Fundamentals of Deep Learning

Raymond Ptucha,
Rochester Institute of Technology
NVIDIA, Deep Learning Institute

Electronic Imaging 2019: SC20
January 15, 2019, 8:30am-12:45pm

Fair Use Agreement

This agreement covers the use of all slides in this document, please read carefully.

• You may freely use these slides, if:
  – You send me an email telling me the conference/venue/company name in advance, and which slides you wish to use.
  – You receive a positive confirmation email back from me.
  – My name (R. Ptucha) appears on each slide you use.

(c) Raymond Ptucha, rwpec@rit.edu
Agenda

• Part I- Intuition and Theory
  – 8:35-9:15pm: Introduction
  – 9:15-10:00pm: Convolutional Neural Networks
  – 10:00-10:40pm: Recurrent Neural Networks
• 10:40-11:00pm: Break
• Part II- Hands on
  – 11:00am-12:45pm: Hands-on exercises

Two Most Important Deep Learning Fields

• Convolutional Neural Networks (CNN)
  – Examine high dimensional input, learn features and classifier simultaneously

• Recurrent Neural Networks (RNN)
  – Learn temporal signals, remember both short and long sequences
Two Most Important Deep Learning Fields

• Convolutional Neural Networks (CNN)
  – Examine high dimensional input, learn features and classifier simultaneously

• Recurrent Neural Networks (RNN)
  – Learn temporal signals, remember both short and long sequences

Fully Connected Layers?

Example:
• 200×200 pixel image.
• 40K input fully connected to 40K hidden (or output) layer.
• 1.6 billion weights!
• Generally don’t have enough training samples to learn that many weights.

Ranzato CVPR'14
Convolution Filter

- Convolution filters apply a transform to an image.
- The above filter detects vertical edges.

Locally Connected Layer

- 200×200 pixel image.
- 40K input.
- Four 10×10 filters, each fully connected.
- 40K×10×10×4=16M weights….getting better!
Locally Connected Layer

- 200×200 pixel image.
- 40K input.
- Four 10×10 filters, each fully connected
- 40K×10×10×4=16M weights….getting better!

- Can we formulate so each filter has similar statistics across all locations?

Convolution Layer

- 200×200 pixel image.
- 40K input.
- Four 10×10 filters, each fully connected
- 40K×10×10×4=16M weights….getting better!

- Require each filter has same statistics across all locations.
- Learn filters.
Convolution Layer

- 200×200 pixel image.
- 40K input.
- Four 10×10 filters, each fully connected
- 40K×10×10×4=16M weights….getting better!
- Require each filter has same statistics across all locations.
- Learn filters.
- To learn four filters we have 4×10×10=400 parameters- great!

Many Flavors of CNNs…

- LeNet-5, LeCun 1989
- AlexNet, Krizhevsky 2012
- VGGNet, Simonyan 2014
- GoogLeNet (Inception), Szegedy 2014
- ResNet, He 2015
- DenseNet, Huang 2017
Image Convolution

By padding \((\text{filterWidth}-1)/2\), output image size matches input image size.

3x3 filter sliding over input image

Max Pooling - Reducing the Size of an Image

Max pool with 2x2 filters and stride 2

In the diagram:

- The single depth slice represents an input image.
- The output image is shown after max pooling.
- The pooling operation reduces the size of the input image by selecting the maximum value within a 2x2 window and moving this window across the image with a stride of 2.

R. Ptucha '19
Convolution Neural Network (CNN) Building Block

Putting it All Together
Learning Filters

32 Learned Filters, each 5×5

32 Filtered images, each is 28×28

Input image
28×28

Use zero padding

Filters

3×3 filter

3×3 filter

3×3×4 filter
Learning Filters

- 32 Learned Filters, each 5×5×3
- 32 Filtered images, each 28×28×1

Input image
28×28×3

Use zero padding

CNN Architecture

(Not so) Toy Example

- Input RGB image: 64×64×3 pixels
- 32 filters, each filter is 5×5×3. 2 pixel pad added to top/bot/left/right so filtered image is same dimension as input image.
- 16 filters, each filter is 5×5×32. 2 pixel pad.
- 32 filters, each filter is 5×5×16. 2 pixel pad.
- 64 filters, each filter is 5×5×32. 2 pixel pad.
- 1×1×64, filter, 0 pixel pad.

Output: prediction of 1 of 10 categories

16 converted to 16 element vector
CNN Example

- Input [32x32x3]
  - CONV with ten 3x3 filters, stride 1, pad 1:
    - Parameters: \((3*3*3)*10 + 10 = 280\)
    - Memory: \(32*32*10\)
  - CONV with ten 3x3 filters, stride 1, pad 1:
    - Parameters: \((3*3*10)*10 + 10 = 910\)
    - Memory: \(32*32*10\)
  - Pool with 2x2 filters, stride 2:
    - Parameters: 0
    - Memory: \(16*16*10\)

Note: Two Conv's between each pool…
CNN Example

• CONV with ten 3x3 filters, stride 1, pad 1:
  • Parameters: \((3\times3\times10)\times10 + 10 = 910\)
  • Memory: \(16\times16\times10\)
• CONV with twenty 3x3 filters, stride 1, pad 1:
  • Parameters: \((3\times3\times10)\times20 + 20 = 1820\)
  • Memory: \(16\times16\times20\)
• Pool with 2x2 filters, stride 2:
  • Parameters: 0
  • Memory: \(8\times8\times20\)

CNN Example

• CONV with ten 3x3 filters, stride 1, pad 1:
  • Parameters: \((3\times3\times20)\times10 + 10 = 1810\)
  • Memory: \(8\times8\times10\)
• CONV with twenty 3x3 filters, stride 1, pad 1:
  • Parameters: \((3\times3\times10)\times20 + 20 = 1820\)
  • Memory: \(8\times8\times20\)
• Pool with 2x2 filters, stride 2:
  • Parameters: 0
  • Memory: \(4\times4\times20\)
CNN Example

- Fully connect (FC) 4x4x20 to 10 output classes
  - Parameters: $(4\times4\times20)\times10 + 10 = 3210$
  - Memory: 10
  - Done!

Case Study

**Case study: VGGNet / OxfordNet**

(runner-up winner of ILSVRC 2014)

[Simonyan and Zisserman]

best model

cs321n, Karpathy, Li
## Case Study

### CNN Visualization

### Note:
- Most memory in early layers
- Most parameters in FC layers

### Table:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Memory (K)</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>224 × 224 3</td>
<td>160K</td>
</tr>
<tr>
<td>CONV1-64</td>
<td>224 × 224</td>
<td>64 × 64</td>
</tr>
<tr>
<td>CONV2-64</td>
<td>112 × 112</td>
<td>64 × 64</td>
</tr>
<tr>
<td>POOL2</td>
<td>112 × 112</td>
<td>64 × 64</td>
</tr>
<tr>
<td>CONV3-128</td>
<td>56 × 56</td>
<td>128 × 128</td>
</tr>
<tr>
<td>CONV3-64</td>
<td>28 × 28</td>
<td>64 × 64</td>
</tr>
<tr>
<td>POOL3</td>
<td>28 × 28</td>
<td>64 × 64</td>
</tr>
<tr>
<td>CONV4-256</td>
<td>14 × 14</td>
<td>256 × 256</td>
</tr>
<tr>
<td>POOL4</td>
<td>7 × 7</td>
<td>256 × 256</td>
</tr>
<tr>
<td>FC</td>
<td>1,000</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

Zeiler, Fergus, 2014
We can compute the partial derivative of input pixels with respect to a cost.
Start with random noise, pretrained network, then iteratively tweak the input as we minimize our cost.

Google Inception v1: layer mixed4a, unit 11

Initial → 4 iterations → 48 iterations → 2048 iterations

CNN Visualization

- Can modify the objective to get different types of insight to what the CNN is responding to.


CNN as Vector Representation

Typical CNN Architecture

R. Ptucha '19
CNN as Vector Representation

- As it turns out, these fully connected layers are excellent descriptors of the input image!
- For example, you can pass images through a pre-trained CNN, then take the output from a FC layer as input to a SVM classifier. (image2vec)
- Images in this vector space generally have the property that similar images are close in this latent representation.

Vision Tasks

**Classification**
- CAT

**Classification + Localization**
- CAT

**Object Detection**
- CAT, DOG, DUCK

**Instance Segmentation**
- CAT, DOG, DUCK

- Single Object
- Multiple Objects
### Classification vs. Classification + Localization

**Classification**
- **Input:** Image
- **Output:** Class label
- **Evaluation metric:** Accuracy

![Image](image1.png) → CAT

**Classification + Localization**
- **Input:** Image
- **Output:** Class label, Box coordinates
- **Evaluation metric:** Intersection over Union (IoU)

![Image](image2.png) → (CAT, x, y, w, h)

---

### Classification with Localization

- Lets allow a few classes:
  1. Car
  2. Truck
  3. Pedestrian
  4. Motorcycle

- For now, let's assume one object per image.
- Each object has \{x, y, w, h\}
- For this image, object location \{x, y, w, h\} = \{0.3, 0.6, 0.4, 0.3\}

![Image](image3.png)

Image from: deeplearning.ai, C4W3L01
Classification with Localization

Four classes:
1. Car
2. Truck
3. Pedestrian
4. Motorcycle

Localization \{x, y, w, h\}

Define y label: \[ y = \begin{bmatrix} b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \]

Probability of an object
Bounding box location
0/1 for each class

Cost function (squared error):
\[ \text{Loss} = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \]

If \( y_i = 1 \):
\[ y = \begin{bmatrix} 1 \\ 0.3 \\ 0.6 \\ 0.4 \\ 0.3 \\ 0 \\ 1 \\ 0 \end{bmatrix} \]

If \( y_i = 0 \):
\[ y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \]

\( ? \) = don't care

Image from: deeplearning.ai, C4W3L01
Classification with Localization

Four classes:
1. Car
2. Truck
3. Pedestrian
4. Motorcycle

Localization \(\{x, y, w, h\}\)

Alternate cost function:
- \(y_1\) can be logistic loss
- \(y_2 \rightarrow y_5\) can be squared error
- \(y_6 \rightarrow y_9\) can be softmax cross entropy

\[
\begin{bmatrix}
1 \\
0.3 \\
0.6 \\
0.4 \\
0.3 \\
0 \\
1 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
P_c \\
b_x \\
b_y \\
b_w \\
b_h \\
c_1 \\
c_2 \\
c_3 \\
c_4
\end{bmatrix} = \begin{bmatrix}
0 \\
? \\
? \\
? \\
? \\
? \\
? \\
? \\
?
\end{bmatrix}
\]

? = don't care

Image from: deeplearning.ai, C4W3L01

Snapchat Facewarp?

- Traditional approach:
  - Viola Jones Face Detection
  - Search for actual point locations using Mahalanobis distance
  - Repeat ~3-5x
  - Average eye and 82 facial feature points
  - Restrict based on PCA statistics
Snapchat Facewarp?

- Deep Learning approach:

  - Test image
  - Image pyramid
  - Stage 1: Candidate faces
  - Stage 2: Refine face selection
  - Stage 3: Facial feature points

  MT-CNN [Zhang et al. 2016]

Localization

- Facial feature

Each face has 68 points, so CNN would output:

  - Face?
  - pt1X
  - pt1Y
  - pt2X
  - pt2Y
  .
  .
  - pt68X
  - pt68Y

  137 outputs

  Of course, need GT for thousands of faces to train model.
Can do same with Body Pose…

Object Detection

More than one object per image?

<table>
<thead>
<tr>
<th>Training set:</th>
<th>Car detection example</th>
</tr>
</thead>
<tbody>
<tr>
<td>x 1</td>
<td>y 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Images from: deeplearning.ai, C4W3L03

Sliding Window Detection

Images from: deeplearning.ai, C4W3L03
Computing FC layers with Convolution

Replacing Sliding Windows w/Fully Convolutional CNNs
Replacing Sliding Windows w/Fully Convolutional CNNs

**Sliding window approach:**
Sequentially evaluate one window at a time

**Fully convolutional approach:**
Evaluate 64 regions at once

Images from: deeplearning.ai, C4W3L04

- Can think of this as evaluating an 8×8 grid, where each of the 64 cells is independently checked for an object:
  
  Each cell has a y label: 
  
  \[
  y = \begin{bmatrix}
  P_c \\
  b_x \\
  b_y \\
  b_w \\
  b_h \\
  C_1 \\
  C_2 \\
  C_3 \\
  C_4 \\
  \end{bmatrix}
  \]

  Prob. of an object

  Object location

  0/1 for each class (Four classes in this example)
Replacing Sliding Windows w/Fully Convolutional CNNs

- Overlay GT of object
- Cell where centroid lie is responsible.

Each cell has a y label: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

Prob. of an object
Object location

0/1 for each class
(Four classes in this example)

Note1: cell upper left (0,0); cell lower right (1,1)
Note2: $b_w$ and $b_h$ can be > 1.0
CNN Results

- Handwritten characters
  - MNIST: 0.17% error, Ciresan et al ’11
  - Arabic & Chinese: Ciresan ‘12

- CIFAR-10 (60K images of 10 classes)
  - 9.3% error, Wan et al. ’13

- Traffic Sign Recognition
  - 0.56% error vs 1.16% for humans, Ciresan ’11

ImageNet

- Amazon Turk did bulk of labeling
- 14M labeled images
- 20K classes

- 1.2M images, 1000 categories
- Image classification, object localization, video detection
ImageNet: Examples of Hammer

Deep Learning- Surpassing The Visual Cortex’s Object Detection and Recognition Capability

Top-5 error on ImageNet

<table>
<thead>
<tr>
<th>Year</th>
<th>Traditional Computer Vision and Machine Learning</th>
<th>Deep Convolution Neural Networks (CNNs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>2011</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>2012</td>
<td>15.4</td>
<td>11.2</td>
</tr>
<tr>
<td>2013</td>
<td>11.2</td>
<td>6.7</td>
</tr>
<tr>
<td>2014</td>
<td>5.11</td>
<td>3.57</td>
</tr>
<tr>
<td>2015</td>
<td>3.57</td>
<td>2.99</td>
</tr>
<tr>
<td>2016</td>
<td>2.99</td>
<td>2.25</td>
</tr>
<tr>
<td>2017</td>
<td>2.25</td>
<td>2018 moved to Kaggle and made localization</td>
</tr>
<tr>
<td>2018</td>
<td>moved to Kaggle and made localization</td>
<td>Similar effect demonstrated on voice and pattern recognition</td>
</tr>
</tbody>
</table>

R. Ptucha ‘19
Deep Learning Specialization, Five courses:
1. Neural Networks and Deep Learning
2. Improving Deep Neural Networks
3. Structured Machine Learning Projects
4. Convolutional Neural Networks
5. Sequence Models

CS231n: Convolutional Neural Networks for Visual Recognition

Li, Johnson, Yeung 2017
http://cs231n.stanford.edu/
For More Information:  [http://www.rit.edu/mil](http://www.rit.edu/mil)

Raymond W. Ptucha  
Assistant Professor, Computer Engineering  
Director, Machine Intelligence Laboratory  
Rochester Institute of Technology  

Email: [rwpeec@rit.edu](mailto:rwpeec@rit.edu)  
Phone: +1 (585) 797-5561  
Office: GLE (09) 3441