Single & Multi-Channel Convolution

Instructor - Simon Lucey

16-720B - Computer Vision

Carnegie Mellon
THE ROBOTICS INSTITUTE


DON'T PANIC

Today

- Why Convolution?
- Reminder: Single Channel Convolution
- 2D Convolution & Separable Filters
- Multi-Channel Convolution

3 Biggest Problems in Computer Vision?

3 Biggest Problems in Computer Vision?

REGISTRATION

REGISTRATION

REGISTRATION

(Prof. Takeo Kanade - Robotics Institute - CMU.)

What is Registration?

“Source Image”

“Template Image”

What is Registration?

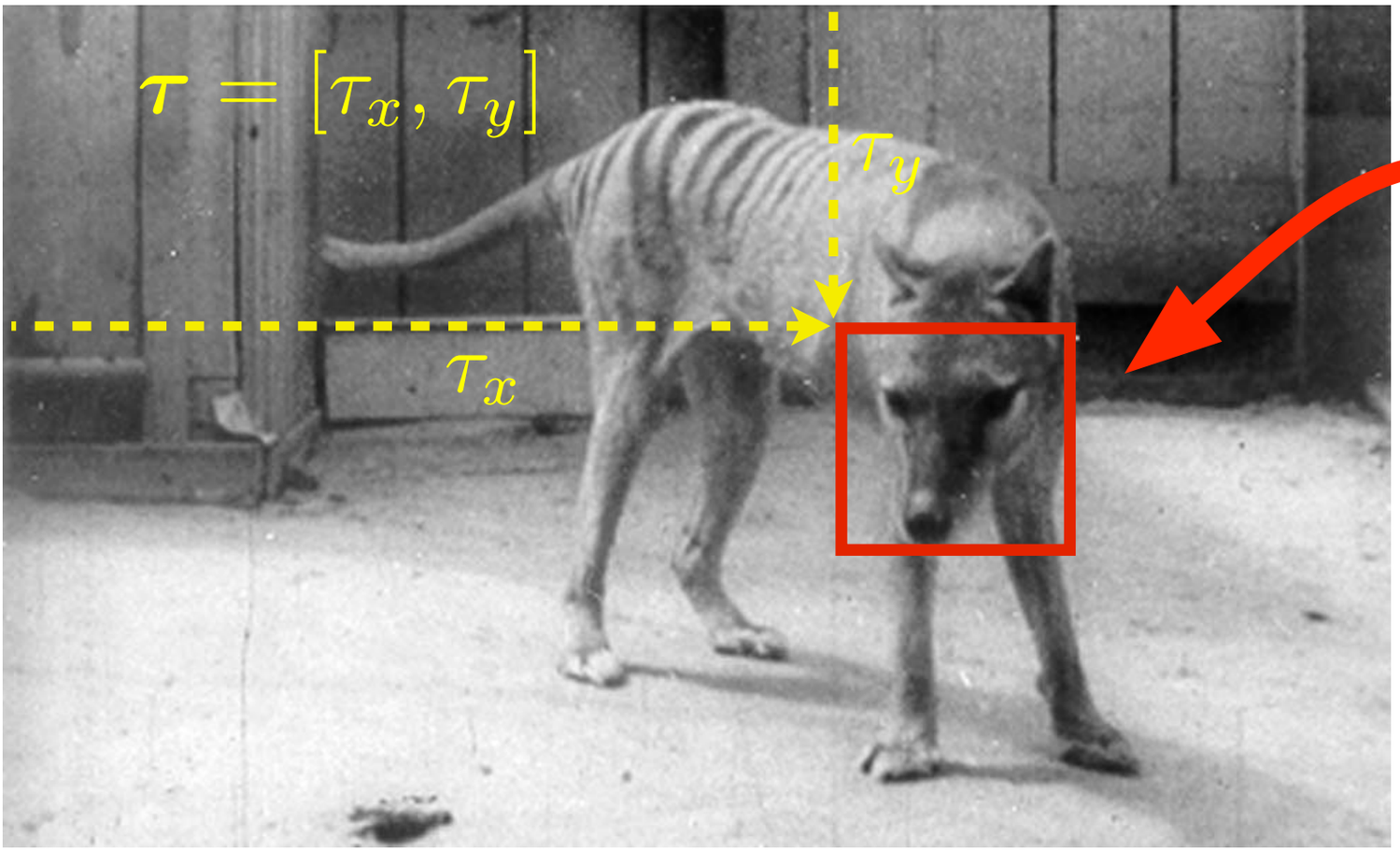
“Source Image”

$\mathbf{X}$

“Template Image”

$\mathbf{H}$

What is Registration?



“Source Image”

$\mathbf{X}$

“Template Image”

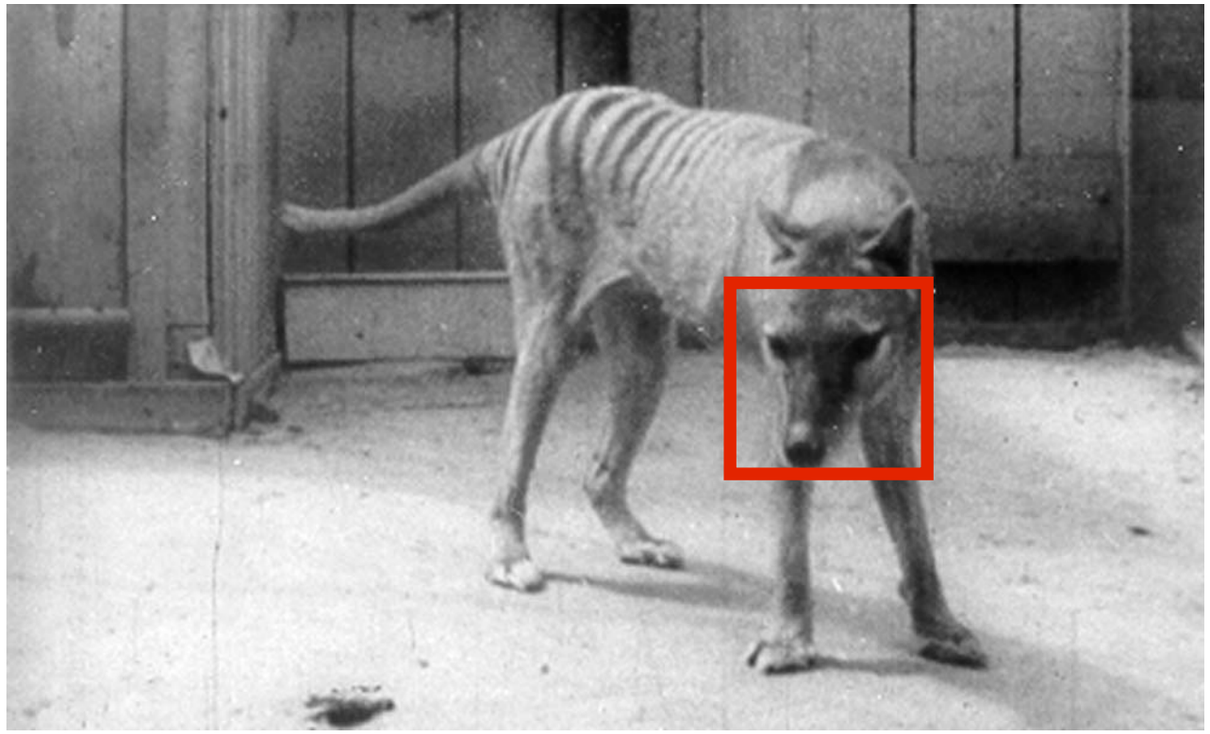
$\mathbf{H}$

$$\mathbf{Y} = \mathbf{X} * \mathbf{H}$$

$$\mathbf{Y} = \mathbf{X} * \mathbf{H}$$

$$\mathbf{Y} = \mathbf{X} * \mathbf{H}$$

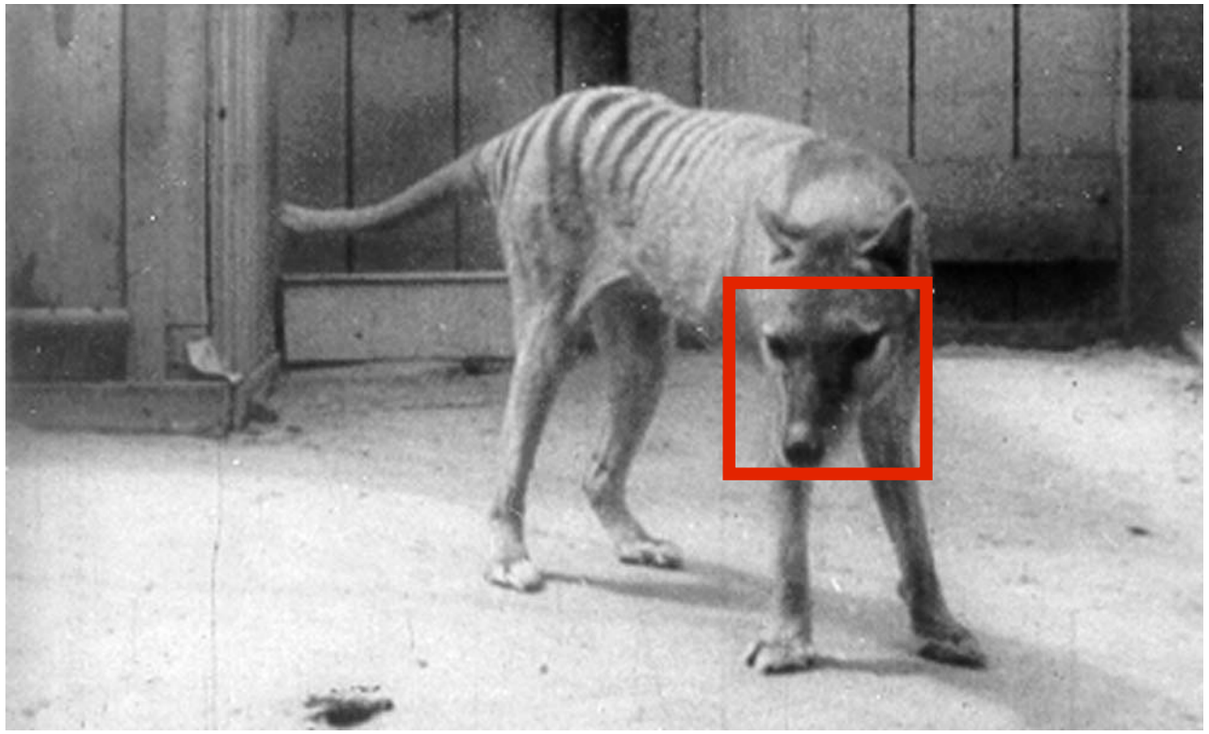
```
>>> from scipy.ndimage import convolve as imconv
>>> Y = imconv(X,H)
```



X

*

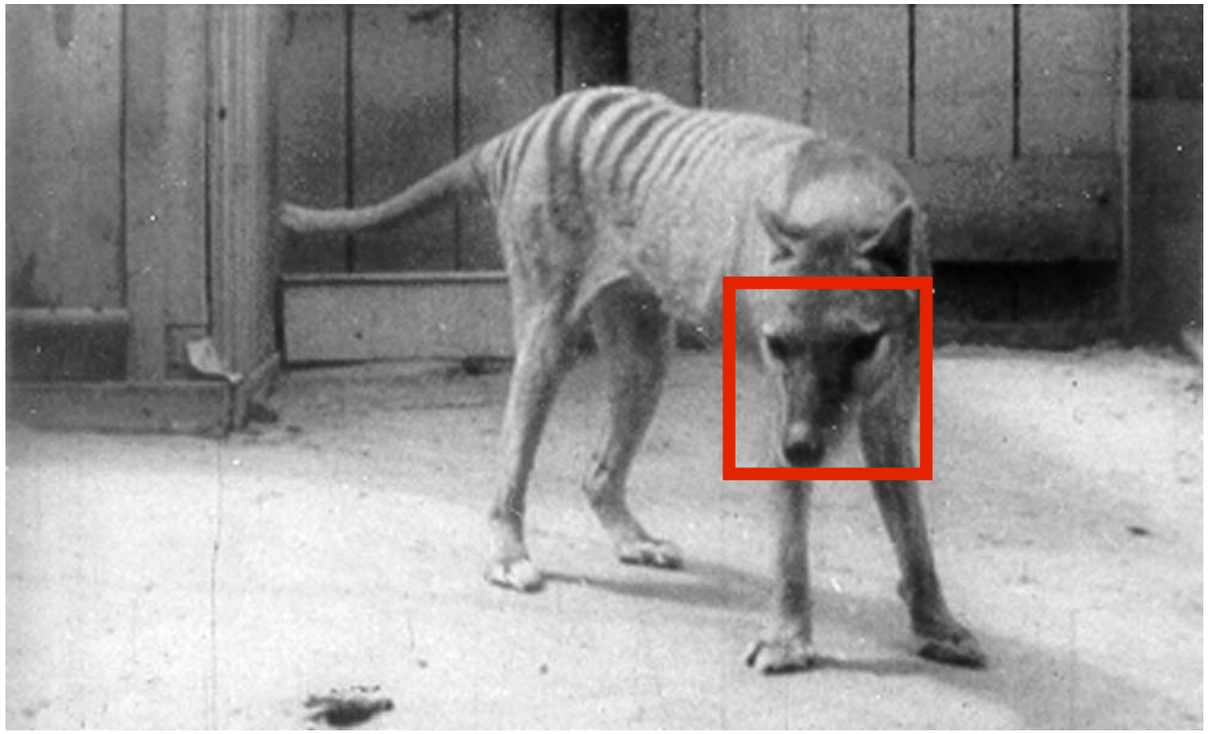
H



X

*

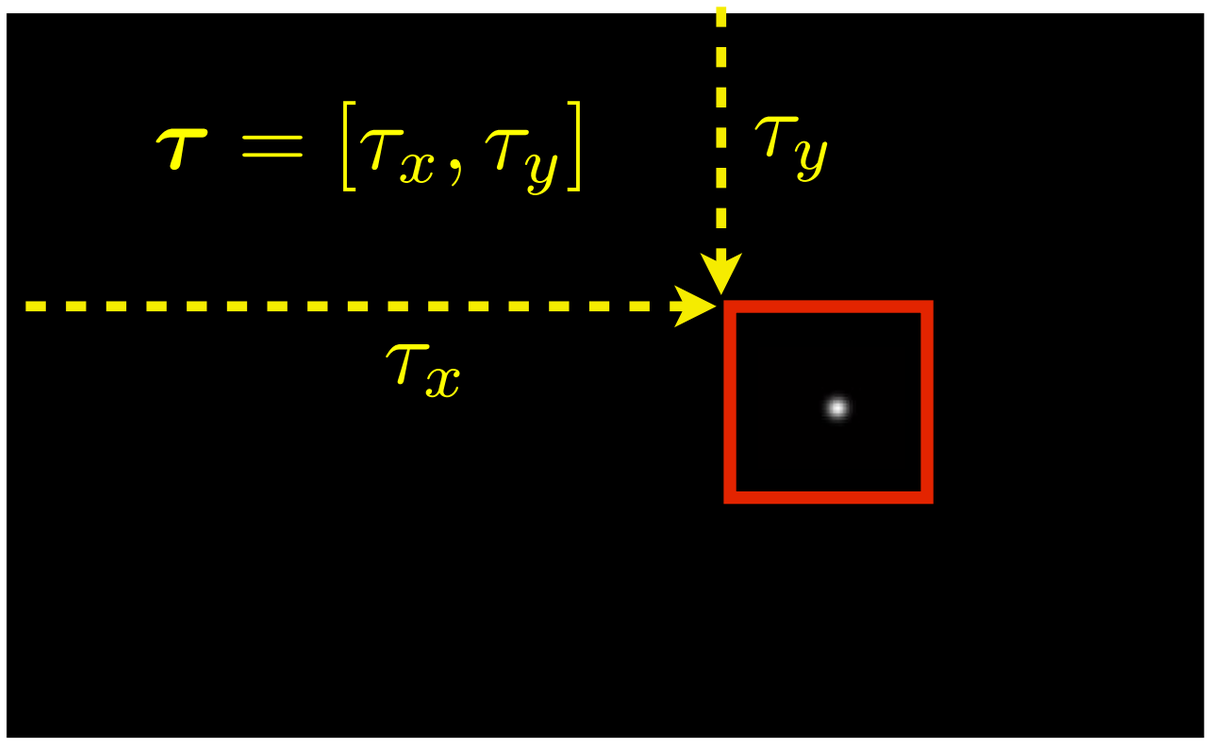
H



X

$*$

H



Y

Reminder

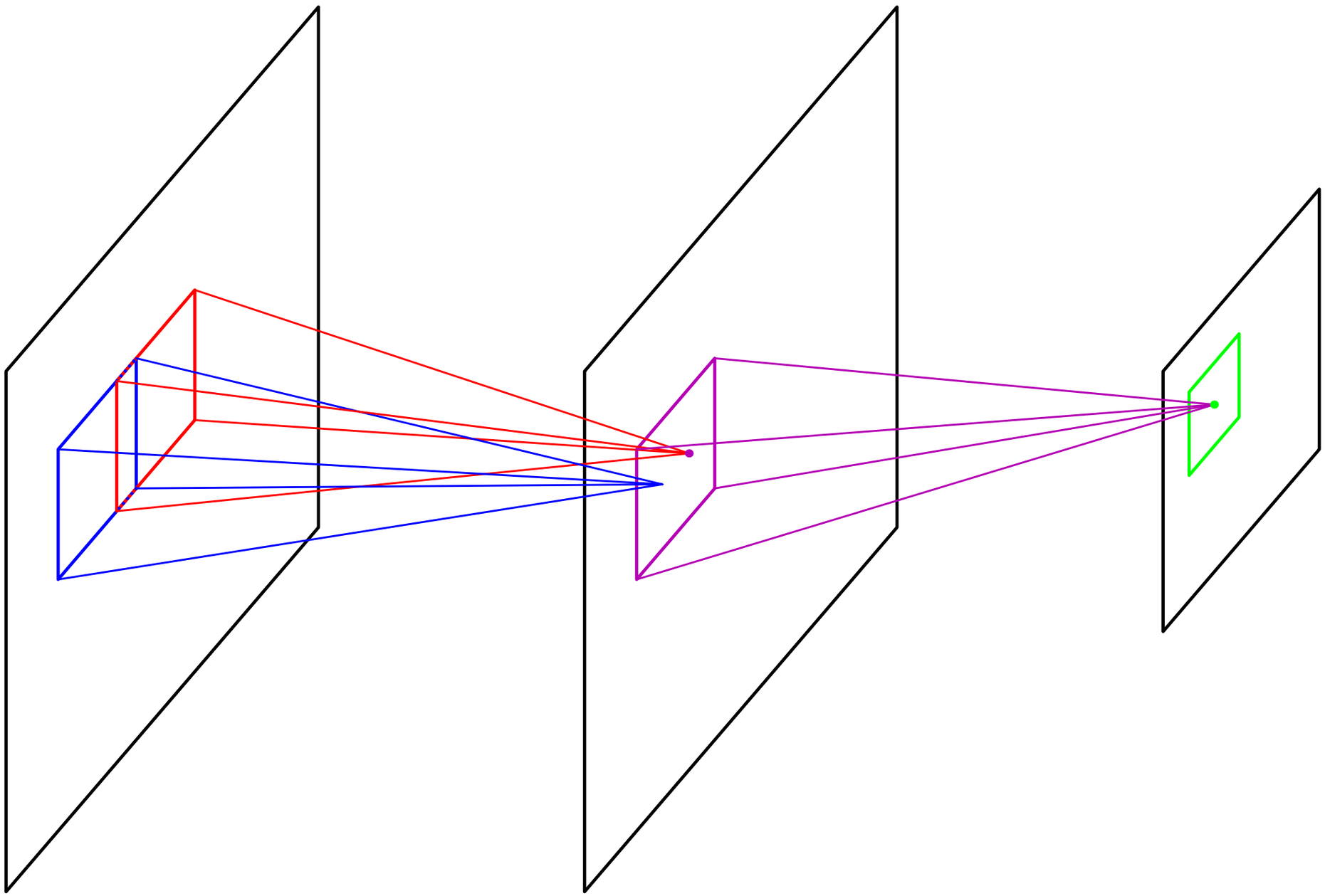
207	245	77	21	247	211	240	1
219	41	58	179	161	154	184	98
215	145	187	71	251	249	65	100
192	2	189	247	166	63	232	213
105	94	66	190	156	61	89	145
159	154	87	184	101	105	72	71
192	111	6	94	60	70	65	226
175	120	210	226	80	183	168	184
134	56	36	240	159	178	76	135
239	244	199	9	132	104	188	185
245	210	78	199	0	92	9	246
5	121	187	122	107	47	12	119
230	171	135	36	82	54	65	37
61	140	79	19	161	96	127	187
56	223	46	6	180	186	142	244
28	20	61	2	178	187	98	220

X

Reminder

$$\phi\{\mathbf{X}\}$$

Convolutional Neural Network



LeCun 1980

Important: My Notation

$a \leftarrow \text{scalar}$ "*lower case italic*"

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_M \end{bmatrix} \leftarrow \text{vector}$$ **"lower case bold"**

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \dots & a_{M,N} \end{bmatrix} \rightarrow \text{matrix}$$ **"Upper Case Bold"**

$\mathcal{A}(x) = \cos(x) \leftarrow \text{function}$ "Upper Case Mathcal"

Today

- Why Convolution?
- Reminder: Single Channel Convolution
- 2D Convolution & Separable Filters
- Multi-Channel Convolution

Reminder: 1D Convolution

$$\mathbf{y}[i] = \sum_{u} \mathbf{x}[u]\mathbf{h}[i-u]$$

Python PyLab Assumption

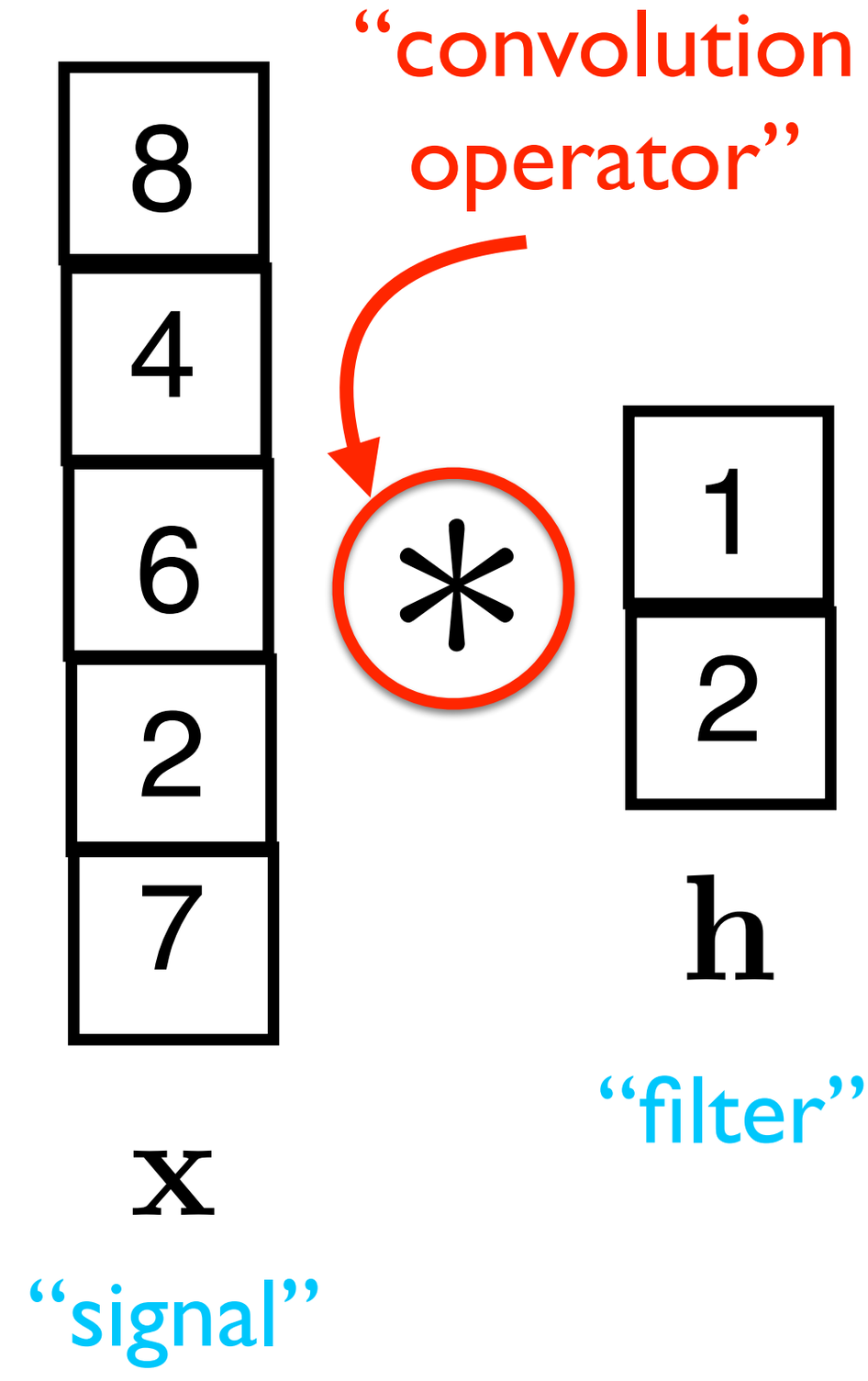
- We will be showing python code snippets in the lecture notes, these will make the assumption that you:-

```
>>> from pylab import *
```

- PyLab is a module that belongs to the Python mathematics library Matplotlib.
- PyLab combines the numerical module numpy with the graphical plotting module pyplot.
- PyLab was designed with the interactive Python interpreter in mind, and therefore many of its functions are short and require minimal typing.
- In scripts, however, you should name everything you import to avoid namespace conflicts.

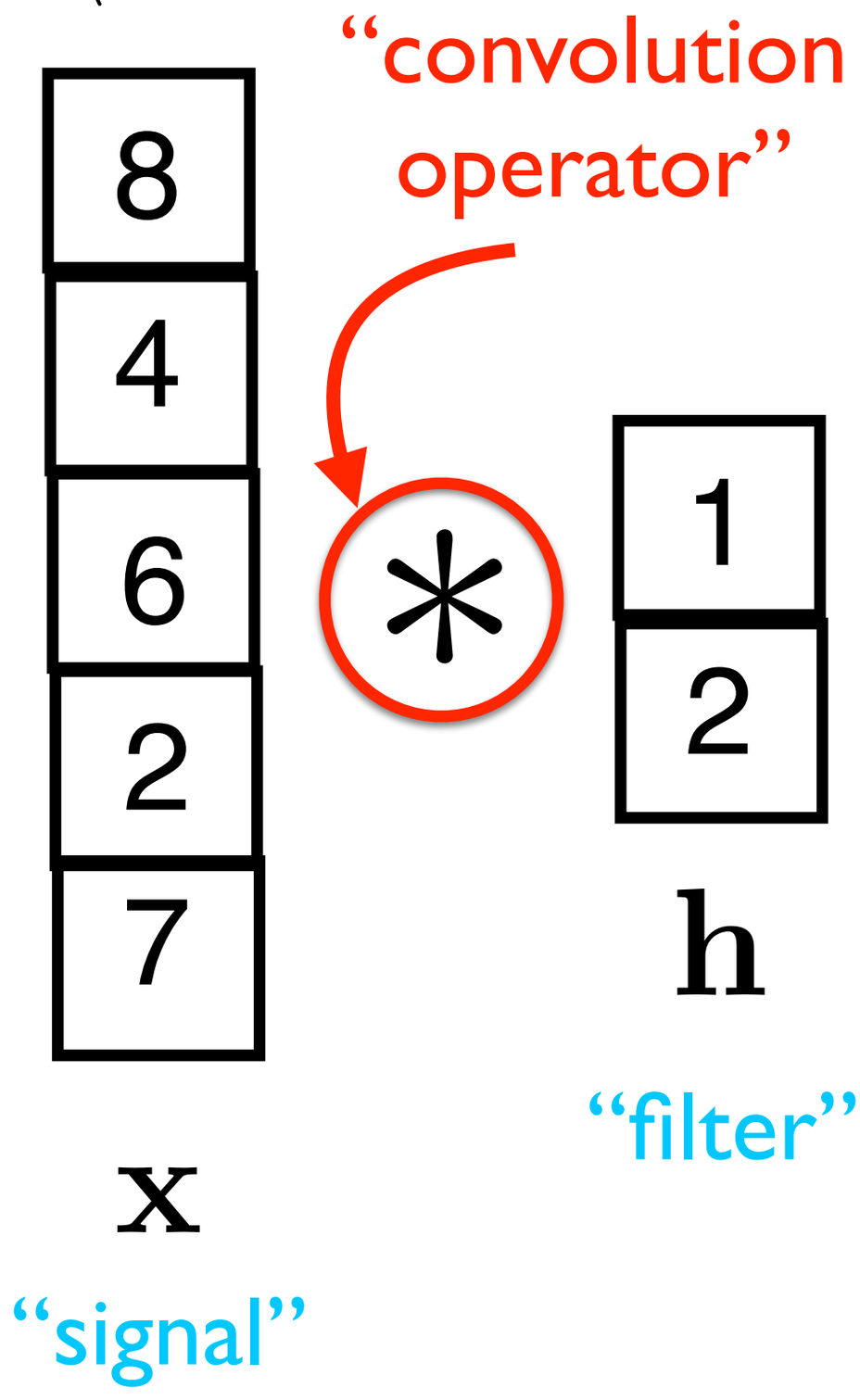
```
>>> import numpy as np
```

Reminder: Convolution

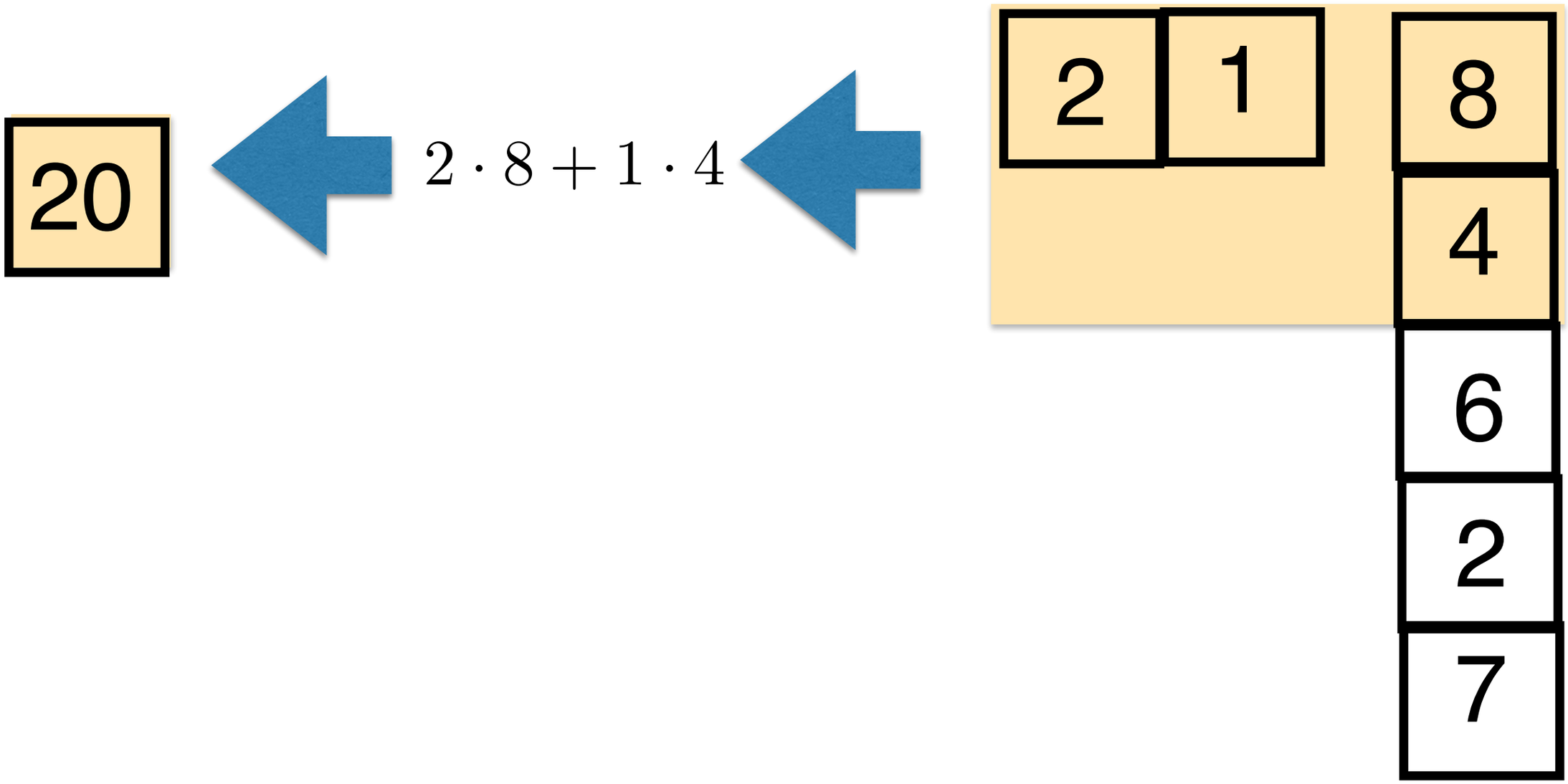


Reminder: Convolution

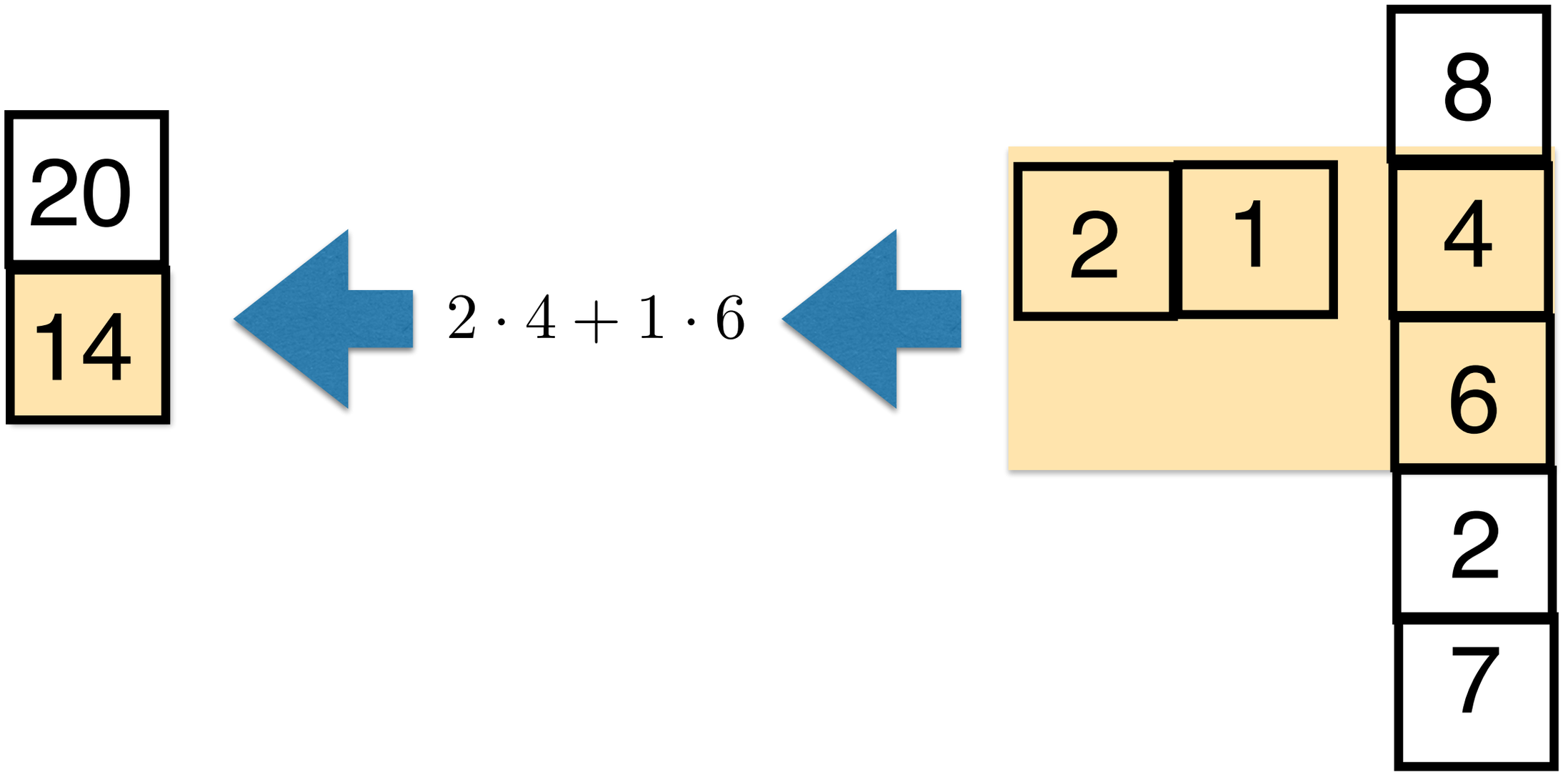
```
>>> from scipy.signal import \
    convolve as conv
>>> conv(x,h,'valid')
array([20, 14, 14, 11])
```



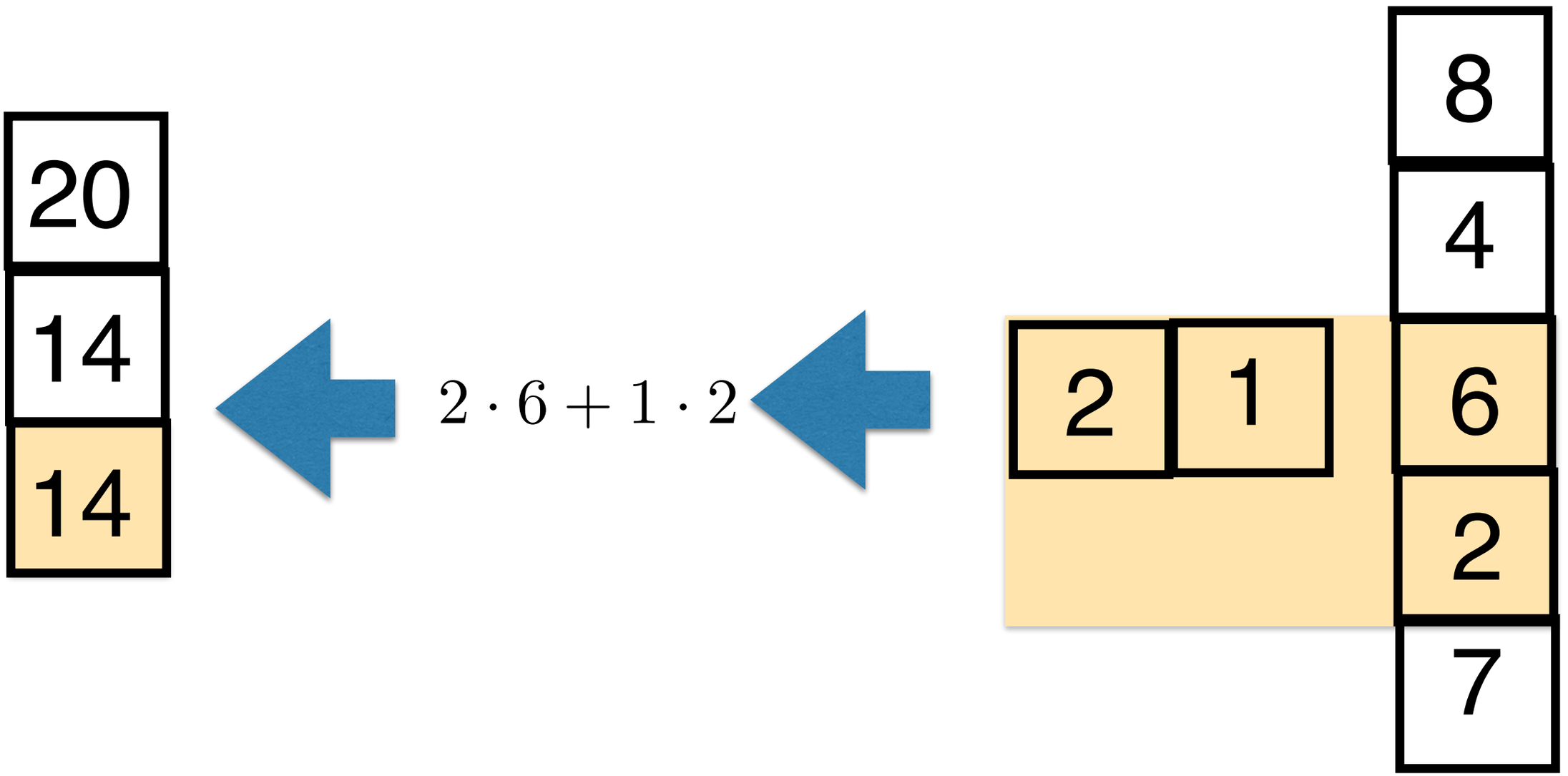
Reminder: 1D Convolution



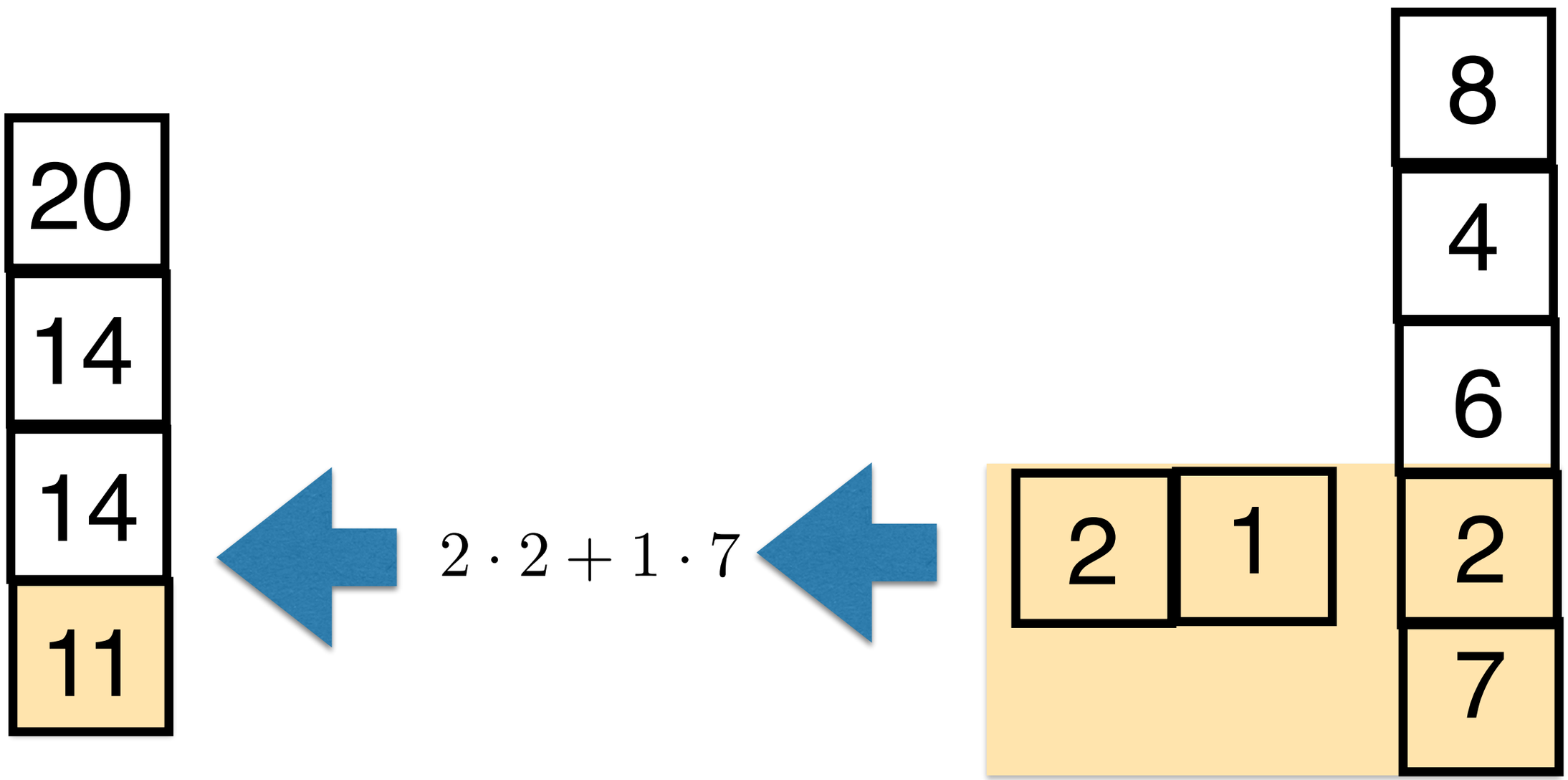
Reminder: 1D Convolution



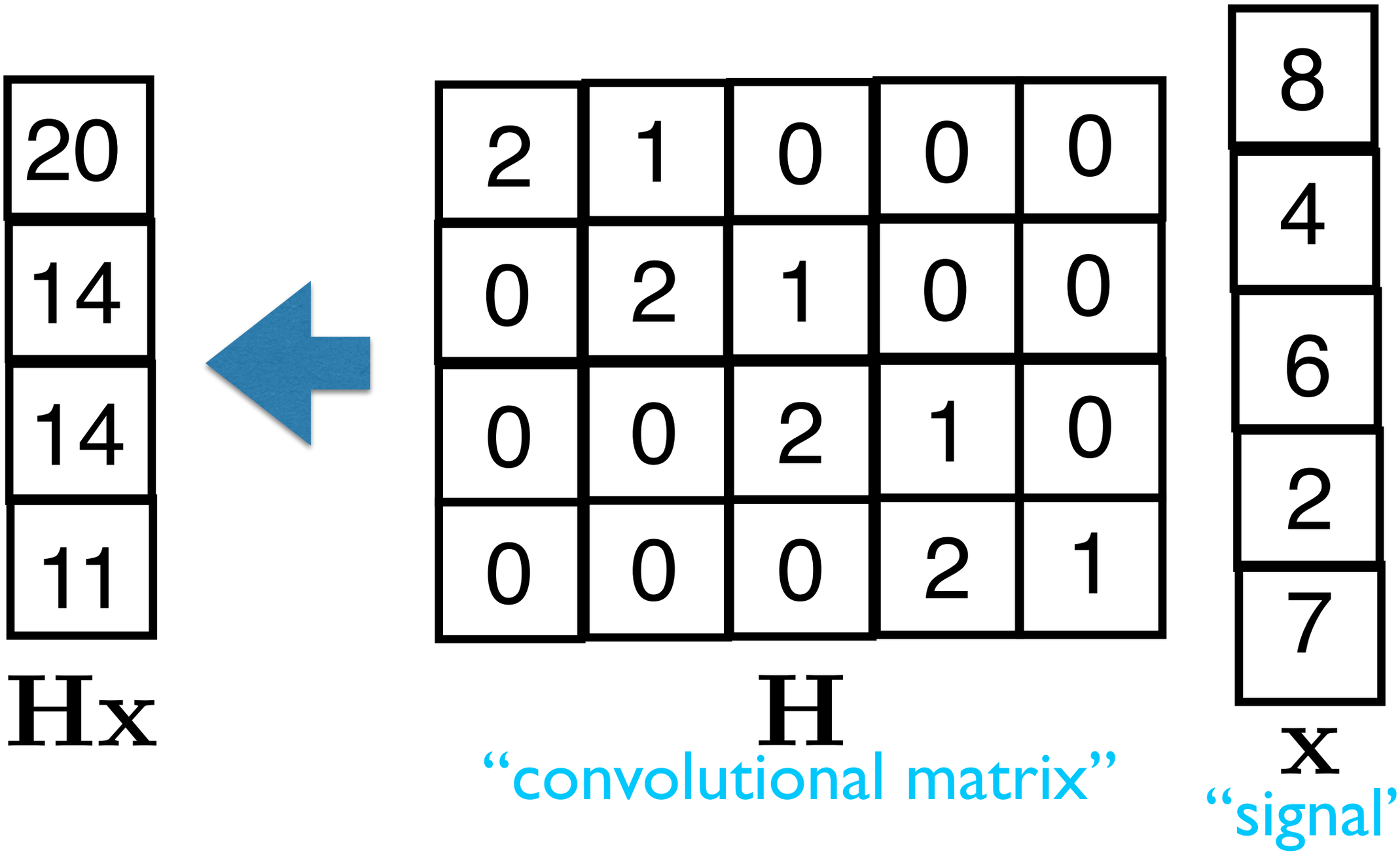
Reminder: 1D Convolution



Reminder: 1D Convolution



Reminder: 1D Convolution



Types of Convolution

- More than just one type of convolutional operator:-
  - “Valid” convolution

```
>>> conv(x,h,'valid')
```

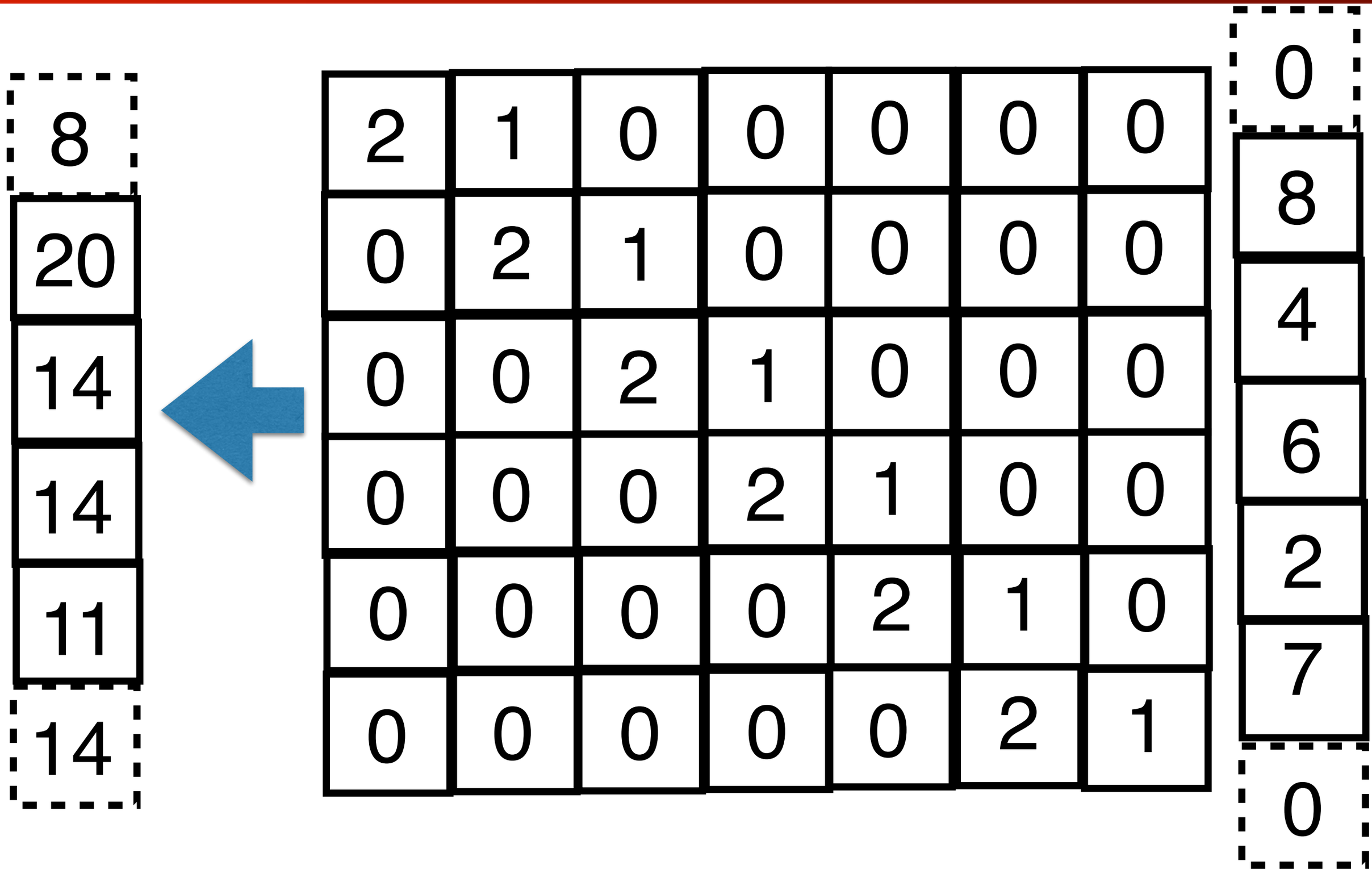
  - “Same” convolution

```
>>> conv(x,h,'same')
```

  - “Full” convolution

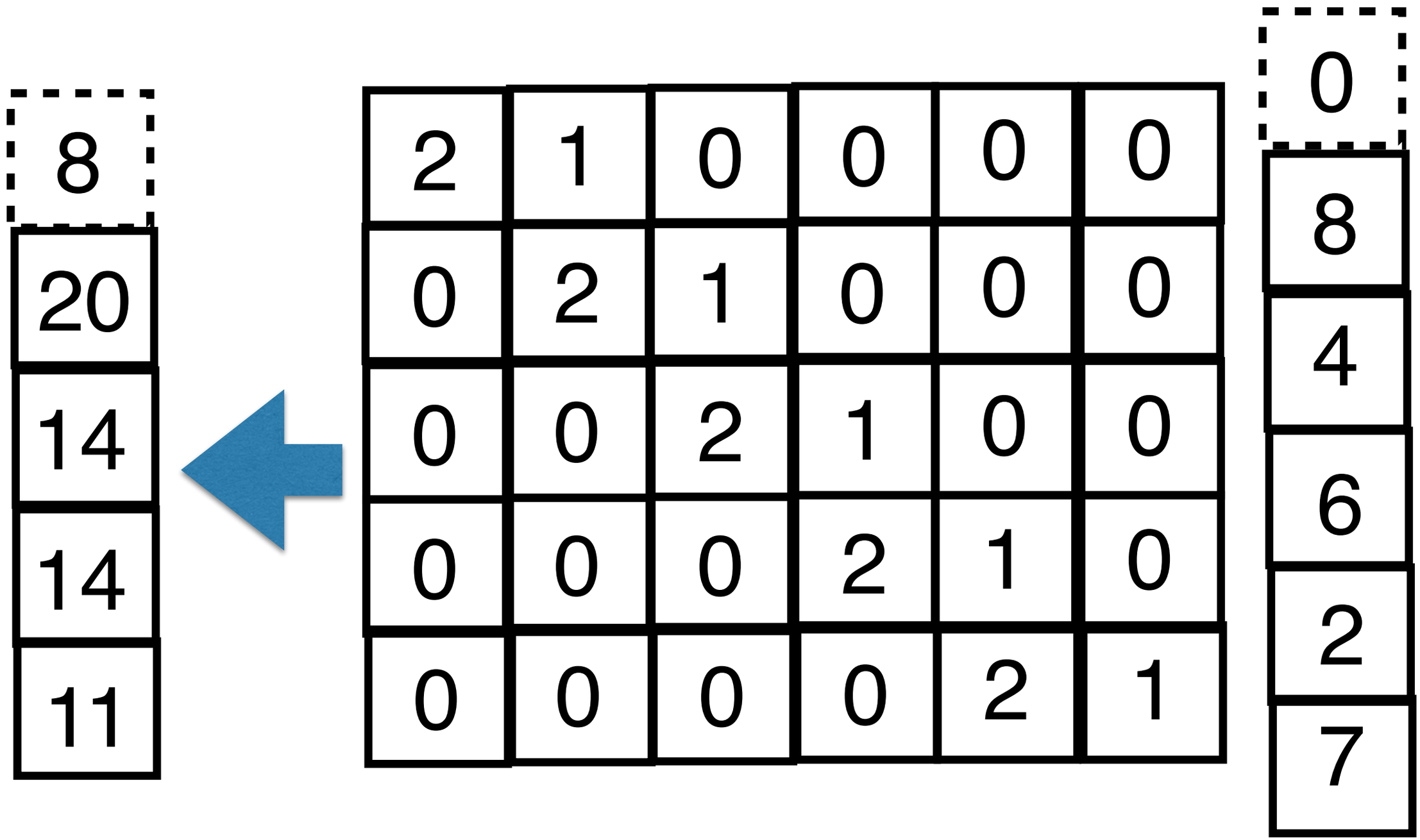
```
>>> conv(x,h,'full')
```

Zero-Padded Convolution



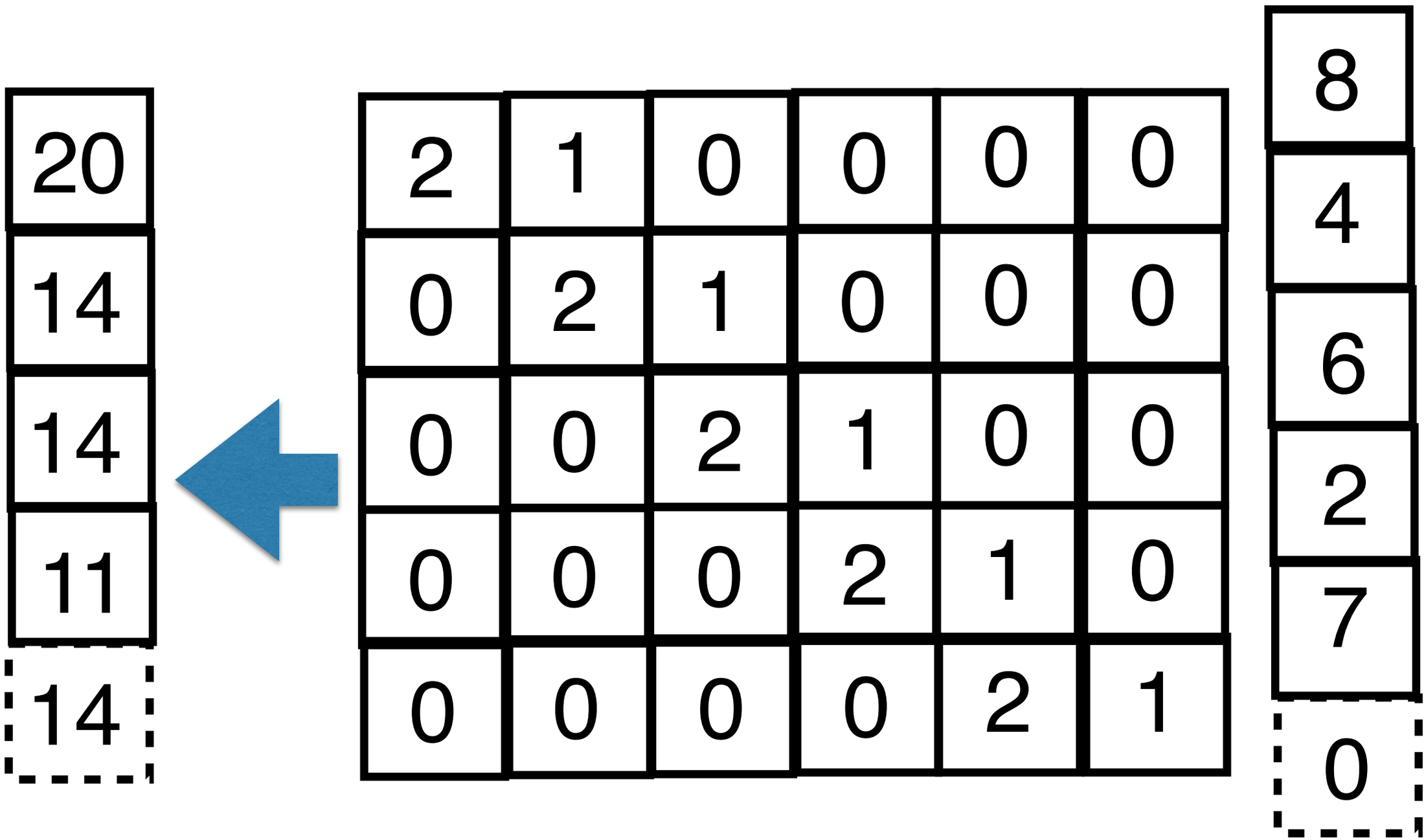

```
>>> conv(x,h,'full')
```

Zero-Padded Convolution



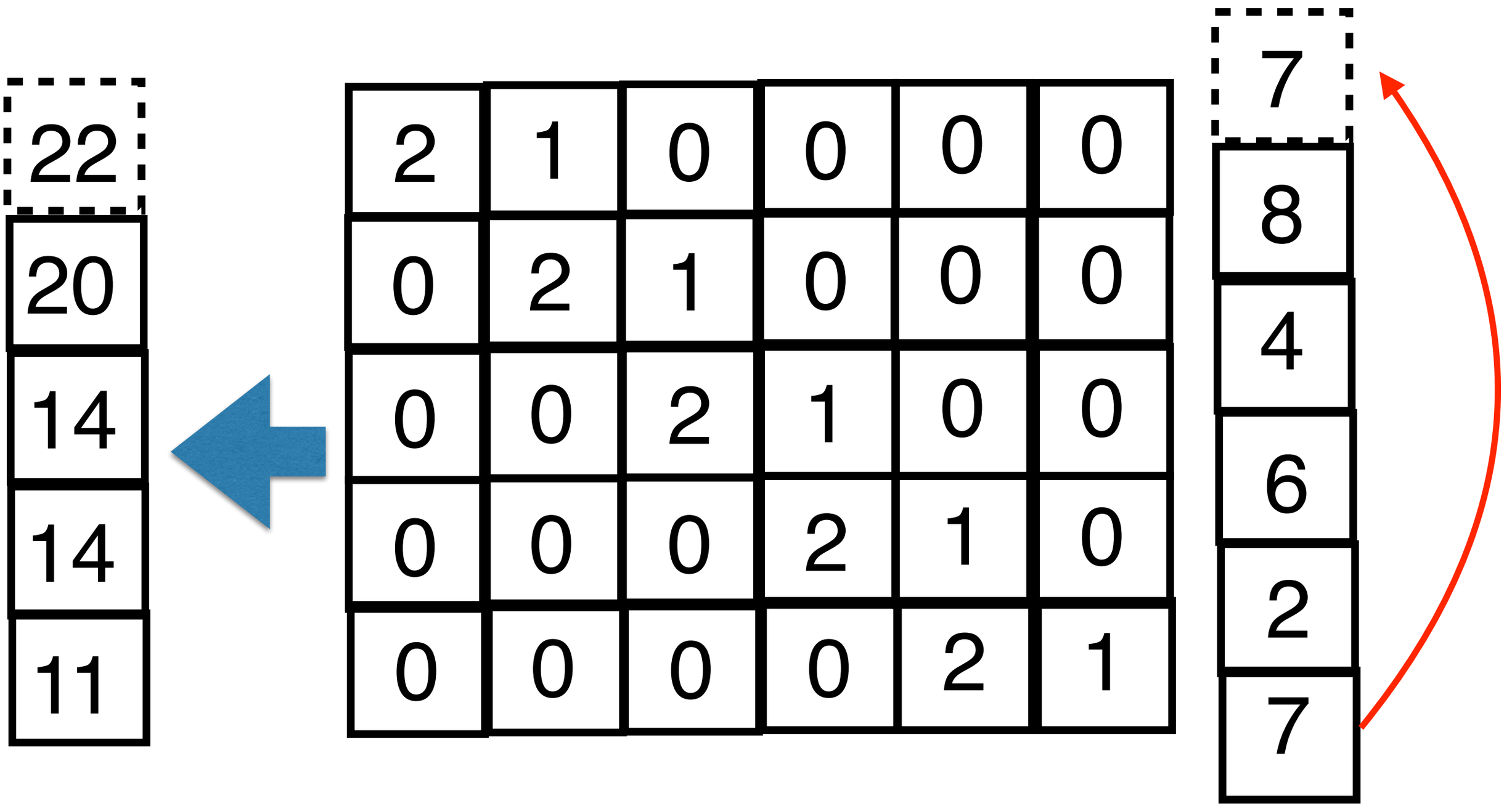

```
>>> conv(x,h,'same')
```

Zero-Padded Convolution

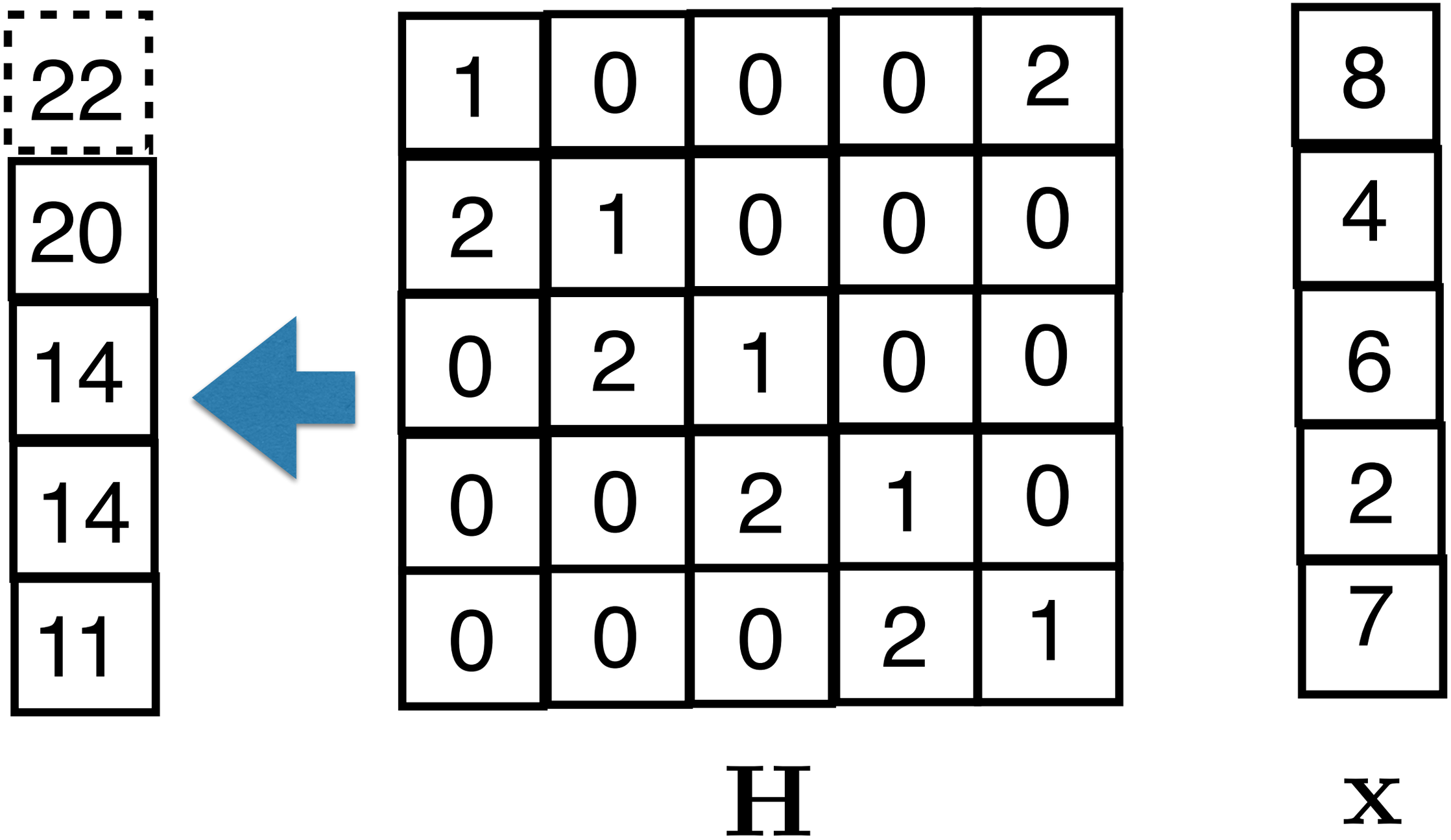



```
>>> imconv(x,h,'same')
```

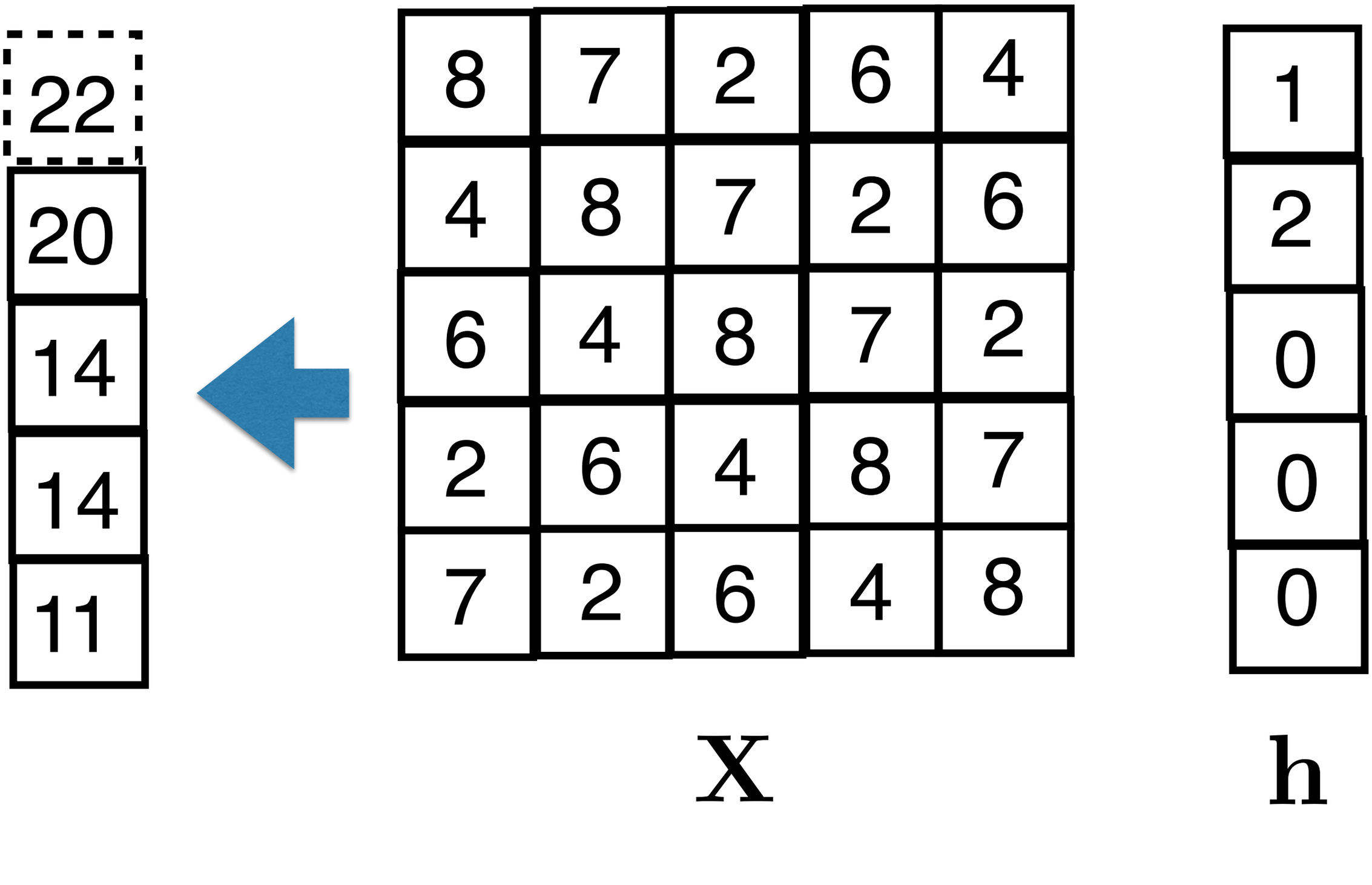
Circular Convolution



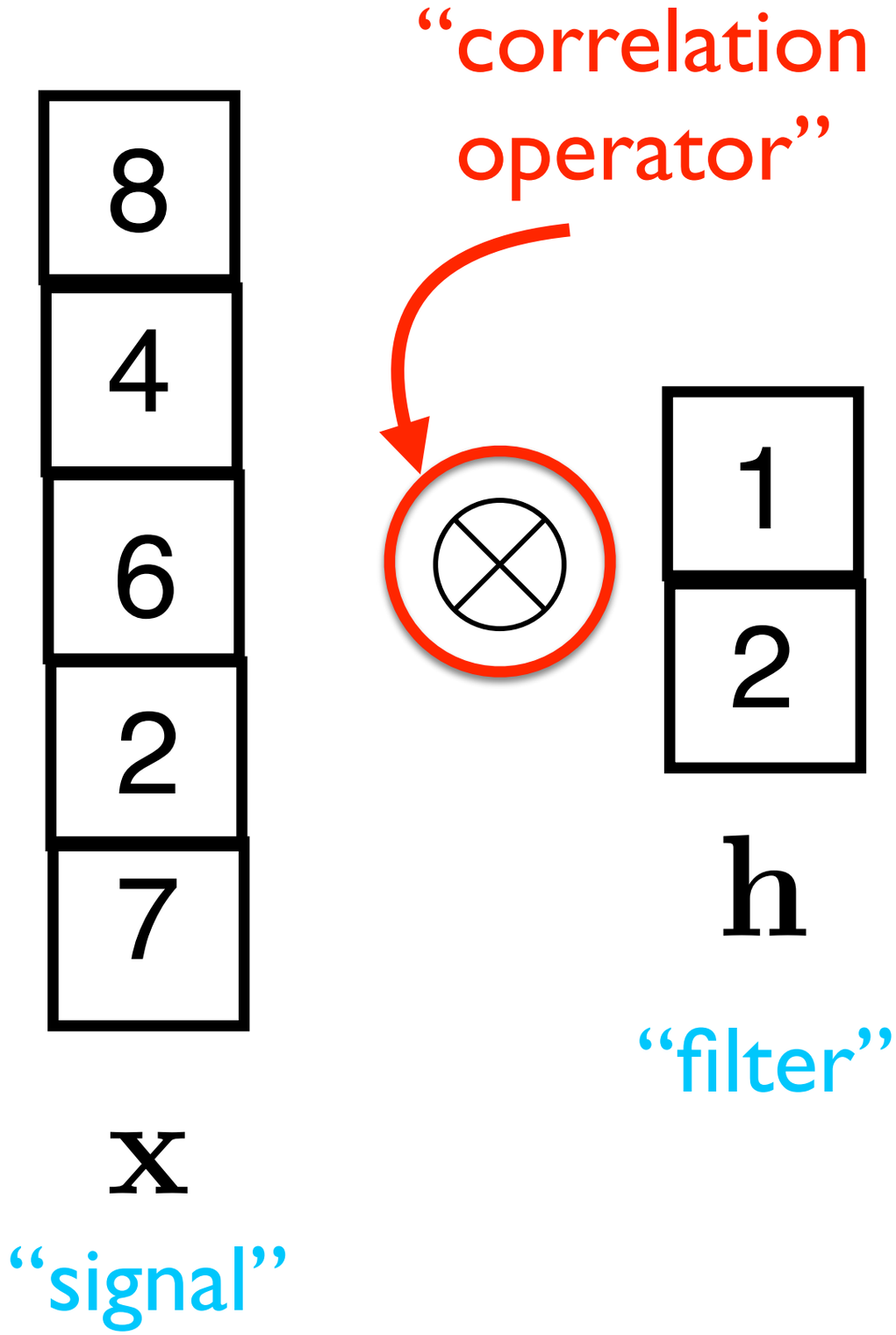
H



Circular Convolution

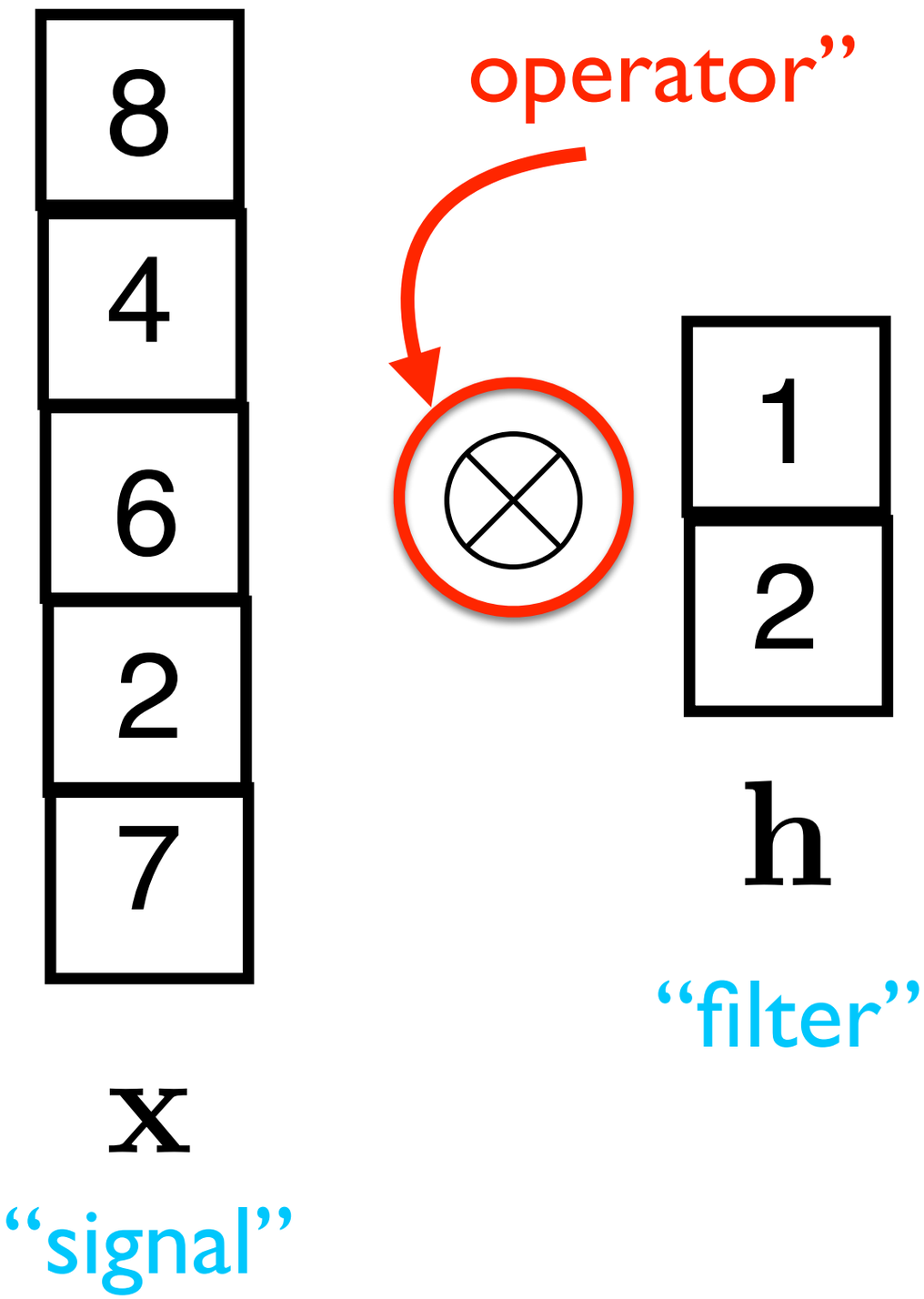


Correlation

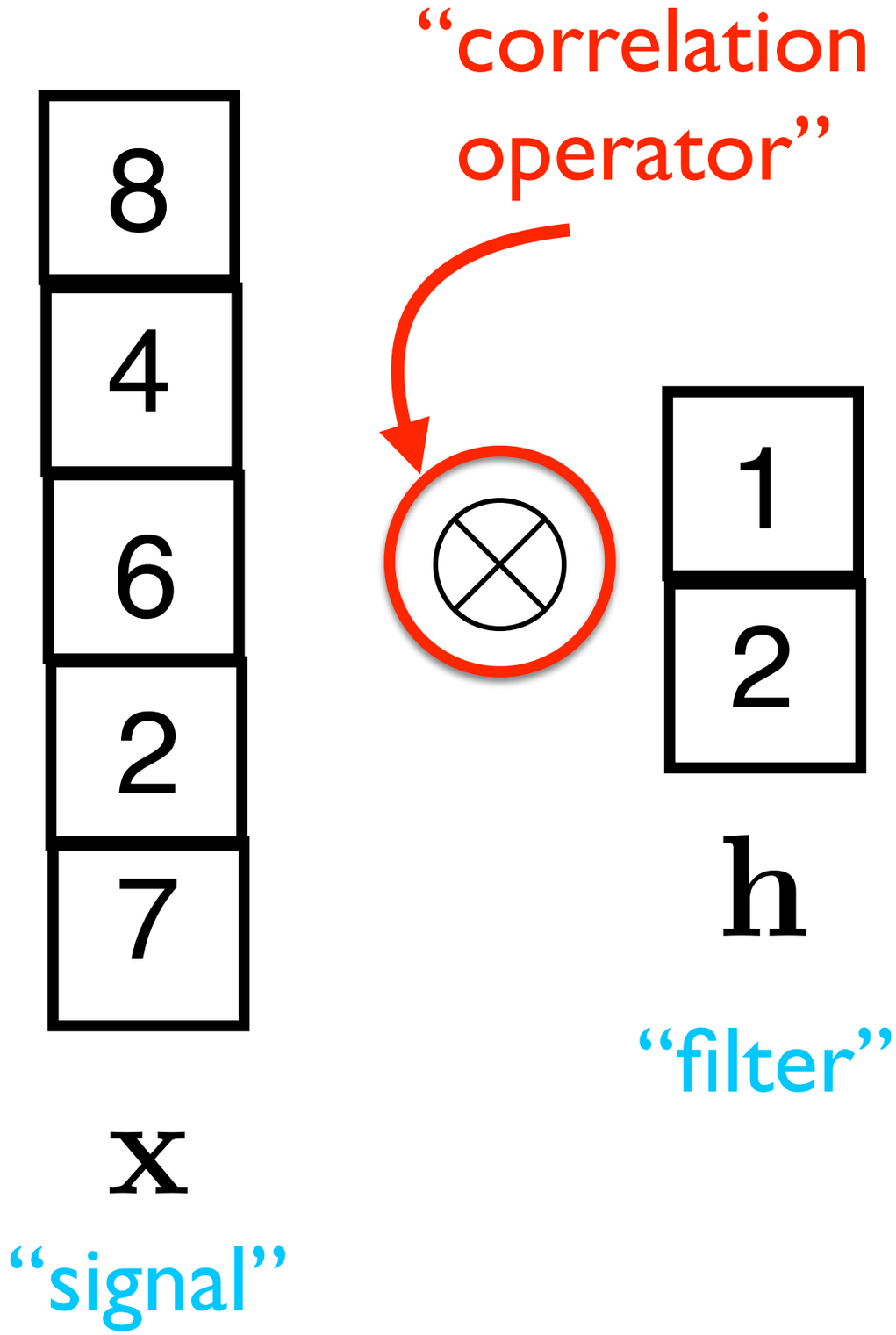


Correlation

```
>>> from scipy.signal import \
    correlate as corr
>>> corr(x,h,'same')
array([16,16,16,10,16])
```

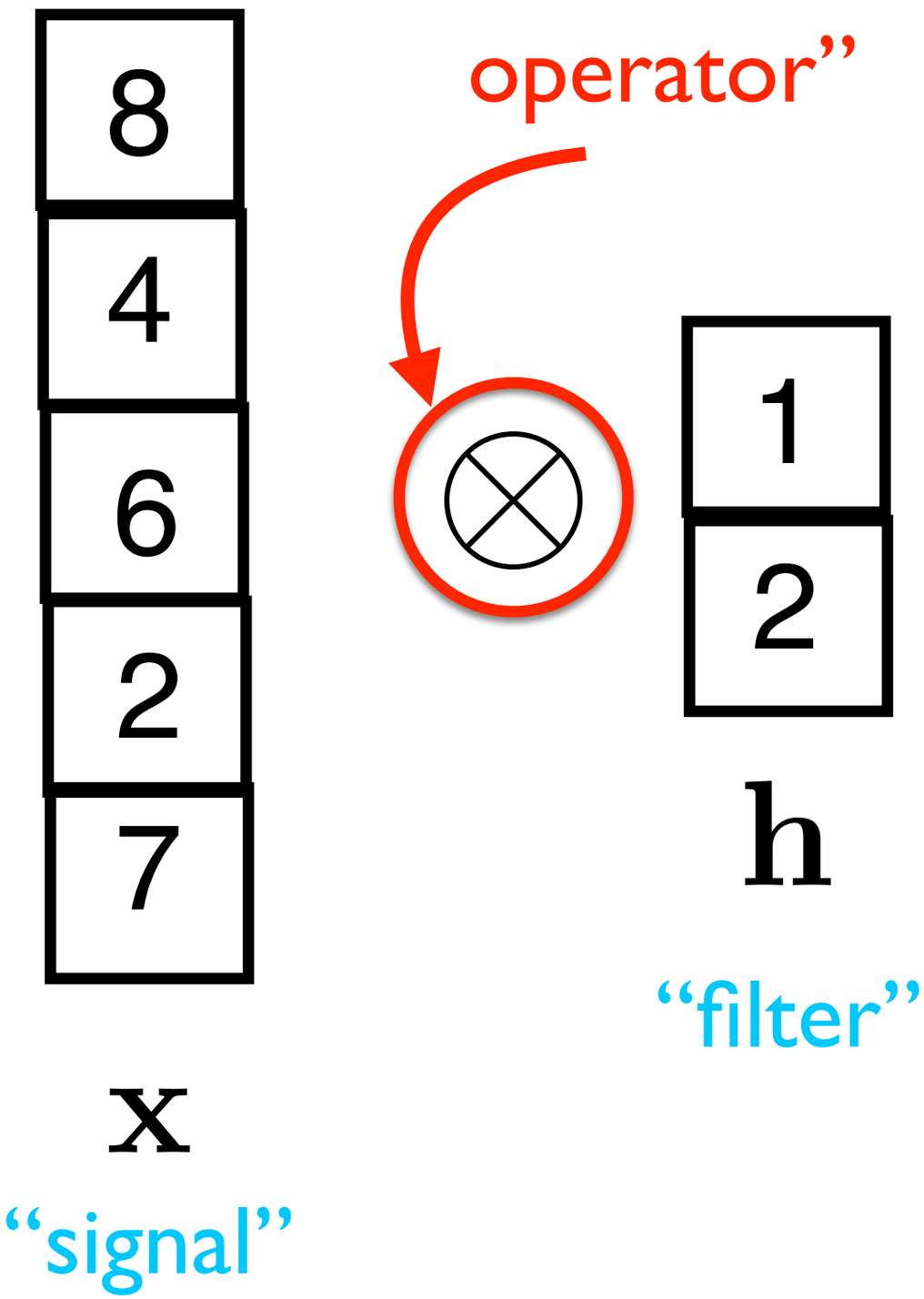


Correlation

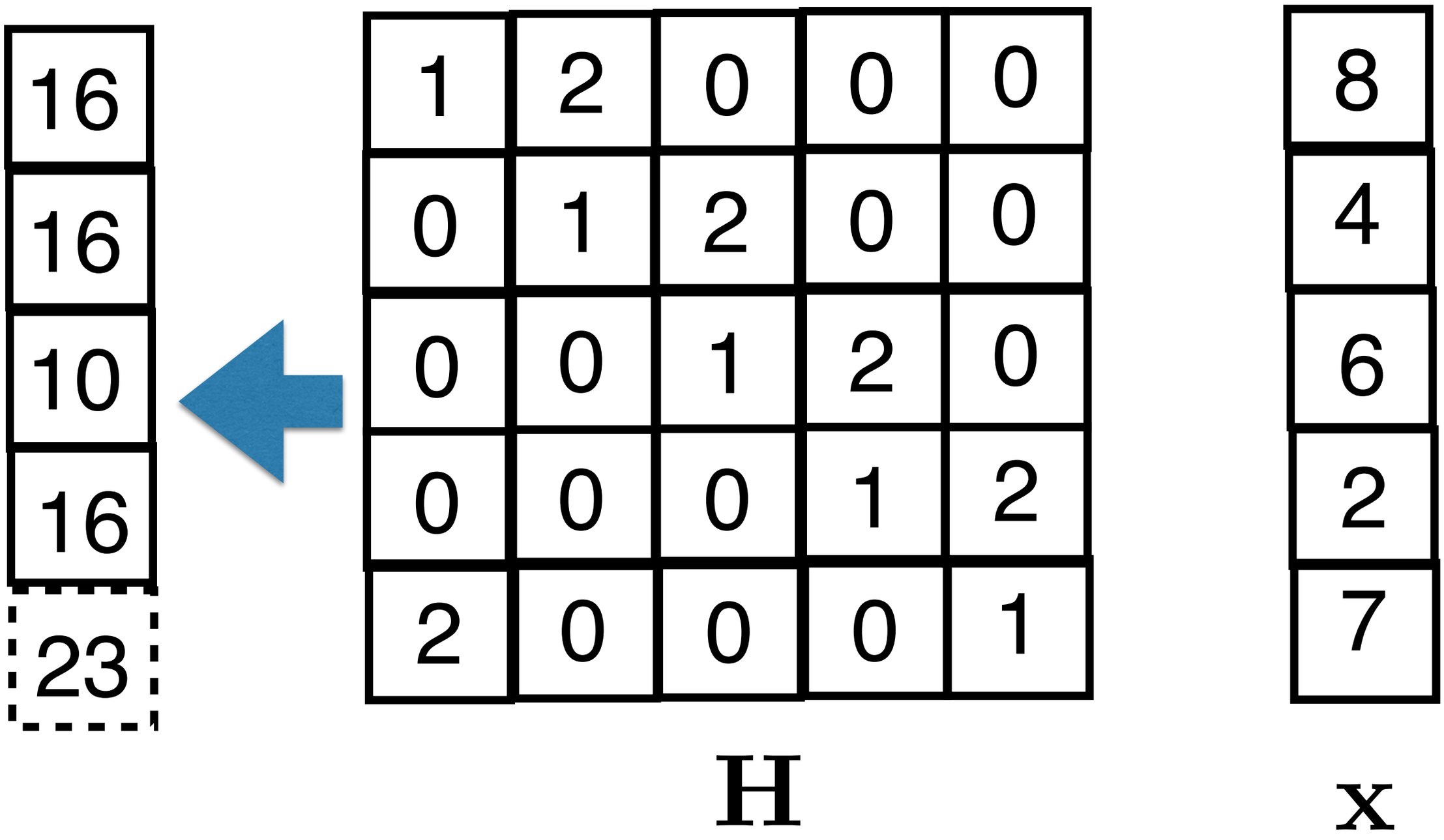


Correlation

```
>>> conv(x,np.flipud(h),'same')
array([16,16,16,10,16])
```



Circular Correlation



Correlation vs. Convolution

- Convolution is preferred mathematically over correlation as it is,

$$\mathbf{g} * (\mathbf{h} * \mathbf{x}) = (\mathbf{g} * \mathbf{h}) * \mathbf{x}$$ **(associative)**

$$\mathbf{h} * \mathbf{x} = \mathbf{x} * \mathbf{h}$$ **(communicative)**

Correlation vs. Convolution

- Convolution is preferred mathematically over correlation as it is,

$$\mathbf{g} * (\mathbf{h} * \mathbf{x}) = (\mathbf{g} * \mathbf{h}) * \mathbf{x} \quad \textbf{(associative)}$$

$$\mathbf{h} * \mathbf{x} = \mathbf{x} * \mathbf{h} \quad \textbf{(communicative)}$$

- Correlation is neither!!!

Correlation vs. Convolution

- Convolution is preferred mathematically over correlation as it is,

$$\mathbf{g} * (\mathbf{h} * \mathbf{x}) = (\mathbf{g} * \mathbf{h}) * \mathbf{x} \quad \textbf{(associative)}$$

$$\mathbf{h} * \mathbf{x} = \mathbf{x} * \mathbf{h} \quad \textbf{(communicative)}$$

- Correlation is neither!!!

$$\mathbf{g} \otimes (\mathbf{h} \otimes \mathbf{x}) \neq (\mathbf{g} \otimes \mathbf{h}) \otimes \mathbf{x}$$

$$\mathbf{h} \otimes \mathbf{x} \neq \mathbf{x} \otimes \mathbf{h}$$

Correlation vs. Convolution

- Convolution is preferred mathematically over correlation as it is,

$$\mathbf{g} * (\mathbf{h} * \mathbf{x}) = (\mathbf{g} * \mathbf{h}) * \mathbf{x}$$ **(associative)**

$$\mathbf{h} * \mathbf{x} = \mathbf{x} * \mathbf{h}$$ **(communicative)**

- Correlation is neither!!!

$$\mathbf{g} \otimes (\mathbf{h} \otimes \mathbf{x}) \neq (\mathbf{g} \otimes \mathbf{h}) \otimes \mathbf{x}$$

$$\mathbf{h} \otimes \mathbf{x} \neq \mathbf{x} \otimes \mathbf{h}$$

- Correlation, however, preferred for signal matching/ detection.

Today

- Why Convolution?
- Reminder: Single Channel Convolution
- 2D Convolution & Separable Filters
- Multi-Channel Convolution

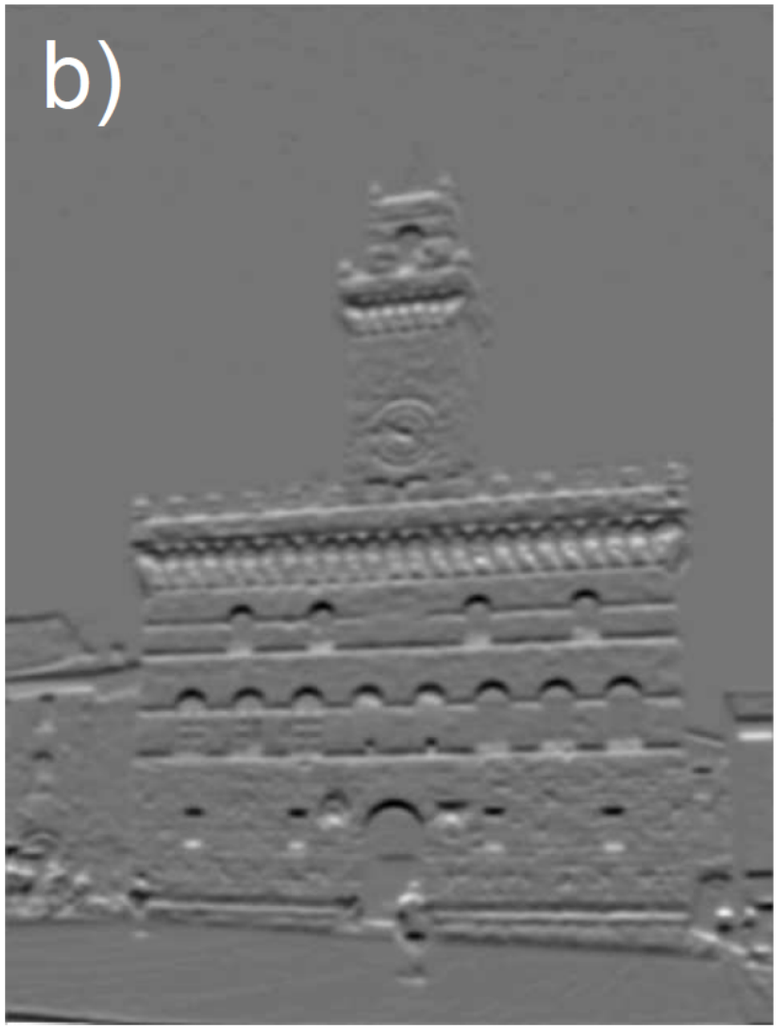
Reminder: 2D Convolution

$$\mathbf{Y}[j,i] = \sum_{v} \sum_{u} \mathbf{X}[u,v]\mathbf{H}[j-v, i-u]$$

Reminder: 2D Convolution

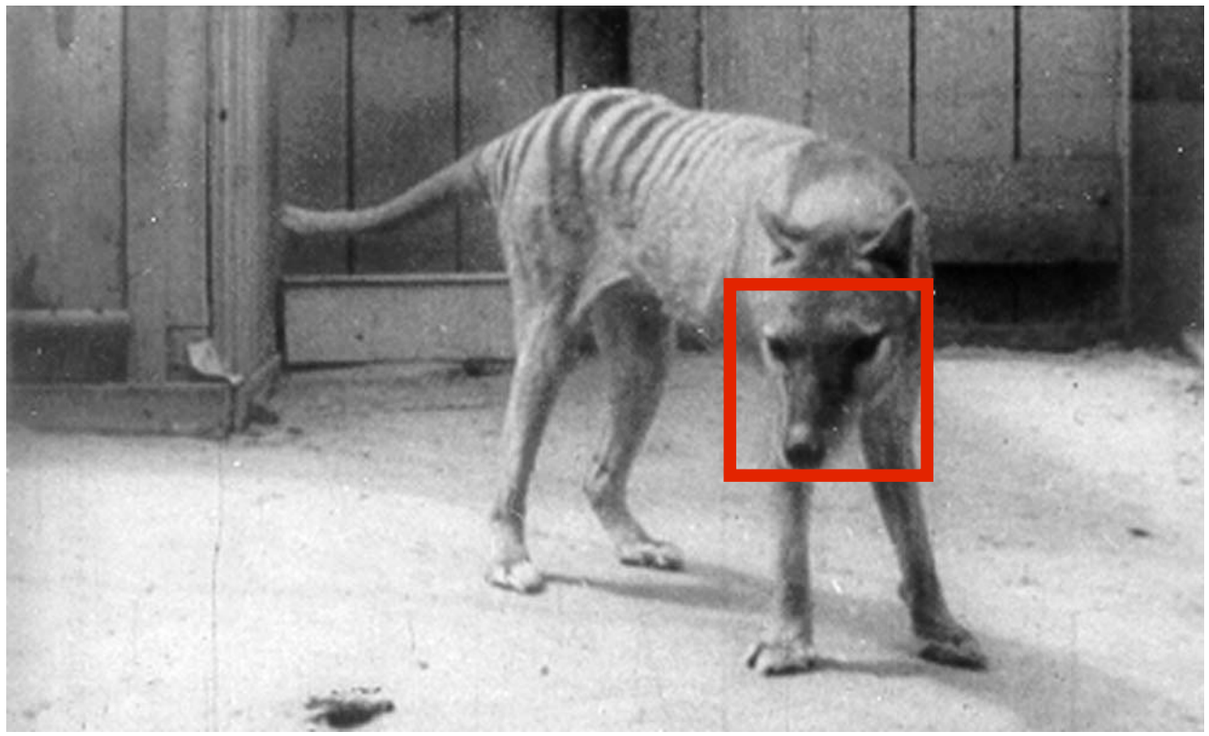
$$\mathbf{Y} = \mathbf{X} * \mathbf{H}$$

Estimating Image Gradients

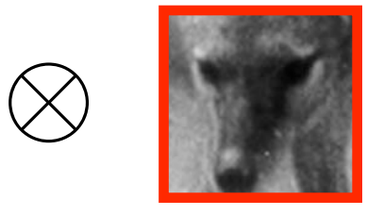


Compute horizontal and vertical gradient images.

2D Convolution vs. Correlation



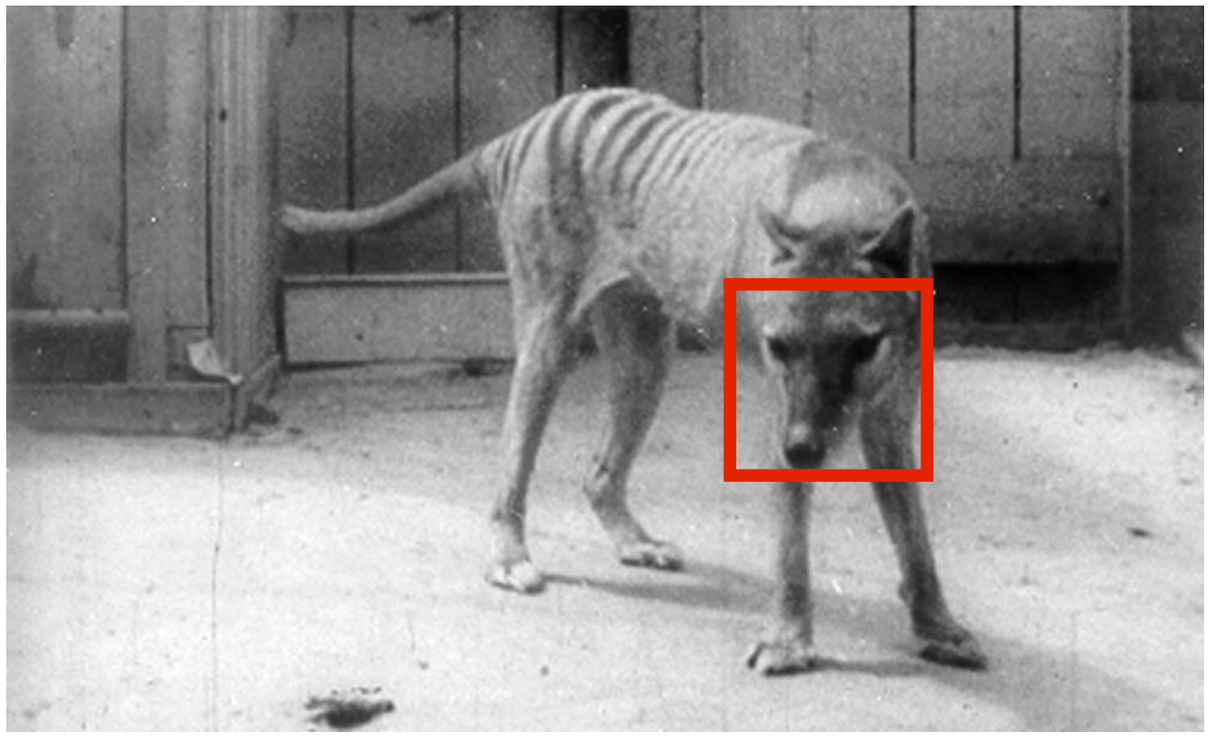
X



G

```
>>> from scipy.ndimage import \
correlation as imcorr
>>> Y = imcorr(X,G)
```

2D Convolution vs. Correlation



$*$

H

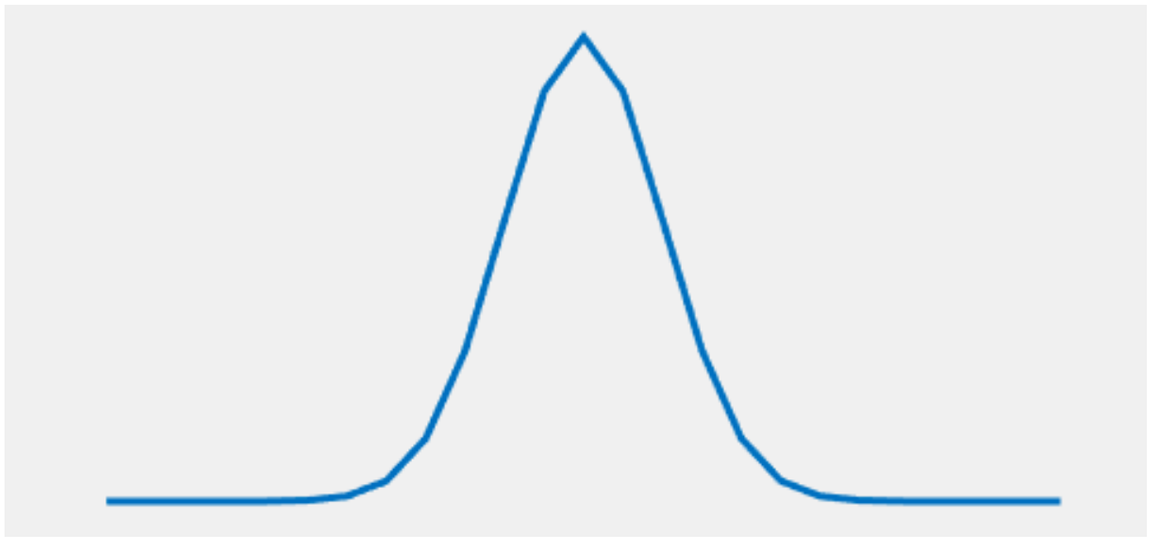
X

```
>>> H = flipud(fliplr(G))
>>> Y = imconv(X,H)
```

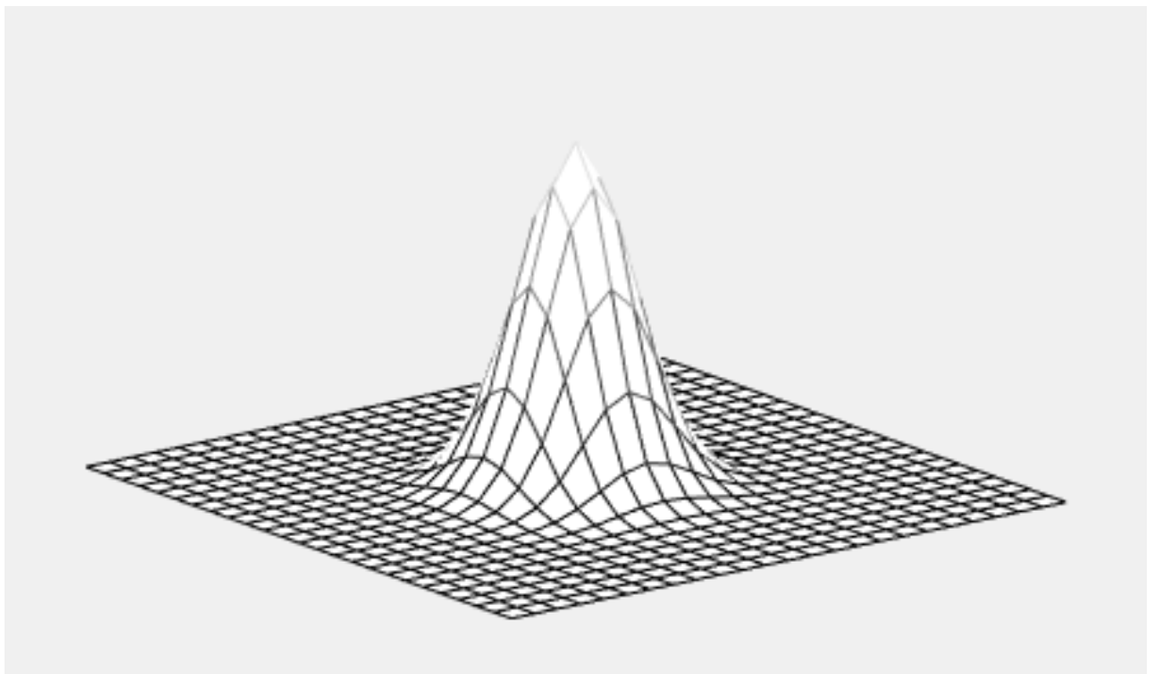
Gaussian Filter is Separable

In Python,

```
>>> from scipy.signal.windows import gaussian
>>> h = asmatrix(gaussian(25,3))
```



```
>>> H = dot(h,h.T)
```



Separable Filters

- In some circumstances the 2D filter will be of low rank such that,

$$\mathrm{rank}(\mathbf{H}) = 1$$

Separable Filters

- In some circumstances the 2D filter will be of low rank such that,

$$\mathbf{H} = (\mathbf{h}\mathbf{g}^T)$$

Separable Filters

- In some circumstances the 2D filter will be of low rank such that,

$$\mathbf{H} = (\mathbf{h}\mathbf{g}^T)$$

- Using the properties of associativity we can therefore show,

$$\mathbf{Y} = \mathbf{X} * \mathbf{H}$$

Separable Filters

- In some circumstances the 2D filter will be of low rank such that,

$$\mathbf{H} = (\mathbf{h}\mathbf{g}^T)$$

- Using the properties of associativity we can therefore show,

$$\mathbf{Y} = \mathbf{X} * (\mathbf{h}\mathbf{g}^T)$$

Separable Filters

- In some circumstances the 2D filter will be of low rank such that,

$$\mathbf{H} = (\mathbf{h}\mathbf{g}^T)$$

- Using the properties of associativity we can therefore show,

$$\mathbf{Y} = (\mathbf{X} * \mathbf{h}) * \mathbf{g}^T$$

Separable Filters

- In some circumstances the 2D filter will be of low rank such that,

$$\mathbf{H} = (\mathbf{h}\mathbf{g}^T)$$

- Using the properties of associativity we can therefore show,

$$\mathbf{Y} = (\mathbf{X} * \mathbf{h}) * \mathbf{g}^T$$

Can you show why?

Vectorized Notation

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \dots & a_{M,N} \end{bmatrix} \rightarrow \text{matrix}$$

$$\mathbf{a} = \text{vec}(\mathbf{A}) = \begin{bmatrix} a_{1,1} \\ \vdots \\ a_{M,N} \end{bmatrix}$$

Vectorized Convolution Notation

$$\mathbf{Y} = \mathbf{X} * \mathbf{H}$$

Vectorized Convolution Notation

$$\mathrm{vec}(\mathbf{Y}) = \mathrm{vec}(\mathbf{X} * \mathbf{H})$$

Vectorized Convolution Notation

$$\mathbf{y} = \mathbf{x} * \mathbf{h}$$

Vectorized Convolution Notation

$$\mathbf{y} = \mathbf{x} * \mathbf{h}$$

"Does not ALWAYS imply 1D convolution"

Vectorized Convolution Notation

$$\mathbf{y} = \mathbf{H}\mathbf{x}$$

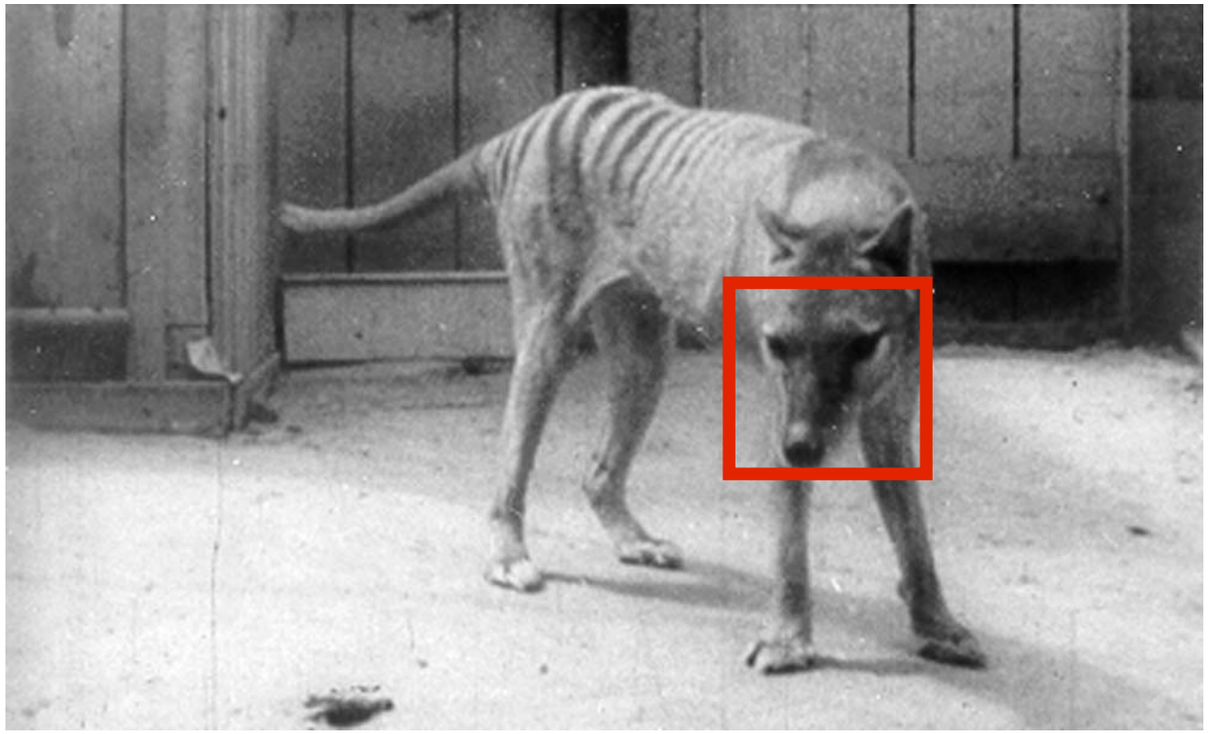
"2D convolutional matrix"

Vectorized Convolution Notation

$$\mathbf{y} = \mathbf{X}\mathbf{h}$$

Today

- Why Convolution?
- Reminder: Single Channel Convolution
- 2D Convolution & Separable Filters
- Multi-Channel Convolution

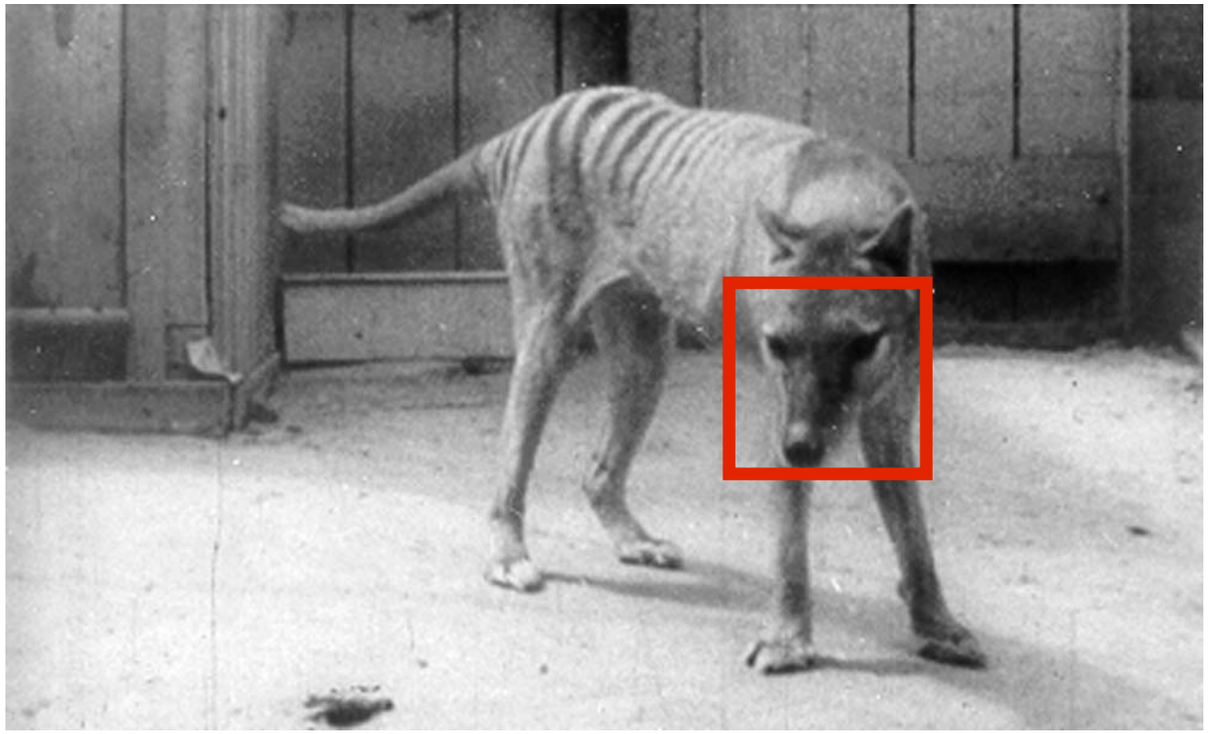


X

*

H

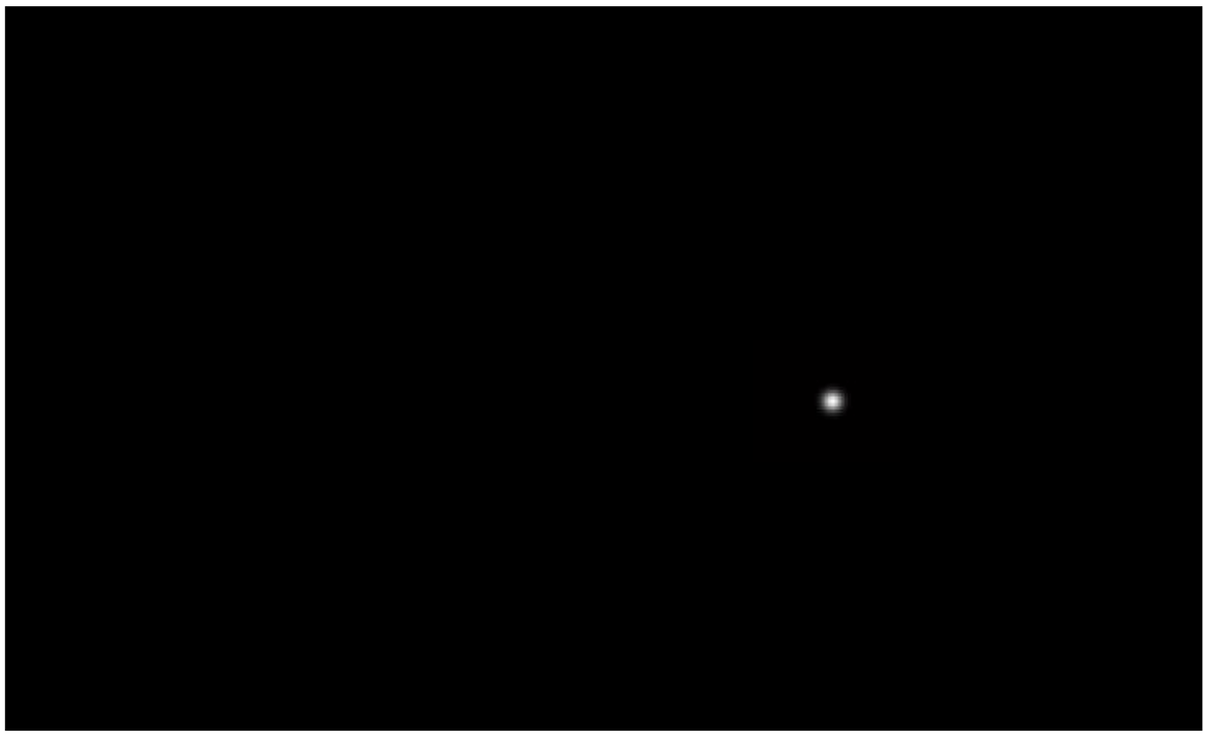
Y



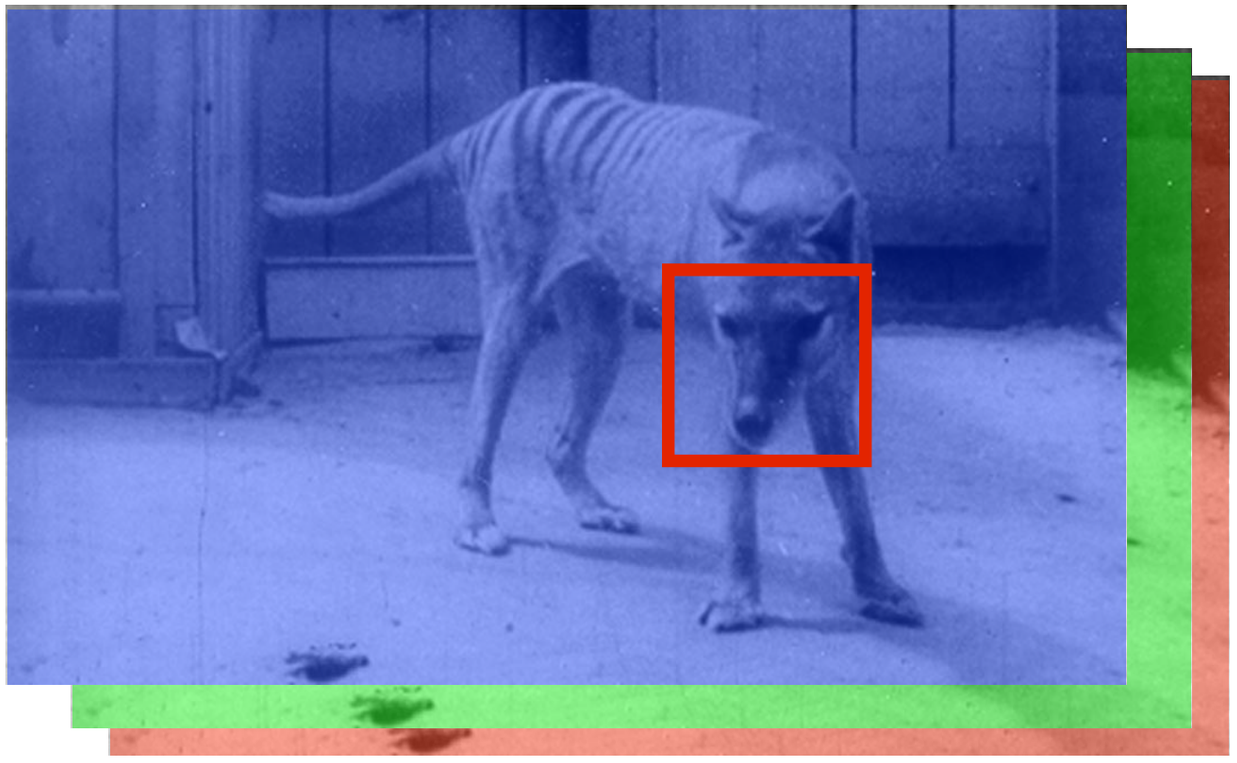
x

*

h

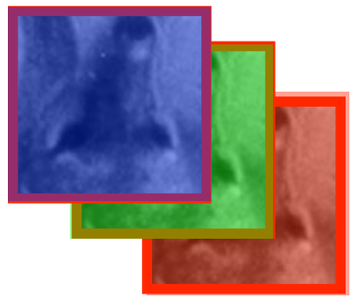


y



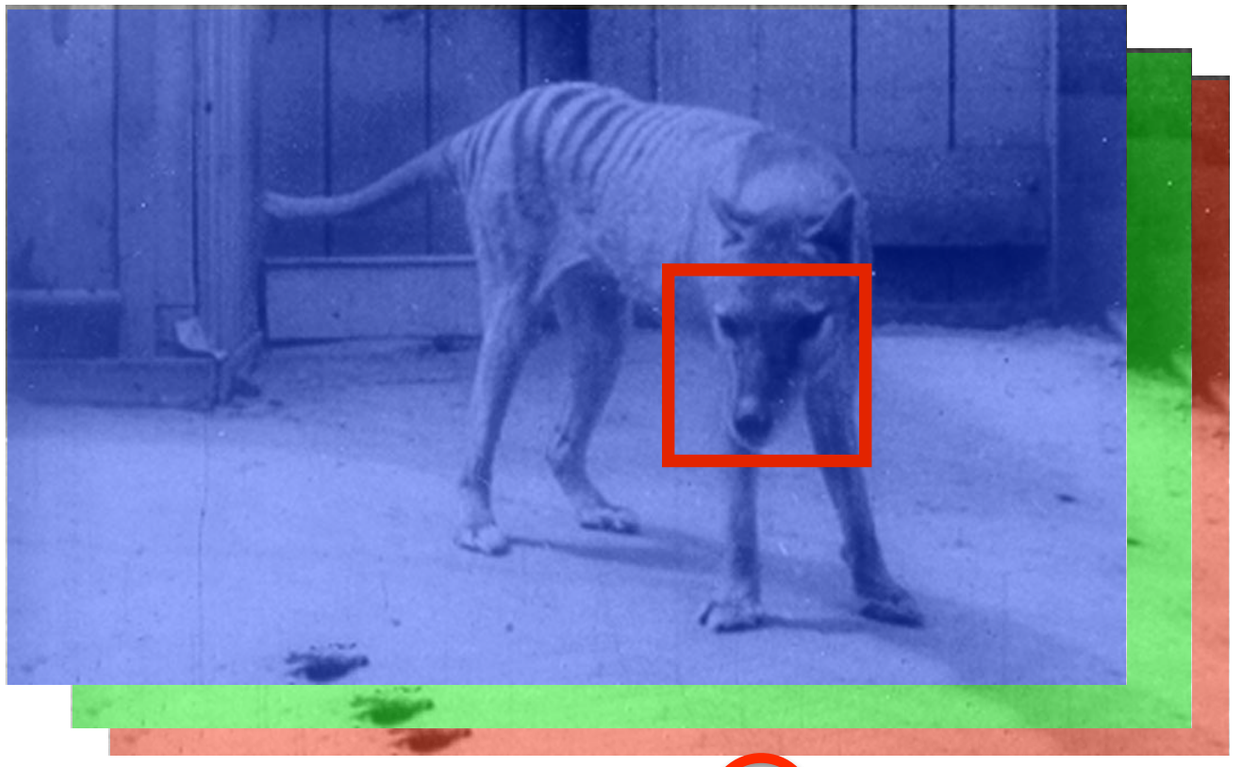
$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$

$*$



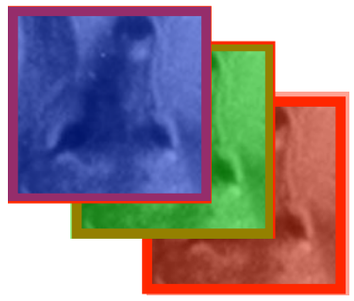
$\{\mathbf{h}^{(k)}\}_{k=1}^{K}$

$\mathbf{y}$



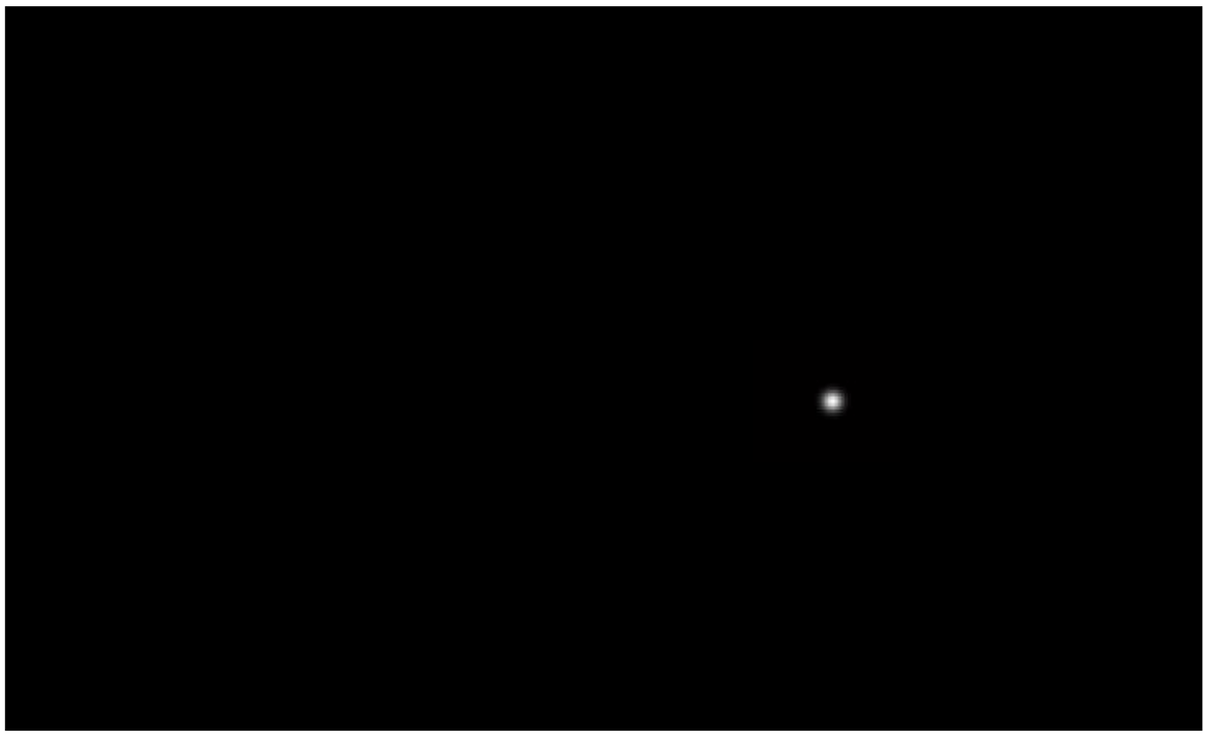
$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$

$*$

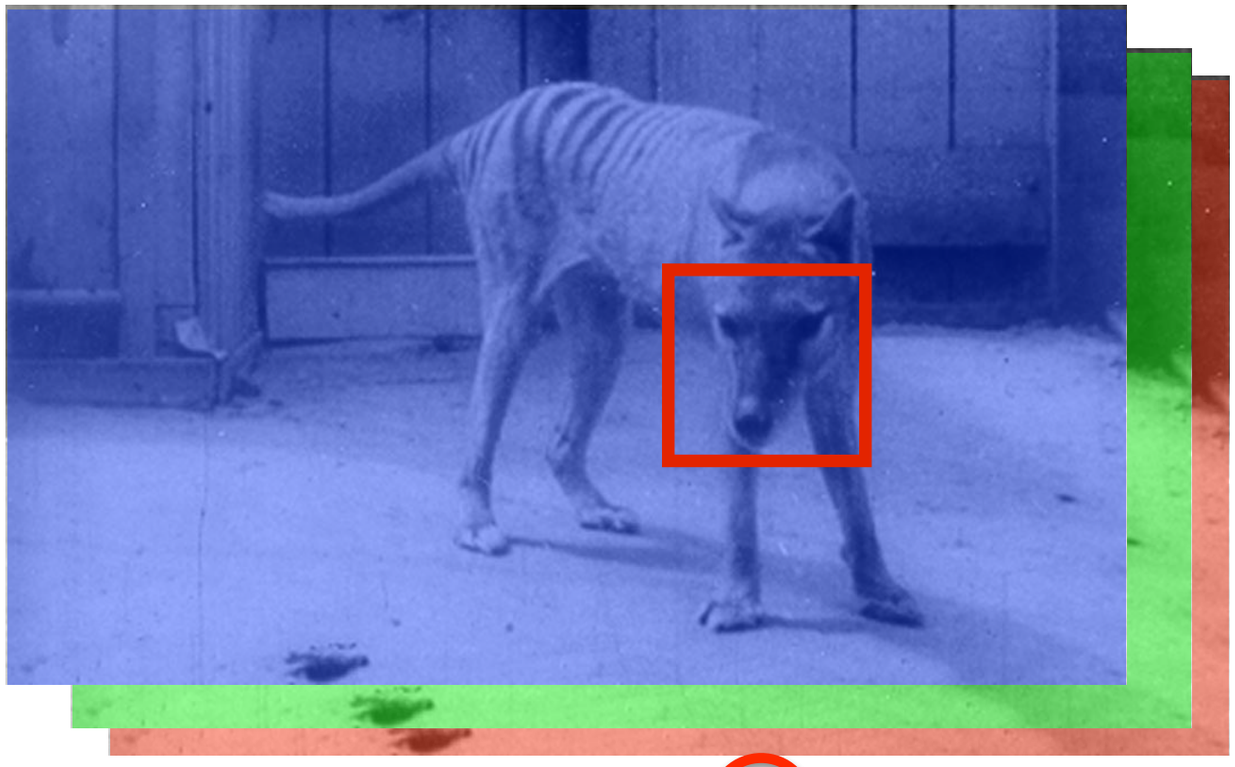


$\{\mathbf{h}^{(k)}\}_{k=1}^{K}$

K = no. of input channels

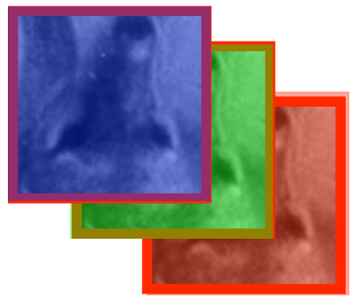


$\mathbf{y}$



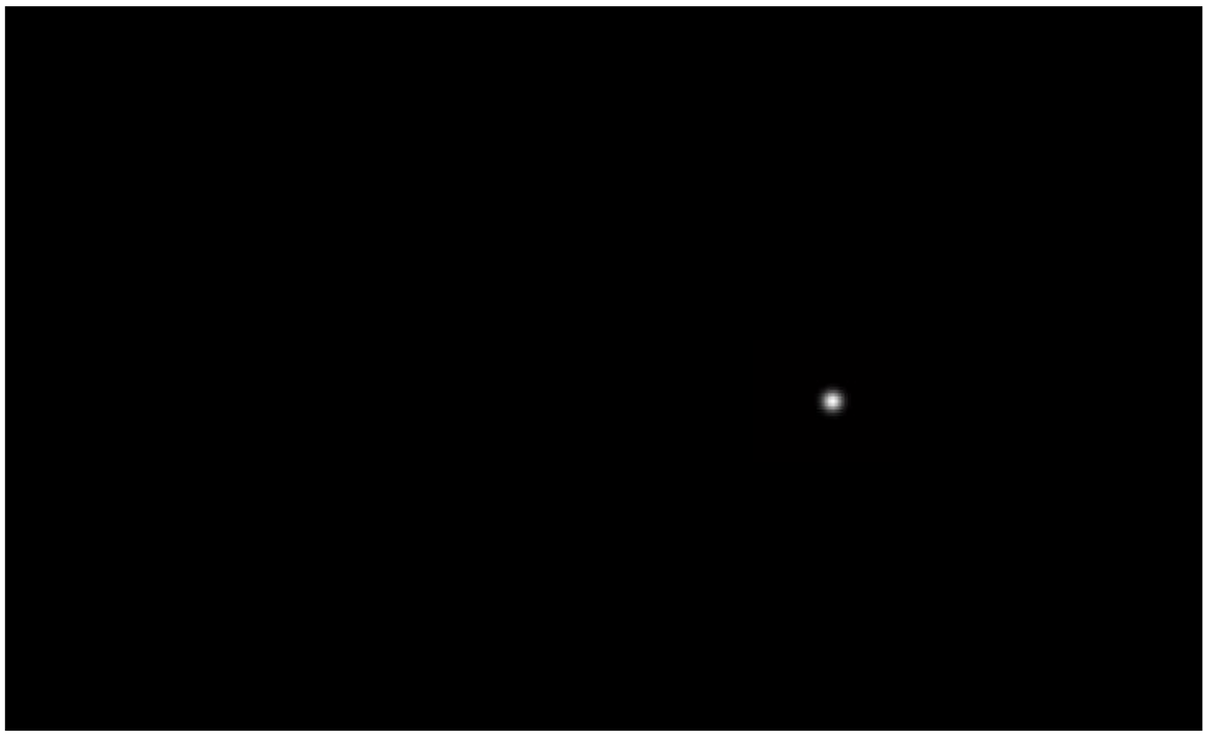
$$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$$

$*$



$$\{\mathbf{h}^{(k)}\}_{k=1}^{K}$$

K = no. of input channels



“output still single channel!!!”

Multi-Channel 2D-Convolution

$$\mathbf{Y}[j,i] = \sum_{k} \sum_{v} \sum_{u} \mathbf{X}^{(k)}[u,v] \mathbf{H}^{(k)}[j-v, i-u]$$

Multi-Channel 2D-Convolution

$$\mathbf{Y}[j,i] = \sum_{k} \sum_{v} \sum_{u} \mathbf{X}^{(k)}[u,v] \mathbf{H}^{(k)}[j-v, i-u]$$

Multi-Channel 2D-Convolution

$$\mathbf{Y}[j,i] = \sum_{k} \sum_{v} \sum_{u} \mathbf{X}^{(k)}[u,v] \mathbf{H}^{(k)}[j-v, i-u]$$

"vectorized form"

$$\mathbf{y} = \sum_{k} \mathbf{x}^{(k)} * \mathbf{h}^{(k)}$$

Multi-Channel 2D-Convolution

$$\mathbf{Y}[j,i] = \sum_{k} \sum_{v} \sum_{u} \mathbf{X}^{(k)}[u,v] \mathbf{H}^{(k)}[j-v, i-u]$$

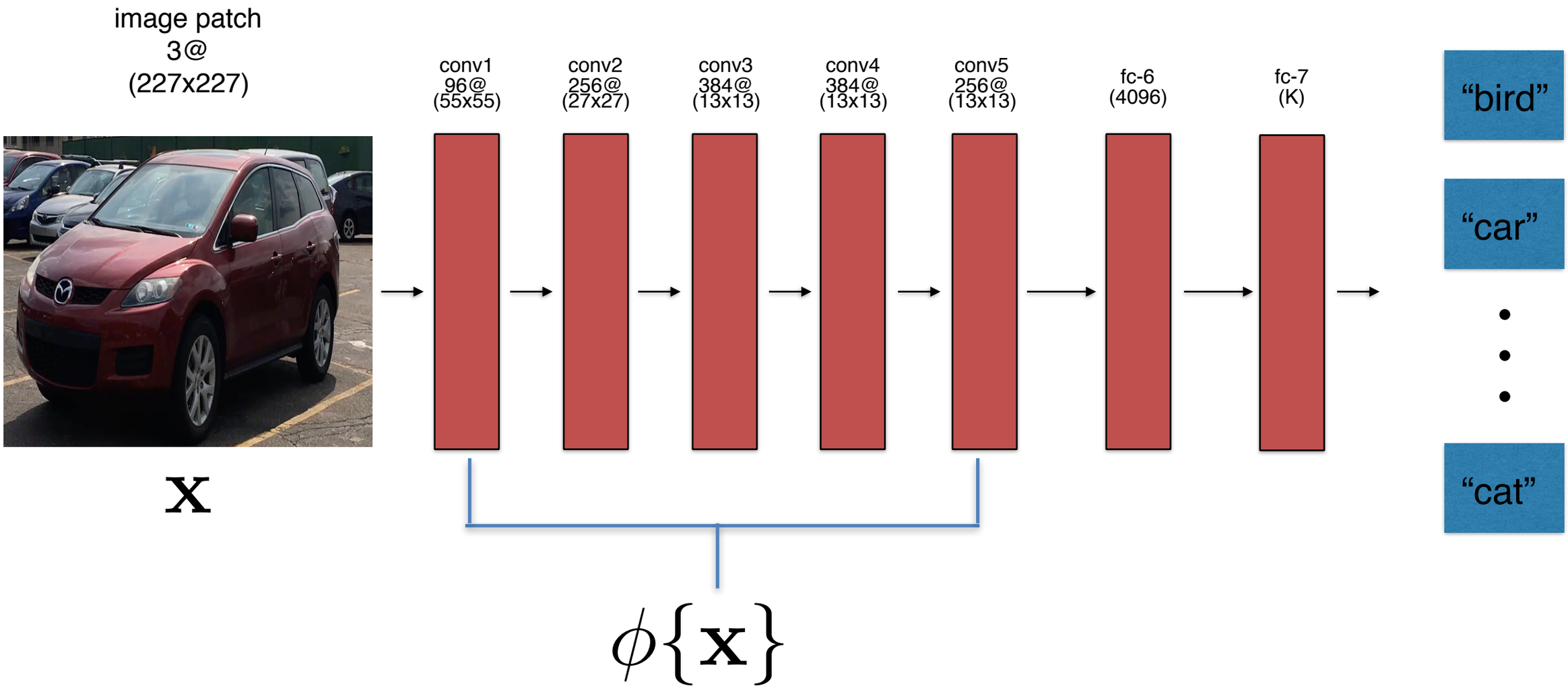
"vectorized form"

$$\mathbf{y} = \sum_{k} \mathbf{x}^{(k)} * \mathbf{h}^{(k)}$$

This is NOT 3D Convolution!!

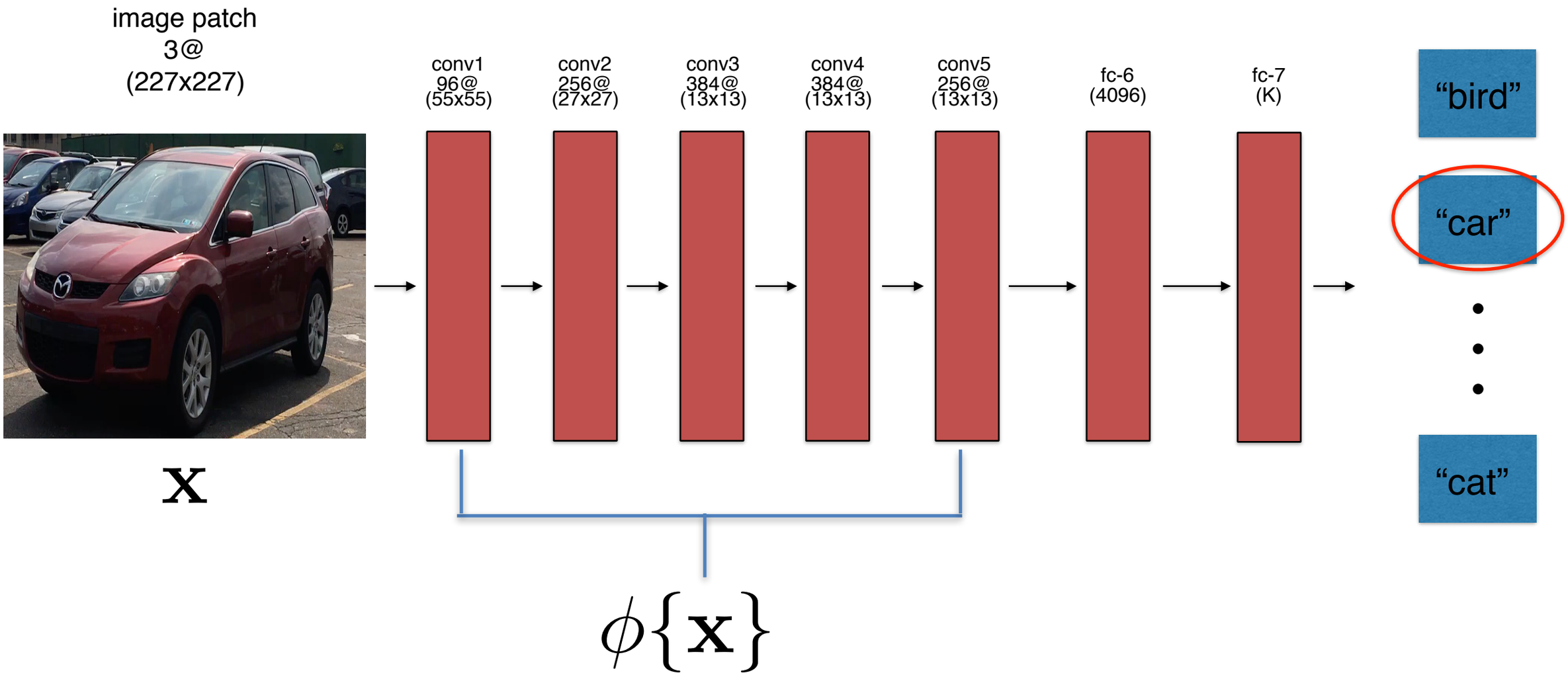
Can you show why?

Importance to Deep Learning



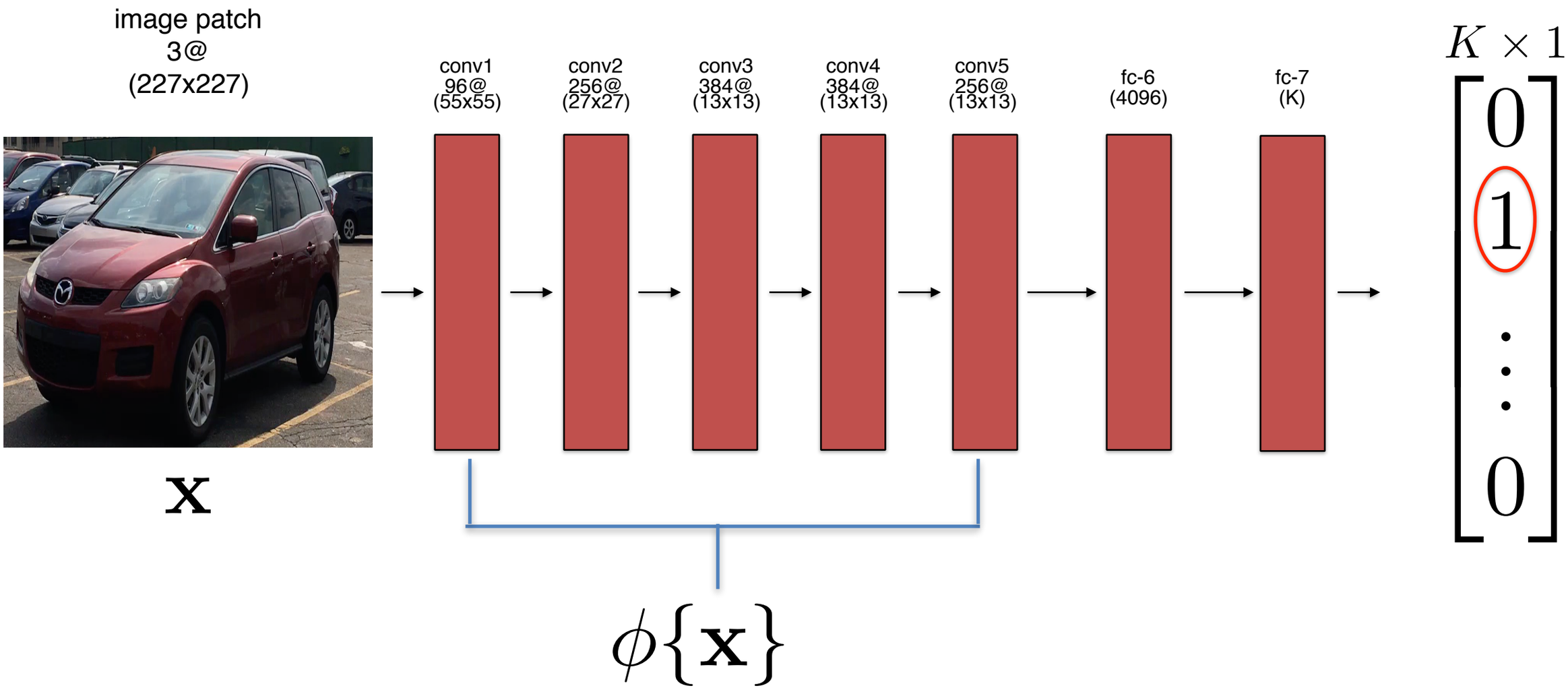
A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning



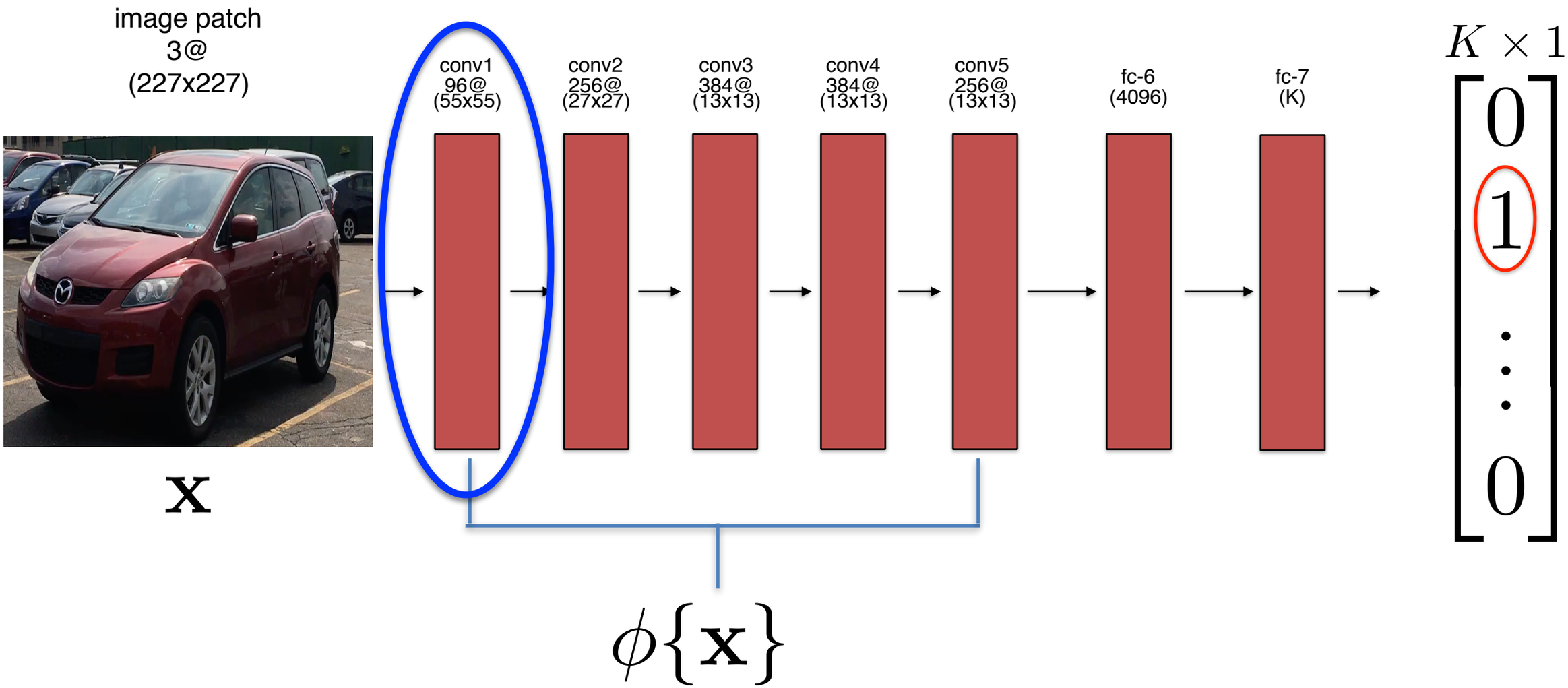
A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning



A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning



A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning

image patch
3@
(227x227)

conv1
96@
(55x55)

x

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning

image patch
3@
(227x227)

conv1
96@
(55x55)

$$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$$

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning

image patch
3@
(227x227)

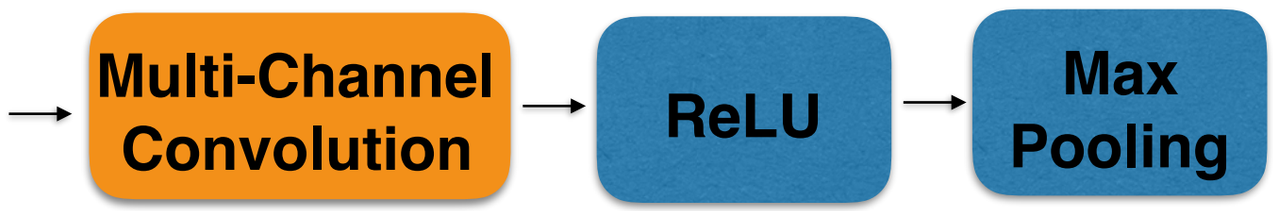
conv1
96@
(55x55)

$$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$$

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning

image patch
3@
(227x227)



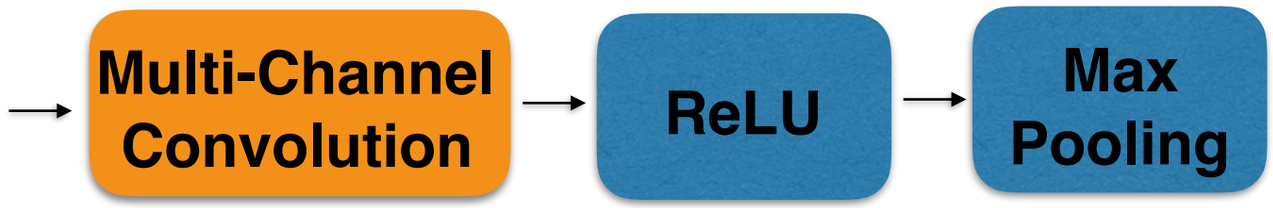
$$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$$

$$\mathbf{y}^{(j)} = \sum_{k=1}^{K} \left[\mathbf{x}^{(k)} * \mathbf{h}^{(j,k)}\right]$$

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning

image patch
3@
(227x227)



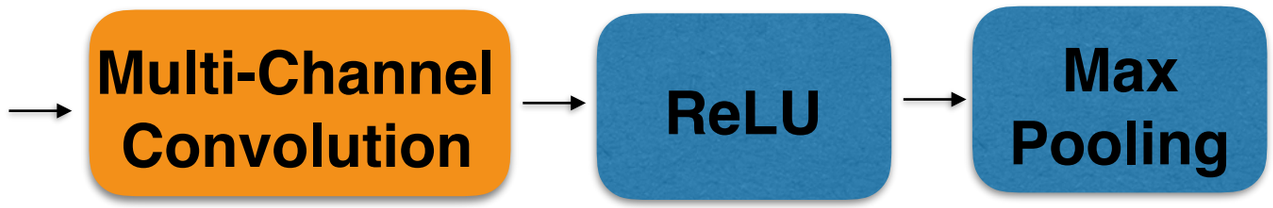
$$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$$

$$\mathbf{y}^{(j)} = \sum_{k=1}^{K} \left[\mathbf{x}^{(k)} * \mathbf{h}^{(j,k)}\right]$$

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

Importance to Deep Learning

image patch
3@
(227x227)



$$\{\mathbf{x}^{(k)}\}_{k=1}^{K}$$

$$\mathbf{y}^{(j)} = \sum_{k=1}^{K} \left[\mathbf{x}^{(k)} * \mathbf{h}^{(j,k)}\right] + b_j$$

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

PyTorch Example

```
>>> x = array([8,4,6,2,7]).astype('float')
>>> h = array([1,2]).astype('float')
```

PyTorch Example

```
>>> x = array([8,4,6,2,7]).astype('float')
>>> h = array([1,2]).astype('float')


>>> import torch
>>> xt = torch.from_numpy(reshape(x,(1,1,1,x.size)))
>>> ht = torch.from_numpy(reshape(h,(1,1,1,h.size)))
```

PyTorch Example

```
>>> x = array([8,4,6,2,7]).astype('float')
>>> h = array([1,2]).astype('float')


>>> import torch
>>> xt = torch.from_numpy(reshape(x,(1,1,1,x.size)))
>>> ht = torch.from_numpy(reshape(h,(1,1,1,h.size)))


>>> xt.shape
    torch.Size([1, 1, 1, 5])
```

PyTorch Example

```
>>> x = array([8,4,6,2,7]).astype('float')
>>> h = array([1,2]).astype('float')


>>> import torch
>>> xt = torch.from_numpy(reshape(x,(1,1,1,x.size)))
>>> ht = torch.from_numpy(reshape(h,(1,1,1,h.size)))


>>> xt.shape
    torch.Size([1, 1, 1, 5])


>>> torch.conv2d(xt,ht)
```

PyTorch Example

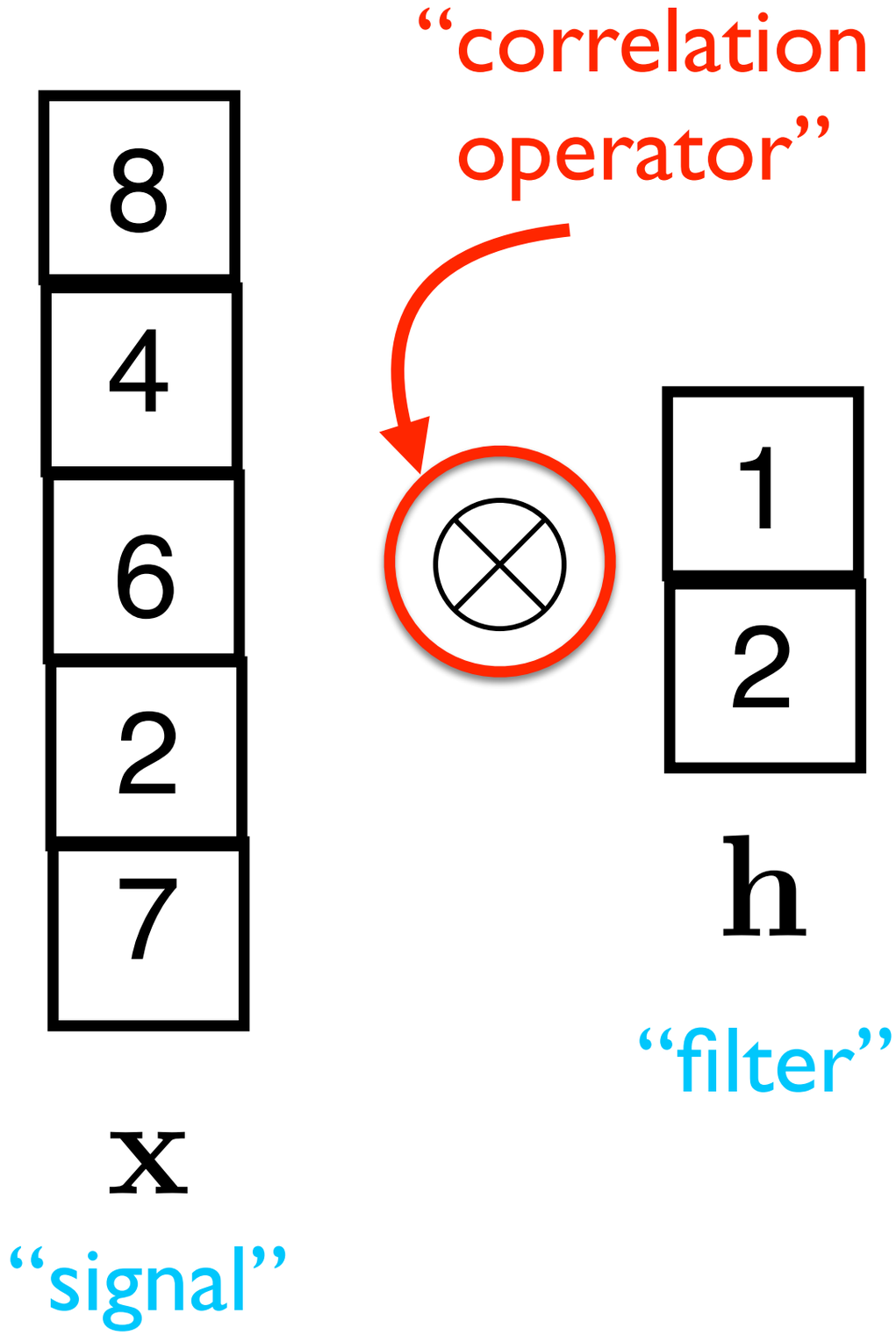
```
>>> x = array([8,4,6,2,7]).astype('float')
>>> h = array([1,2]).astype('float')


>>> import torch
>>> xt = torch.from_numpy(reshape(x,(1,1,1,x.size)))
>>> ht = torch.from_numpy(reshape(h,(1,1,1,h.size)))


>>> xt.shape
    torch.Size([1, 1, 1, 5])


>>> torch.conv2d(xt,ht)
    tensor([[[[16., 16., 10., 16.]]]], dtype=torch.fl
```

PyTorch Example

```
>>> x = array([8,4,6,2,7]).astype('float')
>>> h = array([1,2]).astype('float')


>>> import torch
>>> xt = torch.from_numpy(reshape(x,(1,1,1,x.size)))
>>> ht = torch.from_numpy(reshape(h,(1,1,1,h.size)))


>>> xt.shape
    torch.Size([1, 1, 1, 5])


>>> torch.conv2d(xt,ht)
    tensor([[[[16., 16., 10., 16.]]]], dtype=torch.fl
```

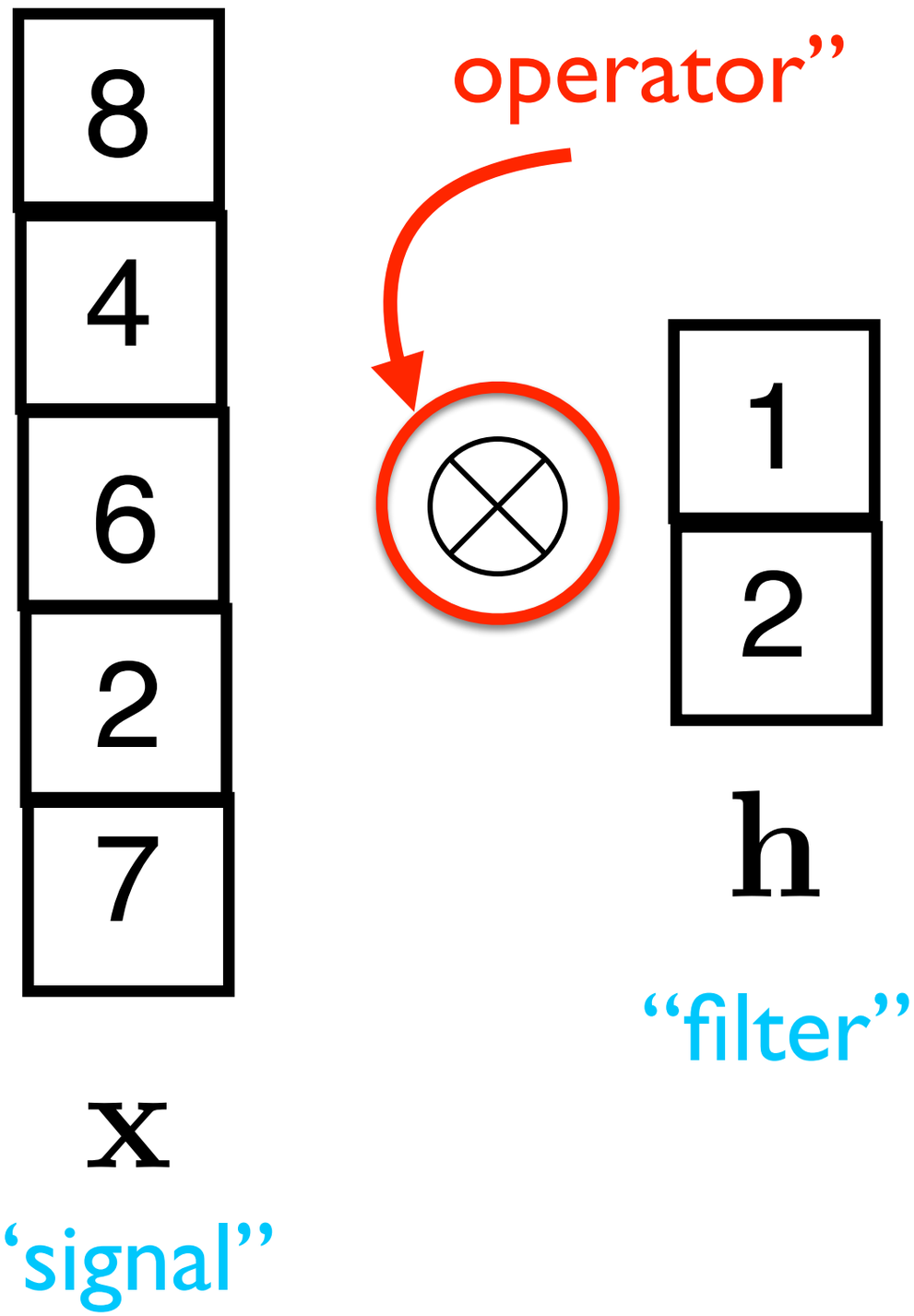
Is this convolution or correlation?

Reminder: Correlation

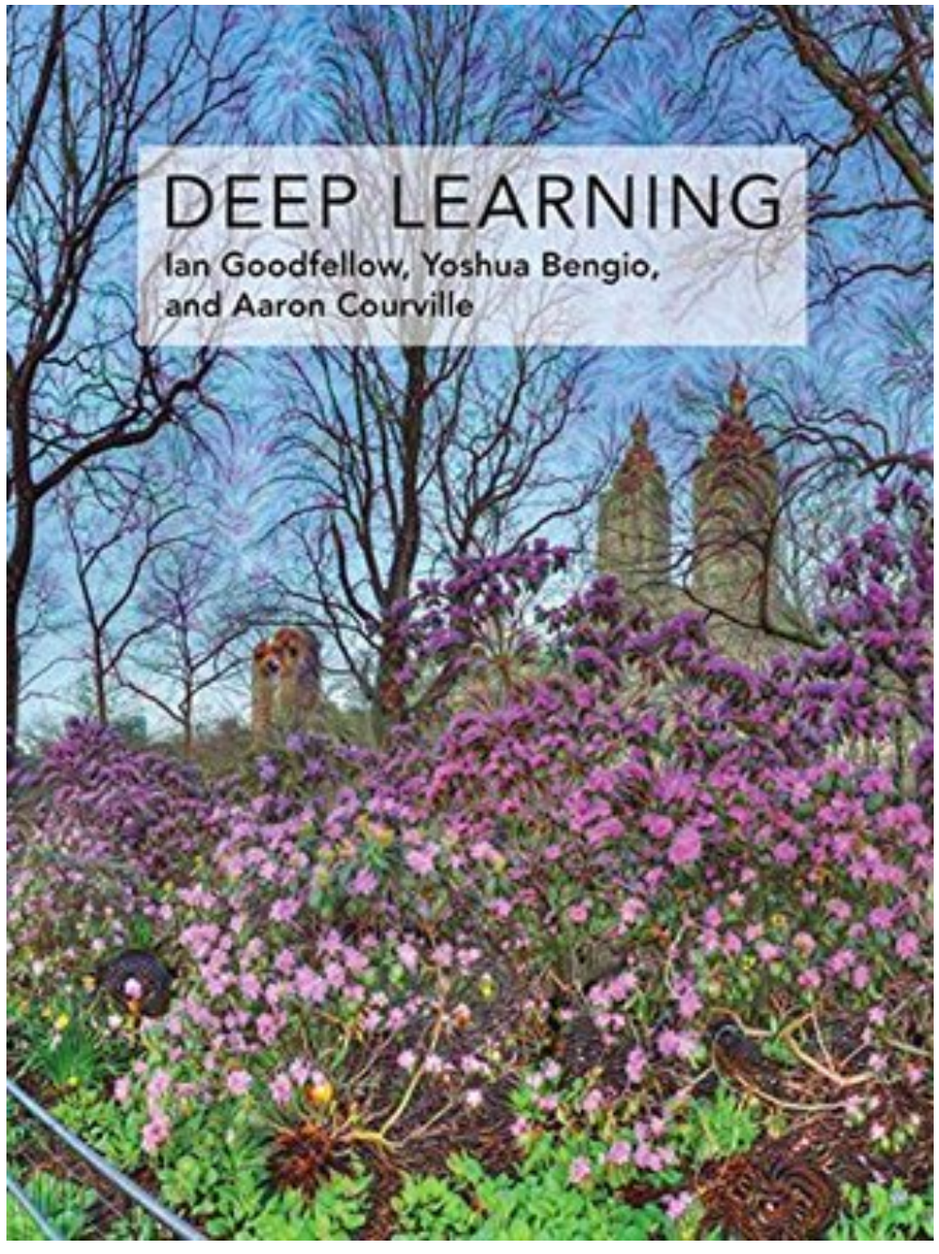


Reminder: Correlation

```
>>> from scipy.signal import \
    correlate as corr
>>> corr(x,h,'same')
array([16,16,16,10,16])
```



More to read…



- Goodfellow et al.
  - Chapter 9, Section 1.