CMPE 258, Deep Learning

Convolutional layer

March 20, 2018

DMH 149A

Taehee Jeong

Ph.D., Data Scientist

SJSU SAN JOSÉ STATE UNIVERSITY

Assigiment_4

Deep Neural Network with tensorflow
Due day is Sunday, Marth 18th.
There is penalty for late submission or re-submission after due day.

In Assignment_3 and mid-term exam, we used pandas and numpy.

For Assignment_4, we will use Tensorflow.

Please build Deep Neural Network with two hidden layers.

For activation function, please use relu or elu for hidden layers, sigmoid for output layer.

For weight initialization, please use xaiver initialization.

For optimization, please use adam optimization.

For regularization, please use dropout.

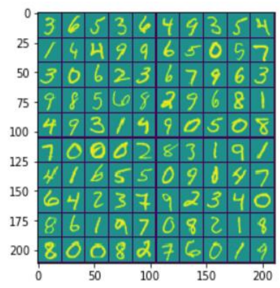
As the final output, please plot train accuracy and test accuracy with probability (0.1 ~ 0.9) of dropout.

Today's lesson

- Compare CNN vs. DNN
- 2D image filter
- Convolution calculation
- Convolution on RGB images
- Multiple filters
- Size of matrix in convolution layers

Digital representation of an image

- Grayscale image is a matrix of pixels (**pic**ture **el**ements)
- Dimensions of this matrix are called image resolution (e.g. 20 x 20)
- Each pixel stores its brightness (or **intensity**) ranging from 0 to 255, 0 intensity corresponds to black color
- Color images store pixel intensities for 3 channels: red, green and blue



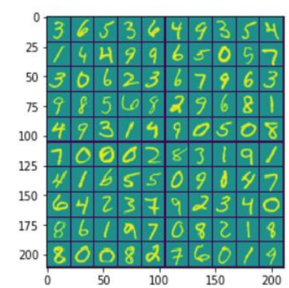
	0	1	2	3	4	5	6	7	8	9	...	391	392	393	394	395	396	397	398	399	y
0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	5
1	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	9
2	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	7
3	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	6
4	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	5

<Intro to deep learning, national research university>

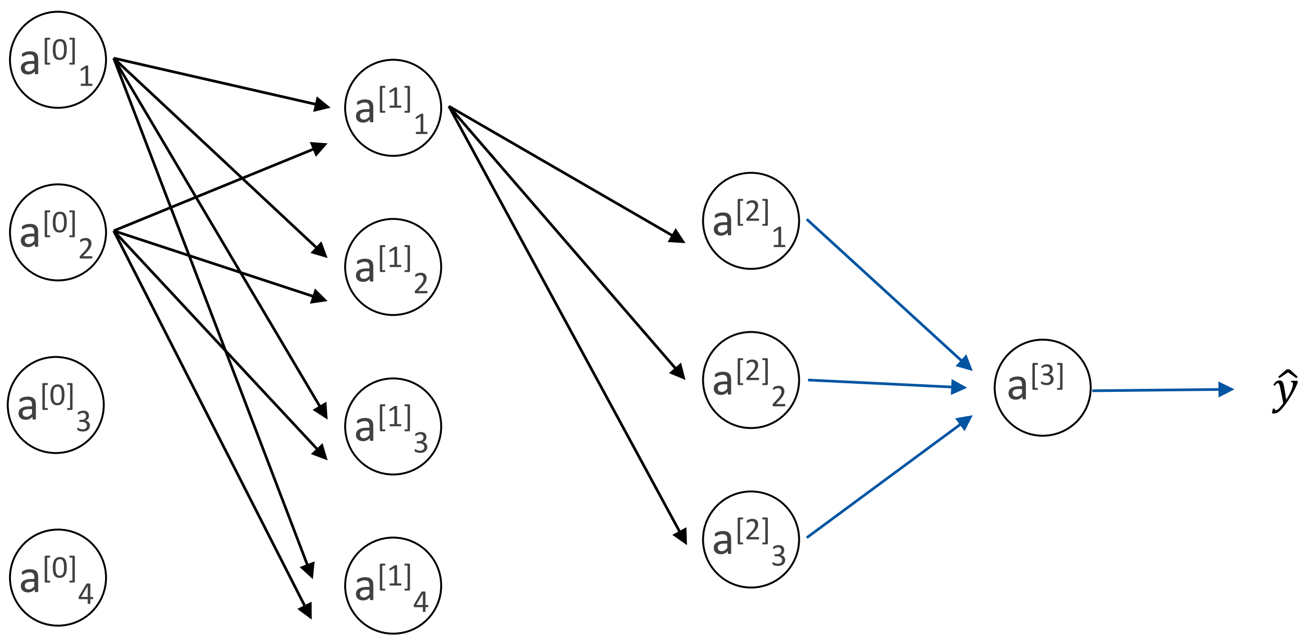
Normalize input pixels

$$x_{norm} = \frac{x}{255} - 0.5$$

	0	1	2	3	4	5	6	7	8	9	...	391	392	393	394	395	396	397	398	399	y
0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	5
1	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	9
2	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	7
3	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	6
4	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	5



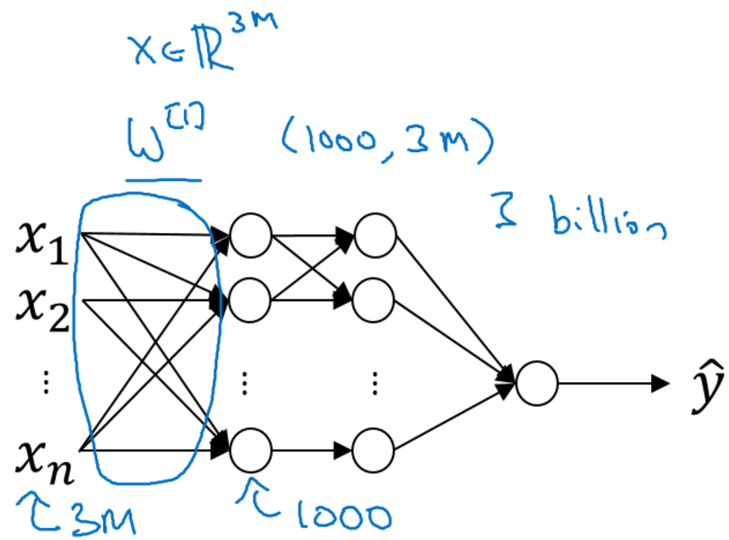
Why not Deep Neural Network?



Why not Deep Neural Network?

CNNs solve this problem using partially connected layers.

$1000 \times 1000 \times 3$
$= 3$ million

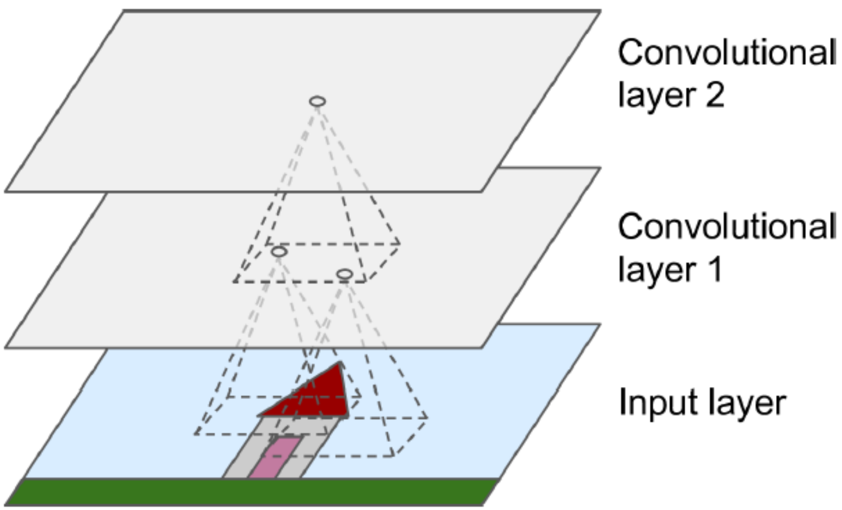


<Deep Neural Network, Andrew Ng>

Convolutional layer

Inspired by the architecture of visual cortex by David H. Hubel and Torsten wiesel

Hierarchical structure



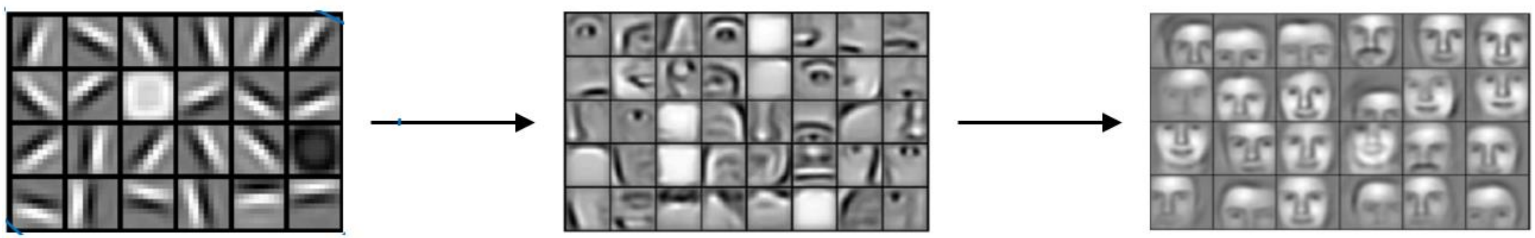
higher-level features in larger areas

low-level features in small areas

<Hands-On ML, A. Geron>

Convolutional layer

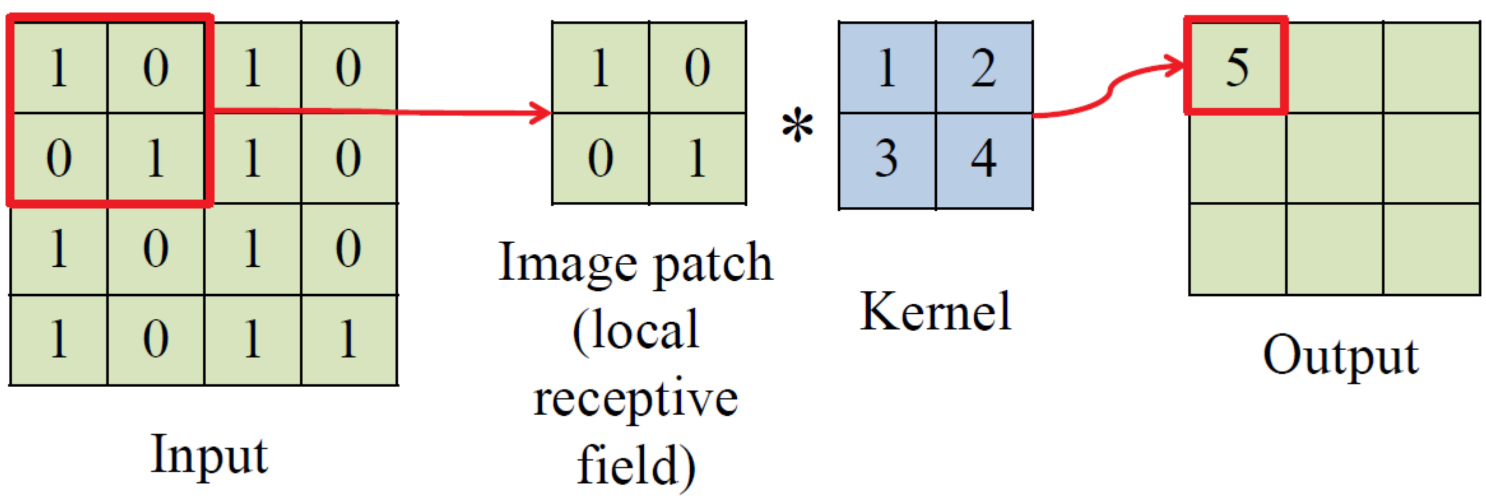
low-level features　　high-level features　　higher-level features



<Deep Neural Network, Andrew Ng>

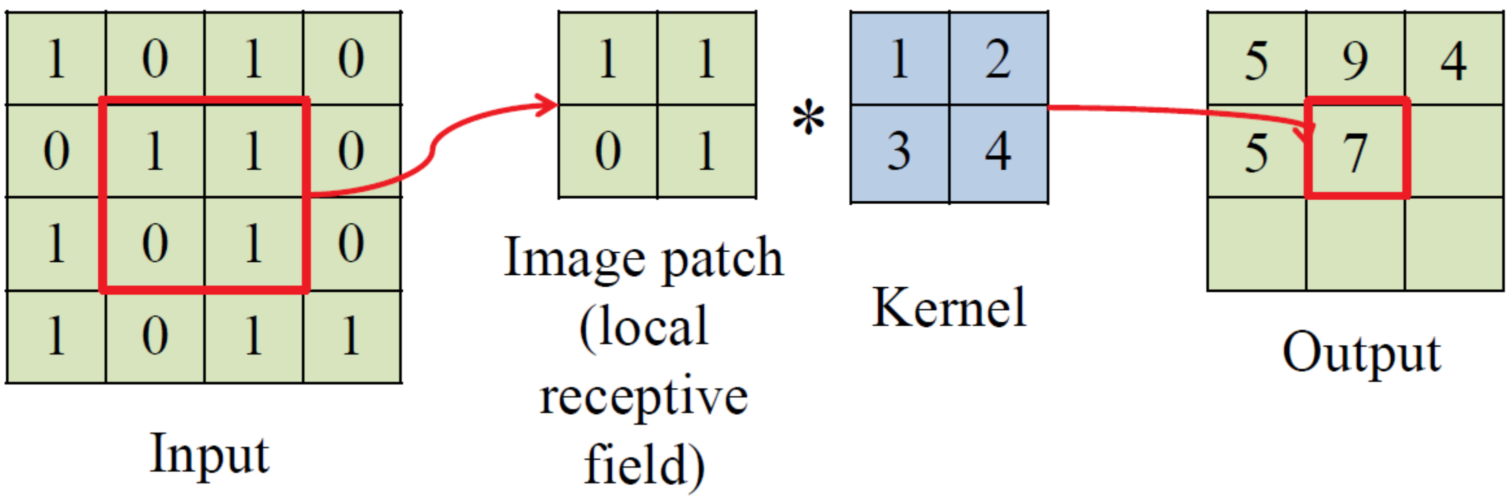
Convolution

Convolution is a dot product of a **kernel** (or filter)
and a patch of an image (**local receptive field**) of the same size



<Intro to deep learning, national research university>

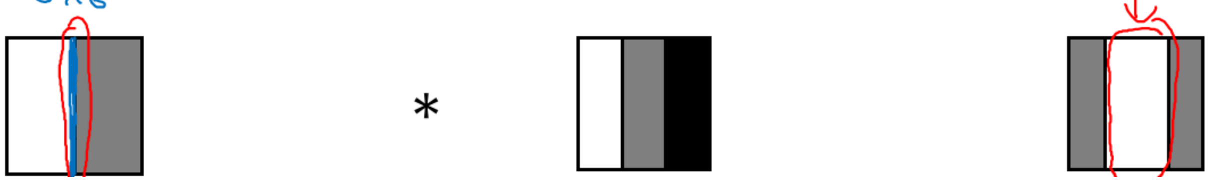
Convolution



<Intro to deep learning, national research university>

Vertical edge detection

$$
\underbrace{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}}_{6 \times 6} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{3 \times 3} = \underbrace{\begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}_{4 \times 4}
$$



<Deep Neural Network, Andrew Ng>

Vertical edge detection

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$

<Deep Neural Network, Andrew Ng>

Horizontal edge detection

$$
\begin{bmatrix}
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
0 & 0 & 0 & 10 & 10 & 10 \\
0 & 0 & 0 & 10 & 10 & 10 \\
0 & 0 & 0 & 10 & 10 & 10
\end{bmatrix}
*
\begin{bmatrix}
1 & 1 & 1 \\
0 & 0 & 0 \\
-1 & -1 & -1
\end{bmatrix}
=
\begin{bmatrix}
0 & 0 & 0 & 0 \\
30 & 10 & -10 & -30 \\
30 & 10 & -10 & -30 \\
0 & 0 & 0 & 0
\end{bmatrix}
$$

6 x 6

<Deep Neural Network, Andrew Ng>

SJSU SAN JOSÉ STATE UNIVERSITY

Edge detection

Kernel

-1	-1	-1
-1	8	-1
-1	-1	-1

Original image

$*$ Kernel $=$

Edge detection

Sums up to 0 (black color) when the patch is a solid fill

<Intro to deep learning, national research university>

Sharpening

Original image

Kernel

$*$

-1	-1	-1
-1	8	-1
-1	-1	-1

$=$

Edge detection

$*$

0	-1	0
-1	5	-1
0	-1	0

$=$

Sharpening

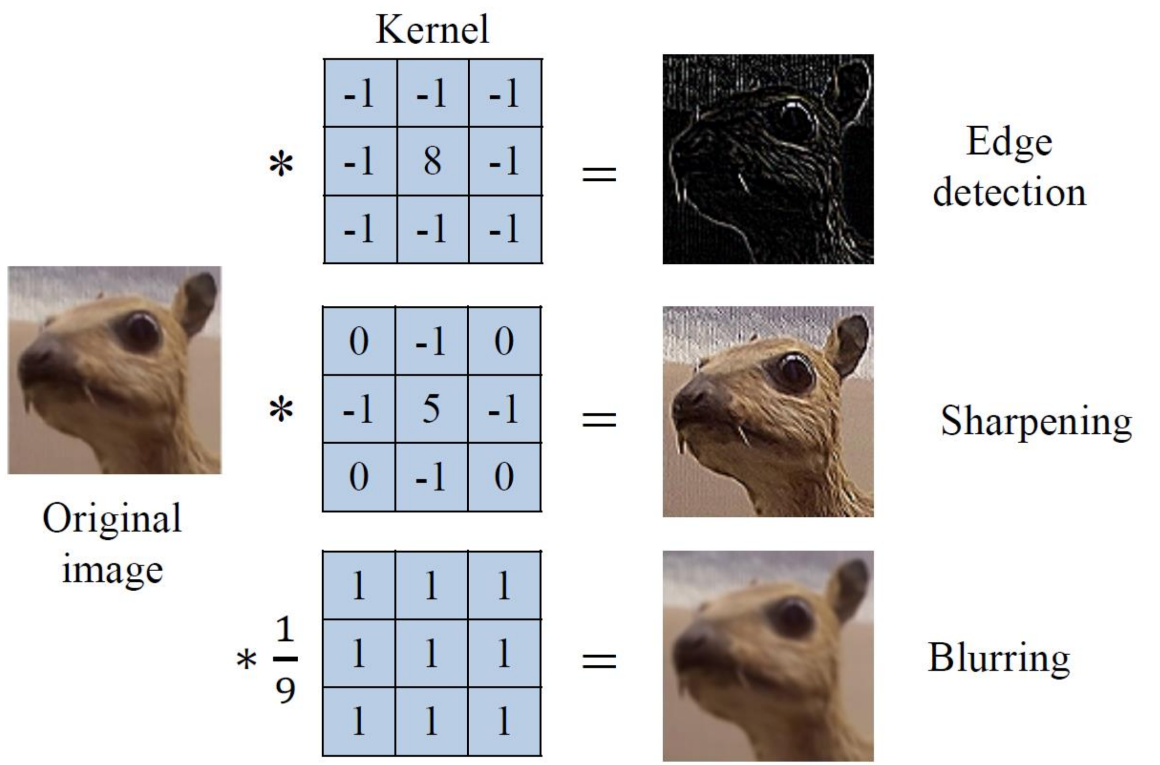
Doesn’t change an image for solid fills

Adds a little intensity on the edges

<Intro to deep learning, national research university>

SJSU SAN JOSÉ STATE UNIVERSITY

Blurring



<Intro to deep learning, national research university>

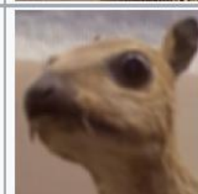
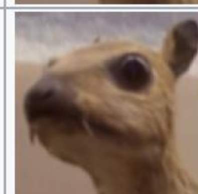
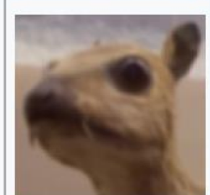
SJSU SAN JOSÉ STATE UNIVERSITY

Kernel (image processing)

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

<Wikipedia>

Kernel (image processing)

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

<Wikipedia>

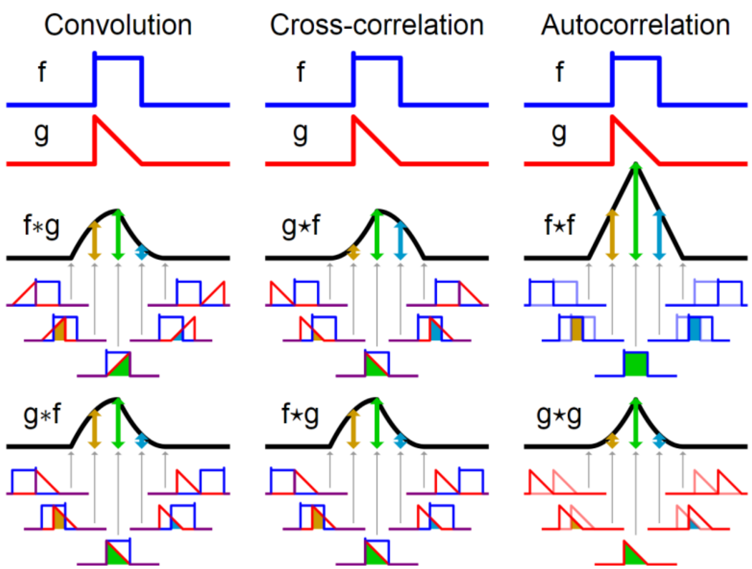
Advantages of a CNN over a fully connected DNN for image classification

- CNN has fewer parameters than a fully connected DNN
- CNN can detect a particular feature anywhere on the image instead of detecting only in that particular location.
- CNN’s architecture is able to identify pixels’ organization (how each pixel is located).

<Hands-On ML, A. Geron>

Convolution

Convolutional layers actually use cross-correlation. <Hands-On ML, A. Geron>

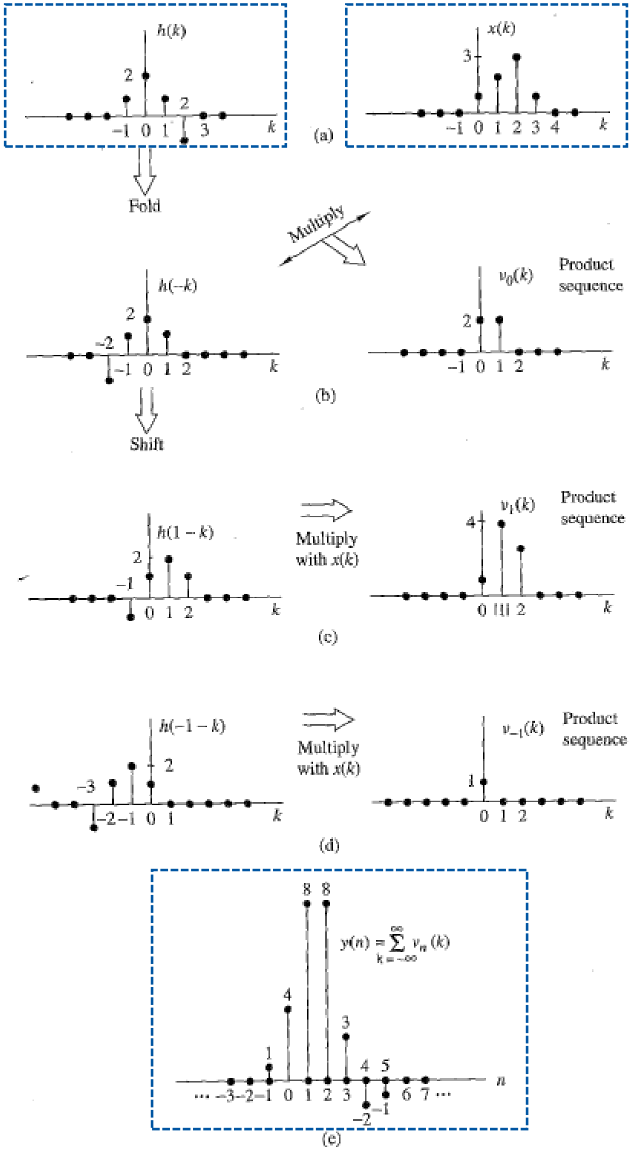


<Wikipedia>

Convolution

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

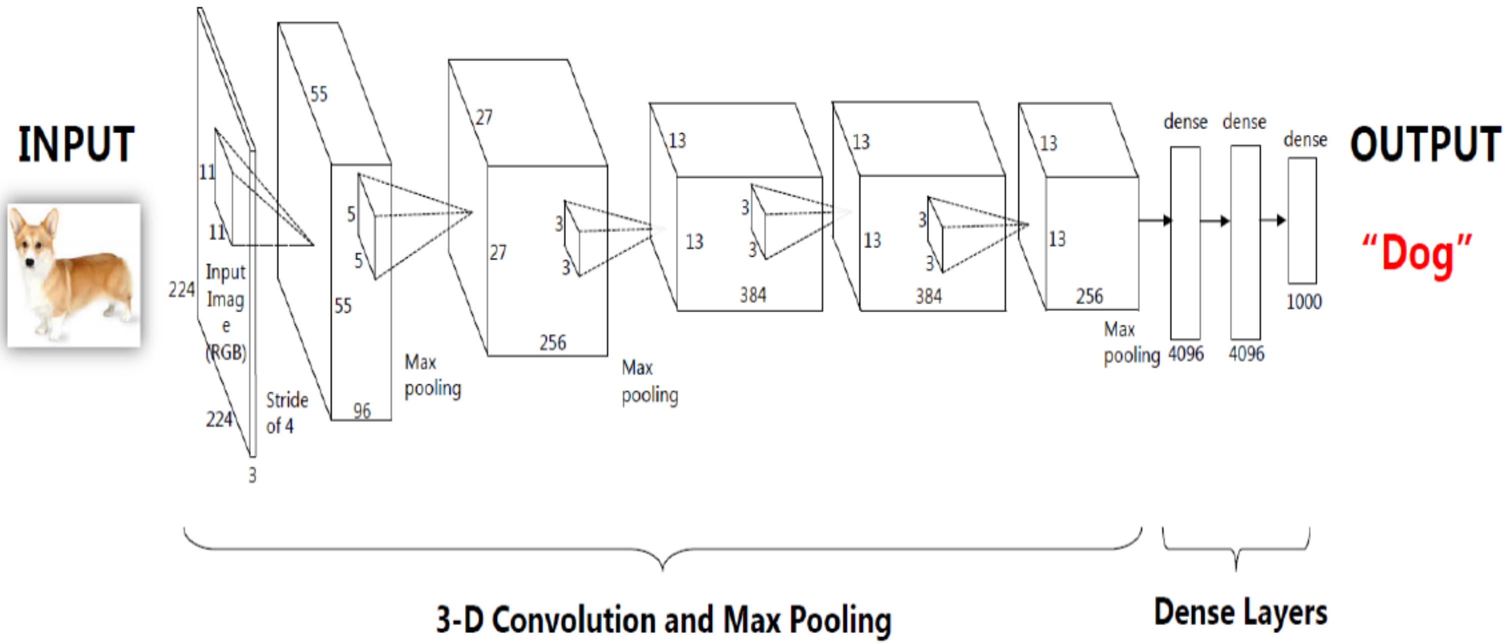
<Digital signal processing, J.G., Proakis et al.>



Convolution is a mathematical operation that slides one function over another and measures the integral of their pointwise multiplication. It is heavily used in signal processing.

<Hands-On ML, A. Geron>

Image classification using Convolution Neural Network



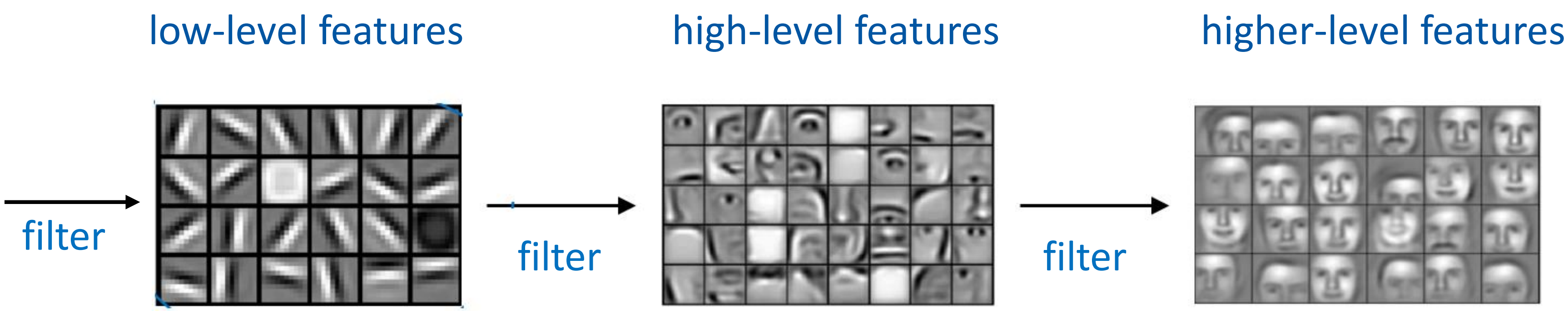
<Ralph Wittig, Power-Efficient Machine Learning using FPGAs on Power systems, OpenPOWERSummit>

Types of layer in a convolutional network

- Convolution
- Pooling
- Fully connected

Convolutional layer

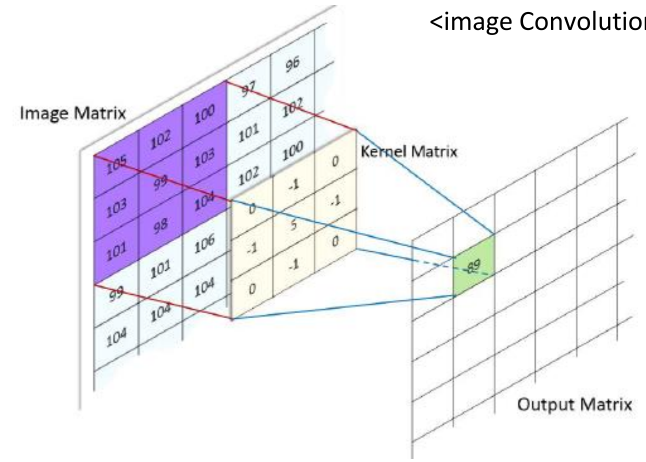
What kind of filters should we use?



<Deep Neural Network, Andrew Ng>

Convolution

<image Convolution>



Input matrix

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

Size: 4 x 4

convolution

*

filter

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

Size: 3 x 3

Output matrix

What is the size?

Convolution

Input matrix

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

Size: 4 x 4

convolution

*

filter

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

Size: 3 x 3

=

Output matrix

Z_1	Z_2
Z_3	Z_4

Size: 2 x 2

Convolution

Input matrix

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

Size: 4 x 4

convolution

*

filter

W_1	W_2
W_3	W_4

Size: 2 x 2

Output matrix

What is the size?

Convolution

Input matrix

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

Size: 4 x 4

convolution

*

filter

W_1	W_2
W_3	W_4

Size: 2 x 2

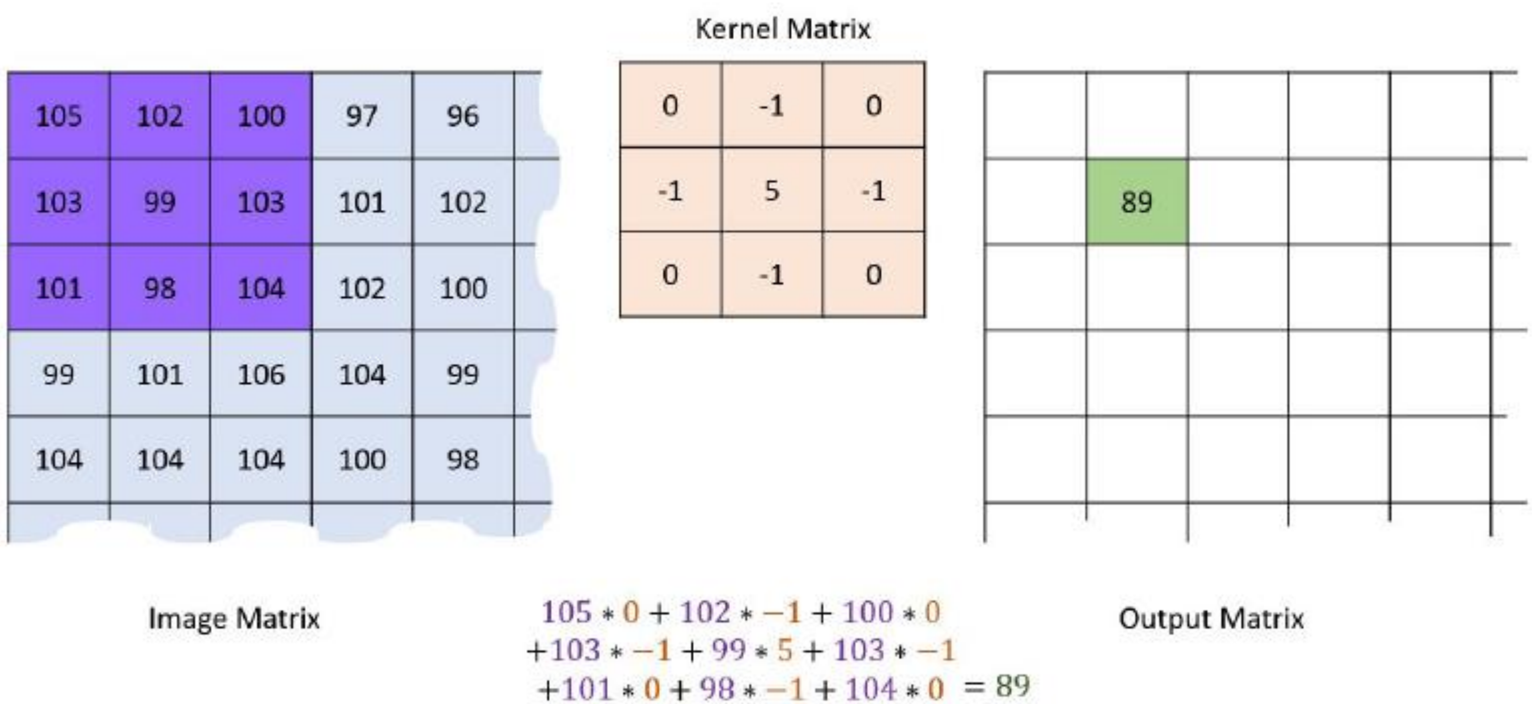
=

Output matrix

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

Size: 3 x 3

Convolution calculation



<image Convolution>
Machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Convolution calculation

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

105	102	100	97	96
103	99	103	101	102
101	98	104	102	100
99	101	106	104	99
104	104	104	100	98

Image Matrix

	89	111		

Output Matrix

$$102*0+100*-1+97*0$$
$$+99*-1+103*5+101*-1$$
$$+98*0+104*-1+102*0=111$$

<image Convolution>
Machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

SJSU SAN JOSÉ STATE UNIVERSITY

Convolution calculation:pre-activation

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

*

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

=

Z_1	Z_2
Z_3	Z_4

$$Z_1 = X_1 \times W_1 + X_2 \times W_2 + X_3 \times W_3 \\ +X_5 \times W_4 + X_6 \times W_5 + X_7 \times W_6 \\ +X_9 \times W_7 + X_{10} \times W_8 + X_{11} \times W_9$$

$$Z_2 = X_2 \times W_1 + X_3 \times W_2 + X_4 \times W_3 \\ +X_6 \times W_4 + X_7 \times W_5 + X_8 \times W_6 \\ +X_{10} \times W_7 + X_{11} \times W_8 + X_{12} \times W_9$$

$$Z_3 = X_5 \times W_4 + X_6 \times W_5 + X_7 \times W_6 \\ +X_9 \times W_7 + X_{10} \times W_8 + X_{11} \times W_9 \\ +X_{13} \times W_{14} + X_{10} \times W_8 + X_{15} \times W_9$$

$$Z_4 = X_6 \times W_4 + X_7 \times W_5 + X_8 \times W_6 \\ +X_{10} \times W_7 + X_{11} \times W_8 + X_{12} \times W_9 \\ +X_{14} \times W_7 + X_{15} \times W_8 + X_{16} \times W_9$$

Convolution: activation

Input matrix

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

Size: 4 x 4

convolution
*

filter

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

Size: 3 x 3

g : activation function
b : bias

=

$g(Z_1 + b)$	$g(Z_2 + b)$
$g(Z_3 + b)$	$g(Z_4 + b)$

=

Output matrix

a_1	a_2
a_3	a_4

Size: 2 x 2

© Taehee Jeong

Edge pixels?

Input matrix

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

convolution

*

filter

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

What about information from edge pixels?

Padding

0	0	0	0	0	0
0	X_1	X_2	X_3	X_4	0
0	X_5	X_6	X_7	X_8	0
0	X_9	X_{10}	X_{11}	X_{12}	0
0	X_{13}	X_{14}	X_{15}	X_{16}	0
0	0	0	0	0	0

*

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

=

Z_1	Z_2	Z_3	Z_4
Z_5	Z_6	Z_7	Z_8
Z_9	Z_{10}	Z_{11}	Z_{12}
Z_{13}	Z_{14}	Z_{15}	Z_{16}

Convolution calculation on borders

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

210	89	111		

Output Matrix

$$
\begin{aligned}
&0 * 0 + 105 * -1 + 102 * 0 \\
&+0 * -1 + 103 * 5 + 99 * -1 \\
&+0 * 0 + 101 * -1 + 98 * 0 \quad = 210
\end{aligned}
$$

<image Convolution>
Machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Convolution calculation on borders

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

320				
210	89	111		

Output Matrix

$$\begin{aligned} & 0 * 0 + 0 * -1 + 0 * 0 \\ & +0 * -1 + 105 * 5 + 102 * -1 \\ & +0 * 0 + 103 * -1 + 99 * 0 \quad = 320 \end{aligned}$$

<image Convolution>
Machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Padding: 1 zero padding

0	0	0	0	0	0
0	X_1	X_2	X_3	X_4	0
0	X_5	X_6	X_7	X_8	0
0	X_9	X_{10}	X_{11}	X_{12}	0
0	X_{13}	X_{14}	X_{15}	X_{16}	0
0	0	0	0	0	0

*

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

=

Z_1	Z_2	Z_3	Z_4
Z_5	Z_6	Z_7	Z_8
Z_9	Z_{10}	Z_{11}	Z_{12}
Z_{13}	Z_{14}	Z_{15}	Z_{16}

$$
\begin{aligned}
Z_1 &= 0 \times W_1 + 0 \times W_2 + 0 \times W_3 \\
&\quad + 0 \times W_4 + X_1 \times W_5 + X_2 \times W_6 \\
&\quad + 0 \times W_7 + X_5 \times W_8 + X_6 \times W_9
\end{aligned}
$$

$$
\begin{aligned}
Z_4 &= 0 \times W_1 + 0 \times W_2 + 0 \times W_3 \\
&\quad + X_3 \times W_4 + X_4 \times W_5 + 0 \times W_6 \\
&\quad + X_7 \times W_7 + X_8 \times W_8 + 0 \times W_9
\end{aligned}
$$

Padding: 2 zero padding

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	X_1	X_2	X_3	X_4	0	0
0	0	X_5	X_6	X_7	X_8	0	0
0	0	X_9	X_{10}	X_{11}	X_{12}	0	0
0	0	X_{13}	X_{14}	X_{15}	X_{16}	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

*

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

Stride

Input matrix

X_1	X_2	X_3	X_4	X_5
X_6	X_7	X_8	X_9	X_{10}
X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}

*

Stride 2

filter

W_1	W_2
W_3	W_4

=

Output matrix

Z_1	Z_2
Z_3	Z_4

Size: 2 x 2

Convolution calculation with stride 1

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

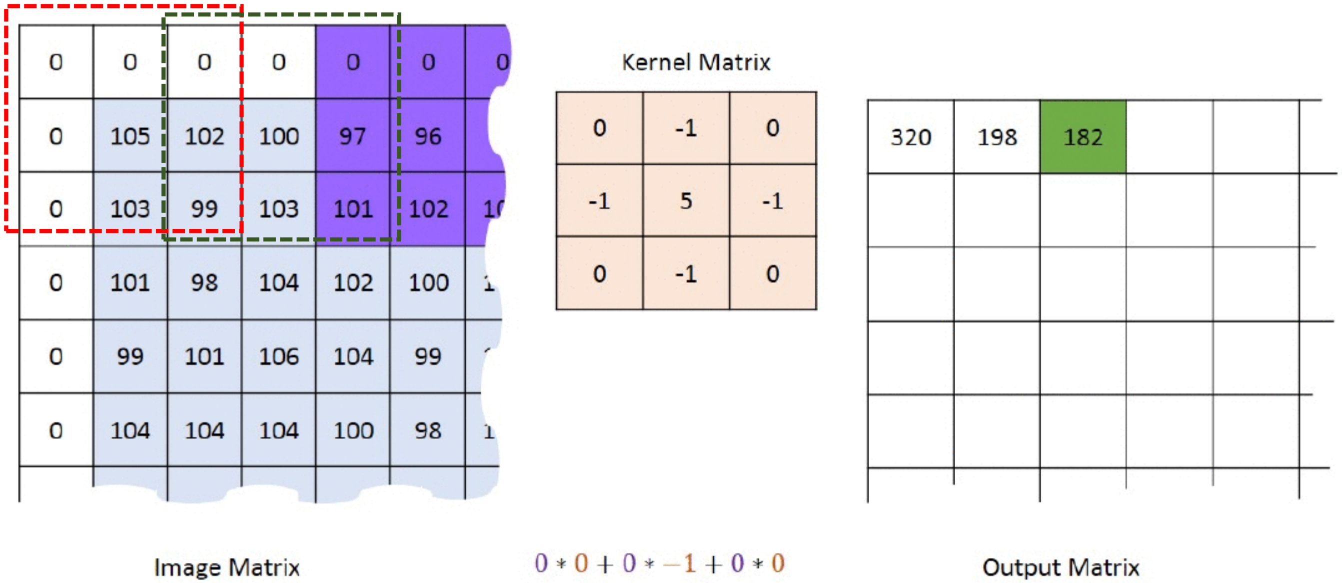
320	206	198	188	
210	89	111		

Output Matrix

$$102 * 0 + 100 * -1 + 97 * 0 \\ +99 * -1 + 103 * 5 + 101 * -1 \\ +98 * 0 + 104 * -1 + 102 * 0 = 111$$

<understanding convolutional layers in convolutional neural networks>
Machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Convolution calculation with stride 2



$$0*0+0*-1+0*0$$
$$+97*-1+96*5+99*-1$$
$$+101*0+102*-1+101*0=182$$

<understanding convolutional layers in convolutional neural networks>
Machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

SJSU SAN JOSÉ STATE UNIVERSITY

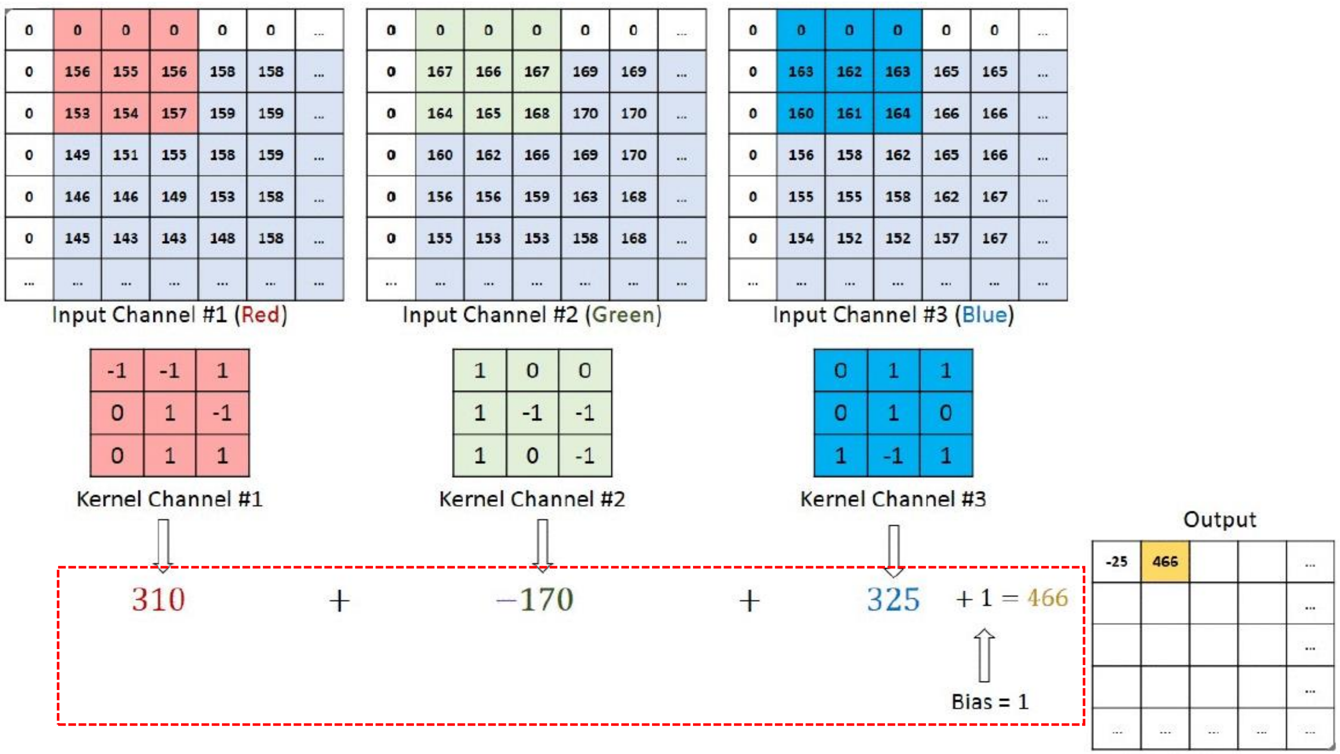
Summary of convolutions

- n x n image
- f x f filter
- padding p
- stride s

Output size:

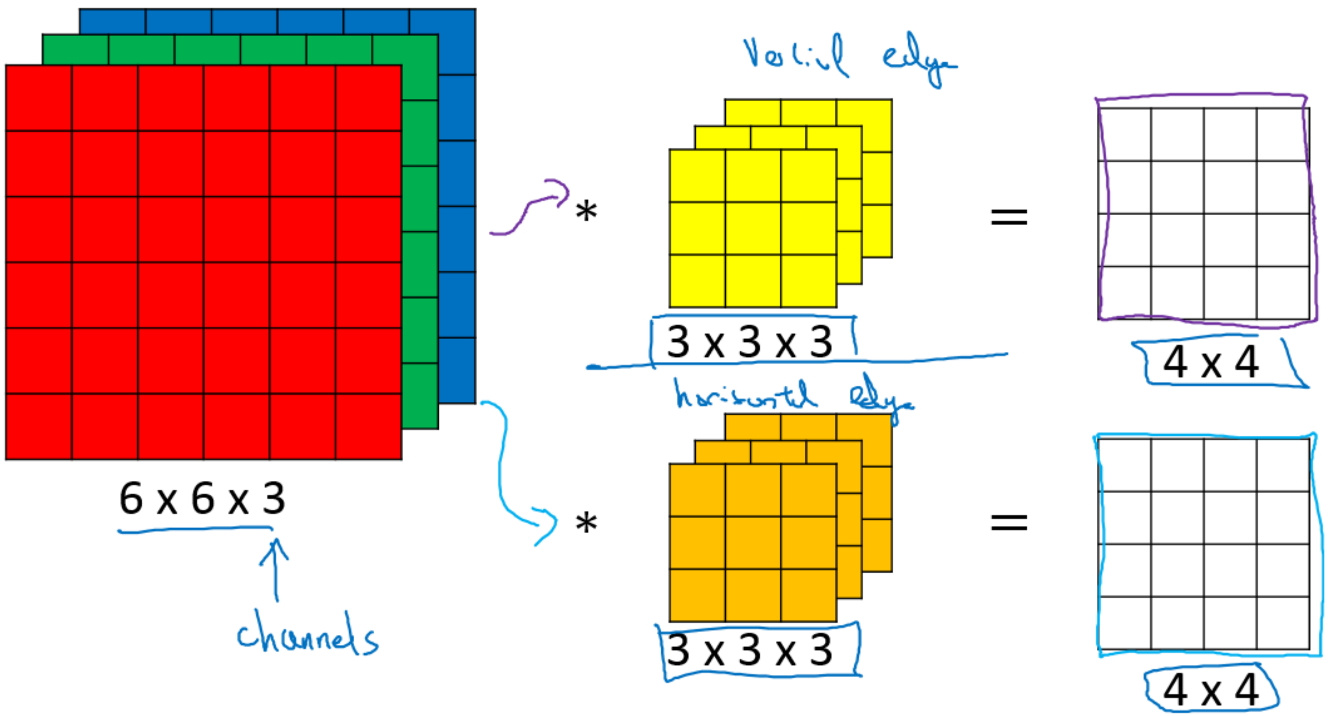
$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

Convolutions on RGB images



<understanding convolutional layers in convolutional neural networks>
Machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Multiple filters (channels)



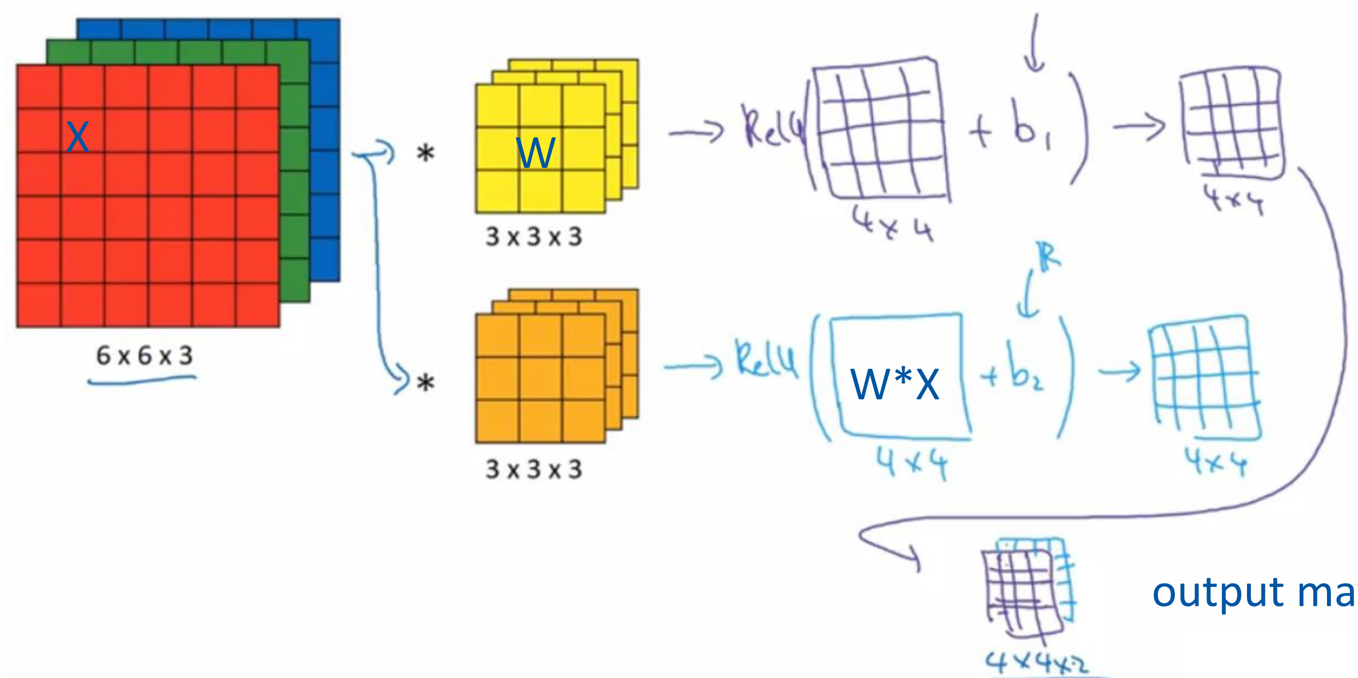
<deep learning, Andrew Ng>

Multiple filters (channels)

Input matrix filters activation

$Z = W*X + b$

$a = ReLU(Z)$



output matrix

<deep learning, Andrew Ng>

Number of parameters in one layer

If there are 10 filters that have the size of 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?

Number of parameters in one layer

If there are 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?

For 1 filter:

3 x 3 x 3 = 27 for W

1 for bias

For 10 filters:

Then, 28 * 10 = 280

Summary of size in matrix

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

Input size: $n_H^{[l-1]} \times n_W^{[l-1]} \times \boxed{n_c^{[l-1]}}$

output size: $n_H^{[l]} \times n_W^{[l]} \times \boxed{n_c^{[l]}}$

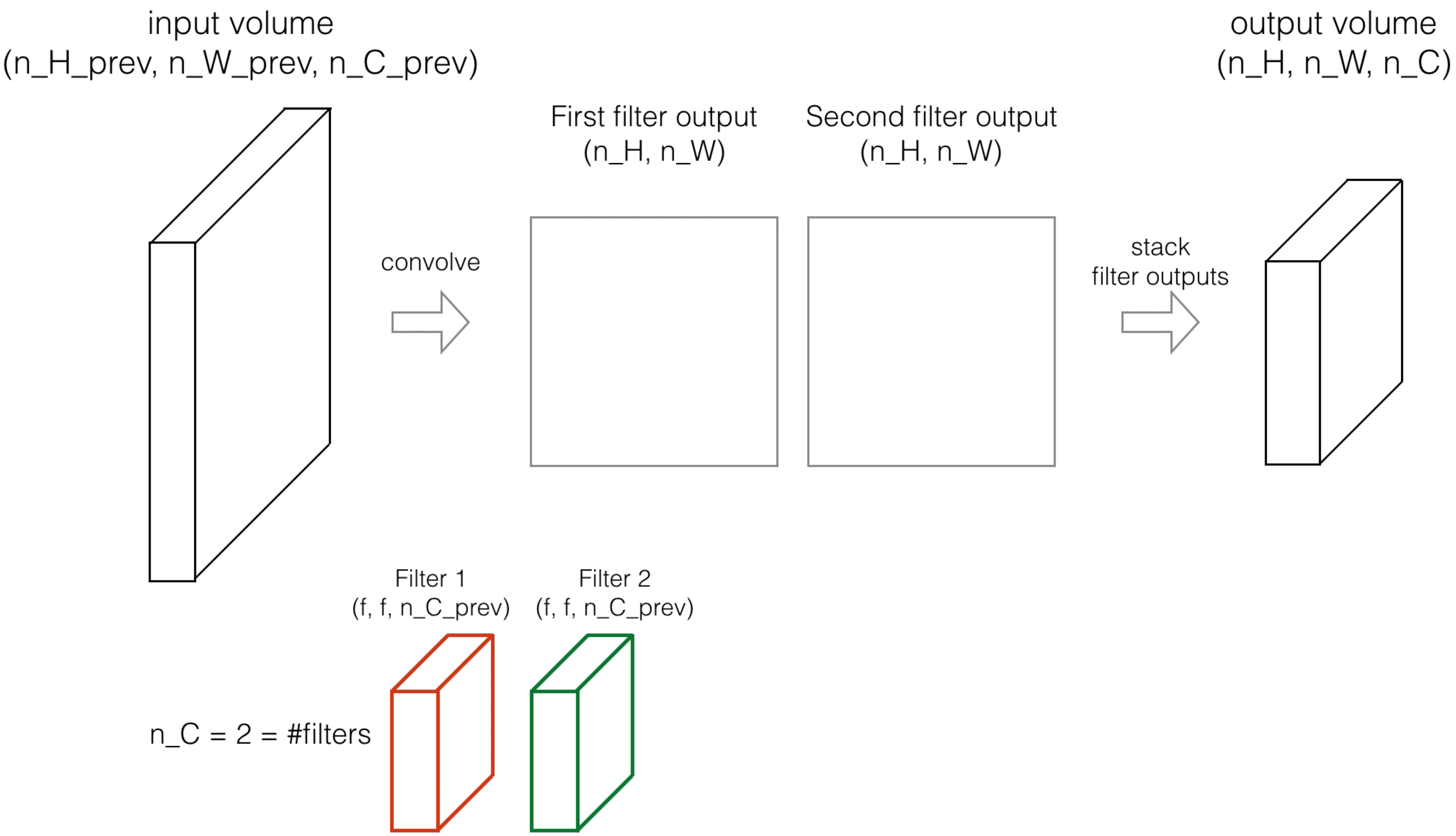
$$n_H^{[l]} = \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$

$$n_W^{[l]} = \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$

Weights: $f^{[l]} \times f^{[l]} \times \boxed{n_c^{[l-1]}} \times \boxed{n_c^{[l]}}$

bias: $1 \times 1 \times 1 \times n_c^{[l]}$

How do convolutions work?



<deep learning, Andrew Ng>

Summary

- Compare CNN vs. DNN
- 2D image filter
- Convolution calculation
- Convolution on RGB images
- Multiple filters
- Size of matrix in convolution layers