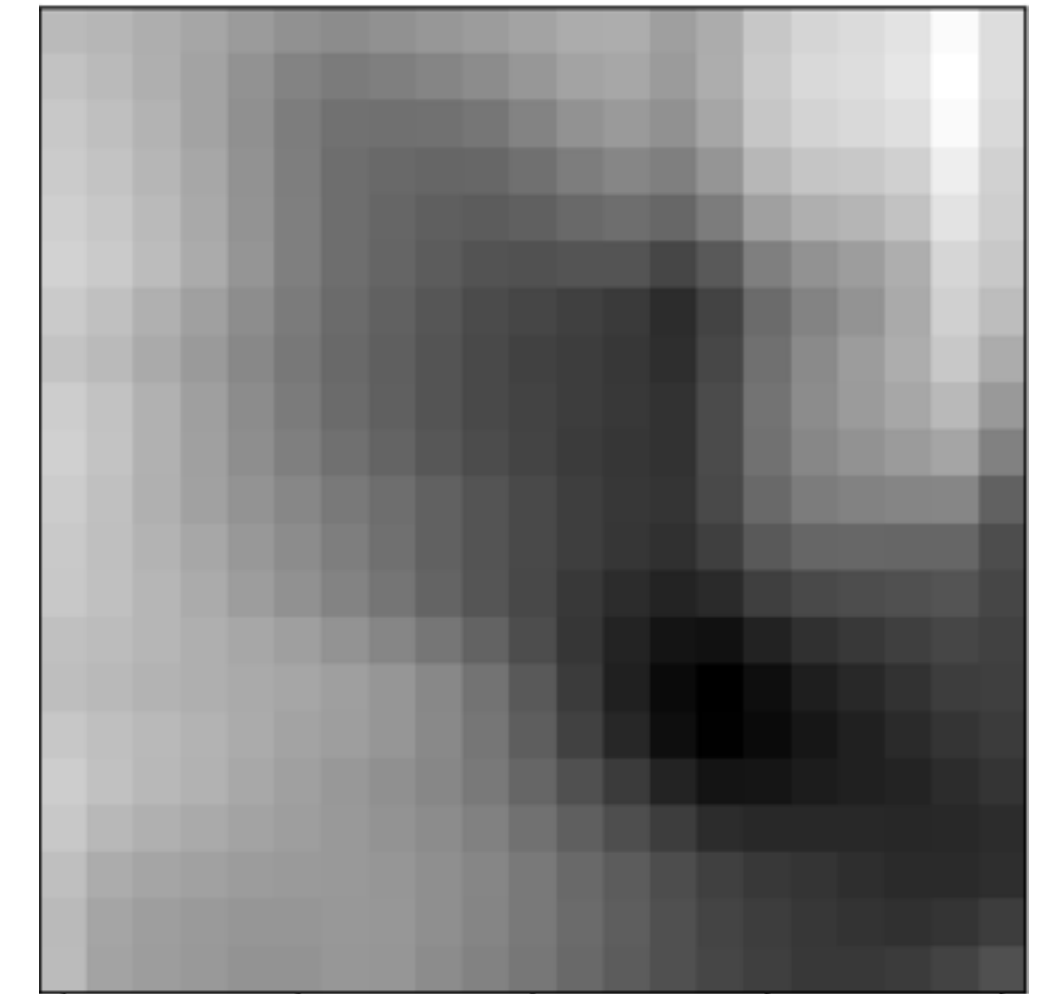
Input kernel



Output



```
nn.Conv2d(1, 1, 6)
```

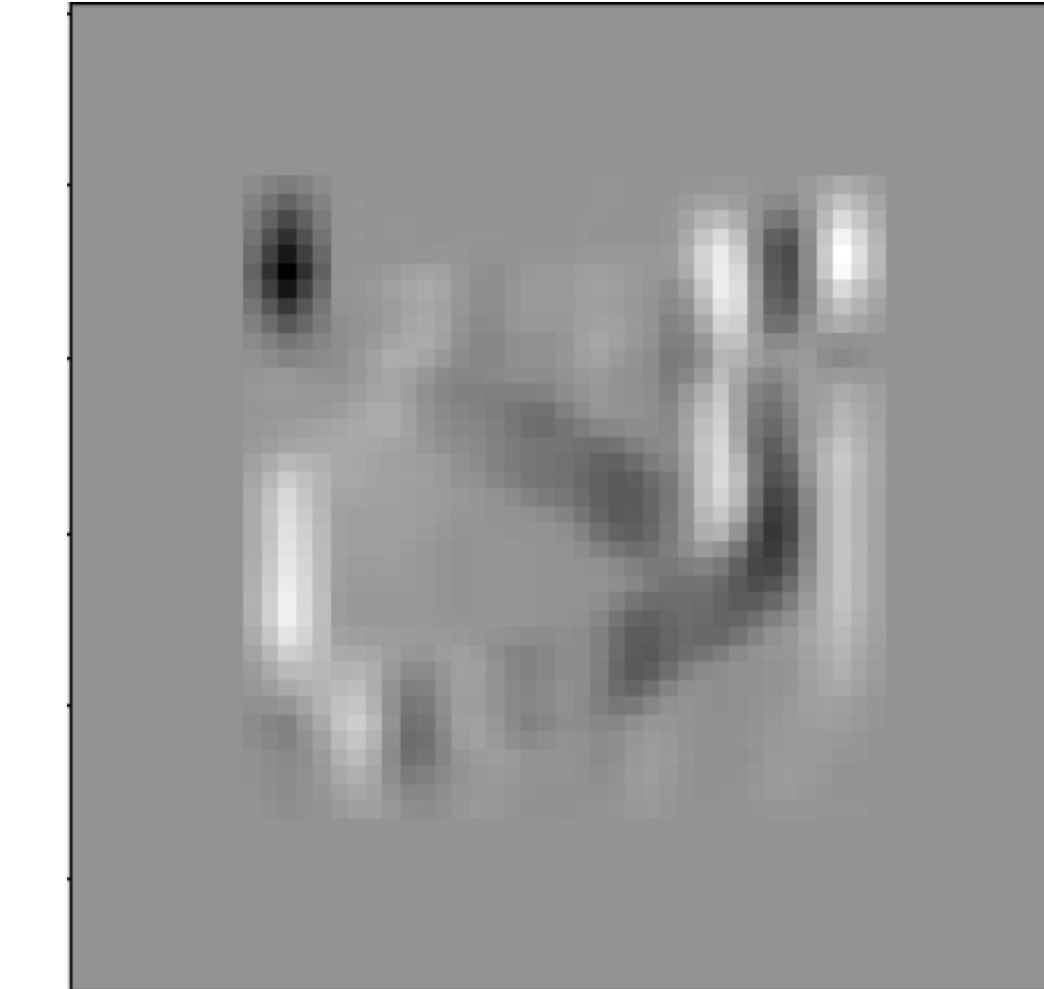


```
nn.Conv2d(1, 1, 12)
```

Meaning of padding and stride

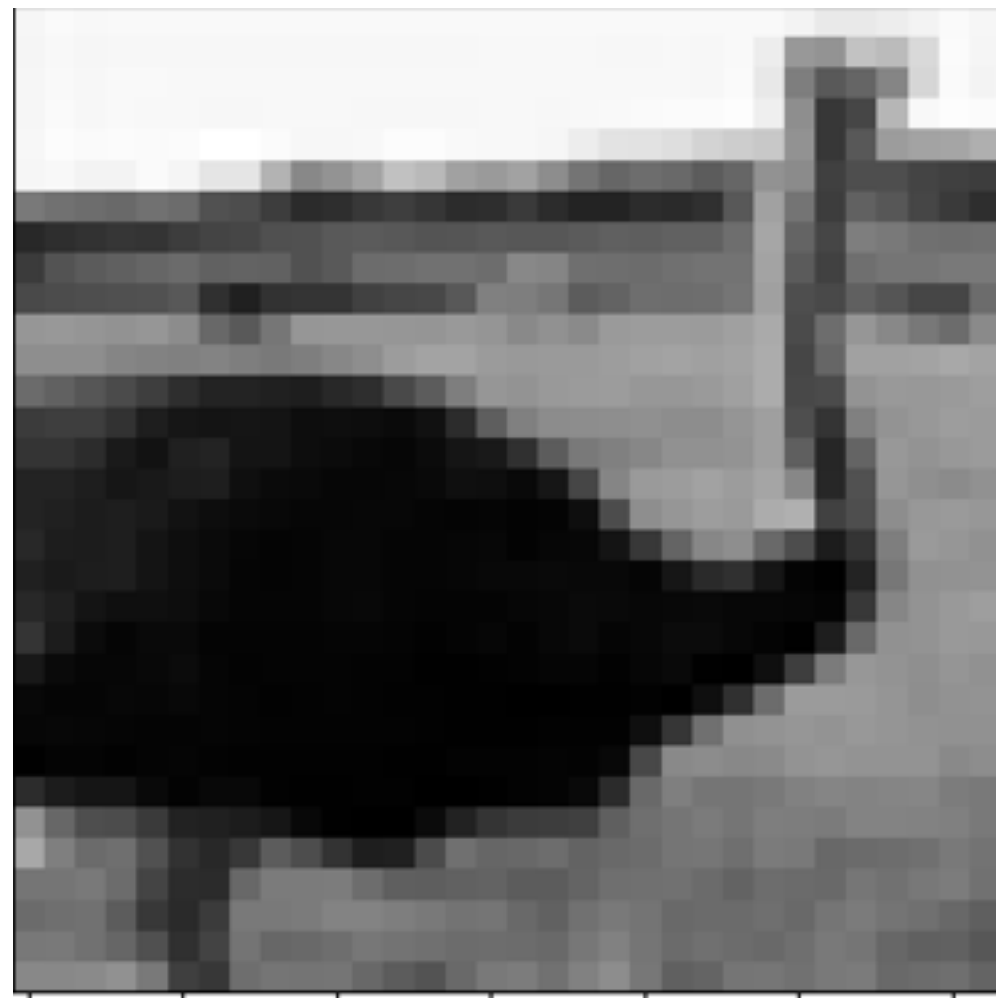


```
nn.Conv2d(1, 1, 6,  
stride=(3, 3))
```

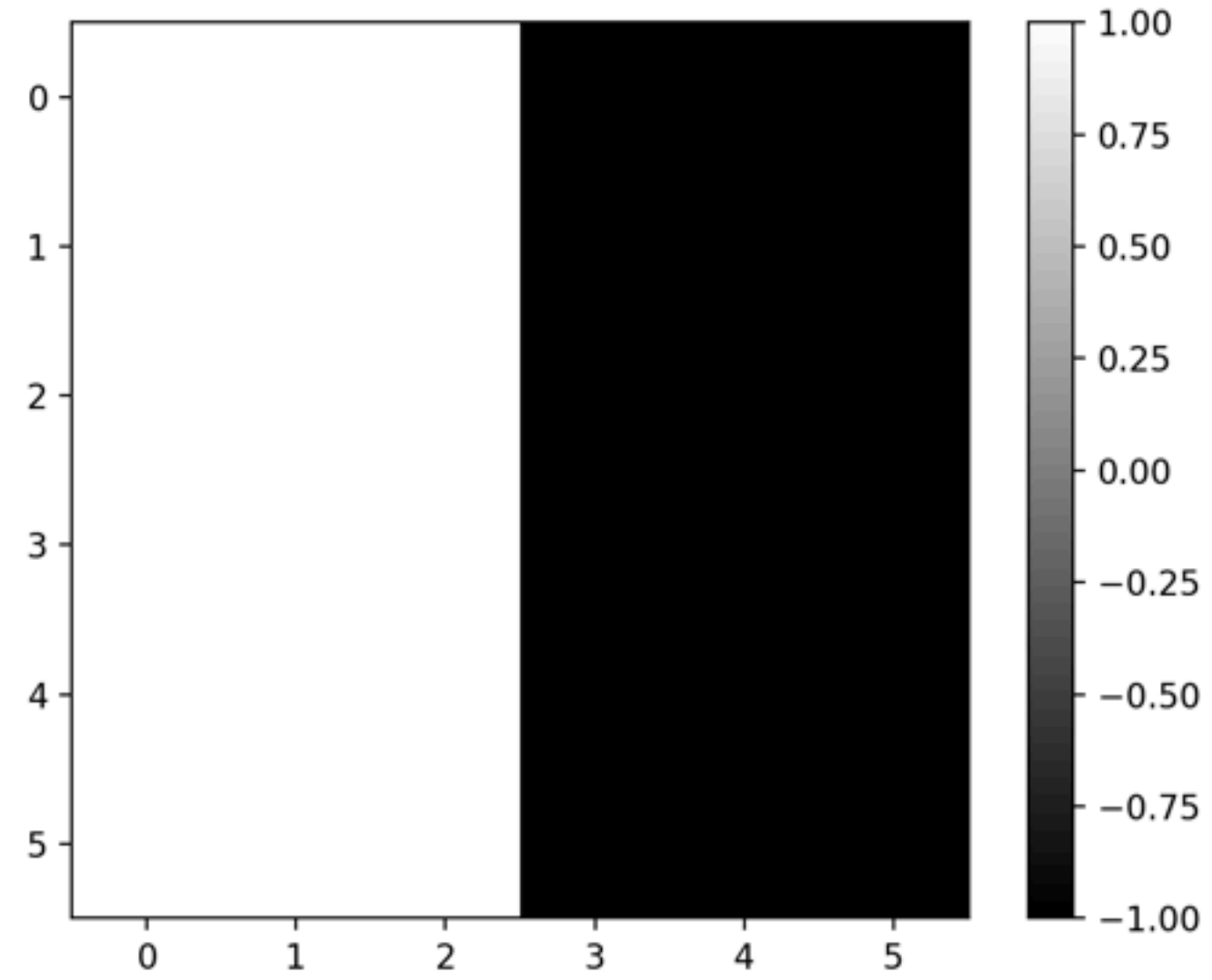


```
nn.Conv2d(1, 1, 6,  
padding=(15, 15))
```

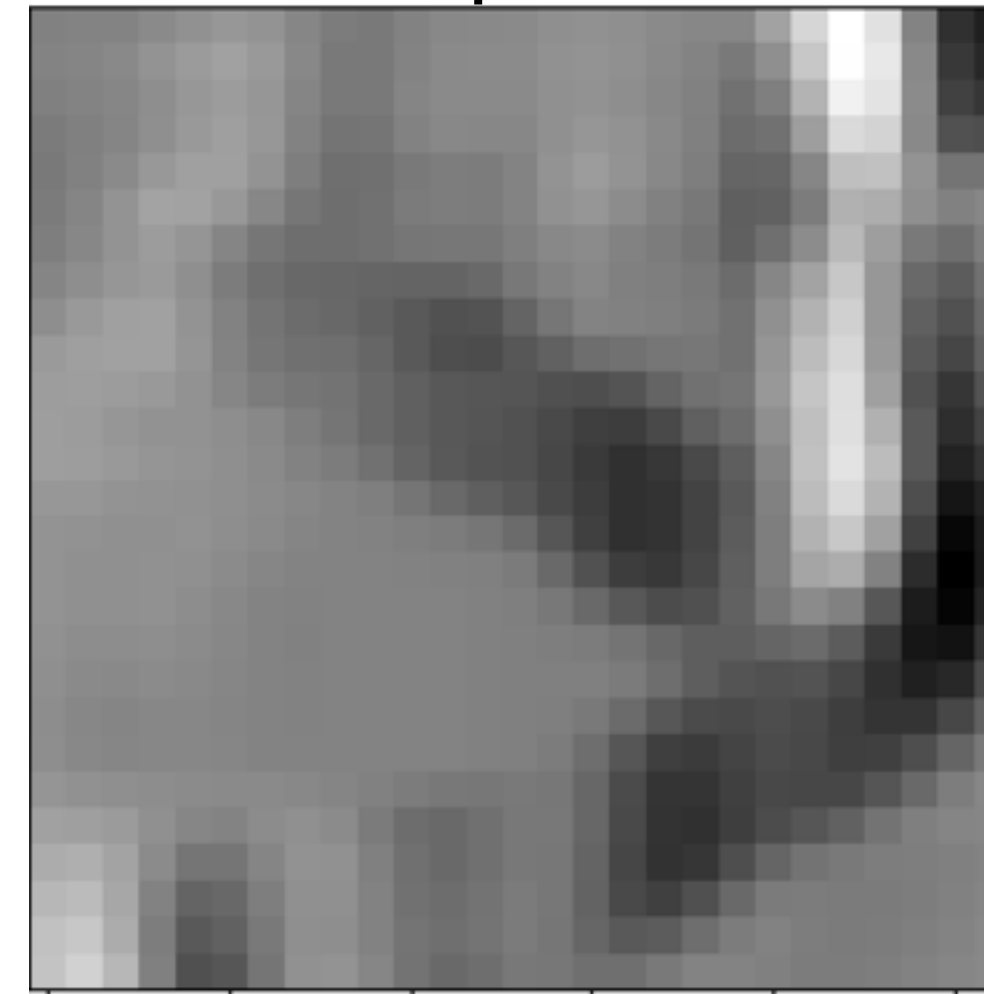

Input image



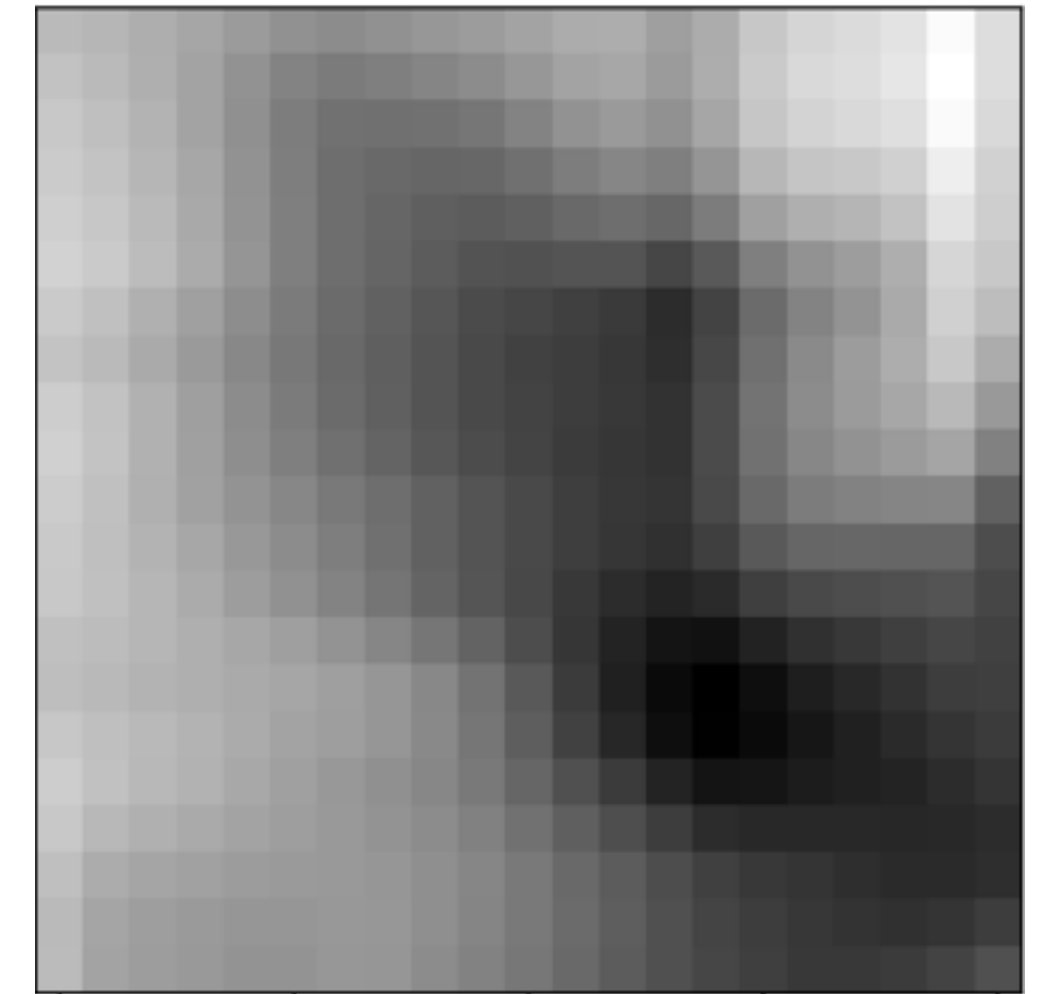
Input kernel



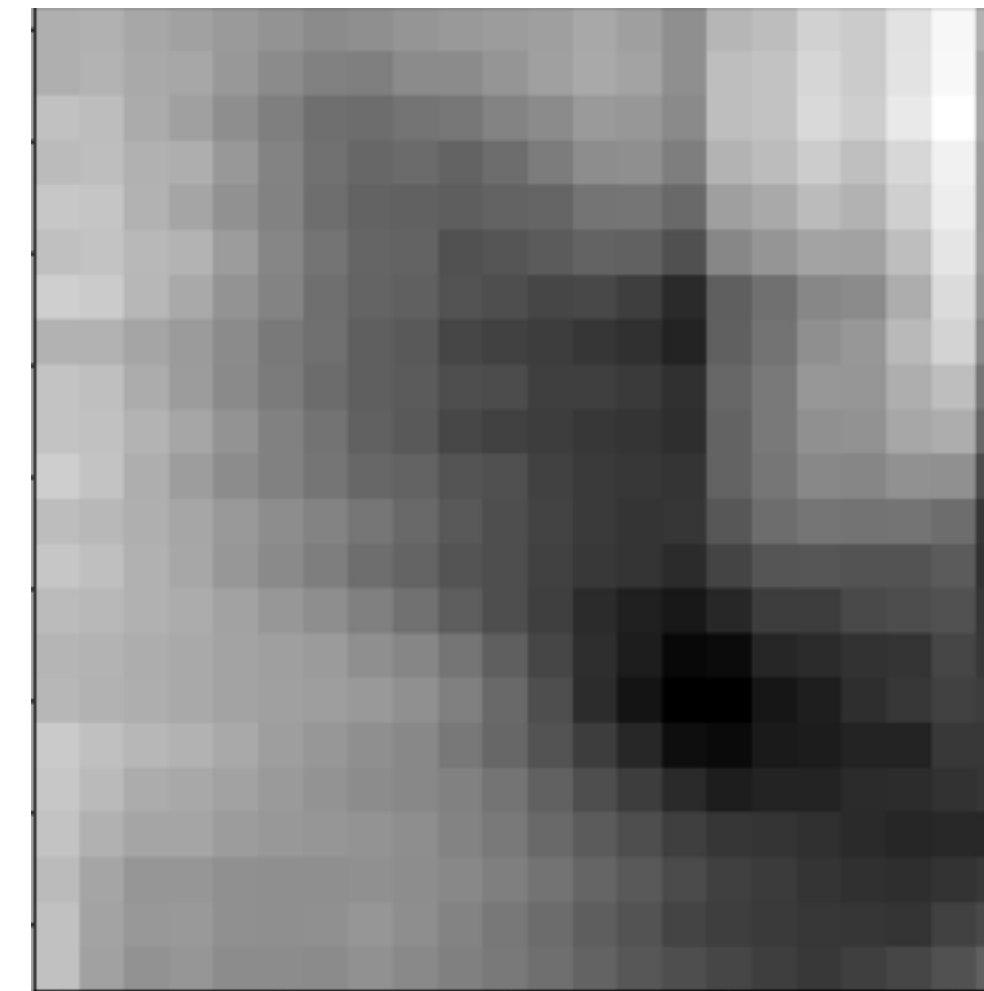
Output



```
nn.Conv2d(1, 1, 6)
```



```
nn.Conv2d(1, 1, 12)
```

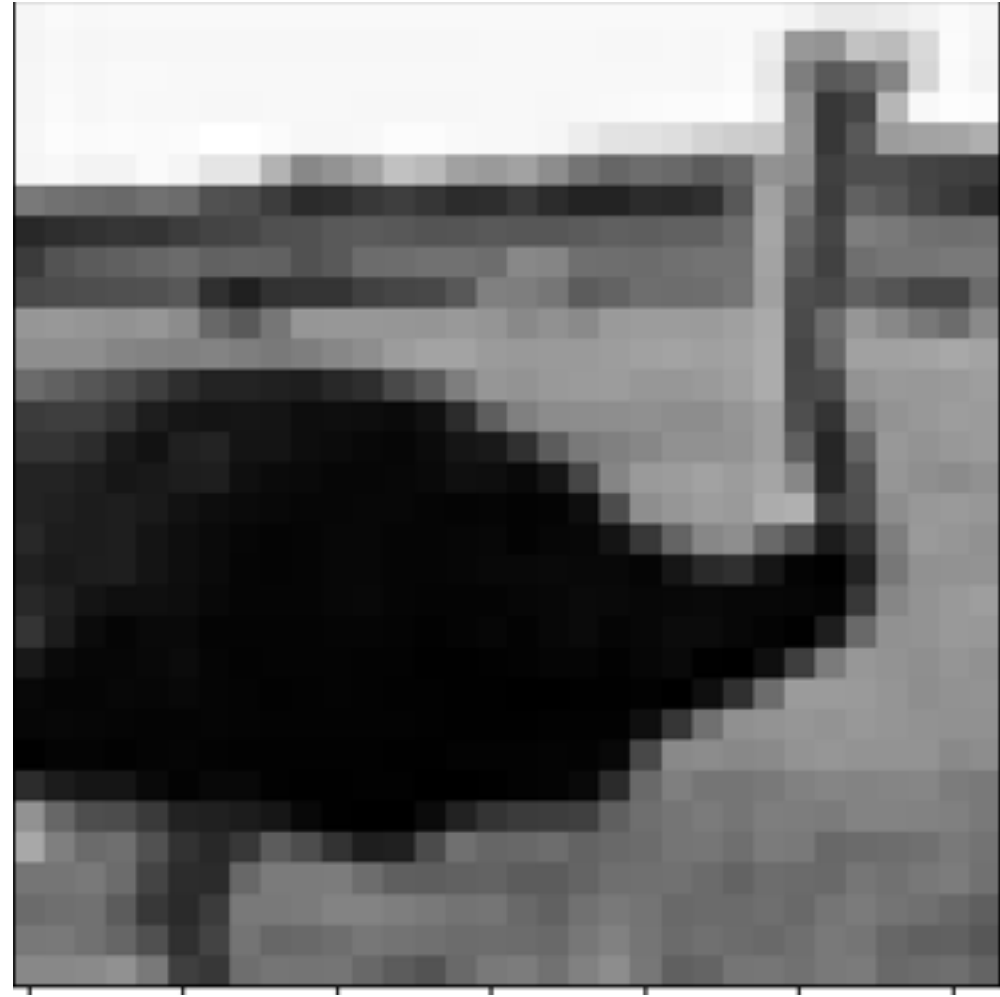


```
nn.Conv2d(1, 1, 6,  
dilation=2)
```

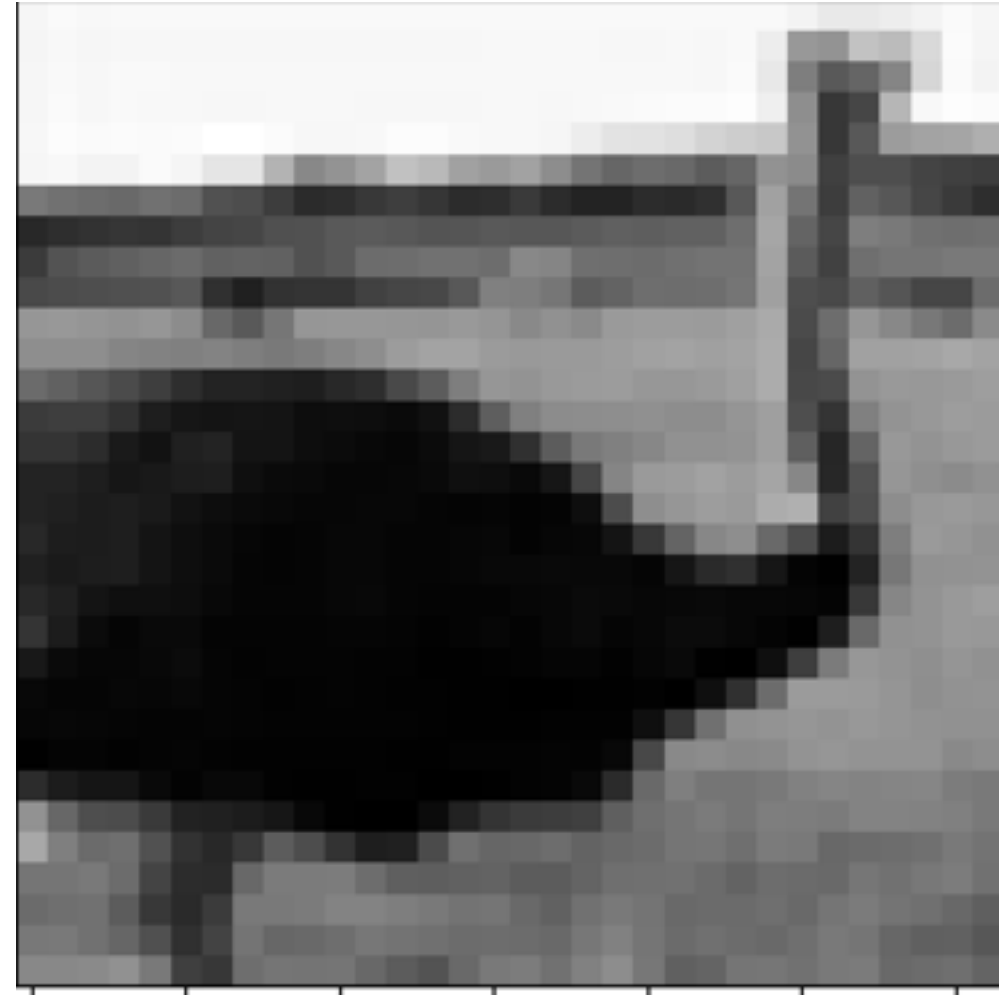
compare!

Meaning of dilation

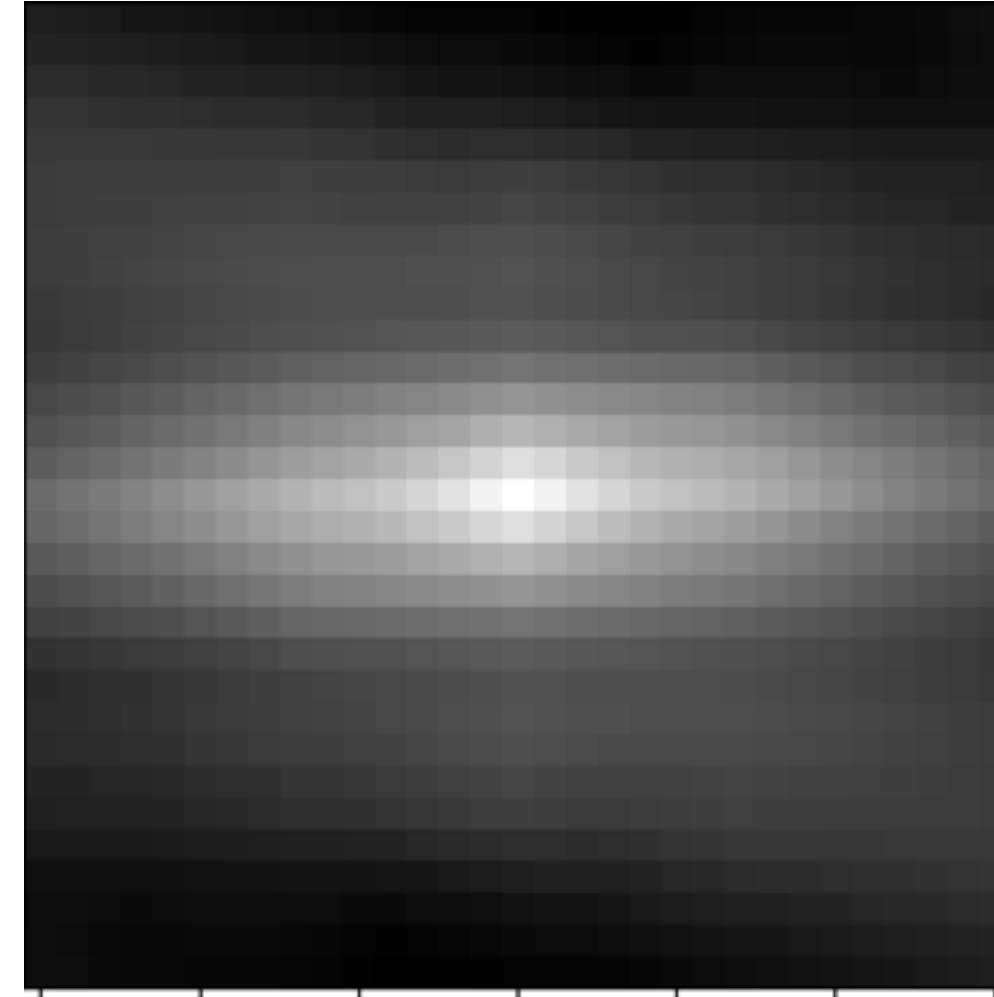
Input image



Input kernel



Output



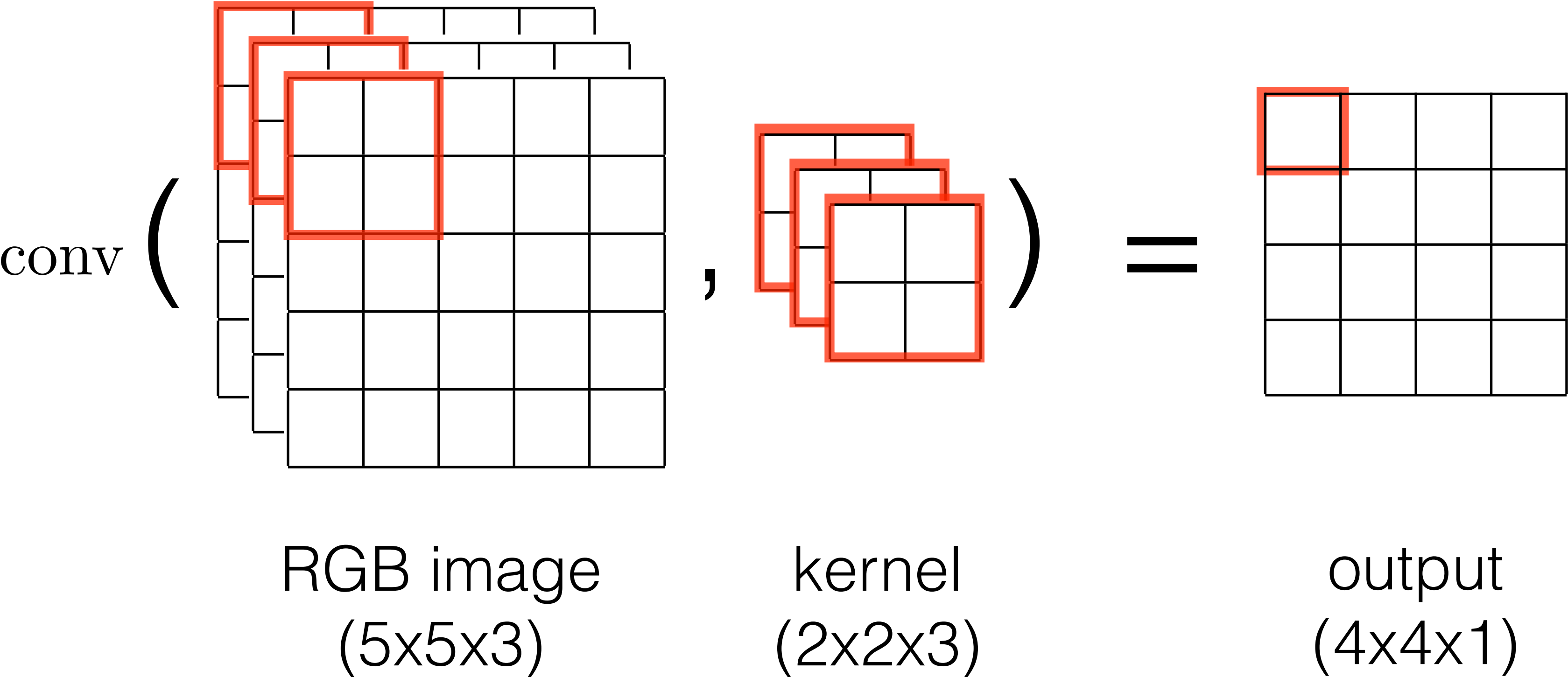
Meaning of kernel

```
nn.Conv2d(1, 1, 32,  
padding=(15, 15))
```

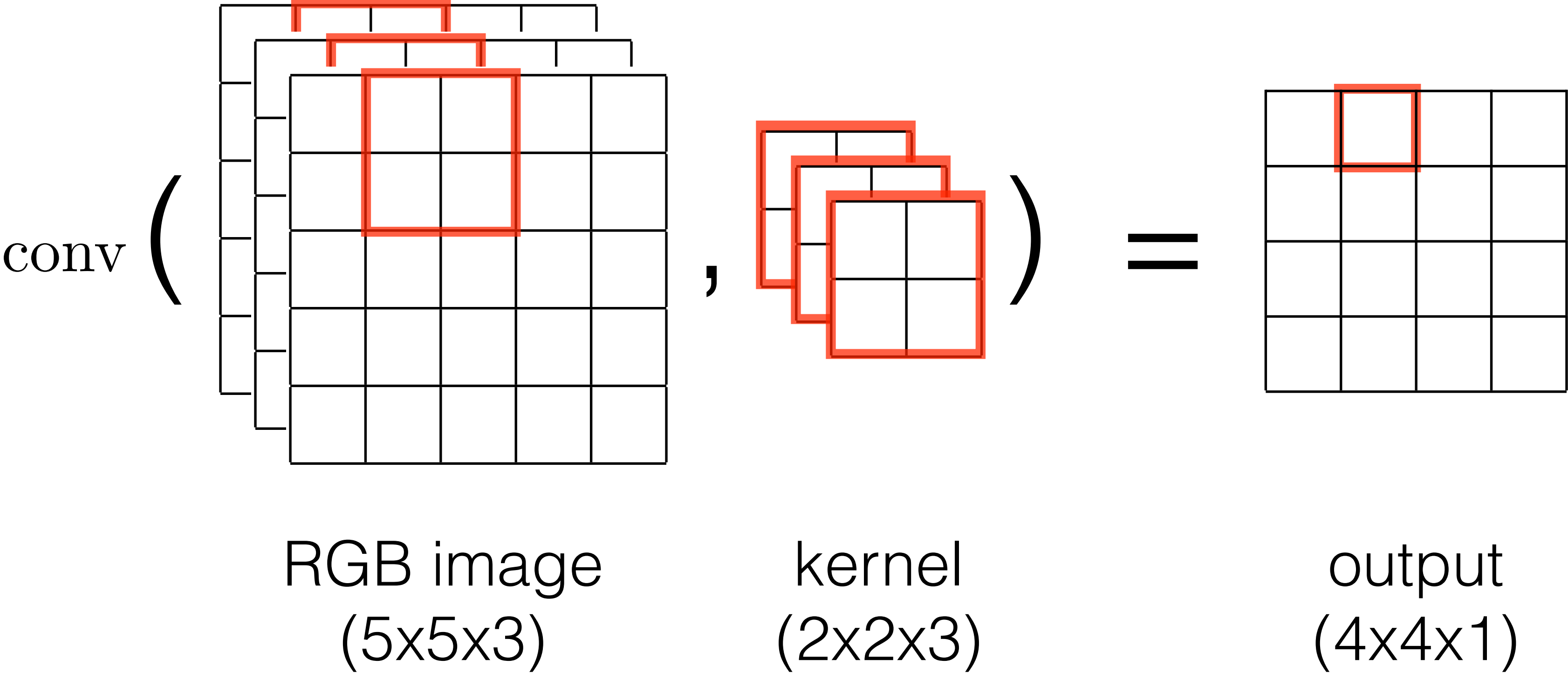
Convolution is locally applied linear classifier that computes correlation between kernel and image

Let's build a convolutional layer !

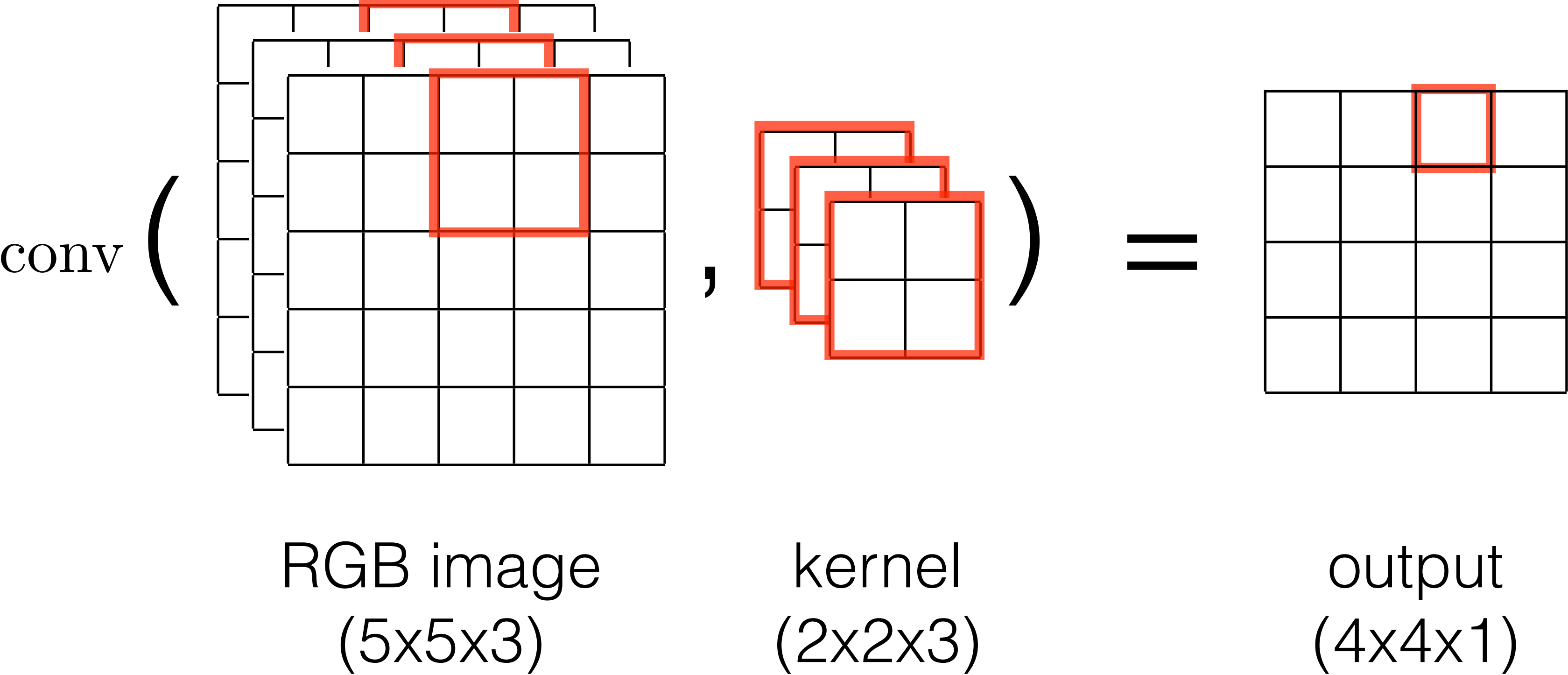
Multi-channel convolution



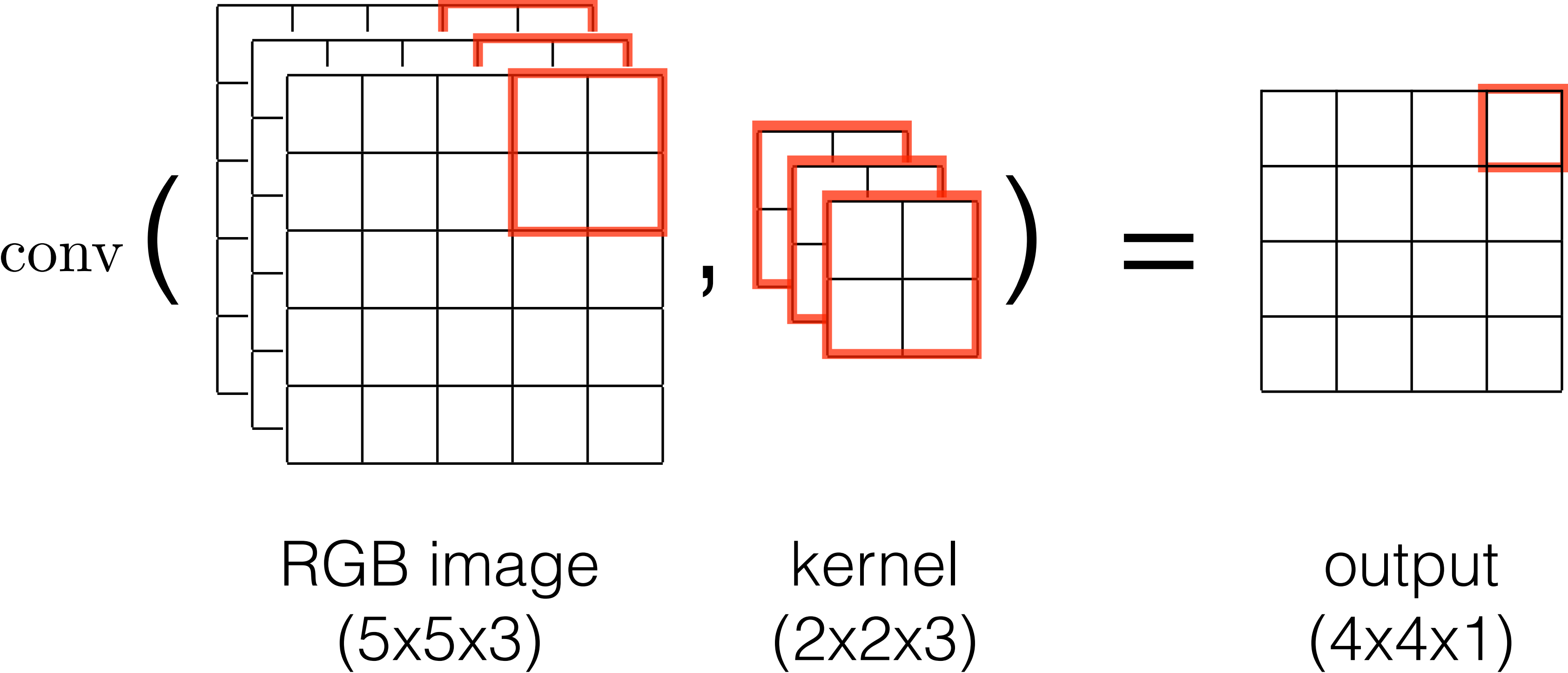
Multi-channel convolution



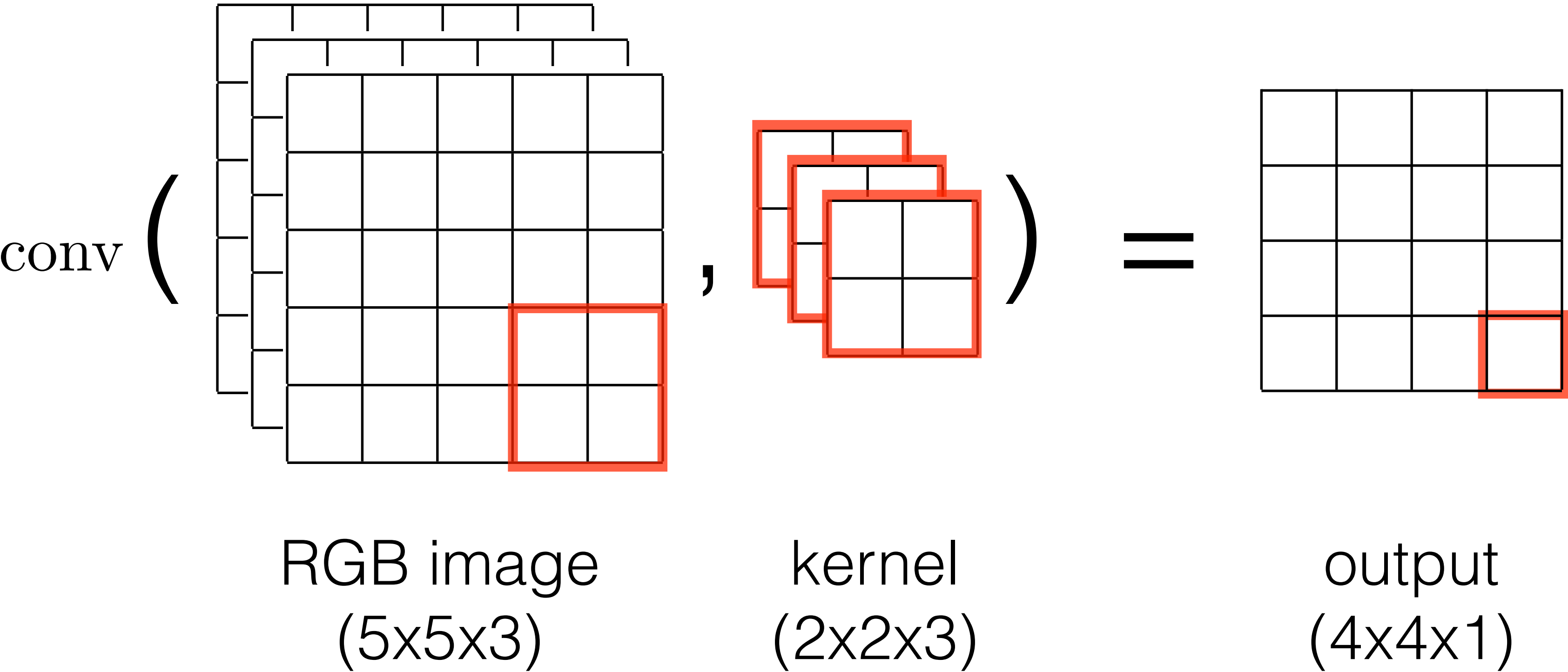
Multi-channel convolution

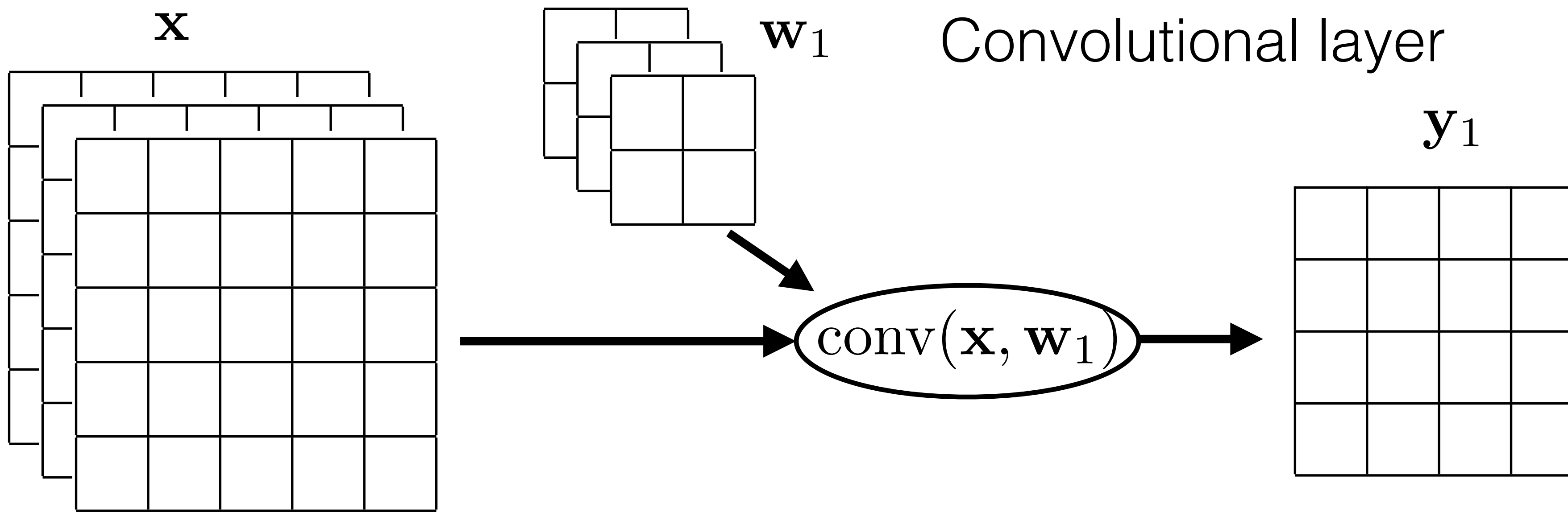


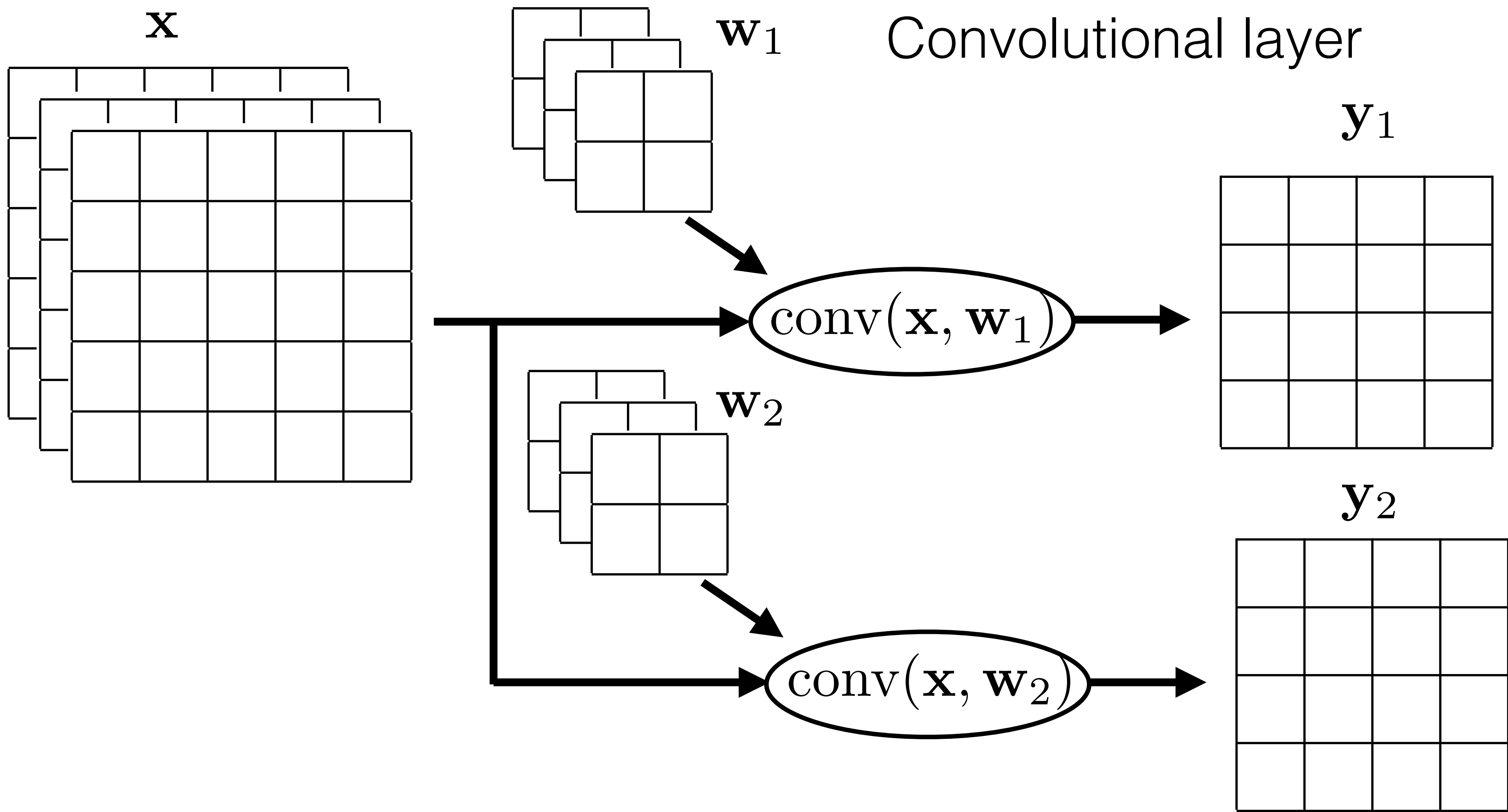
Multi-channel convolution

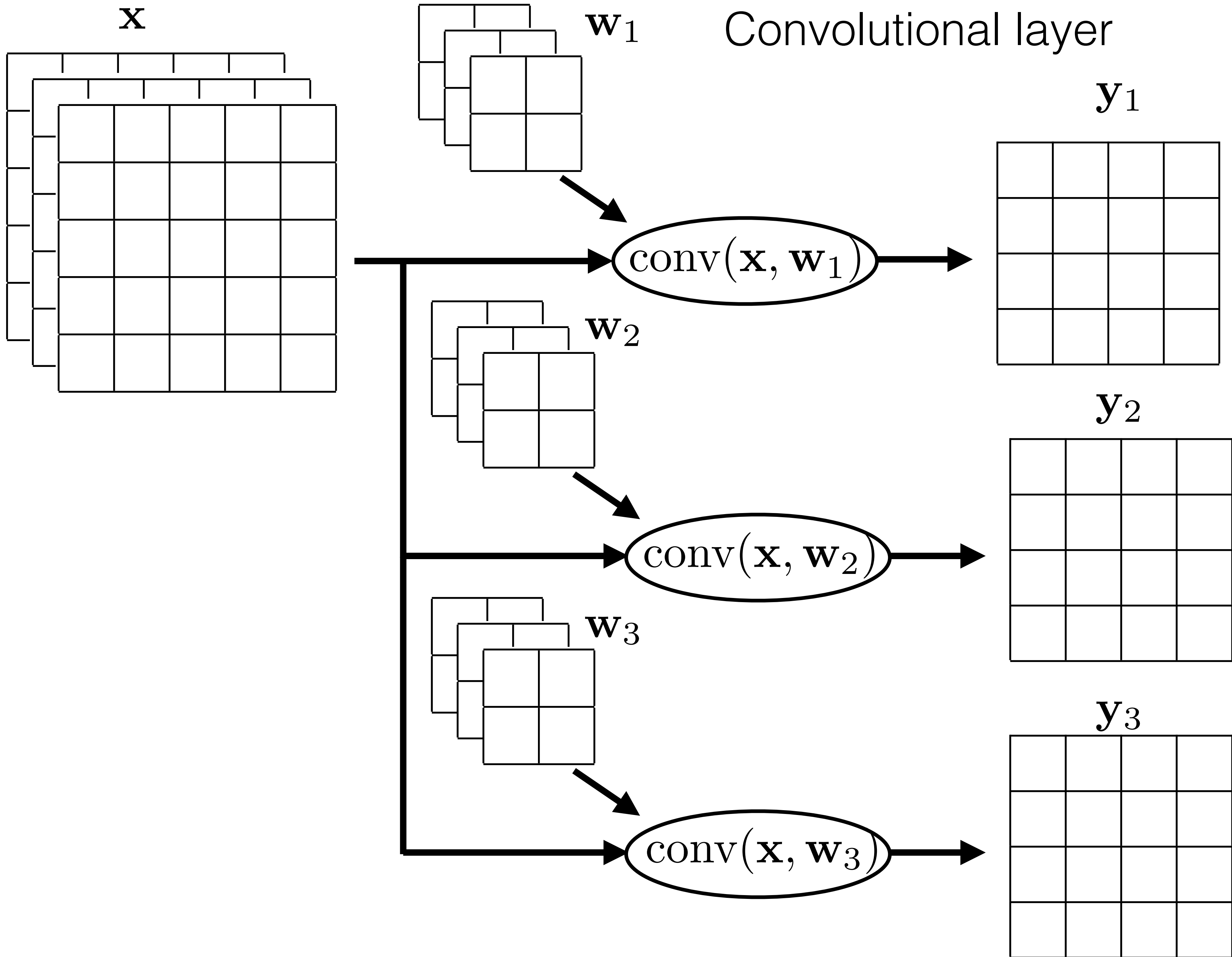


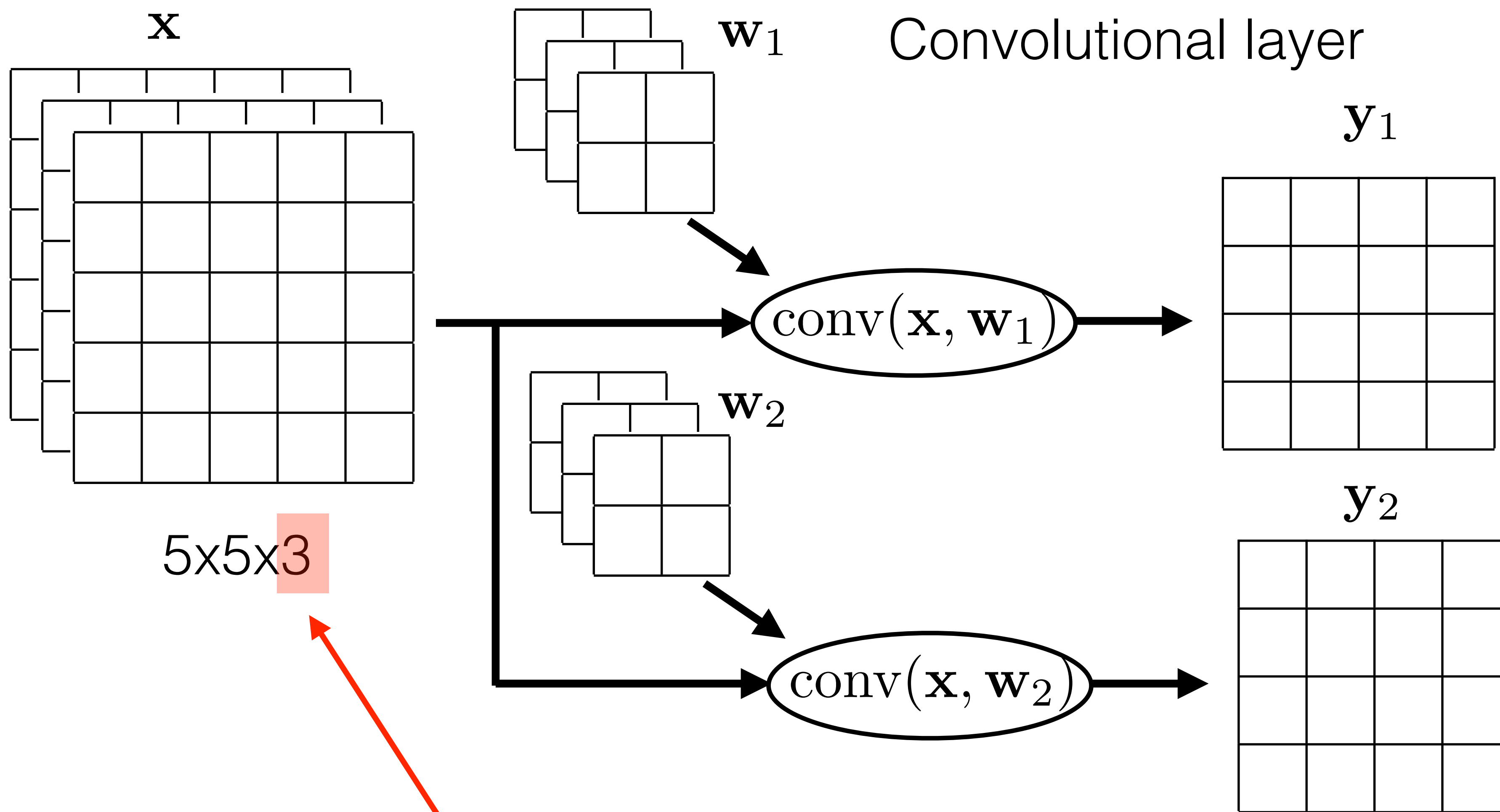
Multi-channel convolution



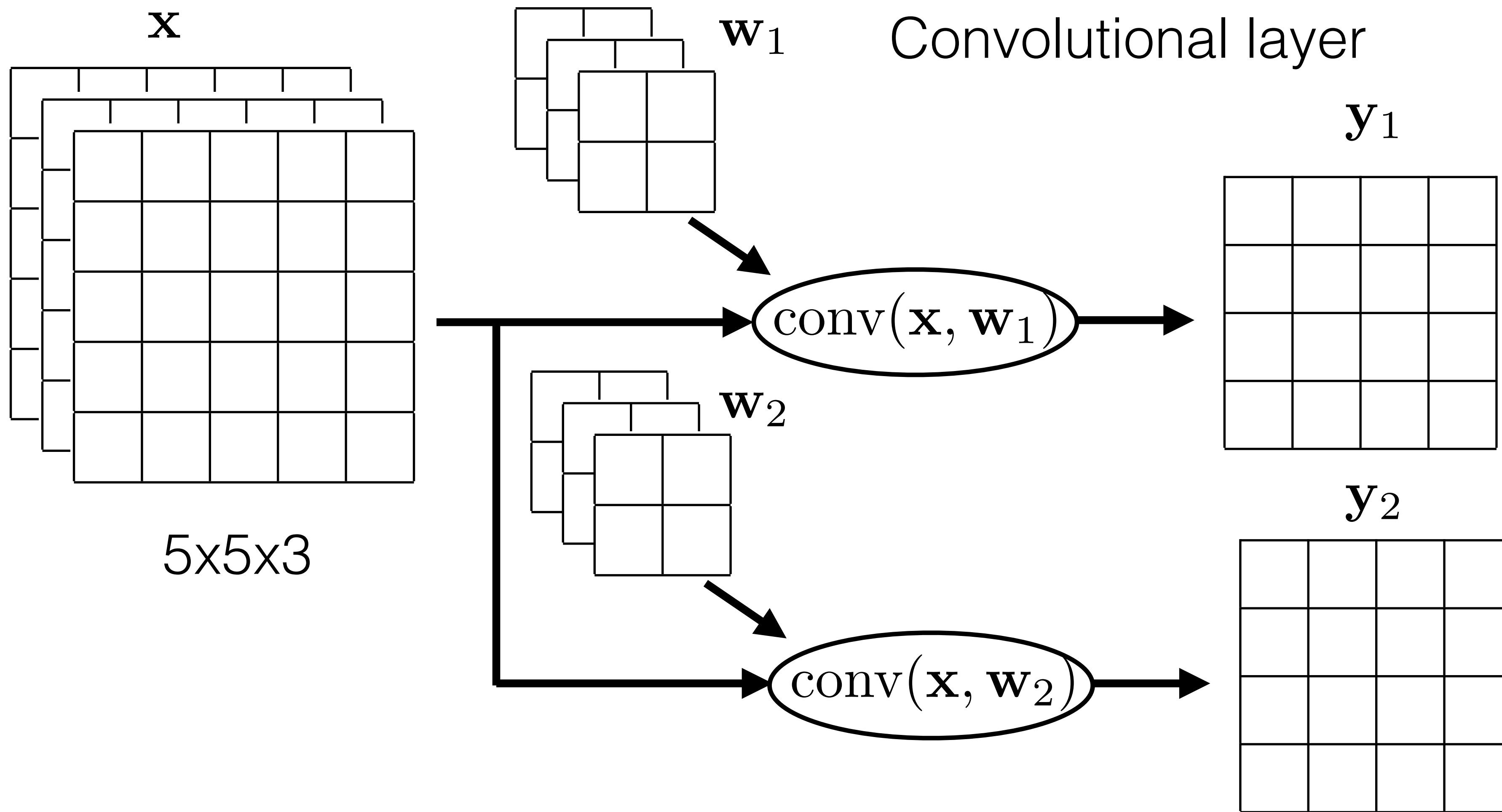








```
# initialise
import torch.nn as nn
# define 2D convolutional layer
first_layer = nn.Conv2d(in_channels=3, out_channels=2, kernel_size=2
                        stride=1, padding=1)
```

```
# initialise
```

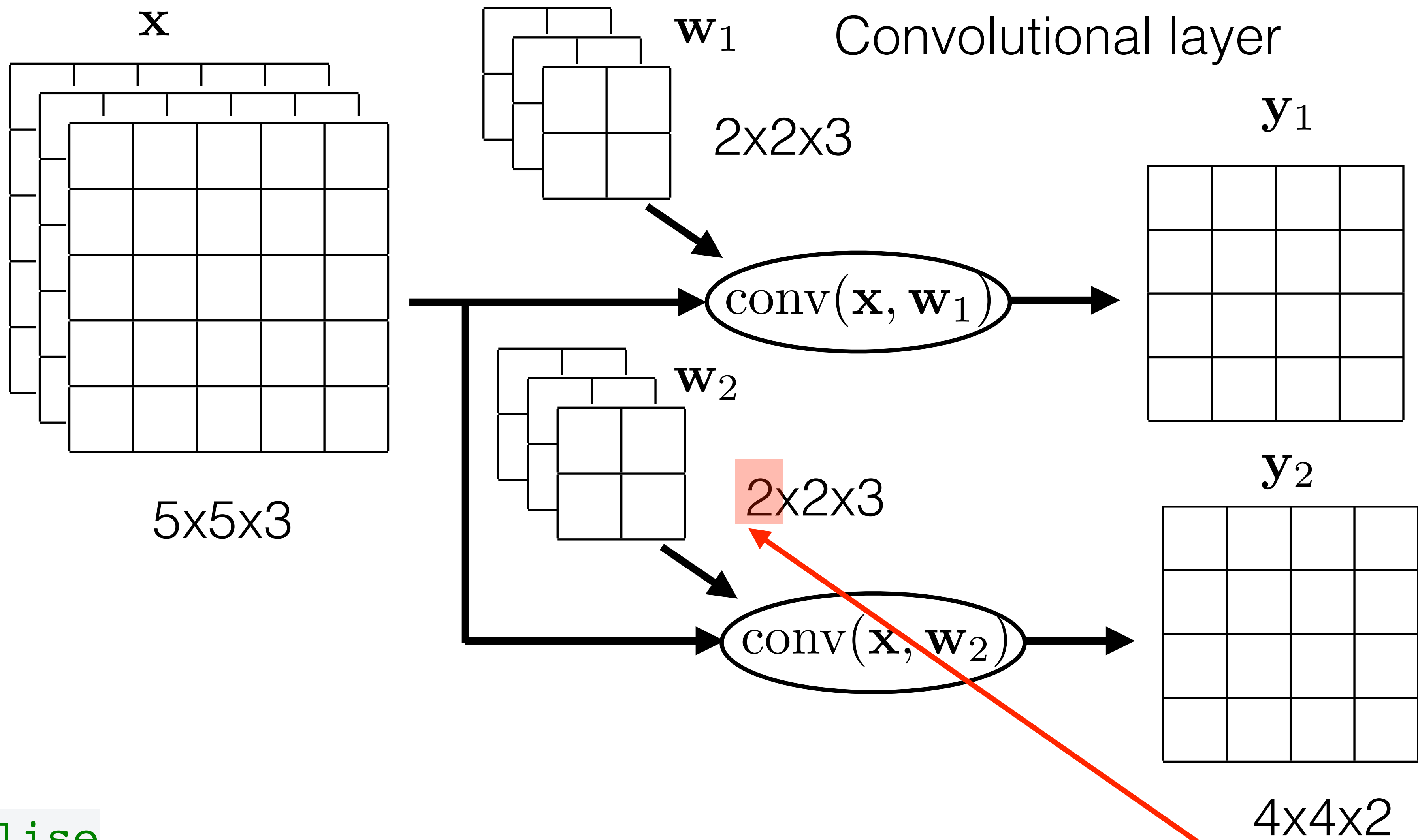
```
import torch.nn as nn
```

```
# define 2D convolutional layer
```

```
first_layer = nn.Conv2d(in_channels=3, out_channels=2, kernel_size=2  
                        stride=1, padding=1)
```

also number
of kernels

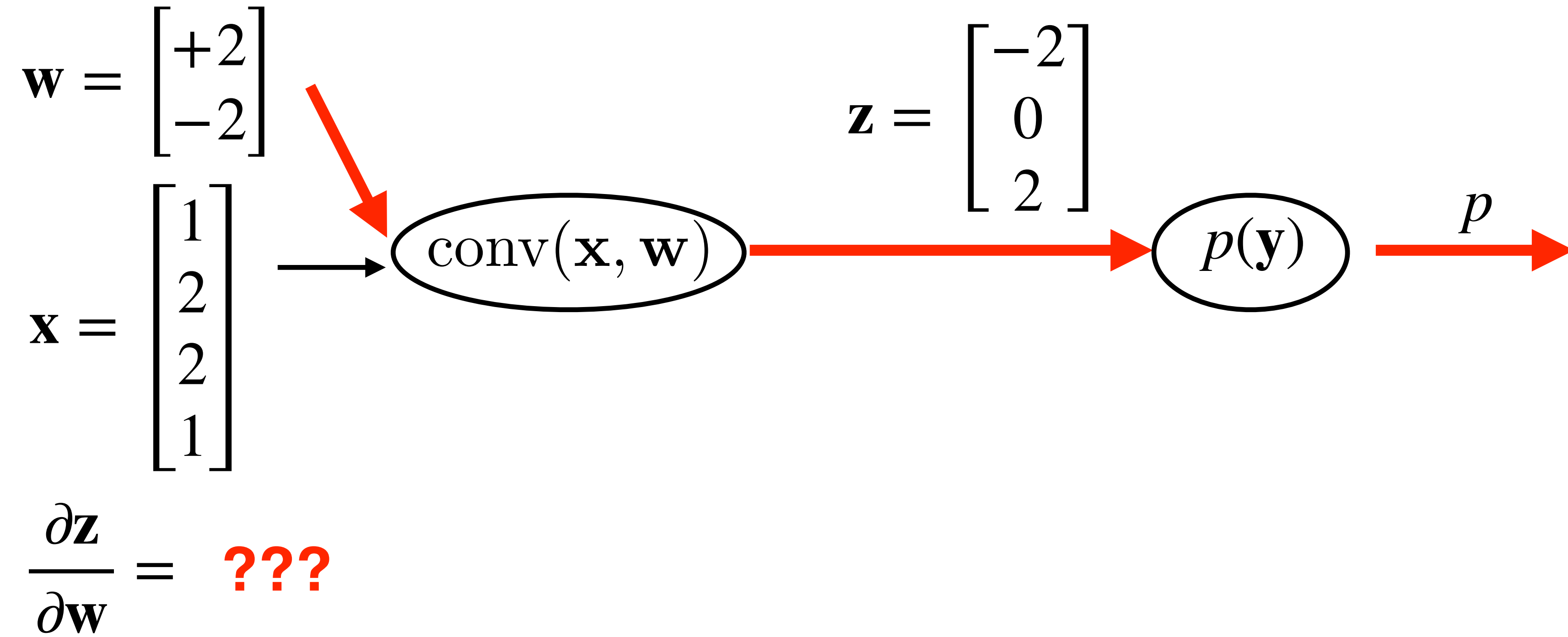
4x4x2



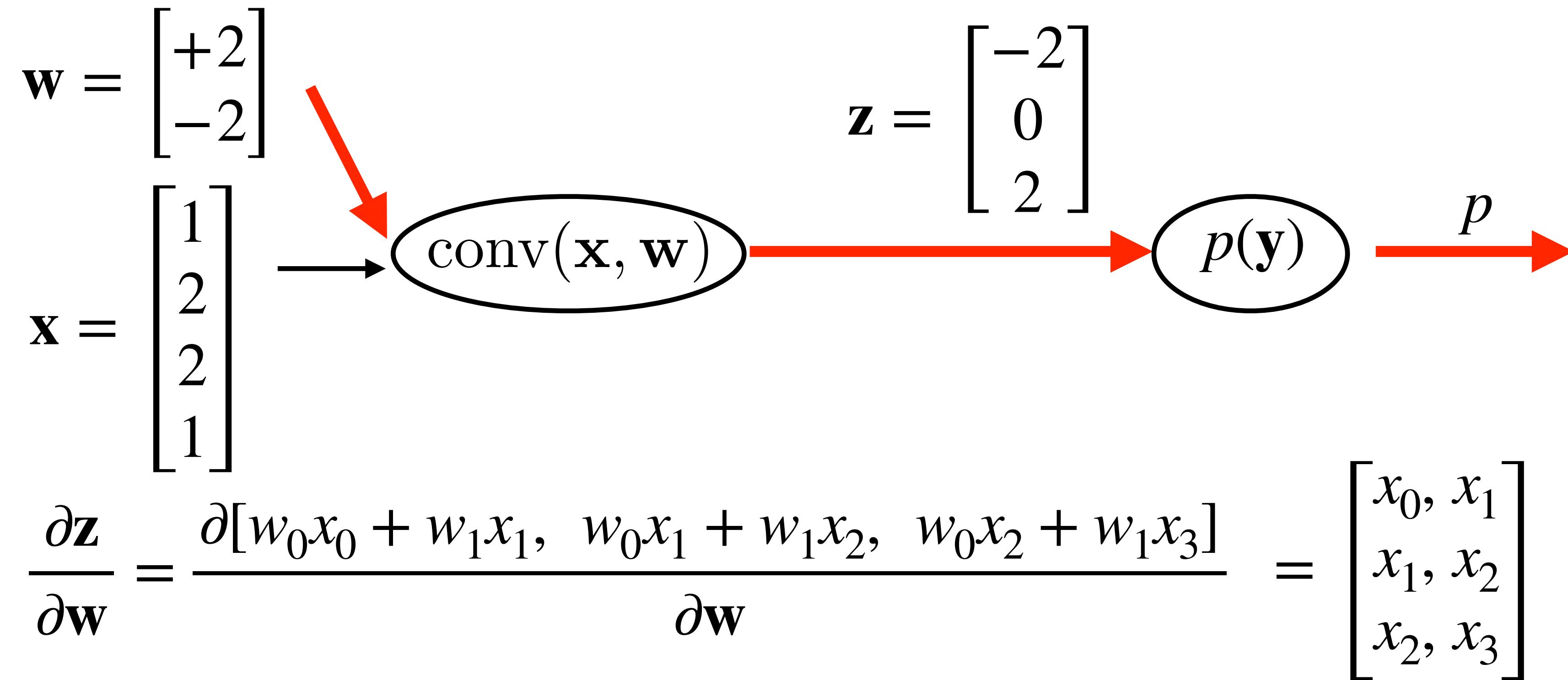
```
# initialise
import torch.nn as nn
# define 2D convolutional layer
first_layer = nn.Conv2d(in_channels=3, out_channels=2, kernel_size=2
                        stride=1, padding=1)
```

Let's train the convolutional network !

Example: 1D convolution backward pass



Example: 1D convolution backward pass

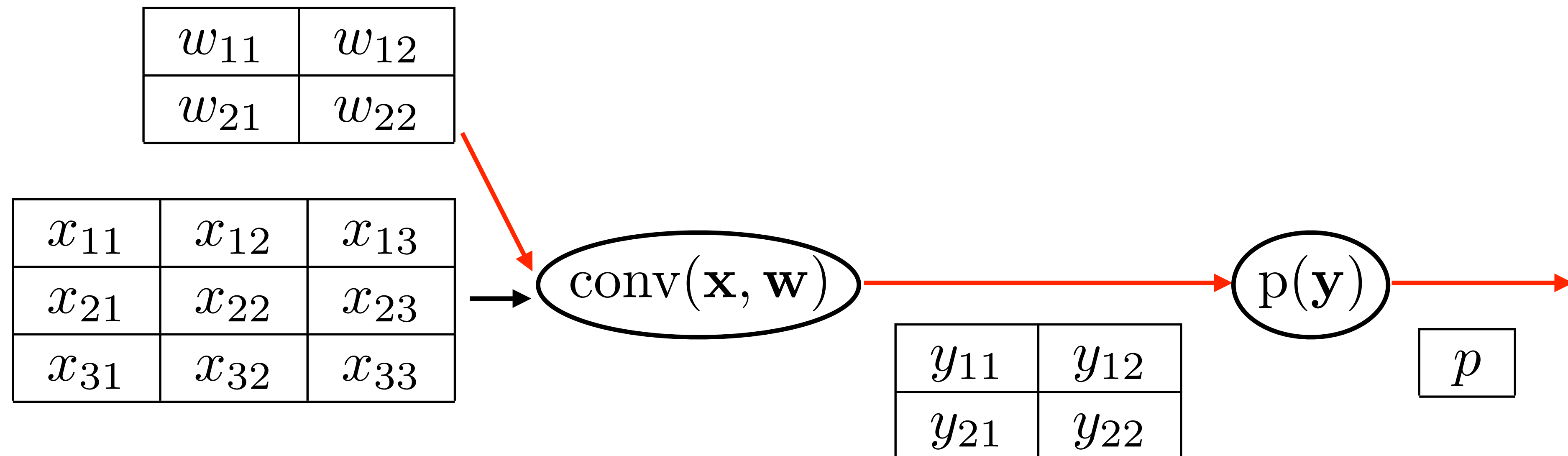


```
def vjp_conv_w(v, x):
```

```
    return v⊤ ·  $\frac{\partial \mathbf{z}}{\partial \mathbf{w}}$  =  $[v_0, v_1, v_2] \cdot \begin{bmatrix} x_0, x_1 \\ x_1, x_2 \\ x_2, x_3 \end{bmatrix}$  =  $[v_0 x_0 + v_1 x_1 + v_2 x_2, v_0 x_1 + v_1 x_2 + v_2 x_3]$   

    = conv(x, v)
```

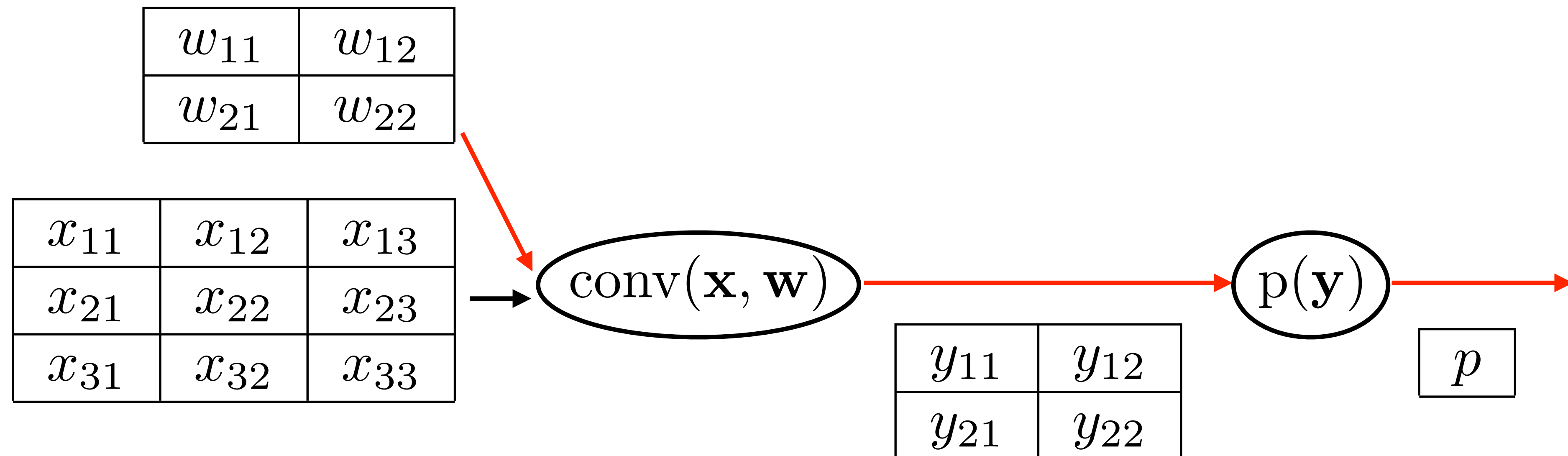
2D Convolution backward pass



2D Convolution backward pass

$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

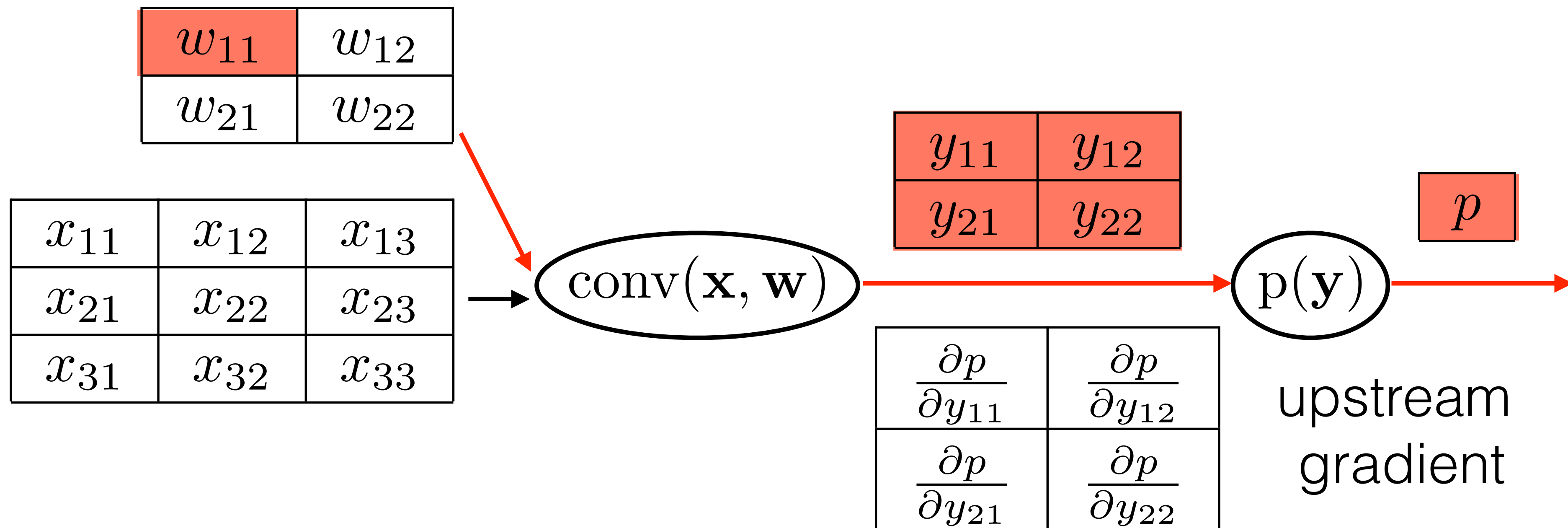
 $=?$



$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

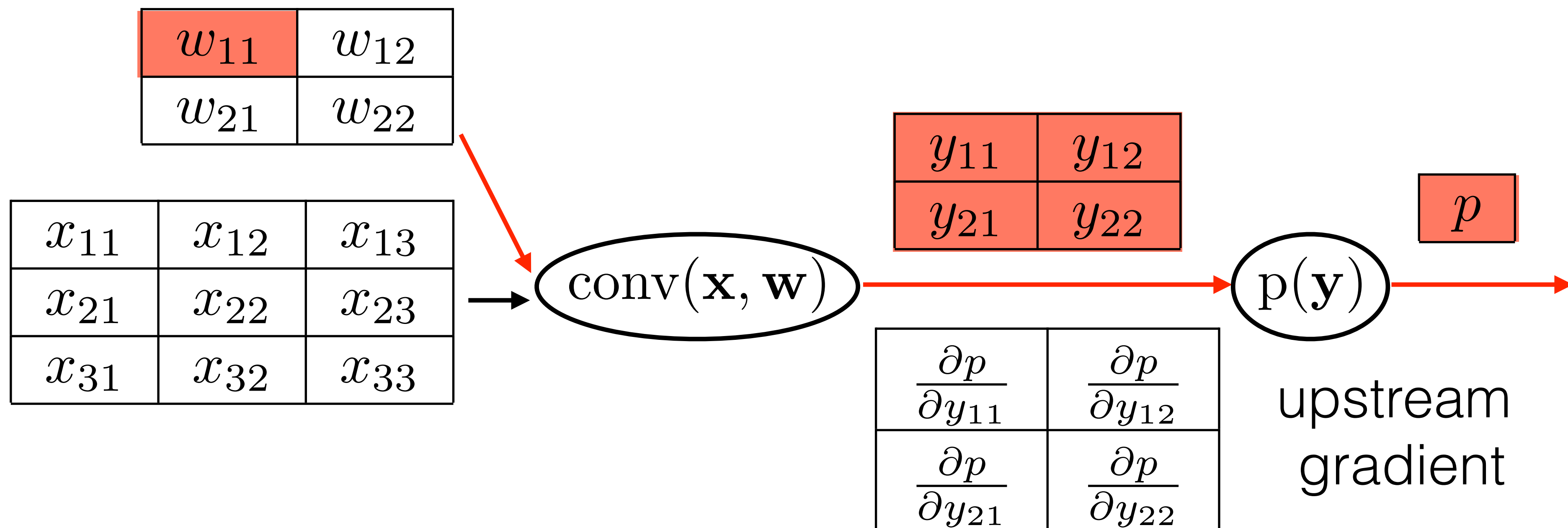
$$\frac{\partial p}{\partial w_{11}} = ?$$



$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}} \frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial p}{\partial y_{12}} \frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial p}{\partial y_{21}} \frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial p}{\partial y_{22}} \frac{\partial y_{22}}{\partial w_{11}}$$

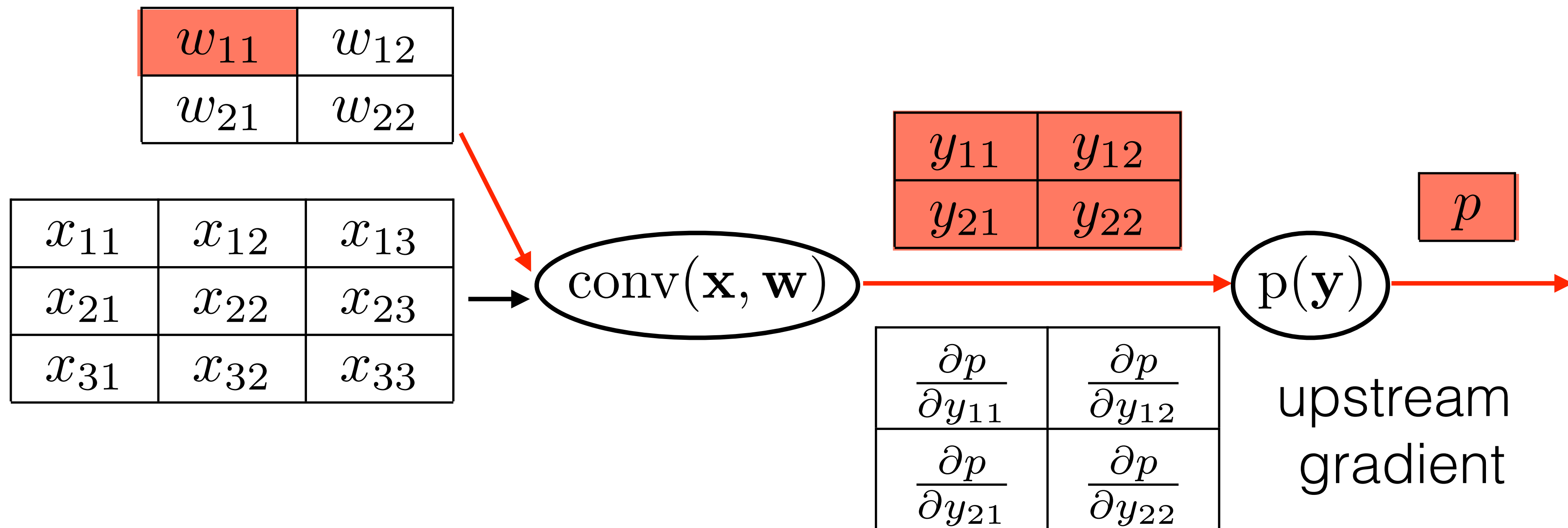


$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}} \frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial p}{\partial y_{12}} \frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial p}{\partial y_{21}} \frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial p}{\partial y_{22}} \frac{\partial y_{22}}{\partial w_{11}}$$

$$\frac{\partial y_{11}}{\partial w_{11}} = ?$$

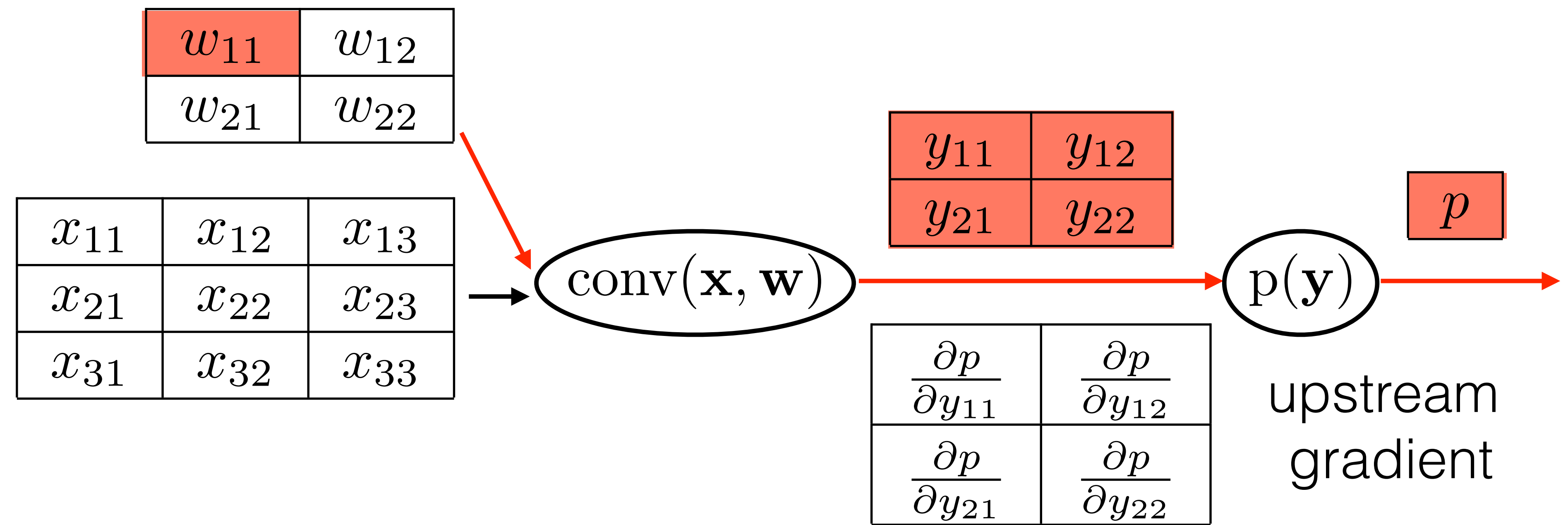


$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}} \frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial p}{\partial y_{12}} \frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial p}{\partial y_{21}} \frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial p}{\partial y_{22}} \frac{\partial y_{22}}{\partial w_{11}}$$

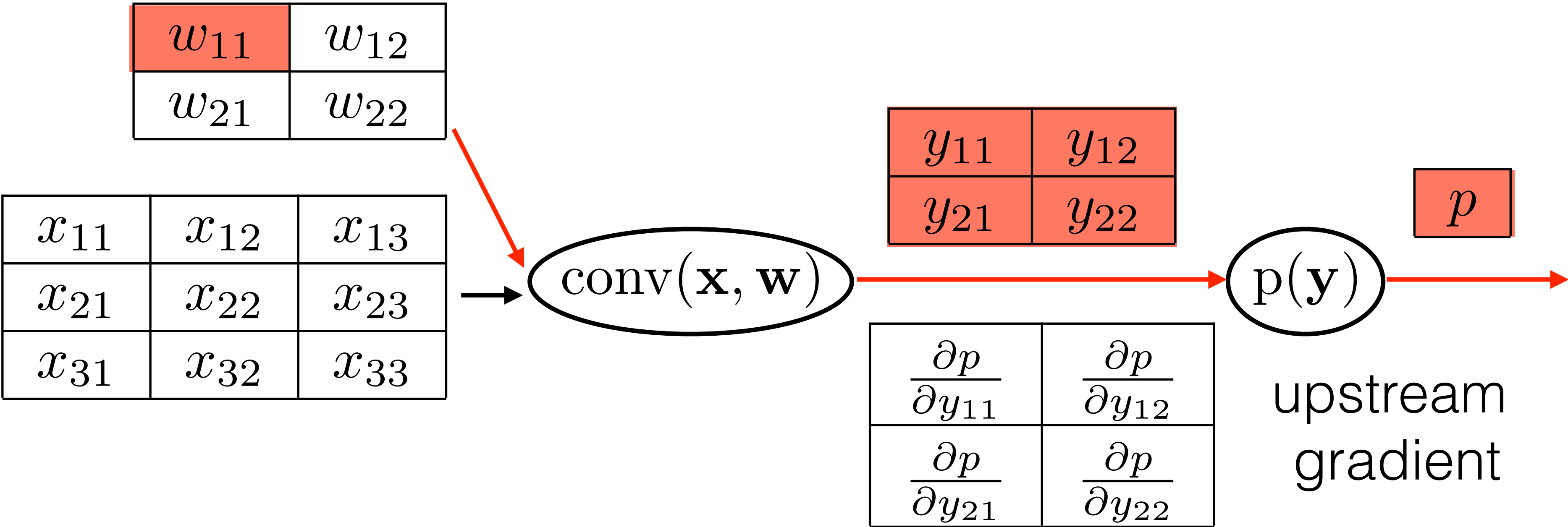
$$\frac{\partial y_{11}}{\partial w_{11}} = \frac{\partial (w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22})}{\partial w_{11}} = x_{11}$$



$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}}x_{11} + \frac{\partial p}{\partial y_{12}}x_{12} + \frac{\partial p}{\partial y_{21}}x_{21} + \frac{\partial p}{\partial y_{22}}x_{22}$$

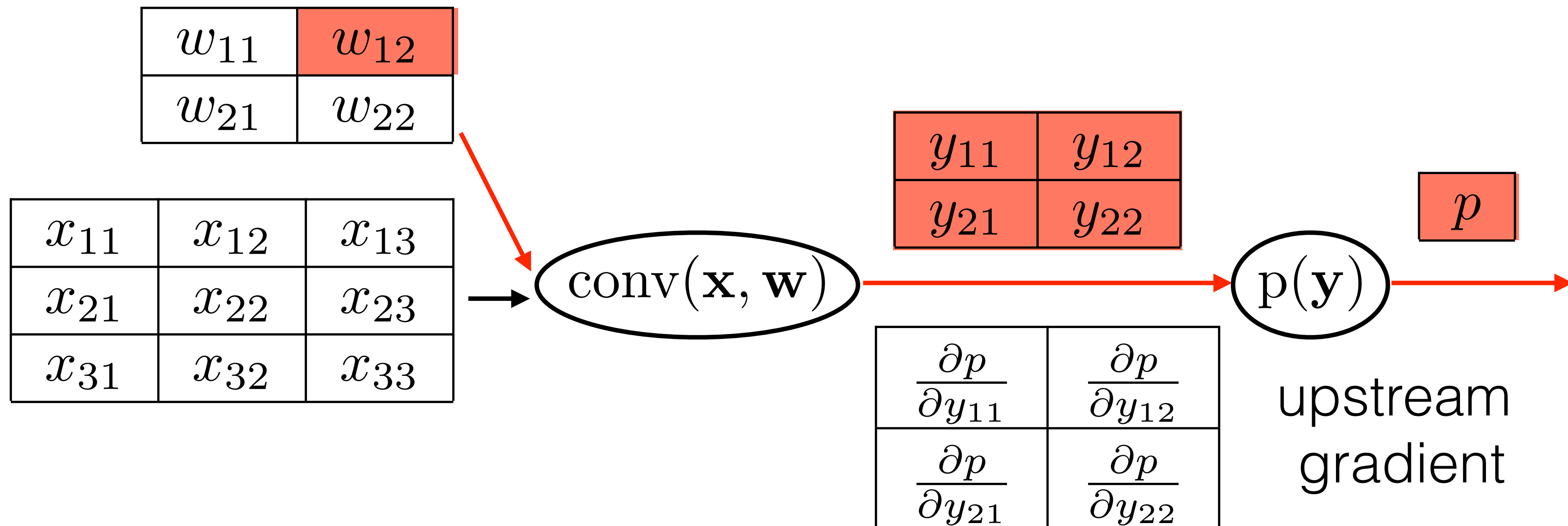


$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}} x_{11} + \frac{\partial p}{\partial y_{12}} x_{12} + \frac{\partial p}{\partial y_{21}} x_{21} + \frac{\partial p}{\partial y_{22}} x_{22}$$

$$\frac{\partial p}{\partial w_{12}} = \text{?}$$

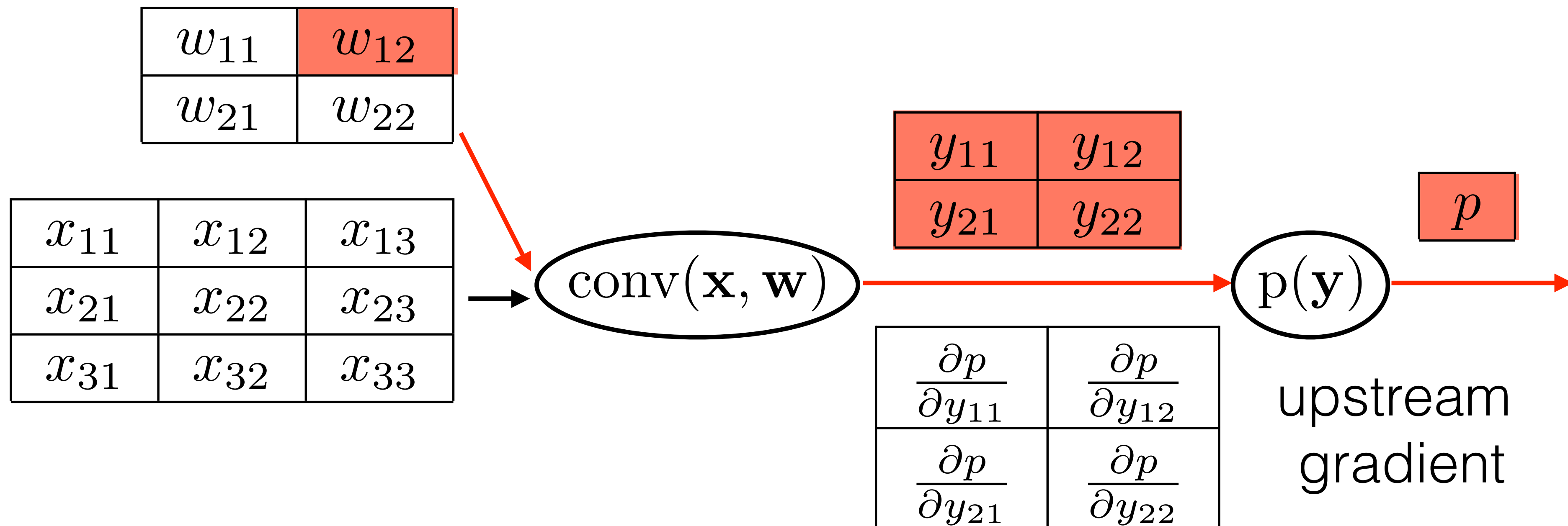


$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}} x_{11} + \frac{\partial p}{\partial y_{12}} x_{12} + \frac{\partial p}{\partial y_{21}} x_{21} + \frac{\partial p}{\partial y_{22}} x_{22}$$

$$\frac{\partial p}{\partial w_{12}} = \frac{\partial p}{\partial y_{11}} x_{12} + \frac{\partial p}{\partial y_{12}} x_{13} + \frac{\partial p}{\partial y_{21}} x_{22} + \frac{\partial p}{\partial y_{22}} x_{23}$$



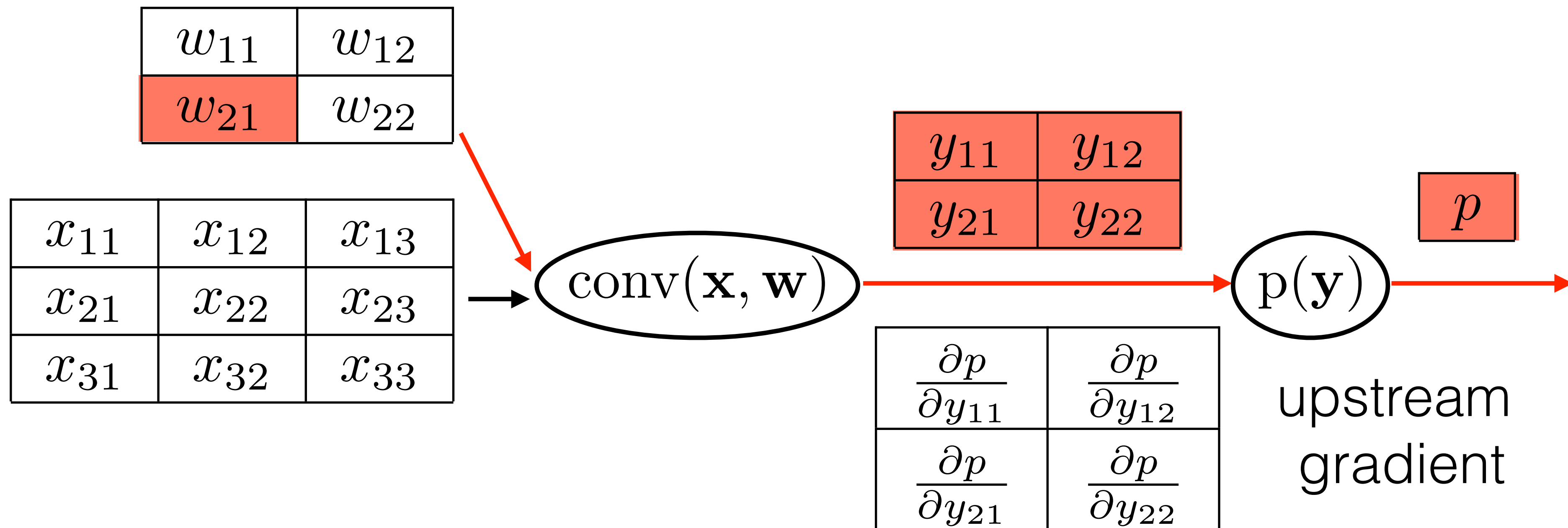
$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}} x_{11} + \frac{\partial p}{\partial y_{12}} x_{12} + \frac{\partial p}{\partial y_{21}} x_{21} + \frac{\partial p}{\partial y_{22}} x_{22}$$

$$\frac{\partial p}{\partial w_{12}} = \frac{\partial p}{\partial y_{11}} x_{12} + \frac{\partial p}{\partial y_{12}} x_{13} + \frac{\partial p}{\partial y_{21}} x_{22} + \frac{\partial p}{\partial y_{22}} x_{23}$$

$$\frac{\partial p}{\partial w_{21}} = \frac{\partial p}{\partial y_{11}} x_{21} + \frac{\partial p}{\partial y_{12}} x_{22} + \frac{\partial p}{\partial y_{21}} x_{31} + \frac{\partial p}{\partial y_{22}} x_{32}$$



$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

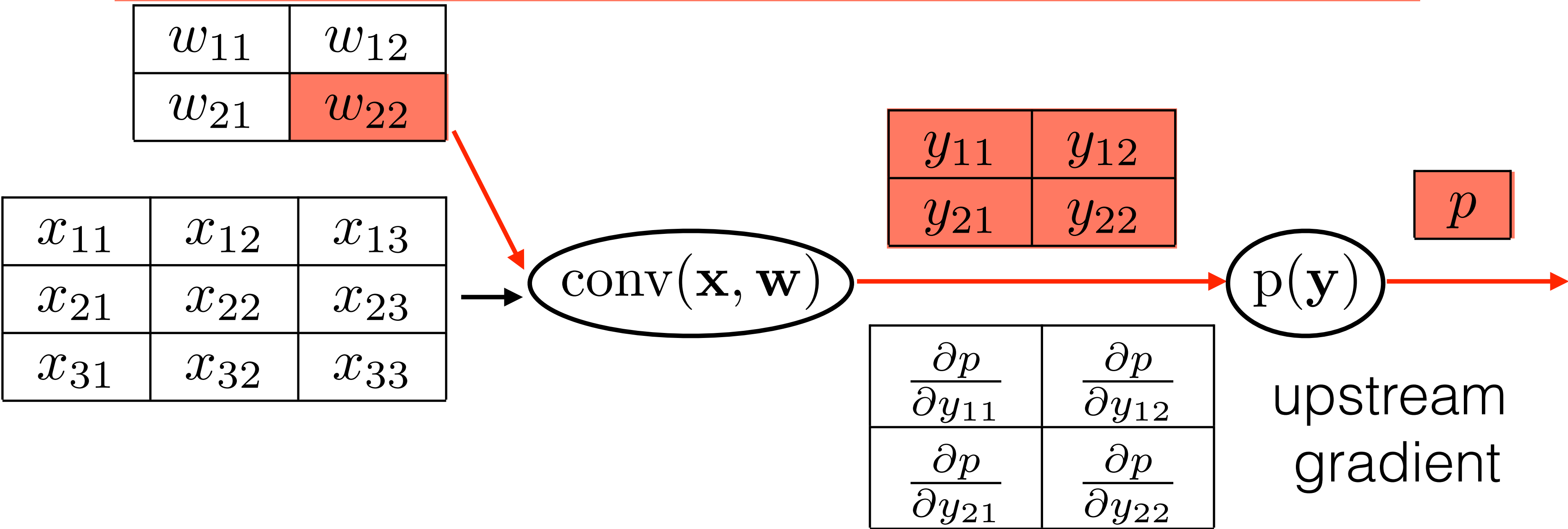
2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}}x_{11} + \frac{\partial p}{\partial y_{12}}x_{12} + \frac{\partial p}{\partial y_{21}}x_{21} + \frac{\partial p}{\partial y_{22}}x_{22}$$

$$\frac{\partial p}{\partial w_{12}} = \frac{\partial p}{\partial y_{11}}x_{12} + \frac{\partial p}{\partial y_{12}}x_{13} + \frac{\partial p}{\partial y_{21}}x_{22} + \frac{\partial p}{\partial y_{22}}x_{23}$$

$$\frac{\partial p}{\partial w_{21}} = \frac{\partial p}{\partial y_{11}}x_{21} + \frac{\partial p}{\partial y_{12}}x_{22} + \frac{\partial p}{\partial y_{21}}x_{31} + \frac{\partial p}{\partial y_{22}}x_{32}$$

$$\frac{\partial p}{\partial w_{22}} = \frac{\partial p}{\partial y_{11}}x_{22} + \frac{\partial p}{\partial y_{12}}x_{23} + \frac{\partial p}{\partial y_{21}}x_{32} + \frac{\partial p}{\partial y_{22}}x_{33}$$



$\frac{\partial p}{\partial w_{11}}$	$\frac{\partial p}{\partial w_{12}}$
$\frac{\partial p}{\partial w_{21}}$	$\frac{\partial p}{\partial w_{22}}$

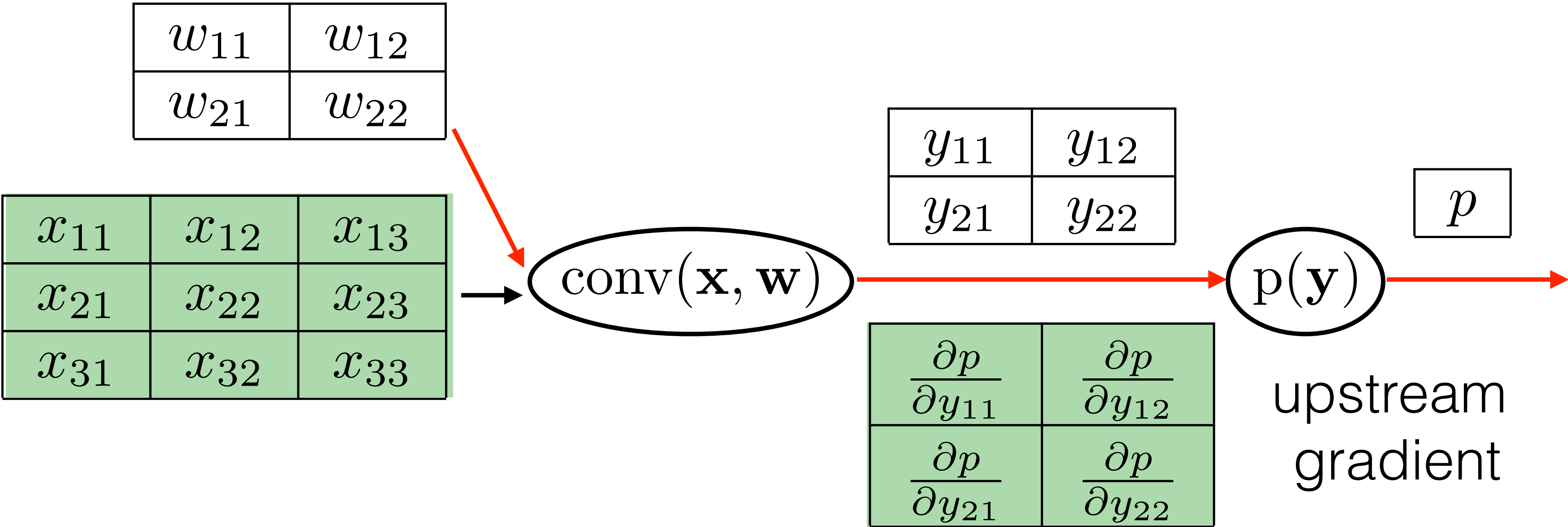
2D Convolution backward pass

$$\frac{\partial p}{\partial w_{11}} = \frac{\partial p}{\partial y_{11}} x_{11} + \frac{\partial p}{\partial y_{12}} x_{12} + \frac{\partial p}{\partial y_{21}} x_{21} + \frac{\partial p}{\partial y_{22}} x_{22}$$

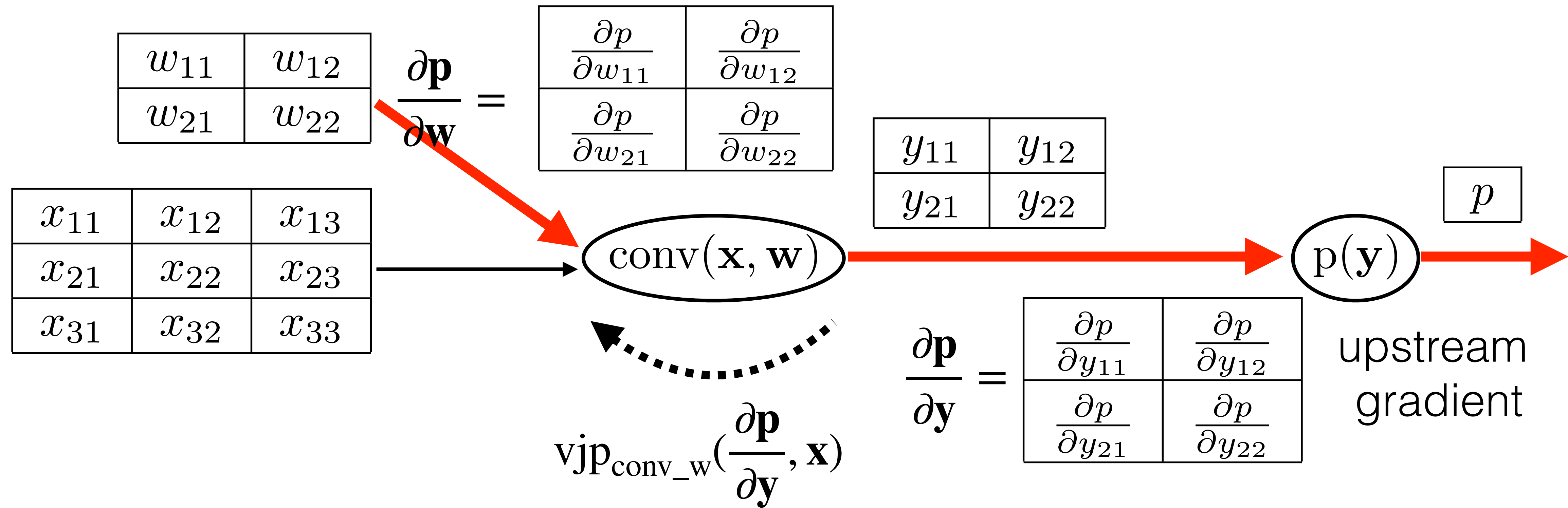
$$\frac{\partial p}{\partial w_{12}} = \frac{\partial p}{\partial y_{11}} x_{12} + \frac{\partial p}{\partial y_{12}} x_{13} + \frac{\partial p}{\partial y_{21}} x_{22} + \frac{\partial p}{\partial y_{22}} x_{23}$$

$$\frac{\partial p}{\partial w_{21}} = \frac{\partial p}{\partial y_{11}} x_{21} + \frac{\partial p}{\partial y_{12}} x_{22} + \frac{\partial p}{\partial y_{21}} x_{31} + \frac{\partial p}{\partial y_{22}} x_{32}$$

$$\frac{\partial p}{\partial w_{22}} = \frac{\partial p}{\partial y_{11}} x_{22} + \frac{\partial p}{\partial y_{12}} x_{23} + \frac{\partial p}{\partial y_{21}} x_{32} + \frac{\partial p}{\partial y_{22}} x_{33}$$



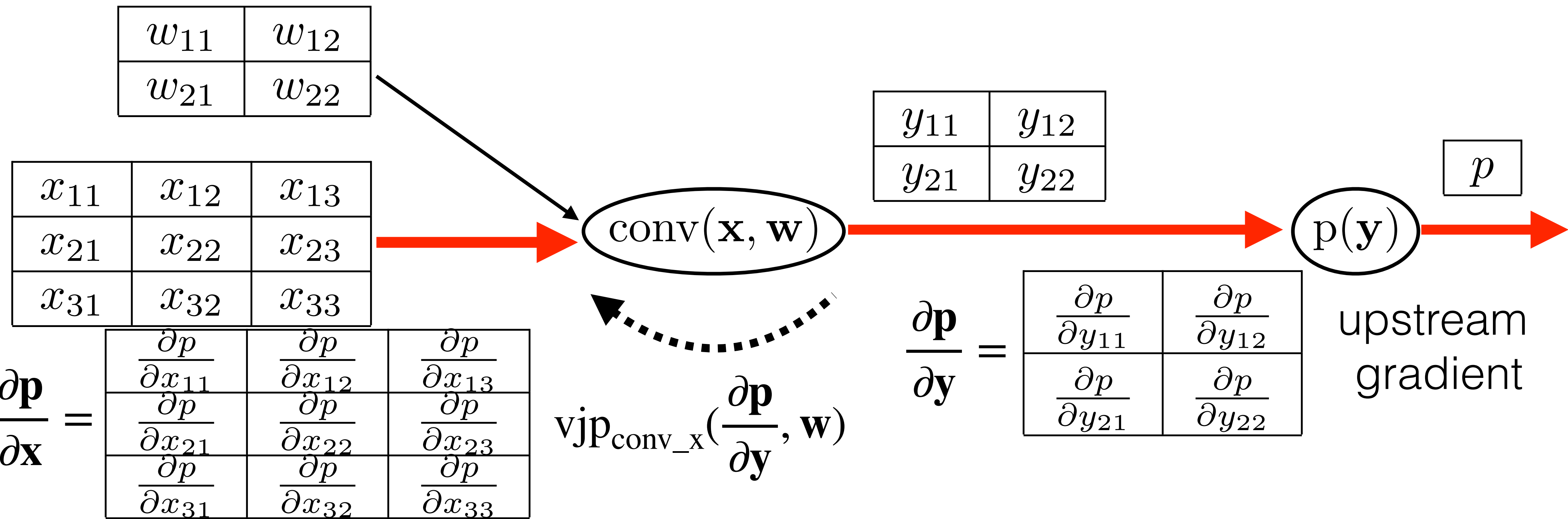
Convolution backward pass wrt weights



$$\text{vjp}_{\text{conv}_w}\left(\frac{\partial p}{\partial y}, \mathbf{x}\right) = \begin{bmatrix} \frac{\partial p}{\partial y_{11}} & \frac{\partial p}{\partial y_{12}} \\ \frac{\partial p}{\partial y_{21}} & \frac{\partial p}{\partial y_{22}} \end{bmatrix} = \text{conv} \left(\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, \begin{bmatrix} \frac{\partial p}{\partial y_{11}} & \frac{\partial p}{\partial y_{12}} \\ \frac{\partial p}{\partial y_{21}} & \frac{\partial p}{\partial y_{22}} \end{bmatrix} \right)$$

- Backpropagation of convolutional layer wrt weights is defined as:
“convolution of input feature map with upstream gradient”

Convolution backward pass wrt feature map



$$\text{vjp}_{\text{conv}_x}(\frac{\partial p}{\partial y}) = \begin{bmatrix} \frac{\partial p}{\partial x_{11}} & \frac{\partial p}{\partial x_{12}} & \frac{\partial p}{\partial x_{13}} \\ \frac{\partial p}{\partial x_{21}} & \frac{\partial p}{\partial x_{22}} & \frac{\partial p}{\partial x_{23}} \\ \frac{\partial p}{\partial x_{31}} & \frac{\partial p}{\partial x_{32}} & \frac{\partial p}{\partial x_{33}} \end{bmatrix} = \text{conv} \left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\partial p}{\partial y_{11}} & \frac{\partial p}{\partial y_{12}} & 0 \\ 0 & \frac{\partial p}{\partial y_{21}} & \frac{\partial p}{\partial y_{22}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{bmatrix} \right)$$

- Backpropagation of convolutional layer wrt input feature map is defined as:
“**convolution of padded upstream gradient with mirrored weights**”

Convolution backward pass wrt input feature map

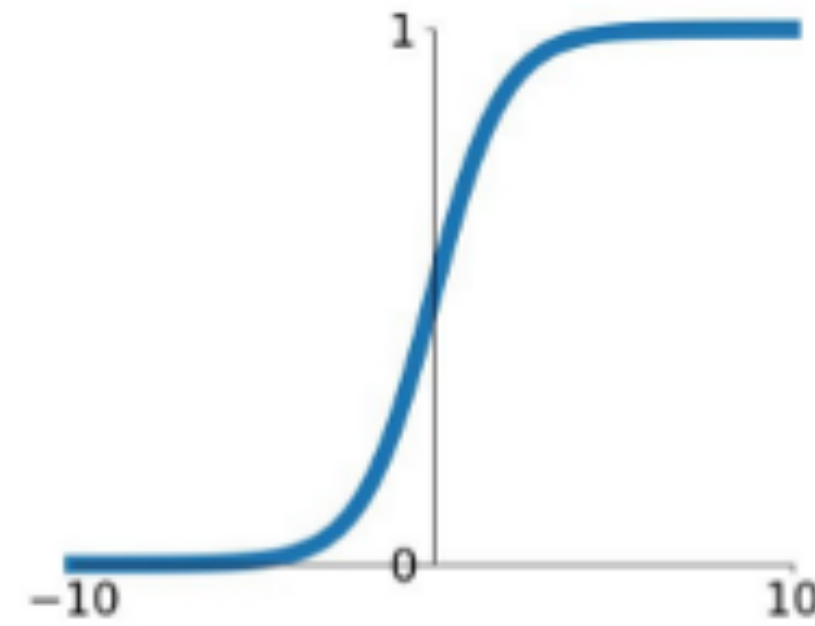
Very important property of convolutional layer is:

Backpropagation is also convolution !!!

Activation functions

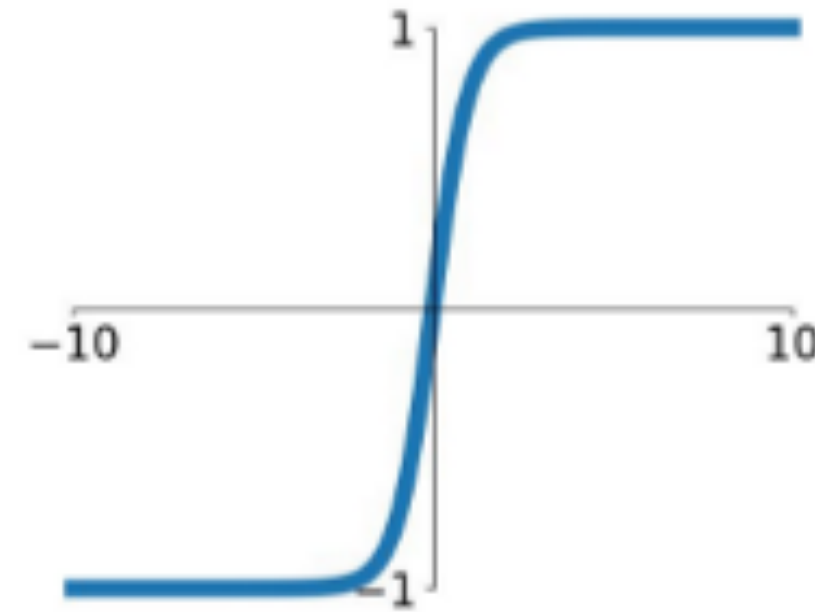
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



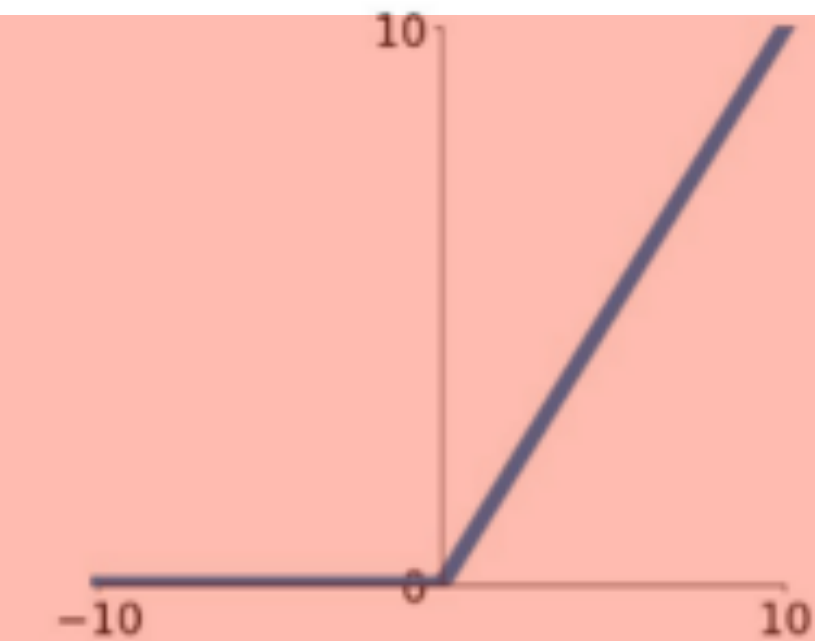
tanh

$$\tanh(x)$$



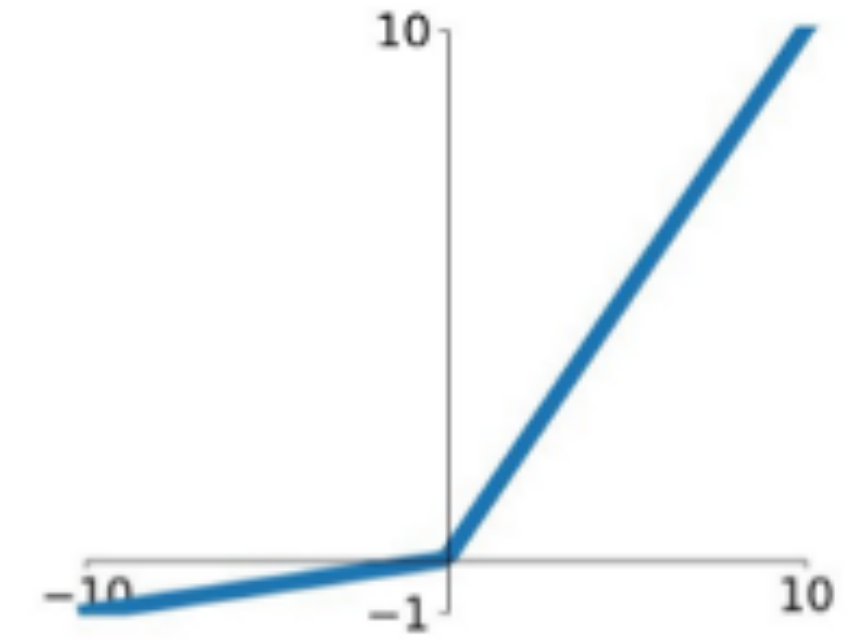
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

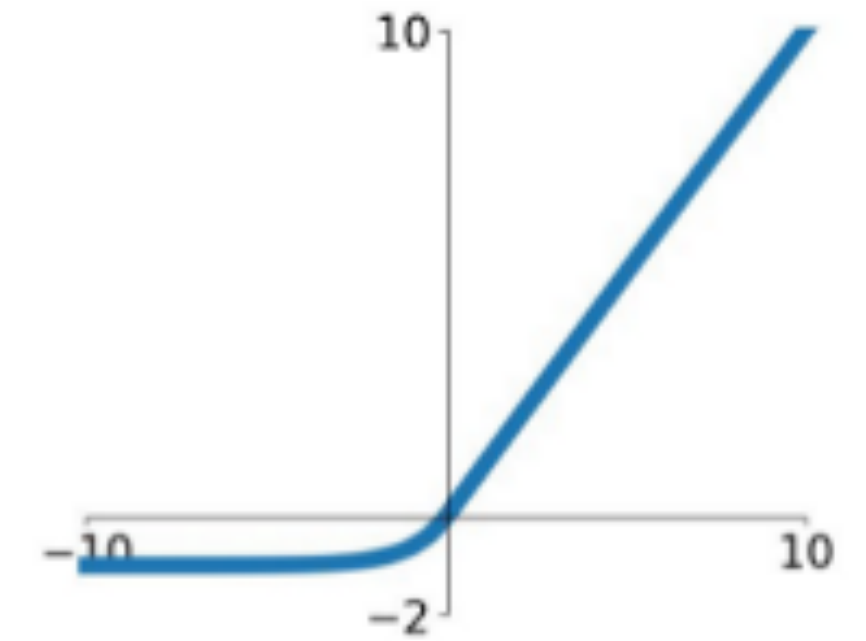


Maxout

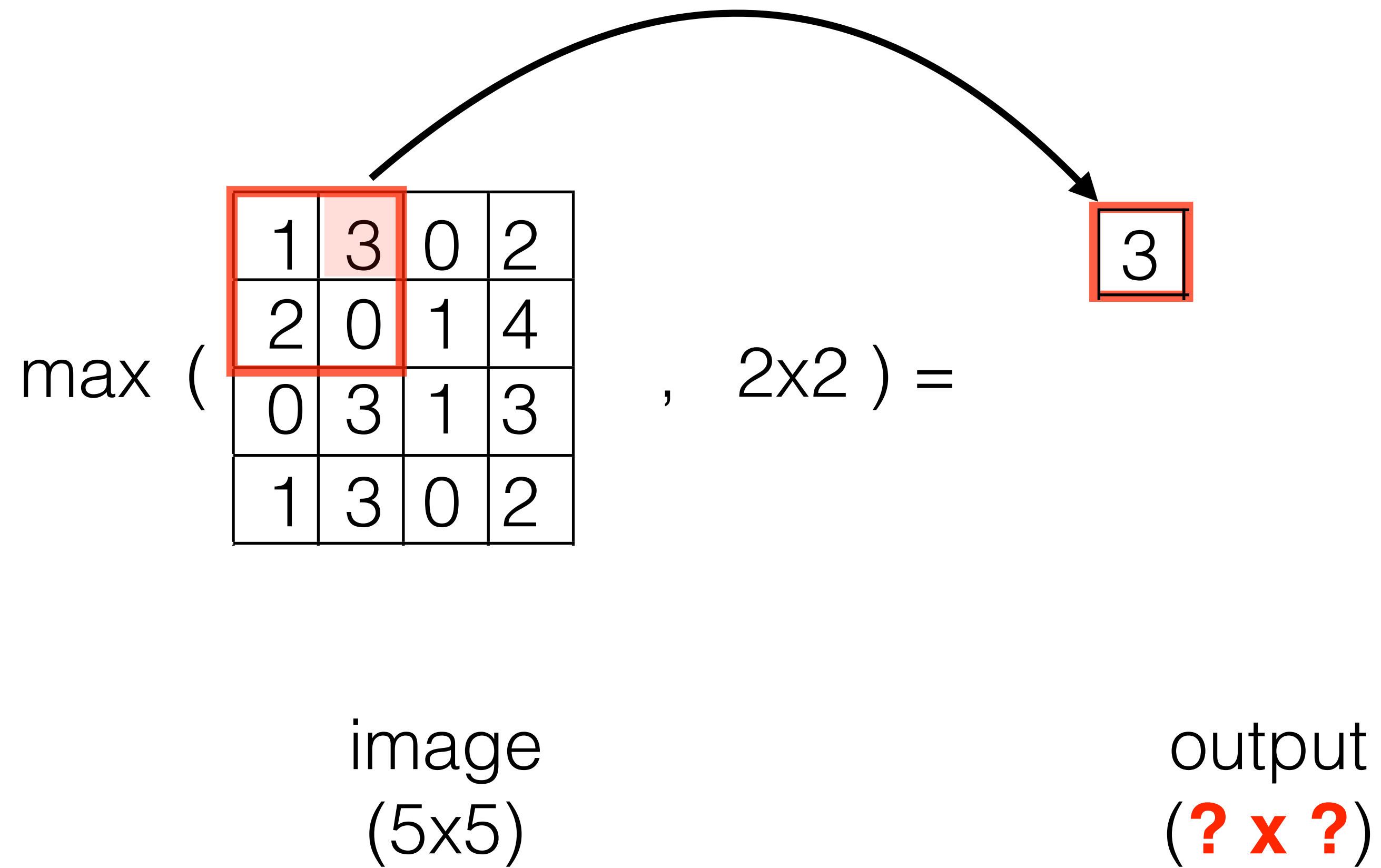
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

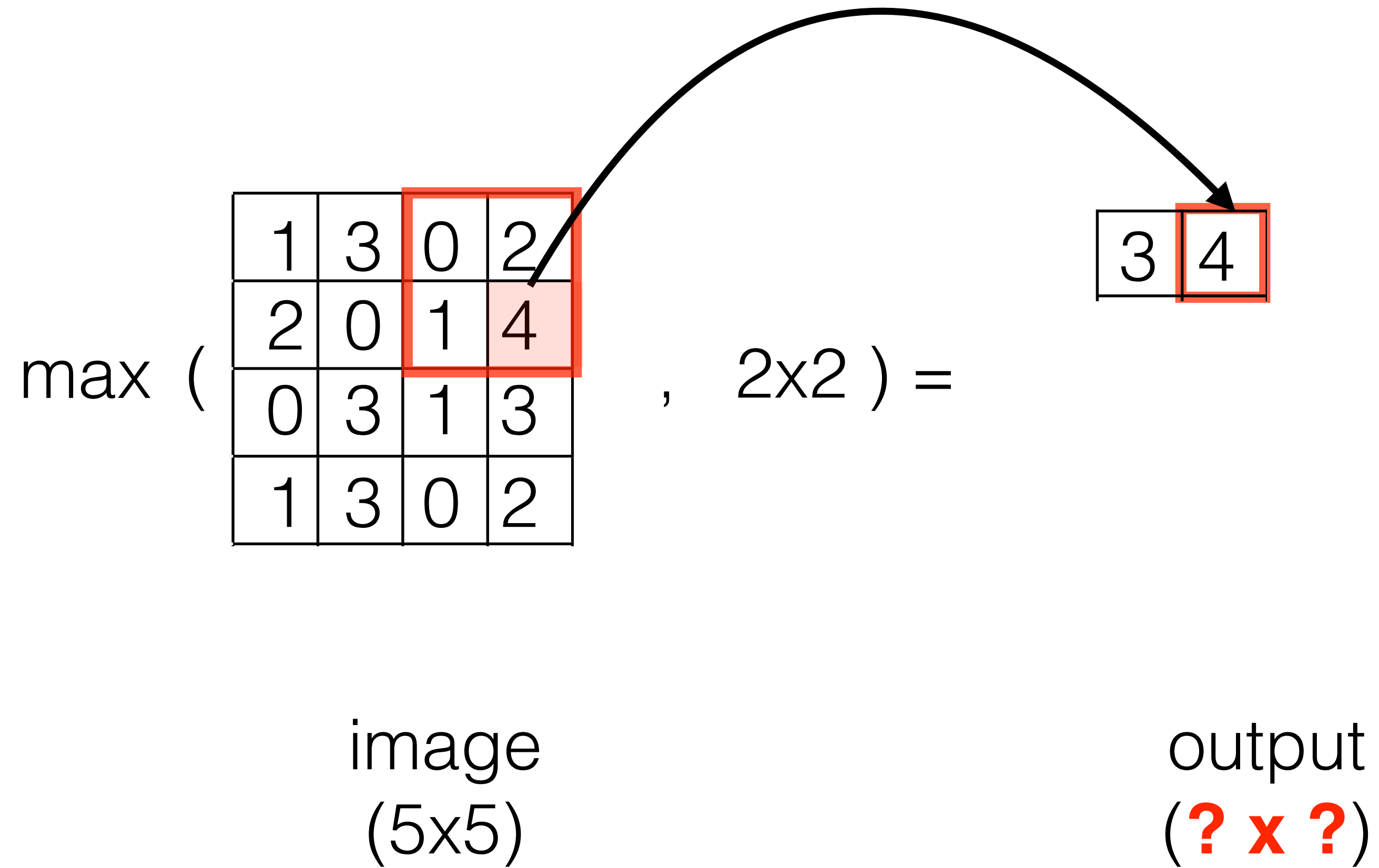
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



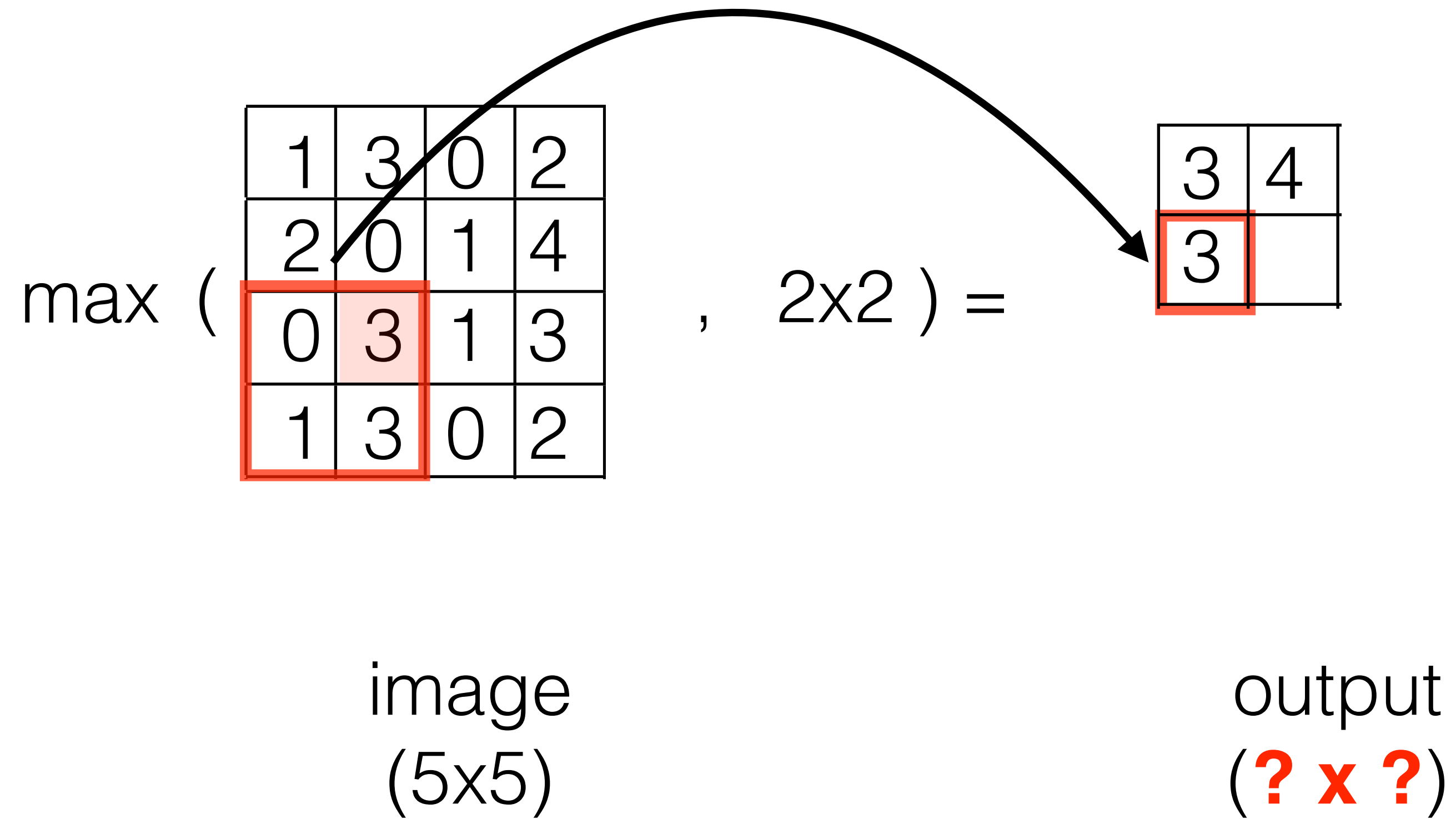
Max-pooling



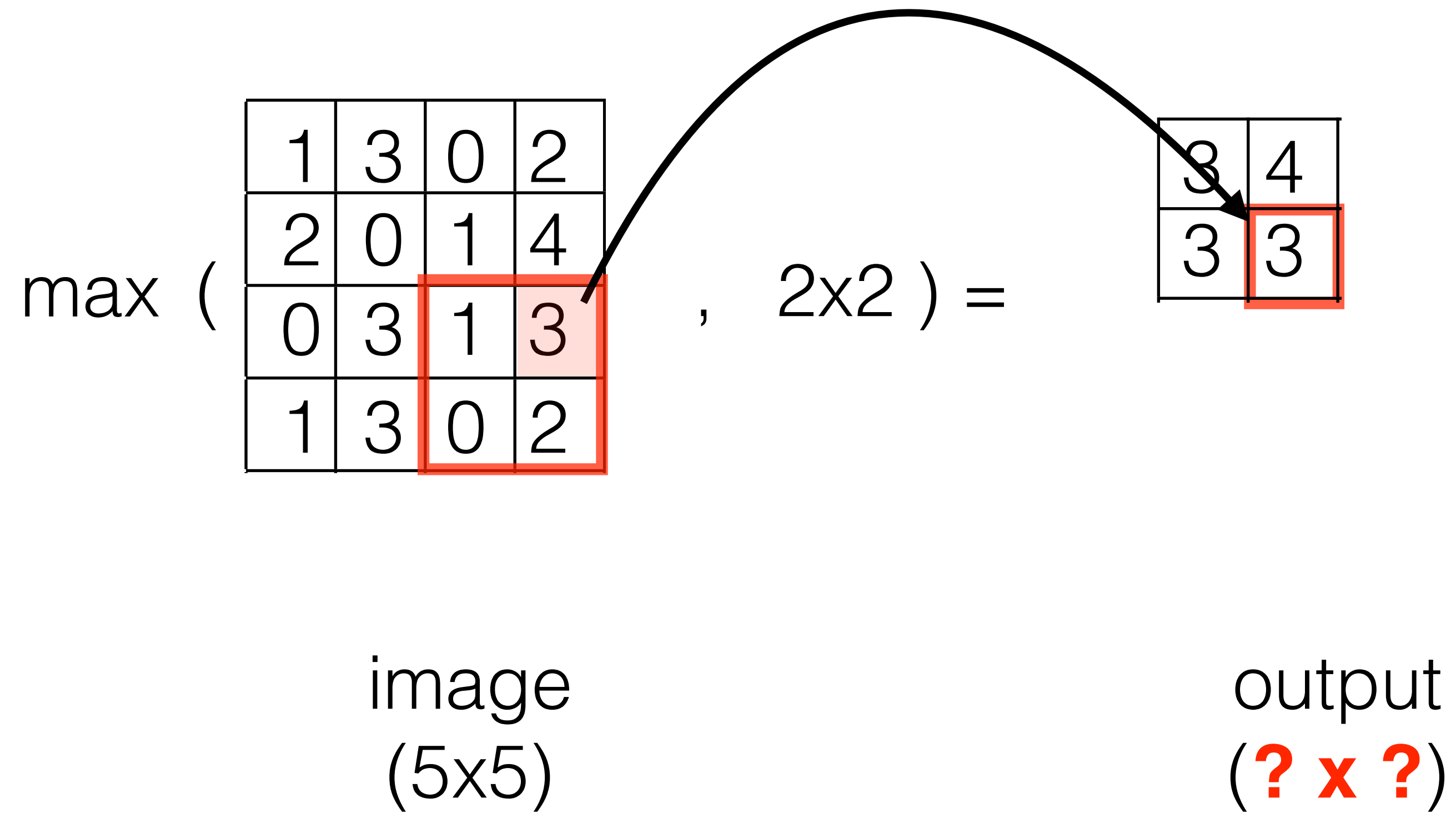
Max-pooling



Max-pooling

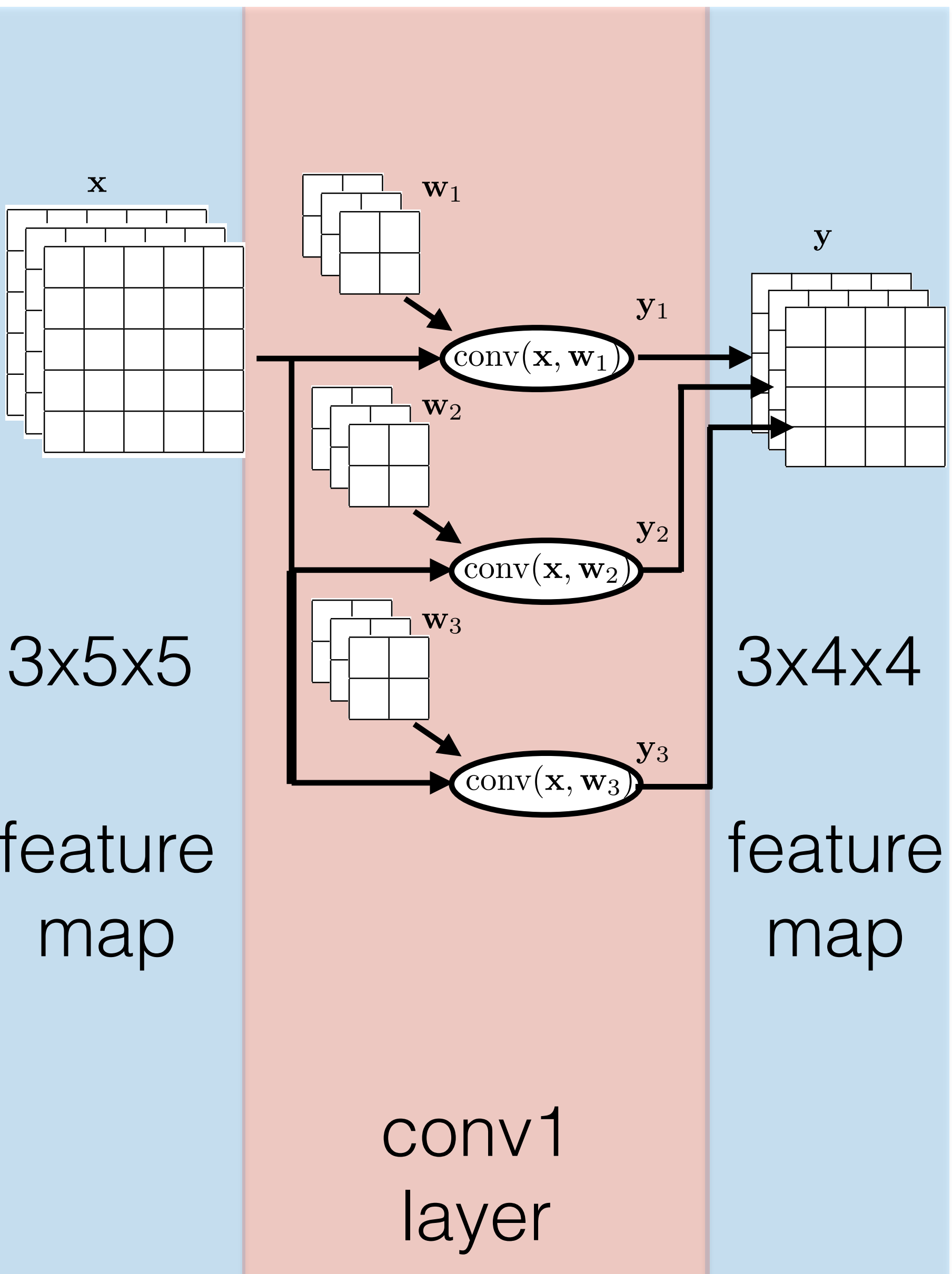


Max-pooling



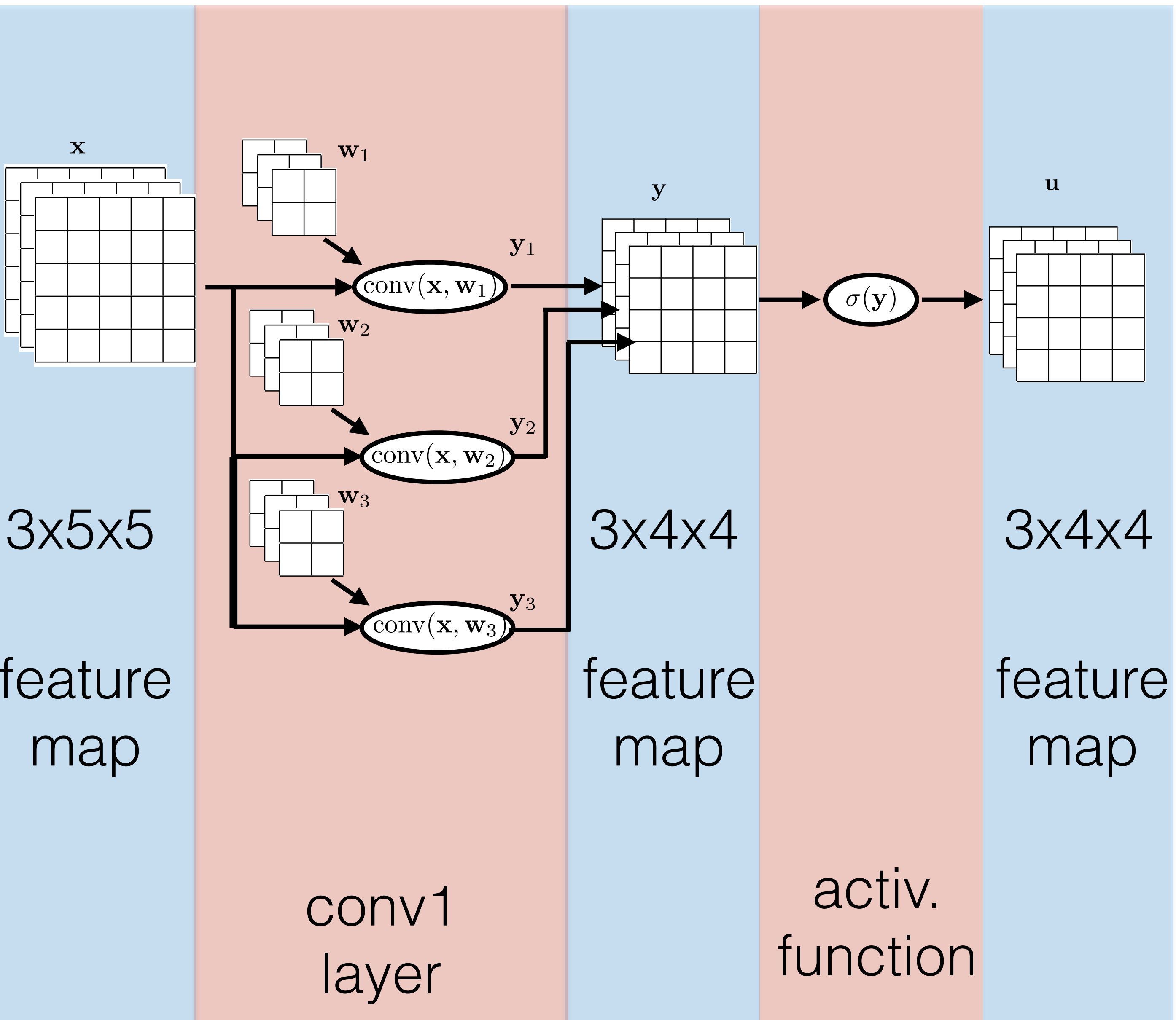
Learning of a simple convolutional network

input 3D tensor: channels x height x width



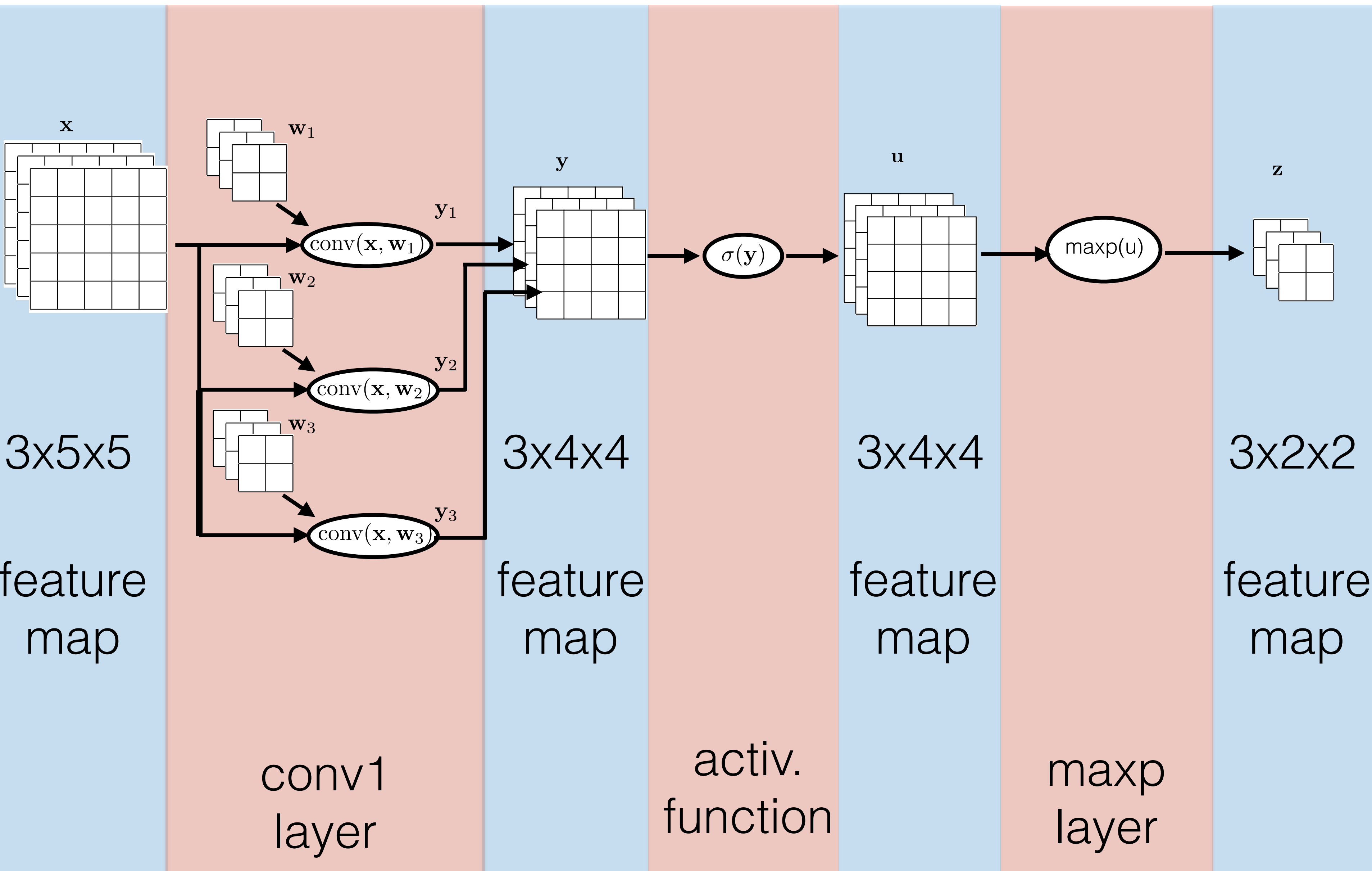
Learning of a simple convolutional network

input 3D tensor: channels x height x width



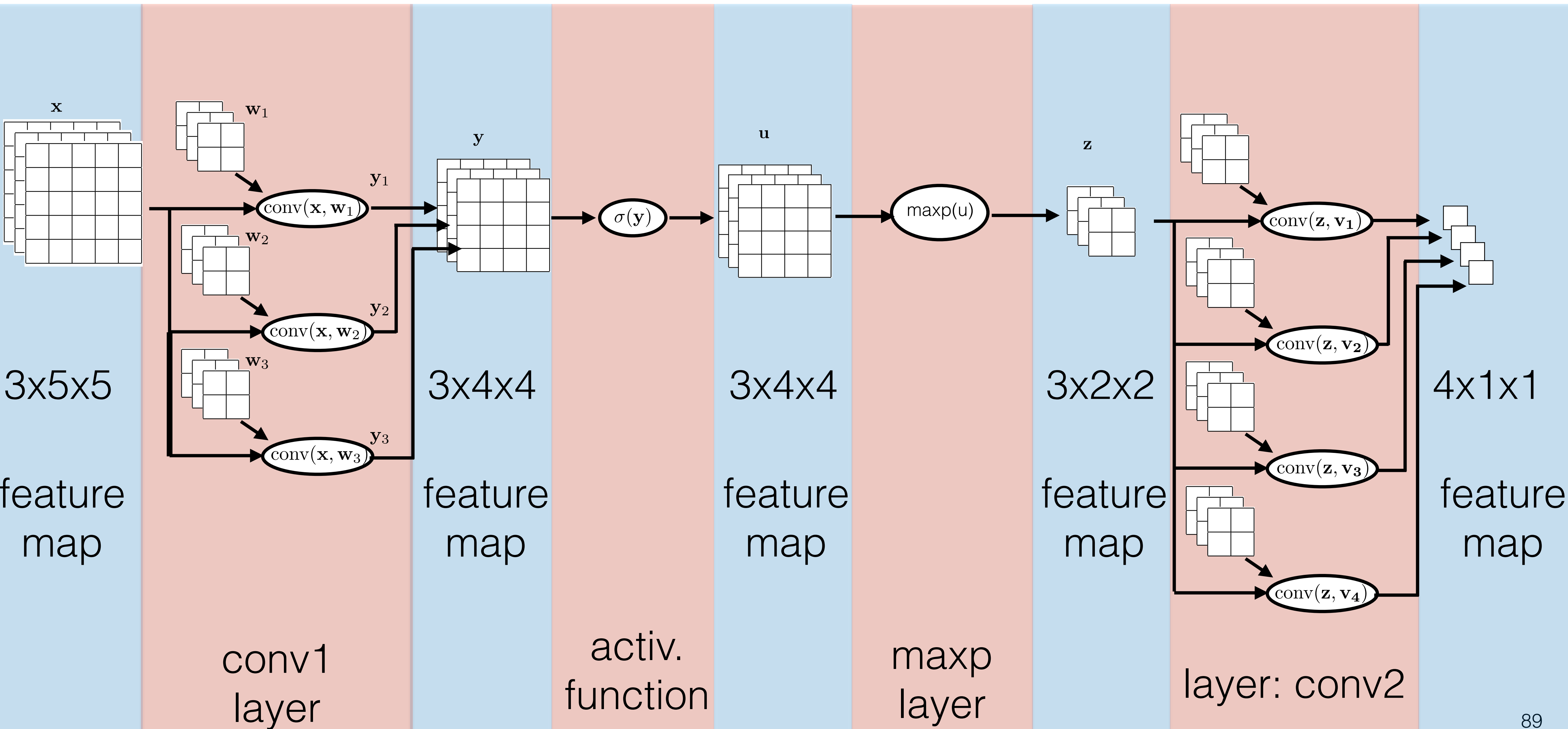
Learning of a simple convolutional network

input 3D tensor: channels x height x width



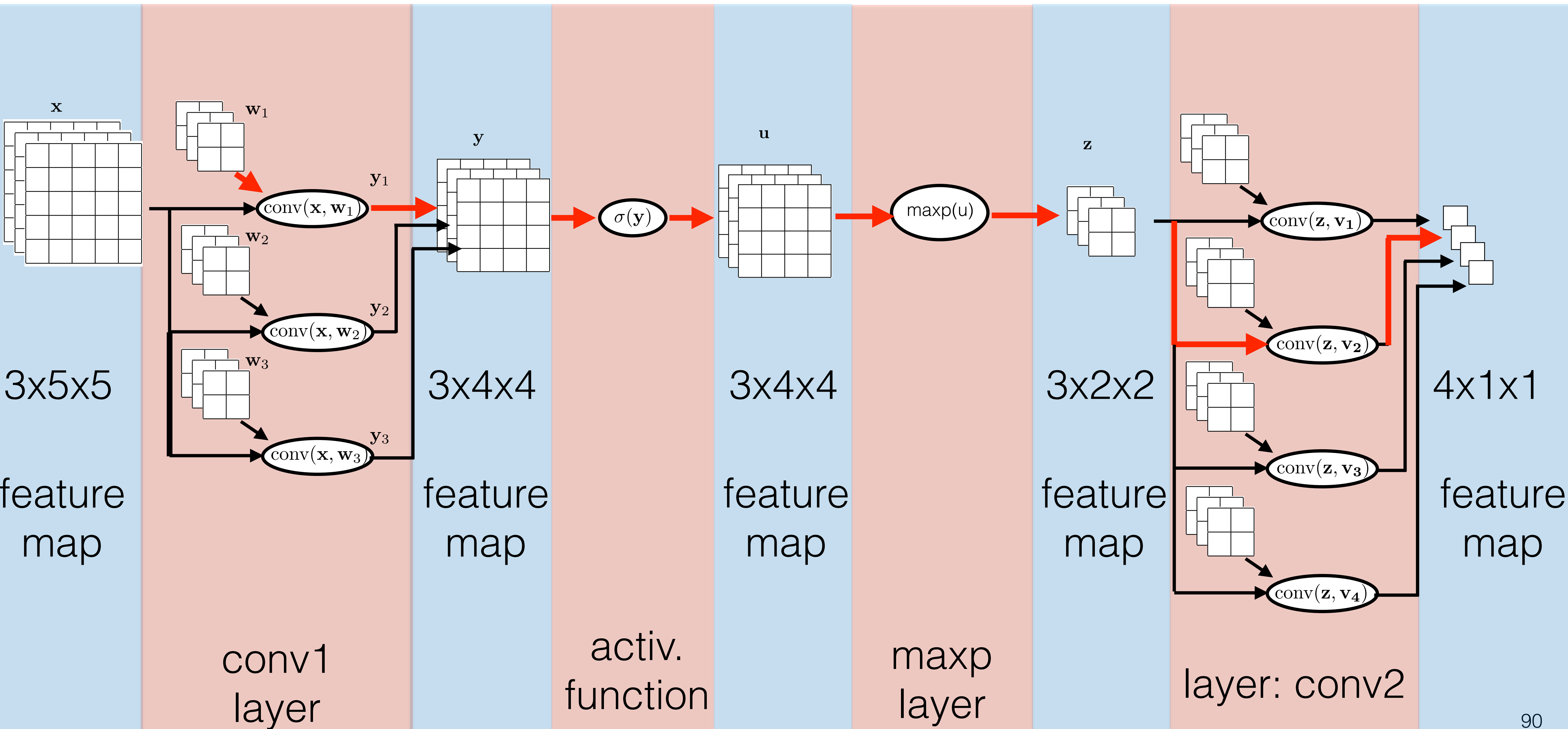
Learning of a simple convolutional network

input 3D tensor: channels x height x width



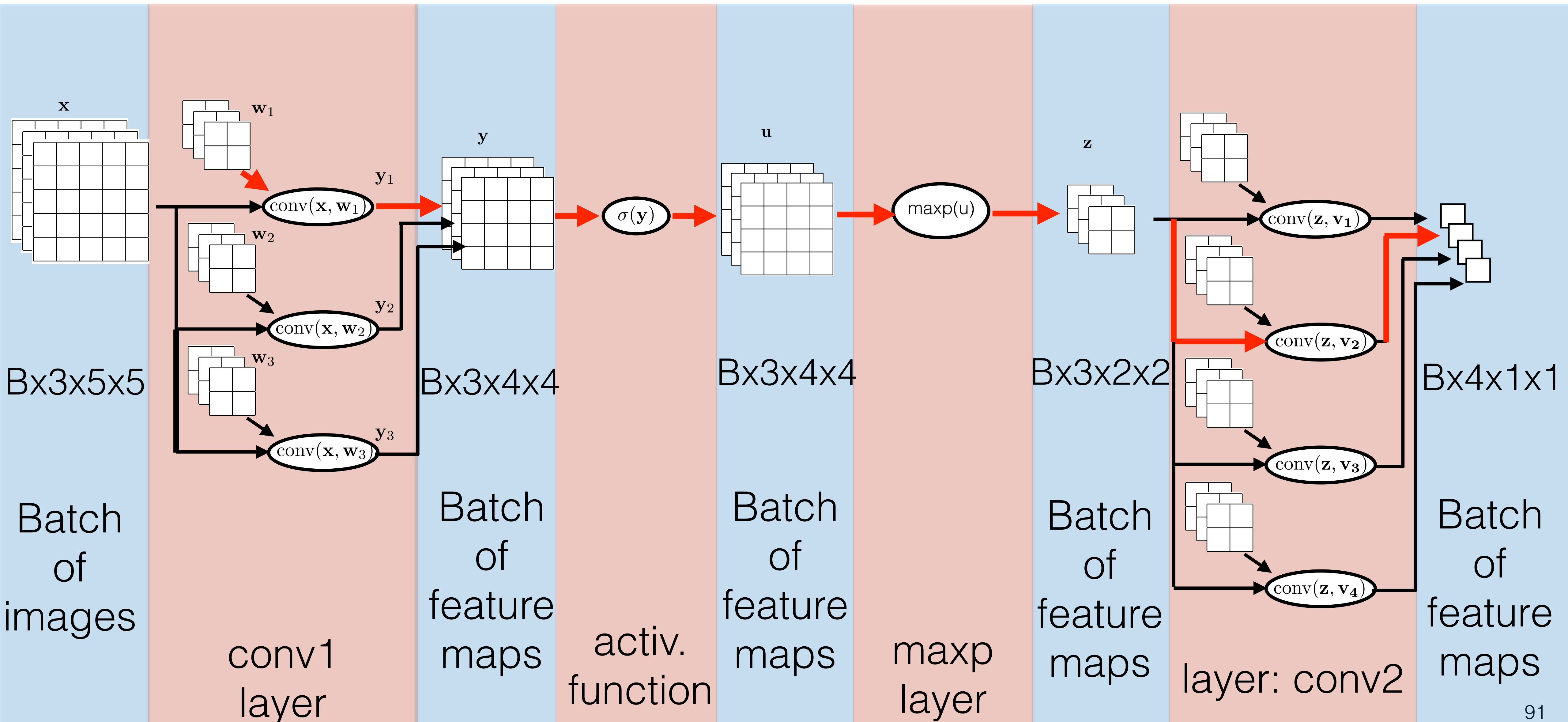
Learning of a simple convolutional network

input 3D tensor: channels x height x width



Learning with mini-batches

input 4D tensor: batch_size x channels x height x width



Convolutional net

- **Convolutional network** (ConvNet) is concatenation of convolutional layers, activation function and optionally max-pooling functions.
- **Backprop in convolutional layer** is convolution of feature maps or kernels or feature-maps with the upstream gradient.
- Feed-forward and backprop are convolutions => efficient implementation on GPU

Kunihiko Fukushima 1980

Biol. Cybernetics 36, 193-202 (1980)

Biological
Cybernetics
© by Springer-Verlag 1980

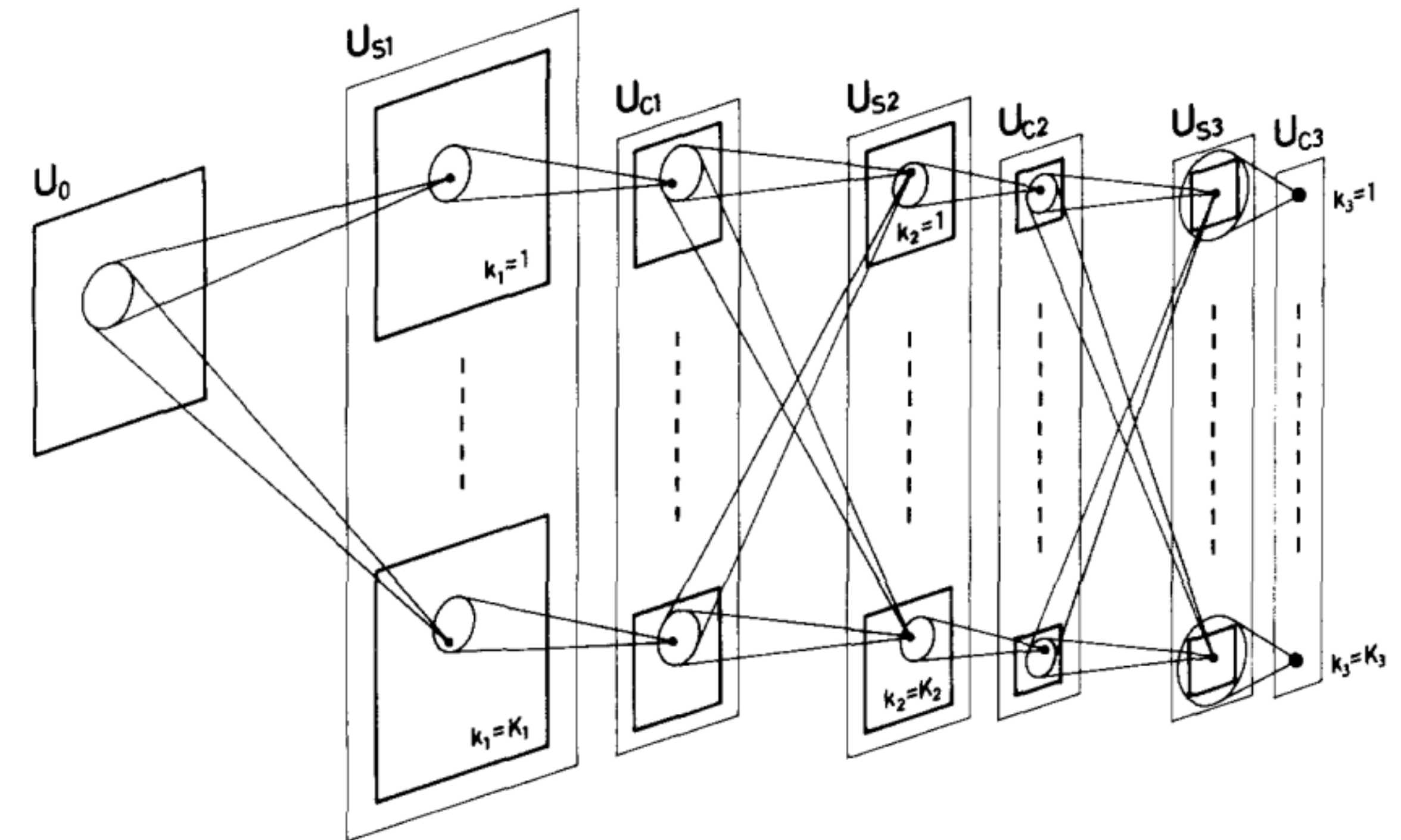
Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position

Kunihiko Fukushima

NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan

Abstract. A neural network model for a mechanism of visual pattern recognition is proposed in this paper. The network is self-organized by “learning without a teacher”, and acquires an ability to recognize stimulus patterns based on the geometrical similarity (Gestalt) of their shapes without affected by their positions. This network is given a nickname “neocognitron”. After completion of self-organization, the network has a structure similar to the hierarchy model of the visual

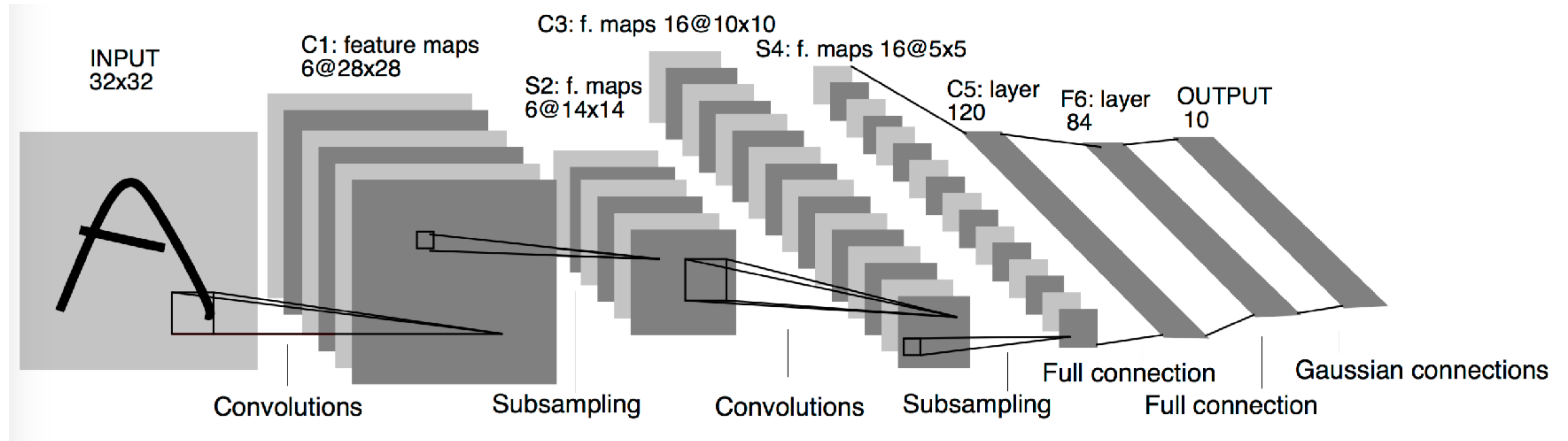
reveal it only by conventional physiological experiments. So, we take a slightly different approach to this problem. If we could make a neural network model which has the same capability for pattern recognition as a human being, it would give us a powerful clue to the understanding of the neural mechanism in the brain. In this paper, we discuss how to synthesize a neural network model in order to endow it an ability of pattern recognition like a human being.



https://en.wikipedia.org/wiki/Kunihiko_Fukushima

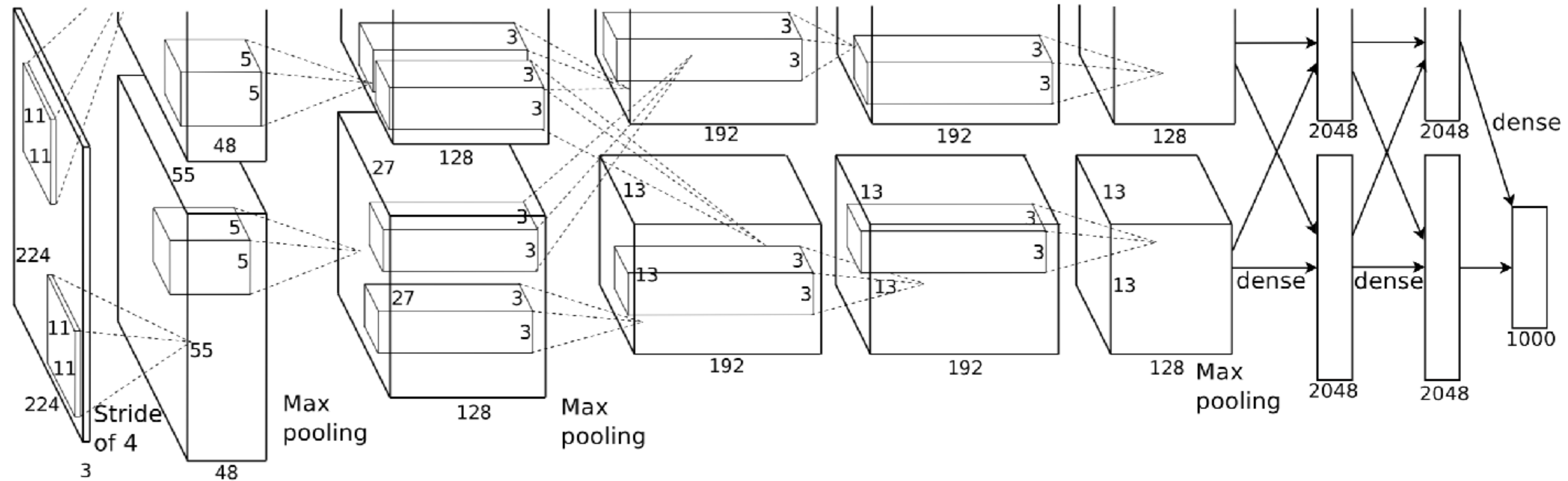
LeCun's letter recognition 1998 (over 13k citations !!!)

backpropagation formulated



LeCun et al, Gradient based learning applied to document recognition, IEEE, 1998
<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

AlexNet on ImageNet 2012 (**over 27k citations !!!**)



Alex Krizhevsky et al, Imagenet classification with deep convolutional neural networks, NIPS, 2012

<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

1.2M images with (227x227x3) resolution

<http://image-net.org/challenges/LSVRC/2017/index>

Steel drum



Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle



Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle



$$\text{Error} = \frac{1}{100,000} \sum_{100,000 \text{ images}} 1[\text{incorrect on image } i]$$

Classification results

AlexNet

8 layers

VGGnet

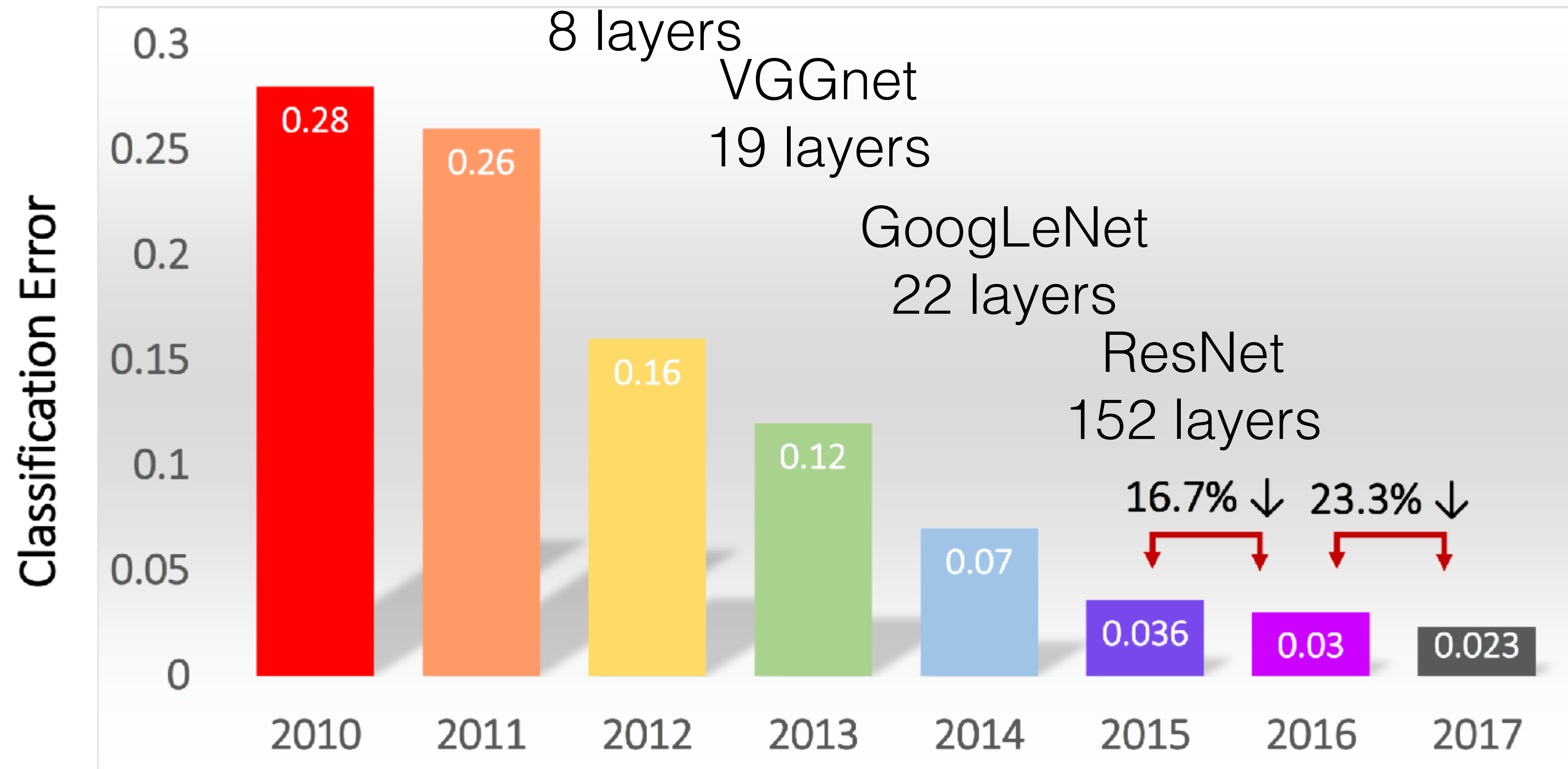
19 layers

GoogLeNet

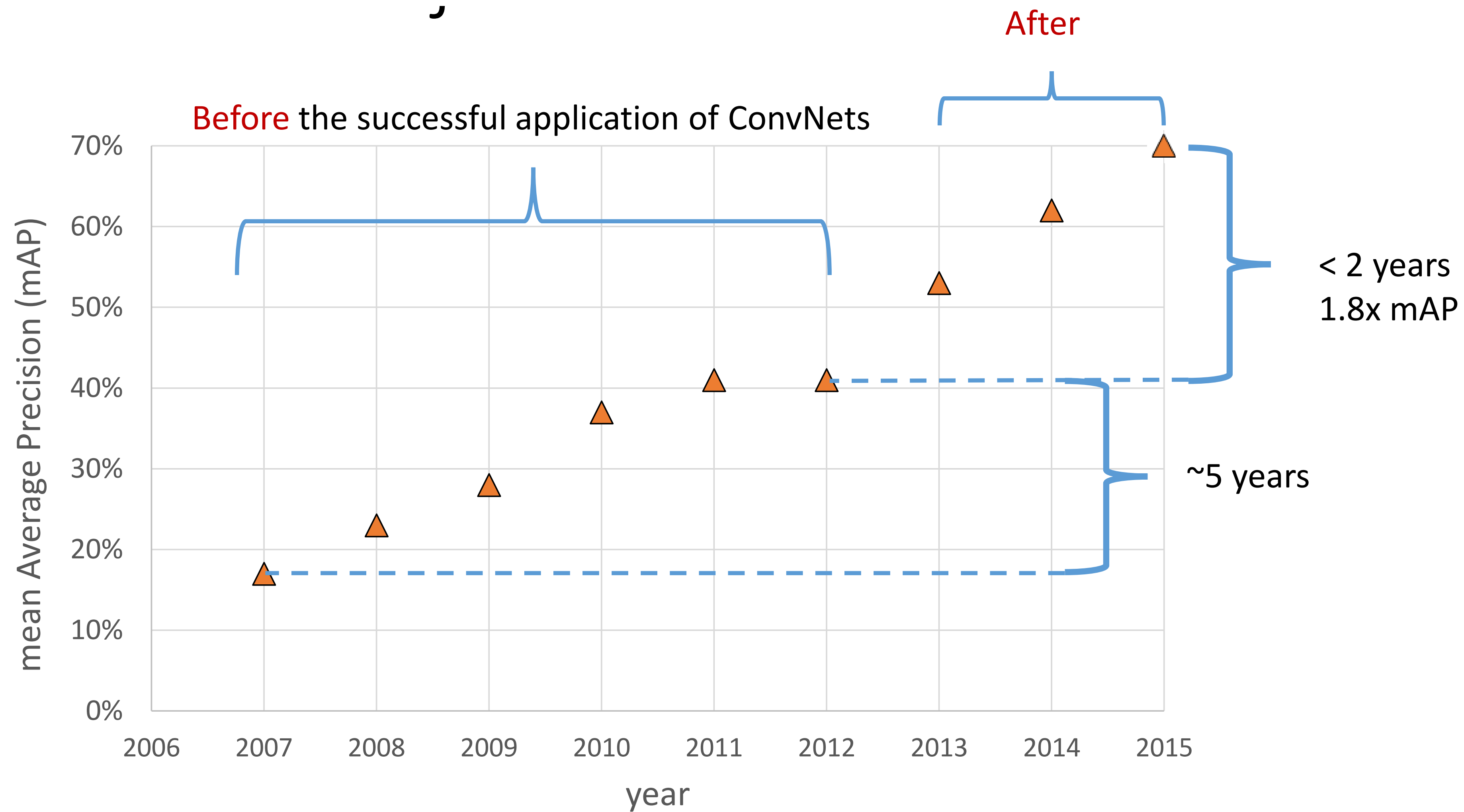
22 layers

ResNet

152 layers



Pascal VOC object detection challenge

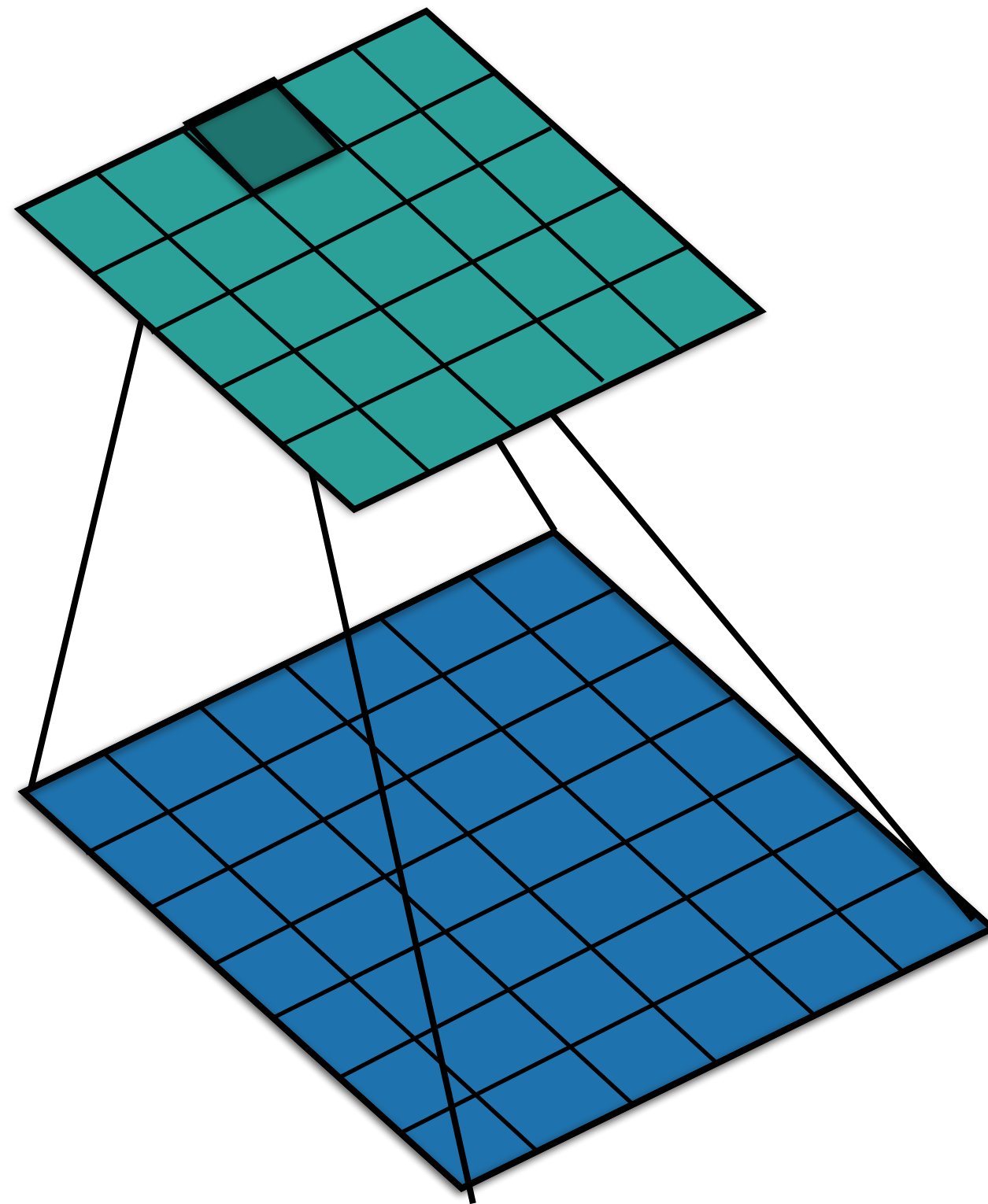


Under the hood of ConvNets

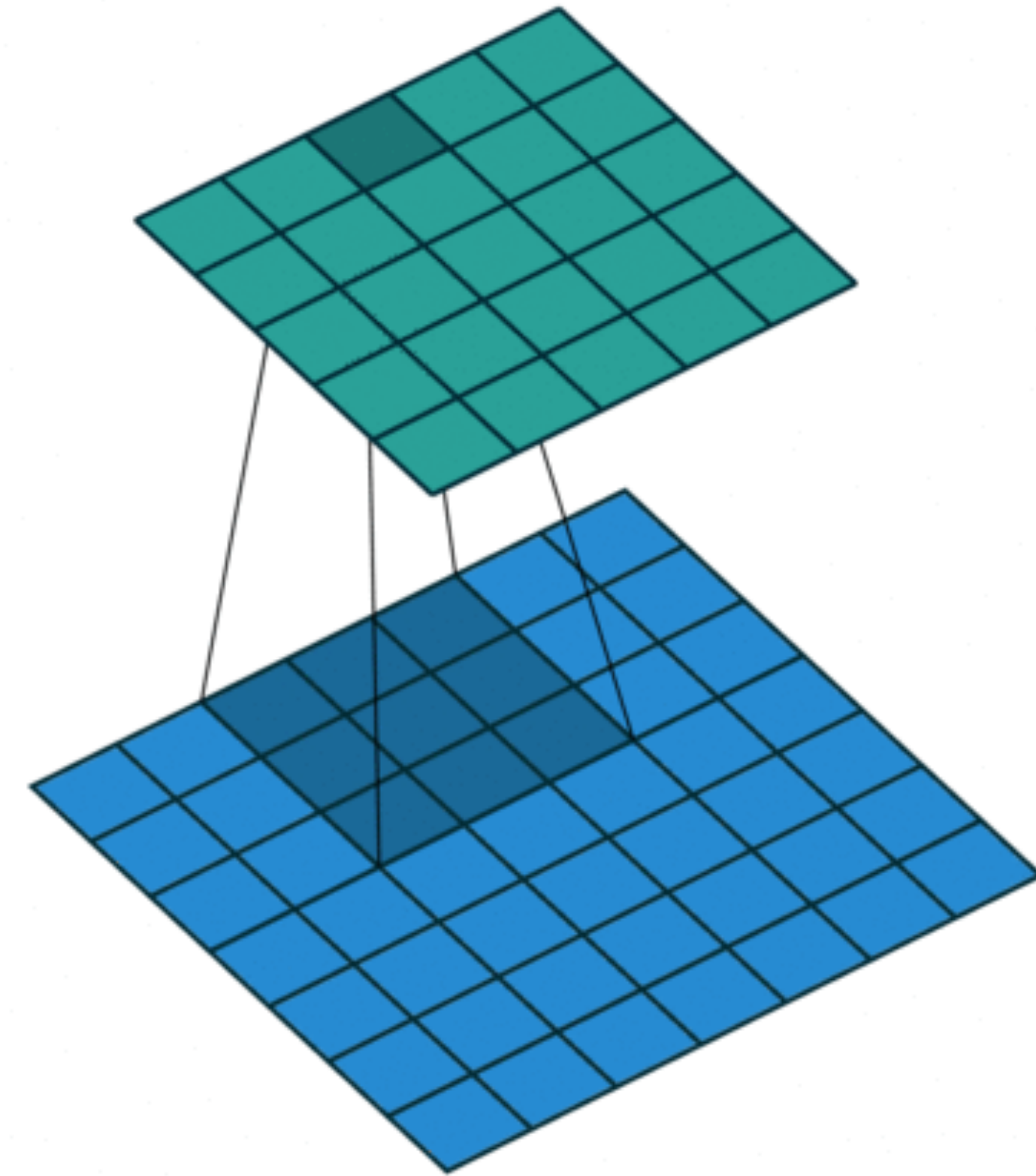
Convolution as spatial attention

FCNN = “global hard attention”

Convolution = “local hard attention”



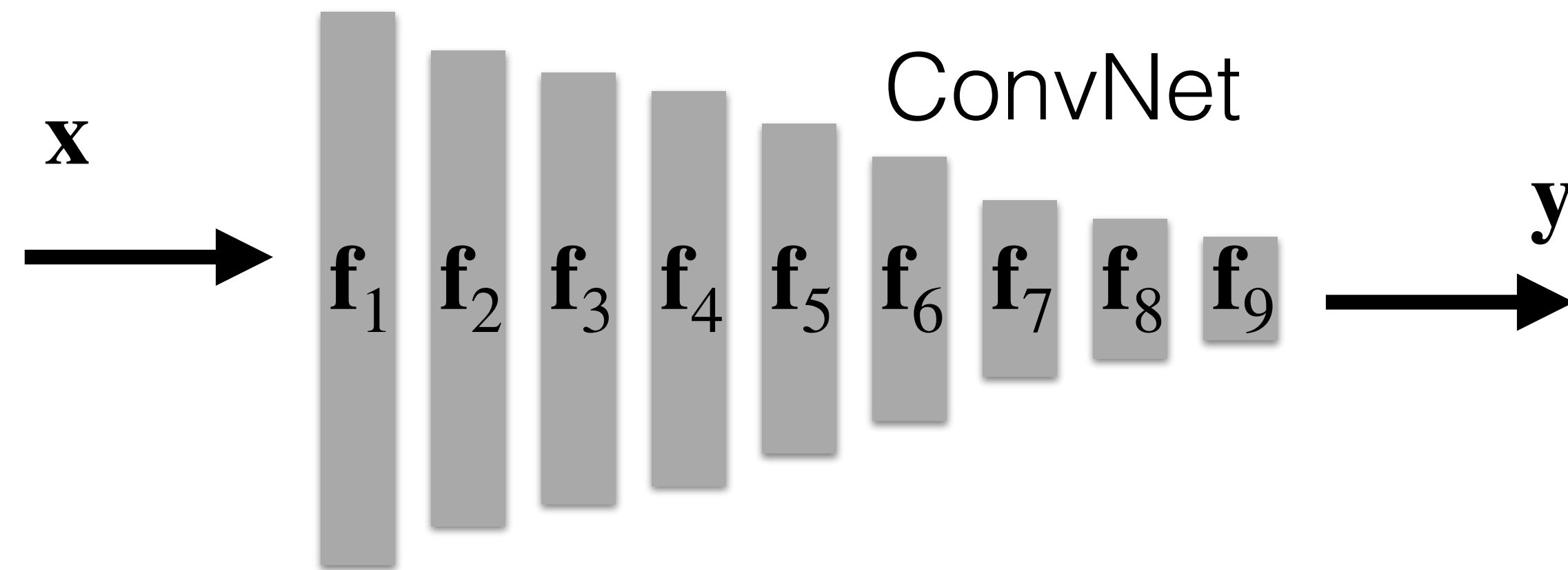
Fully-connected layer



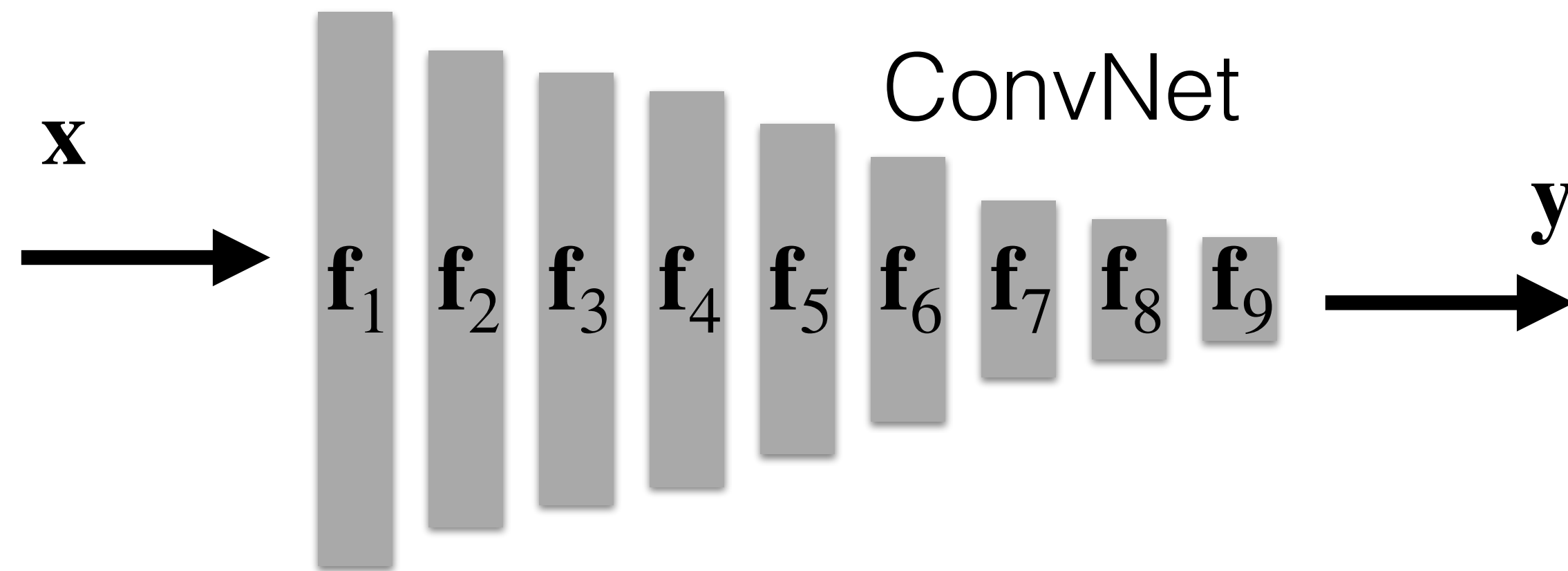
Convolutional layer

Do you see any other suitable attentions?

Trained kernels

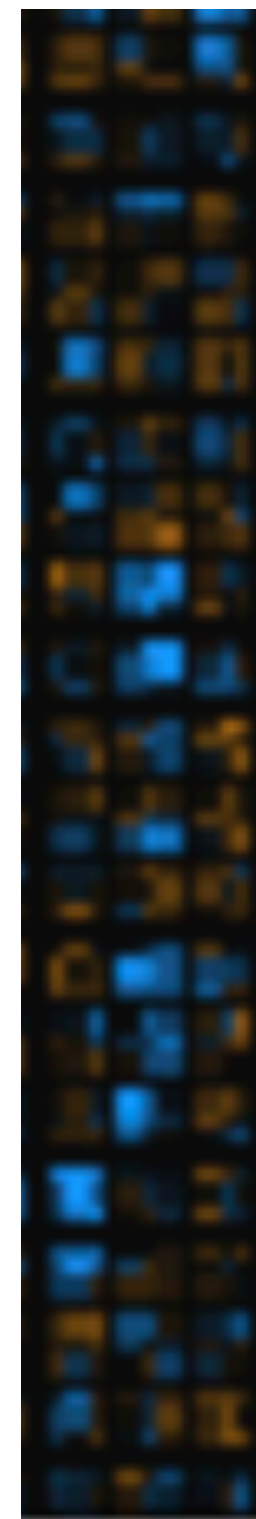
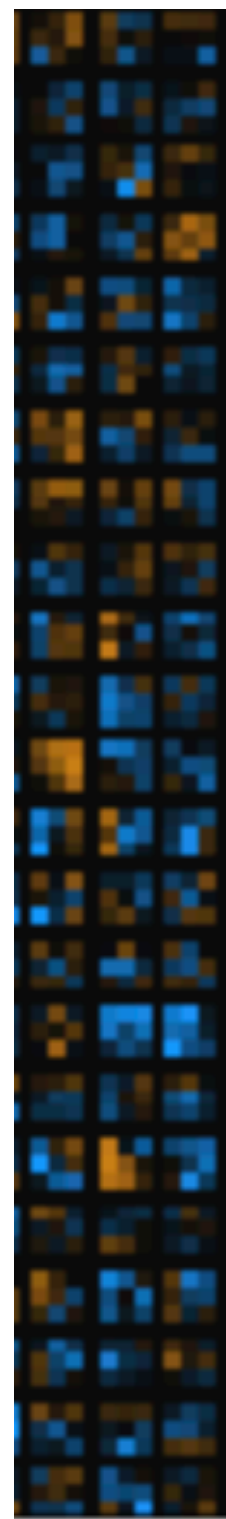
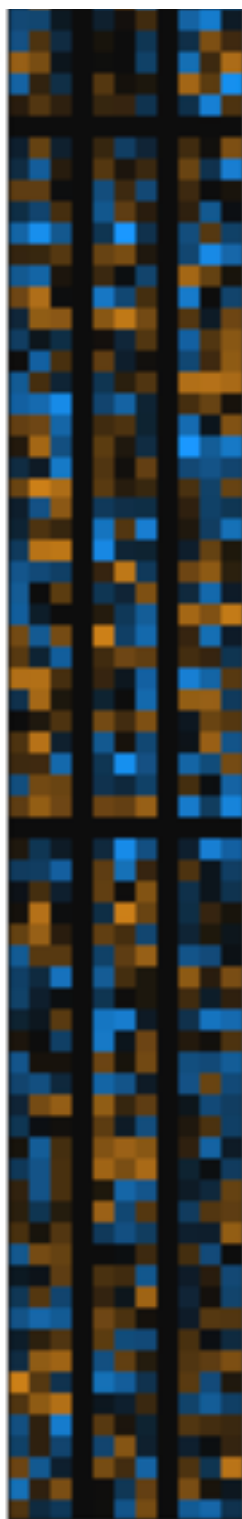


Trained kernels



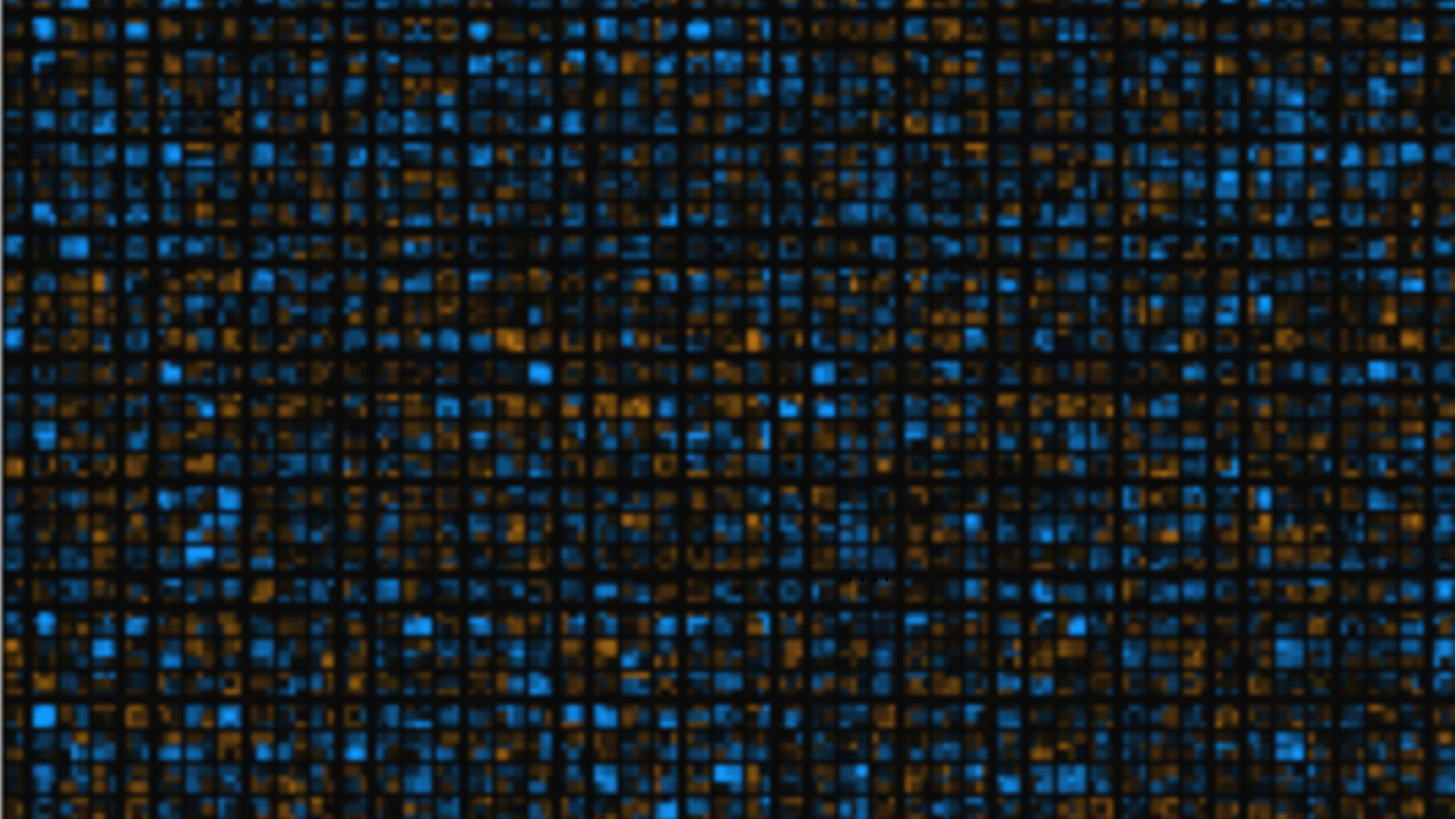
kernels/filters

Kernel for Conv 0 (Channelwise Slices)



....

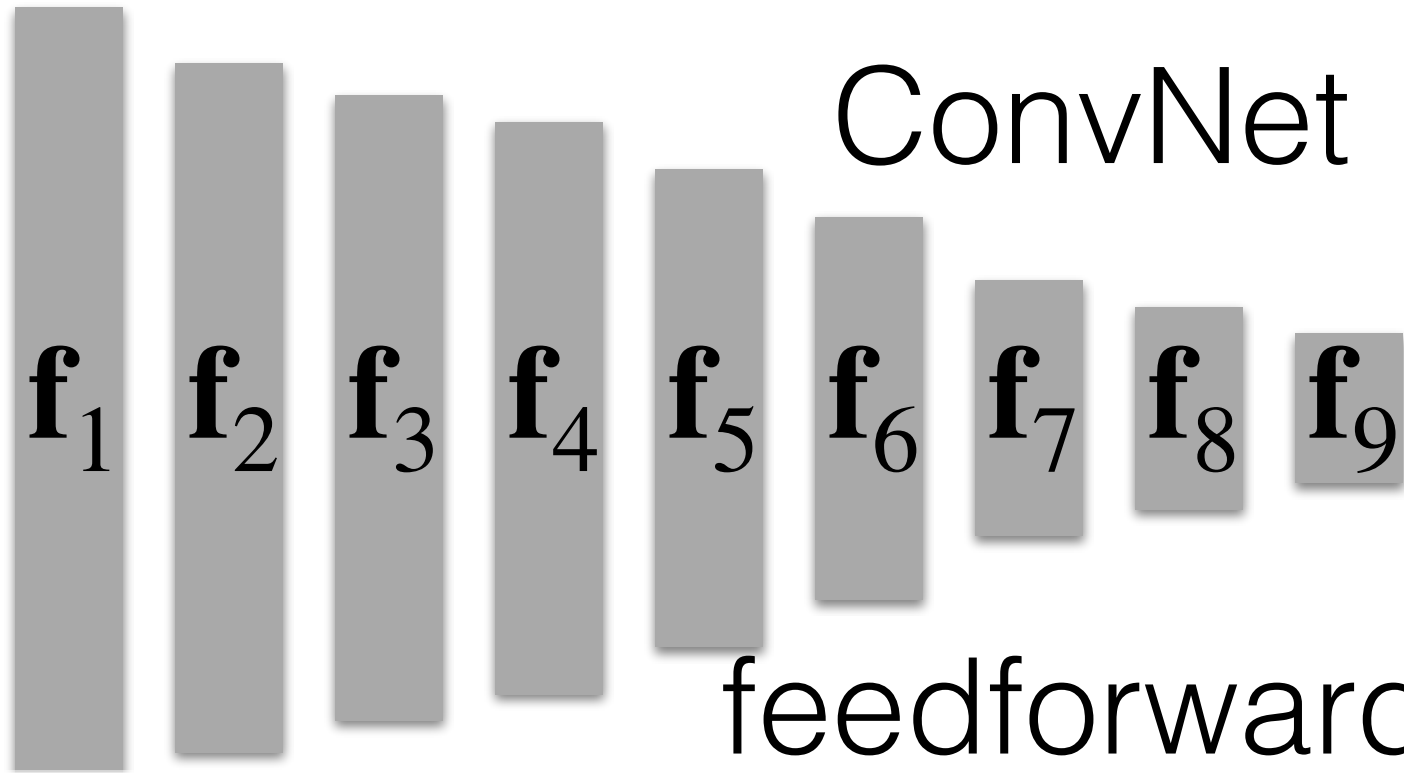
Direct visualization of
trained kernels does not say
too much



Feature maps = low dimensional encoding



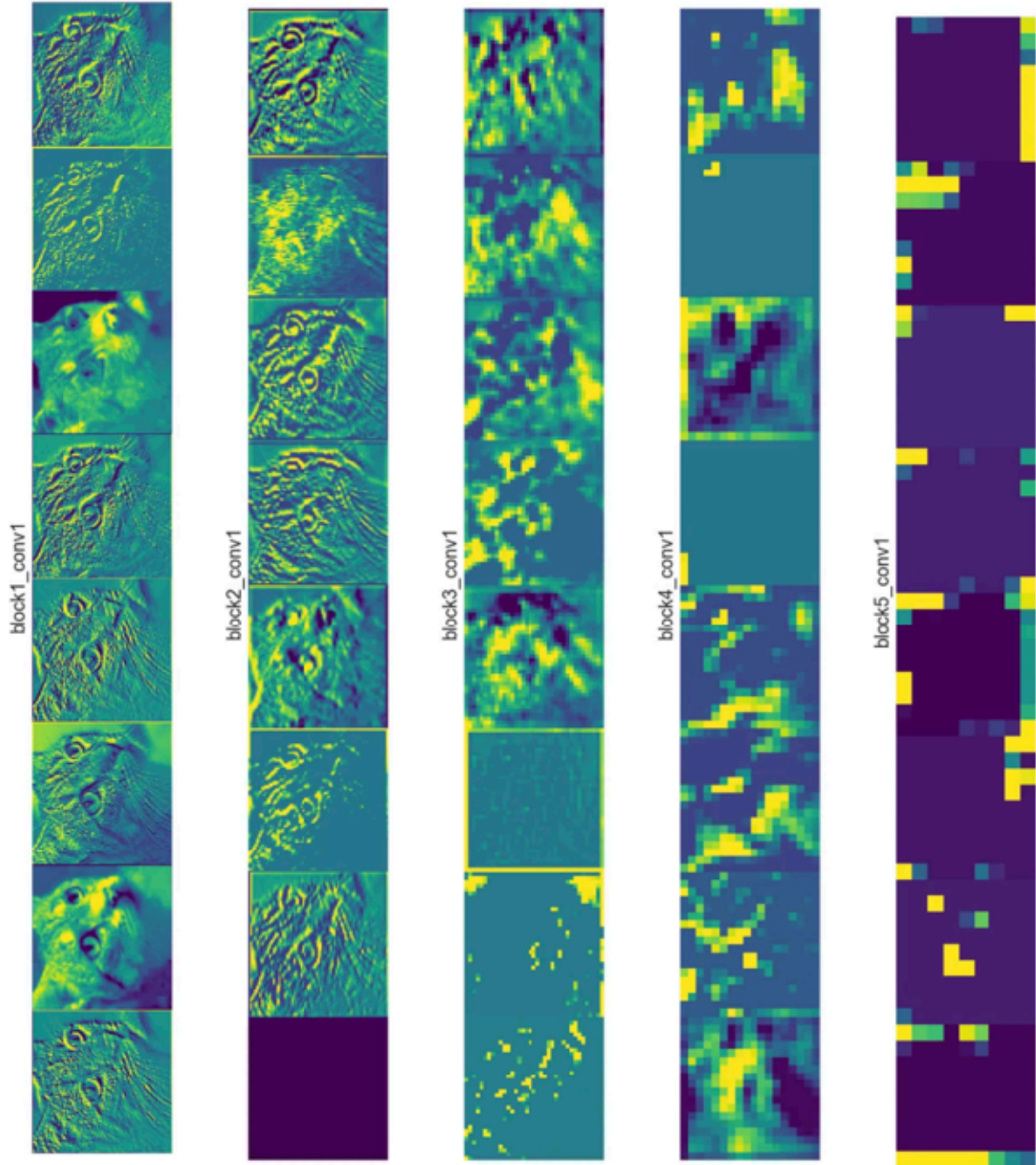
$\mathbf{x}$
→



$\mathbf{y}$

0.2	dog
0.1	ship
0.5	cat
0.0	car
0.1	airplane

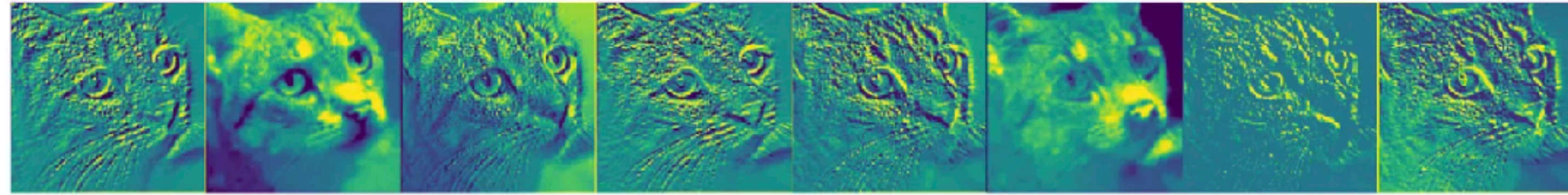
Feature maps:
intermediate results
of feedforward pass



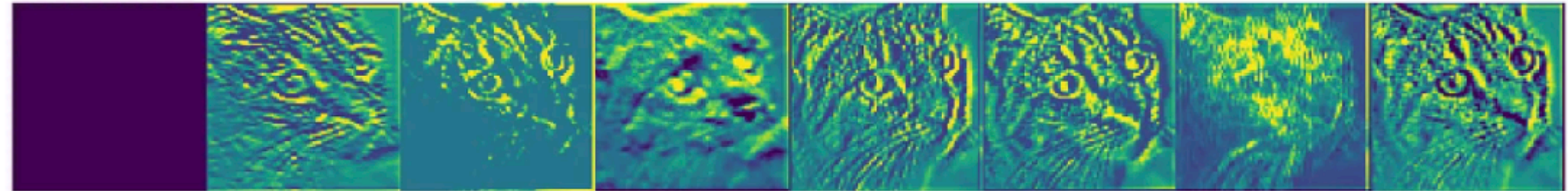
Feature maps = low dimensional encoding



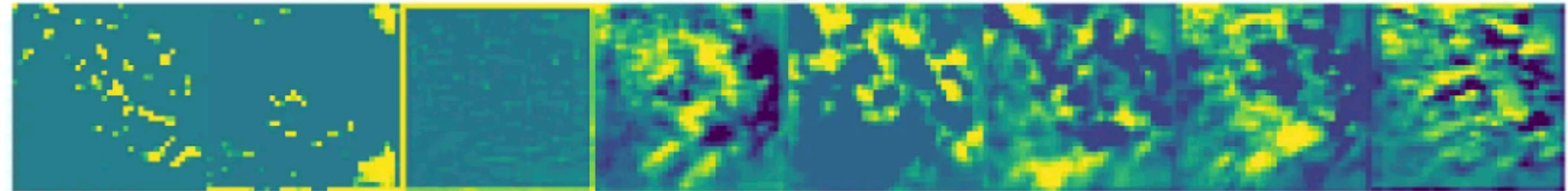
block1_conv1



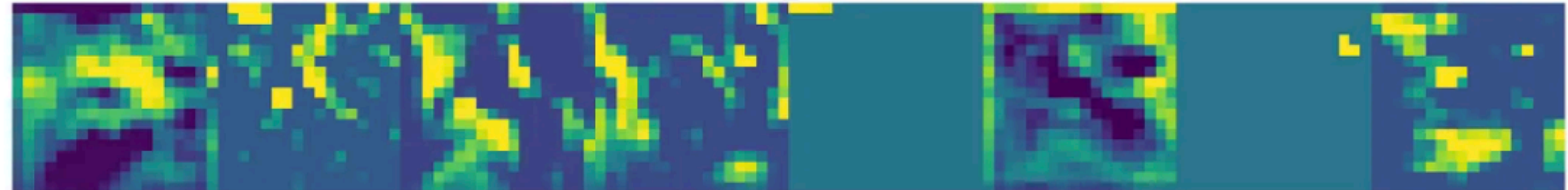
block2_conv1



block3_conv1



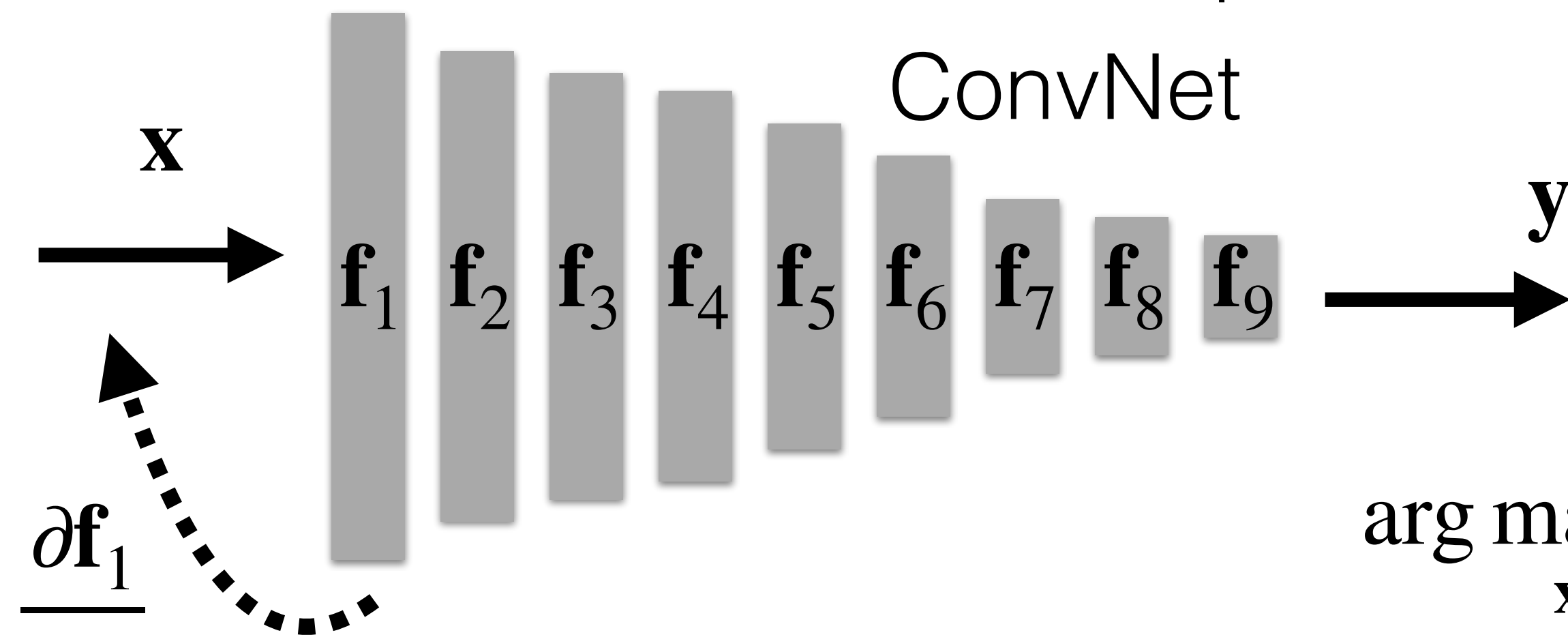
block4_conv1



block5_conv1

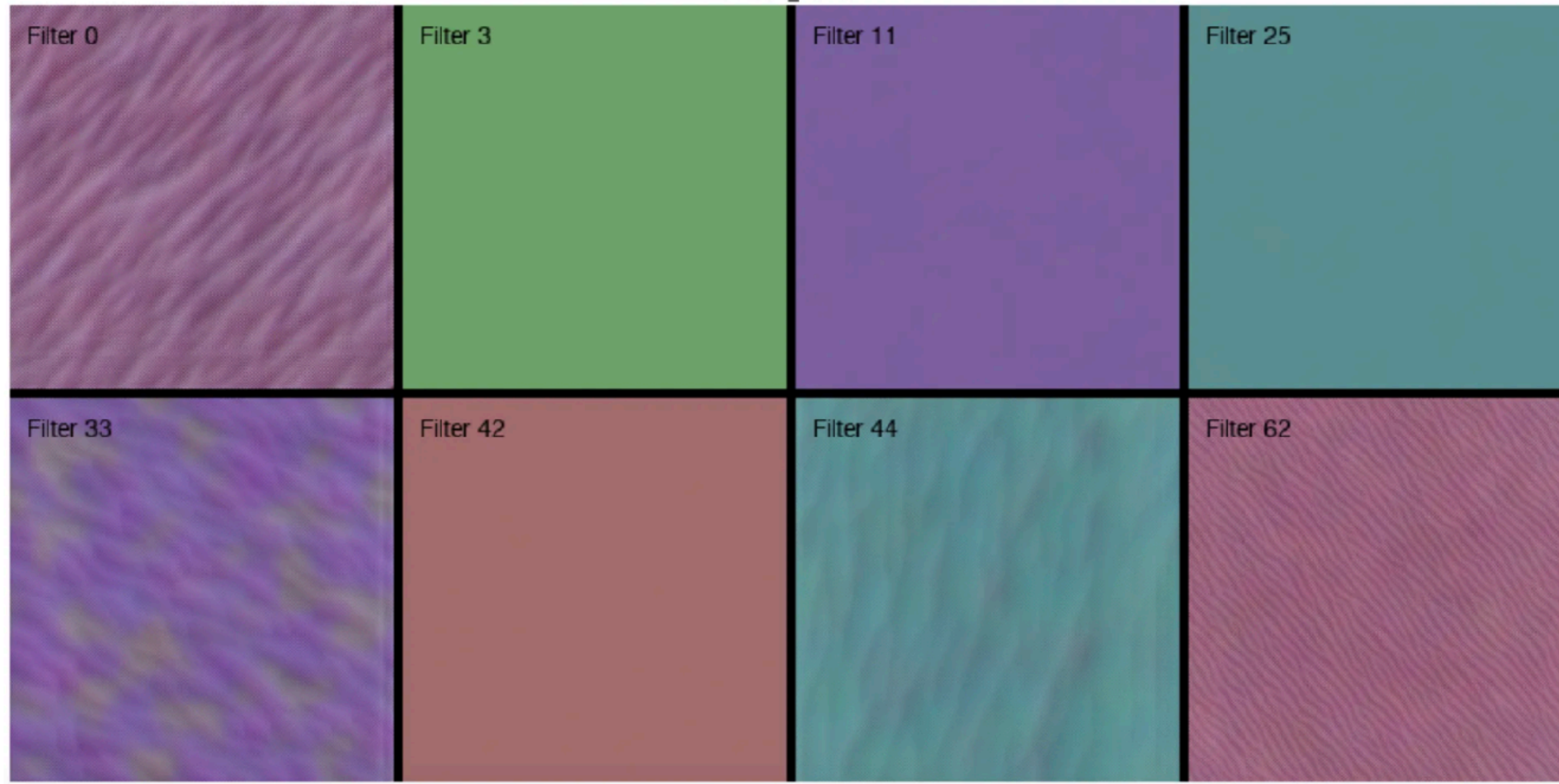


Features maps that maximize filter output

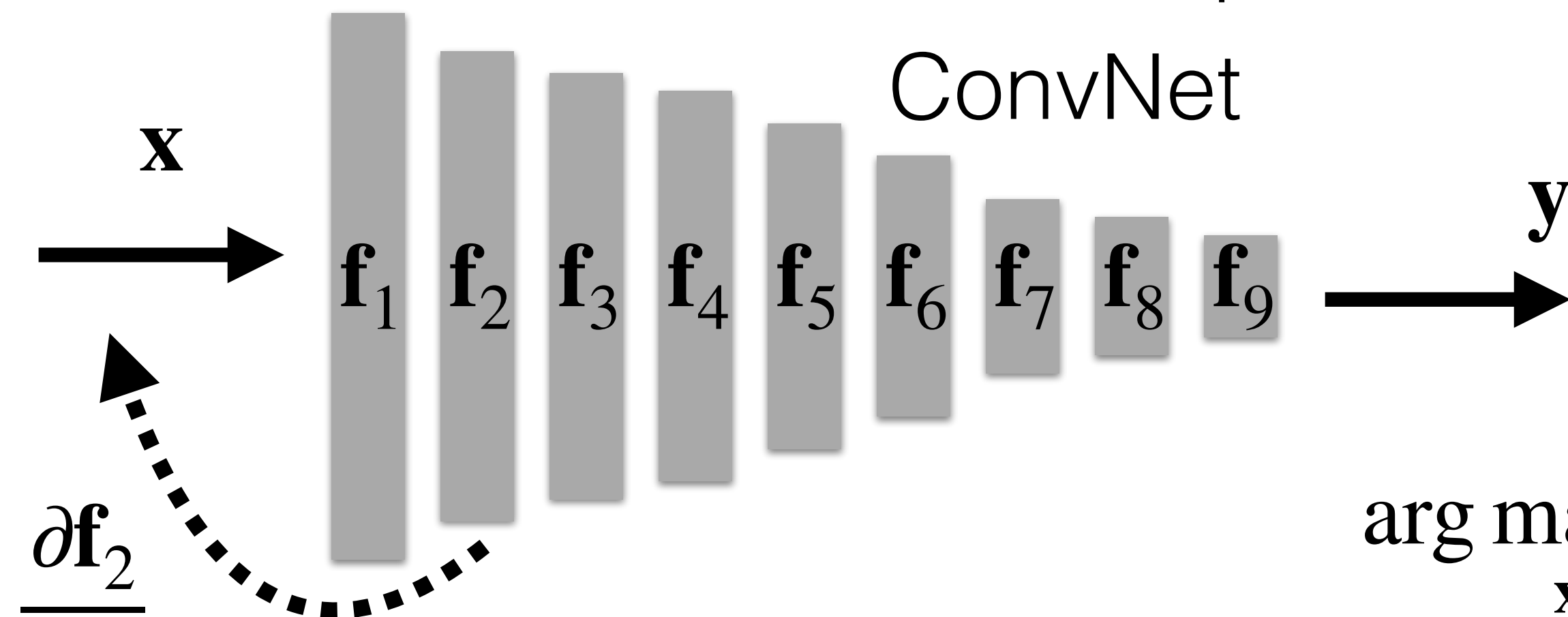


What is the input that maximizes a filter output?

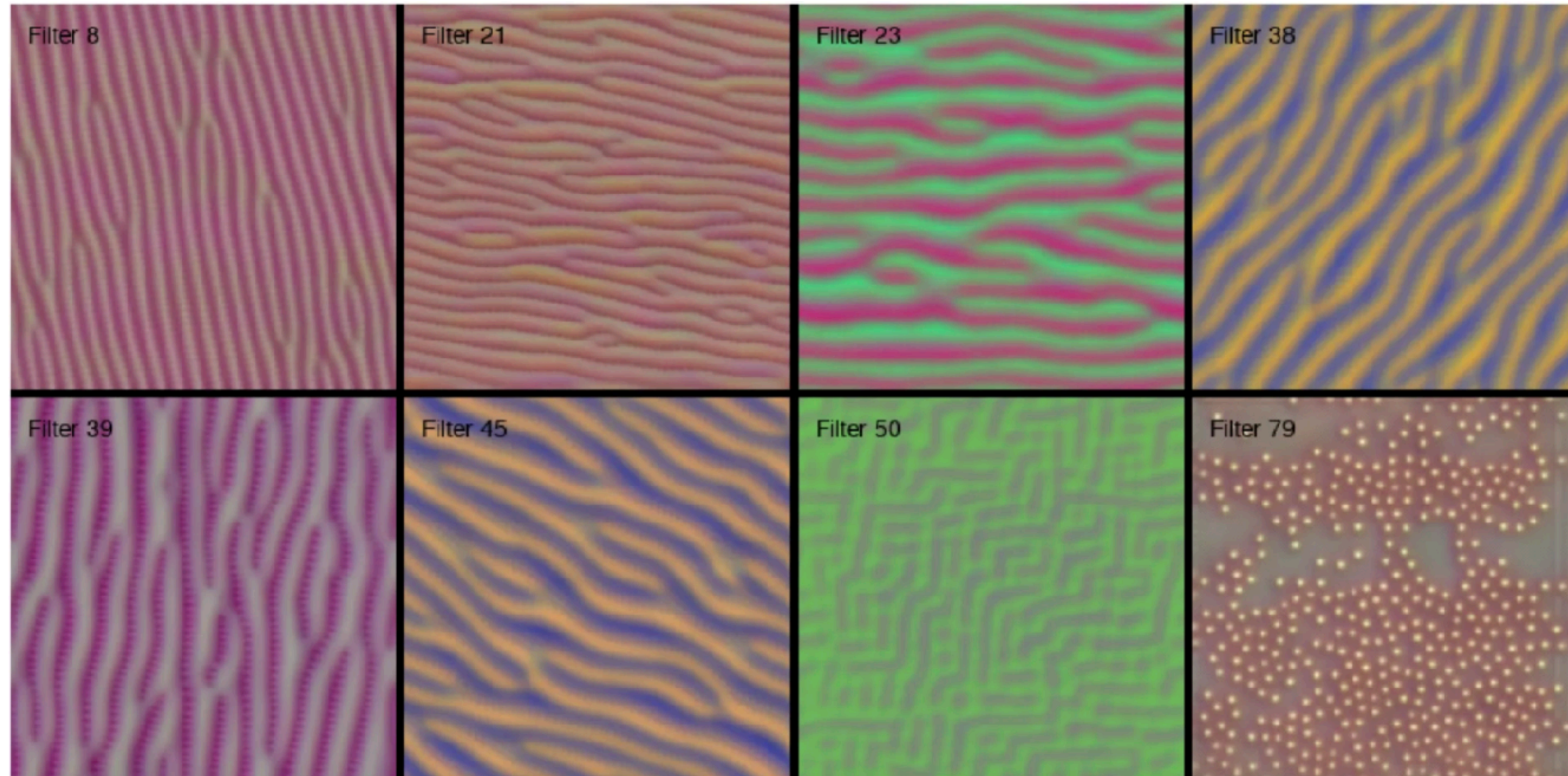
$$\arg \max_{\mathbf{x}} \|\mathbf{f}_1(\mathbf{x}, \mathbf{w})\|$$



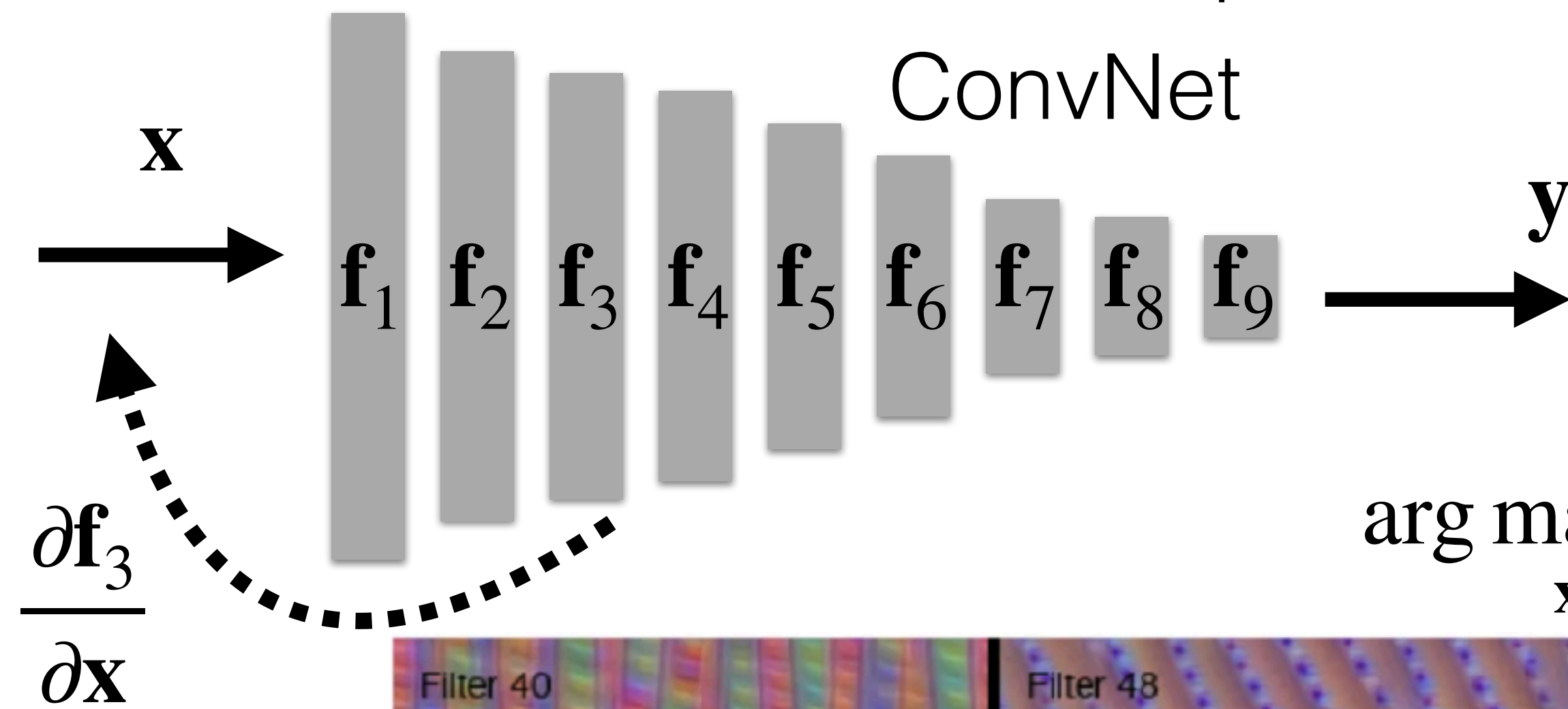
Features maps that maximize filter output



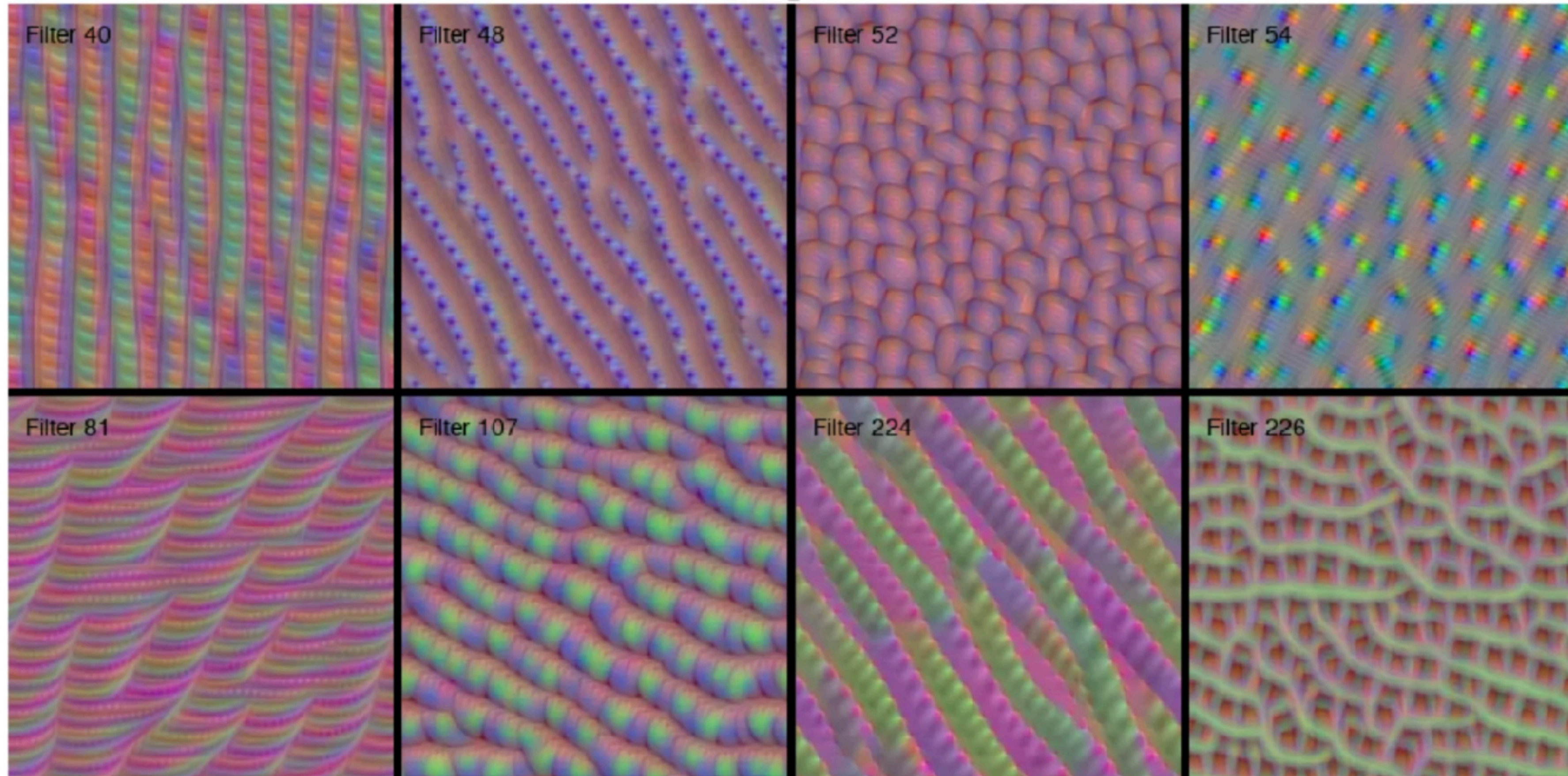
What is the input that maximizes a filter output?



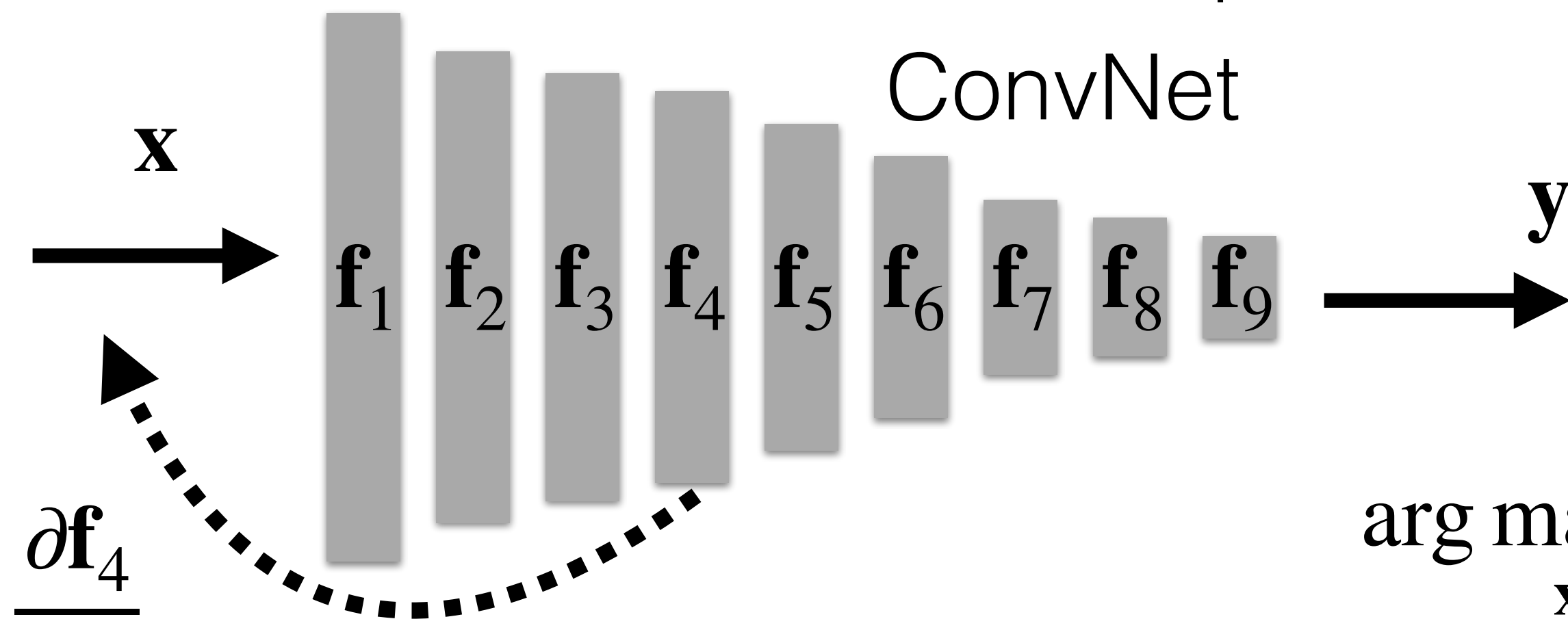
Features maps that maximize filter output



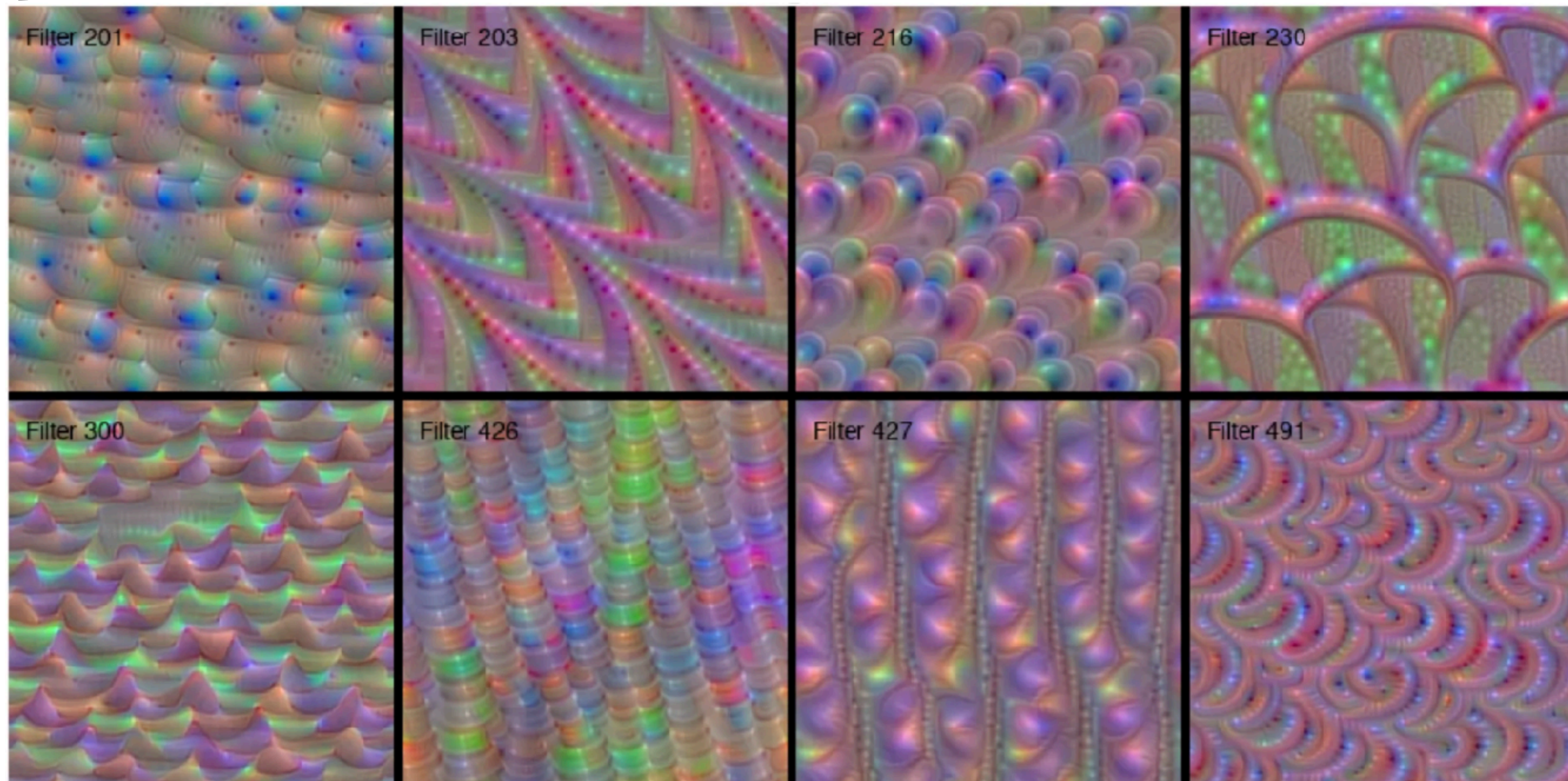
What is the input that maximizes a filter output?



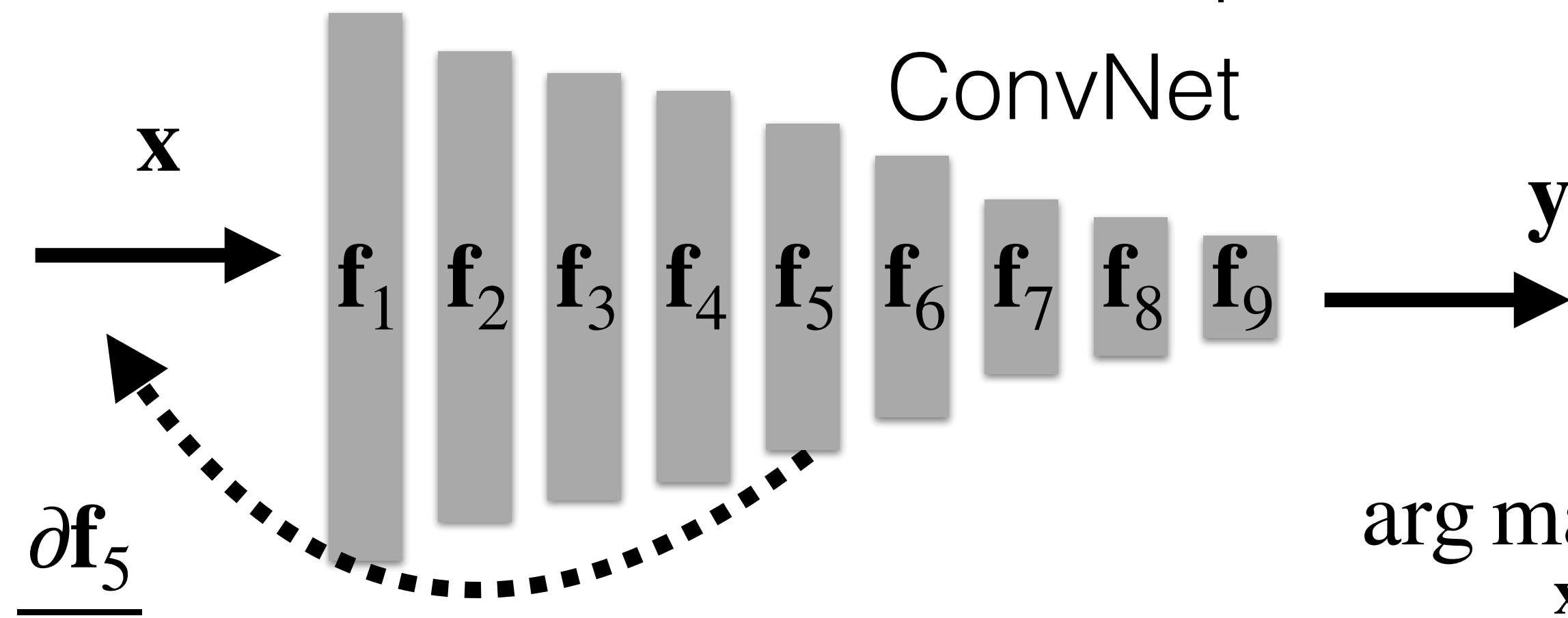
Features maps that maximize filter output



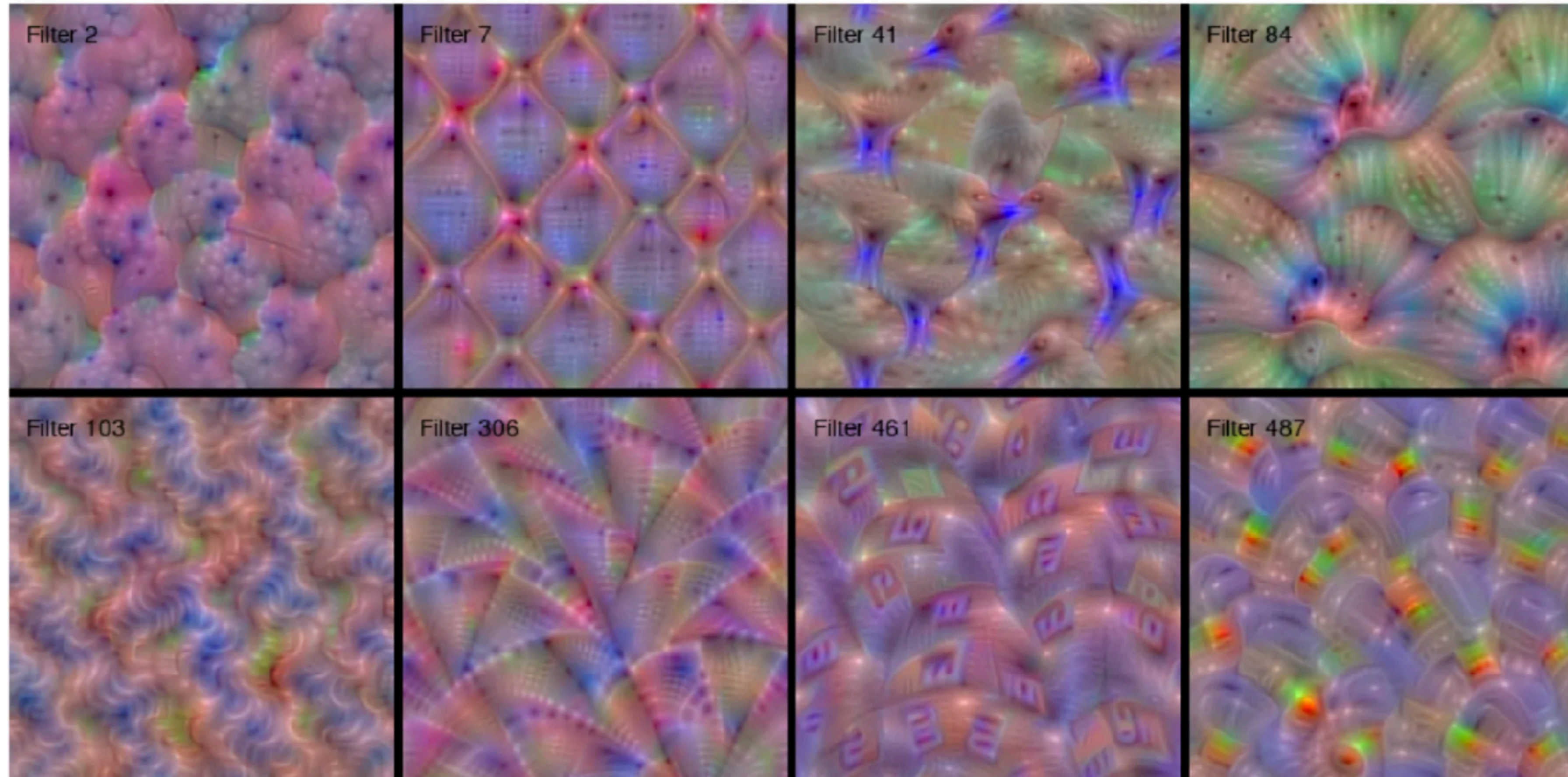
What is the input that maximizes a filter output?



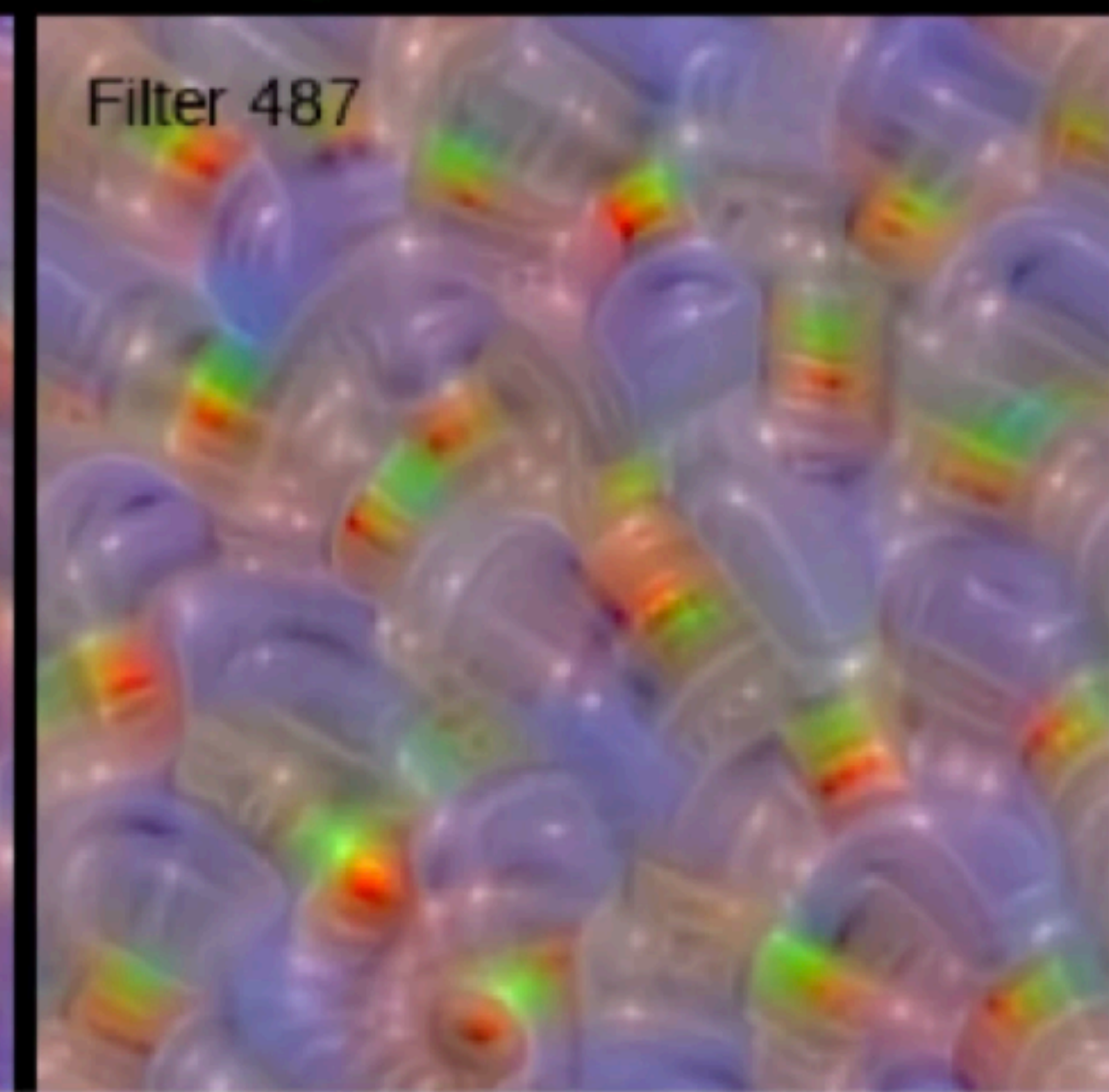
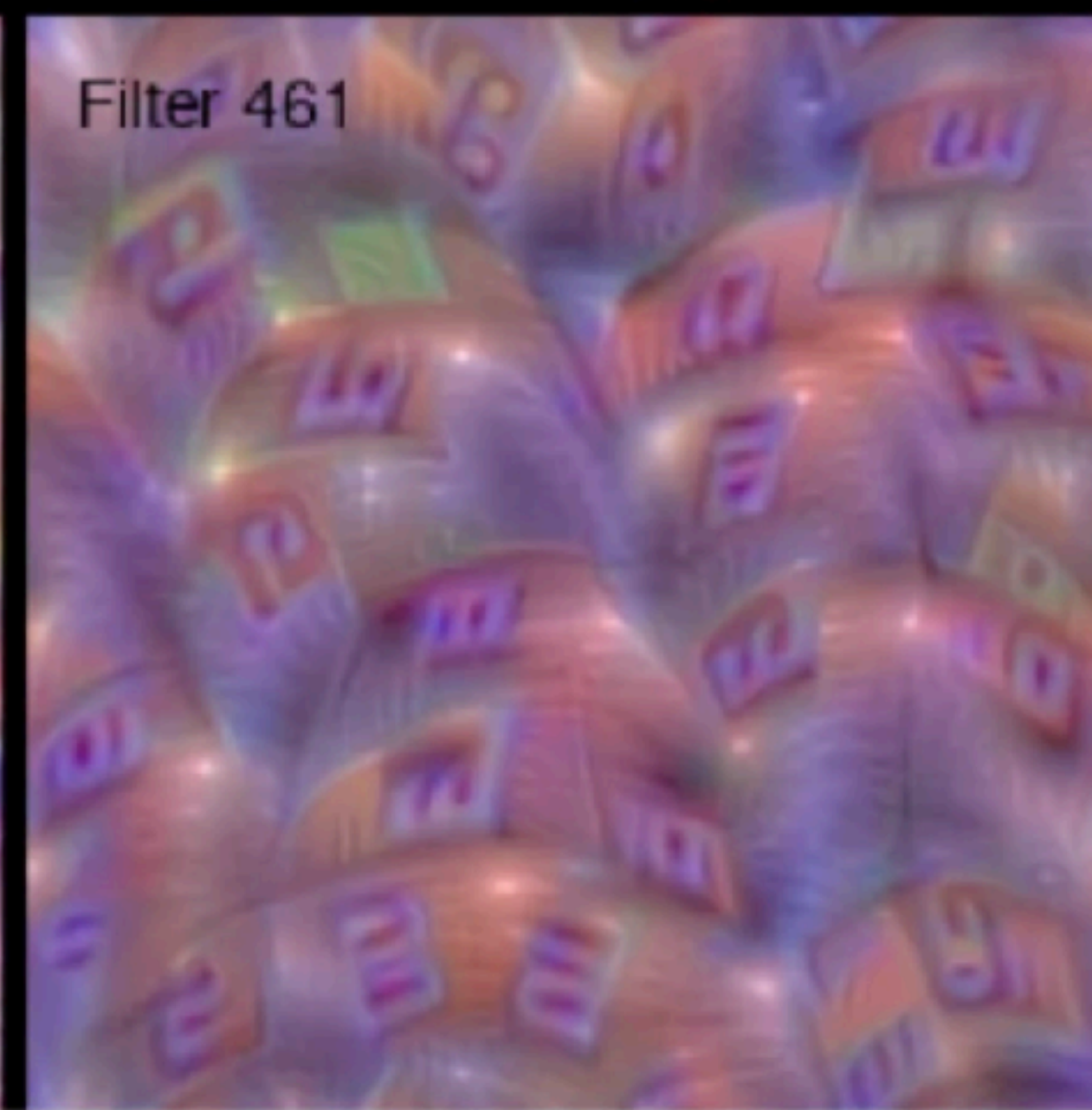
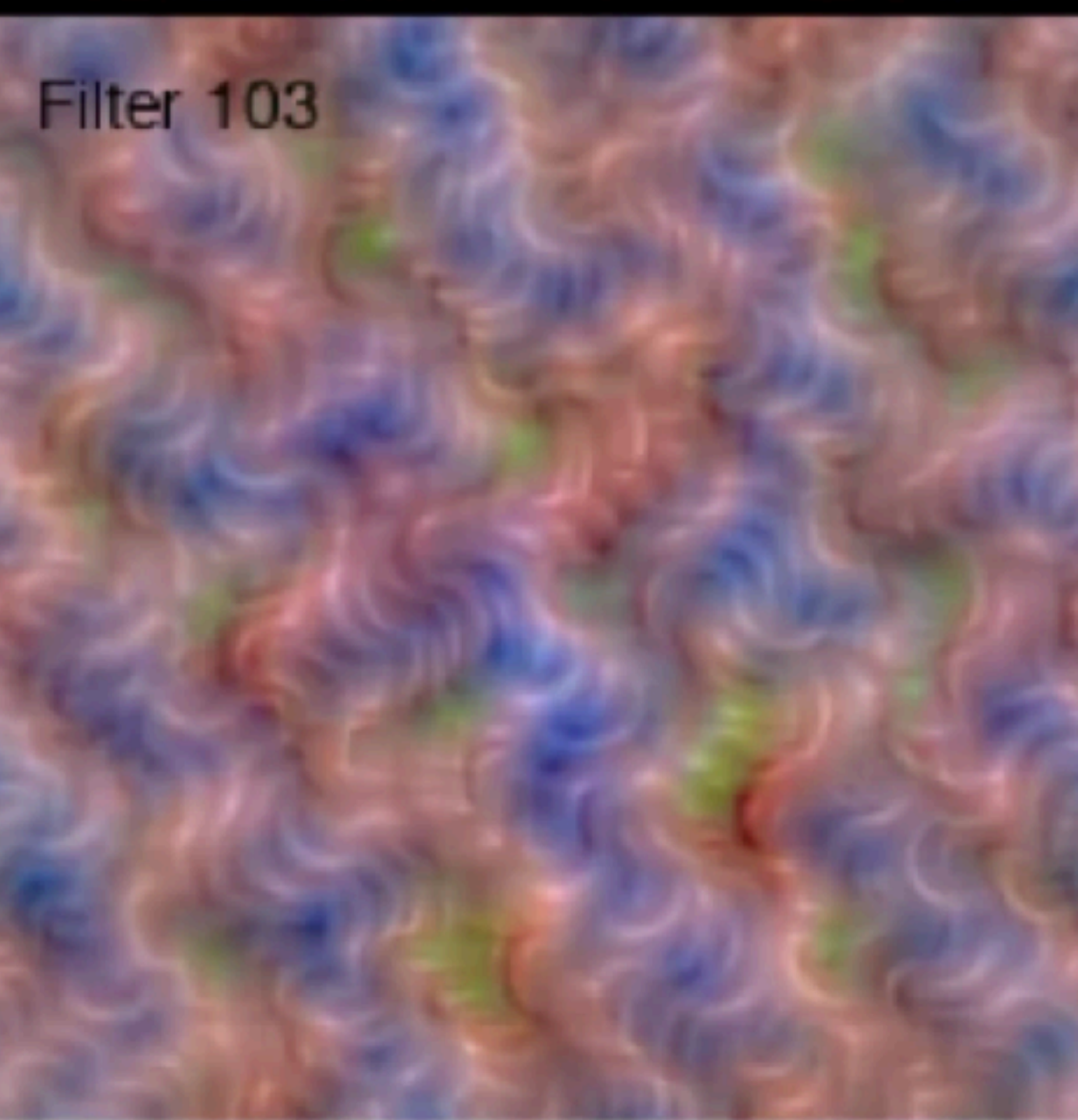
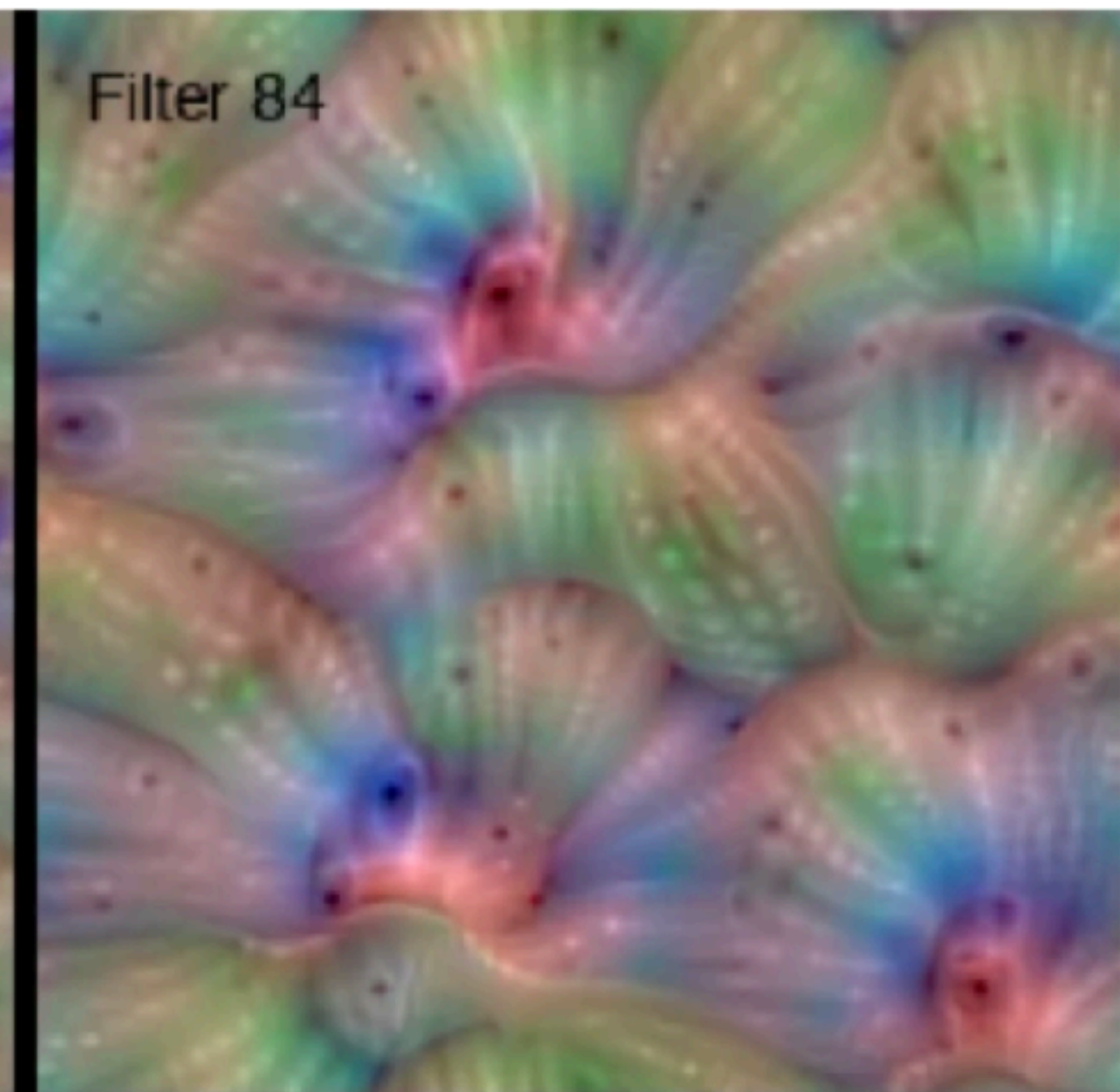
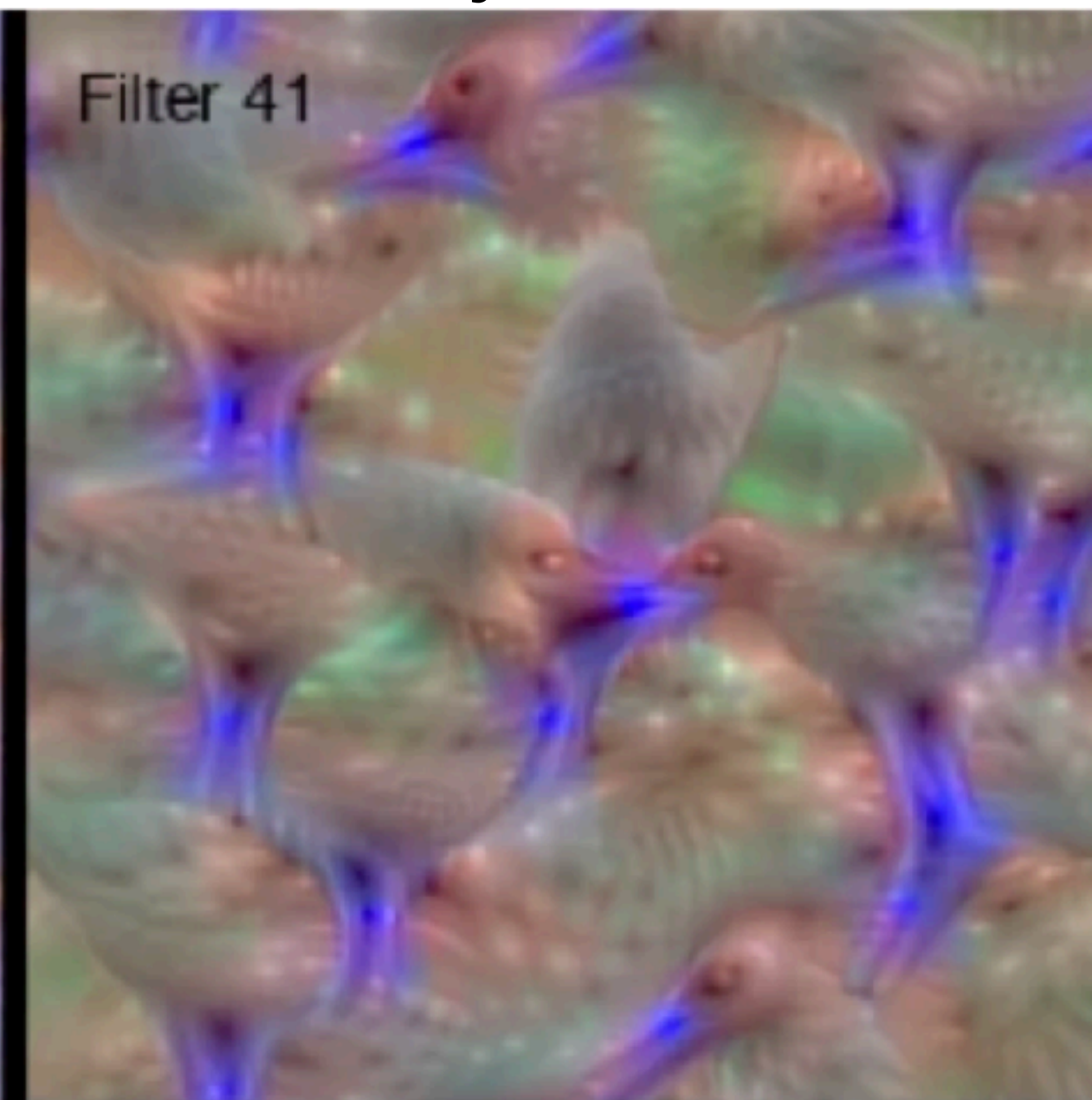
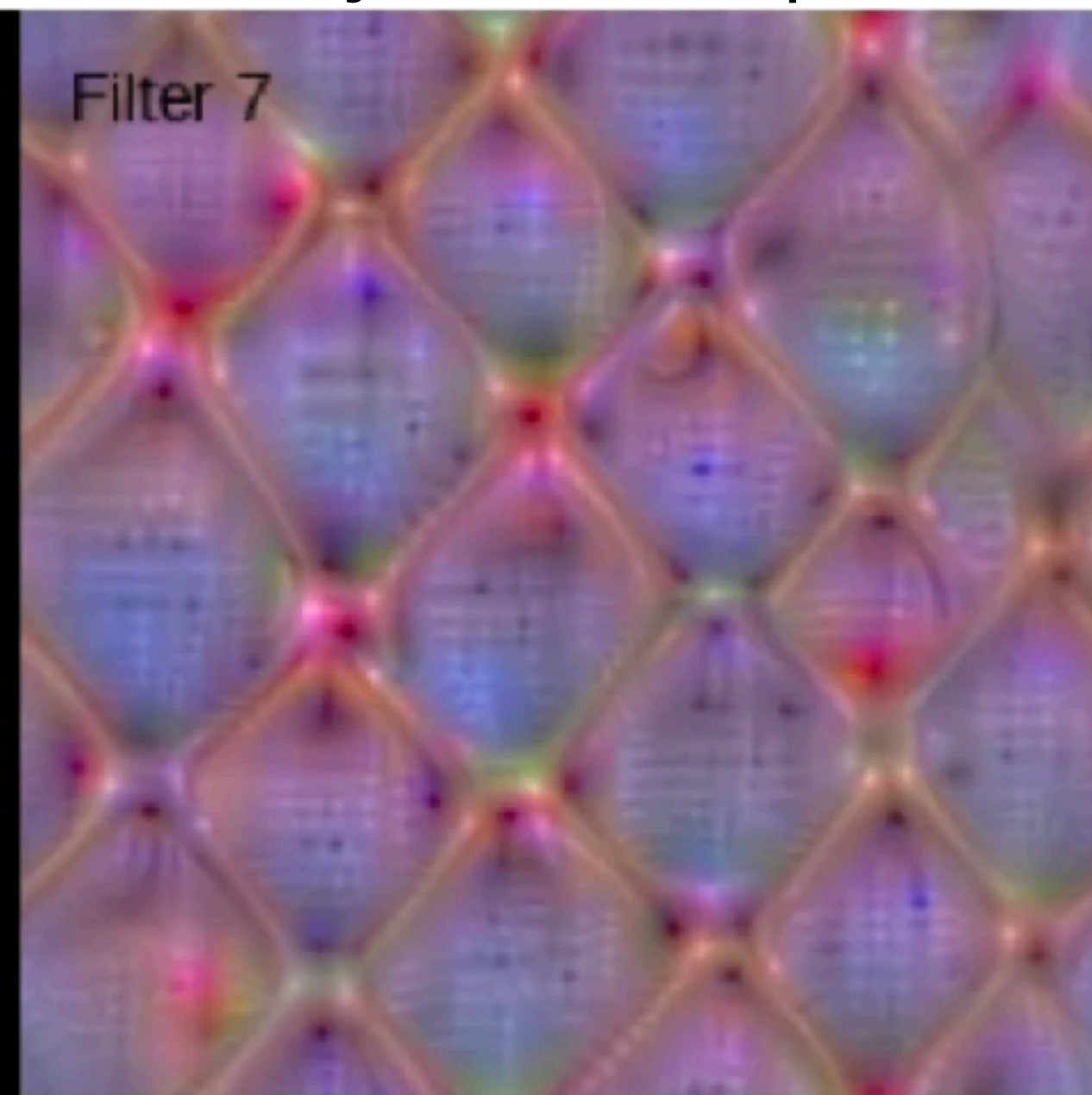
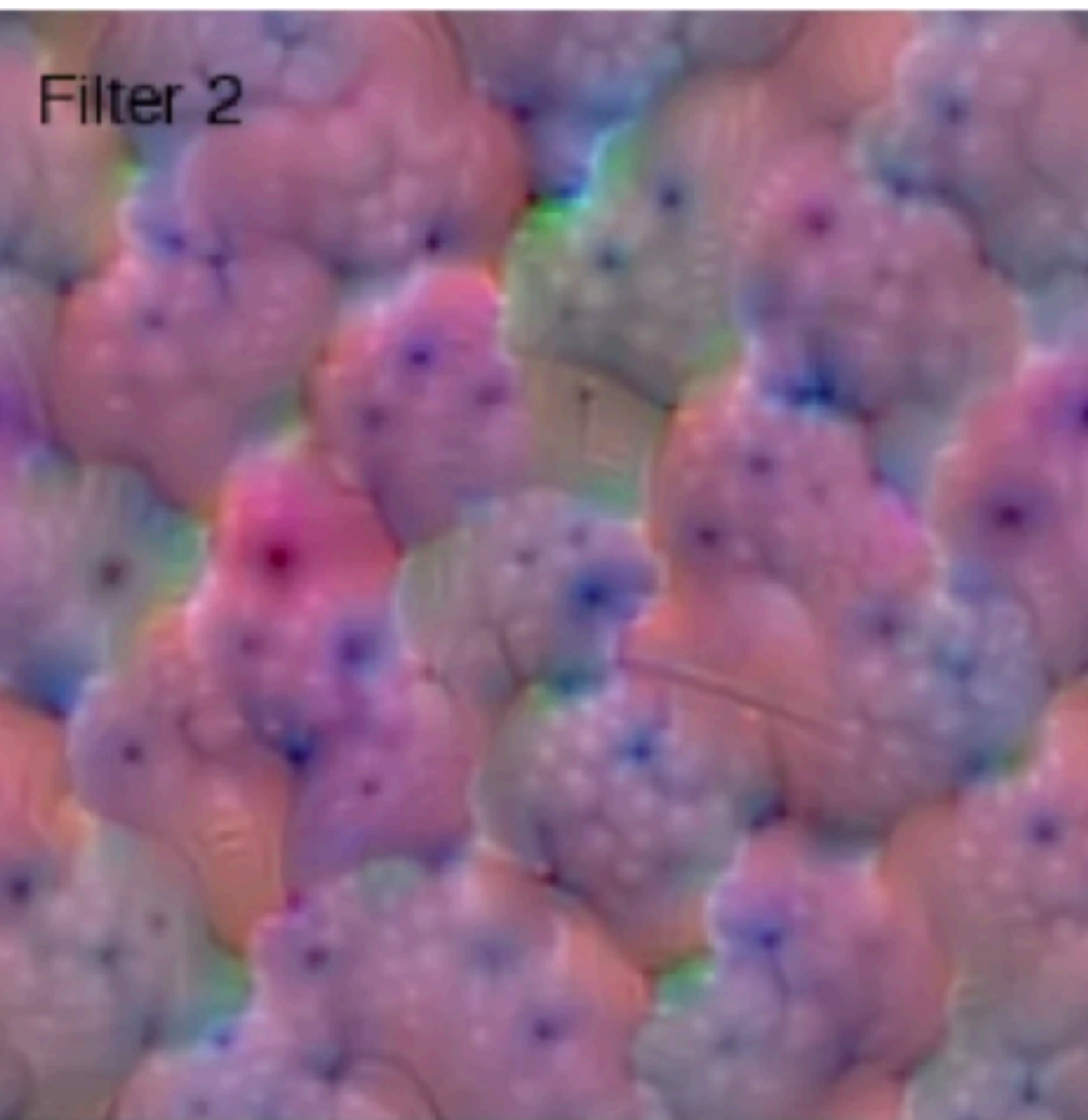
Features maps that maximize filter output



What is the input that maximizes a filter output?

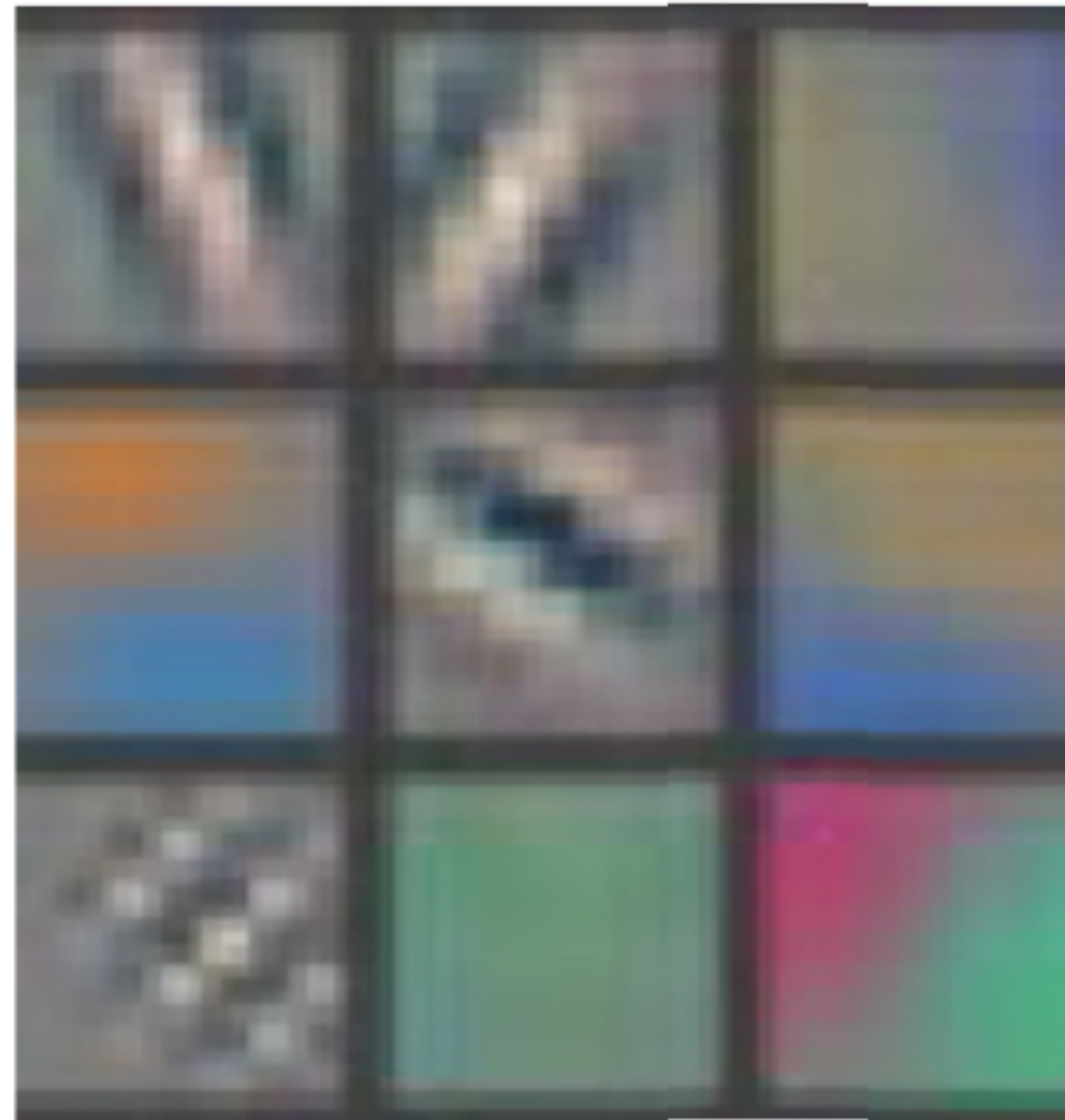


Features maps that maximize filter output
Can you interpret functionality of a filter?



3. Neurons are sensitive to edges and its orientation

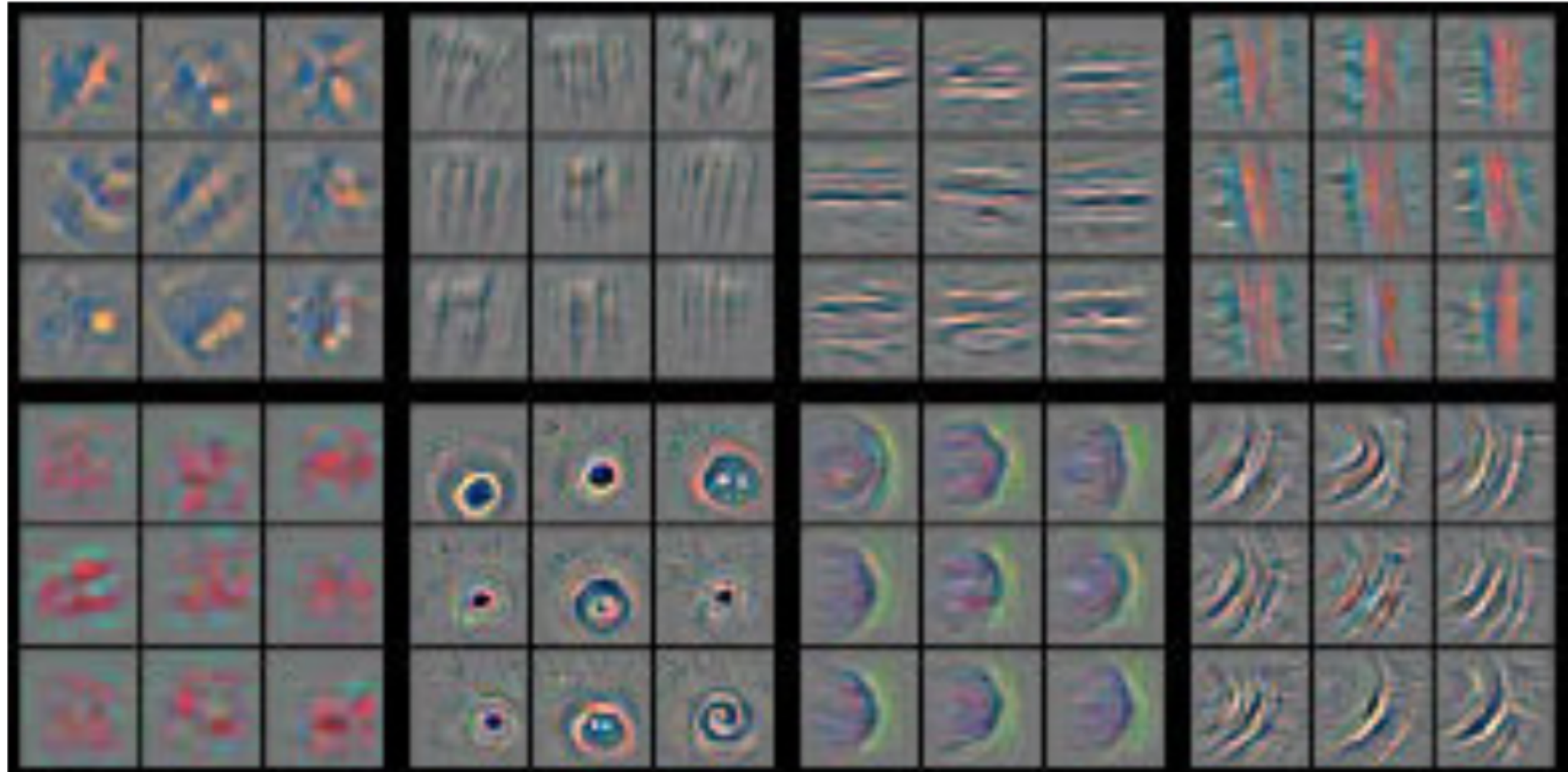
Inputs which maximized output of **layer 1**



[Zeiler and Fergus, ECCV, 2014]

3. Neurons are sensitive to edges and its orientation

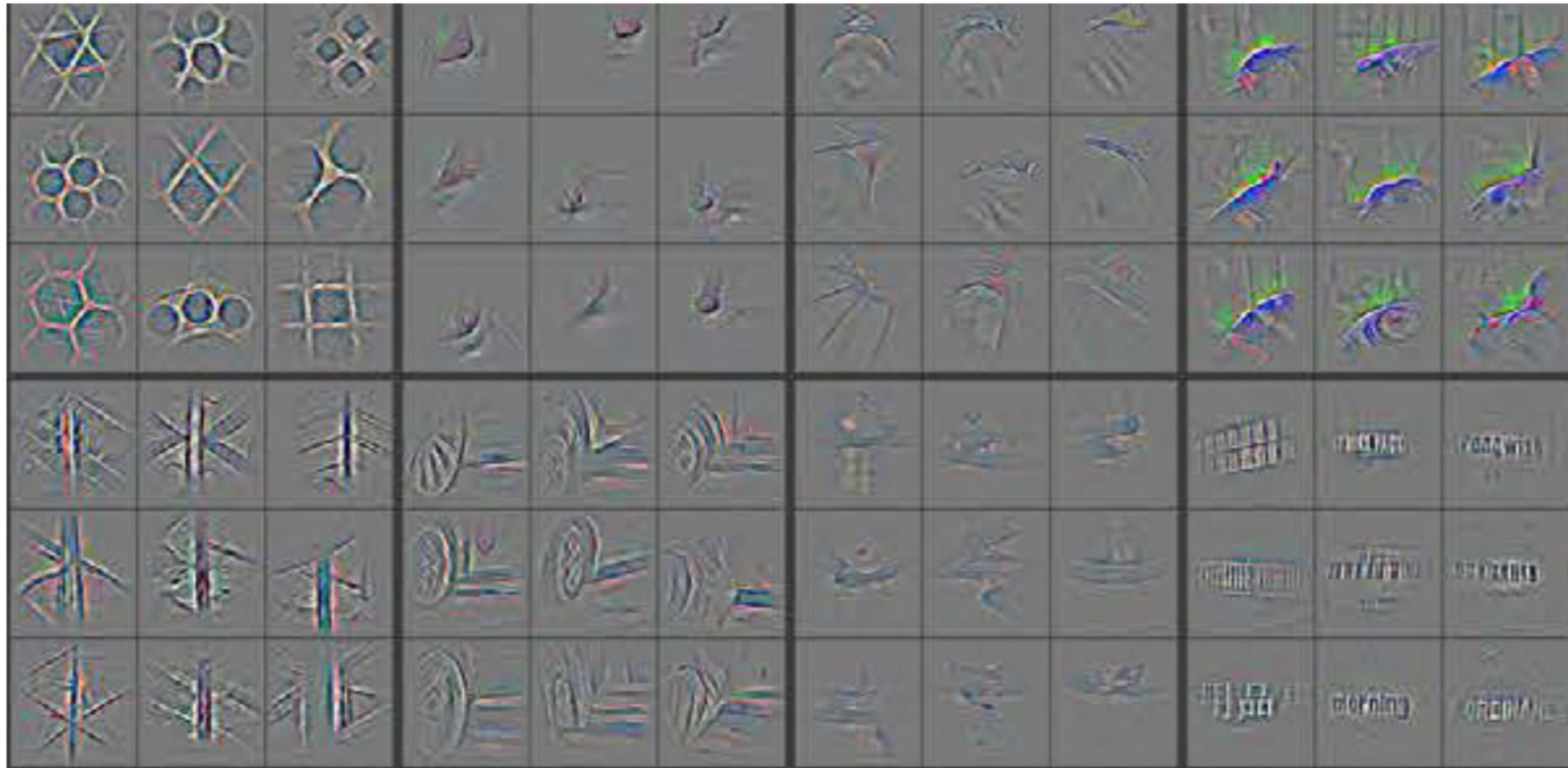
Inputs which maximized output of **layer 2**



[Zeiler and Fergus, ECCV, 2014]

3. Neurons are sensitive to edges and its orientation

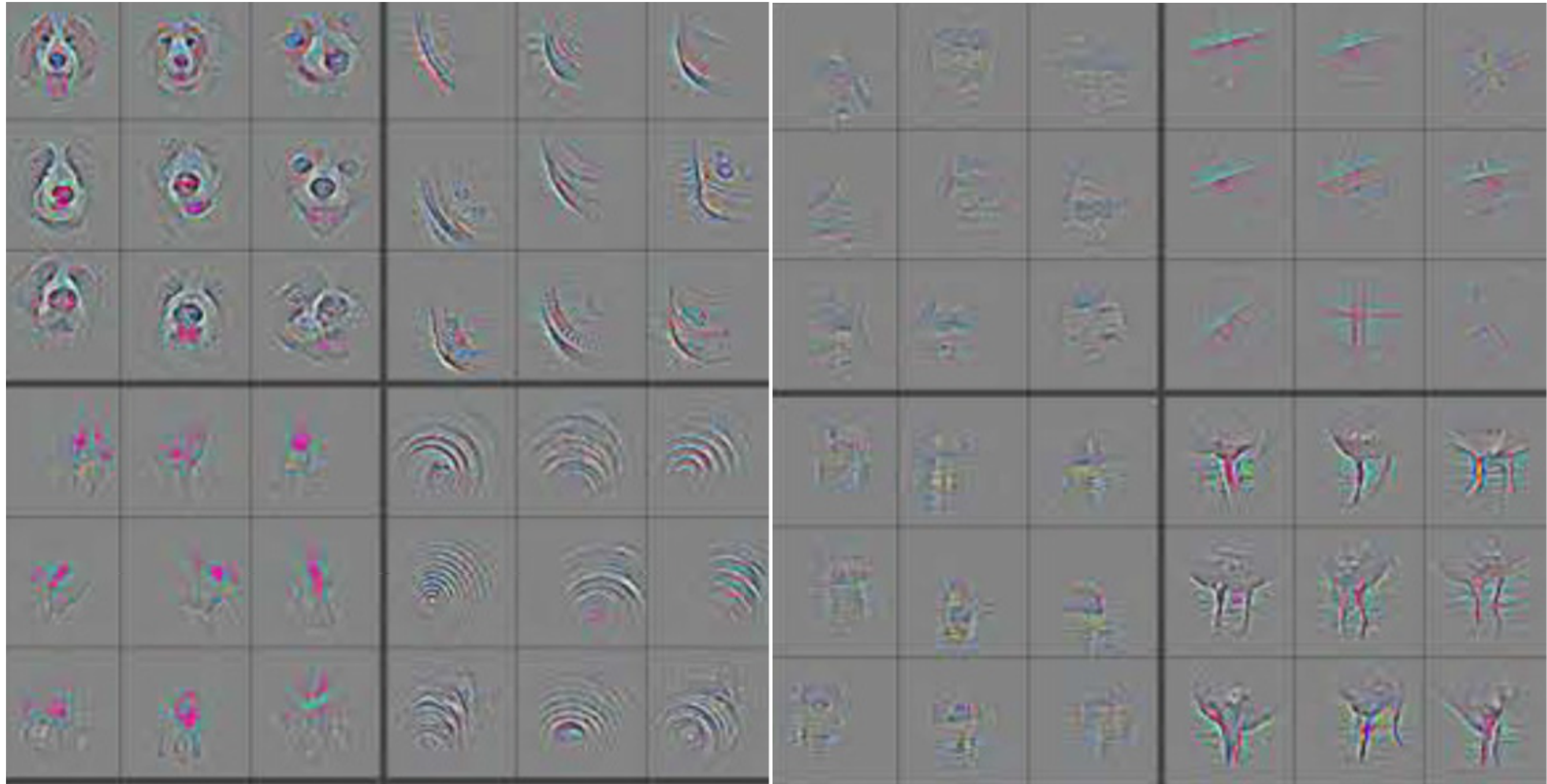
Inputs which maximized output of **layer 3**



[Zeiler and Fergus, ECCV, 2014]

3. Neurons are sensitive to edges and its orientation

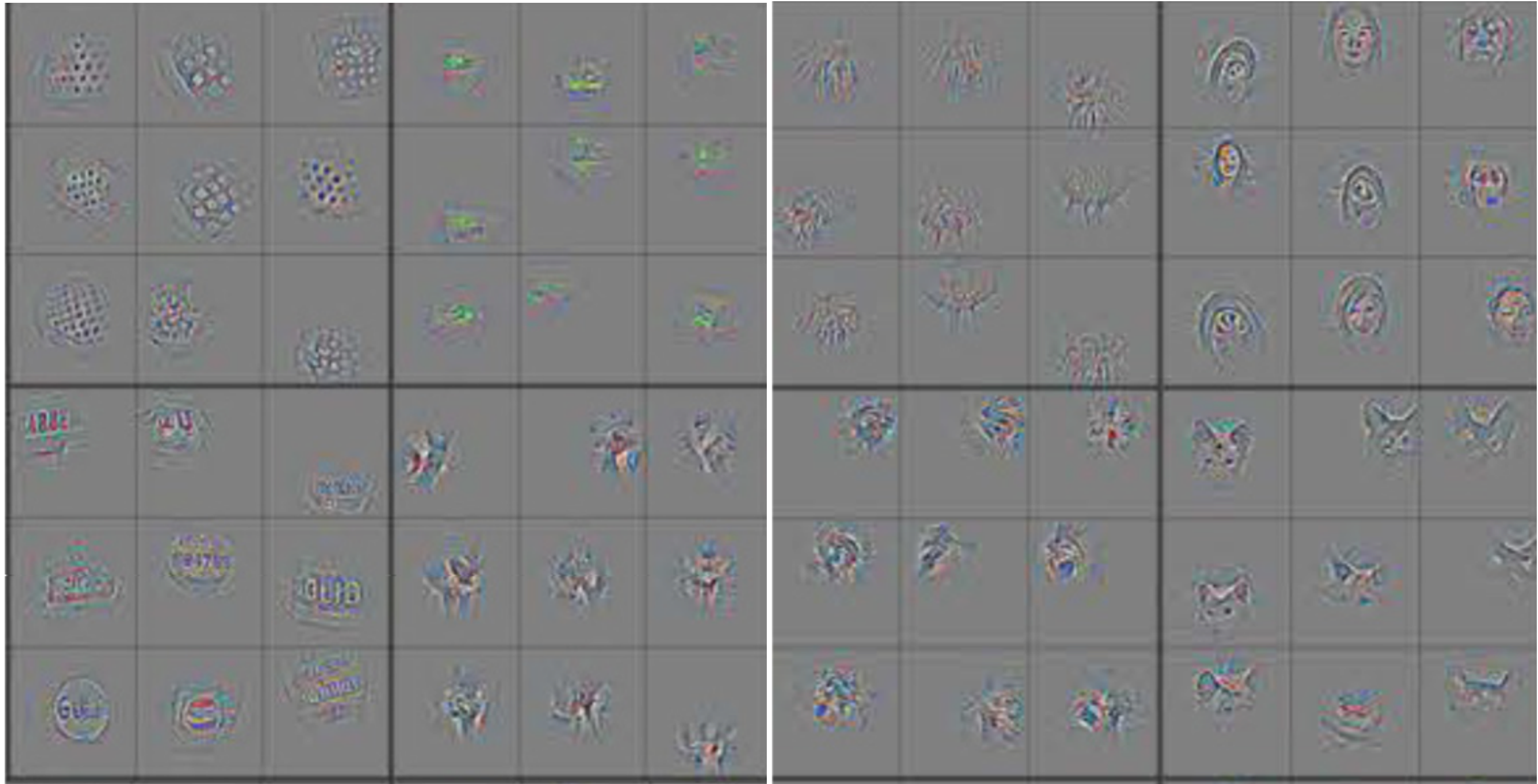
Inputs which maximized output of **layer 4**



[Zeiler and Fergus, ECCV, 2014]

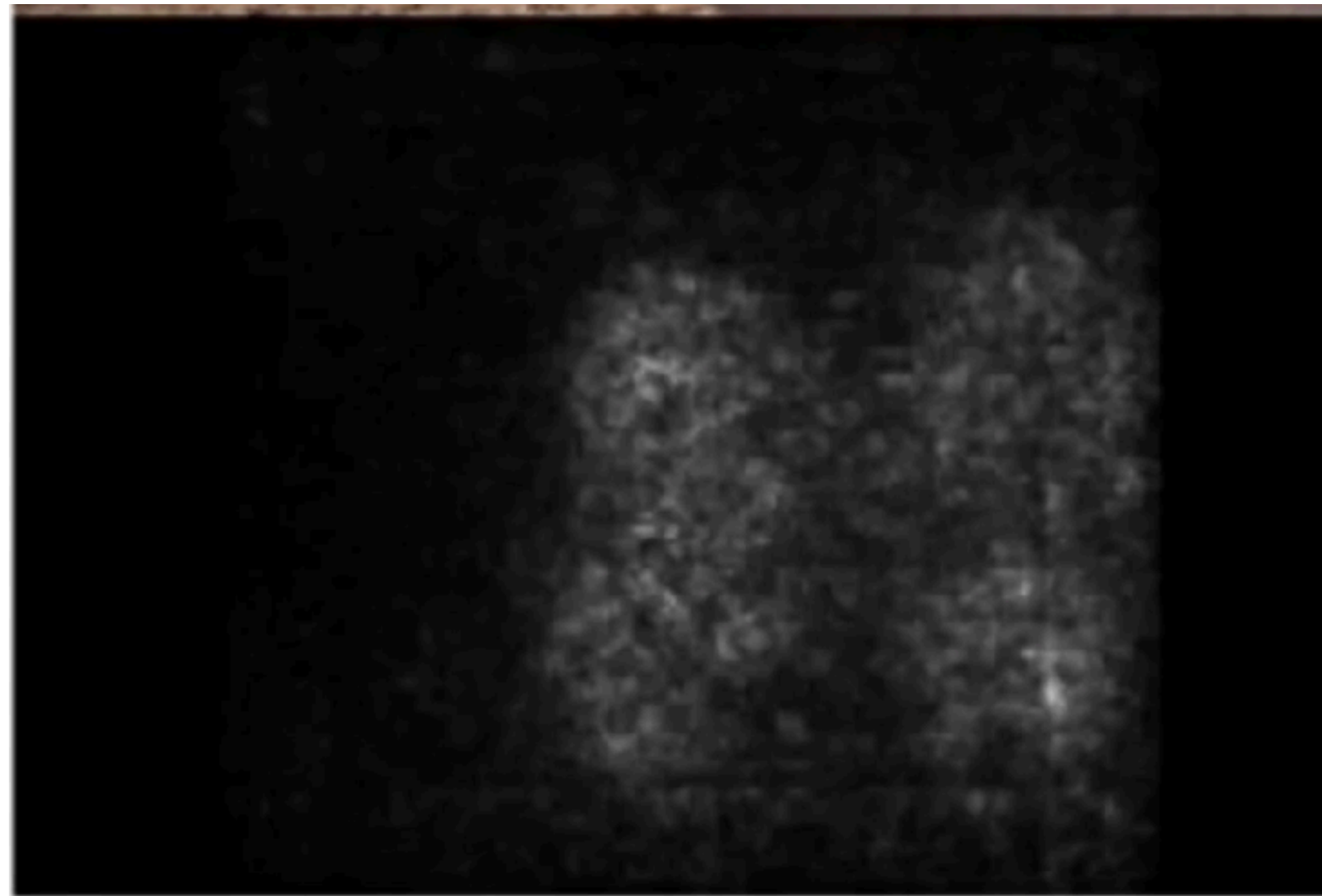
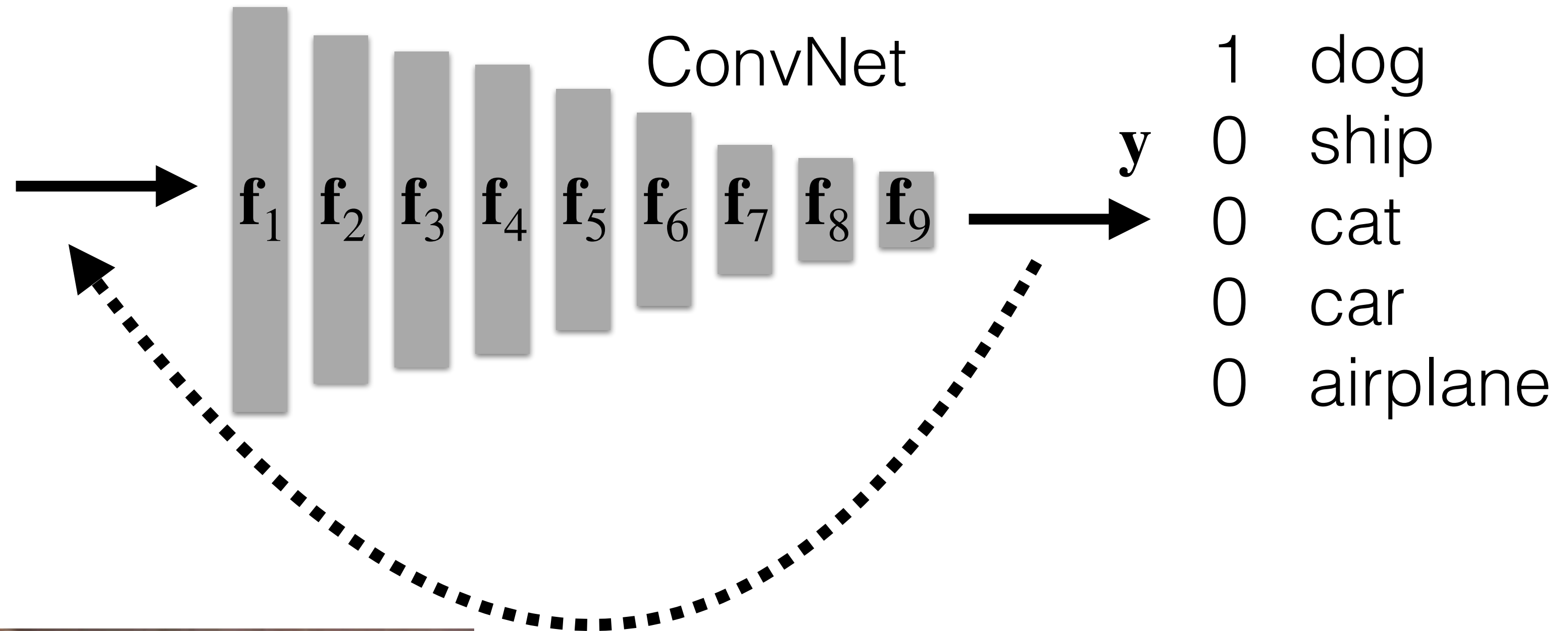
3. Neurons are sensitive to edges and its orientation

Inputs which maximized output of **layer 5**



[Zeiler and Fergus, ECCV, 2014]

Saliency maps

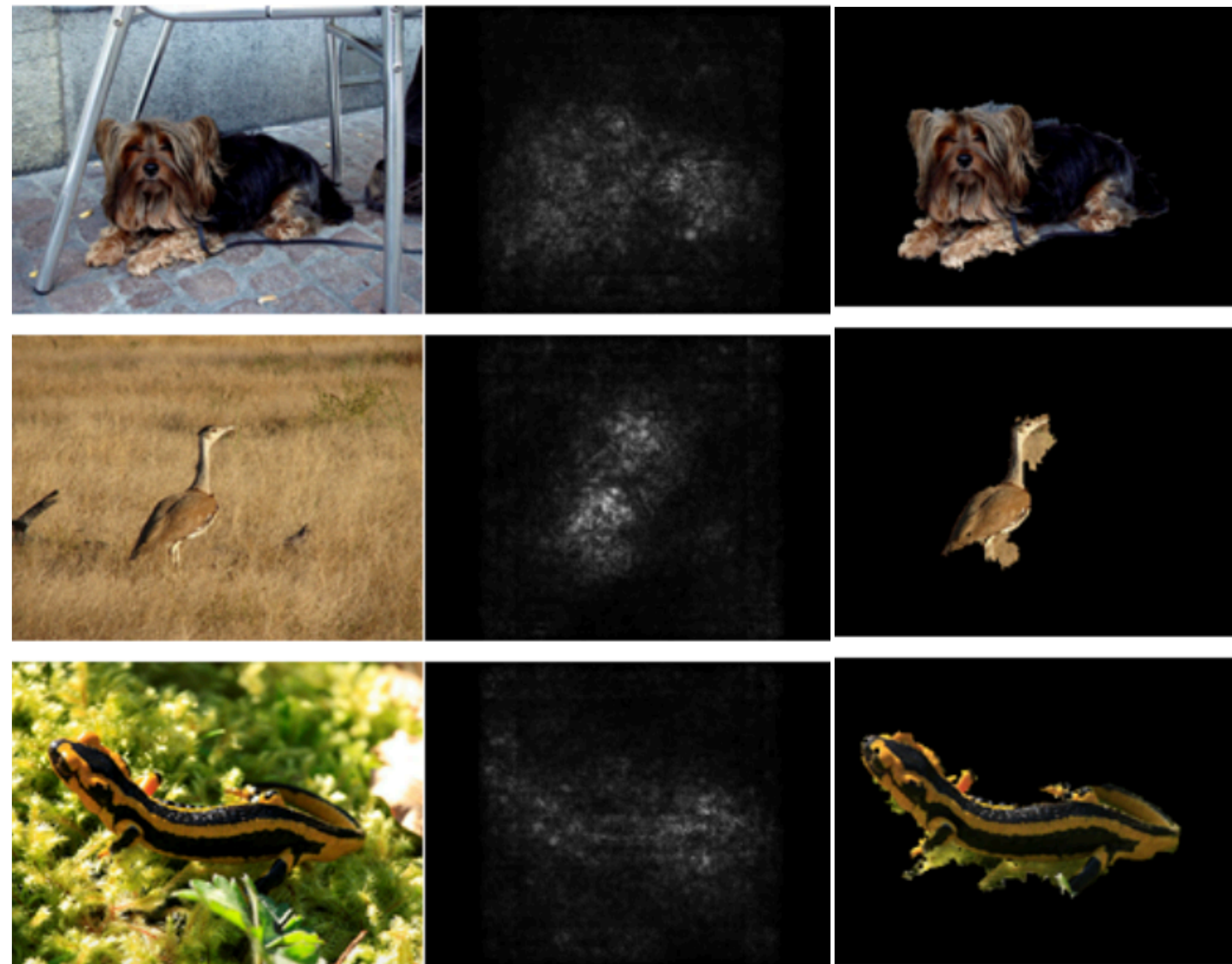
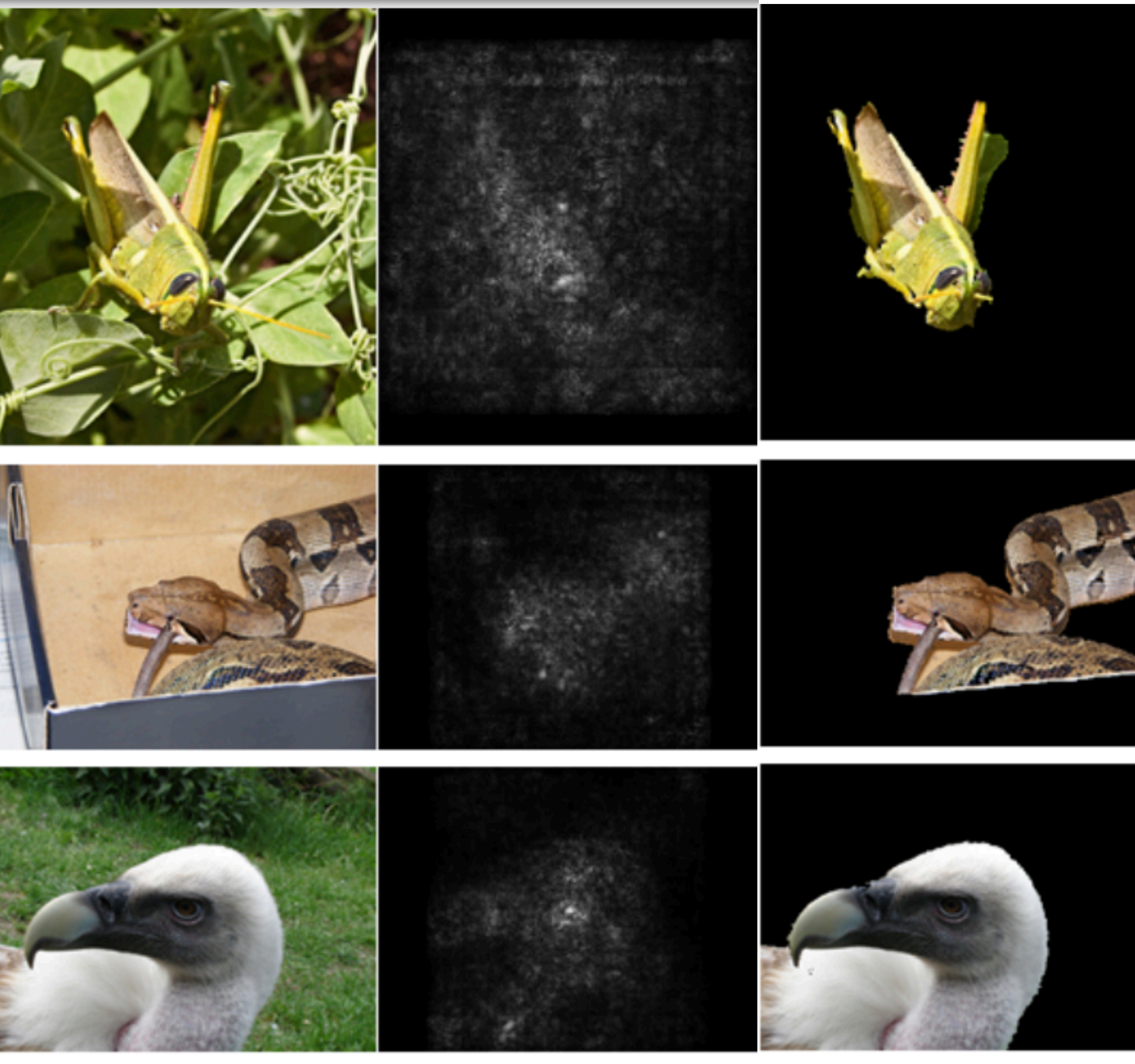


$$\frac{\partial y_1}{\partial \mathbf{x}}$$

What pixels contributed the most
for dog category on the output

Saliency maps

<https://arxiv.org/pdf/1312.6034.pdf>

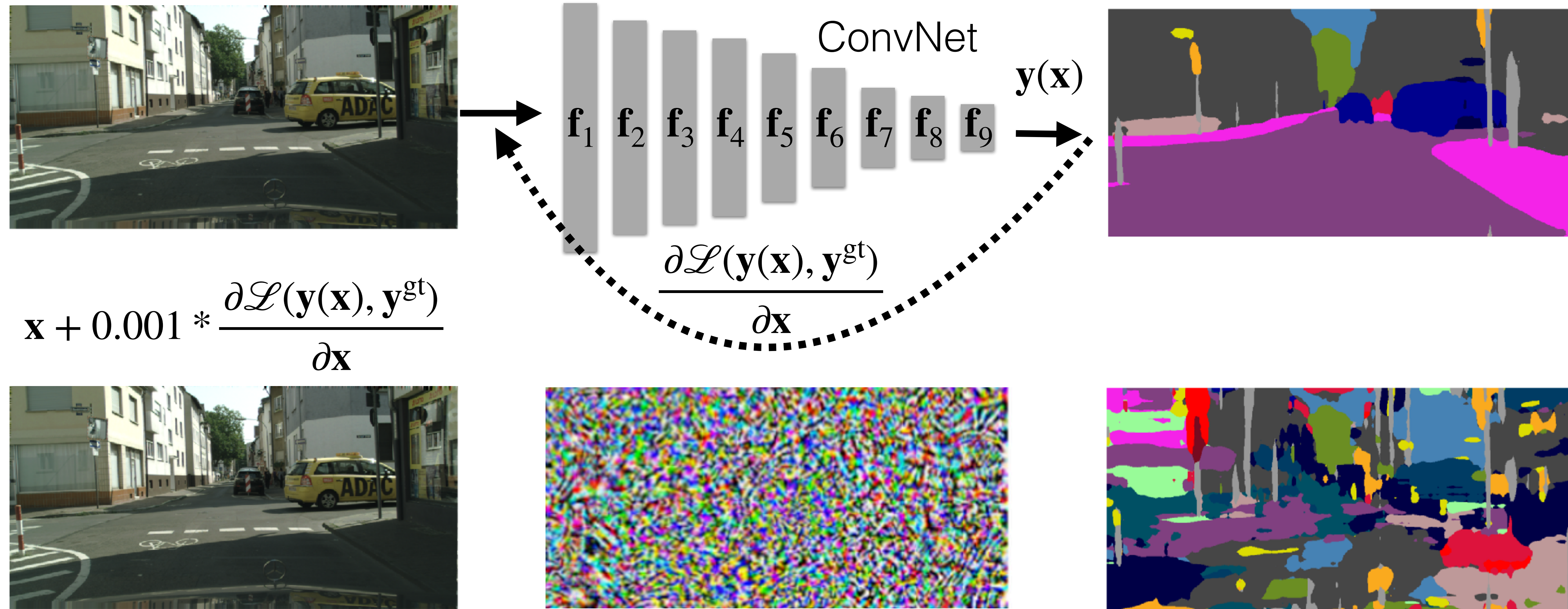


Adversarial attacks [Arnab CVPR 2018]

<https://github.com/hmph/adversarial-attacks>

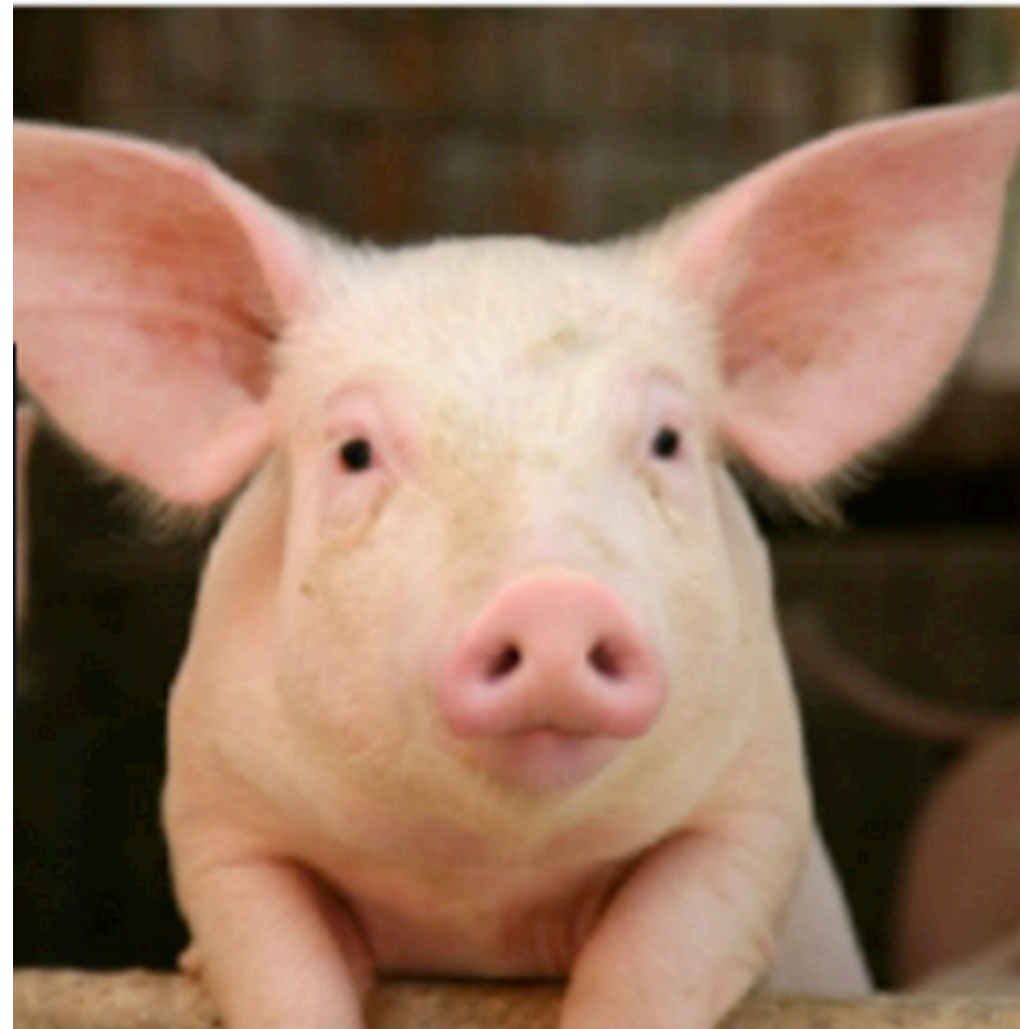
- Given trained network and an image, find the closest image on which the net fails

$\mathbf{x}$



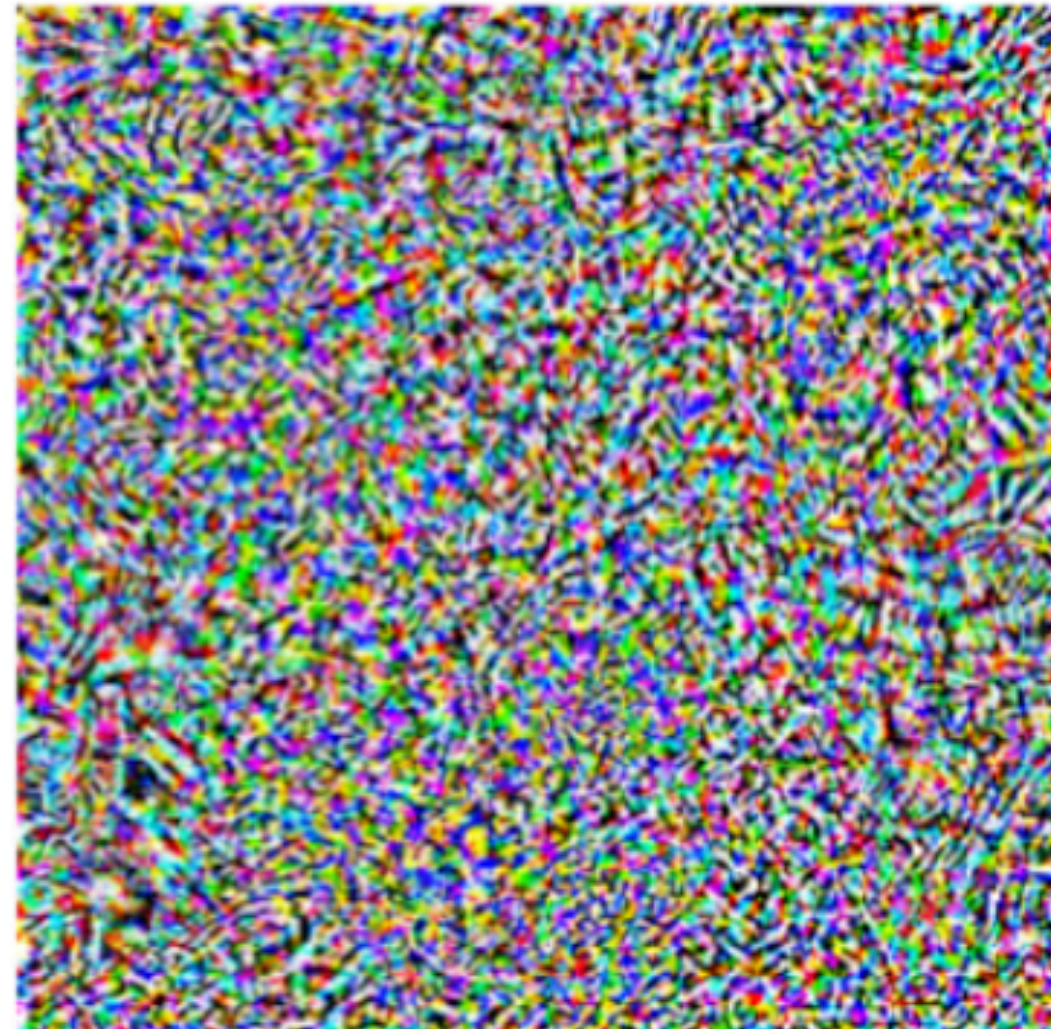
Adversarial attacks [Arnab CVPR 2018]
<https://github.com/hmph/adversarial-attacks>

“pig”



$\mathbf{x}$

+ 0.005 x



=

“airliner”



$$\frac{\partial \mathcal{L}(\mathbf{y}(\mathbf{x}), \mathbf{y}^{\text{gt}})}{\partial \mathbf{x}}$$

$$\mathbf{x} + 0.005 * \frac{\partial \mathcal{L}(\mathbf{y}(\mathbf{x}), \mathbf{y}^{\text{gt}})}{\partial \mathbf{x}}$$

Adversarial noise = direction in image domain that increases the loss the most

High frequency noise

The space is high-dimensional (10+ dimensions)

Access to network architecture + input image => physical attacks unsuccessful

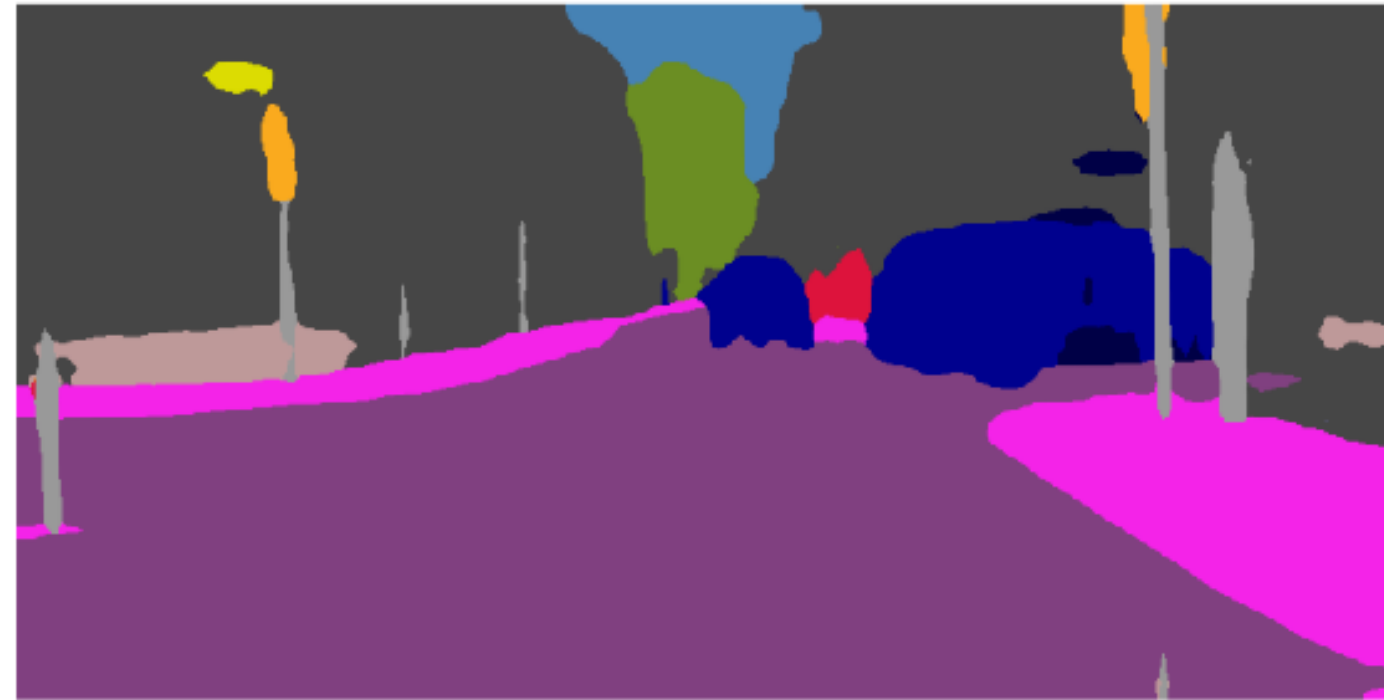
Adversarial attacks [Arnab CVPR 2018]

<https://github.com/hmph/adversarial-attacks>

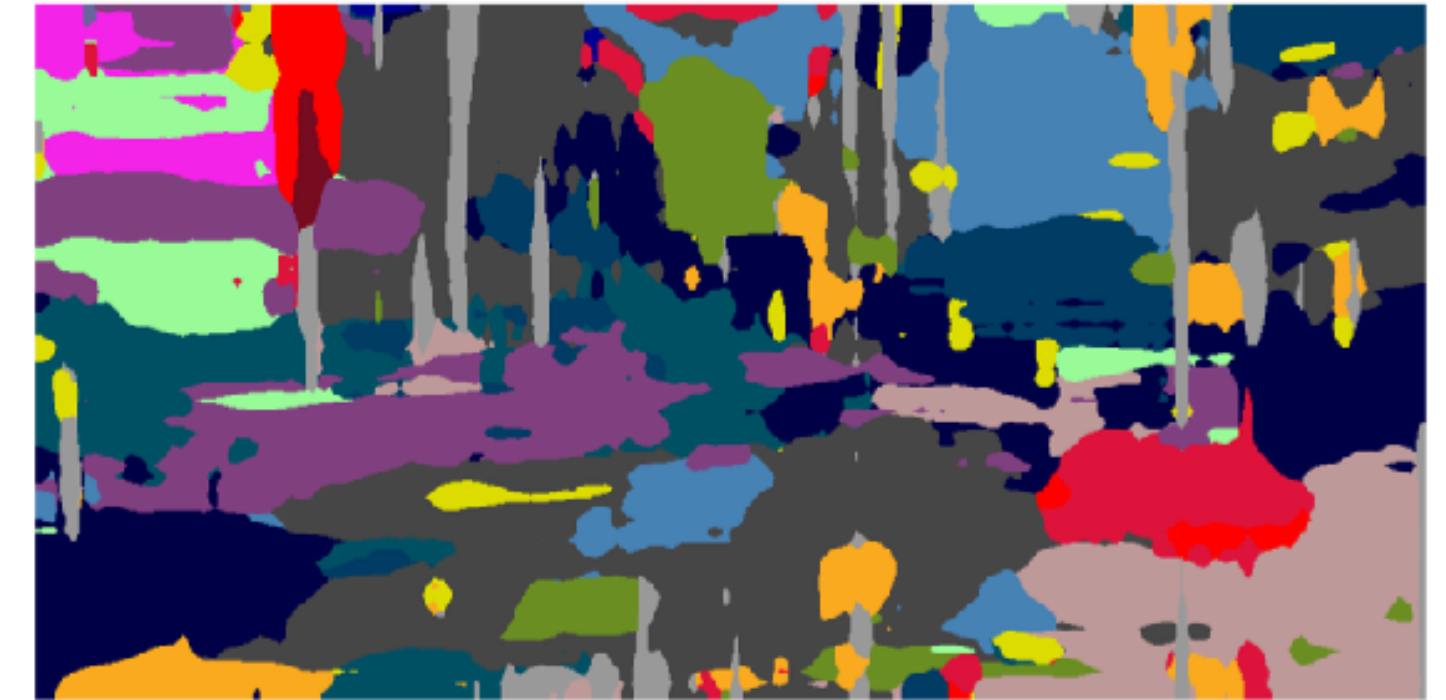
- Given trained network and an image, find the closest image on which the net fails



Input image



Original prediction



Adversarial prediction



Input image



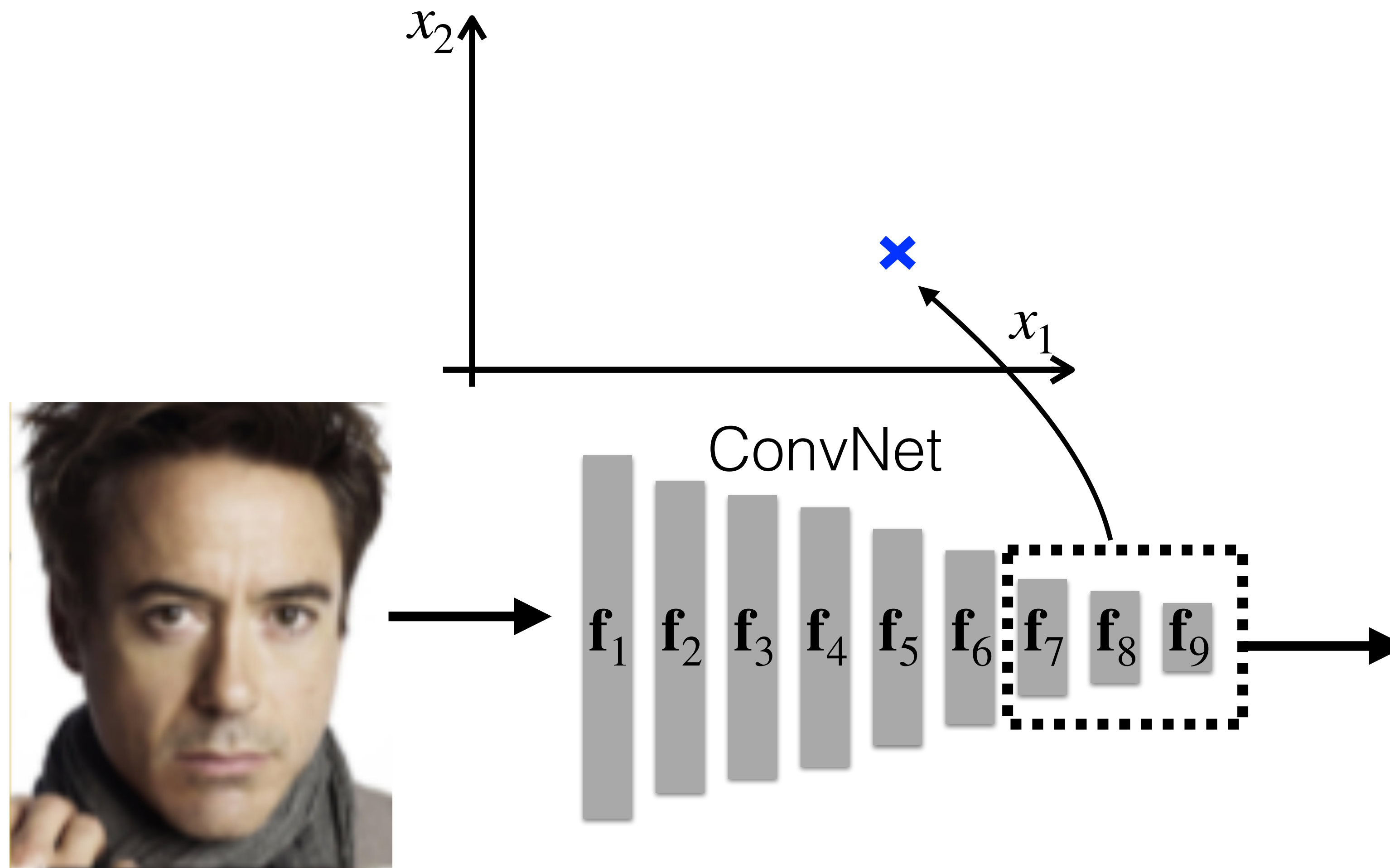
Original prediction



Adversarial prediction

Deep Feature interpolations [Upchurch CVPR 2017]

<https://arxiv.org/pdf/1611.05507.pdf>

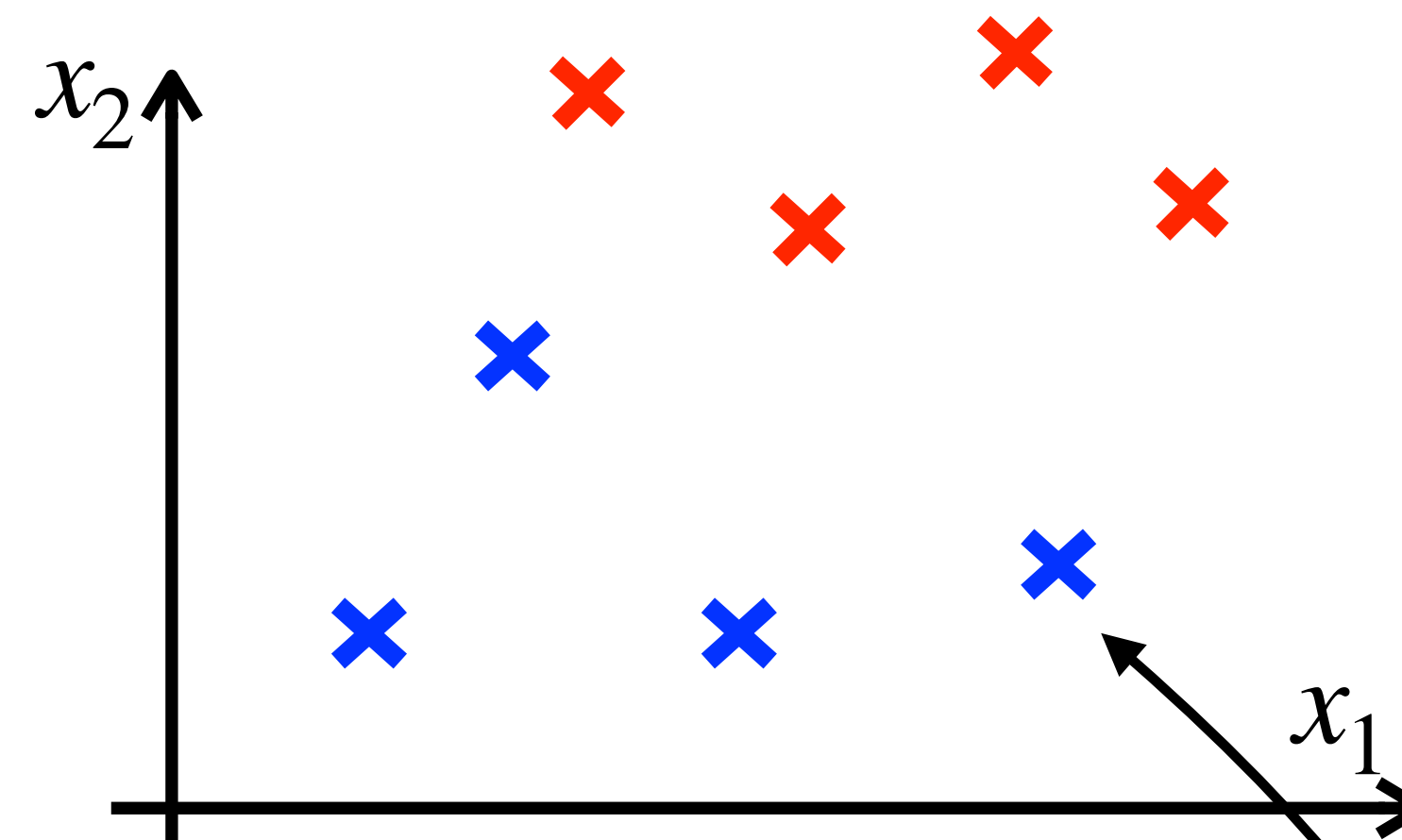


Deep ConvNet project images into a meaningful low-dimensional representation

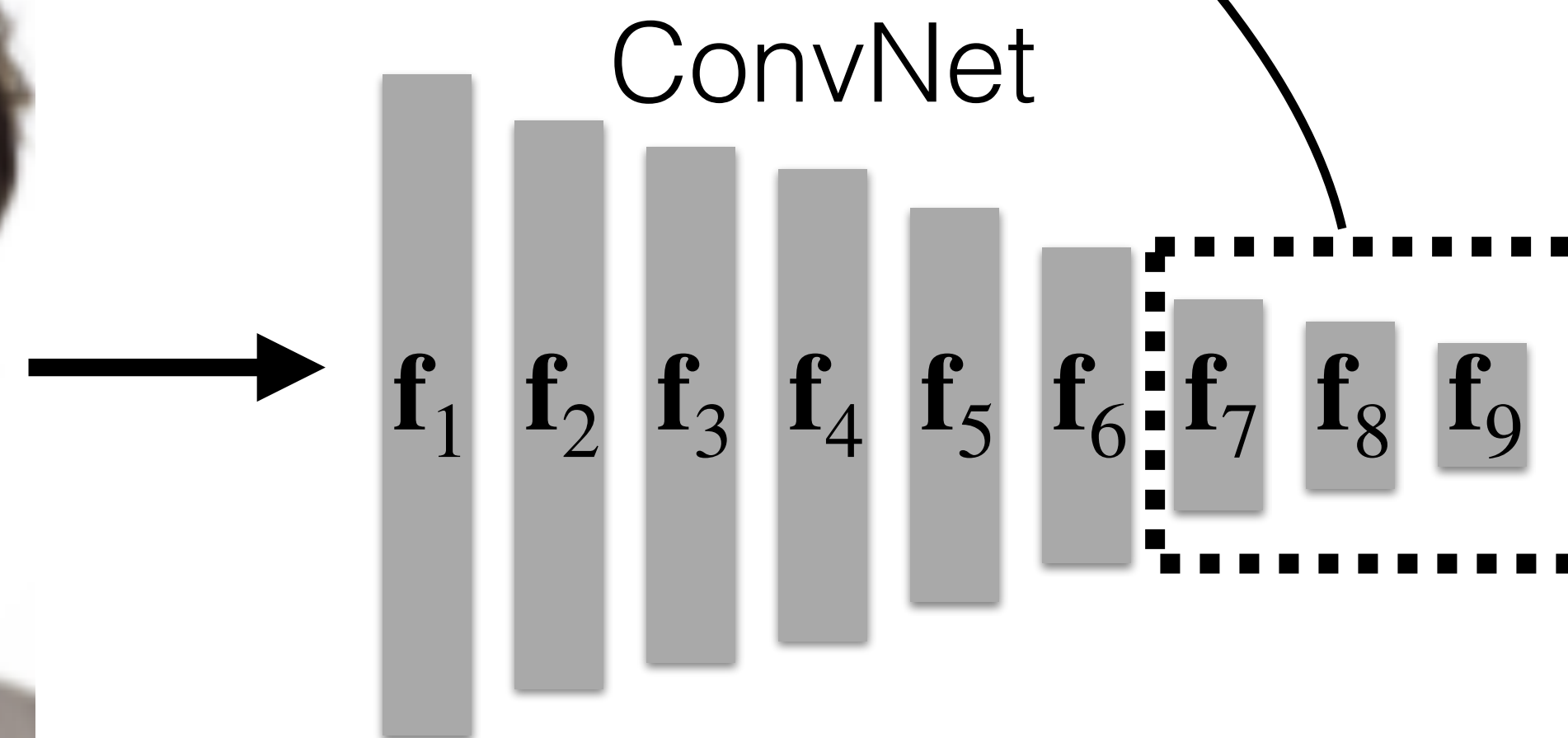
Deep Feature interpolations [Upchurch CVPR 2017]

<https://arxiv.org/pdf/1611.05507.pdf>

✗
faces
with
facial hair



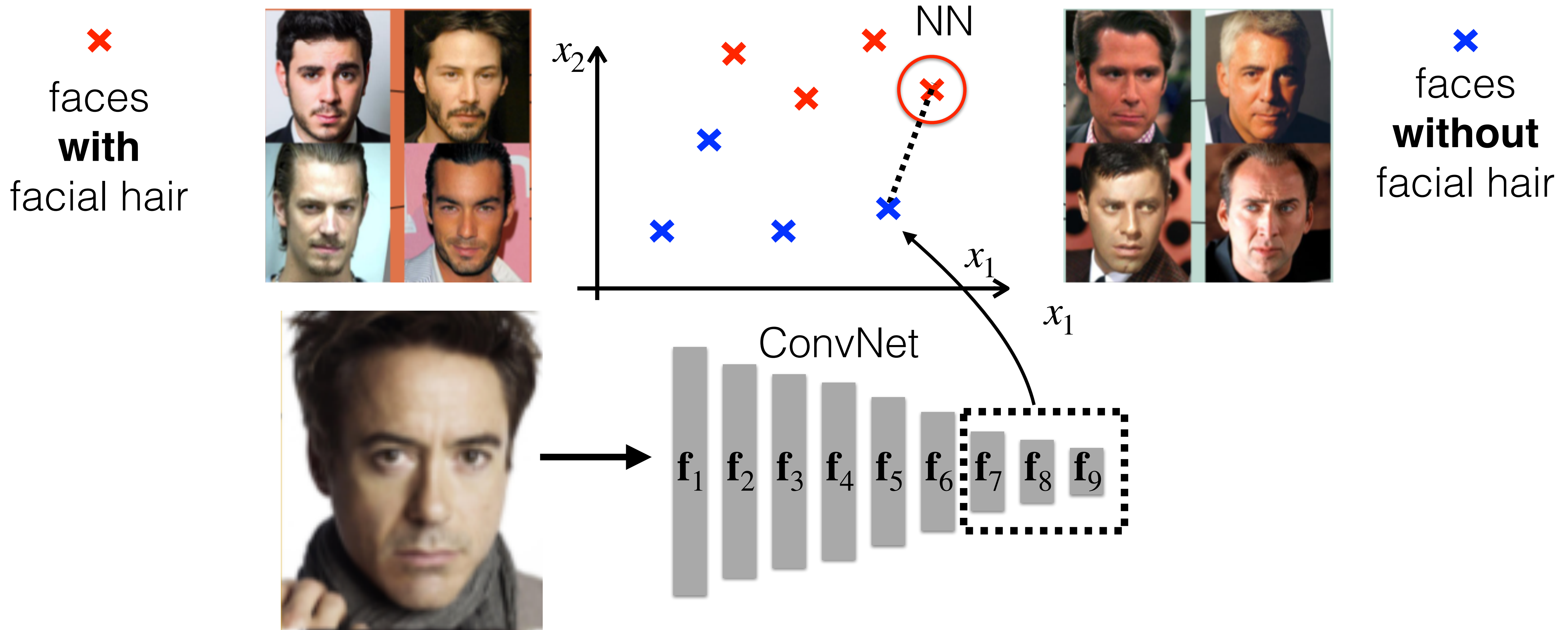
✕
faces
without
facial hair



Deep ConvNet project images into a meaningful low-dimensional representation

Deep Feature interpolations [Upchurch CVPR 2017]

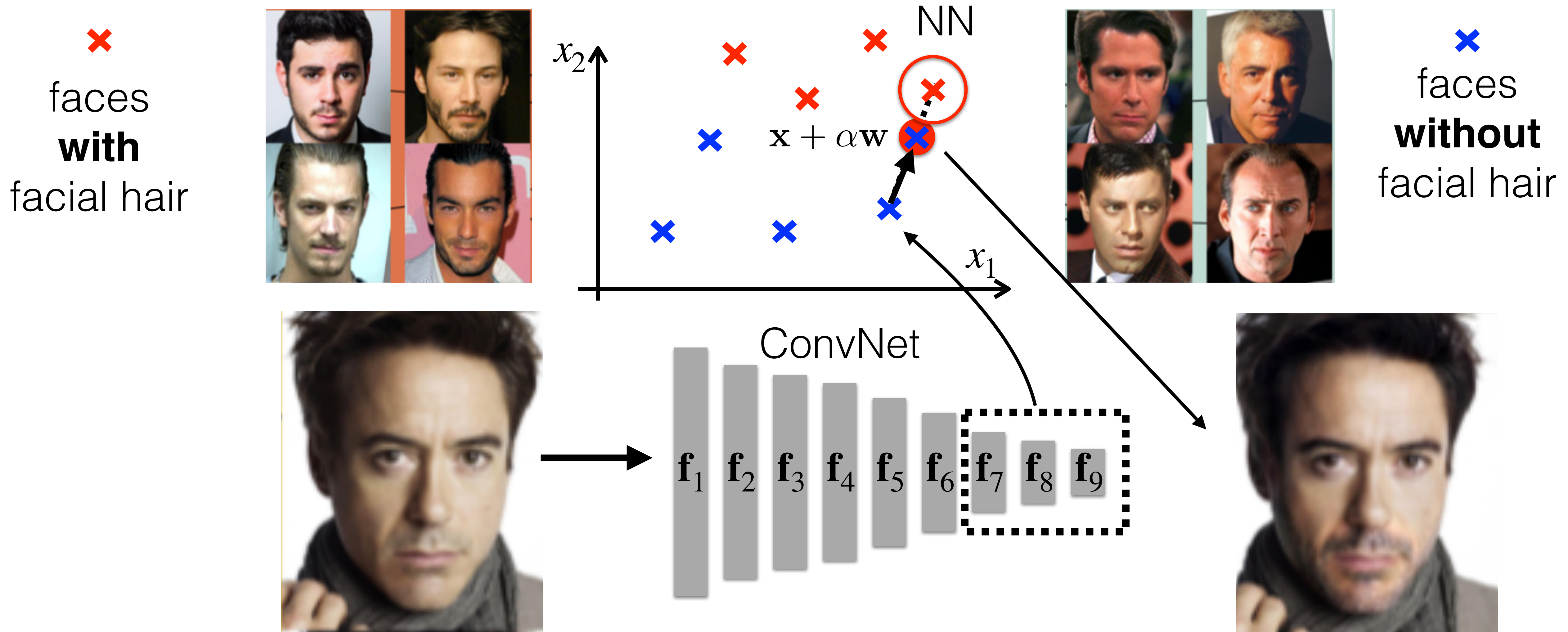
<https://arxiv.org/pdf/1611.05507.pdf>



Deep ConvNet project images into a meaningful low-dimensional representation

Deep Feature interpolations [Upchurch CVPR 2017]

<https://arxiv.org/pdf/1611.05507.pdf>



Deep ConvNet project images into a meaningful low-dimensional representation



Deep Feature interpolations [Upchurch CVPR 2017]

<https://arxiv.org/pdf/1611.05507.pdf>



Shallow architecture

input/measurements



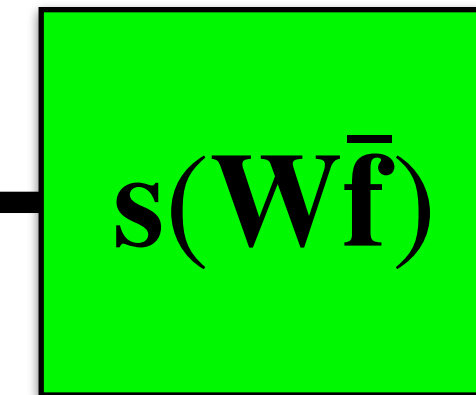
$\mathbf{x}$

features



$\mathbf{f}$

linear classifier



$s(\mathbf{W}\bar{\mathbf{f}})$

$\mathbf{y}$

output

0.2 dog

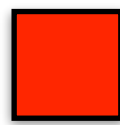
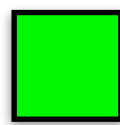
0.1 ship

0.5 cat

0.0 car

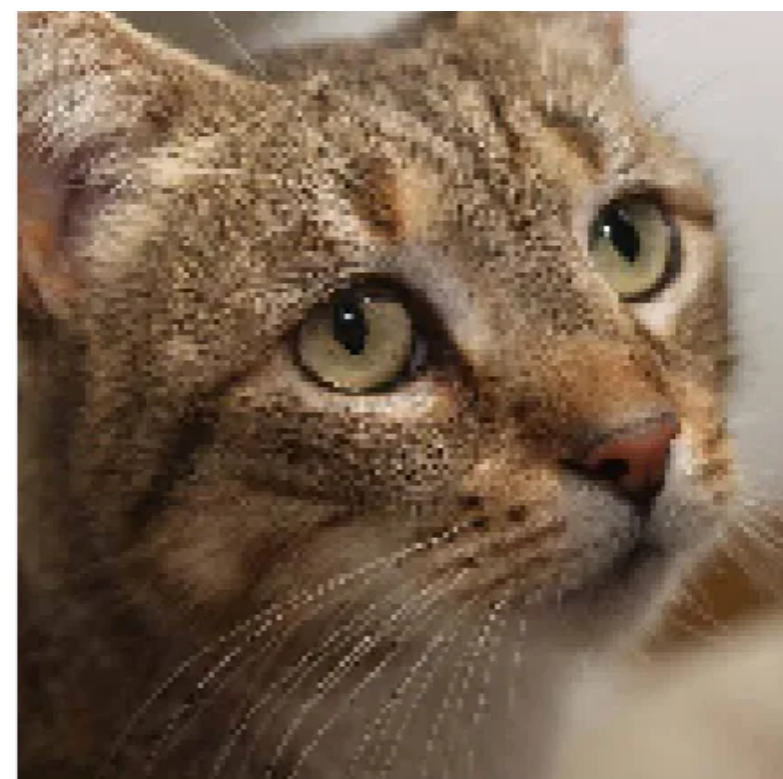
0.1 airplane

Manually estimated adhoc features
(pixels, color histograms, gradients, ...)

 manually estimated
 trained

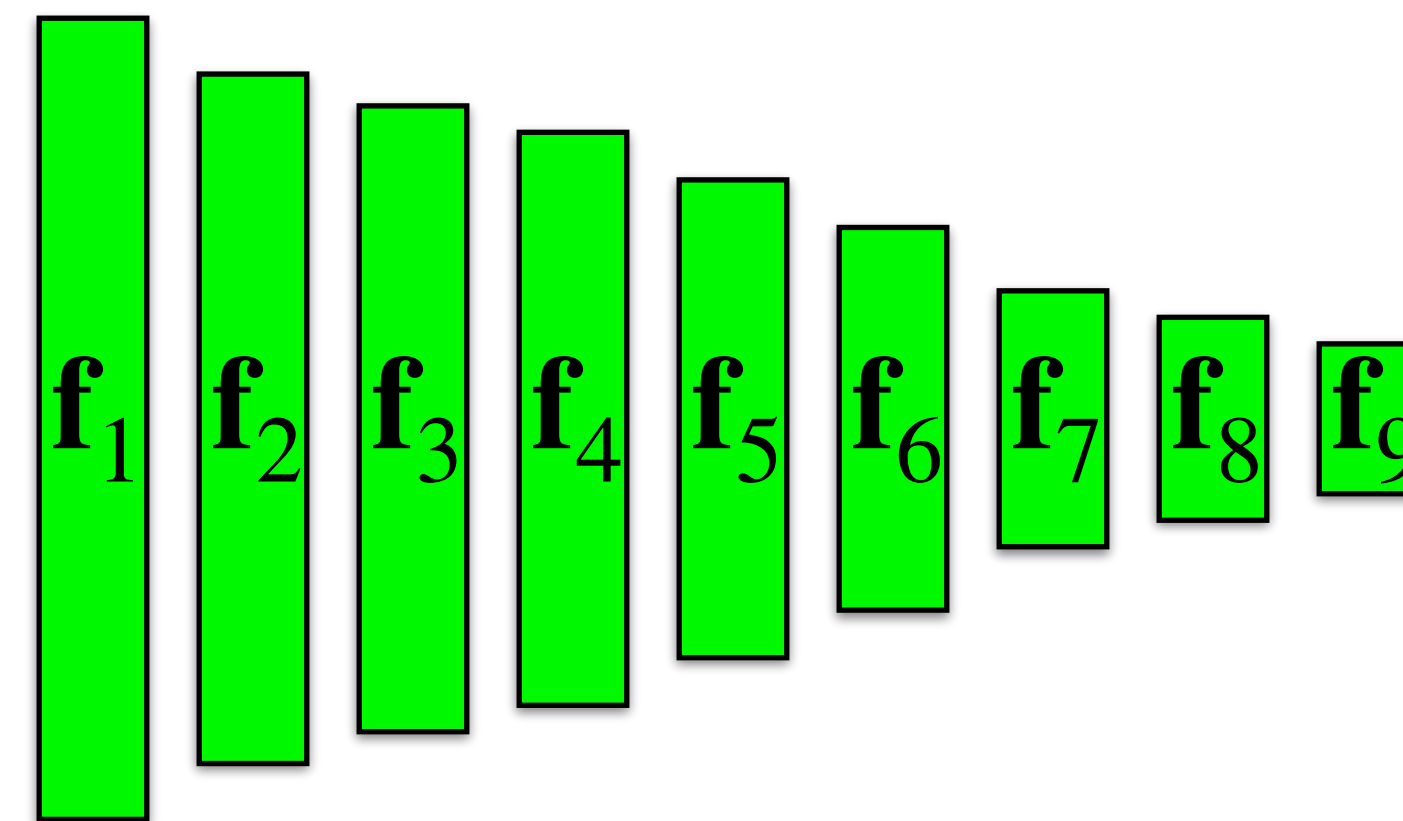
Deep architecture

input/measurements



$\mathbf{x}$

convolutional network



$\mathbf{f}_1$

$\mathbf{f}_2$

$\mathbf{f}_3$

$\mathbf{f}_4$

$\mathbf{f}_5$

$\mathbf{f}_6$

$\mathbf{f}_7$

$\mathbf{f}_8$

$\mathbf{f}_9$

$\mathbf{y}$

output

0.2 dog

0.1 ship

0.5 cat

0.0 car

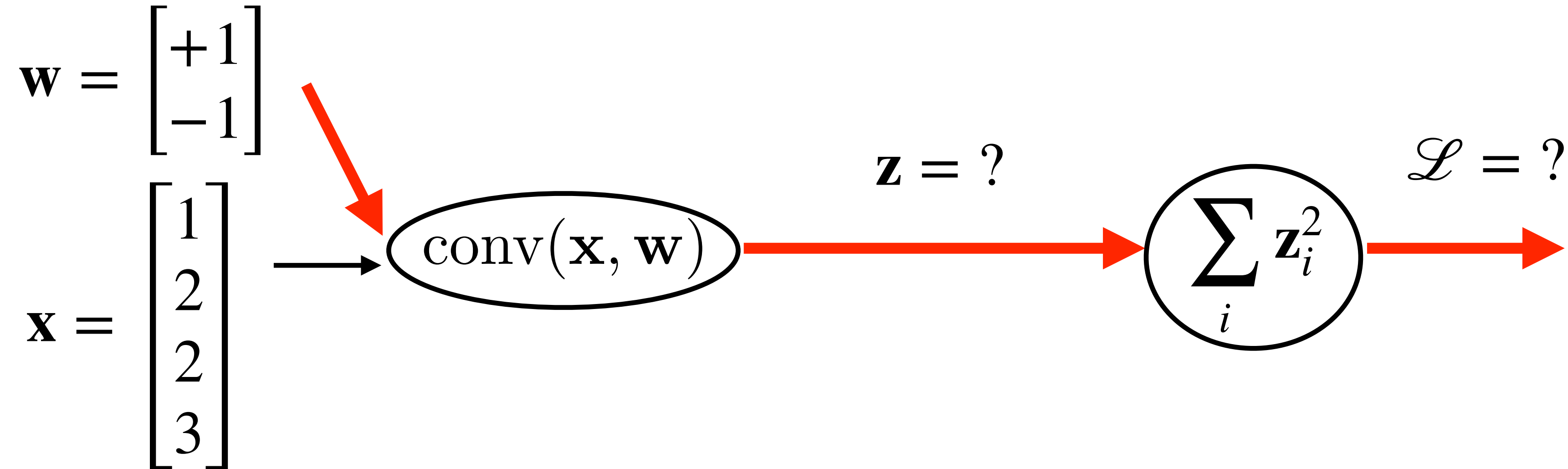
0.1 airplane

Trained features

Test T1 competencies

- ConvNet/Layer feed-forward pass
- ConvNet/Layer backpropagation
- Meaning of kernels, feature maps, stride, dilation, padding, ...

Midterm test 2023



$$\frac{\partial \mathbf{z}}{\partial \mathbf{w}} = \text{???}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} =$$