CS540  Introduction to Artificial Intelligence
Convolutional Neural Networks (II)
Josiah Hanna
University of Wisconsin-Madison

November 4, 2021

Slides created by Sharon Li [modified by Josiah Hanna]
## Announcements

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Slides</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, Nov 2</td>
<td>Machine Learning: Deep Learning I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday, Nov 4</td>
<td>Machine Learning: Deep Learning II</td>
<td></td>
<td>HW 6 Due; HW 7 Released</td>
</tr>
<tr>
<td>Tuesday, Nov 9</td>
<td>Machine Learning: Deep Learning III</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday, Nov 11</td>
<td>Machine Learning: Deep Learning and Neural Network's Summary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday, Nov 16</td>
<td>Search I: Un-Informed search</td>
<td></td>
<td>HW 7 Due; HW 8 Released</td>
</tr>
<tr>
<td>Thursday, Nov 18</td>
<td>Search II: Informed search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday, Nov 23</td>
<td>Games - Part I</td>
<td></td>
<td>HW 8 Due; HW 9 Released</td>
</tr>
<tr>
<td>Thursday, Nov 25</td>
<td>Happy Thanksgiving!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Everything below here is tentative and subject to change.*
Outline
Outline

• Brief review of convolutional computations
Outline

• Brief review of convolutional computations
• Convolutional Neural Networks
Outline

• Brief review of convolutional computations
• Convolutional Neural Networks
  • LeNet (first conv nets)
Outline

• Brief review of convolutional computations
• Convolutional Neural Networks
  • LeNet (first conv nets)
  • AlexNet
How to classify
Cats vs. dogs?
How to classify Cats vs. dogs?
How to classify Cats vs. dogs?
How to classify Cats vs. dogs?

36M floats in a RGB image!
Fully Connected Networks

Cats vs. dogs?
Cats vs. dogs?

Fully Connected Networks

Input

Hidden layer
100 neurons

Output
Cats vs. dogs?

Fully Connected Networks

Input

Hidden layer
100 neurons

Output

36M elements x 100 = 3.6B parameters!

Cat and dog images are used to illustrate the concept.
Review: 2-D Convolution
Review: 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 1 2</td>
<td>19 25</td>
</tr>
<tr>
<td>3 4 5</td>
<td>2 3 3 6</td>
<td>37 43</td>
</tr>
<tr>
<td>6 7 8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Review: 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>

\[
0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19, \\
1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25, \\
3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37, \\
4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.
\]
Review: 2-D Convolution

\[
\begin{align*}
0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 &= 19, \\
1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 &= 25, \\
3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 &= 37, \\
4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 &= 43.
\end{align*}
\]
Review: Multiple Input Channels

• Input and kernel can be 3D, e.g., an RGB image have 3 channels

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

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels

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

• Input and kernel can be 3D, e.g., an RGB image have 3 channels

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

\[
\begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
1 & 2 \\
3 & 4 \\
\end{array}
\end{array}
= \sum \begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\end{array}\begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
1 & 2 \\
3 & 4 \\
\end{array}
\end{array} + \sum \begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\end{array}\begin{array}{c}
\text{Input} \\
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\end{array}
\begin{array}{c}
\text{Kernel} \\
\begin{array}{ccc}
1 & 2 \\
3 & 4 \\
\end{array}
\end{array} = 56
\]
Review: Multiple Input Channels

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels

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

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels

\[
\begin{bmatrix}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{bmatrix}
\times
\begin{bmatrix}
0 & 1 \\
2 & 3 \\
\end{bmatrix}
= 
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\times
\begin{bmatrix}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{bmatrix}
+ 
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix}
\times
\begin{bmatrix}
56 & 72 \\
104 & 120 \\
\end{bmatrix}
\]
Review: Multiple Input Channels

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels

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

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels

![Diagram of multiple input channels with corresponding kernels and convolution process]
Review: Multiple Input Channels

• Input and kernel can be 3D, e.g., an RGB image have 3 channels
• Have a kernel for each channel, and then sum results over channels
Review: Multiple Input Channels

• Input and kernel can be 3D, e.g., an RGB image have 3 channels

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

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

Kernel/filter size

Input size  Pad  Stride
Consider a convolution layer with 16 filters. Each filter has a size of 11x11x3, a stride of 2x2. Given an input image of size 22x22x3, if we don’t allow a filter to fall outside of the input, what is the output size?

A. 11 x 11 x 16
B. 6 x 6 x 16
C. 7 x 7 x16
D. 5 x 5 x16
Consider a convolution layer with 16 filters. Each filter has a size of 11x11x3, a stride of 2x2. Given an input image of size 22x22x3, if we don’t allow a filter to fall outside of the input, what is the output size?

A. 11 x 11 x 16
B. 6 x 6 x 16
C. 7 x 7 x 16
D. 5 x 5 x 16

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

0 because filter not outside of input
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?
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

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

- Returns the maximal value in the sliding window

\[
\max(0, 1, 3, 4) = 4
\]
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
How to train a neural network?

Loss function: \[
\frac{1}{|D|} \sum_{i} \ell(x_i, y_i)
\]
How to train a neural network?

Loss function: \[ \frac{1}{|D|} \sum_i \ell(x_i, y_i) \]

Per-sample loss:

\[ \ell(x, y) = \sum_{j=1}^{K} -y_j \log p_j \]
How to train a neural network?

Loss function: \( \frac{1}{|D|} \sum_i \ell(x_i, y_i) \)

Per-sample loss:

\[ \ell(x, y) = \sum_{j=1}^{K} - y_j \log p_j \]

Also known as cross-entropy loss or softmax loss
How to train a convolutional neural network?
How to train a convolutional neural network?

Input

\[ p_i(x) = \frac{\exp(f_i(x))}{\sum_{j=1}^{N} \exp(f_j(x))}, \quad \text{softmax} \]
Recall Softmax

Turns outputs $f$ into probabilities (sum up to 1 across $k$ classes)

$$e^{z_i} \over \sum_{j=1}^{K} e^{z_j}$$
Recall Softmax

Turns outputs $f$ into probabilities (sum up to 1 across k classes)

Output layer

$\begin{bmatrix}
1.3 \\
5.1 \\
2.2 \\
0.7 \\
1.1
\end{bmatrix}$

Softmax activation function

$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

Probabilities

$\begin{bmatrix}
0.02 \\
0.90 \\
0.05 \\
0.01 \\
0.02
\end{bmatrix}$

Normalized
Cross-Entropy Loss

\[ L_{CE} = \sum_i - Y_i \log(p_i) \]

Goal: push \( p \) and \( Y \) to be identical
Cross-Entropy Loss

\[ L_{CE} = \sum_{i} - Y_i \log(p_i) \]

\[ = - \log(0.8) \]

Goal: push \( p \) and \( Y \) to be identical.
**Cross-Entropy Loss**

Convolutional layers

\[
L_{CE} = \sum_i - Y_i \log(p_i)
\]

\[
= - \log(0.8)
\]

**Goal**: push \( p \) and \( Y \) to be identical
Cross-Entropy Loss

\[ L_{CE} = \sum_{i} - Y_i \log(p_i) \]

\[ = - \log(0.8) \]

Goal: push \( p \) and \( Y \) to be identical
Convolutional Neural Networks
Evolution of neural net architectures
Evolution of neural net architectures

- LeNet
- AlexNet
- Inception Net
- ResNet
- DenseNet
LeNet Architecture
(first conv nets)
Handwritten Digit Recognition
MNIST

- Centered and scaled
- 50,000 training data
- 10,000 test data
- 28 x 28 images
- 10 classes
Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998
Gradient-based learning applied to document recognition
Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998
Gradient-based learning applied to document recognition
LeNet Architecture

Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner
def __init__(self):
    super(LeNet5, self).__init__()
    # Convolution (In LeNet-5, 32x32 images are given as input. Hence padding of 2 is done below)
    self.conv1 = torch.nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1, padding=2, bias=True)
    # Max-pooling
    self.max_pool_1 = torch.nn.MaxPool2d(kernel_size=2)
    # Convolution
    self.conv2 = torch.nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1, padding=0, bias=True)
    # Max-pooling
    self.max_pool_2 = torch.nn.MaxPool2d(kernel_size=2)
    # Fully connected layer
    self.fc1 = torch.nn.Linear(16*5*5, 120)  # convert matrix with 16*5*5 (= 400) features to a matrix of 120 features (columns)
    self.fc2 = torch.nn.Linear(120, 84)  # convert matrix with 120 features to a matrix of 84 features (columns)
    self.fc3 = torch.nn.Linear(84, 10)  # convert matrix with 84 features to a matrix of 10 features (columns)

https://github.com/bollakarthikeya/LeNet-5-PyTorch/blob/master/lenet5_gpu.py
def forward(self, x):
    # convolve, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.conv1(x))
    # max-pooling with 2x2 grid
    x = self.max_pool_1(x)
    # convolve, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.conv2(x))
    # max-pooling with 2x2 grid
    x = self.max_pool_2(x)
    # first flatten 'max_pool_2_out' to contain 16*5*5 columns
    # read through https://stackoverflow.com/a/42482819/7551231
    x = x.view(-1, 16*5*5)
    # FC-1, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.fc1(x))
    # FC-2, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.fc2(x))
    # FC-3
    x = self.fc3(x)

    return x
Let’s walk through an example using PyTorch

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
AlexNet
AlexNet

- AlexNet won ImageNet competition in 2012
AlexNet

- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
AlexNet

- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision
AlexNet

- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision
AlexNet Architecture

**AlexNet**
- 3x3 MaxPool, stride 2
- 11x11 Conv (96), stride 4
- Image (3x224x224)

**LeNet**
- 2x2 AvgPool, stride 2
- 5x5 Conv (6), pad 2
- Image (32x32)
AlexNet Architecture

Larger pool size

AlexNet

3x3 MaxPool, stride 2

11x11 Conv (96), stride 4

image (3x224x224)

LeNet

2x2 AvgPool, stride 2

5x5 Conv (6), pad 2

image (32x32)
AlexNet Architecture

Larger pool size

Larger kernel size, stride because of the increased image size, and more output channels.

AlexNet

- 3x3 MaxPool, stride 2
- 11x11 Conv (96), stride 4
- image (3x224x224)

LeNet

- 2x2 AvgPool, stride 2
- 5x5 Conv (6), pad 2
- image (32x32)
AlexNet Architecture

AlexNet

3x3 MaxPool, stride 2

3x3 Conv (384), pad 1

3x3 Conv (384), pad 1

3x3 Conv (384), pad 1

3x3 MaxPooling, stride 2

5x5 Conv (256), pad 2

LeNet

2x2 AvgPool, stride 2

5x5 Conv (16)
AlexNet Architecture

AlexNet

3x3 MaxPool, stride 2

3x3 Conv (384), pad 1

3x3 Conv (384), pad 1

3x3 MaxPooling, stride 2

5x5 Conv (256), pad 2

3 additional convolutional layers

LeNet

2x2 AvgPool, stride 2

5x5 Conv (16)
AlexNet Architecture

**AlexNet**

- 3x3 MaxPool, stride 2
- 3x3 Conv (384), pad 1
- 3x3 Conv (384), pad 1
- 3x3 MaxPooling, stride 2
- 5x5 Conv (256), pad 2

**LeNet**

- 2x2 AvgPool, stride 2
- 5x5 Conv (16)

**Notes:**

- 3 additional convolutional layers
- More output channels.
AlexNet Architecture

AlexNet

- Dense (4096)
- Dense (1000)

LeNet

- Dense (120)
- Dense (84)
- Dense (10)
AlexNet Architecture

1000 classes output

AlexNet
- Dense (1000)
- Dense (4096)
- Dense (4096)

LeNet
- Dense (10)
- Dense (84)
- Dense (120)
AlexNet Architecture

1000 classes output

Increase hidden size from 120 to 4096

AlexNet:
- Dense (4096)
- Dense (4096)
- Dense (1000)

LeNet:
- Dense (120)
- Dense (84)
- Dense (10)
More Differences…

• Change activation function from sigmoid to ReLu (no more vanishing gradient)
More Differences…

• Change activation function from sigmoid to ReLu (no more vanishing gradient)
More Differences...

- Change activation function from sigmoid to ReLu (no more vanishing gradient)
More Differences…

• Change activation function from sigmoid to ReLu (no more vanishing gradient)
• Data augmentation
## Complexity

<table>
<thead>
<tr>
<th></th>
<th>#parameters</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AlexNet</td>
<td>LeNet</td>
<td></td>
</tr>
<tr>
<td>Conv1</td>
<td>35K</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>Conv2</td>
<td>614K</td>
<td>2.4K</td>
<td></td>
</tr>
<tr>
<td>Conv3-5</td>
<td>3M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense1</td>
<td>26M</td>
<td>0.048M</td>
<td></td>
</tr>
<tr>
<td>Dense2</td>
<td>16M</td>
<td>0.01M</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>46M</td>
<td>0.06M</td>
<td></td>
</tr>
<tr>
<td>Increase</td>
<td>11x</td>
<td>1x</td>
<td></td>
</tr>
</tbody>
</table>

- **AlexNet**: 
  - Conv1: 35K
  - Conv2: 614K
  - Conv3-5: 3M
  - Dense1: 26M
  - Dense2: 16M
  - Total: 46M

- **LeNet**: 
  - Conv1: 150
  - Conv2: 2.4K
  - Conv3-5: 3M
  - Dense1: 0.048M
  - Dense2: 0.01M
  - Total: 0.06M

**Increase**: 11x for AlexNet, 1x for LeNet
# Complexity

<table>
<thead>
<tr>
<th></th>
<th>#parameters</th>
<th>AlexNet</th>
<th>LeNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td></td>
<td>35K</td>
<td>150</td>
</tr>
<tr>
<td>Conv2</td>
<td></td>
<td>614K</td>
<td>2.4K</td>
</tr>
<tr>
<td>Conv3-5</td>
<td></td>
<td>3M</td>
<td></td>
</tr>
<tr>
<td>Dense1</td>
<td></td>
<td>26M</td>
<td>0.048M</td>
</tr>
<tr>
<td>Dense2</td>
<td></td>
<td>16M</td>
<td>0.01M</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>46M</td>
<td>0.06M</td>
</tr>
<tr>
<td>Increase</td>
<td></td>
<td>11x</td>
<td>1x</td>
</tr>
</tbody>
</table>
## Complexity

<table>
<thead>
<tr>
<th></th>
<th>AlexNet</th>
<th>LeNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td>35K</td>
<td>150</td>
</tr>
<tr>
<td>Conv2</td>
<td>614K</td>
<td>2.4K</td>
</tr>
<tr>
<td>Conv3-5</td>
<td>3M</td>
<td></td>
</tr>
<tr>
<td>Dense1</td>
<td>26M</td>
<td>0.048M</td>
</tr>
<tr>
<td>Dense2</td>
<td>16M</td>
<td>0.01M</td>
</tr>
<tr>
<td>Total</td>
<td>46M</td>
<td>0.06M</td>
</tr>
<tr>
<td>Increase</td>
<td>11x</td>
<td>1x</td>
</tr>
</tbody>
</table>

$11 \times 11 \times 3 \times 96 = 35k$
ImageNet Top-5 Classification Accuracy (%)

- ILSVRC’10: 28.2%
- ILSVRC’11: 25.8%
- ILSVRC’12: 8 layers (AlexNet)
- ILSVRC’13: 8 layers
- ILSVRC’14: 19 layers (GoogleNet)
- ILSVRC’14: 22 layers (VGG)
- ILSVRC’14: 6.7%
Which of the following are true about AlexNet? Select all that apply.

A. AlexNet contains 8 layers. The first five are convolutional layers.
B. The last three layers are fully connected layers.
C. Some of the convolutional layers are followed by max-pooling (layers).
D. AlexNet achieved excellent performance in the 2012 ImageNet challenge.

Which of the following are true about AlexNet? Select all that apply.

A. AlexNet contains 8 layers. The first five are convolutional layers.
B. The last three layers are fully connected layers.
C. Some of the convolutional layers are followed by max-pooling (layers).
D. AlexNet achieved excellent performance in the 2012 ImageNet challenge.

All options are true!

VGG
Progress

- LeNet (1995)
  - 2 convolution + pooling layers
  - 2 hidden dense layers
- AlexNet
  - Bigger and deeper LeNet
  - ReLu, preprocessing
- VGG
  - Bigger and deeper AlexNet (repeated VGG blocks)
Which of the following statement is True for the success of deep models?

- Better design of the neural networks
- Large scale training dataset
- Available computing power
- All of the above
Which of the following statement is True for the success of deep models?

• Better design of the neural networks
• Large scale training dataset
• Available computing power
• All of the above
What we’ve learned today
What we’ve learned today

• Brief review of convolutional computations
What we’ve learned today

• Brief review of convolutional computations
• Convolutional Neural Networks
What we’ve learned today

• Brief review of convolutional computations
• Convolutional Neural Networks
  • LeNet (first conv nets)
What we’ve learned today

• Brief review of convolutional computations
• Convolutional Neural Networks
  • LeNet (first conv nets)
  • AlexNet
What we’ve learned today

• Brief review of convolutional computations
• Convolutional Neural Networks
  • LeNet (first conv nets)
  • AlexNet
• PyTorch demo
Acknowledgement:
Some of the slides in these lectures have been adapted/borrowed from materials developed by Yin Li (https://happyharrycn.github.io/CS540-Fall20/schedule/), Alex Smola and Mu Li: https://courses.d2l.ai/berkeley-stat-157/index.html