Convolutional neural networks

Many slides from Rob Fergus, Andrej Karpathy
Outline

• Basic convolutional layer
  • Backward pass
• Max pooling layer
Let’s design a neural network for images

- This kind of design is known as *multi-layer perceptron* (MLP)
Let’s design a neural network for images

- This kind of design is known as multi-layer perceptron (MLP)
- What is wrong with this?

Recall: MLP as bank of whole-image templates
Convolutional architecture

- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights.
Convolutional architecture

• Let's limit the receptive fields of units, tile them over the input image, and share their weights.
Convolutional architecture

- Let's limit the *receptive fields* of units, tile them over the input image, and share their weights.
- This is equivalent to sliding the learned filter over the image, computing dot products at every location.
Convolution example

Adapted from D. Fouhey and J. Johnson
Convolution example

\[ y_{11} = x_{11} \cdot w_{11} + x_{12} \cdot w_{12} + x_{13} \cdot w_{13} + \ldots + x_{33} \cdot w_{33} \]

Adapted from D. Fouhey and J. Johnson
Convolution example

\[
\begin{align*}
\text{Input} & \\
\begin{array}{cccccc}
  x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\
  x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} \\
  x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} \\
  x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} \\
  x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} \\
\end{array} & \\
\text{Filter} & \\
\begin{array}{ccc}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33} \\
\end{array} & \\
\text{Output} & \\
\begin{array}{cc}
  y_{11} & y_{12} \\
\end{array}
\end{align*}
\]

\[
y_{12} = x_{12} \cdot w_{11} + x_{13} \cdot w_{12} + x_{14} \cdot w_{13} + \ldots + x_{34} \cdot w_{33}
\]

Adapted from D. Fouhey and J. Johnson
Convolution example

Adapted from D. Fouhey and J. Johnson
Convolution example

\[ y_{14} = x_{14} \cdot w_{11} + x_{15} \cdot w_{12} + x_{16} \cdot w_{13} + \ldots + x_{36} \cdot w_{33} \]

Adapted from D. Fouhey and J. Johnson
### Convolution Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Filter</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{11}$</td>
<td>$w_{11}$</td>
<td>$y_{11}$</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>$w_{12}$</td>
<td>$y_{12}$</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>$w_{13}$</td>
<td>$y_{13}$</td>
</tr>
<tr>
<td>$x_{14}$</td>
<td>$w_{21}$</td>
<td>$y_{21}$</td>
</tr>
<tr>
<td>$x_{15}$</td>
<td>$w_{22}$</td>
<td></td>
</tr>
<tr>
<td>$x_{16}$</td>
<td>$w_{23}$</td>
<td></td>
</tr>
<tr>
<td>$x_{21}$</td>
<td>$w_{31}$</td>
<td></td>
</tr>
<tr>
<td>$x_{22}$</td>
<td>$w_{32}$</td>
<td></td>
</tr>
<tr>
<td>$x_{23}$</td>
<td>$w_{33}$</td>
<td></td>
</tr>
<tr>
<td>$x_{31}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{32}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{33}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{34}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{35}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{36}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{41}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{42}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{43}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{44}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{45}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{46}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{51}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{52}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{53}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{54}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{55}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{56}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$y_{21} = x_{21} \cdot w_{11} + x_{22} \cdot w_{12} + x_{23} \cdot w_{13} + \ldots + x_{43} \cdot w_{33}$

Adapted from [D. Fouhey and J. Johnson](https://example.com)
### Convolution example

\[
y_{22} = x_{22} \cdot w_{11} + x_{23} \cdot w_{12} + x_{24} \cdot w_{13} + \ldots + x_{44} \cdot w_{33}
\]

Adapted from D. Fouhey and J. Johnson
Convolution example

\[ y_{23} = x_{23} \cdot w_{11} + x_{24} \cdot w_{12} + x_{25} \cdot w_{13} + \ldots + x_{45} \cdot w_{33} \]

Adapted from D. Fouhey and J. Johnson
Convolution example

Input

<table>
<thead>
<tr>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>$x_{13}$</th>
<th>$x_{14}$</th>
<th>$x_{15}$</th>
<th>$x_{16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
<td>$x_{24}$</td>
<td>$x_{25}$</td>
<td>$x_{26}$</td>
</tr>
<tr>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
<td>$x_{34}$</td>
<td>$x_{35}$</td>
<td>$x_{36}$</td>
</tr>
<tr>
<td>$x_{41}$</td>
<td>$x_{42}$</td>
<td>$x_{43}$</td>
<td>$x_{44}$</td>
<td>$x_{45}$</td>
<td>$x_{46}$</td>
</tr>
<tr>
<td>$x_{51}$</td>
<td>$x_{52}$</td>
<td>$x_{53}$</td>
<td>$x_{54}$</td>
<td>$x_{55}$</td>
<td>$x_{56}$</td>
</tr>
</tbody>
</table>

Filter

<table>
<thead>
<tr>
<th>$w_{11}$</th>
<th>$w_{12}$</th>
<th>$w_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{21}$</td>
<td>$w_{22}$</td>
<td>$w_{23}$</td>
</tr>
<tr>
<td>$w_{31}$</td>
<td>$w_{32}$</td>
<td>$w_{33}$</td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>$y_{11}$</th>
<th>$y_{12}$</th>
<th>$y_{13}$</th>
<th>$y_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_{21}$</td>
<td>$y_{22}$</td>
<td>$y_{23}$</td>
<td>$y_{24}$</td>
</tr>
<tr>
<td>$y_{31}$</td>
<td>$y_{32}$</td>
<td>$y_{33}$</td>
<td>$y_{34}$</td>
</tr>
</tbody>
</table>

Adapted from D. Fouhey and J. Johnson
Convolutional architecture

Output feature map resolution depends on *padding* and *stride*.

*learned weights*

*image*

*feature map*

*Input*

*Output*

No padding, stride 1

Animation source
Convolutional architecture

Output feature map resolution depends on *padding* and *stride*.
Convolutional architecture

With padding, stride 2

Output feature map resolution depends on *padding* and *stride*

learned weights

image

With padding, stride 2

Output

Input
Convolutional architecture

Output feature map resolution depends on *padding* and *stride*

With padding, spatial resolution remains the same if stride of 1 is used, is reduced by factor of $1/S$ if stride of $S$ is used
Convolutional architecture

- Image
- Learned weights
- Feature map
Convolutional architecture

Learned weights can be thought of as local templates
Convolution and traditional feature extraction

bank of $K$ filters

$K$ feature maps

image

feature map
Elementwise nonlinearity

Almost always directly followed by a ReLU:

$$\max(0, x)$$

Some alternatives to ReLU:

**Leaky ReLU**

$$\max(0.1x, x)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

Source: Stanford 231n
Three-dimensional convolutions

- What if the input to a convolutional layer is a stack of $K$ feature maps?
Three-dimensional convolutions

- What if the input to a convolutional layer is a stack of $K$ feature maps?
Convolutional layer example

Input Volume (pad 1) (7x7x3)  Filter W0 (3x3x3)  Filter W1 (3x3x3)

```
x[:, :, 0]
x[:, :, 1]
x[:, :, 2]
```

```
w0[:, :, 0]
w0[:, :, 1]
w0[:, :, 2]
```

```
w1[:, :, 0]
w1[:, :, 1]
w1[:, :, 2]
```

Bias b0 (1x1x1)  Bias b1 (1x1x1)

```
b0[:, :, 0]
```

```
b1[:, :, 0]
```

Stride = 2

Convolutional layer example

Stride = 2

Convolutional layer: Computational cost

- Assuming the input feature maps have spatial resolution $H \times W$, how many operations are needed to compute the output feature volume?
  - $F^2KLHW$
More generally: Groupwise convolutions

- Split up the $K$ feature maps into $G$ groups, perform convolutions within each group separately, concatenate the results
Convolutional layer: Details

• Efficient implementation: reshape all image neighborhoods into columns (im2col operation), do matrix-vector multiplication

• Backward pass: special case of linear layer, operations also turn out to be convolutions
  • Downstream gradient (of error w.r.t. input) is a transposed convolution, or convolution of output with filter flipped both horizontally and vertically
Outline

• Basic convolutional layer
• Backward pass
Convolutional layer: Backward pass

- Let’s take a 1D example with a filter of width 3:

\[
\begin{align*}
  z^i &= w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1} 
\end{align*}
\]
Review: Backward pass

Parameter update:
\[
\frac{\partial e}{\partial w} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial w}
\]

Local gradient

Downstream gradient:
\[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x}
\]

Upstream gradient:
\[
\frac{\partial e}{\partial z}
\]

Forward pass

Backward pass
Convolutional layer: Backward pass

Backward pass (w.r.t. $x$)

Vector-matrix form:
\[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x}
\]

Output:
\[
\begin{bmatrix}
\frac{\partial e}{\partial x^{i-1}} \\
\frac{\partial e}{\partial x^i} \\
\frac{\partial e}{\partial x^{i+1}}
\end{bmatrix}
\]

Input:
\[
\begin{bmatrix}
\frac{\partial e}{\partial z^{i-1}} \\
\frac{\partial e}{\partial z^i} \\
\frac{\partial e}{\partial z^{i+1}}
\end{bmatrix}
\]

\[
z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}
\]
Convolutional layer: Backward pass

Backward pass (w.r.t. $x$)

Vector-matrix form:
\[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x}
\]

Output:
\[
\frac{\partial e}{\partial x^{i-1}} \frac{\partial e}{\partial x^i} \frac{\partial e}{\partial x^{i+1}}
\]

Input:
\[
\frac{\partial e}{\partial z^{i-1}} \frac{\partial e}{\partial z^i} \frac{\partial e}{\partial z^{i+1}}
\]

\[
z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}
\]
Convolutional layer: Backward pass

Backward pass (w.r.t. $x$)

\[
\frac{\partial e}{\partial x^i} = \sum_{j=1}^{N} \frac{\partial e}{\partial z^j} \frac{\partial z^j}{\partial x^i} \\
= \frac{\partial e}{\partial z^{i-1}} \frac{\partial z^{i-1}}{\partial x^i} + \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial x^i} + \frac{\partial e}{\partial z^{i+1}} \frac{\partial z^{i+1}}{\partial x^i}
\]

Output: ...

\[\frac{\partial e}{\partial x^{i-1}}\]
\[\frac{\partial e}{\partial x^i}\]
\[\frac{\partial e}{\partial x^{i+1}}\] ...

Input: ...

\[\frac{\partial e}{\partial z^{i-1}}\]
\[\frac{\partial e}{\partial z^i}\]
\[\frac{\partial e}{\partial z^{i+1}}\] ...

\[z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}\]
Convolutional layer: Backward pass

Backward pass (w.r.t. $x$)

$$\frac{\partial e}{\partial x^i} = \sum_{j=1}^{N} \frac{\partial e}{\partial z^j} \frac{\partial z^j}{\partial x^i}$$

$$= \frac{\partial e}{\partial z^{i-1}} \frac{\partial z^{i-1}}{\partial x^i} + \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial x^i} + \frac{\partial e}{\partial z^{i+1}} \frac{\partial z^{i+1}}{\partial x^i}$$

$$= w^3 \frac{\partial e}{\partial z^{i-1}} + w^2 \frac{\partial e}{\partial z^i} + w^1 \frac{\partial e}{\partial z^{i+1}}$$

Input:

$$\frac{\partial e}{\partial z^{i-1}} \quad \frac{\partial e}{\partial z^i} \quad \frac{\partial e}{\partial z^{i+1}}$$

Output:

$$\frac{\partial e}{\partial x^{i-1}} \quad \frac{\partial e}{\partial x^i} \quad \frac{\partial e}{\partial x^{i+1}}$$

$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$
Convolutional layer: Backward pass

Backward pass (w.r.t. $x$)

This is called a *transposed convolution*

$$\frac{\partial e}{\partial x^i} = w^3 \frac{\partial e}{\partial z^{i-1}} + w^2 \frac{\partial e}{\partial z^i} + w^1 \frac{\partial e}{\partial z^{i+1}}$$

Output:

$$\frac{\partial e}{\partial x^i}$$

Input:

$$\frac{\partial e}{\partial z^{i-1}} \quad \frac{\partial e}{\partial z^i} \quad \frac{\partial e}{\partial z^{i+1}}$$

$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$
Backward pass

\[ \frac{\partial e}{\partial w} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial w} \]

Output:
\[ \frac{\partial e}{\partial w^1}, \frac{\partial e}{\partial w^2}, \frac{\partial e}{\partial w^3} \]

Input:
\[ \frac{\partial e}{\partial z^{i-1}}, \frac{\partial e}{\partial z^i}, \frac{\partial e}{\partial z^{i+1}} \]

\[ z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1} \]
Backward pass

\[ \frac{\partial e}{\partial w} = \sum_i \frac{\partial e}{\partial z_i} \frac{\partial z_i}{\partial w} = \sum_i \frac{\partial e}{\partial z_i} x_i \]

\[ z_i = w^1 x_{i-1} + w^2 x_i + w^3 x_{i+1} \]
Backward pass

Backward pass (w.r.t. $w$)

\[
\frac{\partial e}{\partial w^2} = \sum_i \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial w^2} = \sum_i \frac{\partial e}{\partial z^i} x^i
\]

\[
z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}
\]
Backward pass

Backward pass (w.r.t. $w$)

$$\frac{\partial e}{\partial w^3} = \sum_i \frac{\partial e}{\partial z^i} \frac{\partial z^i}{\partial w^3} = \sum_i \frac{\partial e}{\partial z^i} x^{i+1}$$

Output:

$$\frac{\partial e}{\partial w^1} \quad \frac{\partial e}{\partial w^2} \quad \frac{\partial e}{\partial w^3}$$

Input:

$$\frac{\partial e}{\partial z^{i-1}} \quad \frac{\partial e}{\partial z^i} \quad \frac{\partial e}{\partial z^{i+1}}$$

$$z^i = w^1 x^{i-1} + w^2 x^i + w^3 x^{i+1}$$
Outline

- Basic convolutional layer
  - Backward pass
- Max pooling layer
Max pooling layer

Usually: $F = 2$ or $3$, $S = 2$
Max pooling: Example

Single channel

Max pooling with $2 \times 2$ kernel size and stride 2

Source: J. Johnson
Max pooling: Example

Single channel

\[
\begin{array}{cccc}
1 & 1 & 2 & 4 \\
5 & 6 & 7 & 8 \\
3 & 2 & 1 & 0 \\
1 & 2 & 3 & 4 \\
\end{array}
\]

Max pooling with \(2 \times 2\) kernel size and stride 2

Source: J. Johnson
Max pooling: Example

Max pooling with $2 \times 2$ kernel size and stride 2

Source: J. Johnson
Max pooling layer

$K$ feature maps, resolution $1/S$

$F \times F$ pooling window, stride $S$
Usually: $F = 2$ or $3$, $S = 2$

Backward pass: upstream gradient is passed back only to the unit with max value
Simplified CNN pipeline

Softmax layer:
The *receptive field* of a unit is the region of the input feature map whose values contribute to the response of that unit (either in the previous layer or in the initial image).
Receptive field

3x3 convolutions, stride 1

Receptive field size: 5
Receptive field

3x3 convolutions, stride 1

Each successive convolution adds $F - 1$ to the receptive field size.
With $L$ layers the receptive field size is $1 + L \times (F - 1)$.
Receptive field

3x3 convolutions, stride 2

Receptive field size: 3
Receptive field

3x3 convolutions, stride 2
Receptive field

Input

Output

Receptive field size: 7

3x3 convolutions, stride 2
Deep Nets with striding have large receptive fields

Source: https://distill.pub/2019/computing-receptive-fields/