Carnegie Mellon University
Heinz College

Image analysis with CNNs

George Chen

1 slide is by Phillip Isola

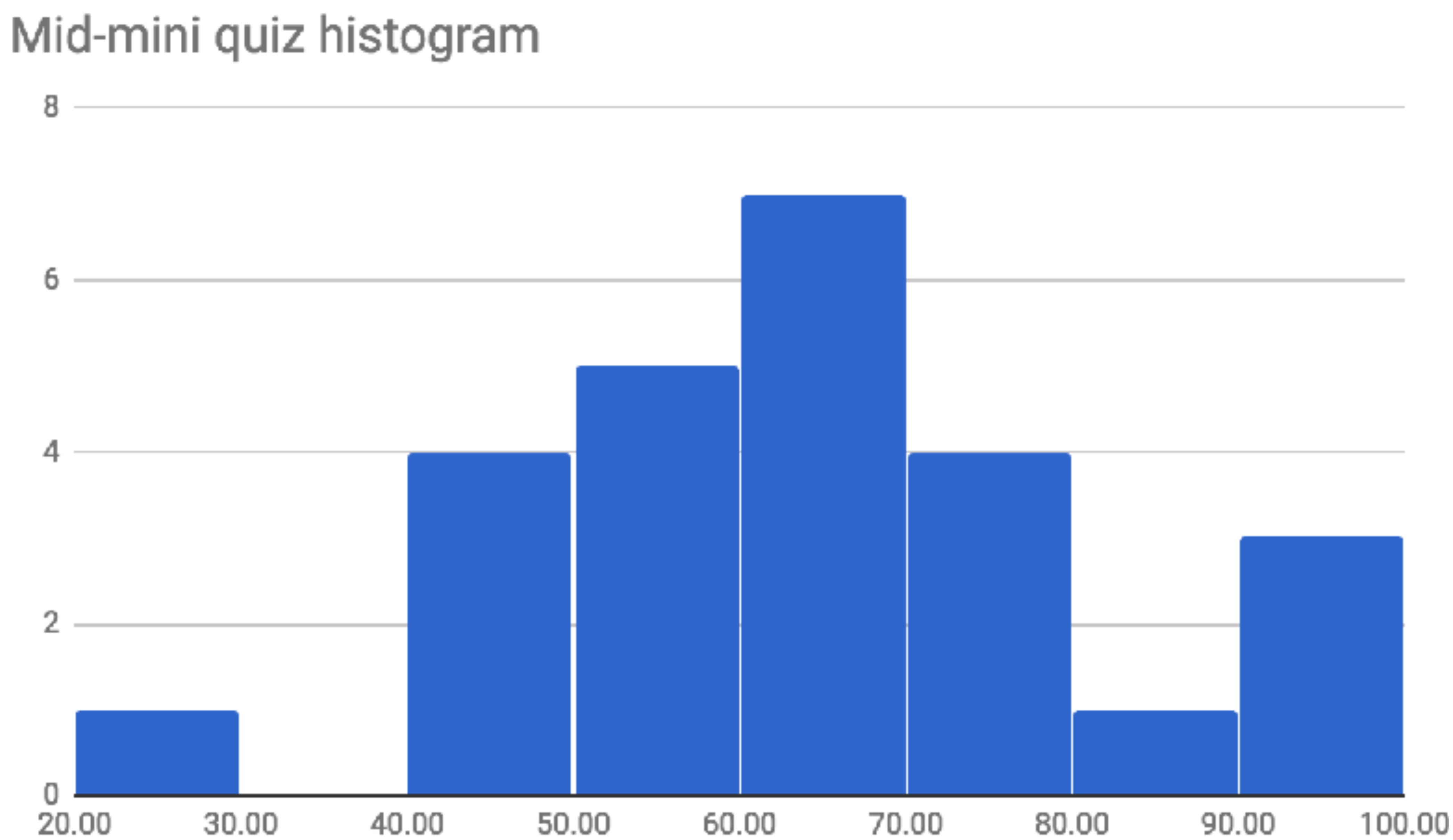
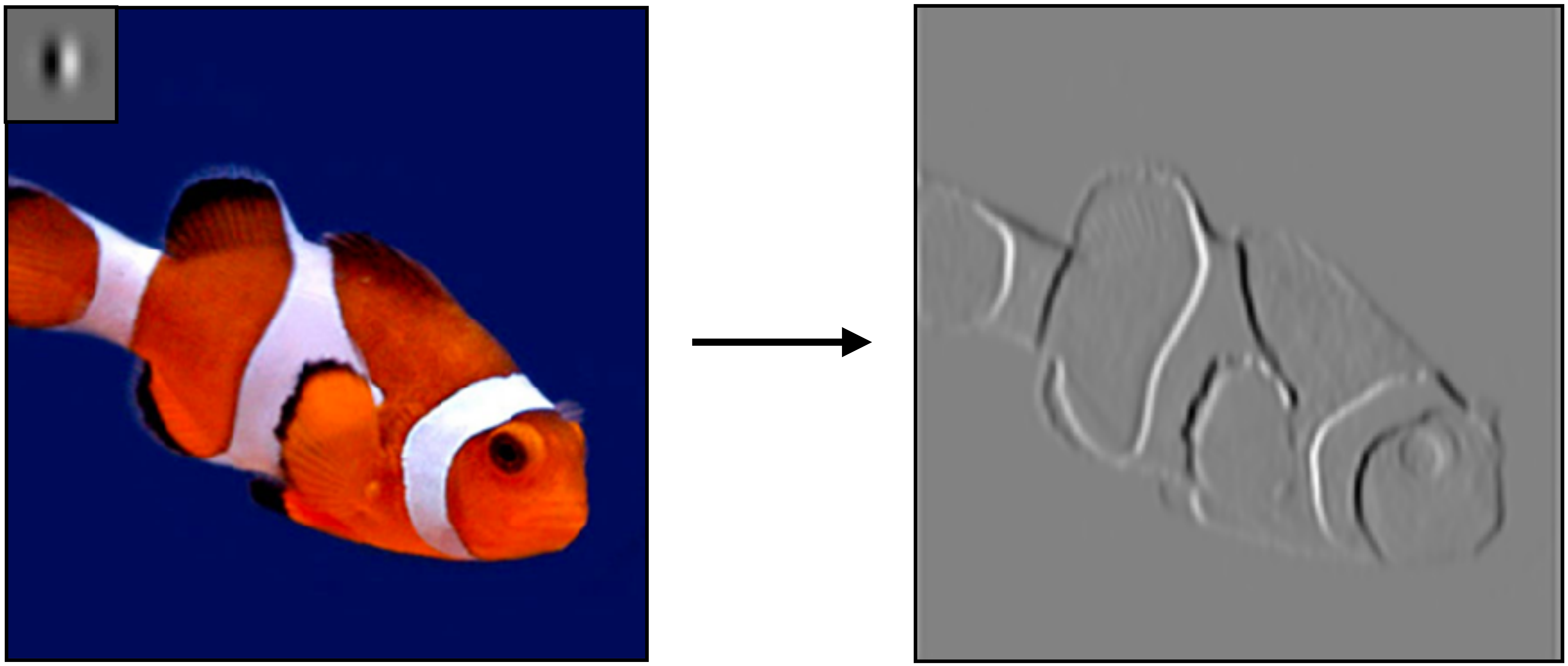
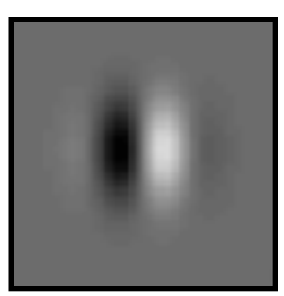

Mid-mini quiz histogram
8
6
4
2
0
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00

Image analysis with Convolutional Neural Nets (CNNs, also called convnets)

Convolution



filter 

Slide by Phillip Isola

Convolution

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input image

0	0	0
0	1	0
0	0	0

Filter
(also called “kernel”)

Convolution

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

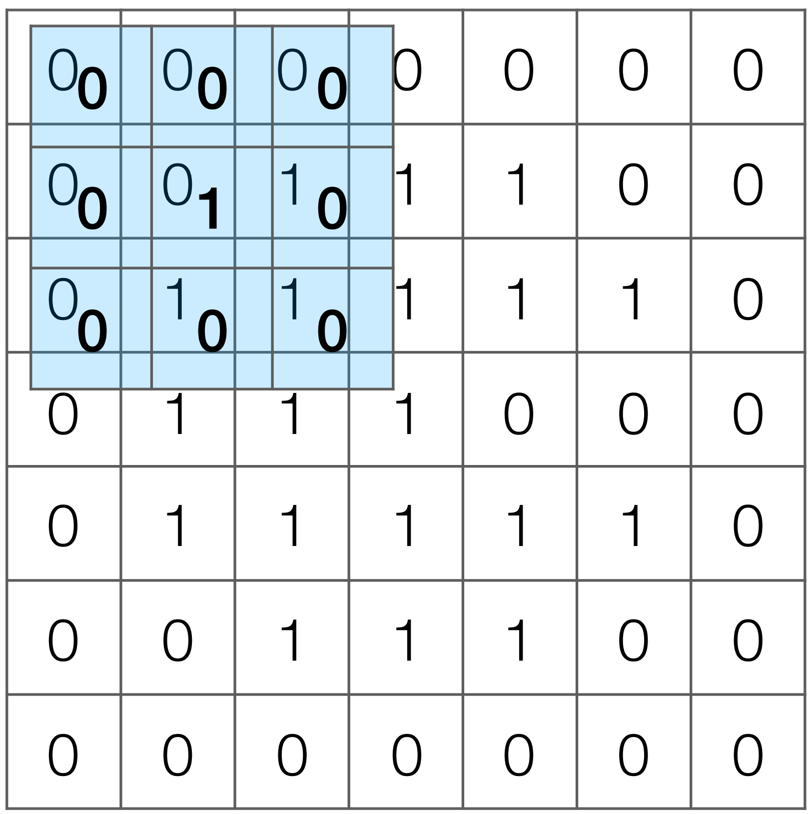
Input image

0	0	0
0	1	0
0	0	0

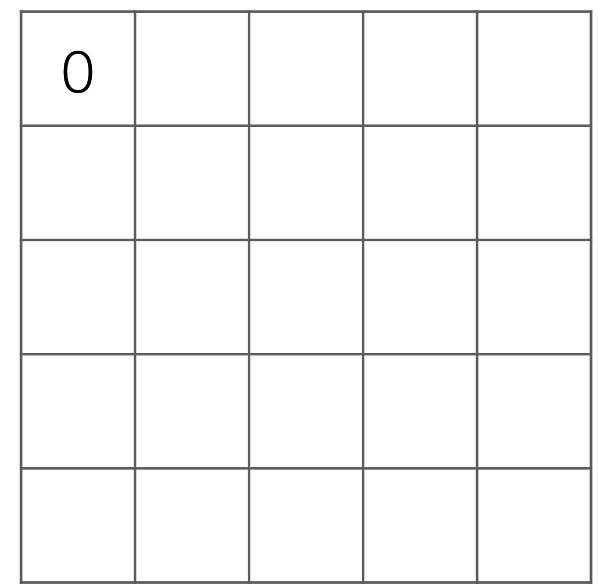
Filter
(also called “kernel”)

Convolution

Take dot product!

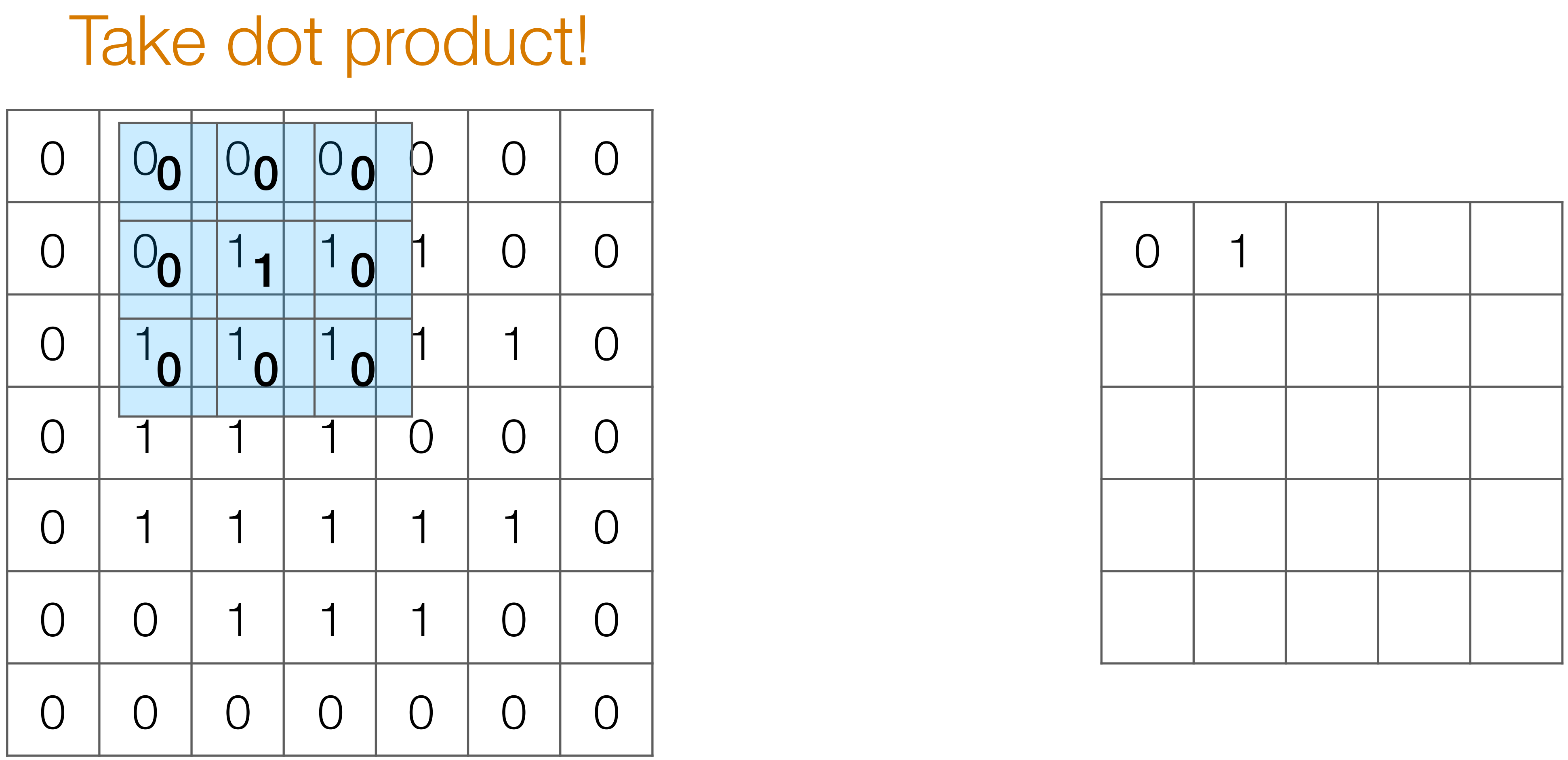


Input image



Output image

Convolution

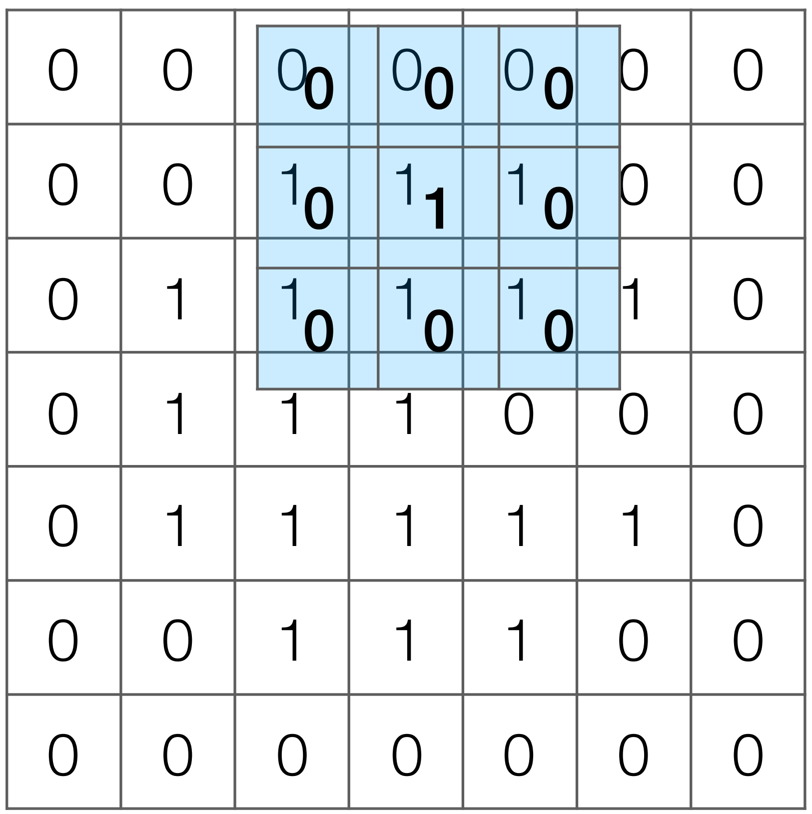


Input image

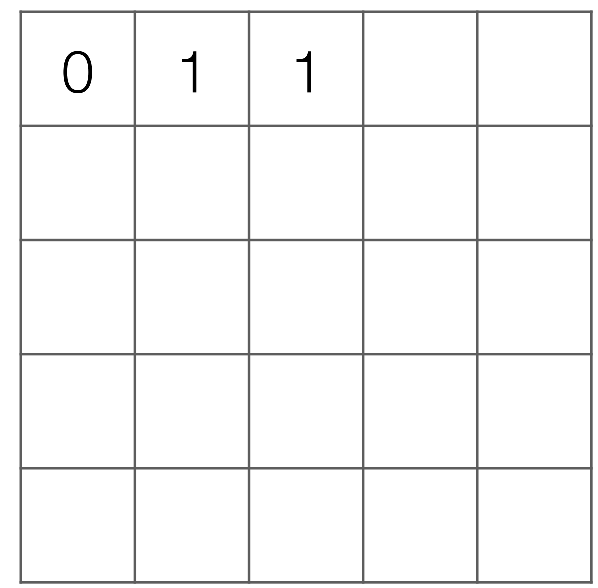
Output image

Convolution

Take dot product!

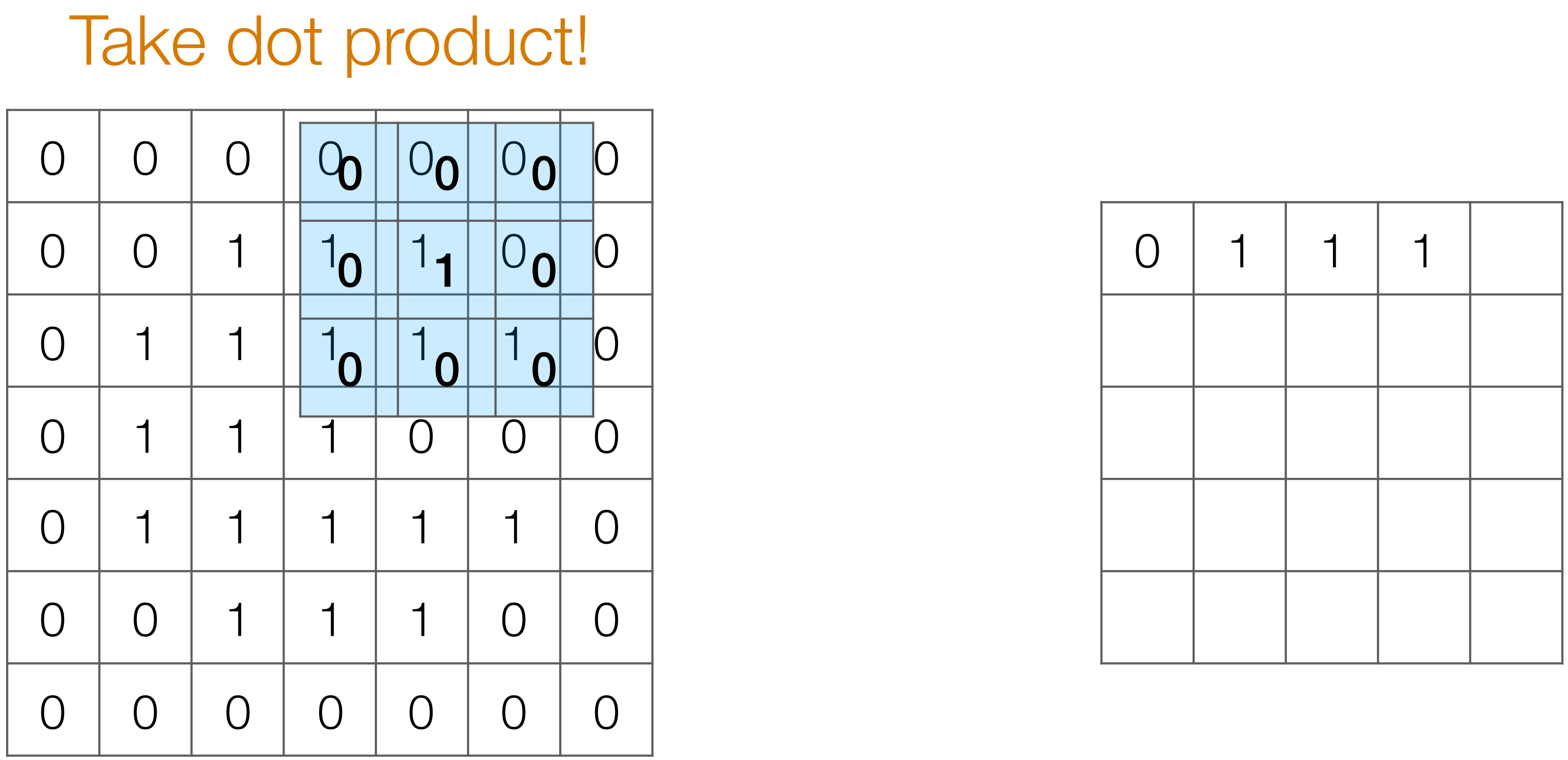


Input image



Output image

Convolution

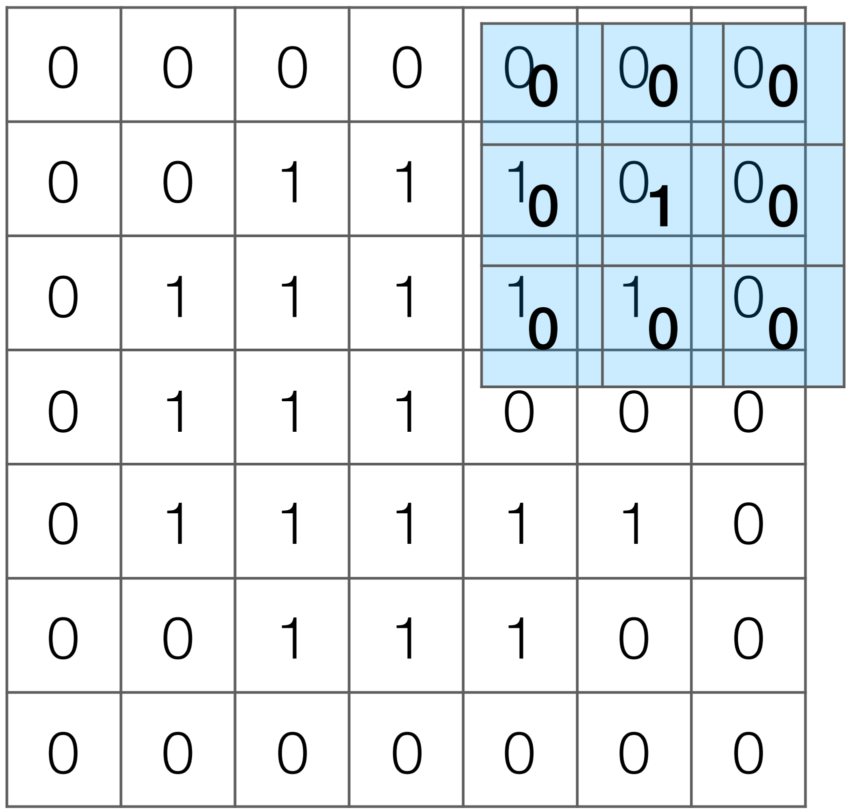


Input image

Output image

Convolution

Take dot product!

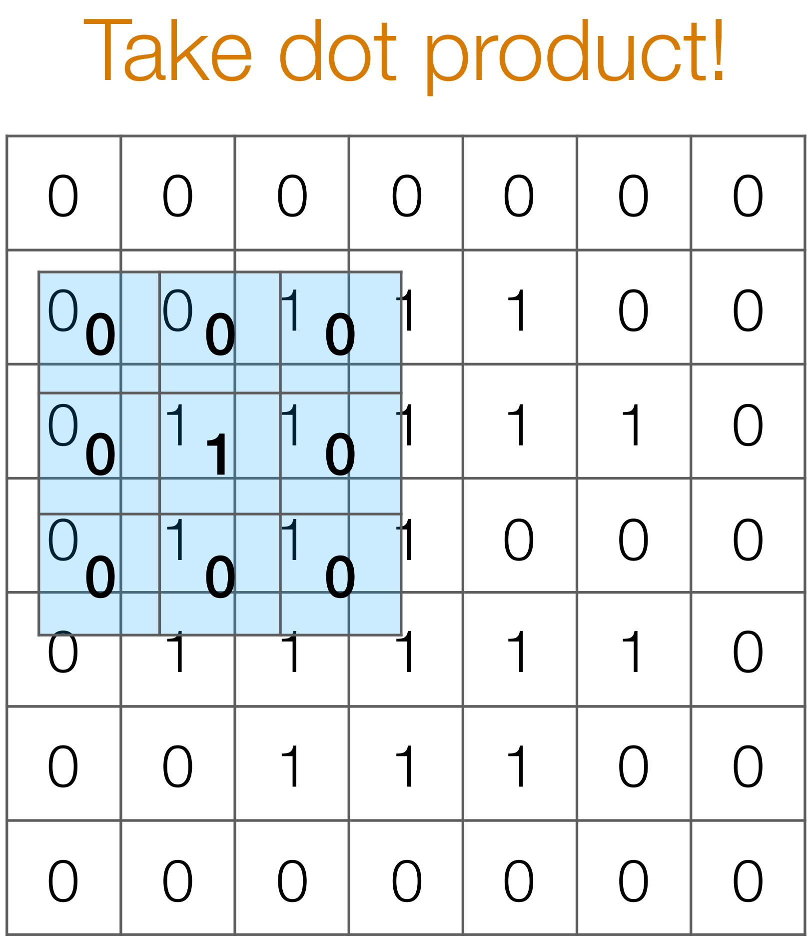


Input image

0	1	1	1	0

Output image

Convolution



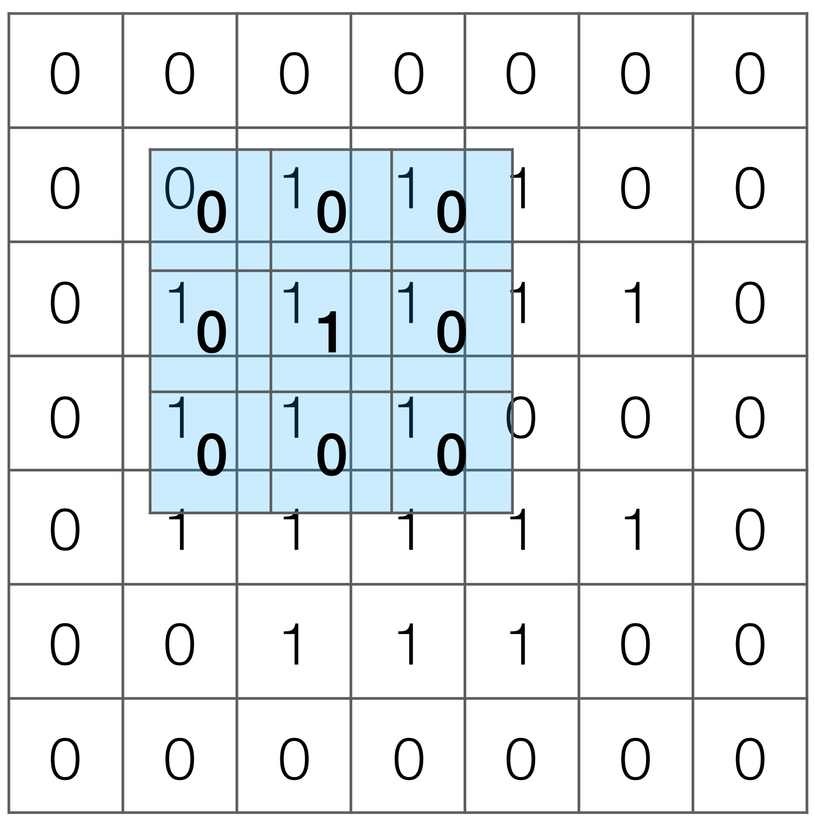
Input image

0	1	1	1	0
1				

Output image

Convolution

Take dot product!



Input image

0	1	1	1	0
1	1			

Output image

Convolution

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{1} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix}
=
\begin{bmatrix}
0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 0
\end{bmatrix}
$$

Input image

Output image

Note: output image is smaller than input image

If you want output size to be same as input, pad 0's to input

Convolution

$$
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \hline
\mathbf{0}&0&0&0&0&0&0&0&\mathbf{0}\\ \hline
\mathbf{0}&0&0&1&1&1&0&0&\mathbf{0}\\ \hline
\mathbf{0}&0&1&1&1&1&1&0&\mathbf{0}\\ \hline
\mathbf{0}&0&1&1&1&0&0&0&\mathbf{0}\\ \hline
\mathbf{0}&0&1&1&1&1&1&0&\mathbf{0}\\ \hline
\mathbf{0}&0&0&1&1&1&0&0&\mathbf{0}\\ \hline
\mathbf{0}&0&0&0&0&0&0&0&\mathbf{0}\\ \hline
\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \hline
\end{array}
*
\begin{array}{|c|c|c|}
\hline
\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \hline
\mathbf{0}&\mathbf{1}&\mathbf{0}\\ \hline
\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \hline
\end{array}
=
\begin{array}{|c|c|c|c|c|c|c|}
\hline
0&0&0&0&0&0&0\\ \hline
0&0&1&1&1&0&0\\ \hline
0&1&1&1&1&1&0\\ \hline
0&1&1&1&0&0&0\\ \hline
0&1&1&1&1&1&0\\ \hline
0&0&1&1&1&0&0\\ \hline
0&0&0&0&0&0&0\\ \hline
\end{array}
$$

Input image

Output image

Note: output image is smaller than input image

If you want output size to be same as input, pad 0's to input

Convolution

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input image

*

0	**0**	**0**
0	**1**	**0**
0	**0**	**0**

=

0	1	1	1	0
1	1	1	1	1
1	1	1	0	0
1	1	1	1	1
0	1	1	1	0

Output image

Convolution

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
* \frac{1}{9}
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
= \frac{1}{9}
\begin{bmatrix}
3 & 5 & 6 & 5 & 3 \\
5 & 8 & 8 & 6 & 3 \\
6 & 9 & 8 & 7 & 4 \\
5 & 8 & 8 & 6 & 3 \\
3 & 5 & 6 & 5 & 3
\end{bmatrix}
$$

Input image

Output image

Convolution

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
\mathbf{-1} & \mathbf{-1} & \mathbf{-1} \\
\mathbf{2} & \mathbf{2} & \mathbf{2} \\
\mathbf{-1} & \mathbf{-1} & \mathbf{-1}
\end{bmatrix}
=
\begin{bmatrix}
0 & 1 & 3 & 1 & 0 \\
1 & 1 & 1 & 3 & 3 \\
0 & 0 & -2 & -4 & -4 \\
1 & 1 & 1 & 3 & 3 \\
0 & 1 & 3 & 1 & 0
\end{bmatrix}
$$

Input image

Output image

Convolution

Very commonly used for:

- Blurring an image

*

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

=

- Finding edges

*

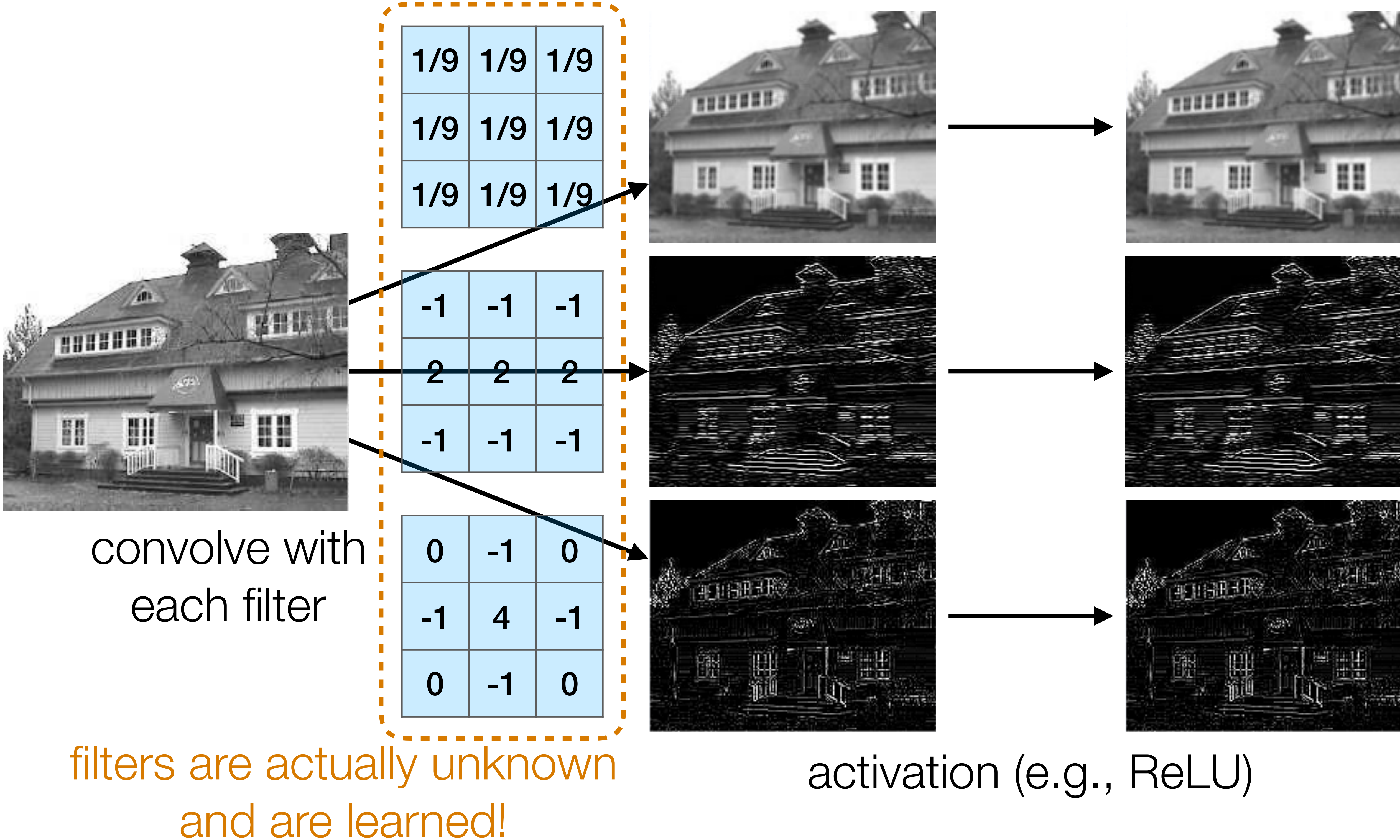
-1	-1	-1
2	2	2
-1	-1	-1

=

(this example finds horizontal edges)

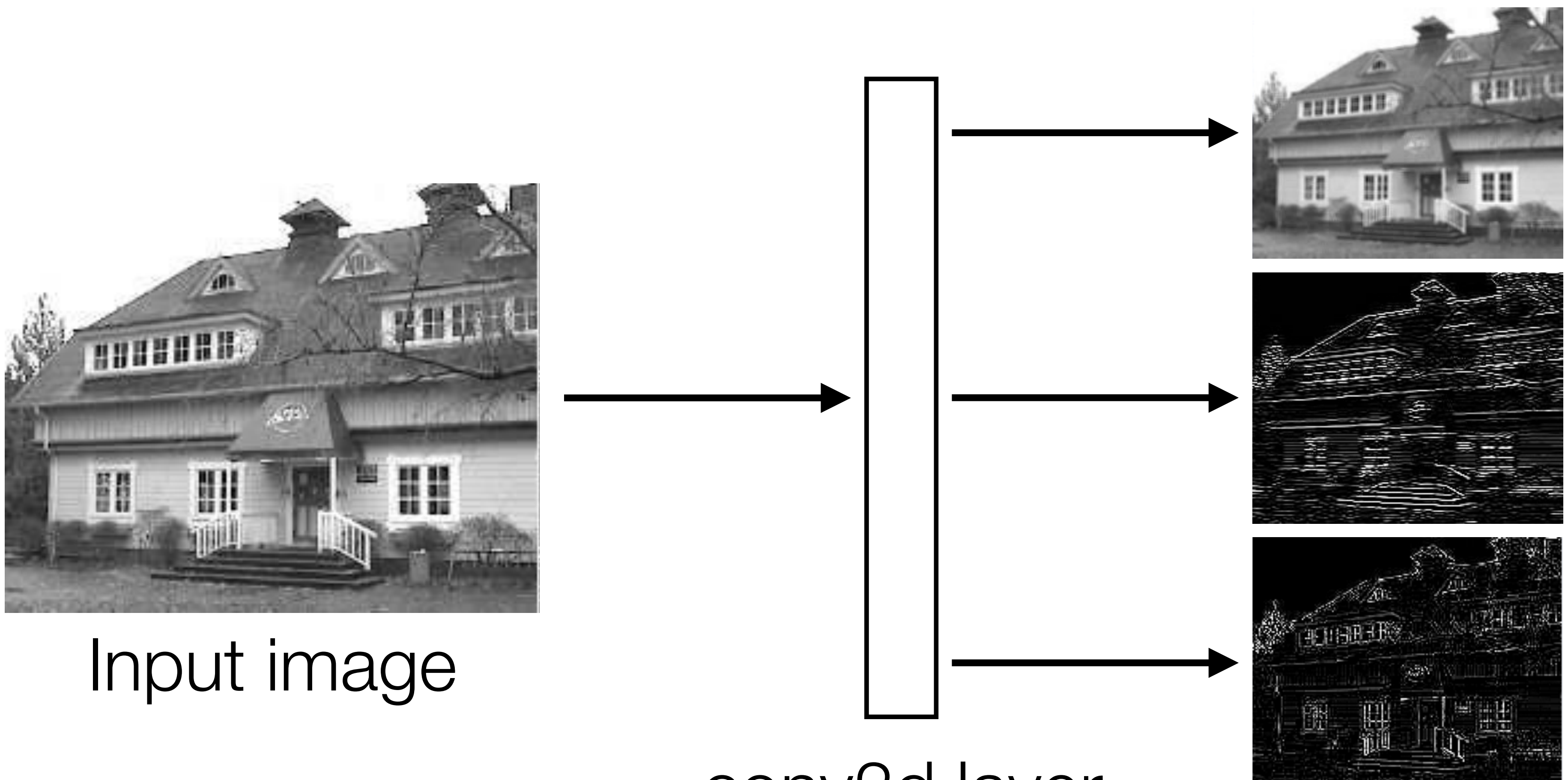
Images from: http://aishack.in/tutorials/image-convolution-examples/

Convolution Layer



Images from: http://aishack.in/tutorials/image-convolution-examples/

Convolution Layer



Input image

Output images

conv2d layer
with ReLu activation
and a three 3x3 kernels

Images from: http://aishack.in/tutorials/image-convolution-examples/

Convolution Layer

Input image

dimensions:
height,
width

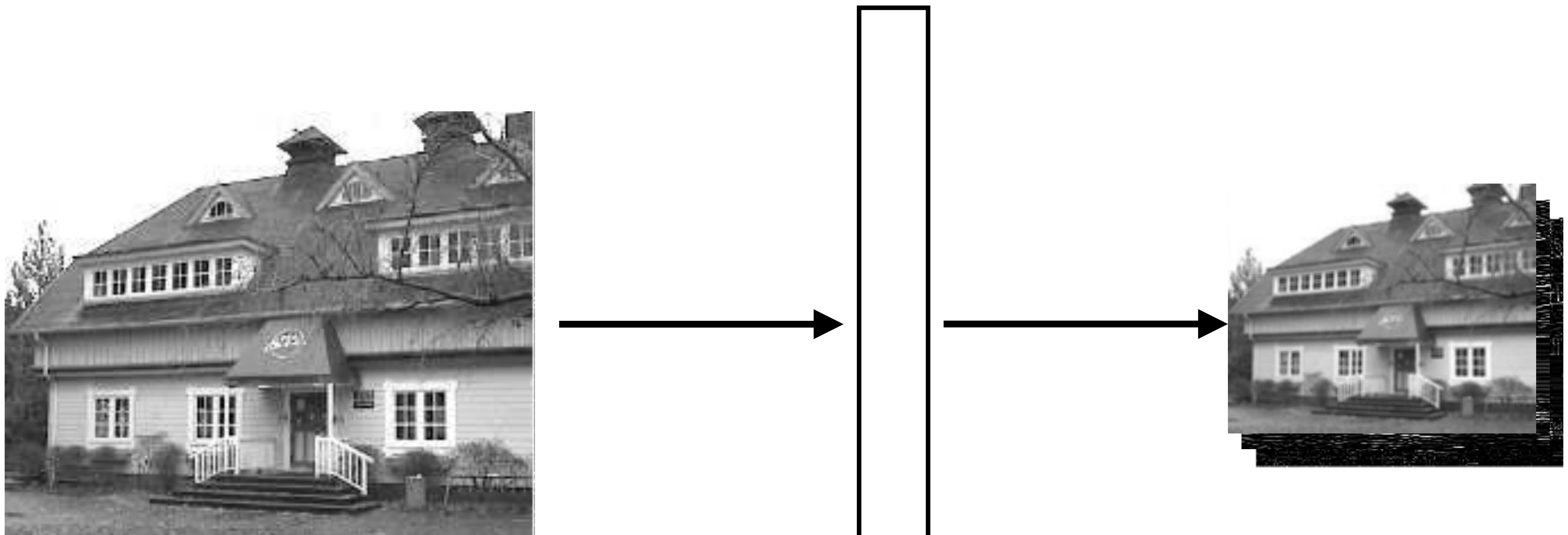
conv2d layer
with ReLu activation
and a three 3x3 kernels

Stack output images into a single "output feature map"

dimensions:
height-2,
width-2,
number of kernels
(3 in this case)

Images from: http://aishack.in/tutorials/image-convolution-examples/

Convolution Layer



Input image

dimensions:
height,
width

conv2d layer
with ReLu activation
and *k* 3x3 kernels

Stack output images into a single “output feature map”

dimensions:
height-2,
width-2,
k

Images from: http://aishack.in/tutorials/image-convolution-examples/

Convolution Layer

Input image

dimensions:
height,
width,
depth *d* (# channels)

conv2d layer
with ReLu activation
and *k* 3x3x*d* kernels

technical detail: there's
also a bias vector

Stack output images into a single "output feature map"

dimensions:
height-2,
width-2,
k

Images from: http://aishack.in/tutorials/image-convolution-examples/

Pooling

- Aggregate local information

- Produces a smaller image
(each resulting pixel captures some “global” information)

Max Pooling

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \mathbf{-1} & \mathbf{-1} & \mathbf{-1} \\ \mathbf{2} & \mathbf{2} & \mathbf{2} \\ \mathbf{-1} & \mathbf{-1} & \mathbf{-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 3 & 1 & 0 \\ 1 & 1 & 1 & 3 & 3 \\ 0 & 0 & -2 & -4 & -4 \\ 1 & 1 & 1 & 3 & 3 \\ 0 & 1 & 3 & 1 & 0 \end{bmatrix}$$

Input image

Max Pooling

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input image

*

-1	**-1**	**-1**
2	**2**	**2**
-1	**-1**	**-1**

=

0	1	3	1	0
1	1	1	3	3
0	0	-2	-4	-4
1	1	1	3	3
0	1	3	1	0

0	1	3	1	0
1	1	1	3	3
0	0	0	0	0
1	1	1	3	3
0	1	3	1	0

Output image after ReLU

Output after max pooling

Max Pooling

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input image

*

-1	**-1**	**-1**
2	**2**	**2**
-1	**-1**	**-1**

=

0	1	3	1	0
1	1	1	3	3
0	0	-2	-4	-4
1	1	1	3	3
0	1	3	1	0

0	1	3	1	0
1	1	1	3	3
0	0	0	0	0
1	1	1	3	3
0	1	3	1	0

Output image after ReLU

1	

Output after max pooling

Max Pooling

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
\mathbf{-1} & \mathbf{-1} & \mathbf{-1} \\
\mathbf{2} & \mathbf{2} & \mathbf{2} \\
\mathbf{-1} & \mathbf{-1} & \mathbf{-1}
\end{bmatrix}
=
\begin{bmatrix}
0 & 1 & 3 & 1 & 0 \\
1 & 1 & 1 & 3 & 3 \\
0 & 0 & -2 & -4 & -4 \\
1 & 1 & 1 & 3 & 3 \\
0 & 1 & 3 & 1 & 0
\end{bmatrix}
$$

Input image

0	1	3	1	0
1	1	1	3	3
0	0	0	0	0
1	1	1	3	3
0	1	3	1	0

Output image after ReLU

1	3

Output after max pooling

Max Pooling

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input image

*

-1	**-1**	**-1**
2	**2**	**2**
-1	**-1**	**-1**

=

0	1	3	1	0
1	1	1	3	3
0	0	-2	-4	-4
1	1	1	3	3
0	1	3	1	0

0	1	3	1	0
1	1	1	3	3
0	0	0	0	0
1	1	1	3	3
0	1	3	1	0

Output image after ReLU

1	3
1	

Output after max pooling

Max Pooling

$$\begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&1&1&1&1&1&0\\ 0&1&1&1&0&0&0\\ 0&1&1&1&1&1&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0 \end{bmatrix} * \begin{bmatrix} \mathbf{-1}&\mathbf{-1}&\mathbf{-1}\\ \mathbf{2}&\mathbf{2}&\mathbf{2}\\ \mathbf{-1}&\mathbf{-1}&\mathbf{-1} \end{bmatrix} = \begin{bmatrix} 0&1&3&1&0\\ 1&1&1&3&3\\ 0&0&-2&-4&-4\\ 1&1&1&3&3\\ 0&1&3&1&0 \end{bmatrix}$$

Input image

0	1	3	1	0
1	1	1	3	3
0	0	0	0	0
1	1	1	3	3
0	1	3	1	0

Output image after ReLU

1	3
1	3

Output after max pooling

Max Pooling

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input image

*

-1	**-1**	**-1**
2	**2**	**2**
-1	**-1**	**-1**

=

0	1	3	1	0
1	1	1	3	3
0	0	-2	-4	-4
1	1	1	3	3
0	1	3	1	0

0	1	3	1	0
1	1	1	3	3
0	0	0	0	0
1	1	1	3	3
0	1	3	1	0

Output image after ReLU

What numbers were involved in computing this 1?

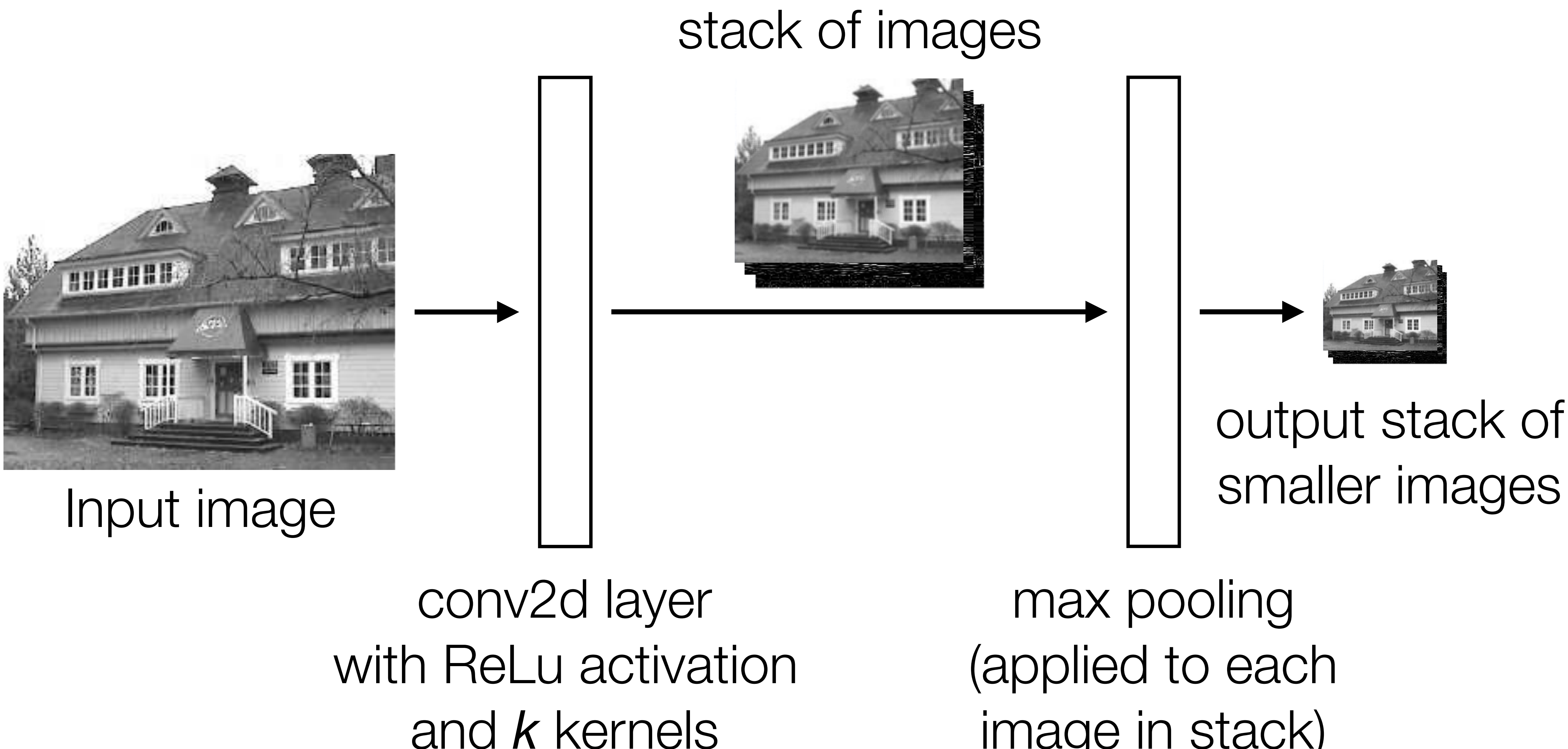
1	3
1	3

Output after max pooling

In this example: 1 pixel in max pooling output captures information from 16 input pixels!

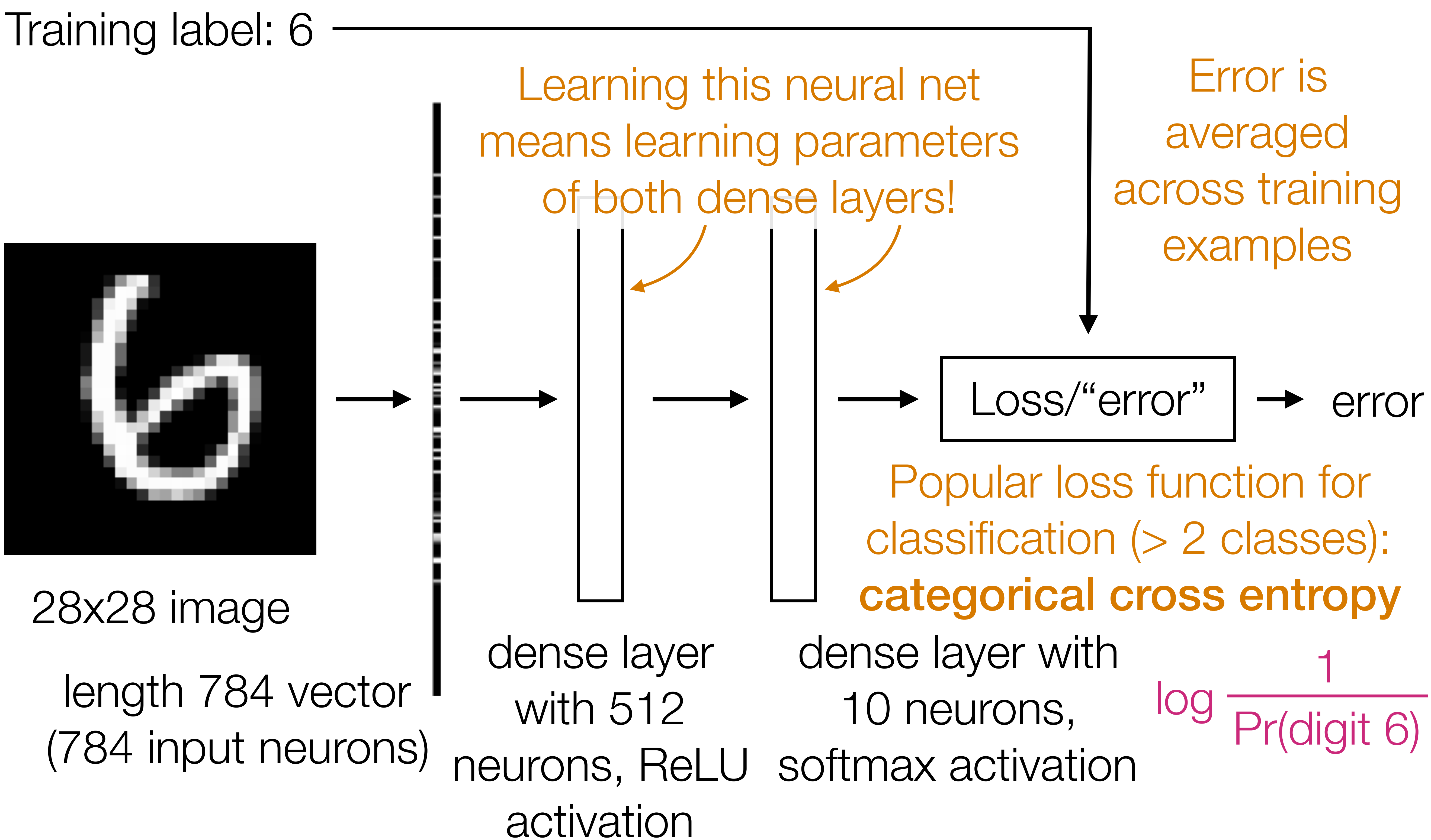
Example: applying max pooling again results in a single pixel that captures info from entire input image!

Basic Building Block of CNN's

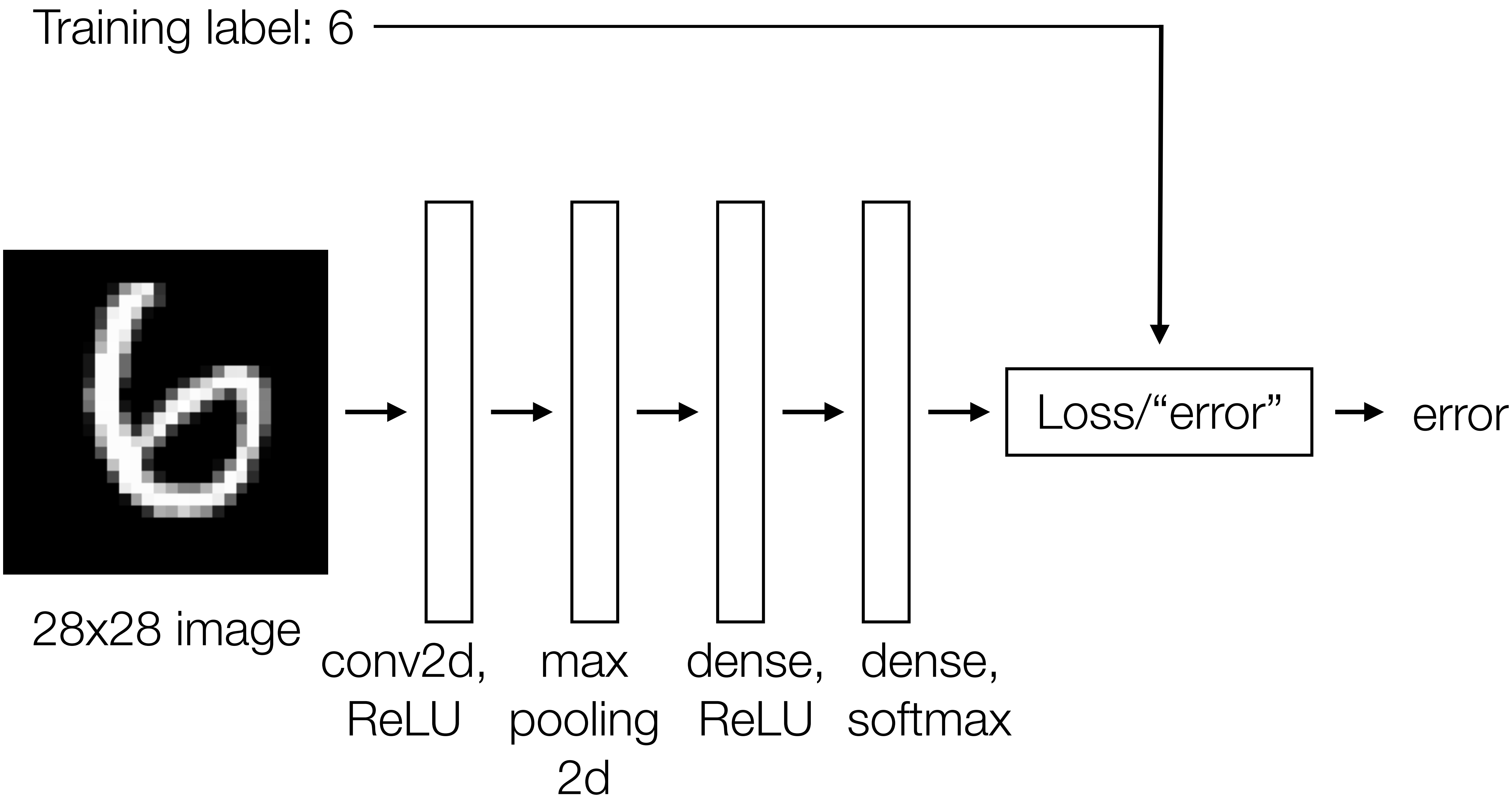


Images from: http://aishack.in/tutorials/image-convolution-examples/

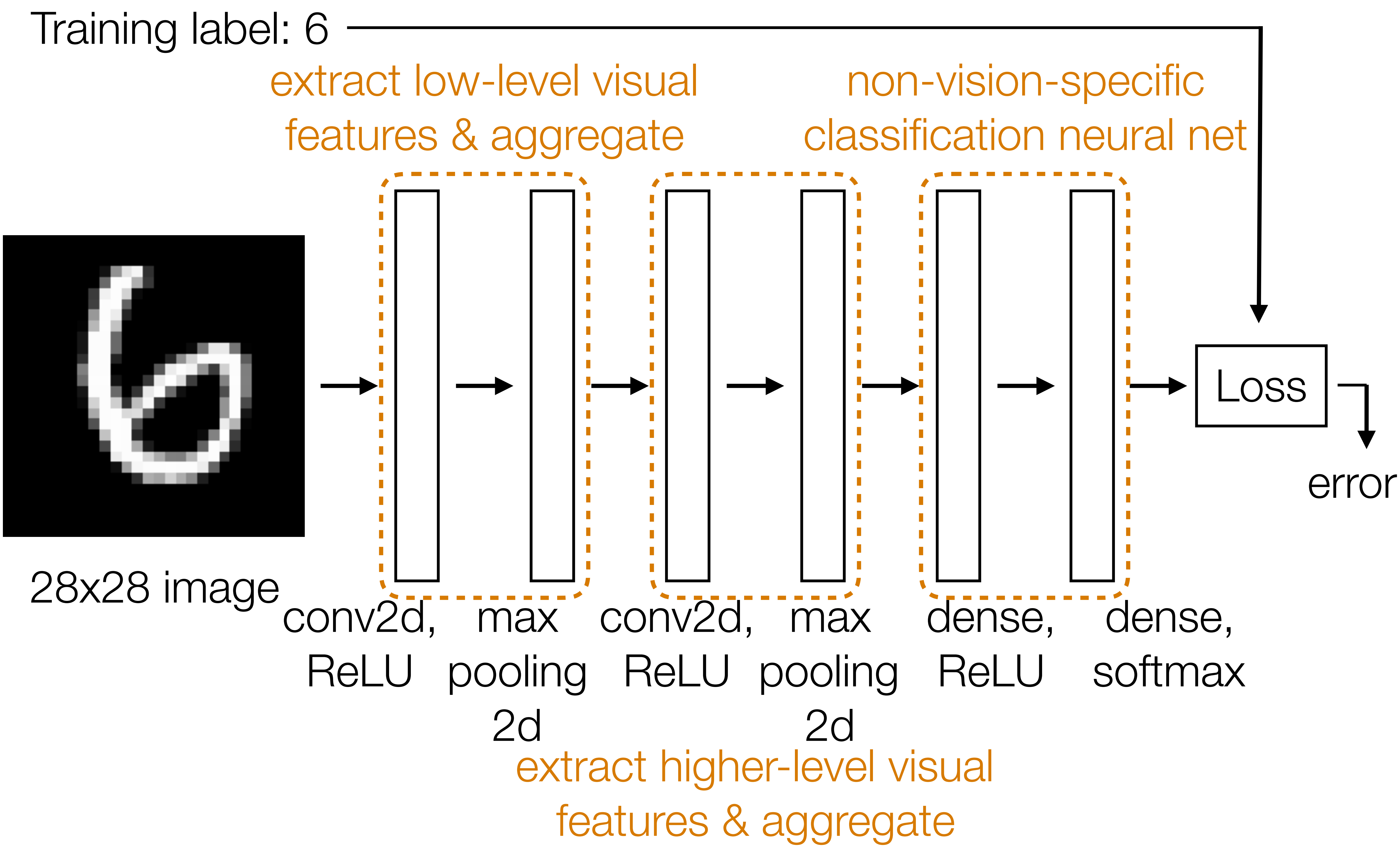
Handwritten Digit Recognition



Handwritten Digit Recognition



Handwritten Digit Recognition



CNN Demo

CNN's

- Learn convolution filters for extracting simple features
- Max pooling aggregates local information
- Can then repeat the above two layers to learn features from increasingly higher-level representations
- Convolution filters are shift-invariant
- In terms of invariance to an object shifting within the input image, this is roughly achieved by pooling