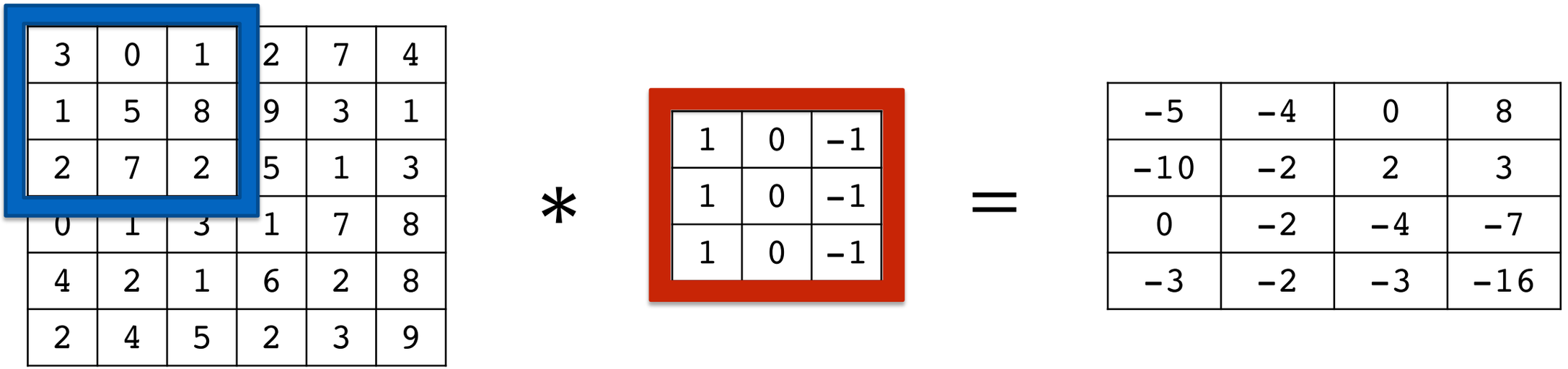
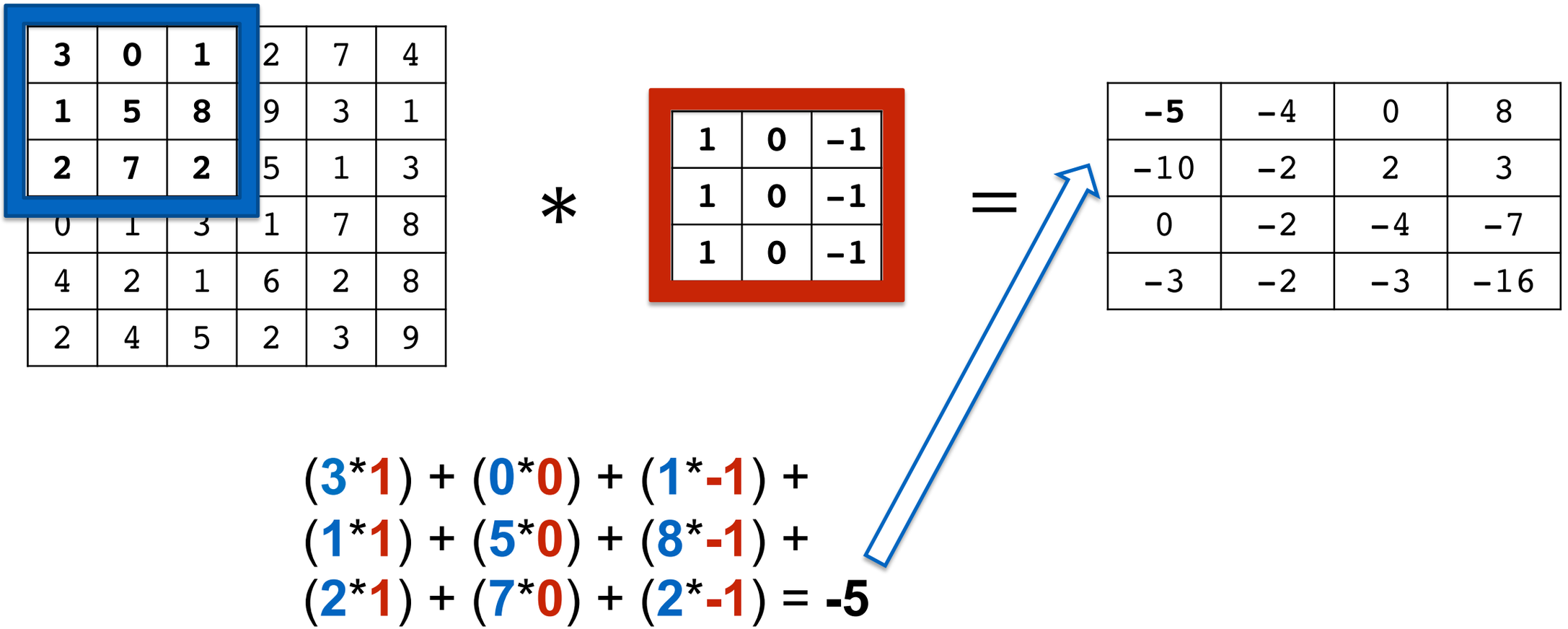
EDGE DETECTION – INTRODUCTION



Pass the **filter** over the **image** and calculate the dot product between the pixels and the filter, storing the result on the right-hand side.

EDGE DETECTION – SQUARE 1



(3*1) + (0*0) + (1*-1) +
(1*1) + (5*0) + (8*-1) +
(2*1) + (7*0) + (2*-1) = **-5**

EDGE DETECTION – SQUARE 2

3	**0**	**1**	**2**	7	4
1	**5**	**8**	**9**	3	1
2	**7**	**2**	**5**	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

1	**0**	**-1**
1	**0**	**-1**
1	**0**	**-1**

=

-5	**-4**	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

(0*1) + (1*0) + (2*-1) +
(5*1) + (8*0) + (9*-1) +
(7*1) + (2*0) + (5*-1) = **-4**

EDGE DETECTION – YOU TRY!

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

Note how this result picks up where the vertical edge exists in the image.

EDGE DETECTION – CHANGE THE FILTER

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

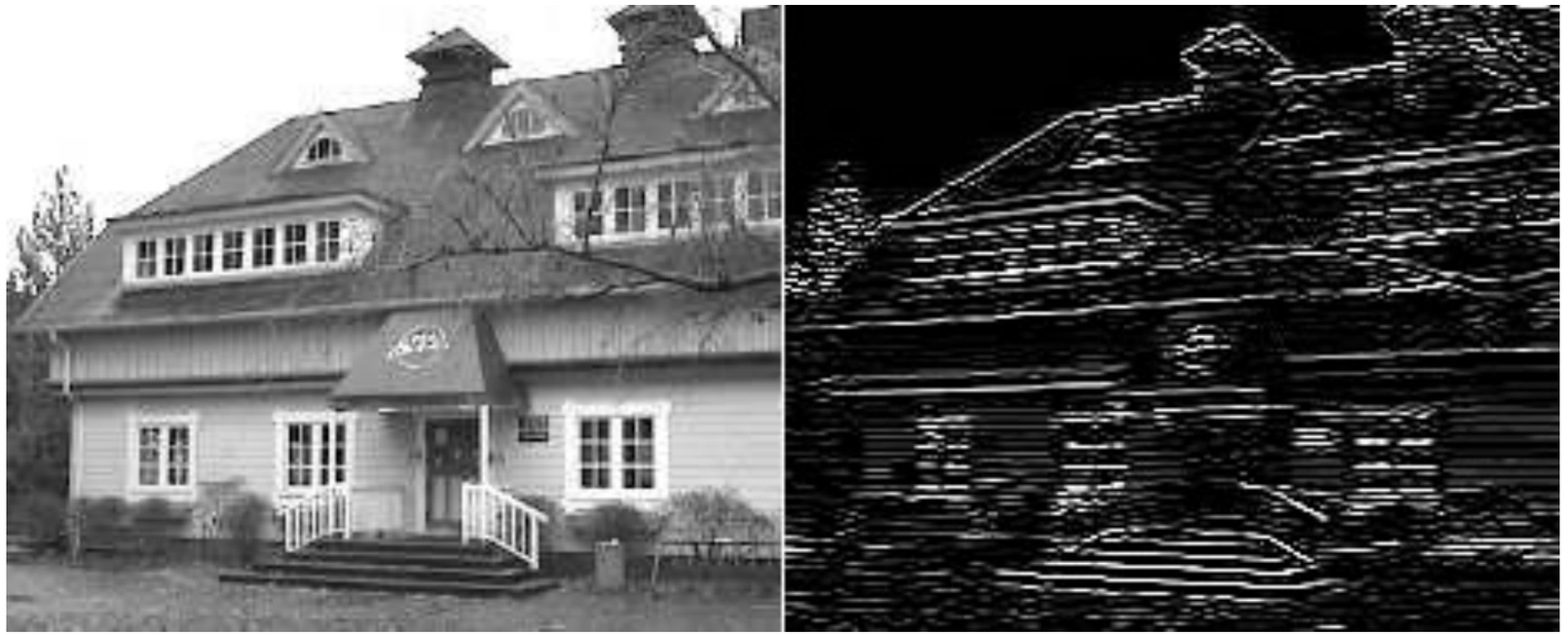
1	1	1
0	0	0
-1	-1	-1

=

Note how this result picks up where the horizontal edge exists in the image.

CONVOLUTIONAL FILTERS

- The two filters we used were geared toward detecting horizontal and vertical lines.



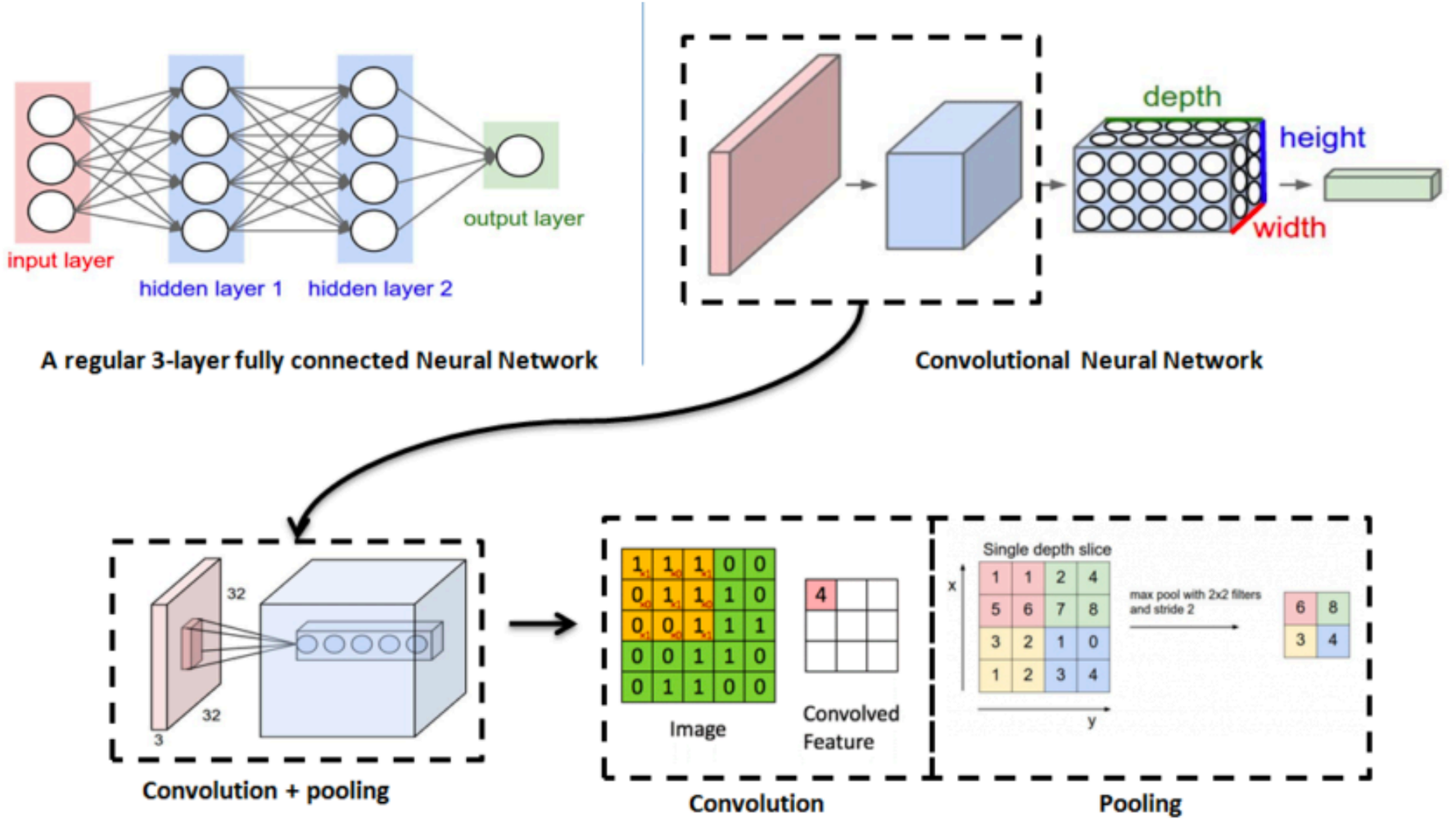
- Source: http://aishack.in/tutorials/image-convolution-examples/

CONVOLUTIONAL FILTERS

- Generally, we don’t specify what the filter will look like. Instead, we instantiate filters with random values inserted, then update these random values through the model fitting process!
  - Specifically, these are updated during backpropagation.
  - These are parameters that our model has to learn.

w1	w2	w3
w4	w5	w6
w7	w8	w9

CNN VISUAL



Source: http://ufldl.stanford.edu/tutorial/images/Convolution_schematic.gif, http://cs231n.github.io/assets/nn1/neural_net2.jpeg, http://cs231n.github.io/assets/cnn/depthcol.jpeg, http://cs231n.github.io/assets/cnn/maxpool.jpeg