Announcements

• **Homeworks:**
  • HW 7 due in two weeks; provide feedback

• **Midterms are being graded**

• **Class roadmap:**

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, Mar 28</td>
<td>Deep Learning I</td>
</tr>
<tr>
<td>Thursday, Mar 30</td>
<td>Deep Learning II</td>
</tr>
<tr>
<td>Tuesday, April 4</td>
<td>Neural Network Review</td>
</tr>
<tr>
<td>Thursday, April 6</td>
<td>Search</td>
</tr>
</tbody>
</table>
Today’s Goals
Today’s Goals

• Build an understanding of convolutional neural networks.
Today’s Goals

• Build an understanding of convolutional neural networks.
• Why do we want convolutional layers?
Today’s Goals

• Build an understanding of convolutional neural networks.
• Why do we want convolutional layers?
• What are convolutional neural networks?
Today’s Goals

• Build an understanding of convolutional neural networks.
• Why do we want convolutional layers?
• What are convolutional neural networks?
  • 2D vs 3D convolutional networks.
Today’s Goals

• Build an understanding of convolutional neural networks.
• Why do we want convolutional layers?
• What are convolutional neural networks?
  • 2D vs 3D convolutional networks.
  • Padding and stride.
Today’s Goals

• Build an understanding of convolutional neural networks.
• Why do we want convolutional layers?
• What are convolutional neural networks?
  • 2D vs 3D convolutional networks.
  • Padding and stride.
• Multiple input and output channels
Today’s Goals

• Build an understanding of convolutional neural networks.
• Why do we want convolutional layers?
• What are convolutional neural networks?
  • 2D vs 3D convolutional networks.
  • Padding and stride.
• Multiple input and output channels
• Pooling
Review: Deep Neural Networks

\[ h_1 = \sigma(W^{(1)}x + b^{(1)}) \]
\[ h_2 = \sigma(W^{(2)}h_1 + b^{(2)}) \]
\[ h_3 = \sigma(W^{(3)}h_2 + b^{(3)}) \]
\[ f = W^{(4)}h_3 + b^{(4)} \]
\[ p = \text{softmax}(f) \]

NNs are composition of nonlinear functions
How to classify
Cats vs. dogs?
How to classify
Cats vs. dogs?
How to classify Cats vs. dogs?

Dual 12MP wide-angle and telephoto cameras
How to classify Cats vs. dogs?

36M floats in a RGB image!
Cats vs. dogs?
Cats vs. dogs?

Fully Connected Networks

Input

Hidden layer
100 neurons

Output
Cats vs. dogs?

~ 36M elements x 100 = ~3.6B parameters!
Convolutions come to rescue!
Where is Waldo?
Why Convolution?

- Translation Invariance
- Locality
2-D Convolution

<table>
<thead>
<tr>
<th>Input</th>
<th>Kernel</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2</td>
<td>0 1 3</td>
<td>19 25</td>
</tr>
<tr>
<td>3 4 5</td>
<td>2 3</td>
<td>37 43</td>
</tr>
<tr>
<td>6 7 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19
\]
2-D Convolution

Input

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Kernel

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Output

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>25</td>
</tr>
<tr>
<td>37</td>
<td>43</td>
</tr>
</tbody>
</table>

\[0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19\]
2-D Convolution

Input

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Kernel

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>19</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>43</td>
</tr>
</tbody>
</table>

$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$

(vdumoulin@ Github)
2-D Convolution

$$1x0 + 2x1 + 4x2 + 5x3 = 25$$
2-D Convolution

<table>
<thead>
<tr>
<th>Input</th>
<th>Kernel</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2</td>
<td>0 1</td>
<td>19 25</td>
</tr>
<tr>
<td>3 4 5</td>
<td>2 3</td>
<td>37 43</td>
</tr>
<tr>
<td>6 7 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37$$
2-D Convolution

\[
\begin{array}{c}
\text{Input} \\
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array} 
\times 
\begin{array}{c}
\text{Kernel} \\
0 & 1 \\
2 & 3 \\
\end{array} 
= 
\begin{array}{c}
\text{Output} \\
19 & 25 \\
37 & 43 \\
\end{array}
\]

\[4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43\]
2-D Convolution Layer

• **X**: $n_h \times n_w$ input matrix
• **W**: $k_h \times k_w$ kernel matrix
• **Y**: $(n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

\[
Y = X * W
\]
2-D Convolution Layer

X: $n_h \times n_w$ input matrix
W: $k_h \times k_w$ kernel matrix
Y: $(n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

Y = X * W

Convolution operator not multiplication
2-D Convolution Layer

\[ Y = X \ast W + b \]

- \( X: n_h \times n_w \) input matrix
- \( W: k_h \times k_w \) kernel matrix
- \( b: \) scalar bias
- \( Y: (n_h - k_h + 1) \times (n_w - k_w + 1) \) output matrix

\( W \) and \( b \) are learnable parameters
Examples

(wikipedia)
Examples

\[
\begin{bmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1 \\
\end{bmatrix}
\]

Edge Detection

(wikipedia)
Examples

\[
\begin{bmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1 \\
\end{bmatrix}
\]

Edge Detection

\[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0 \\
\end{bmatrix}
\]

Sharpen

(wikipedia)
Examples

\[
\begin{bmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1 \\
\end{bmatrix}
\]

Edge Detection

\[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0 \\
\end{bmatrix}
\]

Sharpen

\[
\frac{1}{16}
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{bmatrix}
\]

Gaussian Blur

(wikipedia)
Convolutional Neural Networks

• Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers

• Strong empirical performance in applications – particularly computer vision.

• Examples: image classification, object detection.
Advantage: sparse interaction

Fully connected layer, $m \times n$ edges

$m$ output nodes

$n$ input nodes

Figure from Deep Learning, by Goodfellow, Bengio, and Courville
Advantage: sparse interaction

Convolutional layer, \( \leq m \times k \) edges

\( m \) output nodes

\( k \) kernel size

\( n \) input nodes

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville
Q1. Suppose we want to perform convolution as follows. What’s the output?

\[
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array} 
\ast 
\begin{array}{cc}
0 & 1 \\
1 & -1 \\
\end{array} 
+ 1 = ?
\]

A. \[
\begin{array}{cc}
1 & 2 \\
4 & 5 \\
\end{array}
\]
B. \[
\begin{array}{cc}
1 & 2 \\
3 & 4 \\
\end{array}
\]
C. \[
\begin{array}{cc}
1 & 3 \\
3 & 5 \\
\end{array}
\]
D. \[
\begin{array}{cc}
0 & 1 \\
3 & 4 \\
\end{array}
\]
Q1. Suppose we want to perform convolution as follows. What’s the output?

\[
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array} \ast \begin{array}{cc}
0 & 1 \\
1 & -1 \\
\end{array} + 1 = \begin{array}{cc}
1 & 2 \\
4 & 5 \\
\end{array}
\]

A. \[
\begin{array}{cc}
1 & 2 \\
4 & 5 \\
\end{array}
\]

B. \[
\begin{array}{cc}
1 & 2 \\
3 & 4 \\
\end{array}
\]

B. \[
\begin{array}{cc}
1 & 3 \\
3 & 5 \\
0 & 1 \\
3 & 4 \\
\end{array}
\]

\[
0 \times 0 + 1 \times 1 + 3 \times 1 + 4 \times (-1) + 1 = 1 \\
1 \times 0 + 2 \times 1 + 4 \times 1 + 5 \times (-1) + 1 = 2 \\
3 \times 0 + 4 \times 1 + 6 \times 1 + 7 \times (-1) + 1 = 4 \\
4 \times 0 + 5 \times 1 + 7 \times 1 + 8 \times (-1) + 1 = 5
\]
Padding and Stride
Padding

• Given a 32 x 32 input image
• Apply convolution with 5 x 5 kernel
  • 28 x 28 output with 1 layer
  • 4 x 4 output with 7 layers
Padding

• Given a 32 x 32 input image
• Apply convolution with 5 x 5 kernel
  • 28 x 28 output with 1 layer
  • 4 x 4 output with 7 layers
• Shape decreases faster with larger kernels
• Shape reduces from \( n_h \times n_w \) to
  \[
  (n_h - k_h + 1) \times (n_w - k_w + 1)
  \]
Convolutional Layers: Padding
Convolutional Layers: Padding

**Padding** adds rows/columns around input
Convolutional Layers: Padding

Padding adds rows/columns around input
**Convolutional Layers: Padding**

**Padding** adds rows/columns around input

![Diagram showing convolution with padding]

Input:

```
0 0 0 0 0 0
0 0 1 2 0 0
0 3 4 5 0 0
0 6 7 8 0 0
0 0 0 0 0 0
```

Kernel:

```
0 1
2 3
```

Output:

```
0 3 8 4
9 19 25 10
21 37 43 16
6 7 8 0
```
Convolutional Layers: Padding
Convolutional Layers: Padding

**Padding** adds rows/columns around input
Convolutional Layers: Padding

**Padding** adds rows/columns around input
Convolutional Layers: Padding

Padding adds rows/columns around input

- Why?
Convolutional Layers: Padding

**Padding** adds rows/columns around input

- Why?

1. Keeps *edge information*
Convolutional Layers: Padding

**Padding** adds rows/columns around input

- Why?

1. Keeps **edge information**
2. Preserves sizes / allows deep networks

- ie, for a 32x32 input image, 5x5 kernel, after 1 layer, get 28x28, after 7 layers, **only 4x4**
Convolutional Layers: Padding

Padding adds rows/columns around input

• Why?

1. Keeps **edge information**
2. Preserves sizes / allows deep networks
   • ie, for a 32x32 input image, 5x5 kernel, after 1 layer, get 28x28, after 7 layers, **only 4x4**
3. Can combine different filter sizes
Convolutional Layers: Padding
Convolutional Layers: Padding

- Padding $p_h$ rows and $p_w$ columns, output shape is
Convolutional Layers: Padding

- Padding $p_h$ rows and $p_w$ columns, output shape is

$$\left( n_h-k_h+p_h+1 \right) \times \left( n_w-k_w+p_w+1 \right)$$
Convolutional Layers: Padding

- Padding $p_h$ rows and $p_w$ columns, output shape is

$$ (n_h-k_h+p_h+1) \times (n_w-k_w+p_w+1) $$

- Common choice is $p_h = k_h-1$ and $p_w = k_w-1$

- Odd $k_h$: pad $p_h/2$ on both sides
- Even $k_h$: pad $\text{ceil}(p_h/2)$ on top, $\text{floor}(p_h/2)$ on bottom
**Stride**

- **Stride is the #rows / #columns per slide**

Example: strides of 3 and 2 for height and width

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

\[
0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8
\]
\[
0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6
\]
**Stride**

- Stride is the #rows / #columns per slide.

Example: strides of 3 and 2 for height and width.

**Input**

```
0 0 0 0 0
0 0 1 2 0
0 3 4 5 0
0 6 7 8 0
0 0 0 0 0
```

**Kernel**

```
0 1
2 3
```

**Output**

```
0 8
6 8
```

Stride 2,2

\[
0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8 \\
0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6
\]
Stride

- Stride is the #rows / #columns per slide.

Example: strides of 3 and 2 for height and width.

Input:

```
0 0 0 0 0
0 0 1 2 0
0 3 4 5 0
0 6 7 8 0
0 0 0 0 0
```

Kernel:

```
0 1
2 3
```

Output:

```
0 8
6 8
```

\[0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8\]
\[0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6\]
Convolutional Layers: Stride
Convolutional Layers: Stride

- Given stride $s_h$ for the height and stride $s_w$ for the width, the output shape is
Convolutional Layers: Stride

- Given stride $s_h$ for the height and stride $s_w$ for the width, the output shape is

$$\left\lfloor \frac{n_h - k_h + p_h + s_h}{s_h} \right\rfloor \times \left\lfloor \frac{n_w - k_w + p_w + s_w}{s_w} \right\rfloor$$
Convolutional Layers: Stride

- Given stride $s_h$ for the height and stride $s_w$ for the width, the output shape is

$$\left\lfloor \frac{n_h-k_h+p_h+s_h}{s_h} \right\rfloor \times \left\lfloor \frac{n_w-k_w+p_w+s_w}{s_w} \right\rfloor$$

- Set $p_h = k_h - 1$, $p_w = k_w - 1$, then get
Convolutional Layers: Stride

• Given stride $s_h$ for the height and stride $s_w$ for the width, the output shape is

$$\left\lfloor \frac{(n_h-k_h+p_h+s_h)}{s_h} \right\rfloor \times \left\lfloor \frac{(n_w-k_w+p_w+s_w)}{s_w} \right\rfloor$$

• Set $p_h = k_h - 1$, $p_w = k_w - 1$, then get

$$\left\lfloor \frac{(n_h+s_h - 1)}{s_h} \right\rfloor \times \left\lfloor \frac{(n_w+s_w - 1)}{s_w} \right\rfloor$$
Q2. Suppose we want to perform convolution on a single channel image of size 7x7 (no padding) with a kernel of size 3x3, and stride = 2. What is the dimension of the output?

A. 3x3
B. 7x7
C. 5x5
D. 2x2
Q2. Suppose we want to perform convolution on a single channel image of size 7x7 (no padding) with a kernel of size 3x3, and stride = 2. What is the dimension of the output?

A. 3x3
B. 7x7
C. 5x5
D. 2x2

\[
\left\lfloor \frac{n_h-k_h+p_h+s_h}{s_h} \right\rfloor \times \left\lfloor \frac{n_w-k_w+p_w+s_w}{s_w} \right\rfloor
\]
Multiple Input and Output Channels
Multiple Input Channels

- Color image may have three RGB channels
- Converting to grayscale loses information
Multiple Input Channels

• Color image may have three RGB channels
• Converting to grayscale loses information
Multiple Input Channels

• Have a kernel matrix for each channel, and then sum results over channels
Multiple Input Channels

- Have a kernel matrix for each channel, and then sum results over channels

![Image showing input and kernel matrices and their multiplication result]
Multiple Input Channels

- Have a kernel matrix for each channel, and then sum results over channels
Multiple Input Channels

- Have a kernel matrix for each channel, and then sum results over channels
Multiple Input Channels

• Have a kernel matrix for each channel, and then sum results over channels

\[(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56\]
Multiple Input Channels

• Have a kernel matrix for each channel, and then sum results over channels

\[
\begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Output} \\
\begin{array}{cc}
56 & 72 \\
104 & 120 \\
\end{array}
\end{array}
\end{array}
\]

\[
(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) \\
+ (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) \\
= 56
\]
Convolutional Layers: Channels

"Slices" of tensors

Tensor: generalization of matrix to higher dimensions
Convolutional Layers: Channels

• How to integrate multiple channels?
• Have a kernel for each channel, and then sum results over channels

“Slices” of tensors

Tensor: generalization of matrix to higher dimensions
Convolutional Layers: Channels

• How to integrate multiple channels?
• Have a kernel for each channel, and then sum results over channels

\[ \mathbf{X} : c_i \times n_h \times n_w \]

“Slices” of tensors

Tensor: generalization of matrix to higher dimensions
Convolutional Layers: Channels

- How to integrate multiple channels?
  - Have a kernel for each channel, and then sum results over channels

\[ X : c_i \times n_h \times n_w \]
\[ W : c_i \times k_h \times k_w \]

"Slices" of tensors

Tensor: generalization of matrix to higher dimensions
Convolutional Layers: Channels

- How to integrate multiple channels?
- Have a kernel for each channel, and then sum results over channels

\[
X : c_i \times n_h \times n_w \\
W : c_i \times k_h \times k_w \\
Y : m_h \times m_w
\]

“Slices” of tensors

Tensor: generalization of matrix to higher dimensions
Convolutional Layers: Channels

- How to integrate multiple channels?
  - Have a kernel for each channel, and then sum results over channels

\[
\begin{align*}
X & : c_i \times n_h \times n_w \\
W & : c_i \times k_h \times k_w \\
Y & : m_h \times m_w
\end{align*}
\]

\[
Y = \sum_{i=0}^{c_i} X_{i,:,:} \star W_{i,:,:} \quad \text{"Slices" of tensors}
\]

Tensor: generalization of matrix to higher dimensions
Multiple Output Channels

• No matter how many inputs channels, so far we always get single output channel
• We can have **multiple 3-D kernels**, each one generates an output channel
Multiple Output Channels

- No matter how many inputs channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates an output channel
- Input
- Kernels
- Output
Multiple Output Channels

• No matter how many inputs channels, so far we always get single output channel
• We can have **multiple 3-D kernels**, each one generates an output channel
• Input $X : c_i \times n_h \times n_w$
• Kernels
• Output
Multiple Output Channels

• No matter how many inputs channels, so far we always get single output channel
• We can have multiple 3-D kernels, each one generates an output channel
• Input $X: c_i \times n_h \times n_w$
• Kernels $W: c_o \times c_i \times k_h \times k_w$
• Output
Multiple Output Channels

• No matter how many inputs channels, so far we always get single output channel
• We can have **multiple 3-D kernels**, each one generates an output channel
• Input  \( X : c_i \times n_h \times n_w \)
• Kernels  \( W : c_o \times c_i \times k_h \times k_w \)
• Output  \( Y : c_o \times m_h \times m_w \)
Multiple Output Channels

- No matter how many inputs channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates an output channel
  
<table>
<thead>
<tr>
<th>Input</th>
<th>$X : c_i \times n_h \times n_w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernels</td>
<td>$W : c_o \times c_i \times k_h \times k_w$</td>
</tr>
<tr>
<td>Output</td>
<td>$Y : c_o \times m_h \times m_w$</td>
</tr>
</tbody>
</table>

\[ Y_{i,:,:} = X \star W_{i,:,:} \]
Multiple Output Channels

• No matter how many inputs channels, so far we always get single output channel
• We can have **multiple 3-D kernels**, each one generates an output channel
• Input $X : c_i \times n_h \times n_w$
• Kernels $W : c_o \times c_i \times k_h \times k_w$
• Output $Y : c_o \times m_h \times m_w$

$Y_{i,,,} = X \star W_{i,,,}$
for $i = 1, \ldots, c_o$
Multiple Input/Output Channels

- Each 3-D kernel may recognize a particular pattern
Multiple Input/Output Channels

- Each 3-D kernel may recognize a particular pattern (Gabor filters)
Q3. Suppose we want to perform convolution on an RGB image of size 224x224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. Which is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. 64 x 3 x 3 x 222 x 222
B. 64 x 3 x 3 x 222
C. 3 x 3 x 222 x 222
D. 64 x 3 x 3 x 3 x 222 x 222
Q3. Suppose we want to perform convolution on an RGB image of size 224x224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. Which is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. $64 \times 3 \times 3 \times 222 \times 222$

B. $64 \times 3 \times 3 \times 222$

C. $3 \times 3 \times 222 \times 222$

D. $64 \times 3 \times 3 \times 3 \times 222 \times 222$
Q3. Suppose we want to perform convolution on an RGB image of size 224x224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. Which is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. $64 \times 3 \times 3 \times 222 \times 222$
B. $64 \times 3 \times 3 \times 222$
C. $3 \times 3 \times 222 \times 222$
D. $64 \times 3 \times 3 \times 3 \times 222 \times 222$

For each kernel, we slide the window to 222 x 222 different locations. For each location, the number of multiplication is 3x3x3. So in total $64 \times 3 \times 3 \times 3 \times 222 \times 222$. 
Q4. Suppose we want to perform convolution on a RGB image of size 224 x 224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. 64 x 222 x 222

B. 64 x 3 x 3 x 222

C. 3 x 3 x 3 x 64

D. (3 x 3 x 3 + 1) x 64
Q4. Suppose we want to perform convolution on a RGB image of size 224 x 224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. $64 \times 222 \times 222$
B. $64 \times 3 \times 3 \times 222$
C. $3 \times 3 \times 3 \times 64$
D. $(3 \times 3 \times 3 + 1) \times 64$
Q4. Suppose we want to perform convolution on a RGB image of size 224 x 224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. 64 x 222 x 222
B. 64 x 3 x 3 x 222
C. 3 x 3 x 3 x 64
D. (3 x 3 x 3 + 1) x 64

Each kernel is 3D kernel across 3 input channels, so has 3x3x3 parameters. Each kernel has 1 bias parameter. So in total (3x3x3+1)x64
Pooling Layer
Pooling

Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

Slides Credit: Deep Learning Tutorial by Marc’Aurelio Ranzato
Pooling

By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

Slides Credit: Deep Learning Tutorial by Marc’Aurelio Ranzato
2-D Max Pooling

• Returns the maximal value in the sliding window

\[
\text{max}(0, 1, 3, 4) = 4
\]
2-D Max Pooling

- Returns the maximal value in the sliding window

Input

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Output

max(0,1,3,4) = 4
2-D Max Pooling

- Returns the maximal value in the sliding window

\[
\text{max}(0, 1, 3, 4) = 4
\]

Input

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

Output

\[
\begin{array}{cc}
4 & 5 \\
7 & 8 \\
\end{array}
\]
Padding, Stride, and Multiple Channels

- Pooling layers have similar padding and stride as convolutional layers
- No learnable parameters
- Apply pooling for each input channel to obtain the corresponding output channel

\#output channels = \#input channels
Padding, Stride, and Multiple Channels

• Pooling layers have similar padding and stride as convolutional layers
• No learnable parameters
• Apply pooling for each input channel to obtain the corresponding output channel

#output channels = #input channels
Padding, Stride, and Multiple Channels

- Pooling layers have similar padding and stride as convolutional layers
- No learnable parameters
- Apply pooling for each input channel to obtain the corresponding output channel

#output channels = #input channels
Average Pooling

- Max pooling: the strongest pattern signal in a window
- Average pooling: replace max with mean in max pooling
  - The average signal strength in a window
Q5. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

A. 

<table>
<thead>
<tr>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>90</td>
</tr>
</tbody>
</table>

B. 

<table>
<thead>
<tr>
<th>16</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

C. 

<table>
<thead>
<tr>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

D. 

<table>
<thead>
<tr>
<th>12</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>5</td>
</tr>
</tbody>
</table>
Q5. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

A. 

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>70</td>
<td>90</td>
</tr>
</tbody>
</table>

B. 

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

C. 

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

D. 

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
</tr>
</tbody>
</table>
Q6. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>20</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>70</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>90</td>
<td>3</td>
</tr>
</tbody>
</table>

A.  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>70</td>
<td>90</td>
</tr>
</tbody>
</table>

B.  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

C.  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

D.  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
</tr>
</tbody>
</table>
Q6. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

A. 

B. 

C. 

D. 

![Image of the 2 x 2 max pooling output]
Summary
Summary

• Intro of convolutional computations
  • 2D convolution
  • Padding, stride
  • Multiple input and output channels
  • Pooling
Acknowledgement:

Some of the slides in these lectures have been adapted from materials developed by Alex Smola and Mu Li: