NVIDIA Deep Learning Fundamentals

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

ICFHR 2018

The 16th International Conference on Frontiers in Handwriting Recognition

August 5 - 8, 2018 • Niagara Falls, USA

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 (Ptucha) appears on each slide you use.

Agenda

- Part I- Intuition and Theory
  - 9:00-9:45am: Introduction
  - 9:45-10:30am: Convolutional Neural Networks
- 10:30-10:45pm: Break
- Part II- Hands on
  - 10:45am-Noon: Hands-on exercises

Navigating to Qwiklabs

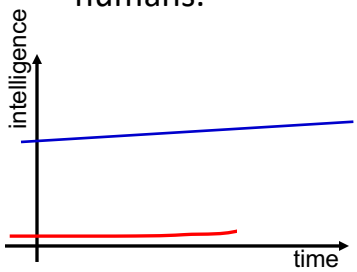
- Have you registered for NVIDIA account yet?
  1. Navigate to: https://nvlabs.qwiklab.com
  2. Login or create a new account
- Select event:
  - ICFHC CV Ambassador Workshop
- Then select class:
  - Image Classification with DIGITS
- You can return to class for up to 60 days

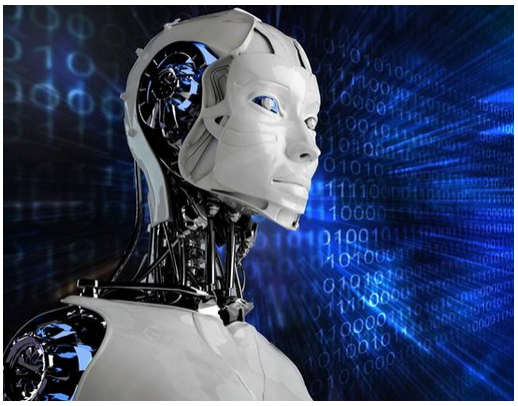
Machine Learning

- Machine learning is giving computers the ability to analyze, generalize, think/reason/behave like humans.
- Machine learning is transforming medical research, financial markets, international security, and generally making humans more efficient and improving quality of life.
- Inspired by the mammalian brain, deep learning is machine learning on steroids- bigger, faster, better!

The point of Singularity

- The point of singularity is when computers become smarter than humans.





Evolution of biology

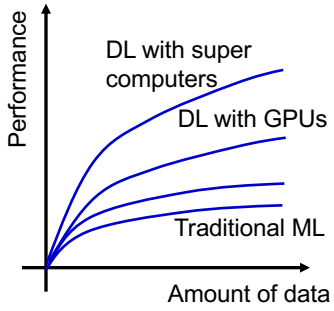
Advancement of technology

Unleashing of Intelligence

- Machines will slowly match, then quickly surpass human capabilities.
- Today it is exciting/scary/fun to drive next to an autonomous car.
- Tomorrow it may be considered irresponsible for a human to relinquish control from a car that has faster reaction times, doesn't drink/text/get distracted/tired, and is communicating with surrounding vehicles and objects.

Why is AI (Deep Learning) Just Now Becoming Practical in Many Day-to-Day Situations?

- Availability of data;
- Sustained advances in hardware capabilities (including GPUs running machine learning workloads);
- Omnipresent connectivity;
- Lower cost and power consumption;
- Sustained advances in algorithmic learning techniques.



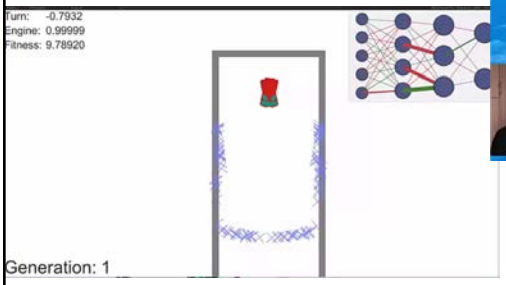
Hot trend:
High performance architecture experts teaming up with deep learning experts

2017: The Year of AI:

The Wall Street Journal, Forbes, and Fortune

NEC Face Recognition

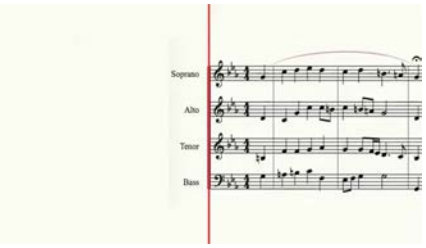
SONY Playstation Virtual Reality



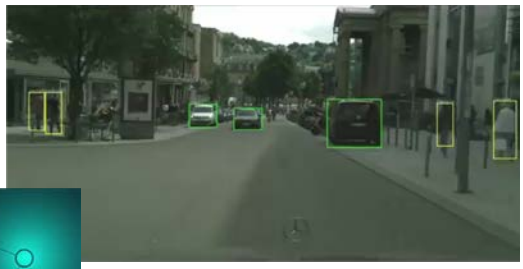
Evolutionary Reinforcement Learning

2017: The Year of AI:

The Wall Street Journal, Forbes, and Fortune



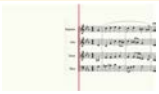
DeepBach



NVIDIA Autonomous Car
Detection & Segmentation

YOLO v2 Object Detection

Some Things to Look for in 2018



CelebA-HQ
1024 × 1024

Progressive growing

CelebA-HQ
1024 × 1024

Latent space interpolations

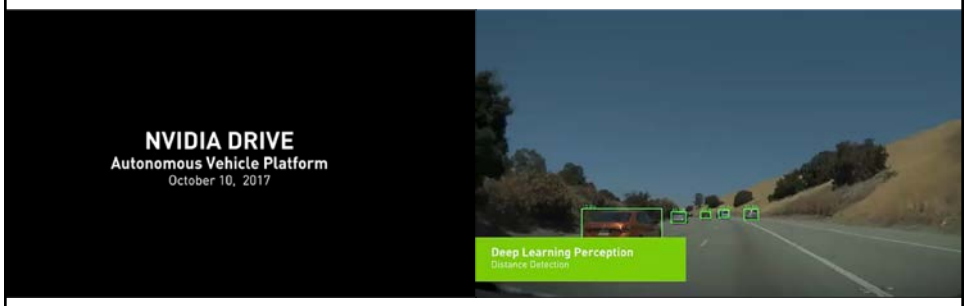
http://research.nvidia.com/sites/default/files/pubs/2017-10_Progressive-Growing-of/karras2017gan-paper.pdf

Some Things to Look for in 2018

Faceshift GDC

Apple iPhone X, Animoji Yourself

Some Things to Look for in 2018



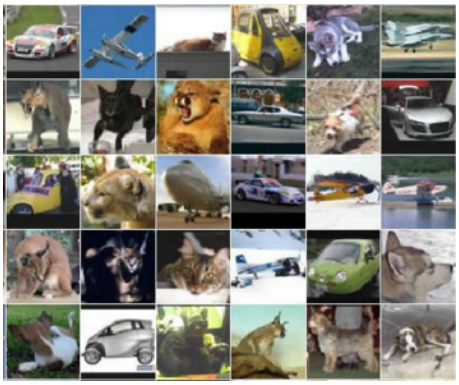
NVIDIA Drive

The Human Brain

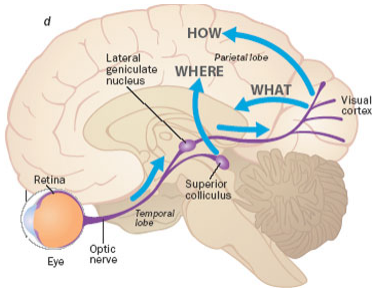
- We've learned more about the brain in the last 5 years than we have learned in the last 5000 years!
- It controls every aspect of our lives, but we still don't understand exactly how it works.

The Brain on Pattern Recognition

- Airplane, Cat, Car, Dog



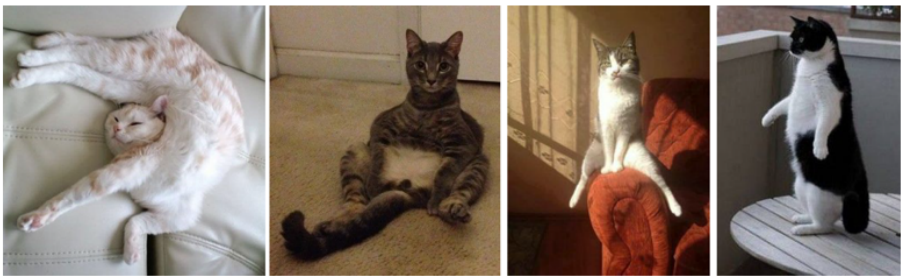
STL-10 dataset



http://thebraingeek.blogspot.com/2012/08/blindsight.html

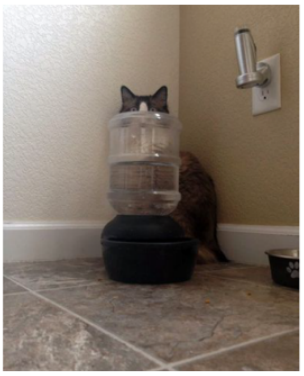
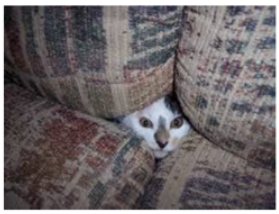
The Brain on Pattern Recognition

Despite Changes in Deformation:



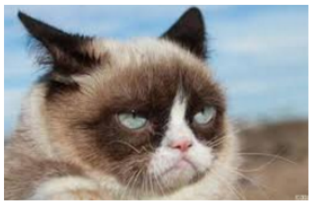
The Brain on Pattern Recognition

Despite Changes in Occlusion:

The Brain on Pattern Recognition

Despite Changes in Size, Pose, Angle:






Tardar Sauce "Grumpy Cat"

The Brain on Pattern Recognition

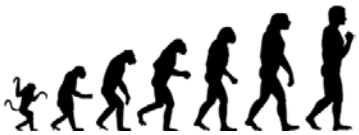
Despite Changes in Background Clutter:

The Brain on Pattern Recognition

Despite Changes in Class Variation…

Teaching Computers to See

- It took evolution 540M years to develop the marvel of the eye-brain.



- Lets say a child collects a new image every 200msec.
- By age 3, this child has processed over 100M images.

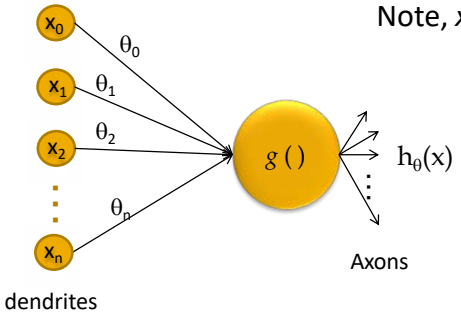
$$(5\ images/sec)(60sec/min)(60min/hr)(12hr/day)(365days/yr)(3yrs) = 236M$$

- Today's computers can do this in a few days…

Neural Nets on Pattern Recognition

- Instead of trying to code simple intuitions/rules on what makes an airplane, car, cat, and dog…
- We feed neural networks a large number of training samples, and it will automatically learn the rules!
- We will learn the magic behind this today!

Artificial Neuron



Note, x_0 is the bias unit, x_0=1

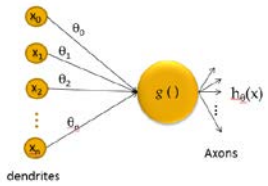
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

Activation function

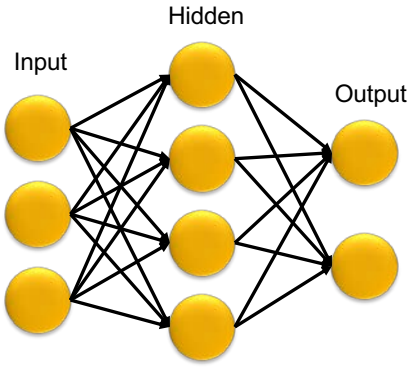
$$h_\theta(x) = g(x_0\theta_0 + x_1\theta_1 + \ \ldots + x_n\theta_n) = g\left(\sum_{i=0}^{n} x_i\theta_i\right)$$

$$h_\theta(x) = g(\theta^T x)$$

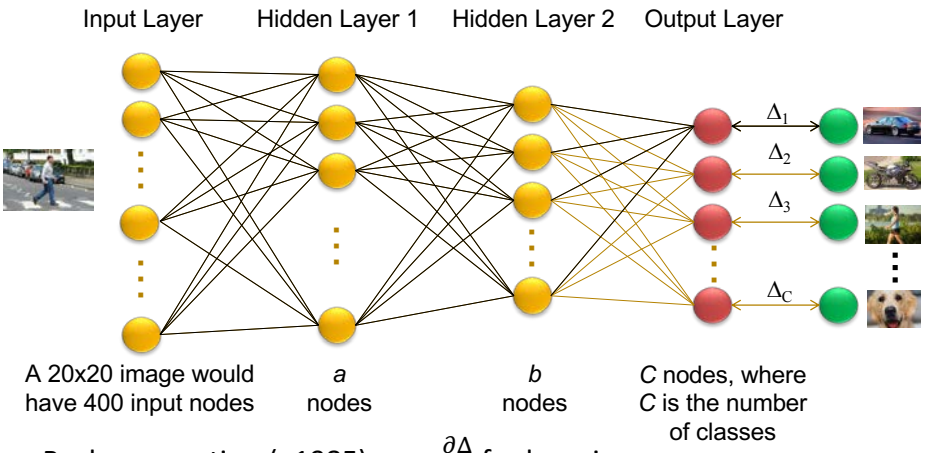
Artificial Neural Networks



- Artificial Neural Network (ANN) – A network of interconnected nodes that "mimic" the properties of a biological network of neurons



4-Layer ANN Fully Connected Topology

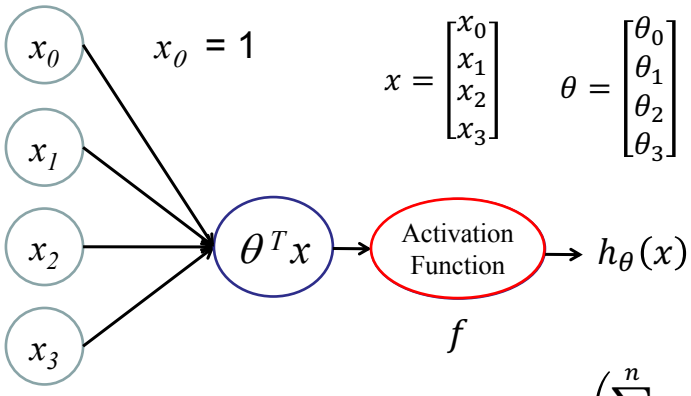


Backpropagation (~1985) uses $\frac{\partial \Delta}{\partial w}$ for learning

Learning happens in the weights- each line is a weight.

Neuron Model

Bias unit



$x_0 = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_\theta(x) = g(x_0\theta_0 + x_1\theta_1 + \ldots + x_n\theta_n) = g\left(\sum_{i=0}^{n} x_i\theta_i\right)$$

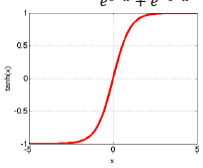
$$h_\theta(x) = g(\theta^T x)$$

Activation Function Comparison

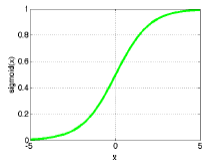
- Tanh
- Sigmoid

Gradient of both saturates at zero. Sigmoid also non-zero centered, so in practice tanh performs better.

$h_\theta(x) = \frac{e^{\theta^T x} - e^{-\theta^T x}}{e^{\theta^T x} + e^{-\theta^T x}}$

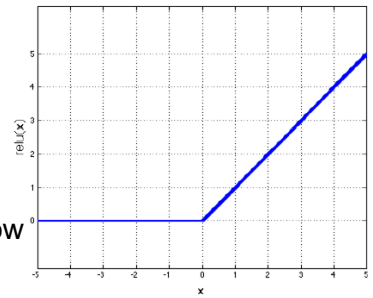


$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$

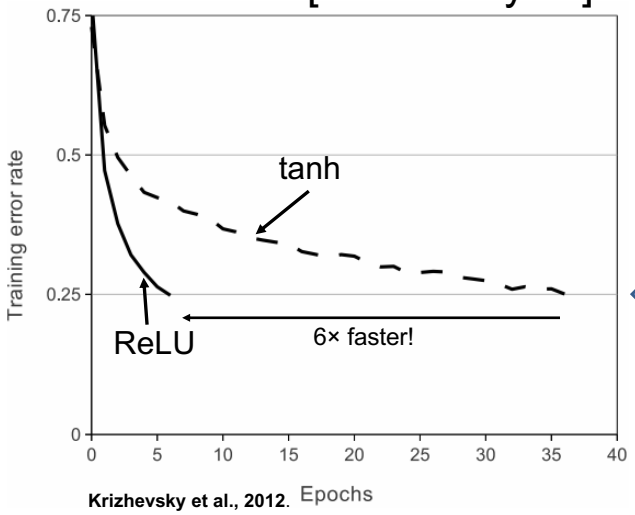


- Rectified Linear Units (ReLU)
  - Better for high dynamic range
  - Faster learning
  - Overall better result
  - Neurons can "die" if allowed to grow unconstrained

$h_\theta(x) = max(0, x)$



Tanh vs. ReLU on CIFAR-10 dataset [Krizhevsky'12]



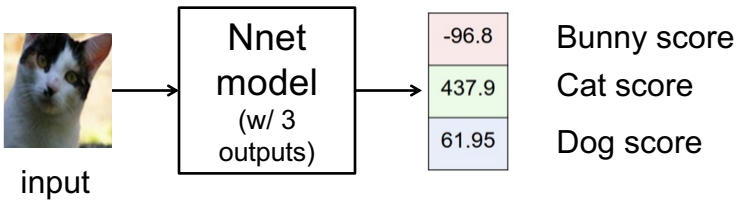
Krizhevsky et al., 2012.

ReLU reaches 25% error 6× faster!
Note: Learning rates optimized for each, no regularization, four layer CNN.

Where Do Weights Come From?

- The weights in a neural network need to be learned such that the errors are minimized.
- Just like logistic regression, we can write a cost function.
- Similar to gradient descent, we can write an iterative procedure to update weights, with each iteration decreasing our cost.
- These iterative methods may be less efficient than a direct analytical solution, but are easier to generalize.

Multiclass Loss Functions



- The input image scores highest against cat, but is also somewhat similar to dog.
- How do we assign a loss function?

Activation Function of Output Layer

- Sigmoid returns 0 or 1 for each output node.
- What if you wanted a confidence interval?
- Use a linear activation function for regression: $a^{(l)}=z^{(l)}$
- Softmax often used for classification:

$$a_c^{(L)} = h_\theta(x)_c = g\left(z_c^{(L)}\right) = \frac{exp\left(z_c^{(L)}\right)}{\sum_{c=1:C} exp\left(z_c^{(L)}\right)}$$

← *exp()* of each output node

← Sum of all output nodes

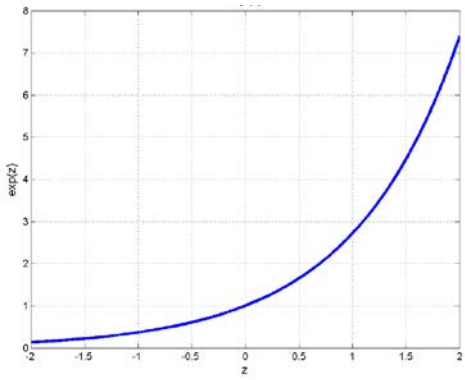
- Note: Only the output layer activation function changes- all hidden layer nodes activation functions would be the sigmoid/tanh/ReLU function.

input

Nnet model (w/ 3 outputs)

-0.97	Bunny score	0.38	0.46
4.38	Cat score	79.8	97.3
0.62	Dog score	1.86	2.26

exp(z), where z is each output node



$exp\left(z_c^{(L)}\right)$

$$\frac{exp\left(z_c^{(L)}\right)}{\sum_{c=1:C} exp\left(z_c^{(L)}\right)} \times 100$$

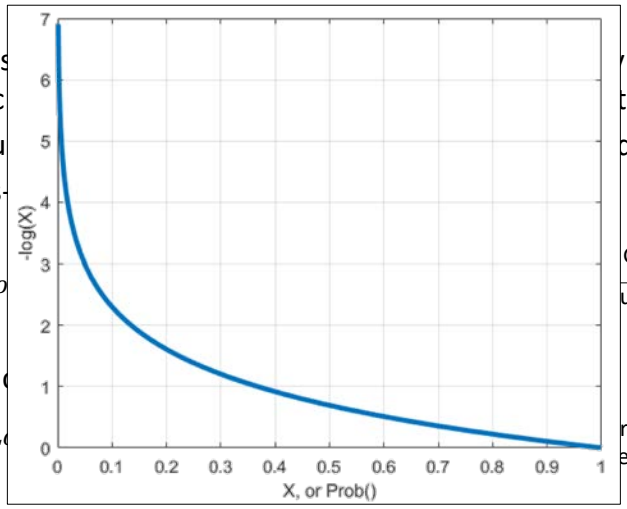
exp() take in values +/- values and returns all positive values.

Most Common Loss Functions

- The cos… from logistic… tion.
- For mu… ds:

1. Cross-…

$Loss^{(i)} = -lo…$ (GT node) / …ut) of all nodes



2. Multi…

$L…$ …rect – …es

Most Common Loss Functions

- The cost function we previously used was a direct copy from logistic regression and works great for binary classification.
- For multi-class, there are two popular data loss methods:

1. Cross-entropy loss, which uses softmax:

$$Loss^{(i)} = -log\left(\frac{exp\left(out_{yi}^{(i)}\right)}{\sum_{c=1:C} exp\left(out_c^{(i)}\right)}\right)$$

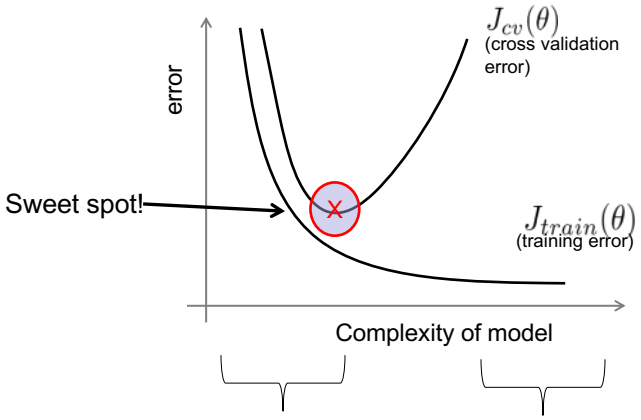
Loss for sample i = $\frac{\text{exp(output of GT node)}}{\text{Sum of exp(output) of all nodes}}$

2. Multiclass SVM Loss (Weston Watkins formulation):

$$Loss^{(i)} = \sum_{j \neq yi} max\left(0, out_j - out_{yi} + \Delta\right)$$

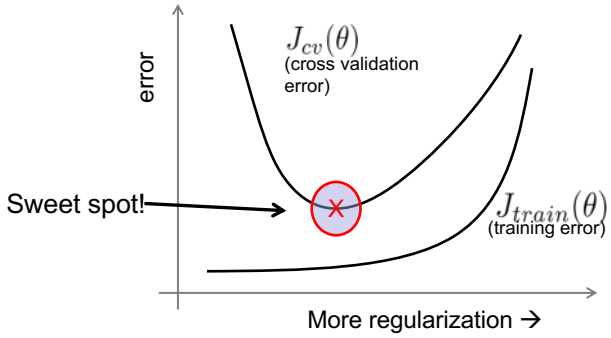
Sum of incorrect – correct classes

Bias (underfit) vs. Variance (overfit) errors



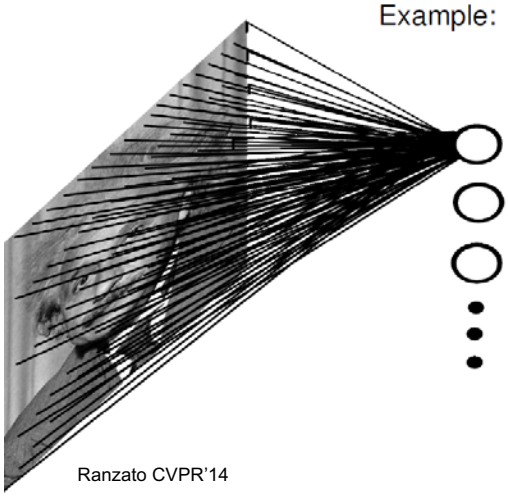
Adopted from: Andrew Ng, ML class

Regularization Tuning



Adopted from: Andrew Ng, ML class

Fully Connected Layers?

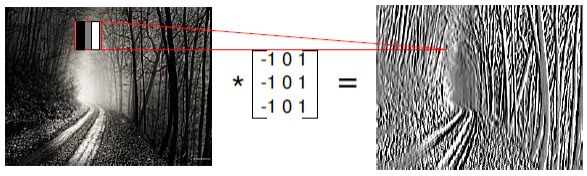


Ranzato CVPR'14

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.

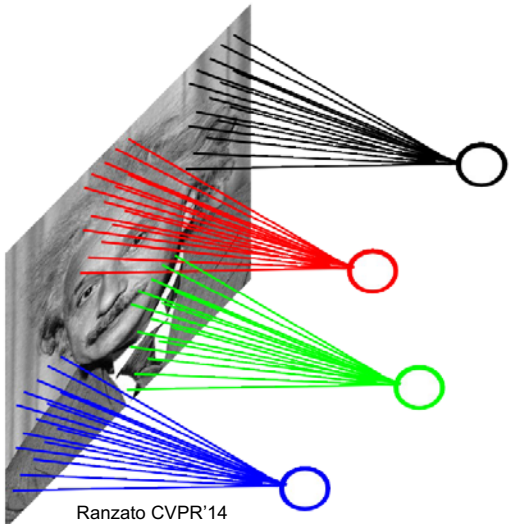
Convolution Filter



Ranzato CVPR'14

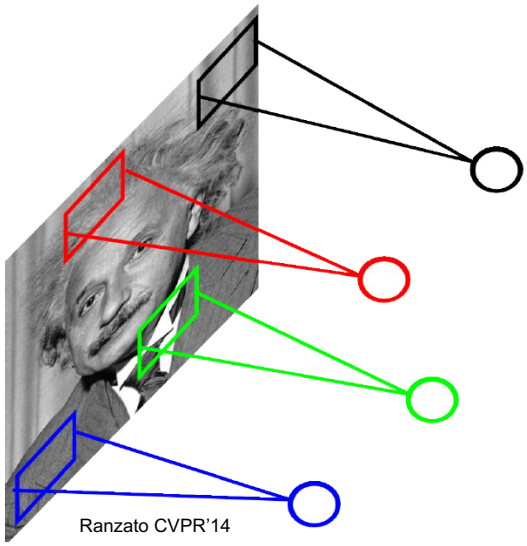
- Convolution filters apply a transform to an image.
- The above filter detects vertical edges.

Locally Connected Layer



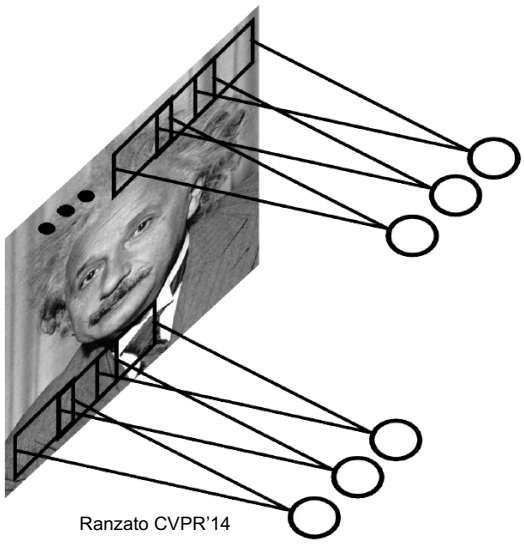
- 200×200 pixel image.
- 40K input.
- Four 10×10 filters, each fully connected
- 40K×10×10×4=16M weights….getting better!

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

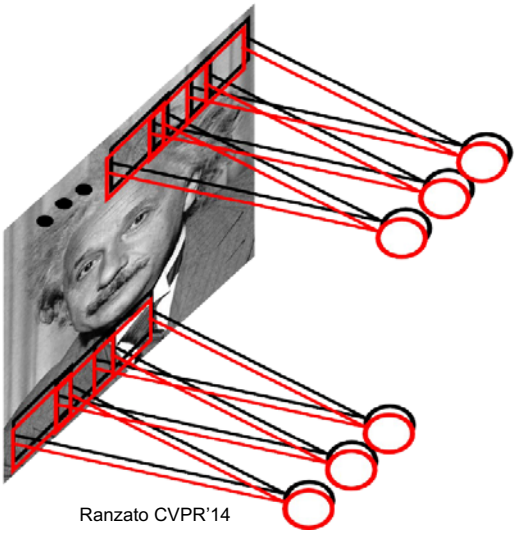


Ranzato CVPR'14

- 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.

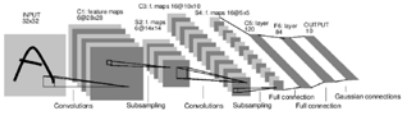
Convolution Layer



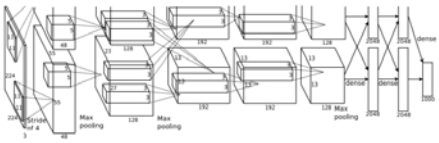
Ranzato CVPR'14

- 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!

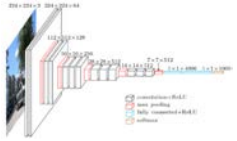
Many Flavors of Convolution Neural Networks (CNNs)…



LeNet-5, LeCun 1989



AlexNet, Krizhevsky 2012



VGGNet, Simonyan 2014

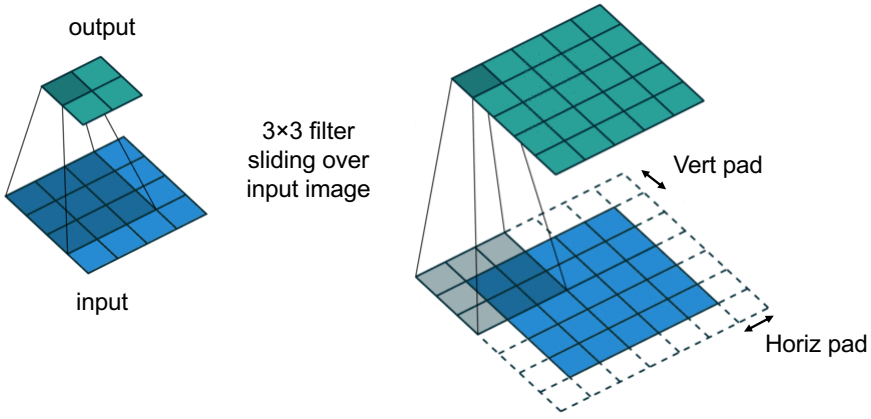
GoogLeNet (Inception), Szegedy 2014

ResNet, He 2015

DenseNet, Huang 2017

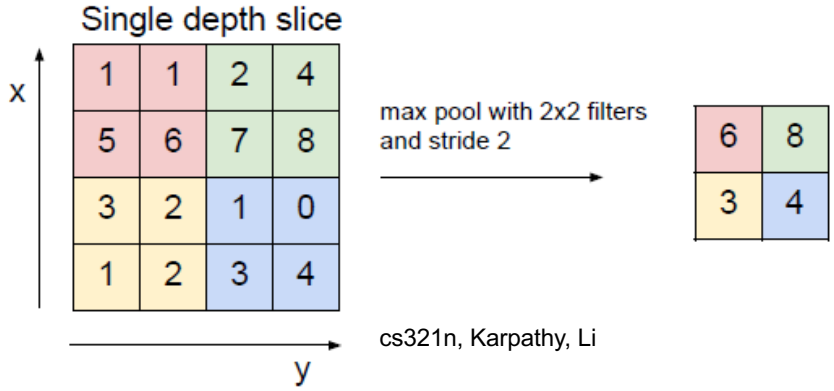
Image Convolution

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



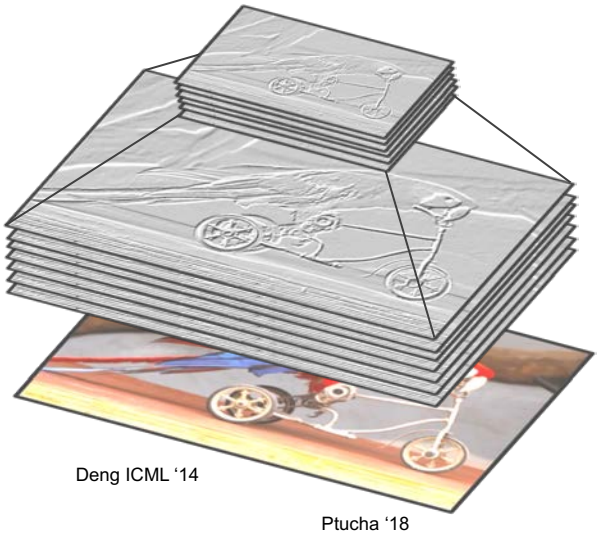
https://github.com/vdumoulin/conv_arithmetic

Max Pooling- Reducing the Size of an Image



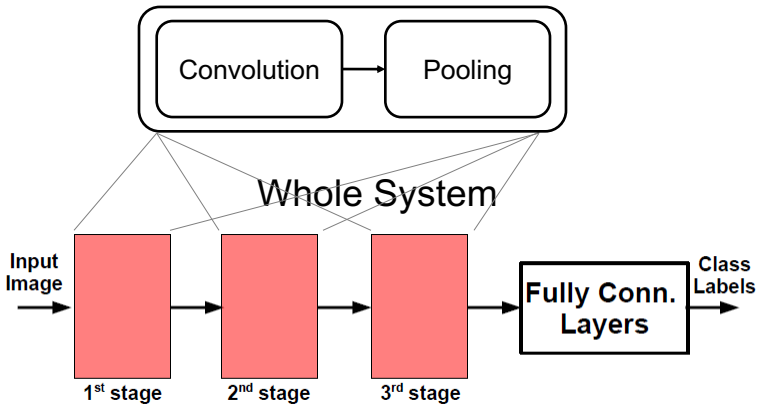
cs321n, Karpathy, Li

Convolution Neural Network (CNN) Building Block



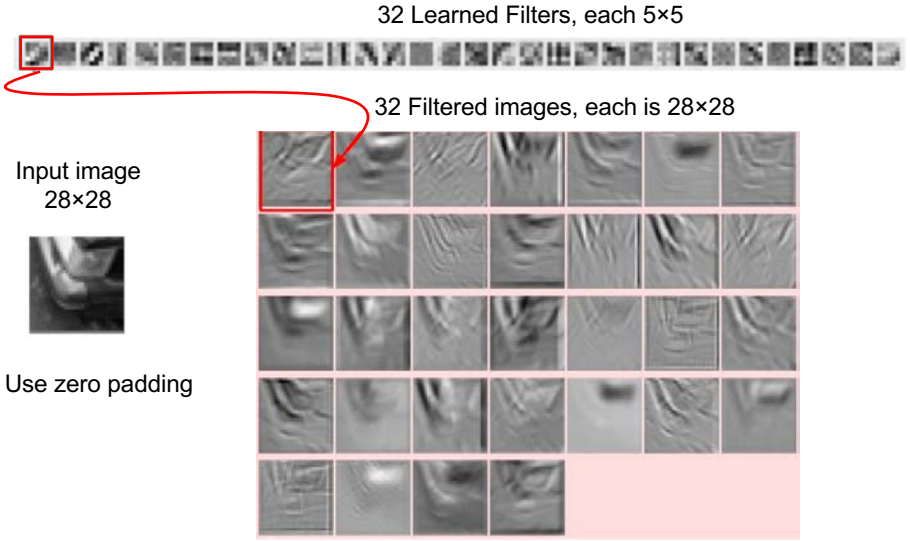
Deng ICML '14

Putting it All Together



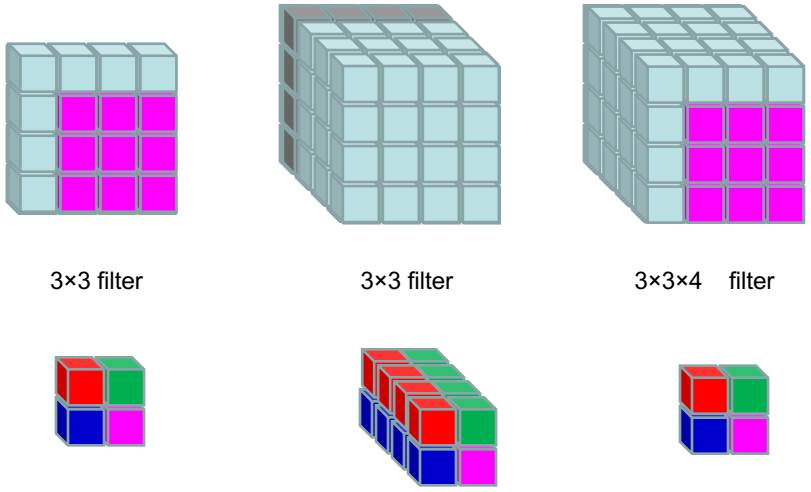
Ptucha '18 76

Learning Filters



Ptucha '18 80

Filters



3×3 filter

3×3 filter

3×3×4 filter

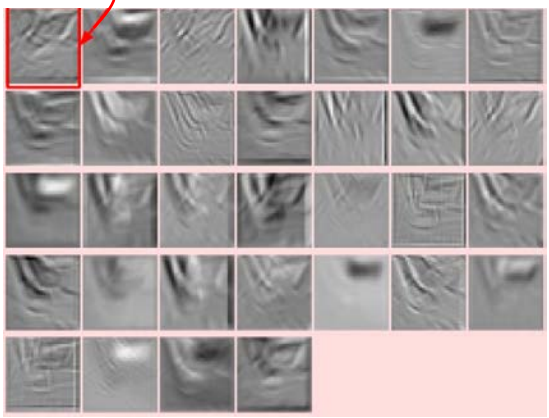
Ptucha '18

81

Learning Filters

32 Learned Filters, each 5×5×3

32 Filtered images, each is 28×28×1



Input image
28×28×3

Use zero padding

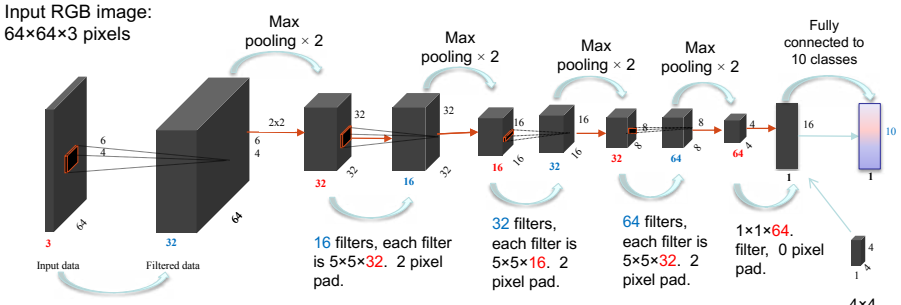
Ptucha '18

82

CNN Architecture

(Not so) Toy Example

Output: prediction of 1 of 10 categories



Case Study

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

best model

cs321n, Karpathy, Li

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					

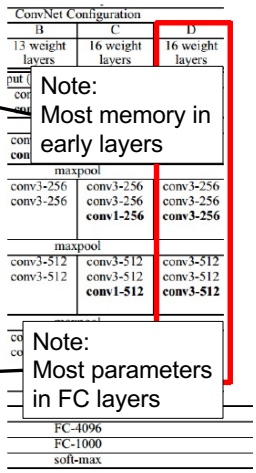
Table 2: **Number of parameters** (in millions).

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

Case Study

(not counting biases)

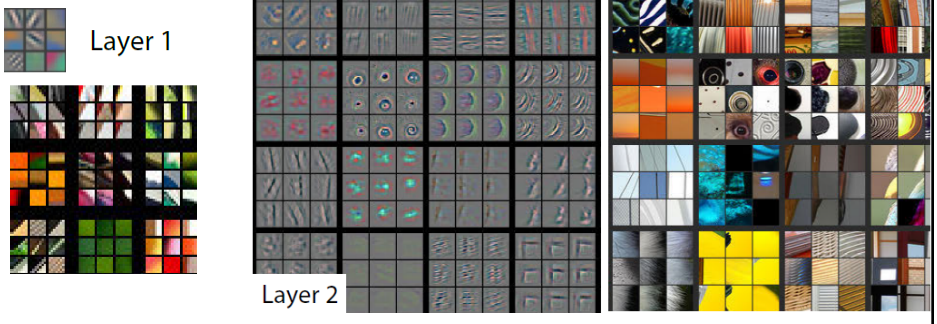
INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M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

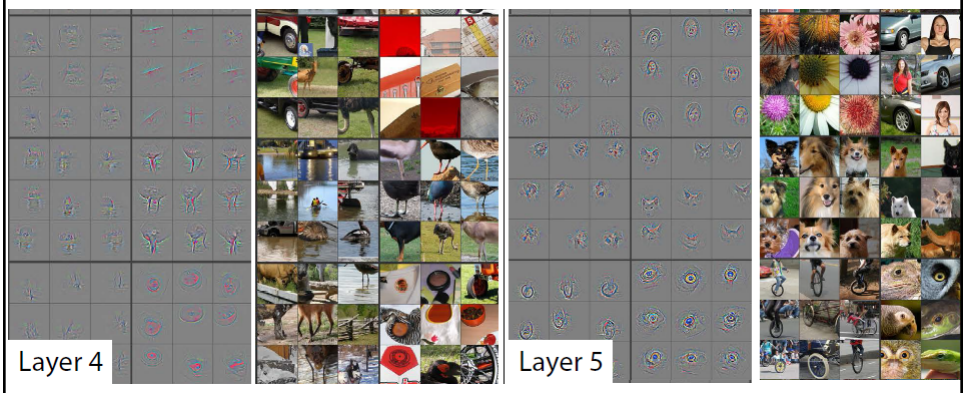
Note:
Most parameters in FC layers

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

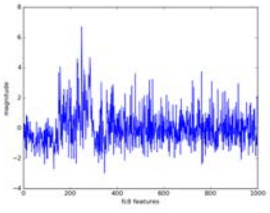
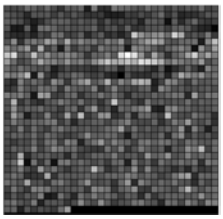


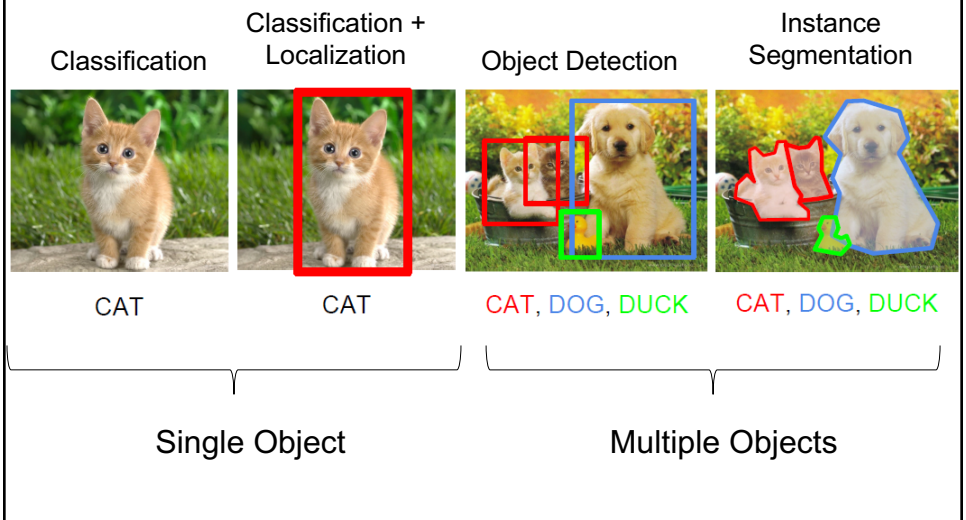
Image of fc8 Feature Vector



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 (image2vec) as input to a SVM classifier.
- Images in this vector space generally have the property that similar images are close in this latent representation.

Vision Tasks



Classification vs. Classification + Localization

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

→ CAT

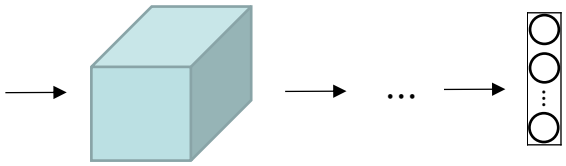
Classification + Localization
- **Input**: Image
- **Output**: Class label, Box coordinates
- **Evaluation metric**: Intersection over Union (IoU)

→ (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

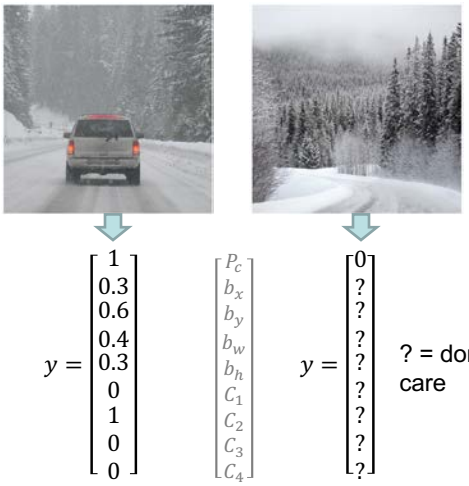
Image from: deeplearning.ai, C4W3L01

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

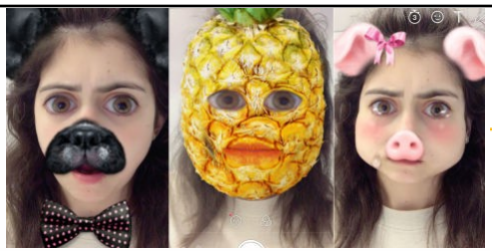


$$y = \begin{bmatrix} 1 \\ 0.3 \\ 0.6 \\ 0.4 \\ 0.3 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

? = don't care

Image from: deeplearning.ai, C4W3L01

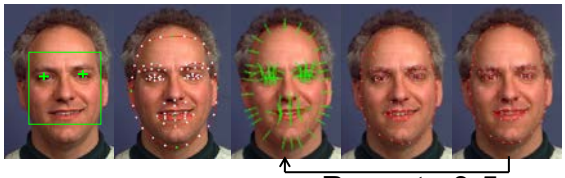
Snapchat Facewarp?



- Traditional approach:

Viola Jones Face Detection

Search for actual point locations using Mahalanobis distance

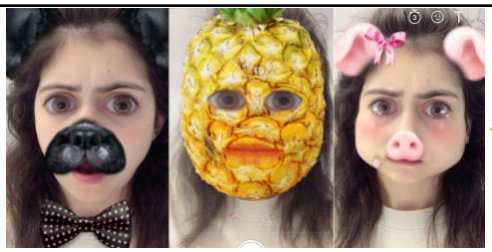


Repeat ~3-5×

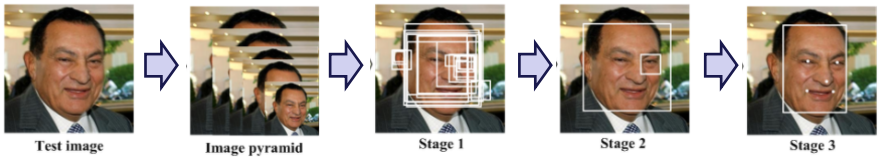
Average eye and 82 facial feature points

Restrict based on PCA statistics

Snapchat Facewarp?



- Deep Learning approach:



Candidate faces

Refine face selection

Facial feature points

MT-CNN [Zhang et al. 2016]

Localization

- Facial feature

Each face has 68 points, so CNN would output:

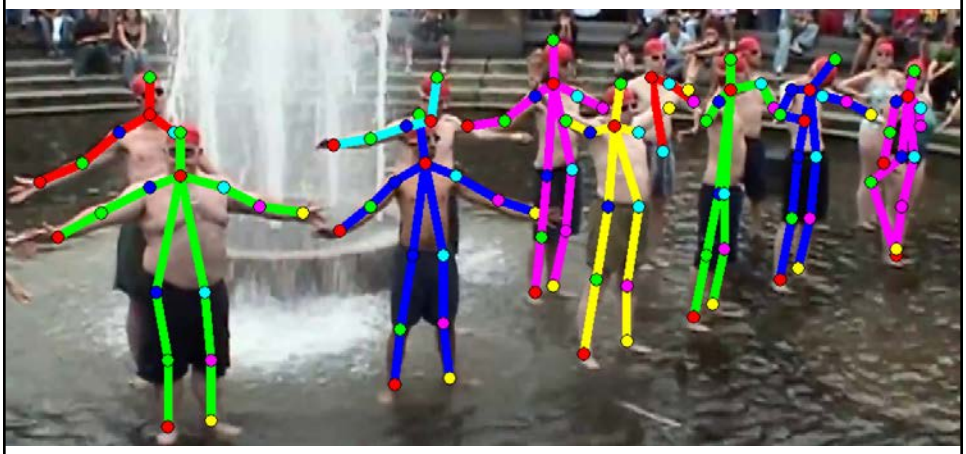
Face?
pt1X
pt1Y
pt2X
pt2Y
.
.
.
pt68X
pt68Y

137 outputs

Of course, need GT for thousands of faces to train model.

ptucha

Can do same with Body Pose…



Pishchulin et al. CVPR'16

Object Detection

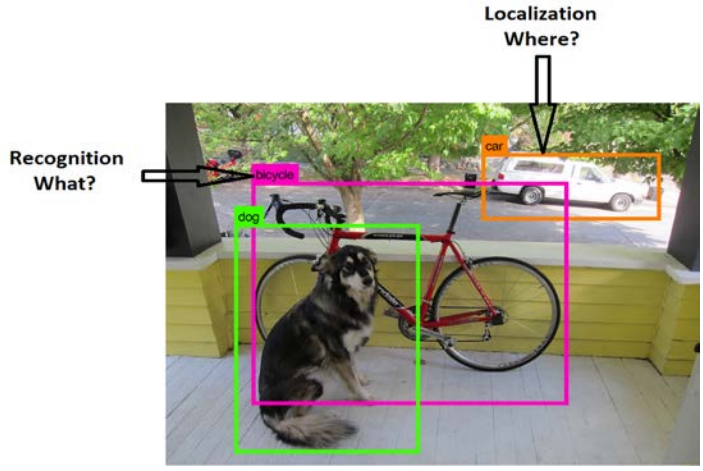


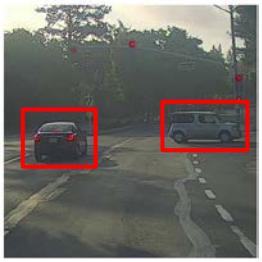
Image Credit: Redmon, Joseph, et al [4]

More than one object per image?

Training set:

x	y
	1
	1
	1
	0
	0

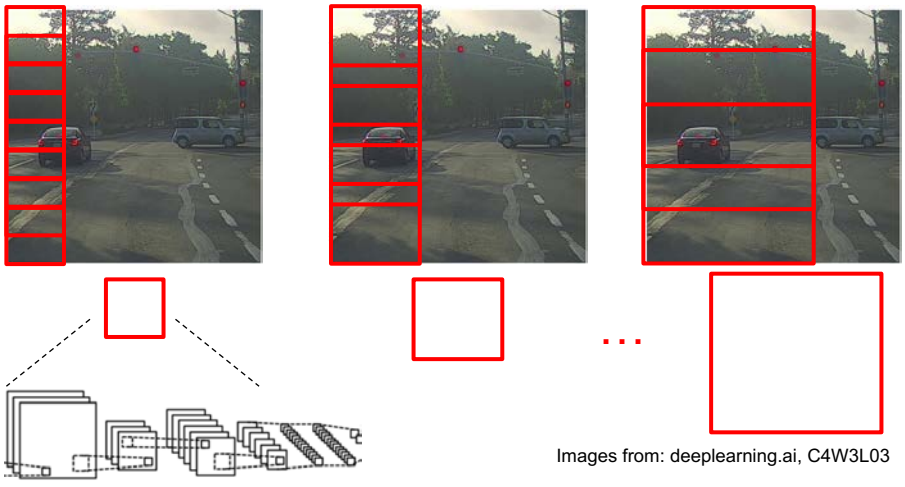
Car detection example



Images from: deeplearning.ai, C4W3L03

Ptucha '18 121

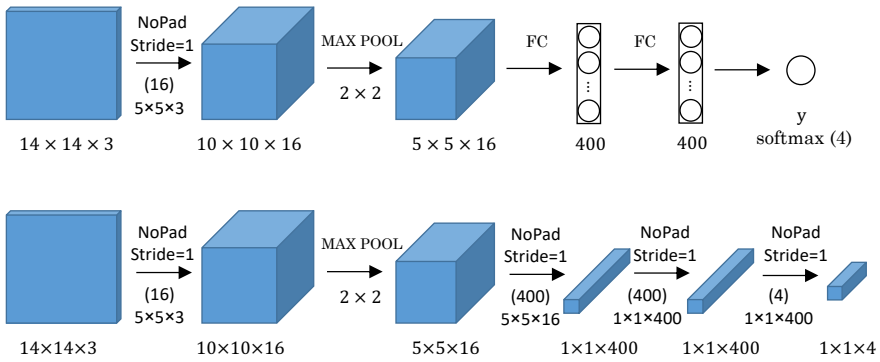
Sliding Window Detection



Images from: deeplearning.ai, C4W3L03

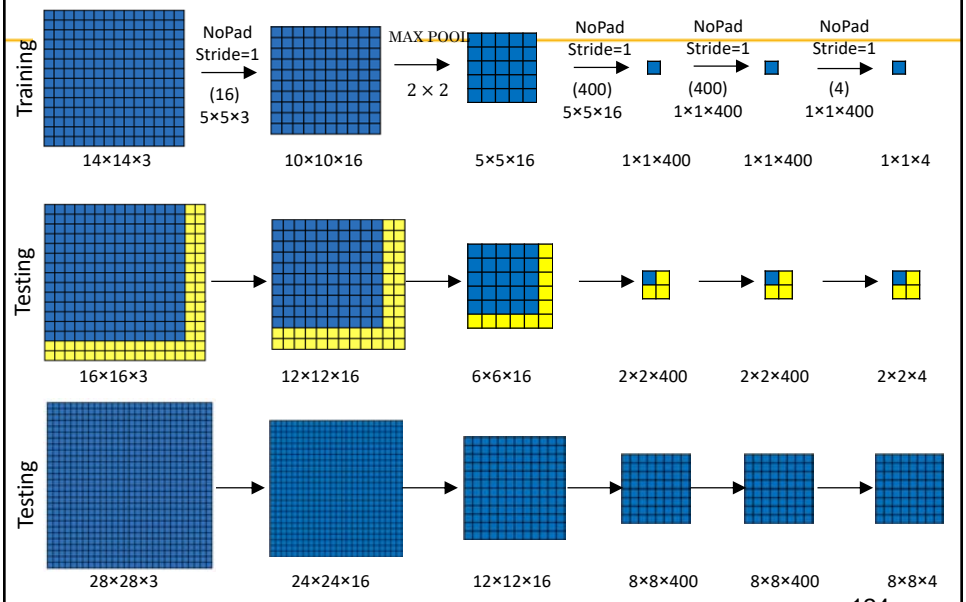
ptucha Ptucha '18 122 122

Computing FC layers with Convolution



Images from: deeplearning.ai, C4W3L04

Replacing Sliding Windows w/Fully Convolutional CNNs

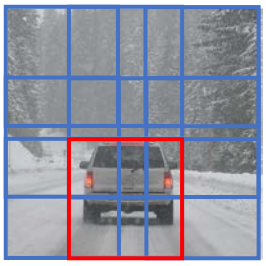


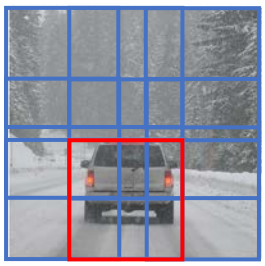
Images from: deeplearning.ai, C4W3L04

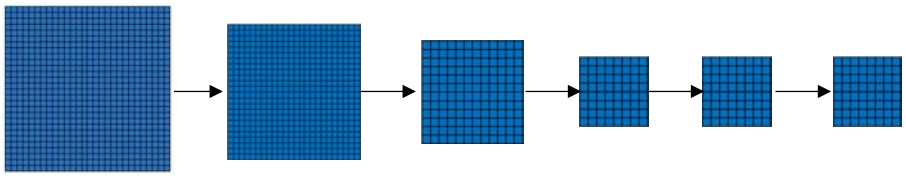
Replacing Sliding Windows w/Fully Convolutional CNNs

Sliding window approach:
Sequentially evaluate one window at a time

Fully convolutional approach:
Evaluate 64 regions at once







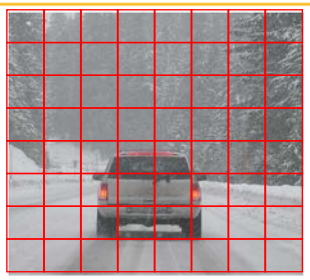
Ptucha '18

125

Images from: deeplearning.ai, C4W3L04

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:



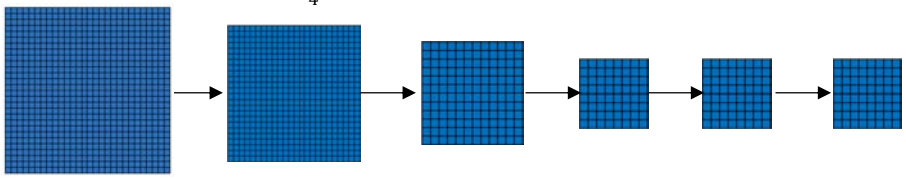
Each cell has a *y* label:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$$

Prob. of an object

Object location

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



Ptucha '18

126

Replacing Sliding Windows w/Fully Convolutional CNNs

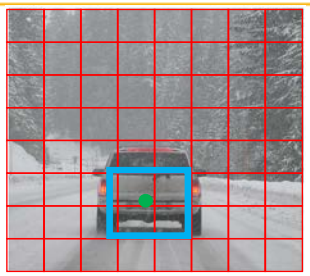
- Overlay GT of object
- Cell where centroid lie is responsible.

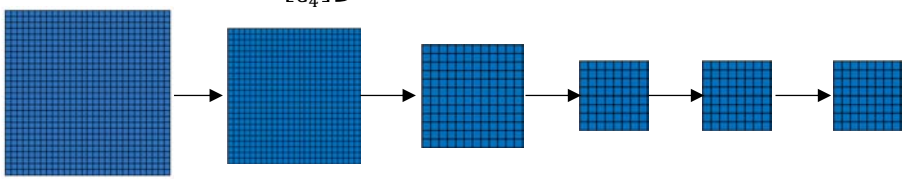
Each cell has a *y* label: $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_w \\ b_h \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$

Prob. of an object

Object location

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



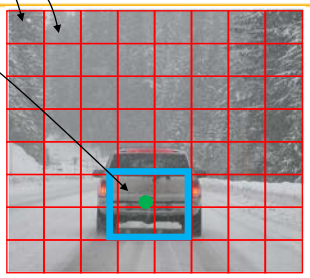


Ptucha '18

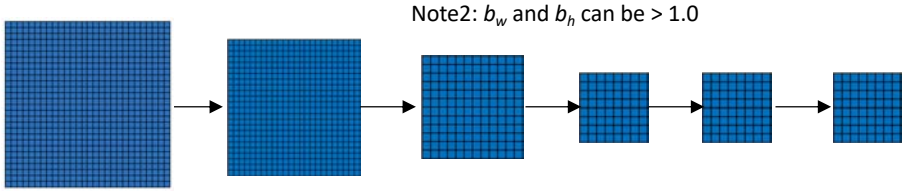
127

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



Ptucha '18

128

deeplearning.ai | coursera

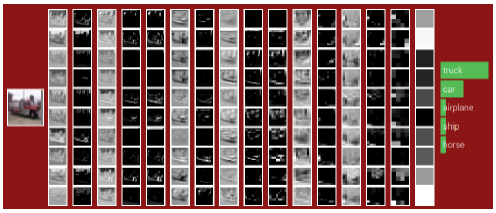
Andrew Ng, 2017

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/

www.nvidia.com/dli

https://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