Introduction to Deep Learning for Facial Understanding
Part II: Convolutional Neural Networks
Raymond Ptucha, Rochester Institute of Technology, USA
Tutorial-9
May 19, 2018

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, rwpeec@rit.edu
Agenda

• 9-9:30am Part I: Introduction
• 9:30-10:00am Part II: Convolutional Neural Nets
• 10:00-10:30am Part III: Fully Convolutional Nets
• 10:30-10:45am Break
• 10:45-11:15am Part IV: Facial Understanding
• 11:15-11:35am Part V: Recurrent Neural Nets
• 11:35-11:55pm Part VI: Generative Adversarial Nets
• 11:55-12:30am Hands-on with NVIDIA DIGITS

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.
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
Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

Source Pixel

Convolution kernel (a.k.a. filter)

New pixel value (destination pixel)

\[
\frac{1}{9} \left( 1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 4 + 1 \times 4 + 1 \times 0 + 1 \times 5 + 1 \times 5 \right) = 2
\]

R. Ptucha '18

13

R. Ptucha '18

14
Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

\[
\frac{1}{9} (1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 4 + 1 \times 4 + 1 \times 0 + 1 \times 5 + 1 \times 5) = 2
\]

Top

Middle

Bottom

R. Ptucha '18

15
Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

\[
\frac{1}{9} (1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 4 + 1 \times 4 + 1 \times 1 + 1 \times 5 + 1 \times 5 + 1 \times 2) = 2.3 \approx 2
\]

Top
Middle
Bottom
R. Ptucha ‘18
Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

\[
\frac{1}{9} (1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 2 + 1 \times 2 + 1 \times 1) = 0.9 \approx 1
\]
Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

\[
\frac{1}{9} (1 \times 0 + 1 \times 4 + 1 \times 4 + 1 \times 0 + 1 \times 5 + 1 \times 5 + 1 \times 0 + 1 \times 4 + 1 \times 5) = 3
\]

Top

Middle

Bottom

R. Ptucha '18

Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

\[
\frac{1}{9} (1 \times 4 + 1 \times 4 + 1 \times 1 + 1 \times 5 + 1 \times 5 + 1 \times 2 + 1 \times 4 + 1 \times 5 + 1 \times 2) = 3.6 \approx 4
\]

Top

Middle

Bottom

R. Ptucha '18
Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

\[
\frac{1}{9} \left( 1 \times 4 + 1 \times 1 + 1 \times 1 \right) + 1 \times 5 + 1 \times 2 + 1 \times 2 + 1 \times 5 + 1 \times 2 + 1 \times 2 = 2.7 \approx 3
\]

Top
Middle
Bottom

R. Ptucha '18

Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

\[
\frac{1}{9} \left( 1 \times 2 + 1 \times 1 + 1 \times 0 \right) + 1 \times 2 + 1 \times 2 + 1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 0 = 0.8 \approx 1
\]

Top
Middle
Bottom

R. Ptucha '18
Convolution

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

Image Convolution

By padding (filterWidth-1)/2, output image size matches input image size

https://github.com/vdumoulin/conv_arithmetic
Max Pooling - Reducing the Size of an Image

Convolution Neural Network (CNN) Building Block
Putting it All Together

- Convolution
- Pooling

Whole System

1st stage → 2nd stage → 3rd stage

Input Image → Fully Conn. Layers → Class Labels

Learning Filters

32 Learned Filters, each 5×5

Input image 28×28

32 Filtered images, each is 28×28

Use zero padding
Filters

- $3 \times 3$ filter
- $3 \times 3$ filter
- $3 \times 3 \times 4$ filter

Learning Filters

- 32 Learned Filters, each $5 \times 5 \times 3$
- 32 Filtered images, each $28 \times 28 \times 1$

Input image $28 \times 28 \times 3$

Use zero padding
CNN Architecture

(Not so) Toy Example

Output: prediction of 1 of 10 categories

Input RGB image: 64x64x3 pixels

Max pooling × 2

Max pooling × 2

Max pooling × 2

Fully connected to 10 classes

16 filters, each filter is 5x5x32. 2 pixel pad.

32 filters, each filter is 5x5x16. 2 pixel pad.

64 filters, each filter is 5x5x32. 2 pixel pad.

Fully connected 4x4x64x16

=⇒ 16 filters, each filter is 4x4x64. filter, 0 pixel pad.

R. Ptucha '18

Case Study

Case study: VGGNet / OxfordNet
(runner-up winner of ILSVRC 2014)

[Simonyan and Zisserman]

best model

cs321n, Karpathy, Li
Case Study

Note: Most memory in early layers

Note: Most parameters in FC layers

CNN Visualization

Zeiler, Fergus, 2014
CNN Visualization

Zeiler, Fergus, 2014

R. Ptucha '18

CNN as Vector Representation

Typical CNN Architecture

Input Image 2D Plot of fc8 Feature Vector Image of fc8 Feature Vector

R. Ptucha '18
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.

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

Andrew Ng, 2017
https://www.deeplearning.ai/
CS231n: Convolutional Neural Networks for Visual Recognition

Li, Johnson, Yeung 2017
http://cs231n.stanford.edu/1

Thank you!!
Ray Ptucha
rwpeec@rit.edu

https://www.rit.edu/mil