Convolutional Neural Networks

Yi-Ting Tsai

National Sun Yat-sen University

2018/10/24

Outline

Review-example

CNN Architectures

LeNet-5 architecture

AlexNet architectures

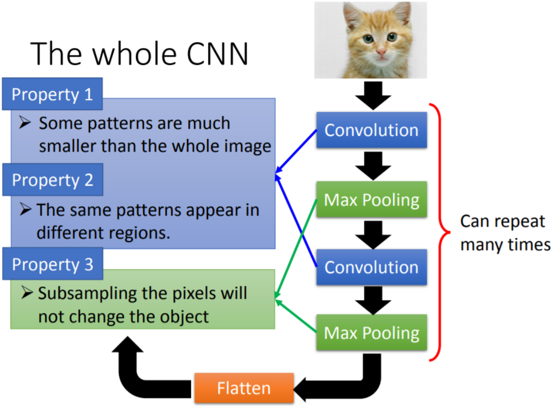
GoogLeNet architectures

ResNet architectures

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

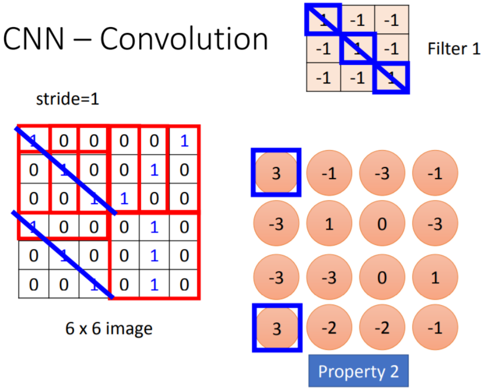
stride=1

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

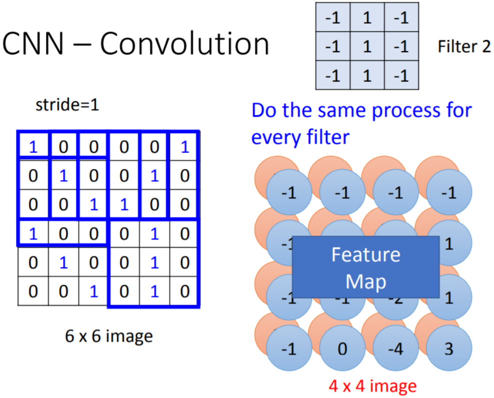
6 x 6 image

3 -1

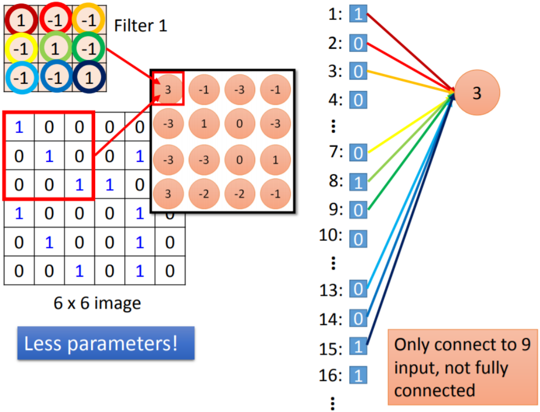
Example:



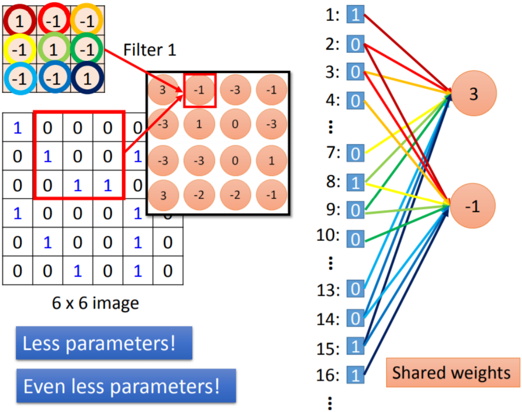
Example:


CNN – Convolution
-1 1 -1
-1 1 -1
-1 1 -1
Filter 2
stride=1
Do the same process for every filter
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 0 -4 3
Feature Map
4 x 4 image

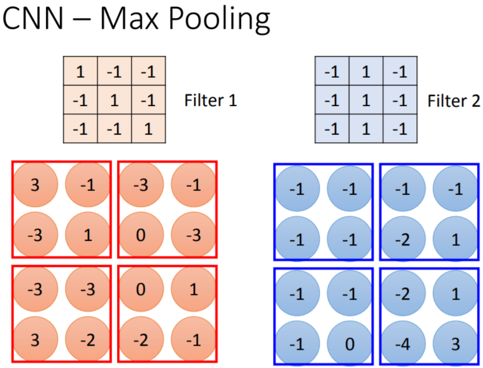
Example:


Filter 1
6 x 6 image
Less parameters!
3
Only connect to 9 input, not fully connected

Example:

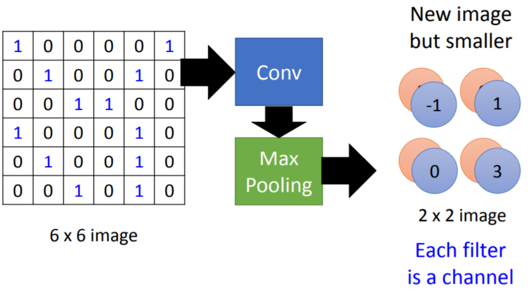

Filter 1
6 x 6 image
Less parameters!
Even less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
-1
Shared weights

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



Correction p359:

The sentence at the bottom of page 361 should be:

Specifically, a neuron located in row i, column j of the feature map k in a given convolutional layer l is connected to the outputs of the neurons in the previous layer l - 1, located in rows i x sh to i x sh + fh - 1 and columns j x sw to j x sw + fw - 1, across all feature maps (in layer l - 1).

Correction p360:

The Equation 13-1 should be (using latexmath):

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} . w_{u,v,k',k} \quad \text{with} \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases} \tag{1}$$

Typical CNN architectures:

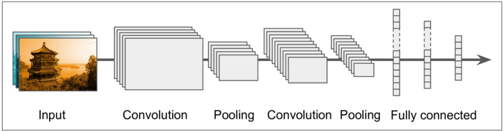


Figure 13-9. Typical CNN architecture

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:

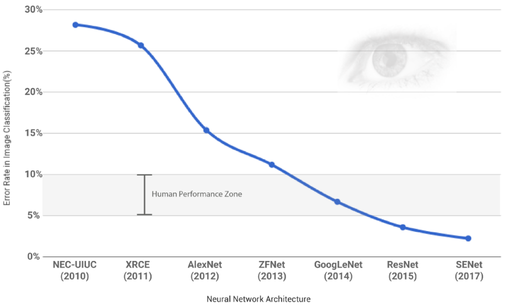


Figure: ILSVRC\CHTseng

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

LeNet-5 architecture:

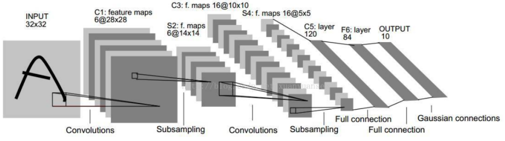


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Figure: LeNet-5 architecture

LeNet-5 architecture:

There are a few extra details to be noted:

- MNIST images are 28×28 pixels, but they are zero-padded to 32×32 pixels and normalized before being fed to the network.
- Most neurons in C3 maps are connected to neurons in only three or four S2 maps (instead of all six S2 maps).

LeNet-5 architecture:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

Figure: LeNet-5 architecture

- The AlexNet CNN architecture 9 won the 2012 ImageNet ILSVRC challenge by a large margin: it achieved 0.17 top-5 error rate while the second best achieved only 26
- It is quite similar to LeNet-5, only much larger and deeper, and it was the first to **stack convolutional layers directly on top of each other**, instead of stacking a pooling layer on top of each convolutional layer.

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

To reduce overfitting, the authors used two regularization techniques we discussed in previous chapters:

- They applied **dropout** during training to the outputs of layers F8 and F9.
- They performed **data augmentation** by randomly shifting the training images by various offsets, flipping them horizontally,and changing the lighting conditions.

AlexNet also uses a competitive normalization step immediately after the ReLU step of layers C1 and C3, called **local response normalization**. Equation 13-2.(Local response normalization):

$$b_i = a_i \left(k + \alpha \sum_{j=j_{low}}^{j_{high}} \alpha_j^2 \right)^{-\beta} \quad with \begin{cases} j_{high} = min\left(i + \frac{r}{2}, f_n - 1\right) \\ j_{low} = max\left(0, i - \frac{r}{2}\right) \end{cases} \tag{2}$$

AlexNet architectures:

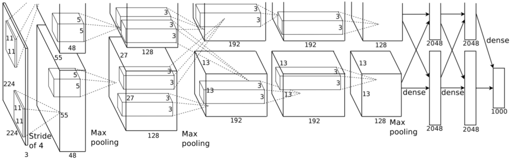


Figure: AlexNet architectures

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:

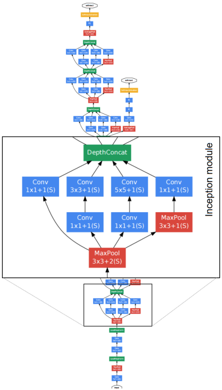


Figure: GoogLeNet architectures

GoogLeNet architectures:

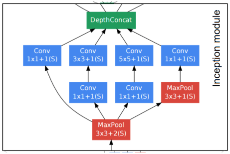


Figure: GoogLeNet architectures

ResNet architectures:

the winner of the ILSVRC 2015 challenge was the Residual Network(or ResNet), developed by Kaiming He et al., which delivered an astounding top-5 error rate under 0.036

The key to being able to train such a deep network is to use **skip connections** (also called **shortcut connections**): the signal feeding into a layer is also added to the output of a layer located a bit higher up the stack. Let' s see why this is useful.

ResNet architectures:

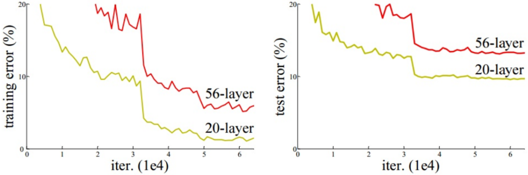


Figure: ResNet architectures

ResNet architectures:

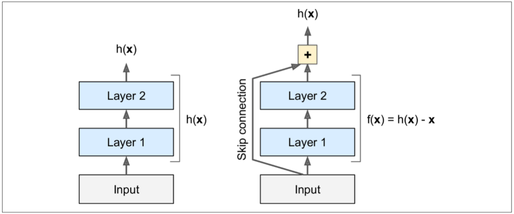


Figure 13-12. Residual learning

Figure: ResNet architectures

ResNet architectures:

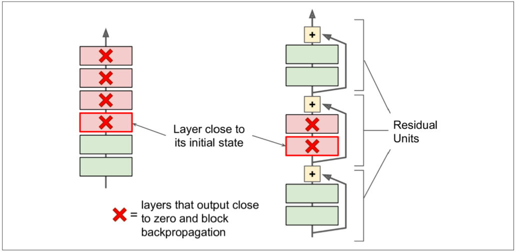


Figure 13-13. Regular deep neural network (left) and deep residual network (right)

Figure: ResNet architectures

ResNet architectures:

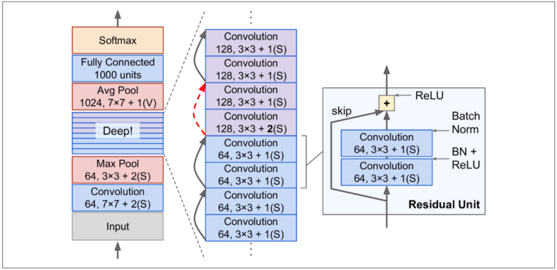


Figure 13-14. ResNet architecture

Figure: ResNet architectures

Thanks for listening