Visual Perception vs. Computational Perception

Image

Neural Network

High-level information
- Depth
- Motion
- Object classes
- Object poses
- Etc.
Mathematic Models

• Try to model the human brain with computational models, e.g., neural networks
Mathematic Models

- What is the form of the function $f(x)$?
  - No idea!
  - Concatenate simple functions (neurons)

\[
\begin{align*}
  & x \\
  \Rightarrow & f(x) \\
  \Rightarrow & y \in \{+1, -1\} \\
  \quad \text{Dog}
\end{align*}
\]
Neural Network: Concatenation of functions

Linear score function: \( f = Wx \)

2-layer Neural Network

\[
f = f_2(f_1(x)) = W_2 \max(0, W_1x)
\]

Non-linearity

\[
h = f_1(X) \\
s = f_2(h)
\]

Need to learn the weights!
Frank Rosenblatt’s Perceptron

Frank Rosenblatt (1928-1971)

Inputs \[ w_0, w_1, w_2, \ldots, w_m \]

Net input function

Activation function

\[ \sigma(w^T x + b) = \begin{cases} 1 & \text{if } w^T x + b \geq 0, \\ 0 & \text{otherwise.} \end{cases} \]
Activation Functions

2-layer Neural Network

\[ f = f_2(f_1(x)) = W_2 \max(0, W_1 x) \]

Rectified Linear Unit (ReLU)
\[ \max(0, x) \]

Introduce non-linearity to the network
Activation Functions

Sigmoid

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

ReLU

\[ \text{max}(0, x) \]

Leaky ReLU

\[ \text{max}(0.1x, x) \]

Maxout

\[ \text{max}(w_1^T x + b_1, w_2^T x + b_2) \]

tanh

\[ \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \]

ELU

\[ f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases} \]

Linear Unit
Fully Connected Layer

\[ y = Wx \]

\[ x : n \times 1 \quad W : m \times n \quad y : m \times 1 \]
Fully Connected Layer

• What is the drawback of only using fully connected layers?

\[ y = Wx \]

• Consider an image with 640 x 480
  • x is with dimension 307,200
  • The weight matrix of the fully connect layer is too large
Convolutional Layers

• Consist of convolutional filters

• Share weights among different image locations

\[ g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

Gaussian Filter

Learn the weights!
Convolutional Neural Networks

Input image → Convolutional layer (translation invariant) → ReLU layer → Pooling layer → Fully connected layer → Output vector
Convolutional Neural Networks

[LeNet-5, LeCun 1980]
Convolutional Layer

32x32x3 image

32 height

32 width

3 depth
Convolutional Layer

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

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

\[ w^T x + b \]
Convolutional Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
Convolutional Layer

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
Convolutional Layer

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
Convolutional Layer

A closer look at spatial dimensions:

- 7x7 input (spatially)
- Assume 3x3 filter, with stride 1
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1
Convolutional Layer

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 1

=> 5x5 output
Convolutional Layer

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter, with stride 2
Convolutional Layer

A closer look at spatial dimensions:

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

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

=> 3x3 output!
A closer look at spatial dimensions:

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

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

doesn’t fit!
cannot apply 3x3 filter on 7x7 input with stride 3.
Convolutional Layer

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

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

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

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

Convolutional Layer
### Convolutional Layer

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

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)

- \(F = 3\) => zero pad with 1
- \(F = 5\) => zero pad with 2
- \(F = 7\) => zero pad with 3

In practice: Common to zero pad the border
Convolutional Layer

32x32x3 image
5x5x3 filter

Convolve (slide) over all spatial locations

activation map
Convolutional Layer

consider a second, green filter

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation maps

32
32
32
32
32
32
32
32
32
32
32
Convolutional Layer

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!
**Preview:** ConvNet is a sequence of Convolution Layers, interspersed with activation functions.
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions.
Convolutional Neural Networks

- Input image
- Convolutional layer
- ReLU layer
- Pooling layer
- Fully connected layer
- Output vector
Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:
Max Pooling Layer

Single depth slice

max pool with 2x2 filters and stride 2
Training: back-propagate errors

Details in next lecture
Case Study: LeNet-5

[LeCun et al., 1998]

Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]
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: \((11 \times 11 \times 3) \times 96 = 35K\)
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: VGGNet  

[Simonyan and Zisserman, 2014]

**Only 3x3 CONV stride 1, pad 1**

and **2x2 MAX POOL stride 2**

11.2% top 5 error in ILSVRC 2013

->

7.3% top 5 error
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

INPUT: [224x224x3] memory: 224*224*3=150K params: 0

CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728

CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864

POOL2: [112x112x64] memory: 112*112*64=800K params: 0

CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728

CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456

POOL2: [56x56x128] memory: 56*56*128=400K params: 0

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824

POOL2: [28x28x256] memory: 28*28*256=200K params: 0

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296

POOL2: [14x14x512] memory: 14*14*512=100K params: 0

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*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

(not counting biases)
Case Study: GoogLeNet

[Szegedy et al., 2014]

Inception module

ILSVRC 2014 winner (6.7% top 5 error)
Case Study: GoogLeNet

Fun features:
- Only 5 million params!

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

LeNet (5 layers)

AlexNet (8 layers)

VGGNet (19 layers)

GoogleNet

ResNet (152 layers)
Case Study: ResNet [He et al., 2015]
Revolution of Depth

ImageNet Classification top-5 error (%)

ILSVRC'15 ResNet | 3.57
ILSVRC'14 GoogleNet | 6.7
ILSVRC'14 VGG | 7.3
ILSVRC'13 | 8 layers (11.7)
ILSVRC'12 AlexNet | 16.4
ILSVRC'11 | 25.8
ILSVRC'10 | 28.2

shallow

(slide from Kaiming He)
Further Reading

• Stanford CS231n, lecture 5, Convolutional Neural Networks
  http://cs231n.stanford.edu/schedule.html

• Deep learning with PyTorch
  https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

• AlexNet (2012):
  https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

