Fundamentals of Deep Learning

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

Electronic Imaging 2020: SC20
January 28, 2019, 8:30am-12:45pm

IS&T International Symposium on
Electronic Imaging 2020
SCIENCE AND TECHNOLOGY

26–30 January 2020
Burlingame, California USA

1

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.

2

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

4

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

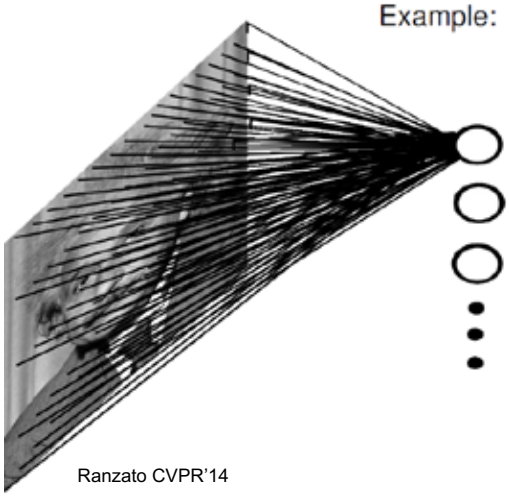
5

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

6

Fully Connected Layers?



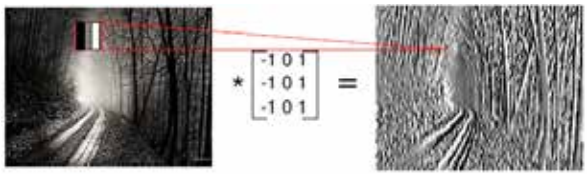
Ranzato CVPR'14

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.

7

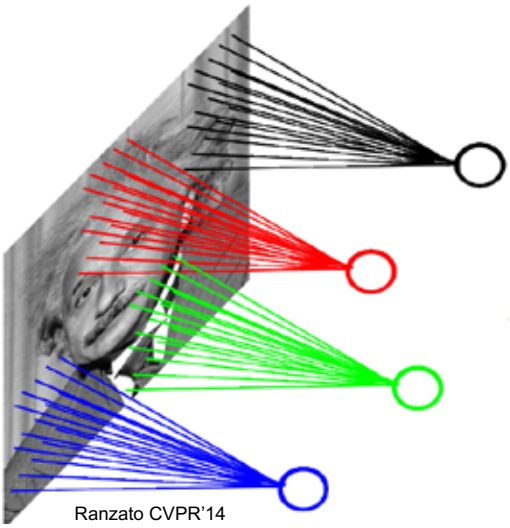
Convolution Filter



Ranzato CVPR'14

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

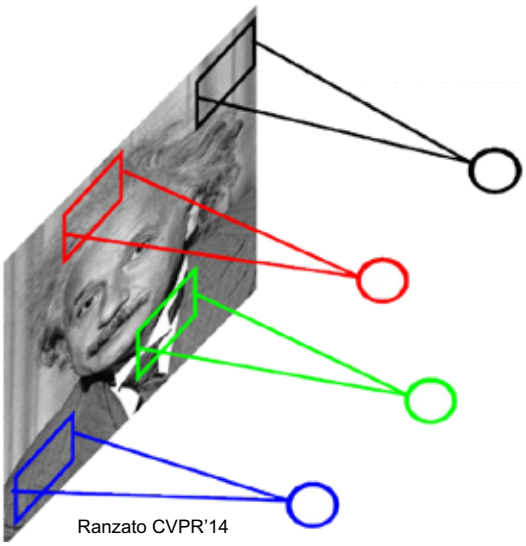
Locally Connected Layer



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

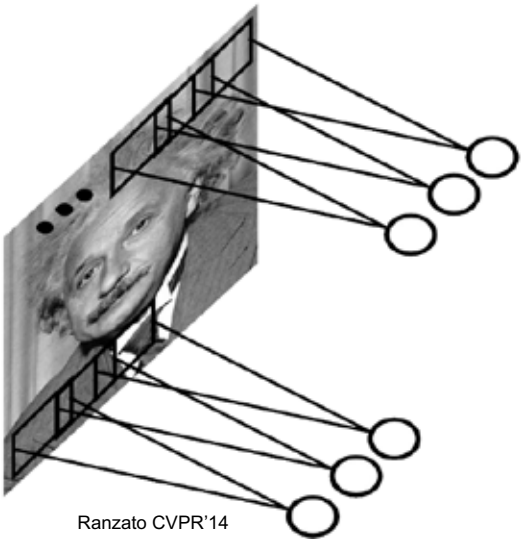


- 200×200 pixel image.
- 40K input.
- Four 10×10 filters, each fully connected
- 40K×10×10×4=16M weights….getting better!

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

10

Convolution Layer

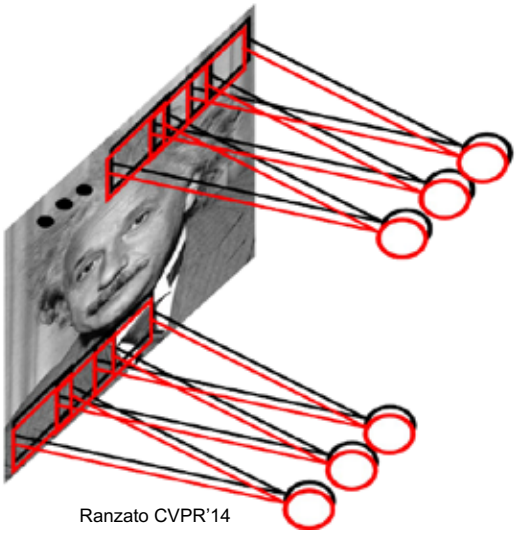


- 200×200 pixel image.
- 40K input.
- Four 10×10 filters, each fully connected
- 40K×10×10×4=16M weights….getting better!

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

11

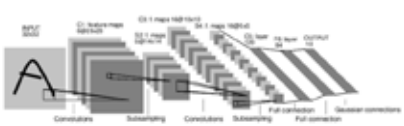
Convolution Layer



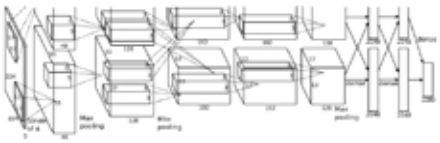
Ranzato CVPR'14

- 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!

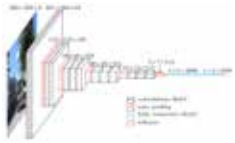
Many Flavors of CNNs…



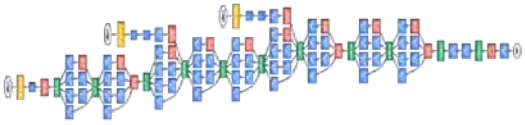
LeNet-5, LeCun 1989



AlexNet, Krizhevsky 2012



VGGNet, Simonyan 2014



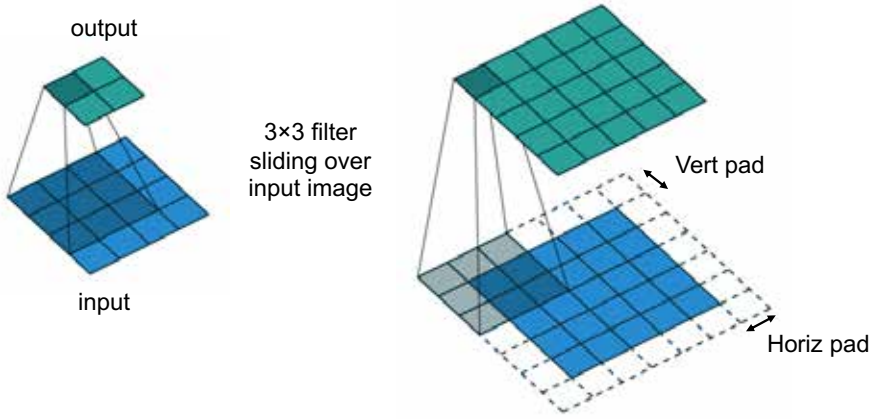
GoogLeNet (Inception), Szegedy 2014

ResNet, He 2015

DenseNet, Huang 2017

Image Convolution

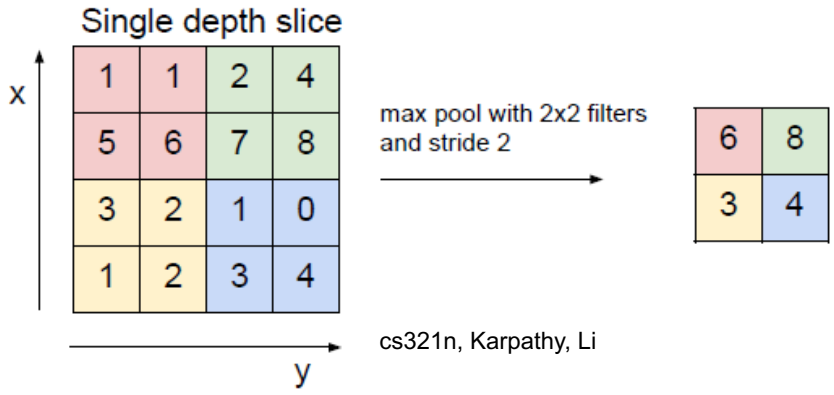
By padding (filterWidth-1)/2, output image size matches input image size



https://github.com/vdumoulin/conv_arithmetic

14

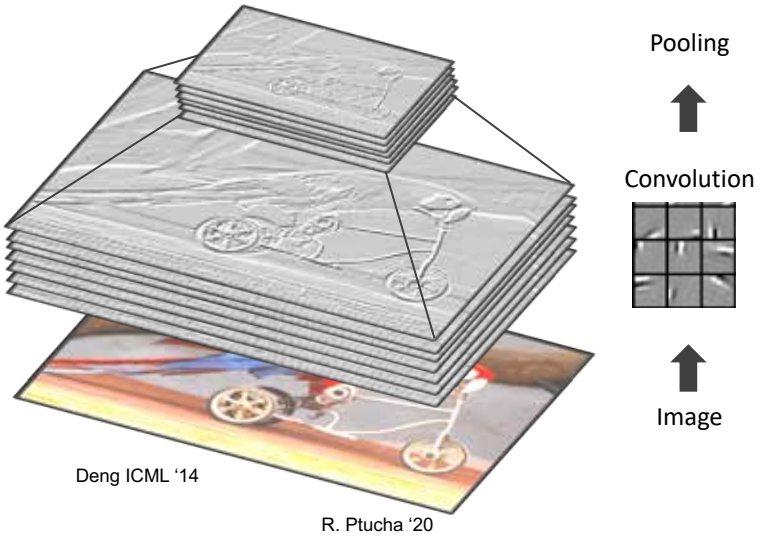
Max Pooling- Reducing the Size of an Image



cs321n, Karpathy, Li

15

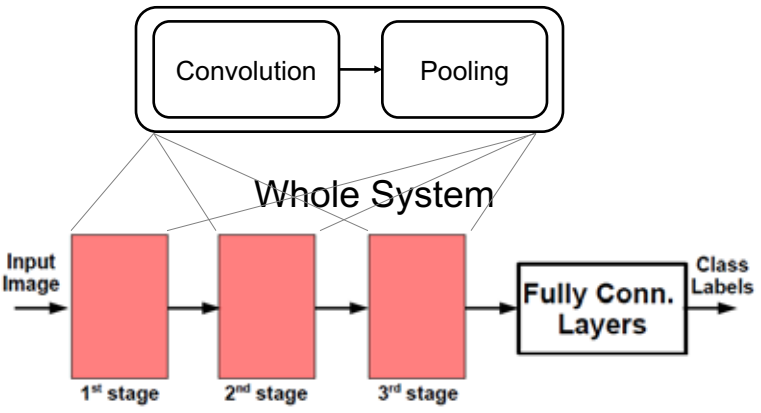
Convolution Neural Network (CNN) Building Block



R. Ptucha '20 16

16

Putting it All Together



R. Ptucha '20 17

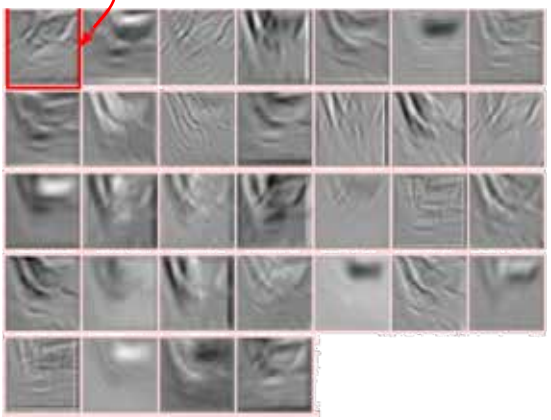
17

Learning Filters

32 Learned Filters, each 5×5

32 Filtered images, each is 28×28

Input image
28×28



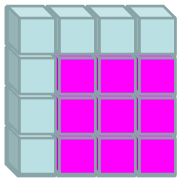
Use zero padding

R. Ptucha '20

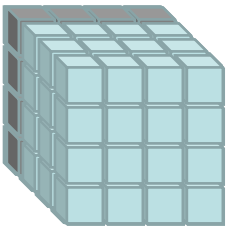
21

21

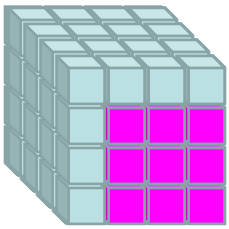
Filters



3×3 filter



3×3 filter



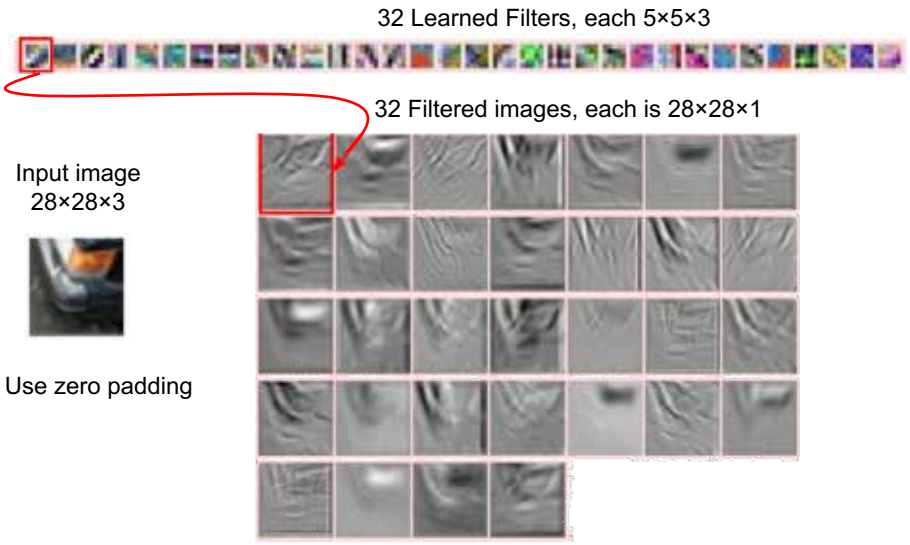
3×3×4 filter

R. Ptucha '20

22

22

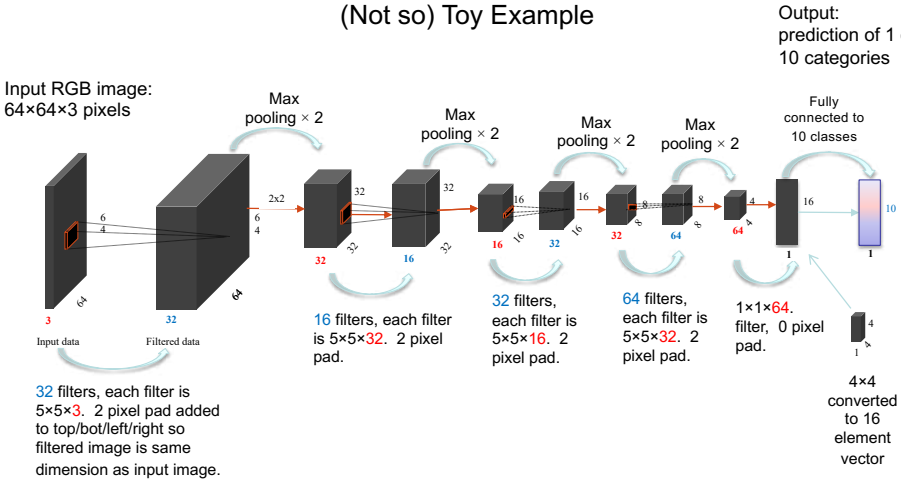
Learning Filters



R. Ptucha '20 23

23

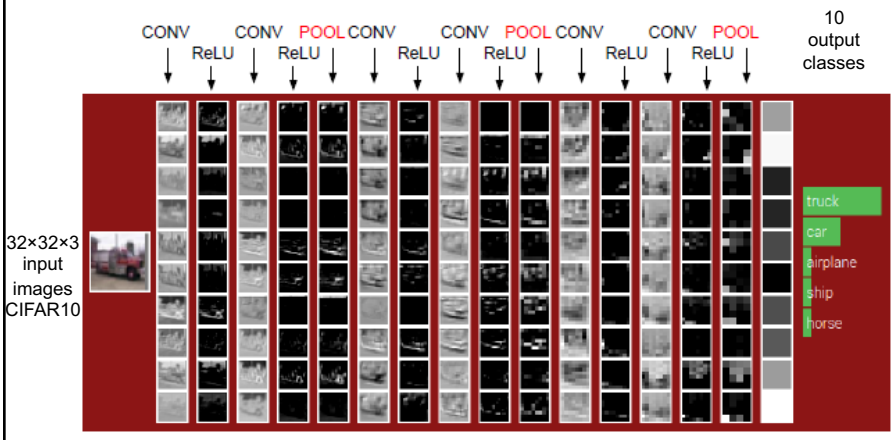
CNN Architecture



R. Ptucha '20 24

24

CNN Example



cs231n, Li, Karpathy, Johnson

Note: Two Conv's between each pool…

25

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

26

CNN Example

Input: 16×16×10

- CONV with five 3×3 filters, stride 1, pad 1:
  - Parameters: (3*3*10)*5 + 5 = 455
  - Memory: 16*16*5
- CONV with twenty 3×3 filters, stride 1, pad 1:
  - Parameters: (3*3*5)*20 + 20 = 920
  - Memory: 16*16*20
- Pool with 2×2 filters, stride 2:
  - Parameters: 0
  - Memory: 8*8*20

CNN Example

Input: 8×8×20

- CONV with ten 3×3 filters, stride 1, pad 1:
  - Parameters: (3*3*20)*10 + 10 = 1810
  - Memory: 8*8*10
- CONV with twenty 5×5 filters, stride 1, pad 2:
  - Parameters: (5*5*10)*20 + 20 = 5020
  - Memory: 8*8*20
- Pool with 2×2 filters, stride 2:
  - Parameters: 0
  - Memory: 4*4*20

CNN Example

- Fully connect (FC) 4x4x20 to 10 output classes
  - Parameters: (4*4*20)*10 + 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

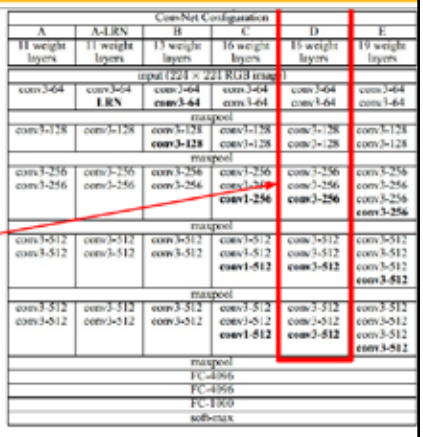


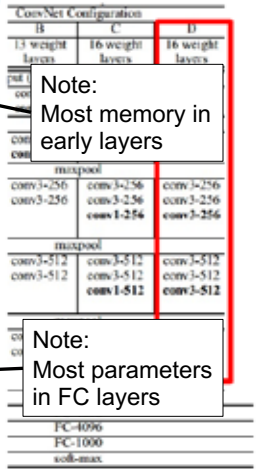
Table 2. **Number of parameters** (in millions).

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

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

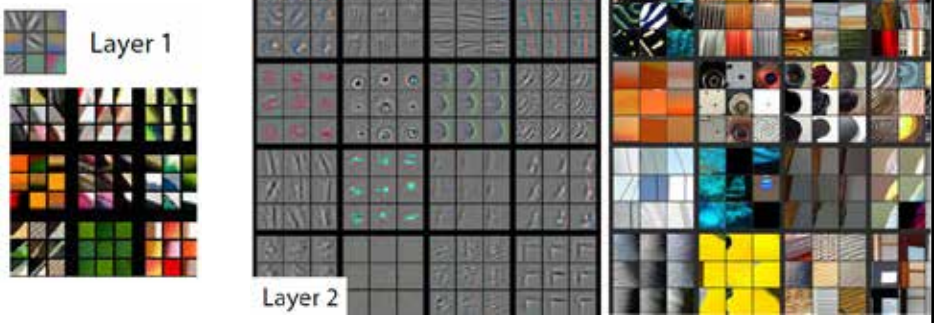


Note:
Most memory in early layers

Note:
Most parameters in FC layers

31

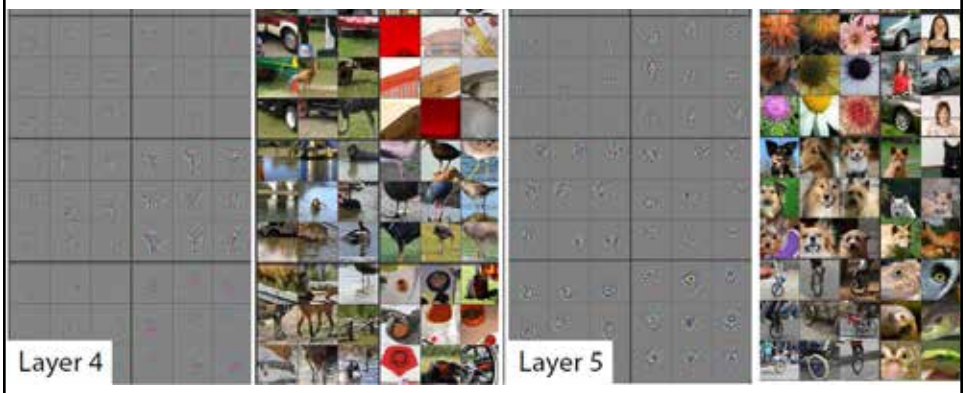
CNN Visualization



Zeiler, Fergus, 2014

32

CNN Visualization



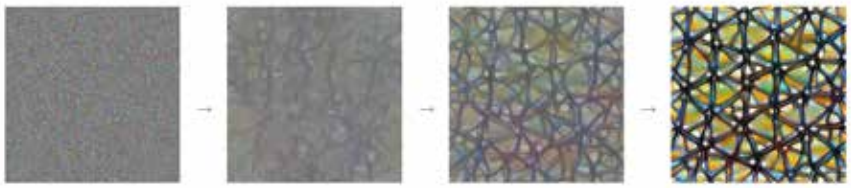
Zeiler, Fergus, 2014

33

CNN Visualization

- We can compute the partial derivative of input pixels with respect to a cost.
- Start with random noise image, pretrained network, then iteratively tweak the input as we minimize our cost.
  - Cost is to activate a particular neuron (in this slide).

Google Inception v1: layer mixed4a, unit 11



Initial 4 iterations 48 iterations 2048 iterations

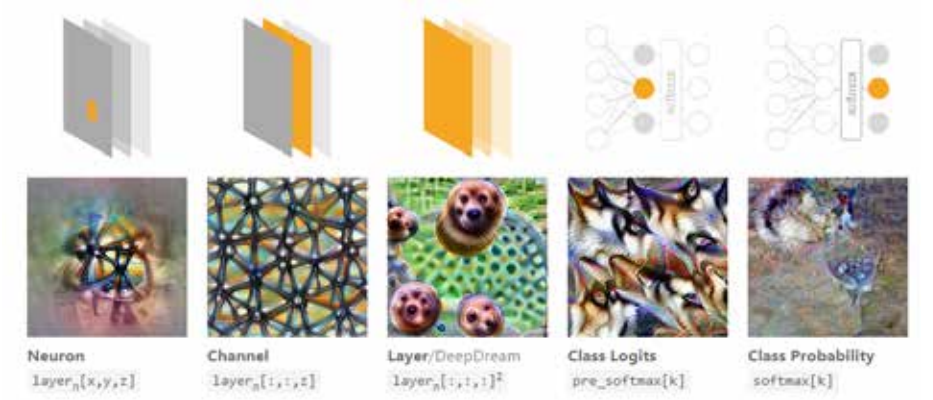
Olah, et al., "Feature Visualization", Distill, 2017.

34

CNN Visualization

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

n layer index
x,y spatial position
z channel index
k class index



Olah, et al., "Feature Visualization", Distill, 2017.

Interrogate a network with different gradients

Start with any image, then when do back prop, use GT label of say a dog, then instead of updating the model weights, update the pixels in the image.

Inceptionism: Going Deeper into Neural Networks
A. Mordvintsev, C. Olah and M. Tyka (2015)

Interrogate a network with different gradients

Delta is the difference between logits and one-hot encoding of dog. After backprop, the next time you pass in the image, it will be closer to being classified as a dog!

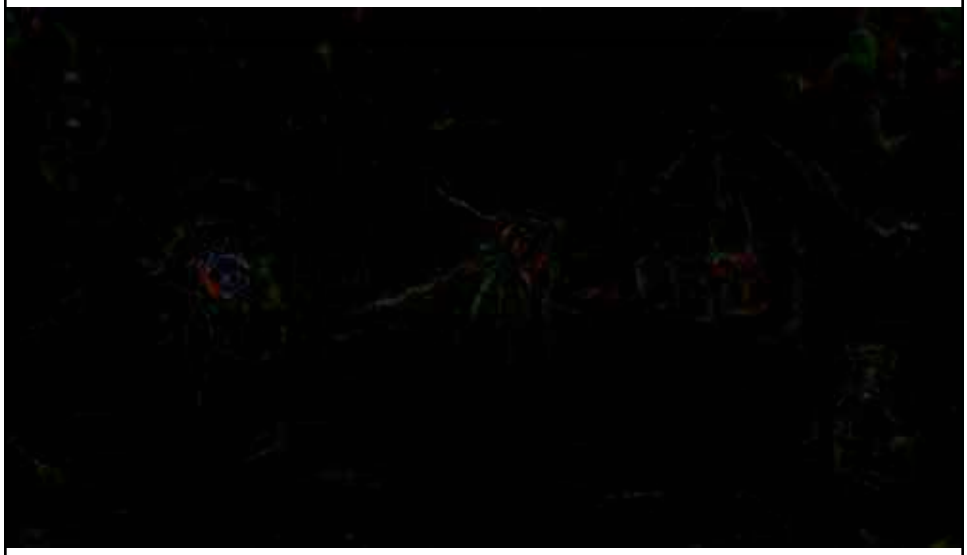
Then iteratively keep updating pixels with dog GT label….eventually, the image will start to "look" like a dog

Inceptionism: Going Deeper into Neural Networks
A. Mordvintsev, C. Olah and M. Tyka (2015)

37

A. Mordvintsev, C. Olah and M. Tyka,
http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html

38



39

CNN as Vector Representation

Typical CNN Architecture

Input Image

2D Plot of fc8 Feature Vector

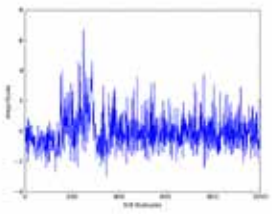
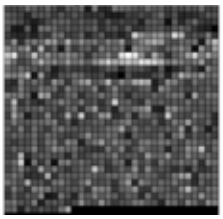


Image of fc8 Feature Vector



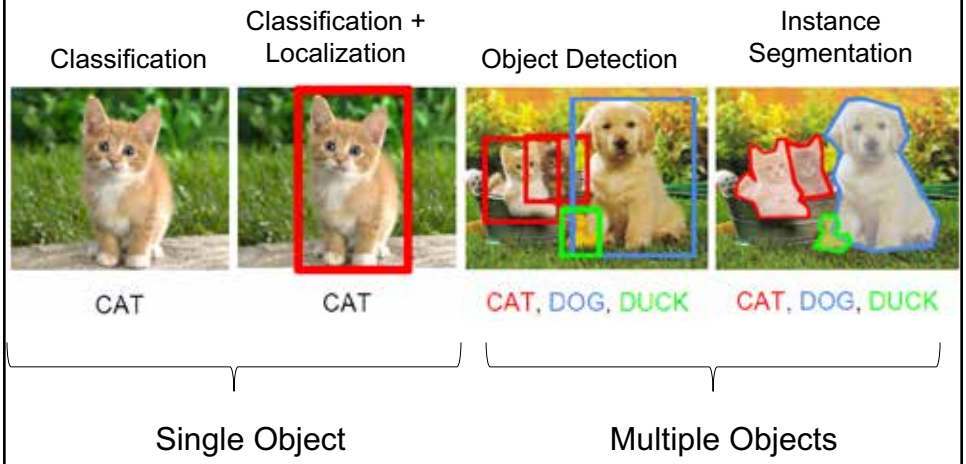
40

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.

41

Vision Tasks



53

Classification vs. Classification + Localization

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

CAT

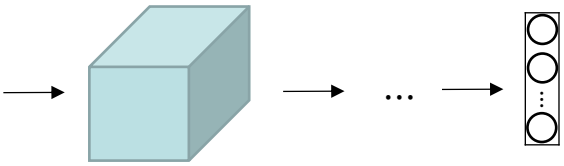
Classification + Localization
- **Input**: Image
- **Output**: Class label, Box coordinates
- **Evaluation metric**: Intersection over Union (IoU)

(CAT,x,y,w,h)

Classification with Localization

(0,0)

(1,1)



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

- For now, lets 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 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: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

- P_c: Probability of an object
- b_x, b_y, b_w, b_h: Bounding box location
- C_1, C_2, C_3, C_4: 0/1 for each class

Image from: deeplearning.ai, C4W3L01

Classification with Localization

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

Localization {x, y, w, h}

Cost function (squared error):

$$Loss = \sum_{i=1}^{9} (\hat{y}_i - y_i)^2$$ If y_1=1

$$Loss = (\hat{y}_1 - y_1)^2$$ If y_1=0

$$y = \begin{bmatrix} 1 \\ 0.3 \\ 0.6 \\ 0.4 \\ 0.3 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

? = 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

$$y = \begin{bmatrix} 1 \\ 0.3 \\ 0.6 \\ 0.4 \\ 0.3 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

? = don't care

Image from: deeplearning.ai, C4W3L01

Snapchat Facewarp?

- Traditional approach:

Viola Jones Face Detection

Search for actual point locations using Mahalanobis distance

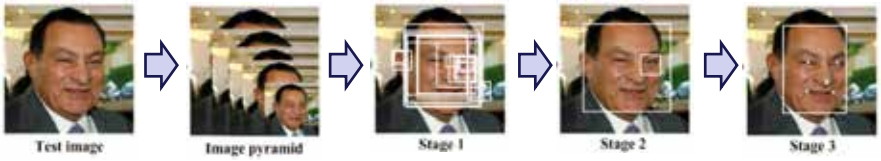
Repeat ~3-5×

Average eye and 82 facial feature points

Restrict based on PCA statistics

Snapchat Facewarp?

- Deep Learning approach:



Candidate faces

Refine face selection

Facial feature points

MT-CNN [Zhang et al. 2016]

60

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.

ptucha

61

Can do same with Body Pose…

Pishchulin et al. CVPR'16

62

Object Detection

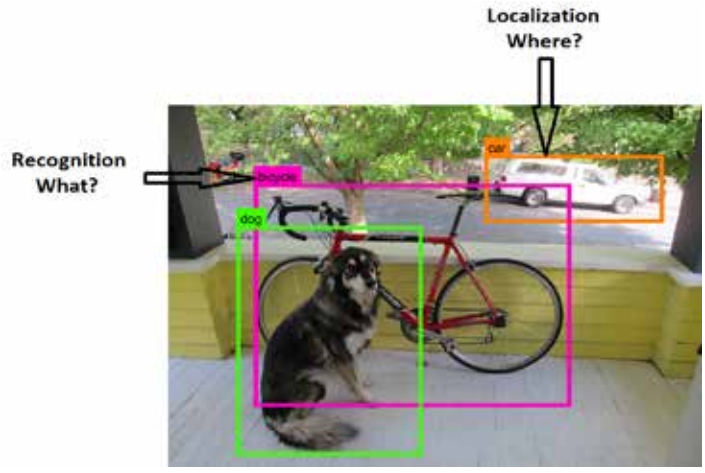


Image Credit: Redmon, Joseph, et al [4]

63

More than one object per image?

Training set:

x	y
	1
	1
	1
	0
	0

Car detection example

Images from: deeplearning.ai, C4W3L03

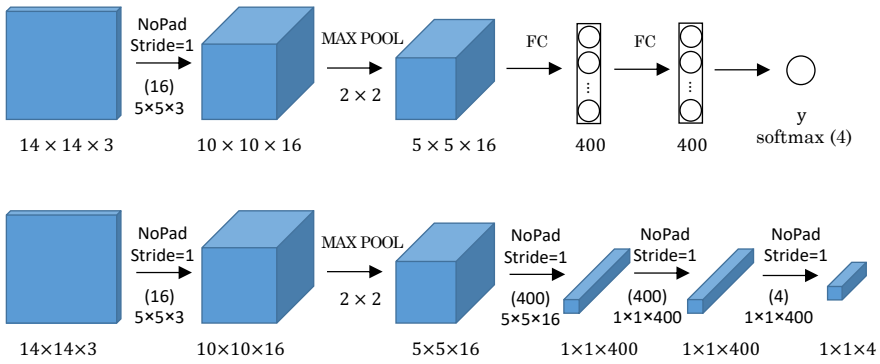
66

Sliding Window Detection

Images from: deeplearning.ai, C4W3L03

67

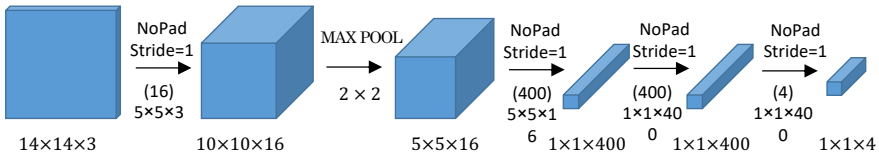
Computing FC layers with Convolution



Images from: deeplearning.ai, C4W3L04

68

Replacing Sliding Windows w/Fully Convolutional CNNs

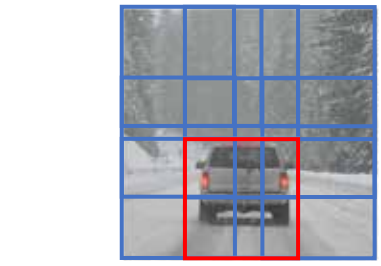


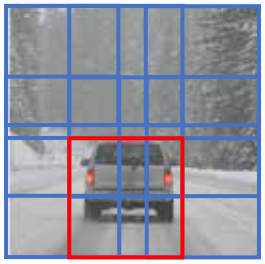
69

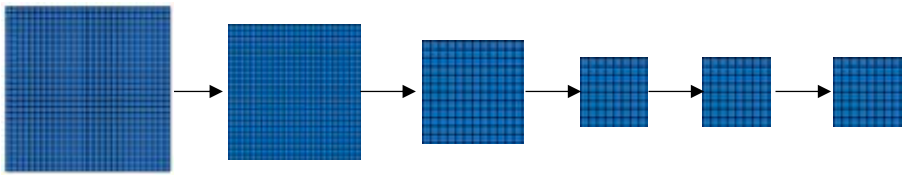
Replacing Sliding Windows w/Fully Convolutional CNNs

Sliding window approach:
Sequentially evaluate one window at a time

Fully convolutional approach:
Evaluate 64 regions at once







Images from: deeplearning.ai, C4W3L04

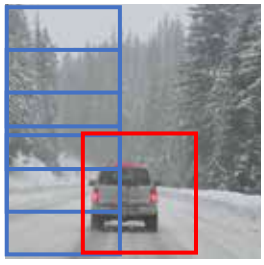
R. Ptucha '20

70

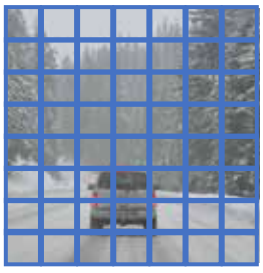
70

More Accurate Bounding Boxes

- Whether using sliding window or fully convolutional approach, bounding boxes often don't align well or may not match actual object.



$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_{pers} \\ C_{car} \\ C_{truck} \\ C_{motorc} \end{bmatrix}$$



- At each location, solve for probability of object as well as actual coordinates, which don't have to be square.

Images from: deeplearning.ai, C4W3L05

R. Ptucha '20

71

71

Replacing Sliding Windows w/Fully Convolutional CNNs

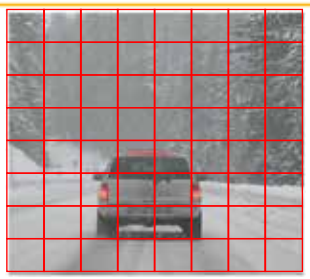
- Can think of this as evaluating 8×8 grid, where each of the 64 cells is independently checked for an object:

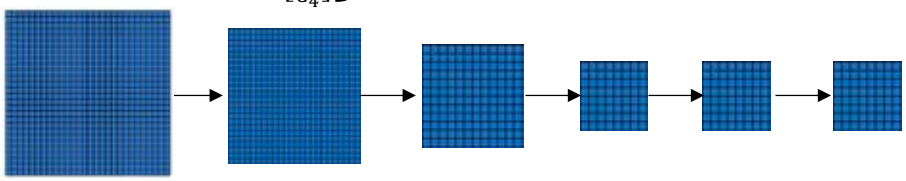
Each cell has a *y* label: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

Prob. of an object

Object location

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





R. Ptucha '20

72

72

Replacing Sliding Windows w/Fully Convolutional CNNs

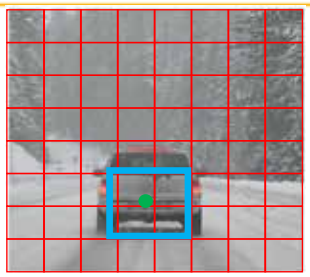
- Overlay GT of object
- Cell where centroid lie is responsible.

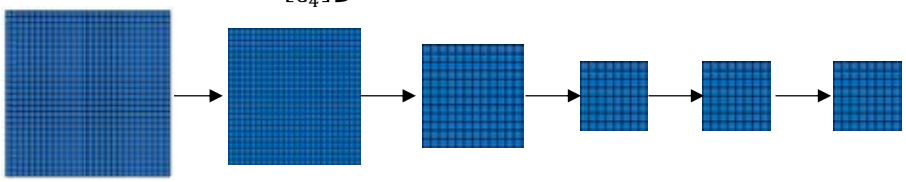
Each cell has a *y* label: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

Prob. of an object

Object location

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





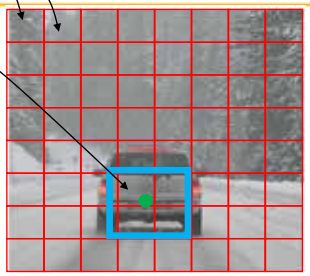
R. Ptucha '20

73

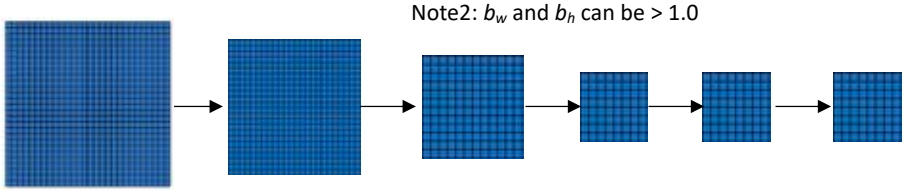
73

Replacing Sliding Windows w/Fully Convolutional CNNs

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_{pers} \\ C_{car} \\ C_{truck} \\ C_{motorc} \end{bmatrix} \quad y1 = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad y2 = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \ldots \quad y44 = \begin{bmatrix} 1 \\ 0.8 \\ 0.9 \\ 2.0 \\ 1.8 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \ldots$$



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



R. Ptucha '20

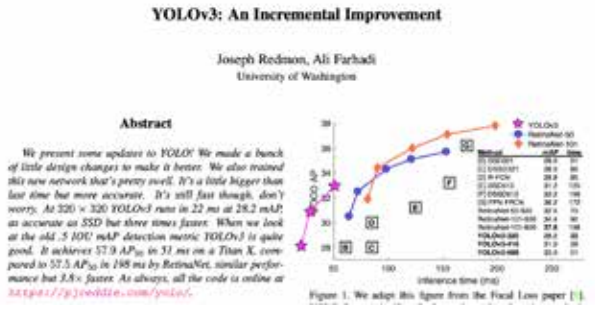
YOLO V3

https://arxiv.org/abs/1804.02767, 2018

The picture can't be displayed.

The picture can't be displayed.

The picture can't be displayed.



YOLOv3: An Incremental Improvement

Joseph Redmon, Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

https://syncedreview.com/2018/03/27/the-yolov3-object-detection-network-is-fast/

R. Ptucha '20

Mask R-CNN

https://arxiv.org/pdf/1703.06870.pdf

FAIR Mask R-CNN, COCO + Places Workshop, ICCV 2017

CNN Results

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

CIFAR, Krizhevsky '09

- 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

Ciresan '11

ImageNet

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

Russakovsky et al., 2015

IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)

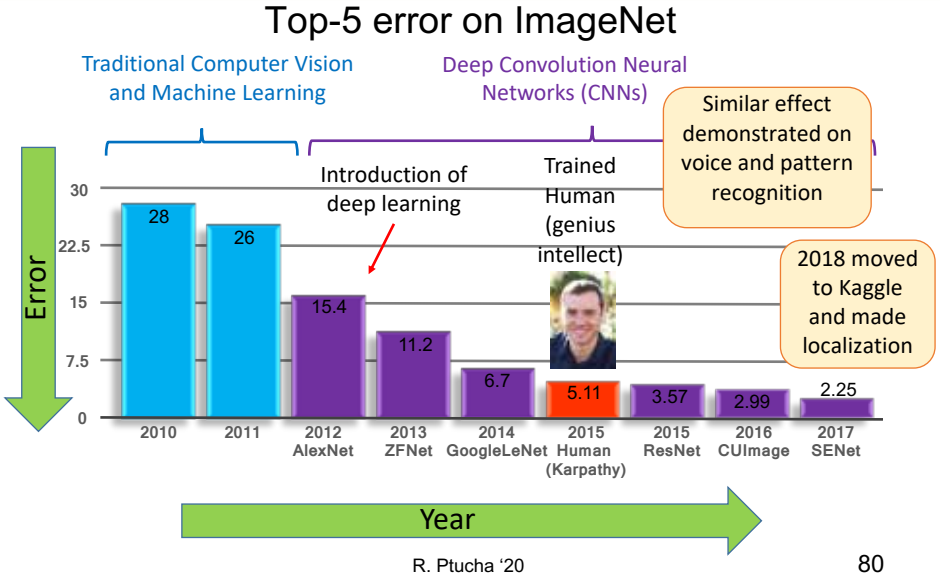
- 1.2M images, 1000 categories
- Image classification, object localization, video detection

78

ImageNet: Examples of Hammer

79

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



R. Ptucha '20 80

80

Andrew Ng, 2017

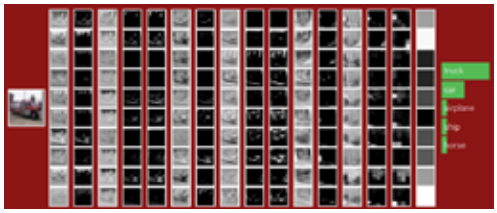
https://www.deeplearning.ai/

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

R. Ptucha '20 81

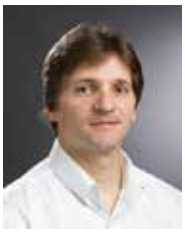
81



Li, Johnson, Yeung 2017

http://cs231n.stanford.edu/ l

For More Information: http://www.rit.edu/mil



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

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