

https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 8: Convolutional Neural Networks

Shen Shen October 25, 2024

Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- Case studies





1. Why do we need a special network for images?
2. Why is CNN (the) special network for images?

Why do we need a special net for images?







784 weights per neuron



784×16 weights 16 biases



Use the same small 2-layer network, need to learn ~3M parameters

Imagine even higher-resolution images (e.g. 1024-1024 already leads to 1-million dimensional as input), or more complex tasks, the number of parameters can just grow very fast.

426-by-426 grayscale image

Q: Why do we need a specialized network?

A: fully-connected nets don't scale well for vision tasks

Recall, more powerful models also has the pitfall of overfitting

Underfitting

Appropriate model

Overfitting



Why do we think



is 9?

Why do we think any of

999999999999999999999

is 9?





13





14







layering is compatible with hierarchical structure

• Visual hierarchy



• Spatial locality



• Translational invariance



- Visual hierarchy
- Spatial locality
- Translational invariance

- layering (with nonlinear activations)
- convolution
- pooling

to handle images efficiently and sensibly.

via

Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- Case studies

typical CNN structure for image classification



Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
 - 1-dimensional and 2-dimensional convolution
 - 3-dimensional *tensors*
- Max pooling
- Case studies

Convolutional layer might sound foreign, but it's very similar to fully connected layer

Layer	Forward pass, do	Backward pass, learn
fully-connected	dot-product	neuron weights
convolutional	convolution	filter (kernels) weights

convolution with filters do these things:











convolution interpretation: template matching



convolution interpretation: "look" locally





convolution interpretation: parameter sharing





convolution interpretation: translational equivariance



hyperparameters

• Zero-padding input



• Stride (e.g. stride of 2)

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

• Filter size (e.g. we saw these two)



2-dimensional convolution

input

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

filter

1

 $\mathbf{2}$

1

0

 $\mathbf{2}$

0

output

12

 $\mathbf{17}$

6

17

19

14

2	12
0	10
2	9

[image edited from vdumoulin]

30	31	2_{2}	1	0
0_2	0_2	10	3	1
30	11	2_{2}	2	3
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

3	30	2_1	1_2	0
0	02	1_2	30	1
3	1_0	2_1	2_{2}	3
2	0	0	2	2
2	0	0	0	1

 $\mathbf{2}$

 2_{1}

 3_{2}

 2_{0}

 $\mathbf{2}$

 1_{2}

 0_{2}

 $\mathbf{2}$

12	12	17
10	17	19
9	6	14

 $\mathbf{14}$

 $\mathbf{12}$

3	3	2_0	1_1	0_2
0	0	1_2	32	10
3	1	2_0	2_1	32
2	0	0	2	2
2	0	0	0	1

 $\mathbf{2}$

 $\mathbf{2}$

 $\mathbf{2}$

	1	0		
)	31	1_2	12	
2	2_{2}	30	10	:
)	2_{1}	2_2	9	
	0	1		

3	2	1	0	
0	1	3	1	
1	2_0	2_1	32	
0	0_2	2_2	2_0	

U.

12	12	17
10	17	19
9	6	14

[in nj

3	3	2	1	0	
00	01	1_2	3	1	
3_2	1_2	2_0	2	3	
2_{0}	01	02	2	2	
2	0	0	0	1	

3	3	2	1	0	
0	0	1	3	1	
30	1	2_{2}	2	3	
2_{2}	02	00	2	2	
2_0	0,	02	0	1	

12	12	17
10	17	19
9	6	14

6	14		2
		-	2
			3
			_

12	12	17	
10	17	19	
9	6	14	

mage edit	ted from	ו vdumo	ulir
-----------	----------	---------	------

stride of 2

input

filter

output

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

0	1	2
2	2	0
0	1	2

12	17
9	14

stride of 2





3	3	2_0	1_1	02	
0	0	1_2	32	1_0	
3	1	2_0	2_1	32	
2	0	0	2	2	
2	0	0	0	1	



3	3	2	1	0	
0	0	1	3	1	
30	11	2_{2}	2	3	
2_{2}	0_2	00	2	2	
2_0	0,	0_2	0	1	



3	3	2	1	0
0	0	1	3	1
3	1	2_0	2_1	32
2	0	02	2_2	2_0
2	0	0	0.	1,



[image edited from vdumoulin]

stride of 2, with padding of size 1

input

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

filter

output

0	1	2	
2	2	0	
0	1	2	

6	17	3
8	17	13
6	4	4

		Г [.]		0 0 0 0 1 0 2
$0_2 3_2 3_0 2 1 0 0$	$\begin{bmatrix} 0 & 3 & 3_2 & 2_2 & 1_0 & 0 \end{bmatrix}$	• ۲ ۱ () 3 3	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
$0_0 0_1 0_2 1 3 1 0 6 17$	3 0 0 0 1 3 1 0 6		0 0	$1 3_0 1_1 0_2$
0 3 1 2 2 3 0 8 17	13 0 3 1 2 2 3 0 8	17 13 I (0 3 1	2 2 3 0
0 2 0 0 2 2 0 6 4			0 2 0	0 2 2 0
		; (0 2 0	0 0 1 0
		! (L .		
		Г [.] , (
		⊢ · ' () 3 3	2 1 0 0
$0_0 0_1 0_2 1 3 1 0 6 17$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		0 0	$1 3_0 1_1 0_2$
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	13 0 3 $1_2 2_2 2_0 3 0$	17 13 I (0 3 1	$\begin{array}{ c c c c c }\hline 2 & 2 & 2 & 3 & 0 \\ \hline \end{array}$
$0_0 2_1 0_2 0 2 2 0 6 4$			0 2 0	0 $2_0 2_1 0_2$
			0 2 0	0 0 1 0
		! (L .		
· · · · · · · · · · · · · · · · · · ·		г. , (
		⊢ · ¦ () 3 3	
0 0 0 1 3 1 0 6 17	³ 10 0 0 1 3 1 01		0 0	
0 3 1 2 2 3 0 8 17		5 17 13 I (0 3 1	2 2 3 0
$0_0 2_1 0_2 0 2 2 0 6 4$			0 2 0	$\begin{array}{ c c c c c }\hline 0 & 2_0 & 2_1 & 0_2 \\ \hline \end{array}$
$0_2 2_2 0_0 0 0 1 0$	$\begin{bmatrix} - & - & - & - \\ 0 & 2 & 0_2 & 0_2 & 0_0 & 1 & 0 \end{bmatrix}$		0 2 0	$\begin{array}{ c c c c c }\hline 0 & 0_2 & 1_2 & 0_0 \\ \hline \end{array}$
$0_0 0_1 0_2 0 0 0 0$	$0 0 0_0 0_1 0_2 0 0$			0 0_0 0_1 0_2

6	17	3
8	17	13
6	4	4

6	17	3
8	17	13
6	4	4

6	17	3
8	17	13
6	4	4

[image edited from vdumoulin]

convolution interpretation:

- Looking locally
- Parameter sharing
- Template matching
- Translational equivariance



[video credit Lena Voita]



Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional tensors
- Max pooling
- Case studies

A tender intro to tensor:



We'd encounter 3d tensor due to: 1. color input





blue



green



red





[Photo by Zayn Shah, Unsplash]





filter 2

We'd encounter 3d tensor due to

1. color input

2. the use of multiple filters





But, we *don't* typically do 3-dimensional convolution. Instead:



- 3d tensor input, depth d
- 3d tensor filter, depth d
- 2d convolution, 2d output



We *don't* typically do 3-dimensional convolution, because







input tensor

2d output







- 3d tensor input, depth *d*
- 3d tensor filter, depth *d*
- 2d tensor (matrix) output







multiple output matrices

		•••
		•••
•••	•••	•••





We'd encounter 3d tensor due to:

1. color input



2. the use of multiple filters -- in doing 2-dimensional convolution



Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- Case studies

Ĩ	Ĩ	Ĩ	Â	Ĩ

1-dimensional pooling



filter weights are the learnable parameter

no learnable parameter

2-dimensional max pooling (example)

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3	3
3	3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

3	3
3	3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



[image edited from vdumoulin]

2-dimensional max pooling (example)

- can choose filter size
- typically choose to have no padding
- typically a stride >1
- reduces spatial dimension

*** ***	
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	
80 80 80 80 80 80 80 80 82 86 89 10 10 10 10 10 10 10 10 80 80 80 80 80 80	
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.7 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0	0.0
80 80 80 80 80 80 80 80 80 84 10 10 9 85 85 85 85 97 10 10 10 82 88 88 88 88	
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 0.1	
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 0.0 0.1 0.5 0.8 0.8 0.8 1.0 1.0 1.0 0.9 0.4 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 00 00 02 09 10 10 10 10 10 10 10 10 00 07 00 00 00 00 00 00 00 00	
80 80 80 80 80 80 80 80 82 89 10 10 10 10 10 10 10 10 10 10 10 83 80 80 80 80 80 80 80	
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 0.5 0.2 0.2 0.2 0.2 0.2 0.6 1.0 1.0 0.3 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	
80 80 80 80 82 87 10 81 80 80 80 81 84 99 10 10 93 83 88 88 88 88 88 88 88	
80 80 80 80 8.8 10 10 8.6 8.5 8.5 8.5 8.5 10 10 10 10 07 83 88 88 88 88 88 88 88 88	
0.0 0.0 0.0 0.0 0.2 0.5 1.0 1.0 1.0 1.0 1.0 0.8 0.8 0.5 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 00 00 00 00 00 00 00 00	
<u></u>	
aa	
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 00 00 00 00 00 00 00 00	
	_
00 00 00 00 00 00 00 00 00 00 00 00 02 06 08 10 10 04 03 06 02 00 00 00 00 00 00 00	0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 00 00 00 00 00 02 0.9 1.0 0.8 0.3 0.0 0.1 0.7 1.0 0.5 0.0 00 00 00 00 00	
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 1.0 1.0 0.7 0.1 0.0 0.0 0.4 1.0 1.0 0.9 0.1 0.0 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 00 05 10 10 07 00 00 00 00 03 10 10 05 00 00 00 00 00 00 00 00 00 00	
0.0 0.0 0.0 0.0 0.1 10 10 0.1 0.1 0.5 0.8 0.9 10 10 1.0 0.3 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0	
80 80 80 80 80 81 99 10 10 10 10 10 09 09 10 09 80 80 80 80 80 80 80 80 80 80 80	
00 00 00 00 00 00 02 03 10 0.0 02 0.1 0.0 10 0.0 0.0 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 00 00 00 00 00 00 02 10 10 04 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00	
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 0.0 0.0	
00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 02 09 10 10 08 03 10 10 05 00 00 00 00 00 00	0.0
ww	
00 00 00 00 00 00 00 00 02 09 10 10 07 01 03 05 01 00 00 10 09 02 00 00 00 00 00	
aa	
00 00 00 00 00 00 00 0.8 1.0 0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0	
00 00 00 00 00 0.0 0.3 1.0 0.9 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 1.0 0.7 0.1 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 03 10 05 00 00 00 00 00 00 00 02 07 10 07 00 00 00 00 00 00 00 00	
00 00 00 00 00 03 10 09 00 00 00 00 00 00 05 09 10 06 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 10 10 10 0.0 0.2 0.1 0.3 0.7 0.9 10 0.9 0.7 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	
00 00 00 00 00 00 0.0 10 10 10 0.0 0.0 10 10 10 0.0 0.	
aa	
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 0.0 0.0	

Pooling across *spatial* locations achieves invariance w.r.t. small translations:



so the *channel* dimension remains *unchanged* after pooling.

Pooling across *spatial* locations achieves invariance w.r.t. small translations:



large response regardless of exact position of edge

[image credit Philip Isola]

Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- Case studies



[AlexNet paper]

ImageNet Classification Error (Top 5)



[image credit Philip Isola]



[image credit Philip Isola]





- increased depth: about 16 or 19 layers
- stack the same modules

"Very Deep Convolutional Networks for Large-Scale Image Recognition", Simonyan & Zisserman. ICLR 2015

[image credit Philip Isola and Kaiming He]

VGG '14 3x3 conv, 64 3x3 conv, 64, pool/2 3x3 conv, 128 3x3 conv, 128, pool/2 3x3 conv, 256 3x3 conv, 256 3x3 conv, 256 3x3 conv, 256, pool/2 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512, pool/2 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512, pool/2 fc, 4096 fc, 4096 fc, 1000

ResNet '16 7x7 conv, 64, /2 pool, /2 3x3 conv, 64 ★ 3x3 conv, 64 3x3 conv, 64 * 3x3 conv, 64 3x3 conv, 64 3x3 conv, 64 *---3x3 conv, 128, /2 3x3 conv, 128 3x3 conv, 256, /2 3x3 conv, 256 3x3 conv, 512, /2 3x3 conv, 512 avg pool fc 1000

Main developments:

- Residual block -- gradients can propagate faster (via the identity mapping)
- increased depth: > 100 layers



[He et al: Deep Residual Learning for Image Recognition, CVPR 2016] [image credit Philip Isola and Kaiming He]

Summary

- Though NN are technically "universal approximators", designing the NN structure so that it matches what we know about the underlying structure of the problem can substantially improve generalization ability and computational efficiency.
- Images are a very important input type and they have important properties that we can take advantage of: visual hierarchy, translation invariance, spatial locality.
- Convolution is an important image-processing technique that builds on these ideas. It can be interpreted as locally connected network, with weight-sharing.
- Pooling layer helps aggregate local info effectively, achieving bigger receptive field.
- We can train the parameters in a convolutional filtering function using backprop and combine convolutional filtering operations with other neural-network layers.

We'd love to hear

your thoughts. Thanks!