Introduction to Deep Learning for Facial Understanding

Part II: Convolutional Neural Networks

Raymond Ptucha,
Rochester Institute of Technology, USA

www.nvidia.com/dli

Tutorial-9
May 19, 2018

The 13th IEEE Conference on Automatic Face and Gesture Recognition
FG 2018 Xi'an, China | 15-19 May 2018

Fair Use Agreement

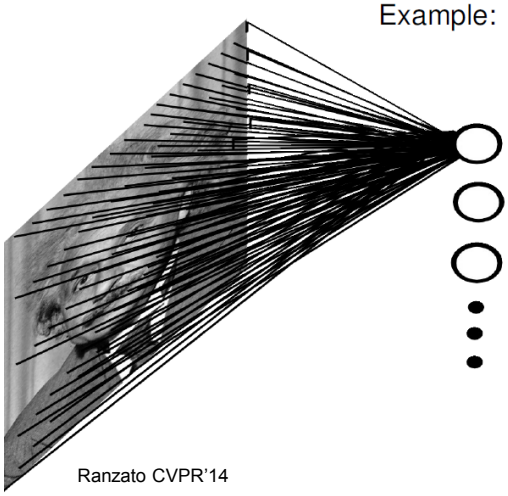
This agreement covers the use of all slides in this document, please read carefully.

- 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.

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?

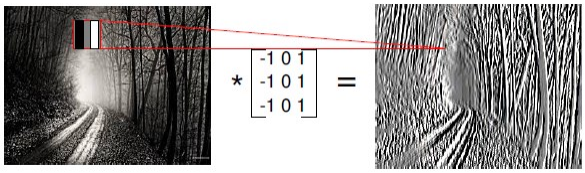


Ranzato CVPR'14

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.

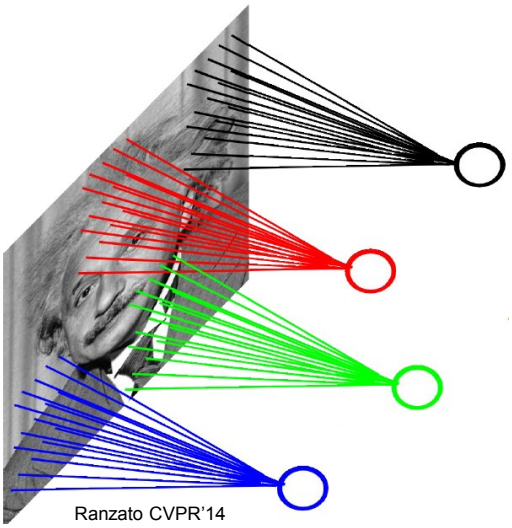
Convolution Filter



Ranzato CVPR'14

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

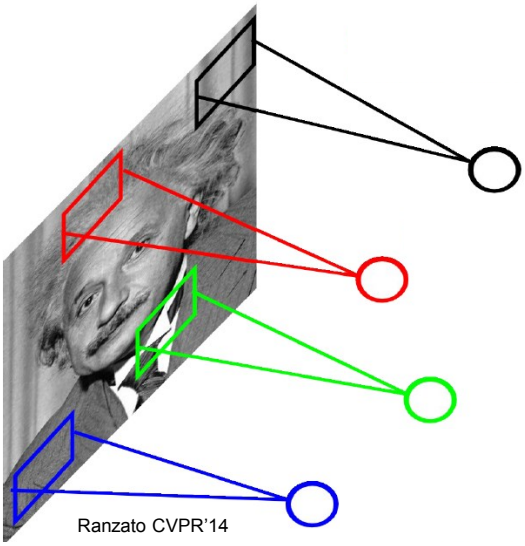
Locally Connected Layer



Ranzato CVPR'14

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

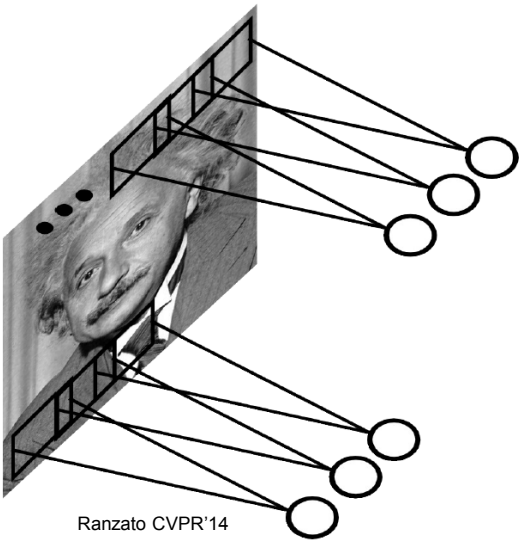
Locally Connected Layer



- 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?

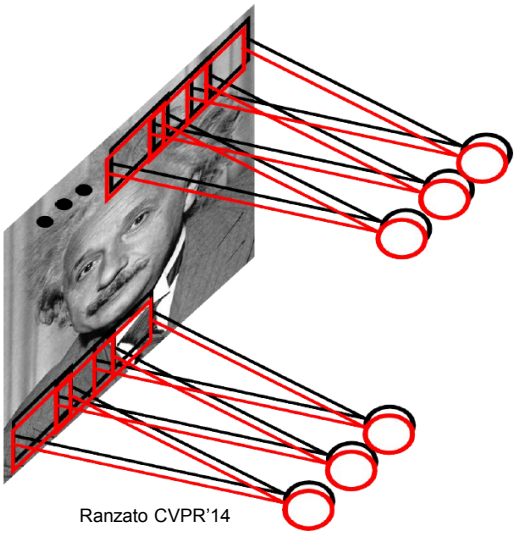
Convolution Layer



- 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.

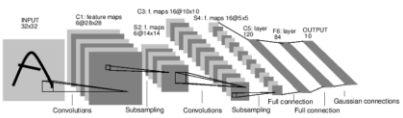
Convolution Layer



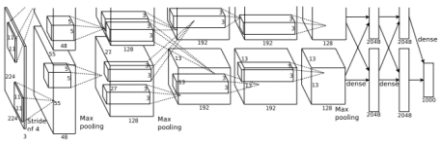
Ranzato CVPR'14

- 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!

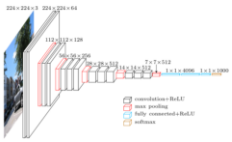
Many Flavors of CNNs…



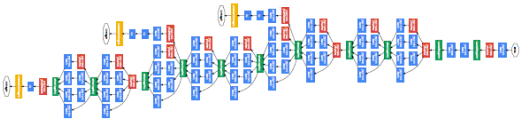
LeNet-5, LeCun 1989



AlexNet, Krizhevsky 2012



VGGNet, Simonyan 2014

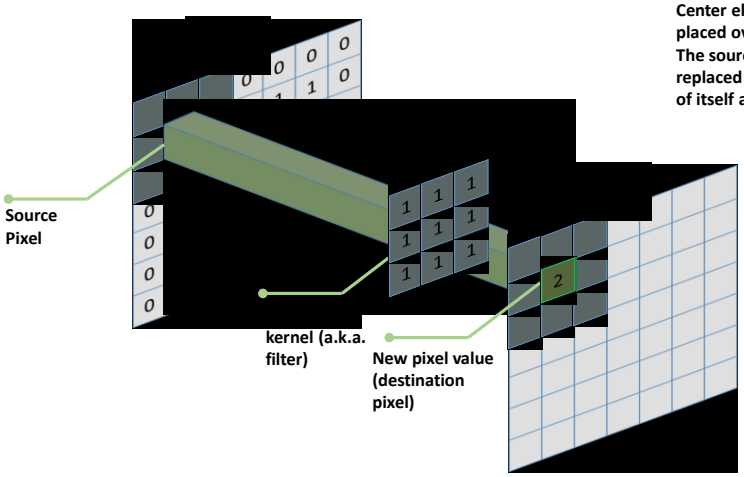


GoogLeNet (Inception), Szegedy 2014

ResNet, He 2015

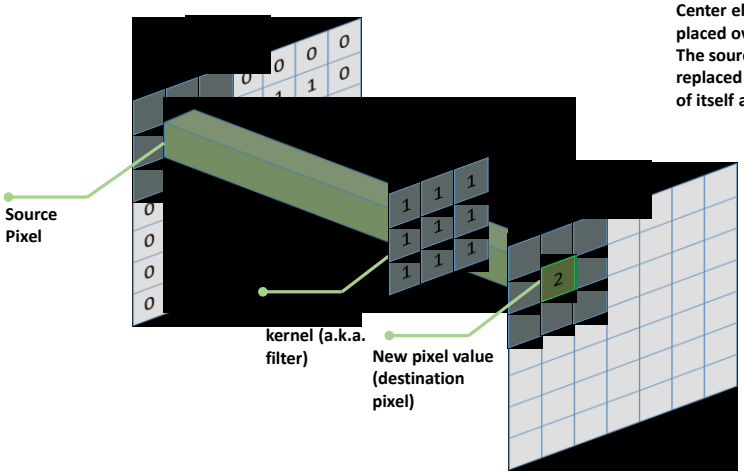
DenseNet, Huang 2017

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.

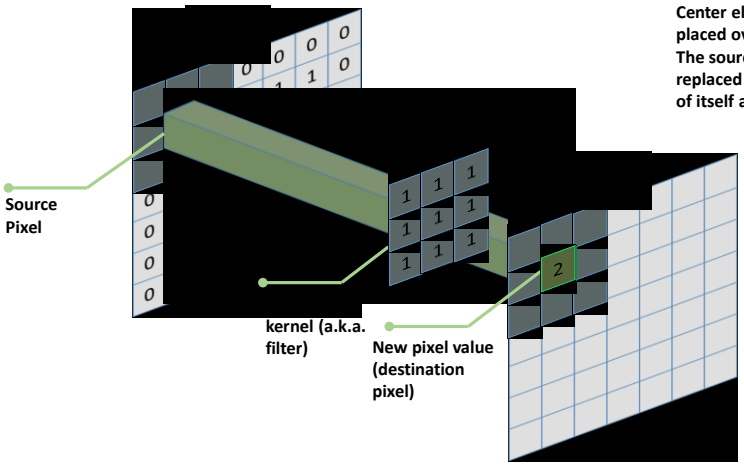
Convolution



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.

$$\frac{1}{9}(\underbrace{1*0+1*0+1*0}_{\text{Top}}+1*0+1*4+1*4+1*0+1*5+1*5)=2$$

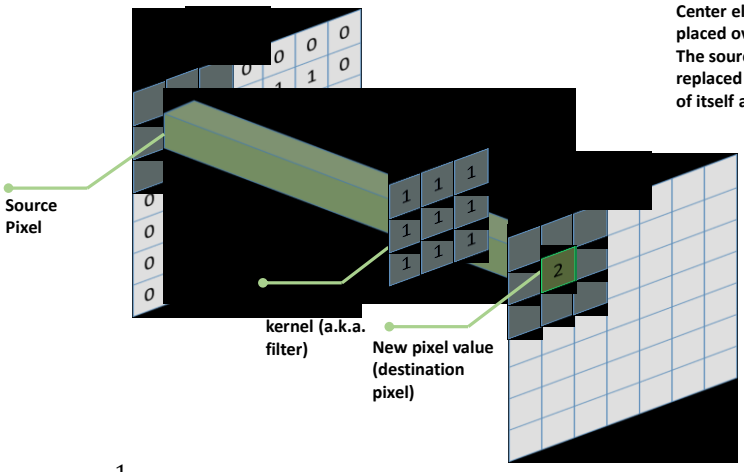
Convolution



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.

$$\frac{1}{9}(\underbrace{1*0+1*0+1*0}_{\text{Top}}+\underbrace{1*0+1*4+1*4}_{\text{Middle}}+1*0+1*5+1*5)=2$$

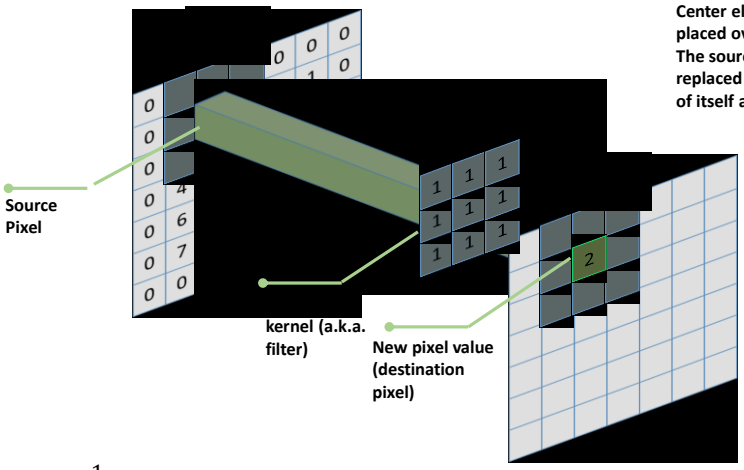
Convolution



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.

$$\frac{1}{9}(\underbrace{1*0+1*0+1*0}_{\text{Top}}+\underbrace{1*0+1*4+1*4}_{\text{Middle}}+\underbrace{1*0+1*5+1*5}_{\text{Bottom}})=2$$

Convolution



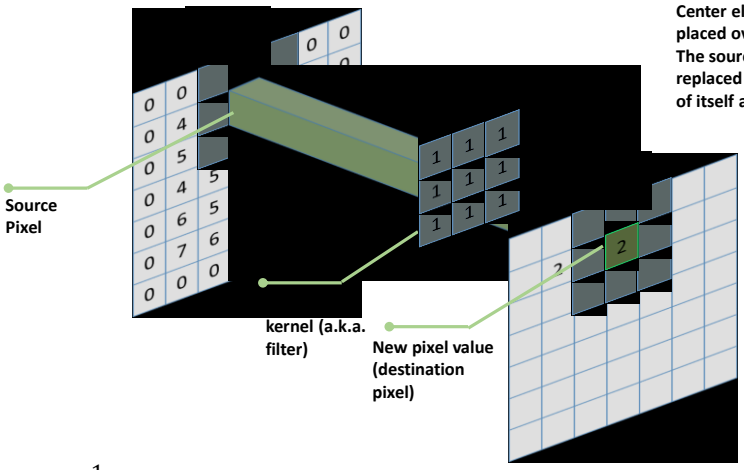
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

$$\frac{1}{9}(\underbrace{1*0+1*0+1*0}_{\text{Top}}+\underbrace{1*4+1*4+1*1}_{\text{Middle}}+\underbrace{1*5+1*5+1*2}_{\text{Bottom}})=2.3\approx 2$$

R. Ptucha '18

17

Convolution



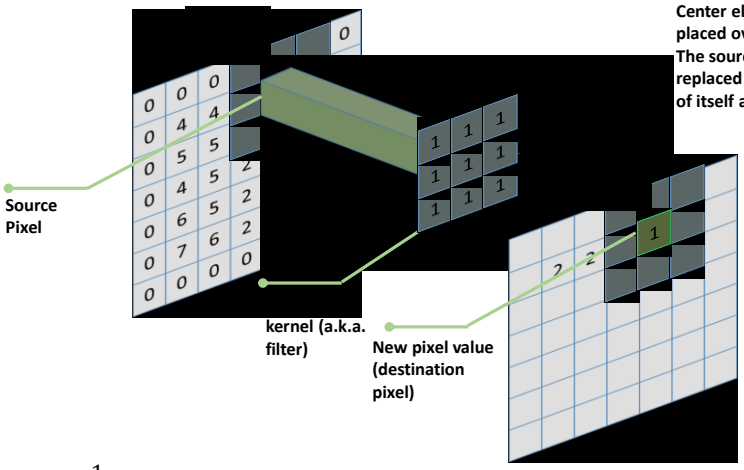
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

$$\frac{1}{9}(\underbrace{1*0+1*0+1*0}_{\text{Top}}+\underbrace{1*4+1*1+1*1}_{\text{Middle}}+\underbrace{1*5+1*2+1*2}_{\text{Bottom}})=1.7\approx 2$$

R. Ptucha '18

18

Convolution

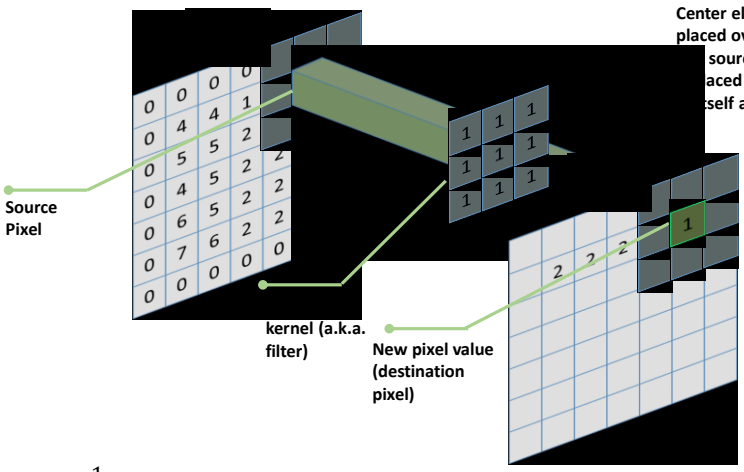


$$\frac{1}{9}(\underbrace{1*0+1*0+1*0}_{\text{Top}}+\underbrace{1*1+1*1+1*1}_{\text{Middle}}+\underbrace{1*2+1*2+1*1}_{\text{Bottom}})=0.9\approx 1$$

R. Ptucha '18

19

Convolution

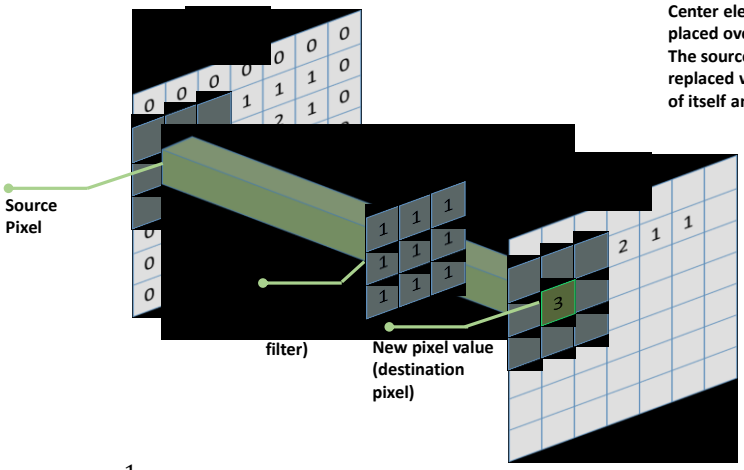


$$\frac{1}{9}(\underbrace{1*0+1*0+1*0}_{\text{Top}}+\underbrace{1*1+1*1+1*0}_{\text{Middle}}+\underbrace{1*2+1*1+1*0}_{\text{Bottom}})=0.6\approx 1$$

R. Ptucha '18

20

Convolution



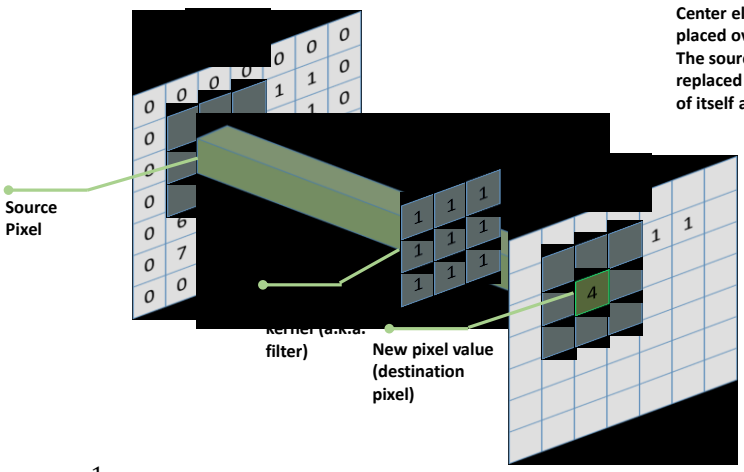
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

$$\frac{1}{9}(\underbrace{1*0+1*4+1*4}_{\text{Top}}+\underbrace{1*0+1*5+1*5}_{\text{Middle}}+\underbrace{1*0+1*4+1*5}_{\text{Bottom}})=3$$

R. Ptucha '18

21

Convolution



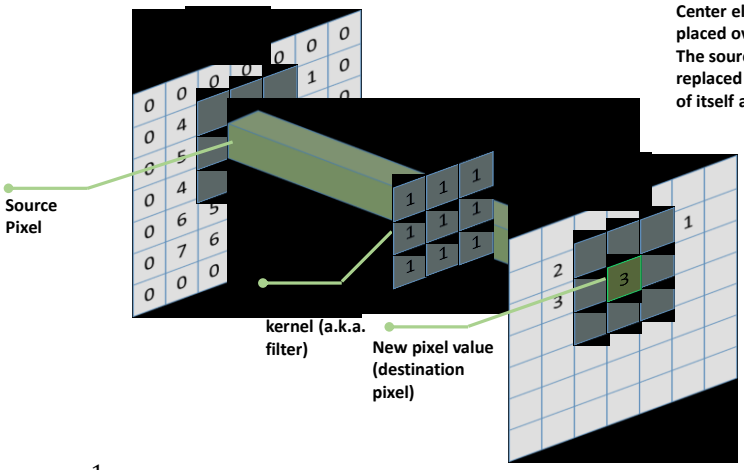
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

$$\frac{1}{9}(\underbrace{1*4+1*4+1*1}_{\text{Top}}+\underbrace{1*5+1*5+1*2}_{\text{Middle}}+\underbrace{1*4+1*5+1*2}_{\text{Bottom}})=3.6\approx 4$$

R. Ptucha '18

22

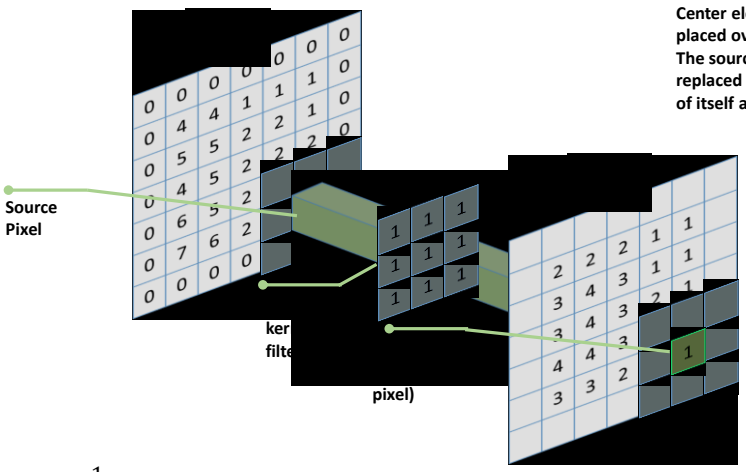
Convolution



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.

$$\frac{1}{9}(\underbrace{1*4+1*1+1*1}_{\text{Top}}+\underbrace{1*5+1*2+1*2}_{\text{Middle}}+\underbrace{1*5+1*2+1*2}_{\text{Bottom}})=2.7\approx 3$$

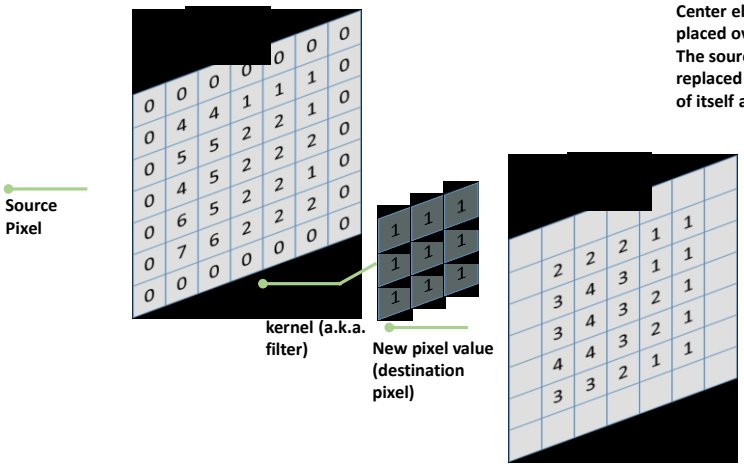
Convolution



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.

$$\frac{1}{9}(\underbrace{1*2+1*1+1*0}_{\text{Top}}+\underbrace{1*2+1*2+1*0}_{\text{Middle}}+\underbrace{1*0+1*0+1*0}_{\text{Bottom}})=0.8\approx 1$$

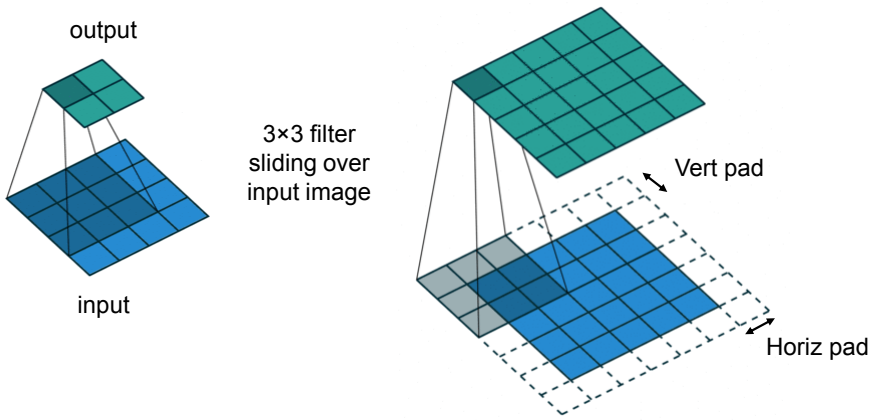
Convolution



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.

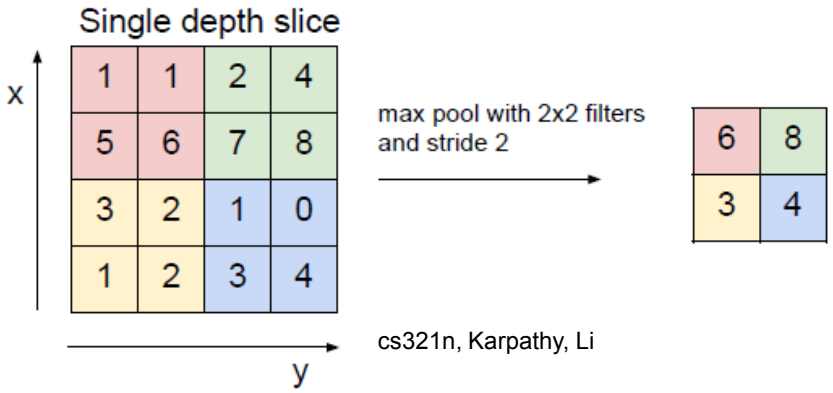
Image Convolution

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



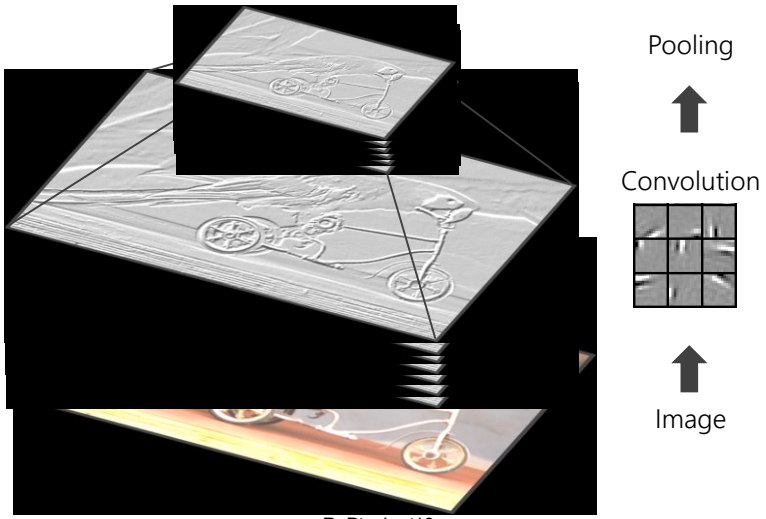
https://github.com/vdumoulin/conv_arithmetic

Max Pooling- Reducing the Size of an Image

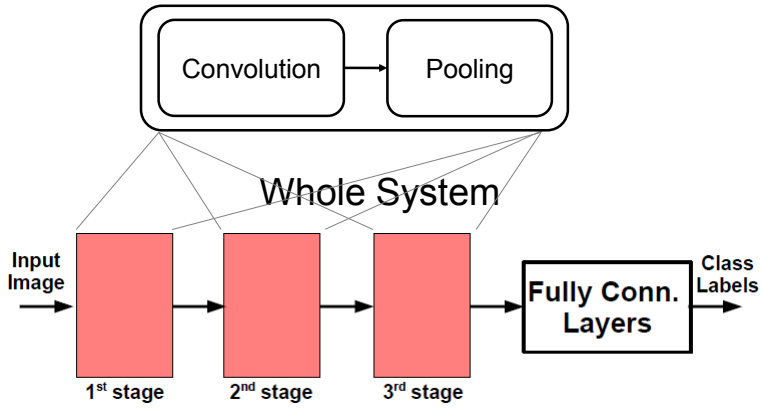


cs321n, Karpathy, Li

Convolution Neural Network (CNN) Building Block



Putting it All Together



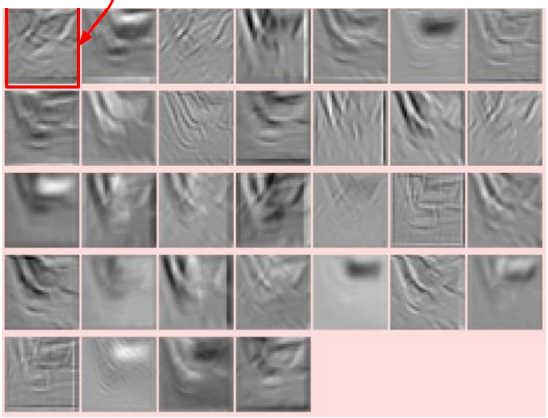
Learning Filters

32 Learned Filters, each 5×5

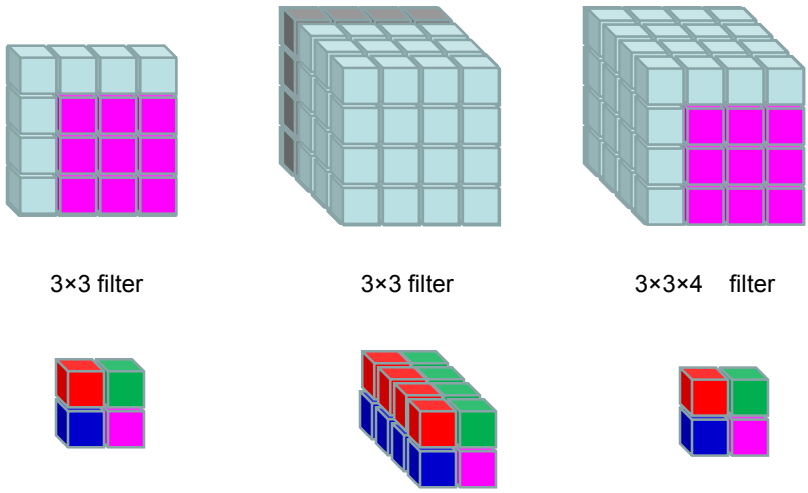
32 Filtered images, each is 28×28

Input image
28×28

Use zero padding



Filters

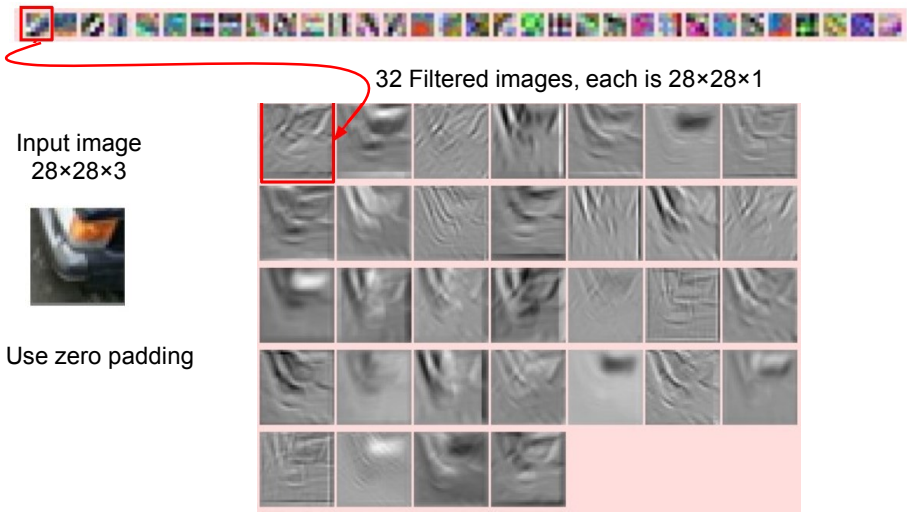


R. Ptucha '18

32

Learning Filters

32 Learned Filters, each 5×5×3



R. Ptucha '18

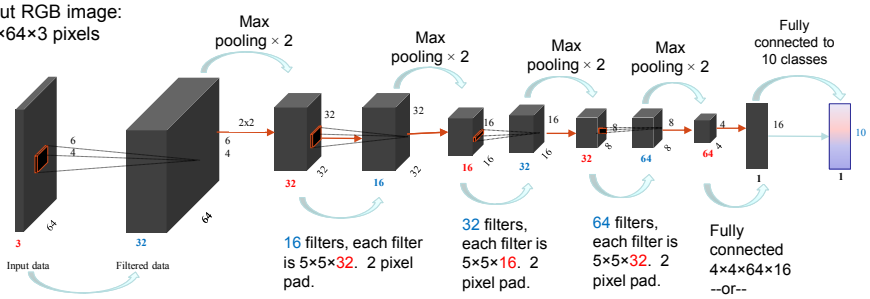
33

CNN Architecture

(Not so) Toy Example

Output: prediction of 1 of 10 categories

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.

Fully connected 4×4×64×16 --or-- 16 filters, each filter is 4×4×64. filter, 0 pixel pad.

Case Study

Case study: VGGNet / OxfordNet
(runner-up winner of ILSVRC 2014)
[Simonyan and Zisserman]

best model

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

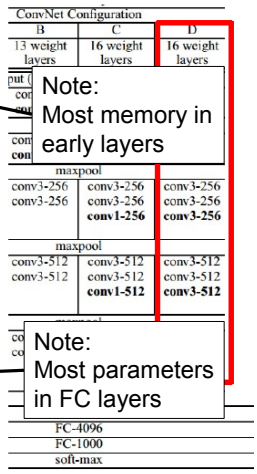
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

cs321n, Karpathy, Li

Case Study

(not counting biases)

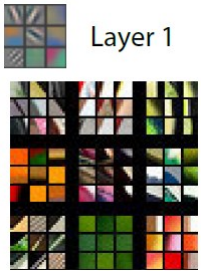
INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M 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

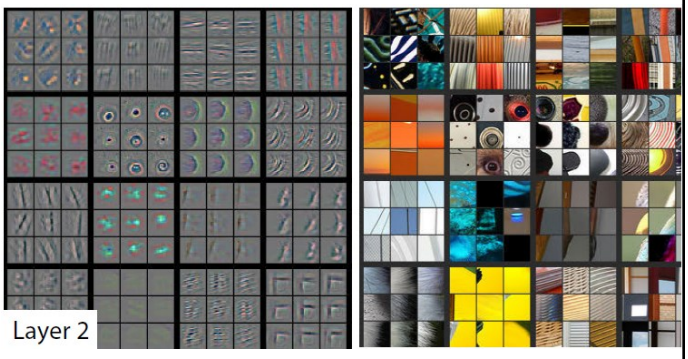


Note:
Most memory in early layers

Note:
Most parameters in FC layers

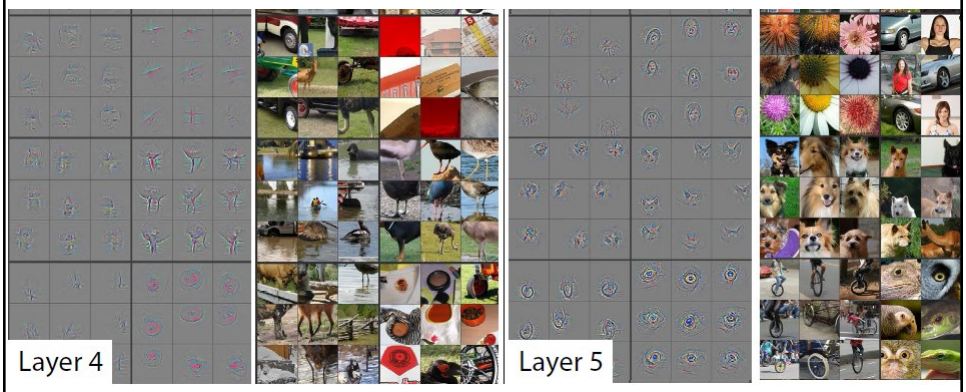
CNN Visualization





Zeiler, Fergus, 2014

CNN Visualization

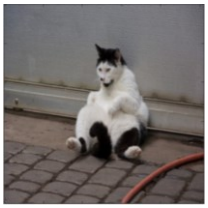


Zeiler, Fergus, 2014

CNN as Vector Representation

Typical CNN Architecture

Input Image



2D Plot of fc8 Feature Vector

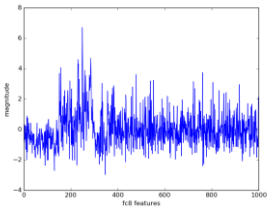
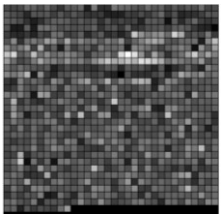


Image of fc8 Feature Vector



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.

deeplearning.ai | coursera

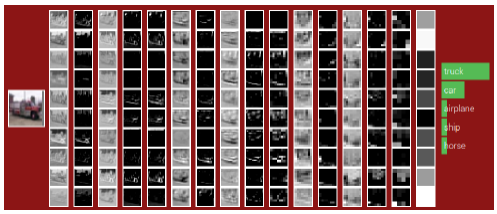
Andrew Ng, 2017

https://www.deeplearning.ai/

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/ l

Thank you!!

Ray Ptucha
rwpeec@rit.edu

https://www.rit.edu/mil