Convolutional Neural Networks

Yi-Ting Tsai

National Sun Yat-sen University

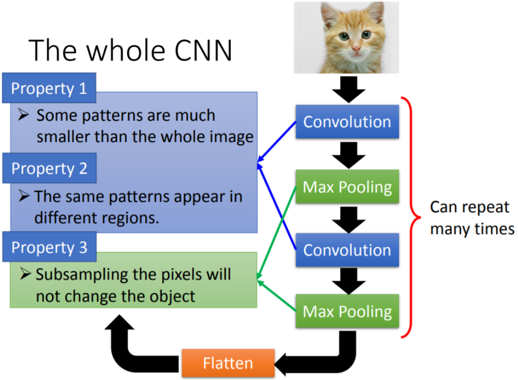
2018/10/24

Outline

Why CNN for Image:

- Some patterns are much smaller than the whole image
- The same patterns appear in different regions.
- Subsampling the pixels will not change the object

Example:


The whole CNN
Property 1
Some patterns are much smaller than the whole image
Property 2
The same patterns appear in different regions.
Property 3
Subsampling the pixels will not change the object
Convolution
Max Pooling
Convolution
Max Pooling
Can repeat many times
Flatten

Example:

CNN – Convolution

Those are the network parameters to be learned.

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1 Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2 Matrix

⋮

Property 1 Each filter detects a small pattern (3 x 3).

Example:

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

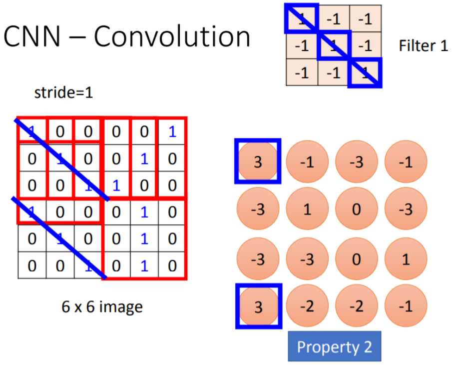
stride=1

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

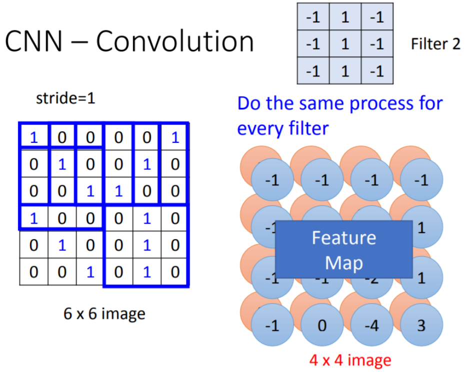
6 x 6 image

3 -1

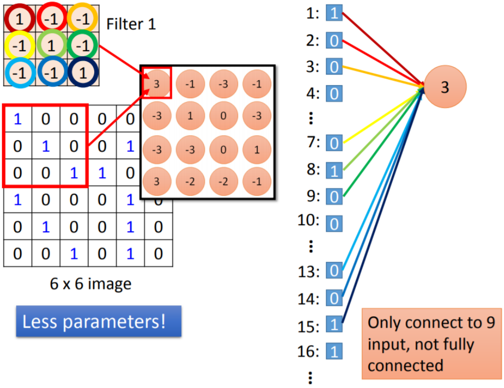
Example:


CNN – Convolution
Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
stride=1
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
Property 2

Example:

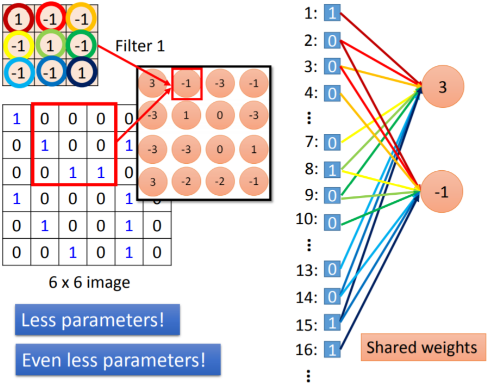

CNN – Convolution
-1 1 -1
-1 1 -1
-1 1 -1
Filter 2
stride=1
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Do the same process for every filter
Feature Map
4 x 4 image

Example:



Filter 1
6 x 6 image
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected

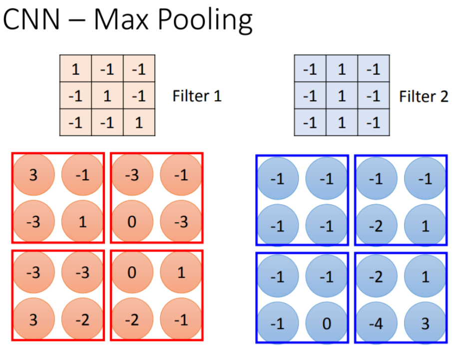
Example:



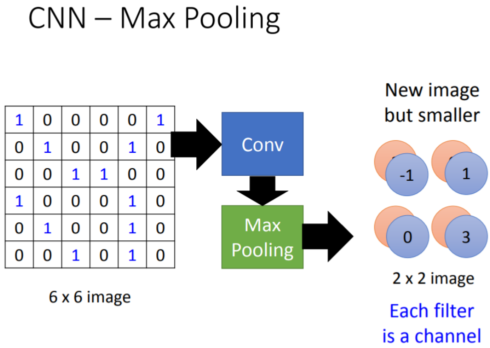
Less parameters!

Even less parameters!

Example:


CNN – Max Pooling
1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
-1 1 -1
-1 1 -1
-1 1 -1
Filter 2
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
-1 -1 -1 -1
-1 -1 -2 1
-1 -1 -2 1
-1 0 -4 3

Example:


CNN – Max Pooling
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Conv
Max Pooling
New image but smaller
-1
1
0
3
2 x 2 image
Each filter is a channel

Pooling Layer:

Their goal is to *subsample* (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters

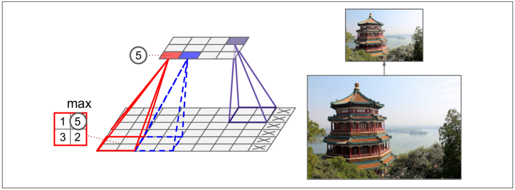


Figure 13-8. Max pooling layer (2 × 2 pooling kernel, stride 2, no padding)

Figure: *Max pooling layer*(2 × 2 pooling kernel, stride 2, no padding)

Typical CNN architectures:

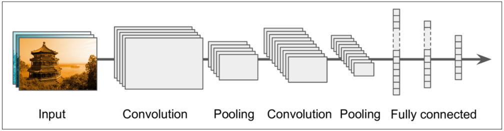


Figure 13-9. Typical CNN architecture

Figure: Typical CNN architecture

CNN Architectures:

A good measure of this progress is the error rate in competitions such as the ILSVRC ImageNet challenge. In this competition the top-5 error rate for image classification fell from over 0.26 to barely over 0.03 in just five years.

We will first look at the classical LeNet-5 architecture (1998), then three of the winners of the ILSVRC challenge: AlexNet (2012), GoogLeNet (2014), ResNet(2015) , and SENet(2017) .

CNN Architectures:

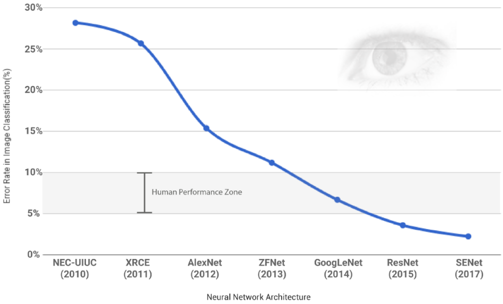


Figure: ILSVRC ImageNet challenge

LeNet-5 architecture:

Table 13-1. LeNet-5 architecture

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

Figure: LeNet-5 architecture

AlexNet architectures:

Table 13-2. AlexNet architecture

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13×13	3×3	1	SAME	ReLU
C6	Convolution	384	13×13	3×3	1	SAME	ReLU
C5	Convolution	384	13×13	3×3	1	SAME	ReLU
S4	Max Pooling	256	13×13	3×3	2	VALID	–
C3	Convolution	256	27×27	5×5	1	SAME	ReLU
S2	Max Pooling	96	27×27	3×3	2	VALID	–
C1	Convolution	96	55×55	11×11	4	SAME	ReLU
In	Input	3 (RGB)	224×224	–	–	–	–

Figure: AlexNet architectures

AlexNet architectures:

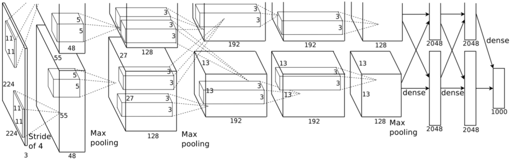


Figure: AlexNet architectures

GoogLeNet architectures:

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

Figure: GoogLeNet architectures

GoogLeNet architectures:

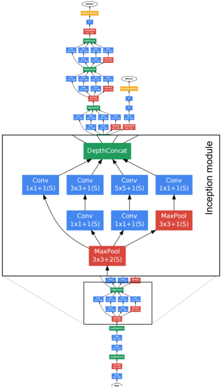


Figure: GoogLeNet architectures

ResNet architectures:

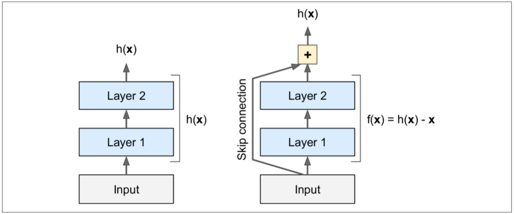


Figure 13-12. Residual learning

Figure: ResNet architectures

ResNet architectures:

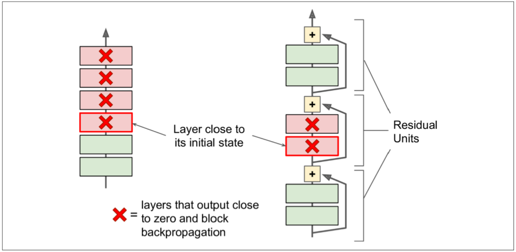


Figure 13-13. Regular deep neural network (left) and deep residual network (right)

Figure: ResNet architectures

ResNet architectures:

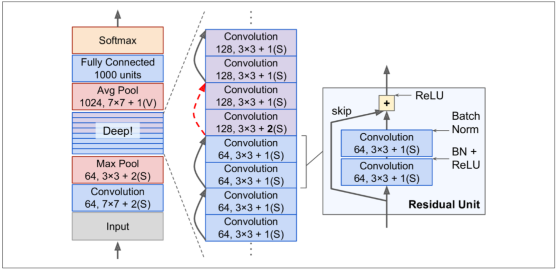


Figure 13-14. ResNet architecture

Figure: ResNet architectures

ResNet architectures:

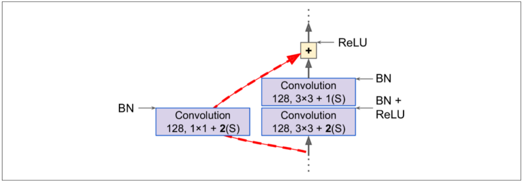


Figure 13-15. Skip connection when changing feature map size and depth

Figure: ResNet architectures

SENet architectures:

Thanks for listening