CS 4803 / 7643: Deep Learning

Topics:
- (Finish) Computing Gradients
  - Backprop in Conv Layers
- Forward mode vs Reverse mode AD
- Modern CNN Architectures

Zsolt Kira
Georgia Tech
Case Studies

- There are several generations of ConvNets
  - 2012 – 2014: AlexNet, ZNet, VGGNet
    - Conv-Relu, Pooling, Fully connected, Softmax
    - Deeper ones (VGGNet) tend to do better
  - 2014
    - Fully-convolutional networks for **semantic segmentation**
    - Matrix outputs rather than just one probability distribution
  - 2014-2016
    - Fully-convolutional networks for **classification**
    - Less parameters, faster than comparable Gen1 networks
    - GoogleNet, ResNet
  - 2014-2016
    - Detection layers (proposals)
    - Caption generation (combine with RNNs for language)
Case Study: AlexNet
[Krizhevsky et al. 2012]

Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Q: what is the output volume size? Hint: \((227-11)/4+1 = 55\)
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume [55x55x96]

Q: What is the total number of parameters in this layer?
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images

**First layer (CONV1):** 96 11x11 filters applied at stride 4

=>

Output volume **[55x55x96]**  
Parameters: $$(11*11*3)*96 = 35K$$
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96

**Second layer (POOL1):** 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!
Case Study: AlexNet

[Krizhevsky et al. 2012]

Input: 227x227x3 images
After CONV1: 55x55x96
After POOL1: 27x27x96

...
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)
Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10
  manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%
Case Study: ZFNet [Zeiler and Fergus, 2013]

AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 15.4% -> 14.8%
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013

7.3% top 5 error
### ImageNet Configuration

<table>
<thead>
<tr>
<th>Layer</th>
<th>Memory (MBytes)</th>
<th>Params (MParameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT:</td>
<td>224x224x3</td>
<td>150K, 0</td>
</tr>
<tr>
<td>CONV3-64: [224x224x64]</td>
<td>224K, 3.2M, (3<em>3</em>3)*64 = 1,728</td>
<td></td>
</tr>
<tr>
<td>CONV3-64: [224x224x64]</td>
<td>224K, 3.2M, (3<em>3</em>3)*64 = 1,728</td>
<td></td>
</tr>
<tr>
<td>POOL2: [112x112x64]</td>
<td>112K, 800K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-128: [112x112x128]</td>
<td>112K, 1.6M, (3<em>3</em>64)*128 = 73,728</td>
<td></td>
</tr>
<tr>
<td>CONV3-128: [112x112x128]</td>
<td>112K, 1.6M, (3<em>3</em>128)*128 = 147,456</td>
<td></td>
</tr>
<tr>
<td>POOL2: [56x56x128]</td>
<td>56K, 400K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-256: [56x56x256]</td>
<td>56K, 800K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-256: [56x56x256]</td>
<td>56K, 800K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-256: [56x56x256]</td>
<td>56K, 800K, 0</td>
<td></td>
</tr>
<tr>
<td>POOL2: [28x28x256]</td>
<td>28K, 200K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-512: [28x28x512]</td>
<td>28K, 400K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-512: [28x28x512]</td>
<td>28K, 400K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-512: [28x28x512]</td>
<td>28K, 400K, 0</td>
<td></td>
</tr>
<tr>
<td>POOL2: [14x14x512]</td>
<td>14K, 100K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-512: [14x14x512]</td>
<td>14K, 100K, 0</td>
<td></td>
</tr>
<tr>
<td>CONV3-512: [14x14x512]</td>
<td>14K, 100K, 0</td>
<td></td>
</tr>
<tr>
<td>POOL2: [7x7x512]</td>
<td>7K, 25K, 0</td>
<td></td>
</tr>
<tr>
<td>FC: 4096</td>
<td>4096, 0</td>
<td></td>
</tr>
<tr>
<td>FC: 4096</td>
<td>4096, 0</td>
<td></td>
</tr>
<tr>
<td>FC: 1000</td>
<td>1000, 4,096,000</td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL memory:** 24M * 4 bytes \(\approx\) 93MB / image (only forward! \(\approx\)*2 for bwd)

**TOTAL params:** 138M parameters
INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: \(224^2 \times 224 \times 64 = 3.2M\) params: \((3 \times 3 \times 3) \times 64 = 1,728\)
CONV3-64: [224x224x64] memory: \(224^2 \times 224 \times 64 = 3.2M\) params: \((3 \times 3 \times 64) \times 64 = 36,864\)
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: \((3 \times 3 \times 64) \times 128 = 73,728\)
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: \((3 \times 3 \times 128) \times 128 = 147,456\)
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: \((3 \times 3 \times 128) \times 256 = 294,912\)
CONV3-256: [56x56x256] memory: 56*56*256=800K params: \((3 \times 3 \times 256) \times 256 = 589,824\)
CONV3-256: [56x56x256] memory: 56*56*256=800K params: \((3 \times 3 \times 256) \times 256 = 589,824\)
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: \((3 \times 3 \times 256) \times 512 = 1,179,648\)
CONV3-512: [28x28x512] memory: 28*28*512=400K params: \((3 \times 3 \times 512) \times 512 = 2,359,296\)
CONV3-512: [28x28x512] memory: 28*28*512=400K params: \((3 \times 3 \times 512) \times 512 = 2,359,296\)
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: \((3 \times 3 \times 512) \times 512 = 2,359,296\)
CONV3-512: [14x14x512] memory: 14*14*512=100K params: \((3 \times 3 \times 512) \times 512 = 2,359,296\)
CONV3-512: [14x14x512] memory: 14*14*512=100K params: \((3 \times 3 \times 512) \times 512 = 2,359,296\)
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000
TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Note:
Most memory is in early CONV
Most params are in late FC
Case Study: GoogLeNet

[ Szegedy et al., 2014 ]

Inception module

ILSVRC 2014 winner (6.7% top 5 error)
image
Natural images are locally heavily correlated.
Filter activations reflect image correlations
Image correlations reflected in filter bank correlations

# of filters

image

filter bank
Correlations in natural images are multi-scale
Correlations in natural images are multi-scale

Going Deeper with Convolutions
C Szegedy et al (2014)
Replace convolution with multi-scale convolution

Going Deeper with Convolutions
C Szegedy et al (2014)
Employ multi-scale and dimensional reduction.
Summary of Inception architecture.

- Multi-scale architecture to mirror correlation structure in images.
- Dimensional reduction to constrain representation along each spatial scale.

Going Deeper with Convolutions
C Szegedy et al (2014)
An Aside

- Auxiliary classifier connected to intermediate layers
### Case Study: GoogLeNet

<table>
<thead>
<tr>
<th>type</th>
<th>patch size/stridel</th>
<th>output size</th>
<th>depth</th>
<th>#1 x 1 reduce</th>
<th>#3 x 3 reduce</th>
<th>#5 x 5 reduce</th>
<th>#5 x 5 pool proj</th>
<th>params</th>
<th>ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>convolution</td>
<td>7 x 7 / 2</td>
<td>112 x 112 x 64</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2.7K</td>
<td>34M</td>
</tr>
<tr>
<td>max pool</td>
<td>3 x 3 / 2</td>
<td>56 x 56 x 64</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>convolution</td>
<td>3 x 3 / 1</td>
<td>56 x 56 x 192</td>
<td>2</td>
<td>64</td>
<td>192</td>
<td></td>
<td></td>
<td>112K</td>
<td>360M</td>
</tr>
<tr>
<td>max pool</td>
<td>3 x 3 / 2</td>
<td>28 x 28 x 192</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inception (1a)</td>
<td>28 x 28 x 256</td>
<td>2</td>
<td>64</td>
<td>66</td>
<td>128</td>
<td>16</td>
<td>32</td>
<td>389K</td>
<td>304M</td>
</tr>
<tr>
<td>inception (1b)</td>
<td>28 x 28 x 480</td>
<td>2</td>
<td>128</td>
<td>128</td>
<td>192</td>
<td>32</td>
<td>96</td>
<td>150K</td>
<td>128M</td>
</tr>
<tr>
<td>max pool</td>
<td>3 x 3 / 2</td>
<td>14 x 14 x 480</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inception (4a)</td>
<td>14 x 14 x 512</td>
<td>2</td>
<td>192</td>
<td>96</td>
<td>208</td>
<td>16</td>
<td>48</td>
<td>384K</td>
<td>73M</td>
</tr>
<tr>
<td>inception (4b)</td>
<td>14 x 14 x 512</td>
<td>2</td>
<td>160</td>
<td>112</td>
<td>224</td>
<td>24</td>
<td>64</td>
<td>437K</td>
<td>82M</td>
</tr>
<tr>
<td>inception (4c)</td>
<td>14 x 14 x 512</td>
<td>2</td>
<td>132</td>
<td>128</td>
<td>256</td>
<td>24</td>
<td>64</td>
<td>450K</td>
<td>100M</td>
</tr>
<tr>
<td>inception (4d)</td>
<td>14 x 14 x 528</td>
<td>2</td>
<td>112</td>
<td>144</td>
<td>288</td>
<td>32</td>
<td>64</td>
<td>580K</td>
<td>119M</td>
</tr>
<tr>
<td>inception (4e)</td>
<td>14 x 14 x 832</td>
<td>2</td>
<td>256</td>
<td>160</td>
<td>320</td>
<td>32</td>
<td>128</td>
<td>840K</td>
<td>170M</td>
</tr>
<tr>
<td>max pool</td>
<td>3 x 3 / 2</td>
<td>7 x 7 x 832</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>477K</td>
<td>58M</td>
</tr>
<tr>
<td>inception (5a)</td>
<td>7 x 7 x 832</td>
<td>2</td>
<td>256</td>
<td>160</td>
<td>320</td>
<td>32</td>
<td>128</td>
<td>1072K</td>
<td>54M</td>
</tr>
<tr>
<td>inception (5b)</td>
<td>7 x 7 x 1024</td>
<td>2</td>
<td>384</td>
<td>192</td>
<td>384</td>
<td>48</td>
<td>128</td>
<td>1388K</td>
<td>71M</td>
</tr>
<tr>
<td>avg pool</td>
<td>7 x 7 / 1</td>
<td>1 x 1 x 1024</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dropout (40%)</td>
<td>1 x 1 x 1024</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linear</td>
<td>1 x 1 x 1000</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100K</td>
<td>1M</td>
</tr>
<tr>
<td>softmax</td>
<td>1 x 1 x 1000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fun features:**
- Only 5 million params! (Removes FC layers completely)

**Compared to AlexNet:**
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

- AlexNet: 60M params
- ZNet: 75M
- VGG: 138M
- GoogleNet: 5M
Case Study: ResNet  [He et al., 2015]
ILSVRC 2015 winner (3.6% top 5 error)

MSRA @ ILSVRC & COCO 2015 Competitions

• 1st places in all five main tracks
  • ImageNet Classification: “Ultra-deep” (quote Yann) 152-layer nets
  • ImageNet Detection: 16% better than 2nd
  • ImageNet Localization: 27% better than 2nd
  • COCO Detection: 11% better than 2nd
  • COCO Segmentation: 12% better than 2nd

*Improvements are relative numbers


Slide from Kaiming He’s recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w
Revolution of Depth

3.57

ILSVRC'15
ResNet

22 layers
6.7

ILSVRC'14
GoogleNet

19 layers
7.3

ILSVRC'14
VGG

11.7

ILSVRC'13
AlexNet

8 layers
8 layers

ILSVRC'12

ILSVRC'11

ILSVRC'10

ImageNet Classification top-5 error (%)

25.8
28.2

shallow

(slide from Kaiming He’s recent presentation)
Importance of Depth

• After a while, adding depth decreases performance
• At first, vanishing/exploding gradients
  • normalized initialization
  • Batch normalization
  • 2\textsuperscript{nd} order methods
• Then, optimization limitation
  – Deeper network should be able to mimic shallow ones
Case Study: ResNet [He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)

VGG, 19 layers (ILSVRC 2014)

ResNet, 152 layers (ILSVRC 2015)

(slide from Kaiming He’s recent presentation)
Case Study: ResNet

[He et al., 2015]

spatial dimension only 56x56!
Case Study: ResNet [He et al., 2015]

- Plain net
  - $x$ → weight layer → relu → weight layer → relu → $H(x)$

- Residual net
  - $x$ → weight layer → relu → weight layer → $F(x)$ → relu → $H(x) = F(x) + x$
Case Study: ResNet  [He et al., 2015]

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used
Case Study: ResNet [He et al., 2015]

all-3x3  

similar complexity  

bottleneck  
(for ResNet-50/101/152)
Case Study: ResNet

[He et al., 2015]

(this trick is also used in GoogLeNet)
Case Study: ResNet [He et al., 2015]

<table>
<thead>
<tr>
<th>layer name</th>
<th>output size</th>
<th>18-layer</th>
<th>34-layer</th>
<th>50-layer</th>
<th>101-layer</th>
<th>152-layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>112×112</td>
<td>7×7, 64, stride 2</td>
<td>3×3 max pool, stride 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv2_x</td>
<td>56×56</td>
<td>[3×3, 64] x2</td>
<td>[3×3, 64] x3</td>
<td>[1×1, 64]</td>
<td>[1×1, 64]</td>
<td>[1×1, 64]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 64]</td>
<td>[3×3, 64]</td>
<td>[1×1, 256]</td>
<td>[1×1, 256]</td>
<td>[1×1, 256]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 64]</td>
<td>[3×3, 64]</td>
<td>[1×1, 256]</td>
<td>[1×1, 256]</td>
<td>[1×1, 256]</td>
</tr>
<tr>
<td>conv3_x</td>
<td>28×28</td>
<td>[3×3, 128] x2</td>
<td>[3×3, 128] x4</td>
<td>[1×1, 128]</td>
<td>[1×1, 128]</td>
<td>[1×1, 128]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 128]</td>
<td>[3×3, 128]</td>
<td>[1×1, 512]</td>
<td>[1×1, 512]</td>
<td>[1×1, 512]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 128]</td>
<td>[3×3, 128]</td>
<td>[1×1, 512]</td>
<td>[1×1, 512]</td>
<td>[1×1, 512]</td>
</tr>
<tr>
<td>conv4_x</td>
<td>14×14</td>
<td>[3×3, 256] x2</td>
<td>[3×3, 256] x6</td>
<td>[1×1, 256]</td>
<td>[1×1, 256]</td>
<td>[1×1, 256]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 256]</td>
<td>[3×3, 256]</td>
<td>[1×1, 1024]</td>
<td>[1×1, 1024]</td>
<td>[1×1, 1024]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 256]</td>
<td>[3×3, 256]</td>
<td>[1×1, 1024]</td>
<td>[1×1, 1024]</td>
<td>[1×1, 1024]</td>
</tr>
<tr>
<td>conv5_x</td>
<td>7×7</td>
<td>[3×3, 512] x2</td>
<td>[3×3, 512] x3</td>
<td>[1×1, 512]</td>
<td>[1×1, 512]</td>
<td>[1×1, 512]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 512]</td>
<td>[3×3, 512]</td>
<td>[1×1, 2048]</td>
<td>[1×1, 2048]</td>
<td>[1×1, 2048]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3×3, 512]</td>
<td>[3×3, 512]</td>
<td>[1×1, 2048]</td>
<td>[1×1, 2048]</td>
<td>[1×1, 2048]</td>
</tr>
<tr>
<td>1×1</td>
<td>average pool, 1000-d fc, softmax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOPs</td>
<td>1.8×10^9</td>
<td>3.6×10^9</td>
<td>3.8×10^9</td>
<td>7.6×10^9</td>
<td>11.3×10^9</td>
<td></td>
</tr>
</tbody>
</table>
Localization and Detection

Results from Faster R-CNN, Ren et al 2015
Computer Vision Tasks

- **Classification**: Single object (CAT)
- **Classification + Localization**: Single object (CAT)
- **Object Detection**: Multiple objects (CAT, DOG, DUCK)
- **Instance Segmentation**: Multiple objects (CAT, DOG, DUCK)
Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation
Classification + Localization

Classification: C classes
 Input: Image
 Output: Class label
 Evaluation metric: Accuracy

Localization:
 Input: Image
 Output: Box in the image (x, y, w, h)
 Evaluation metric: Intersection over Union

Classification + Localization: Do both
1000 classes (same as classification)
Each image has 1 class, at least one bounding box
~800 training images per class
Algorithm produces 5 (class, box) guesses
Example is correct if at least one guess has correct class AND bounding box at least 0.5 intersection over union (IoU)

<rizhevsky et. al. 2012>
Idea 1: Localization as Regression

**Input:** image

**Neural Net**

**Output:**
Box coordinates (4 numbers)

**Correct output:**
Box coordinates (4 numbers)

**Loss:**
L2 distance

Only one object, simpler than detection
Step 1: Train (or download) a classification model (AlexNet, VGG, GoogLeNet)
Step 2: Attach new fully-connected “regression head” to the network
Step 3: Train the regression head only with SGD and L2 loss
Step 4: At test time use both heads
Per-Class vs. Class Agnostic

Assume classification over C classes:

Classification head:
- C numbers
  (one per class)

Class agnostic:
- 4 numbers
  (one box)

Class specific:
- C x 4 numbers
  (one box per class)
Where to attach?
Multiple Objects

Want to localize exactly K objects in each image

(e.g. whole cat, cat head, cat left ear, cat right ear for K=4)

K x 4 numbers (one box per object)
Human Pose Estimation

Represent a person by K joints

Regress (x, y) for each joint from last fully-connected layer of AlexNet

(Details: Normalized coordinates, iterative refinement)