Lecture 7: Convolutional Neural Networks

Shen Shen
March 22, 2024

(videos edited from 3b1b; some slides adapted from Phillip Isola and Kaiming He)
Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary
A (feed-forward) neural network is
Recap: Backpropogation

Forward propagation to obtain the output (model’s guess)

\[ X = A^0 \xrightarrow{W^1} Z^1 \xrightarrow{f^1} A^1 \xrightarrow{W^2} Z^2 \xrightarrow{f^2} \ldots \xrightarrow{A^{L-1}} W^L \xrightarrow{Z^L} f^L \xrightarrow{A^L} \text{Loss} \]

\[ \frac{\partial \text{loss}}{\partial Z^1} \xleftarrow{\partial A^1} \frac{\partial \text{loss}}{\partial A^2} \xleftarrow{\partial Z^2} \ldots \xleftarrow{\partial A^{L-1}} \frac{\partial \text{loss}}{\partial Z^L} \xleftarrow{\partial A^L} \]

Backpropagation to obtain gradients with respect to the loss
Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary
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
Why do we need a specialized network?

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

Imagine even higher-resolution images, or more complex tasks...

Q: Why do we need a specialized network?

A: fully-connected nets don't scale well to (interesting) images
Why do we think 9 is 9?
Why do we think any of

9999999999999999

is a

9?
• Visual hierarchy

layering would help take care of that
- Visual hierarchy
- Spatial locality
- Translational invariance
CNN cleverly exploits

- Visual hierarchy
- Spatial locality
- Translational invariance

via

- layering (with nonlinear activations)
- convolution
- pooling

to handle images efficiently
cleverly exploits via to handle efficiently
Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN

- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors

- Max pooling
  - Larger window

- Typical architecture and summary
Convolutional layer might sound foreign, but...

<table>
<thead>
<tr>
<th>Layer</th>
<th>Forward-pass {do}</th>
<th>Back-prop {learn}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully-connected</td>
<td>Dot-product</td>
<td>Neurons</td>
</tr>
<tr>
<td>Convolutional</td>
<td>Convolution</td>
<td>Filters (kernels)</td>
</tr>
</tbody>
</table>
\[(0 \times -1) + (1 \times 1) = 1\]
\[(1 * -1) + (0 * 1) = -1\]
input image

| 0 | 1 | 0 | 1 | 1 |

filter

| -1 | 1 |

\((0 \times -1) + (1 \times 1) = 1\)

output image

| 1 | -1 | 1 |
input image
\[
\begin{bmatrix}
0 & 1 & 0 & 1 & 1
\end{bmatrix}
\]

filter
\[
\begin{bmatrix}
-1 & 1
\end{bmatrix}
\]

output image
\[
\begin{bmatrix}
1 & -1 & 1 & 0
\end{bmatrix}
\]

\[(1 \ast -1) + (1 \ast 1) = 0\]
input image  

|   | 0 | 1 | -1 | 1 | 1 |

filter  

|   | -1 | 1 |

output image  

|   | 1 | -1 | 2 | 0 |

(0 * -1) + (1 * 1) = 1  

(-1 * -1) + (1 * 1) = 2

- 'look' locally  
- parameter sharing  
- "template" matching
- 'look' locally

input image

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

filter

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

output image

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
- parameter sharing
- parameter sharing

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

convolve with

<table>
<thead>
<tr>
<th>-1</th>
<th>1</th>
</tr>
</thead>
</table>

or dot with

<table>
<thead>
<tr>
<th>-1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

= 1 -1 1 0
convolve with ? = 

0 1 0 1 1

dot-product with ? = 

0 1 0 1 1
convolve with

\[
\begin{array}{c}
1
\end{array}
\]

dot-product with

\[
I_{5 \times 5}
\]
Convolution: a 2-D example

input

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

filter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

output
## Convolution: a 2-D example

<table>
<thead>
<tr>
<th>input</th>
<th>filter</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Input Matrix" /></td>
<td><img src="image2" alt="Filter Matrix" /></td>
<td><img src="image3" alt="Output Matrix" /></td>
</tr>
</tbody>
</table>
Convolution: a 2-D example

input

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

filter

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

output

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>-4</td>
</tr>
</tbody>
</table>
Convolution: a 2-D example

input

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

filter

\[
\begin{array}{ccc}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

output

\[
\begin{array}{cccccccc}
-3 & -4 & -4 \\
\end{array}
\]
## Convolution: a 2-D example

<table>
<thead>
<tr>
<th>input</th>
<th>filter</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 1</td>
<td>-3 -4 -4 -4</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>1 2 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 0 0</td>
<td>-1 -2 -1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

The convolution operation is demonstrated by sliding the filter over the input matrix and calculating the output matrix.
Convolution: a 2-D example

input

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

filter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

output

<table>
<thead>
<tr>
<th>-3</th>
<th>-4</th>
<th>-4</th>
<th>-4</th>
<th>-4</th>
<th>-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>-4</td>
<td>-4</td>
<td>-3</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Convolution: padding

input: $H \times W = 8 \times 8$

filter

output: $H \times W = 6 \times 6$
Convolution: padding

input: $H \times W = 8 \times 8$

output: $H \times W = 6 \times 6$
Convolution: padding

input: $H \times W = 8 \times 8$

output: $H \times W = 6 \times 6$
Convolution: padding

input: $H \times W = 8 \times 8$

output: $H \times W = 6 \times 6$
Convolution: padding

input: $H \times W = 8 \times 8$

output: $H \times W = 6 \times 6$
Convolution: padding

input: $H \times W = 8 \times 8$

output: $H \times W = 6 \times 6$
Convolution: padding

input: $H \times W = 8 \times 8$

output: $H \times W = 6 \times 6$
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
Convolution: padding

input: 8 × 8, + pad

output: H × W = 8 × 8
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
**Convolution: padding**

**input: 8 × 8, + pad**

```
  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
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
```

**output: H × W = 8 × 8**

```
  0
  0
  0
  0
  0
  0
  0
  1
  0
  0
  0
  0
  0
  0
  0
```
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
Convolution: padding

input: $8 \times 8$, + pad

output: $H \times W = 8 \times 8$
Convolution: stride

Input

Output

Stride = 2
Convolution: stride

input

filter

output

stride = 2
Convolution: stride

stride = 2
Convolution: stride

stride = 2
Convolution: stride

Input:

```
  o  o  o  o  o
  o  o  o  o  o
  o  o  o  o  o
  o  o  o  o  o
  o  o  o  o  o
```

Filter:

```
  o  o  o
  o  o  o
  o  o  o
```

Output:

```
  o  o  o  o  o
  o  o  o  o  o
  o  o  o  o  o
```
Convolution: stride

input

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

filter

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

output

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

stride = 2
Convolution: stride

input

output
Convolution: stride

input: $H \times W = 8 \times 8$

![Input Grid]

filter

output: $H \times W = 4 \times 4$

![Output Grid]

stride = 2
Convolution: stride

input: $H \times W = 8 \times 8$

stride = 2

output: $H \times W = 4 \times 4$
Outline

• Recap (fully-connected net)
• Motivation and big picture ideas of CNN
• Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
• Max pooling
  - Larger window
• Typical architecture and summary
A tender intro to tensor: [image credit: tensorflow]
- 3d tensor input, depth $d$
- 3d tensor filter, depth $d$
- 2d tensor (matrix) output
input tensor

filters

outputs
3d tensor input, depth $d$

$k$ 3d filters:
- each filter of depth $d$
- each filter makes a 2d tensor (matrix) output

total output 3d tensor, depth $k$
Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary
Pooling across spatial locations achieves stability w.r.t. small translations:
Pooling across spatial locations achieves
stability w.r.t. small translations:

large response regardless of exact position of edge
Pooling across spatial locations achieves stability w.r.t. small translations:
Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary
We'd love it for you to share some lecture feedback.

Thanks!