Topics:
  – Convolutional Neural Networks
    – Stride, padding
    – Pooling layers
    – Fully-connected layers as convolutions
Administrativia

• HW1 Reminder
  – Due: 10/02, 11:55pm
Recap from last time
Jacobian of ReLU

\[ g(x) = \max(0, x) \] (elementwise)

4096-d input vector \( \mathbf{h} \in \mathbb{R}^{4096} \)

4096-d output vector \( \mathbf{h}^o \in \mathbb{R}^{4096} \)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Jacobian of ReLU

4096-d input vector

\[ g(x) = \max(0, x) \] (elementwise)

4096-d output vector

Q: what is the size of the Jacobian matrix?

\[
\frac{\partial h_i}{\partial h_j} = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{otherwise}
\end{cases}
\]

\[ 4096 \times 4096 \]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Jacobians of FC-Layer
Jacobians of FC-Layer
Jacobians of FC-Layer
Convolutional Neural Networks
(without the brain stuff)
- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Example: 200x200 image
40K hidden units
~2B parameters!!!

Slide Credit: Marc'Aurelio Ranzato
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Locally Connected Layer

Slide Credit: Marc'Aurelio Ranzato
Locally Connected Layer

Statistics is similar at different locations
Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels
Convolutions!

math $\rightarrow$ CS $\rightarrow$ programming
Convolutions for mathematicians

\[ x(t) \quad y(t) \quad w(t) \]

\[ y(t) = (x \ast w)(t) \]

\[ = (w \ast x)(t) \]

\[ = \int x(t-a) w(a) \, da \]

\[ t = -\infty \]

\[ = \int x(a) w(t-a) \, da \]
Convolutions for mathematicians

\[ w(a) \rightarrow w(-a) \]

\[ w(-a) \rightarrow w(-(a-t)) \]

\[ \int_{a}^{b} x(t) w(a-t) \]

\[ = y(t) \]
\[ y(t_1, t_2) = \int \int x(t_1 - a, t_2 - b) \, da \, db \]
Convolutions for computer scientists

\[
 y[\tau, c] = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} x[\tau(-a), c-b] w[a, b]
\]
Convolutions for computer scientists
Convolutions for programmers
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer

(C) Dhruv Batra

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer
Mathieu et al. “Fast training of CNNs through FFTs” ICLR 2014
Plan for Today

• Convolutional Neural Networks
  – Stride, padding
  – Pooling layers
  – Fully-connected layers as convolutions
Convolution Explained

- [https://github.com/bruckner/deepViz](https://github.com/bruckner/deepViz)
Convolutional Layer

\[ x \ast \begin{pmatrix} 1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \]

activation

\[ y \]
Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters
FC vs Conv Layer

\[ h_i^l = \sum_{j=1}^{l-1} h_j^{l-1} \cdot w_{ij} + b_i \]

\[ h_i^l = \sum_{j=1}^{C_2} h_j^{l-1} \ast w_{ij} + b_i \]

\[ h_i^l [x, c] = \sum_{j=1}^{C_1} \sum_{a=0}^{k_1-1} \sum_{b=0}^{k_2-1} h_j^{l-1} [x+a, c+b] \cdot w[a, b] \]

\[ W \in \mathbb{R}^{(k_1 \times k_2 \times C_1) \times C_2} \]
FC vs Conv Layer
Convolution Layer

32x32x3 image -> preserve spatial structure
Convolution Layer

32x32x3 image

5x5x3 filter

**Convolve** the filter with the image i.e. “slide over the image spatially, computing dot products”

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Convolution Layer

32x32x3 image

5x5x3 filter

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Filters always extend the full depth of the input volume

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Convolution Layer

32x32x3 image

5x5x3 filter \( w \)

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. \( 5 \times 5 \times 3 = 75 \)-dimensional dot product + bias)

\[ w^T x + b \]
**Convolution Layer**

- **32x32x3 image**
- **5x5x3 filter**
- Convolve (slide) over all spatial locations
- **Activation map**

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Im2Col

Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
GEMM

Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
Time Distribution of AlexNet

CPU Forward Time Distribution

fc6 19.6%
relu1 1%
conv5 9.4%
conv4 14.7%
conv3 18.7%
norm2 0.6%
pool1 4%
pool2 4%
conv2 1.2%
relu2 0.4%

GPU Forward Time Distribution

fc7 0.8%
fc6 1.8%
conv5 17.7%
conv4 17.8%
relu3 0.2%
conv3 17.8%
conv2 21.9%
relu2 0.7%
pool2 0.5%
conv1 16.9%
relu1 0.7%
pool1 1%

Figure Credit: Yangqing Jia, PhD thesis
Convolution Layer

consider a second, green filter

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation maps

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions.
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Convolutional Neural Networks

**INPUT** 32x32

C1: feature maps 6@28x28

S2: f. maps 6@14x14

C3: f. maps 16@10x10

S4: f. maps 16@5x5

C5: layer 120

F6: layer 84

OUTPUT 10

Convolutions

Subsampling

Convolutions

Subsampling

Full connection

Gaussian connections

**Convolution layer**

**Sub-sampling layer**

**Fully connected MLP**

Image Credit: Yann LeCun, Kevin Murphy
preview:
Visualizations of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].

[Zeiler and Fergus 2013]

Low-level features → Mid-level features → High-level features → Linearly separable classifier

VGG-16 Conv1_1 → VGG-16 Conv3_2 → VGG-16 Conv5_3

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
example 5x5 filters (32 total)

one filter =>
one activation map
Visualizing Learned Filters

Figure Credit: [Zeiler & Fergus ECCV14]
Visualizing Learned Filters

Figure Credit: [Zeiler & Fergus ECCV14]
Visualizing Learned Filters

Figure Credit: [Zeiler & Fergus ECCV14]
A closer look at spatial dimensions:

32x32x3 image

5x5x3 filter

convolve (slide) over all spatial locations

activation map

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter

stride = 1

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter

=> 5x5 output
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied **with stride 2**

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied with stride 2
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied with stride 2
=> 3x3 output!

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied with stride 3?
A closer look at spatial dimensions:

7x7 input (spatially) assume 3x3 filter applied **with stride 3**?

doesn’t fit! cannot apply 3x3 filter on 7x7 input with stride 3.
\[(N - F) \times \text{stride} \]

\[\frac{N - F}{\text{stride}} + 1\]

Output size:

\[(N - F) / \text{stride} + 1\]

e.g. N = 7, F = 3:
strides 1 => \((7 - 3)/1 + 1 = 5\)
strides 2 => \((7 - 3)/2 + 1 = 3\)
strides 3 => \((7 - 3)/3 + 1 = 2.33\)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
In practice: Common to zero pad the border

e.g. input 7x7
3x3 filter, applied with \texttt{stride 1}

pad with 1 pixel border => what is the output?

(recall:)
\[
\frac{(N - F)}{\text{stride}} + 1
\]

\[
\frac{N - F - 1}{2} \geq \text{pad} + 1
\]

\[
\frac{F - 1}{2}
\]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
In practice: Common to zero pad the border

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

e.g. input 7x7
3x3 filter, applied with **stride 1**
**pad with 1 pixel** border => what is the output?

7x7 output!
In practice: Common to zero pad the border

e.g. input 7x7
3x3 filter, applied with **stride 1**
**pad with 1 pixel** border => what is the output?

7x7 output!
in general, common to see CONV layers with
stride 1, filters of size FxF, and zero-padding with
\((F-1)/2\). (will preserve size spatially)
e.g. \(F = 3\) => zero pad with 1
\(F = 5\) => zero pad with 2
\(F = 7\) => zero pad with 3

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Remember back to…
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn’t work well.
Examples time:

Input volume: \(32 \times 32 \times 3\)  
10 5x5 filters with stride 1, pad 2

Output volume size: ?

\[
\frac{(32)}{N_1^1} \times \frac{(32)}{N_2^2} \times \frac{(10)}{C_2}
\]
Examples time:

Input volume: $32 \times 32 \times 3$
10 5x5 filters with stride 1, pad 2

Output volume size:
$\frac{32+2 \times 2-5}{1} + 1 = 32$ spatially, so
$32 \times 32 \times 10$
Examples time:

Input volume: $32 \times 32 \times 3$

10,5x5 filters with stride 1, pad 2

Number of parameters in this layer?

$$\left( \frac{5 \times 5 \times 3 + 1}{10} \times 10 = 760 \right)$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Examples time:

Input volume: \(32 \times 32 \times 3\)
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has \(5 \times 5 \times 3 + 1 = 76\) params (+1 for bias)
=> \(76 \times 10 = 760\)
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and $K$ biases.
- In the output volume, the $d$-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the $d$-th filter over the input volume with a stride of $S$, and then offset by $d$-th bias.
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and $K$ biases.
- In the output volume, the $d$-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the $d$-th filter over the input volume with a stride of $S$, and then offset by $d$-th bias.

Common settings:

- $F = 3$, $S = 1$, $P = 1$
- $F = 5$, $S = 1$, $P = 2$
- $F = 5$, $S = 2$, $P = ?$ (whatever fits)
- $F = 1$, $S = 1$, $P = 0$