Neural Networks

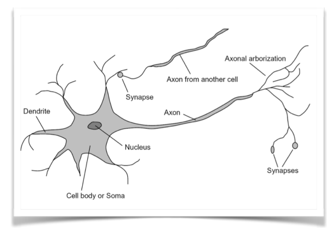
Subhransu Maji

CMPSCI 670: Computer Vision

November 8, 2016

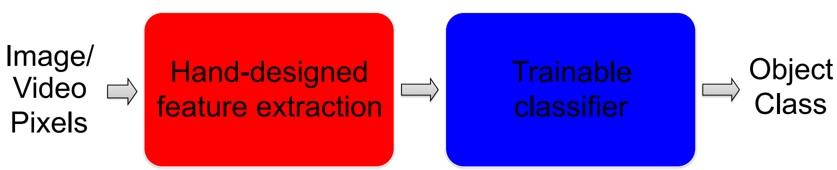
Motivation

- One of the main weakness of linear models is that they are linear
- Decision trees can model non-linear boundaries
- Neural networks are yet another non-linear classifier
- Take the biological inspiration further by chaining together perceptrons
- Allows us to use what we learned about linear models:
  - Loss functions, regularization, optimization



neuron

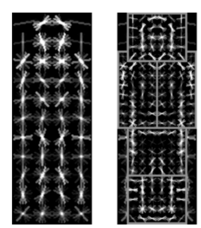
Traditional recognition approach



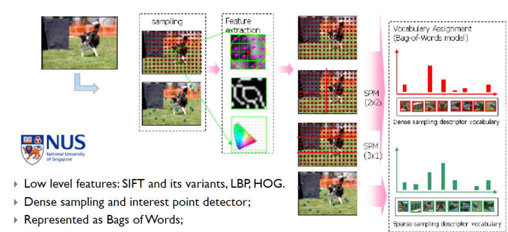
- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

Traditional recognition approach

- Features are key to recent progress in recognition
- Multitude of hand-designed features currently in use
  - SIFT, HOG, …………
- Where next? Better classifiers? Or keep building more features?



Felzenszwalb, Girshick, McAllester and Ramanan, PAMI 2007



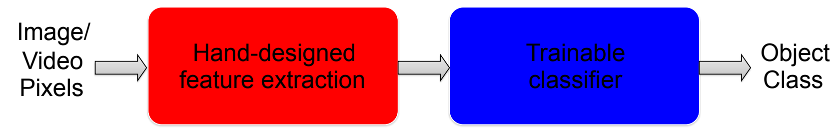
Yan & Huang (Winner of PASCAL 2010 classification competition)

What about learning the features?

- Learn a ***feature hierarchy*** all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly

"Shallow" vs. "deep" architectures

Traditional recognition: "Shallow" architecture



Deep learning: "Deep" architecture

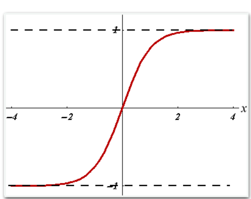
Two-layer network architecture

$$y = \mathbf{v}^T \mathbf{h}$$

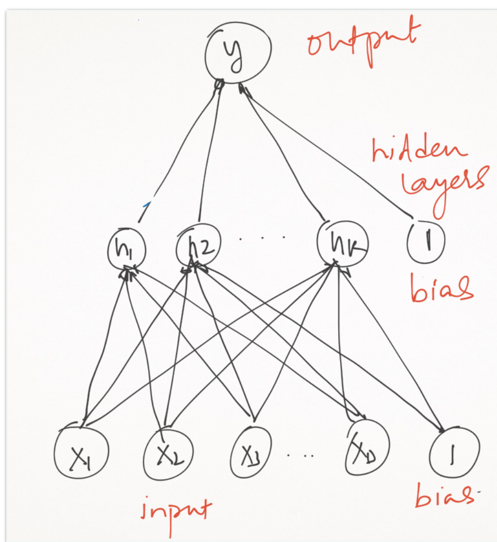
Non-linearity is important

link function

$$h_i = f(\mathbf{w}_i^T \mathbf{x})$$



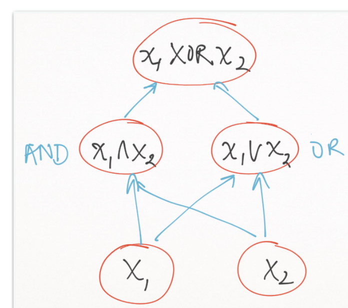
$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$



The XOR function

- We saw that a perceptron cannot learn the XOR function
- **Exercise:** come up with the parameters of a two layer network with two hidden units that computes the XOR function
  - Here is a table with a bias feature for XOR

y	x_0	x_1	x_2
+1	+1	+1	+1
+1	+1	-1	-1
-1	+1	+1	-1
-1	+1	-1	+1



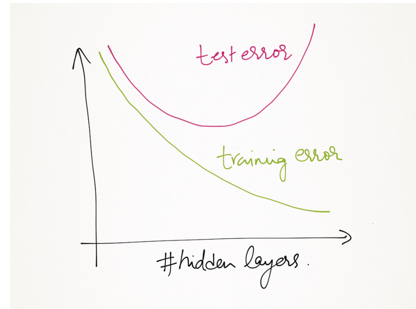
Do we gain anything beyond two layers?

Expressive power of a two-layer network

- **Theorem [Kurt Hornik et al., 1989]:** Let F be a continuous function on a bounded subset of D-dimensional space. Then there exists a two-layer network F with finite number of hidden units that approximates $\hat{F}$ arbitrarily well. Namely, for all x in the domain of F, $|F(x)-\hat{F}(x)| < \epsilon$
- Colloquially "a two-layer network can approximate any function"
  - This is true for arbitrary link function
- Going from one to two layers dramatically improves the representation power of the network

How many hidden units?

- D dimensional data with K hidden units has(D+2)K+1 parameters
  - (D+1)K in the first layer (1 for the bias) and K+1 in the second layer
- With N training examples, set the number of hidden units K ~ N/D to keep the number of parameters comparable to size of training data
- K is both a form of regularization and inductive bias
- Training and test error vs. K



Training a two-layer network

- Optimization framework:

$$\min_{W,v} \sum_n \frac{1}{2}\left(y_n - \sum_i \mathbf{v}_i f(\mathbf{w}_i^T \mathbf{x}_n)\right)^2$$

- Loss minimization: replace squared-loss with any other
- Regularization:
  - Add a regularization (e.g. l_2-norm of the weights)
  - Other ideas: dropout, batch normalization, etc
- Optimization by gradient descent
  - Highly non-convex problem so no guarantees about optimality

Training a two-layer network

- Optimization framework:

$$\min_{W,v} \sum_n \frac{1}{2}\left(y_n - \sum_i \mathbf{v}_i f(\mathbf{w}_i^T \mathbf{x}_n)\right)^2$$

or equivalently,

$$\min_{W,v} \sum_n \frac{1}{2}\left(y_n - \mathbf{v}^T \mathbf{h}_n\right)^2 \qquad \mathbf{h}_{i,n} = f(\mathbf{w}_i^T \mathbf{x}_n)$$

- Computing gradients: second layer

$$\frac{dL_n}{d\mathbf{v}} = -\left(y_n - \mathbf{v}^T \mathbf{h}_n\right) \mathbf{h}_n$$

least-squares regression

Training a two-layer network

- Optimization framework:

$$\min_{W,v} \sum_n \frac{1}{2}\left(y_n - \sum_i \mathbf{v}_i f(\mathbf{w}_i^T \mathbf{x}_n)\right)^2$$

or equivalently,

$$\min_{W,v} \sum_n \frac{1}{2}\left(y_n - \mathbf{v}^T \mathbf{h}_n\right)^2 \qquad \mathbf{h}_{i,n} = f(\mathbf{w}_i^T \mathbf{x}_n)$$

- Computing gradients: first layer
  - Chain rule of derivatives

$$\frac{dL_n}{d\mathbf{w}_i} = \sum_j \frac{dL_n}{d\mathbf{h}_j}\frac{d\mathbf{h}_j}{d\mathbf{w}_i} \rightarrow \frac{dL_n}{d\mathbf{w}_i} = -\left(y_n - v^T h_n\right) v_i f'(\mathbf{w}_i^T \mathbf{x_n})\mathbf{x}_n$$

O if $i \neq j$

also called as back-propagation

Neural Networks

Subhransu Maji

CMPSCI 670: Computer Vision

November 10, 2016

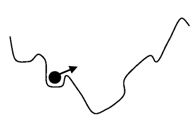
Practical issues: gradient descent

- Easy to get gradients wrong!
  - One strategy is to learn v by fixing W (least-squares) and then learn W by fixing v and iterate between the two steps.
- Use online gradients (or stochastic gradients)

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{dL_n}{d\mathbf{w}} \qquad \frac{dL}{d\mathbf{w}} = \sum_n \frac{dL_n}{d\mathbf{w}}$$

batch online

- Learning rate: start with a high value and reduce it when the validation error stops decreasing
- Momentum: move out small local minima
  - Usually set to a high value: $\beta = 0.9$

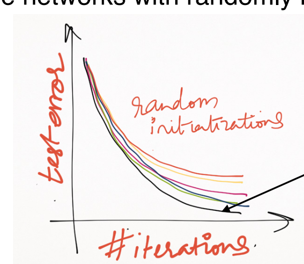


$$\Delta\mathbf{w}^{(t)} = \beta\Delta\mathbf{w}^{(t-1)} + (1-\beta)\left(-\eta\frac{dL_n}{d\mathbf{w}^{(t)}}\right)$$

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^t + \Delta\mathbf{w}^{(t)}$$

Practical issues: initialization

- Initialization didn't matter for linear models
  - Guaranteed convergence to global minima as long as step size is suitably chosen since the objective is convex
- Neural networks are sensitive to initialization
  - Many local minima
  - Symmetries: reorder the hidden units and change the weights accordingly to get another network that produces identical outputs
- Train multiple networks with randomly initialized weights

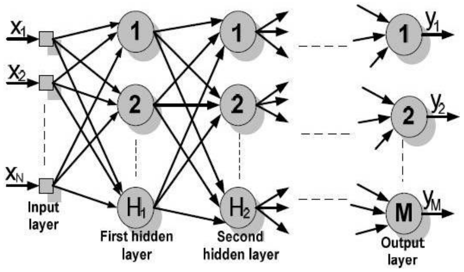


pick the best

Beyond two layers

- The architecture generalizes to any directed acyclic graph (DAG)
  - For example a multi-layer network
  - One can order the vertices in a DAG such that all edges go from left to right (topological sorting)

prediction: forward propagation



gradients: backward propagation

Breadth vs. depth

- Why train deeper networks?
- We will borrow ideas from theoretical computer science
  - A boolean circuit is a DAG where each node is either an input, an AND gate, an OR gate, or a NOT gate. One of these is designated as an output gate.
  - Circuit complexity of a boolean function f is the size of the smallest circuit (i.e., with the fewest nodes) that can compute f.

- **The parity function:** the number of 1s is even or odd

$$\text{parity}(\mathbf{x}) = \left(\sum_d x_d\right) \mod 2$$

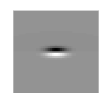
- **[Håstad, 1987]** A depth-*k* circuit requires $\exp\left(n^{\frac{1}{k-1}}\right)$ to compute the parity function of *n* inputs

Breadth vs. depth

- Why <u>**not**</u> train deeper networks?
- Selecting the architecture is daunting
  - How many hidden layers
  - How many units per hidden layer
- Vanishing gradients
  - Gradients shrink as one moves away from the output layer
  - Convergence is slow
- Training deep networks is an active area of research
  - Layer-wise initialization (perhaps using unsupervised data)
  - Engineering: GPUs to train on massive labelled datasets

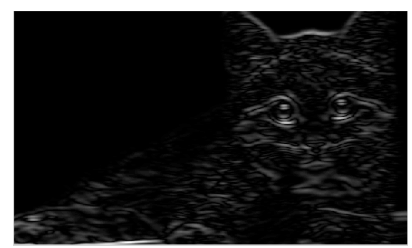
Convolutional neural networks

- Images are not just a collection of pixels
  - Lots of local structure: edges, corners, etc
  - These statistics are translation invariant
- The convolution operation:



filter: horizontal edge

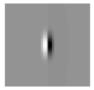
image



absolute value of the output of convolution of the image and filter

Convolutional neural networks

- Images are not just a collection of pixels
  - Lots of local structure: edges, corners, etc
  - These statistics are translation invariant
- The convolution operation:



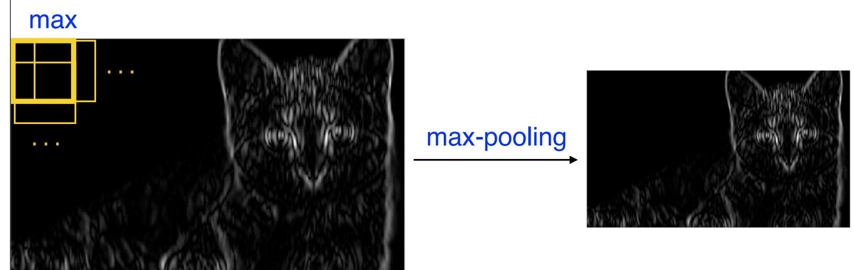
filter: vertical edge

image

absolute value of the output of convolution of the image and filter

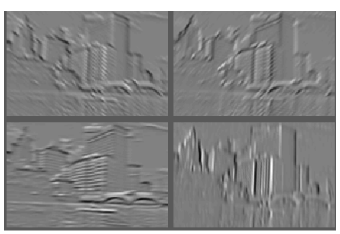
Convolutional neural networks

- Images are not just a collection of pixels
  - Lots of local structure: edges, corners, etc
  - These statistics are translation invariant
- The pooling operation: subsample the output
  - Invariance to small shifts
  - Options: max, sum   Parameters: window size, stride

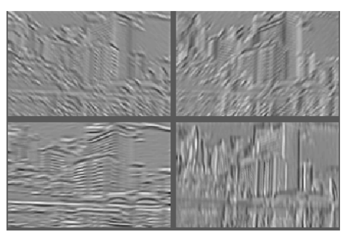


Normalization

- Within or across feature maps
- Before or after spatial pooling



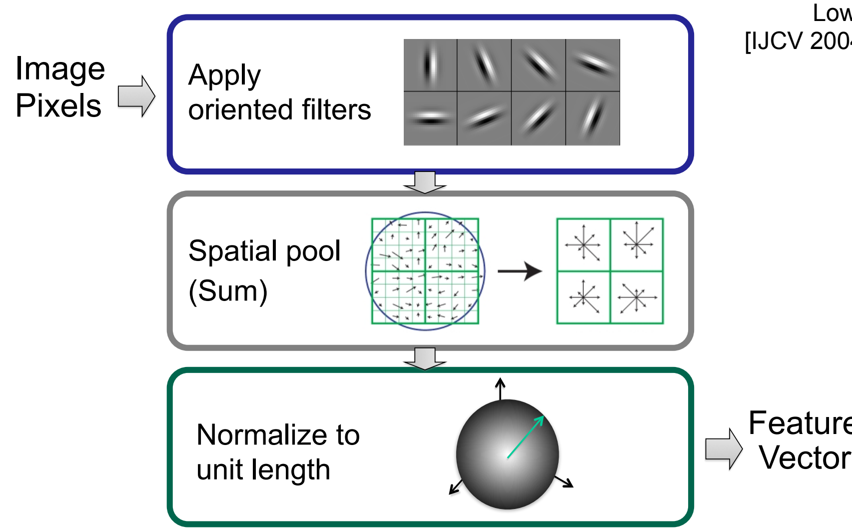
Feature Maps



Feature Maps After Contrast Normalization

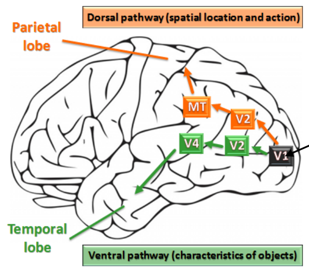
Compare: SIFT Descriptor

Lowe [IJCV 2004]



Convolutional neural networks

- A CNN unit contains the following layers:
  1. Convolutional layer containing a set of filters
  2. Pooling layer
  3. Non-linearity
- Deep CNN: a stack of multiple CNN units
  - Inspired by the human visual system (V1, V2, V3 ….)



Hubel and Weisel, 1968
Structure of V1
“simple” cells: edge detectors

Example: LeNet5

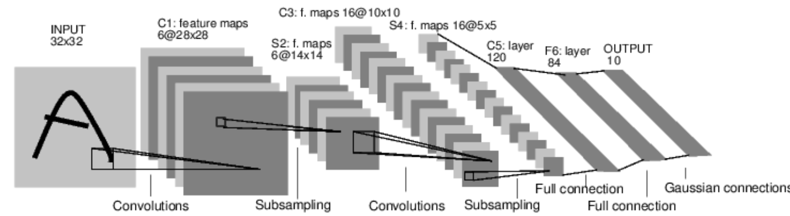


image ⟶ 6 5x5 ⟶ 2x2 ⟶ 16 6x6 ⟶ 2x2 ⟶ 120 5x5 ⟶ full ⟶ full

- C1: Convolutional layer with 6 filters of size 5x5
- Output: 6x28x28
- Number of parameters: (5x5+1)*6 = 156
- Connections: (5x5+1)x(6x28x28) = 122304
- Connections in a fully connected network: (32x32+1)x(6X28x28)

LeCun 98

Example: LeNet5

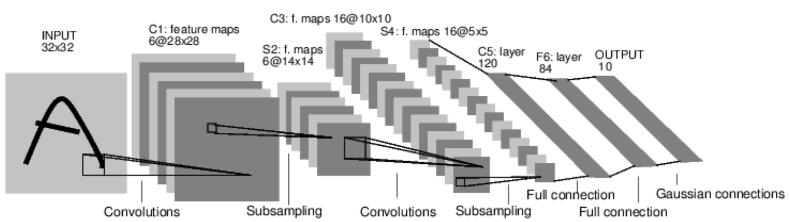


image ⟶ 6 5x5 ⟶ 2x2 ⟶ 16 6x6 ⟶ 2x2 ⟶ 120 5x5 ⟶ full ⟶ full

- S2: Subsampling layer
- Subsample by taking the sum of non-overlapping 2x2 windows
  - Multiply the sum by a constant and add bias
- Number of parameters: 2x6=12
- Pass the output through a sigmoid non-linearity
- Output: 6x14x14

Example: LeNet5

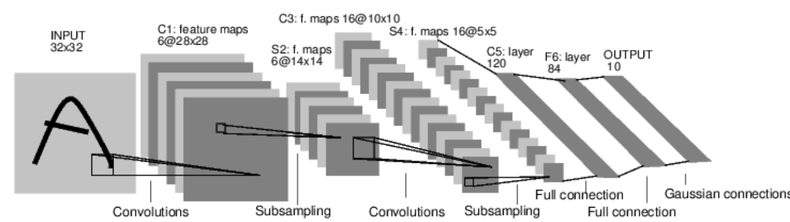


image ⟶ 6 5x5 ⟶ 2x2 ⟶ 16 6x6 ⟶ 2x2 ⟶ 120 5x5 ⟶ full ⟶ full

- C3: Convolutional layer with 16 filters of size 6x6
- Each is connected to a subset:
- Number of parameters: 1,516
- Number of connections: 151,600
- Output: 16x10x10

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

Example: LeNet5

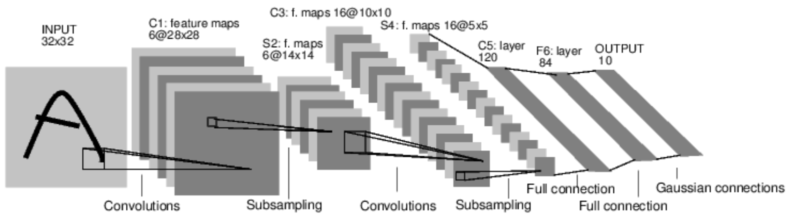


image → 6 5x5 → 2x2 → 16 6x6 → 2x2 → 120 5x5 → full → full

- S4: Subsampling layer
- Subsample by taking the sum of non-overlapping 2x2 windows
  - Multiply by a constant and add bias
- Number of parameters: 2x16 = 32
- Pass the output through a sigmoid non-linearity
- Output: 16x5x5

Example: LeNet5

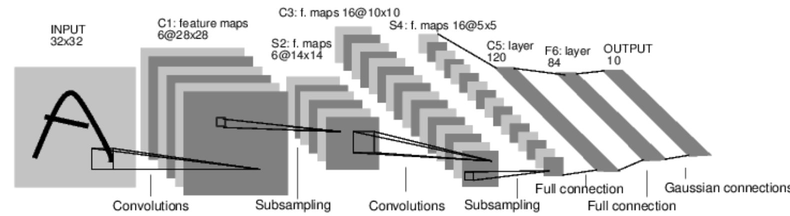


image → 6 5x5 → 2x2 → 16 6x6 → 2x2 → 120 5x5 → full → full

- C5: Convolutional layer with 120 outputs of size 1x1
- Each unit in C5 is connected to all inputs in S4
- Number of parameters: (16x5x5+1)*120 = 48120

Example: LeNet5

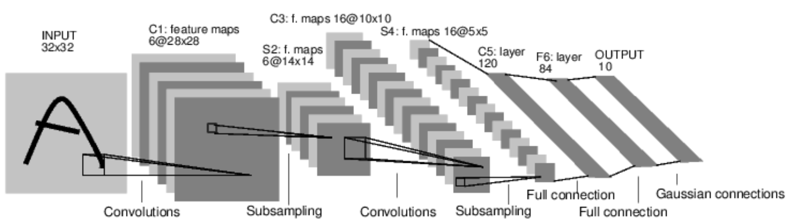


image → 6 5x5 → 2x2 → 16 6x6 → 2x2 → 120 5x5 → full → full

- F6: fully connected layer
- Output: 1x1x84
- Number of parameters: (120+1)*84 = 10164

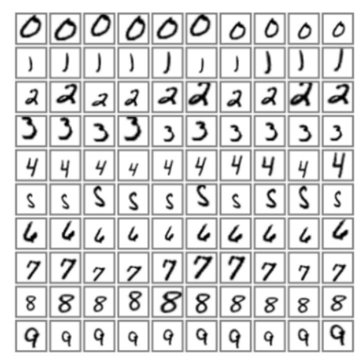
- OUTPUT: 10 Euclidean RBF units (one for each digit class)

$$y_i = \sum_j (x_j - w_{ij})^2.$$

MNIST dataset

60,000 original datasets

Test error: 0.95%



540,000 artificial distortions

+ 60,000 original

Test error: 0.8%

3-layer NN, 300+100 HU [distortions]
Test error: 2.5%

http://yann.lecun.com/exdb/mnist/

MNIST dataset: errors on the test set

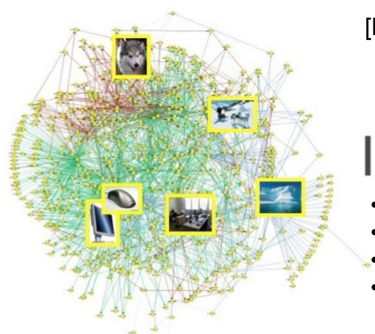
Neural Networks

Subhransu Maji

CMPSCI 670: Computer Vision

November 15, 2016

ImageNet Challenge 2012



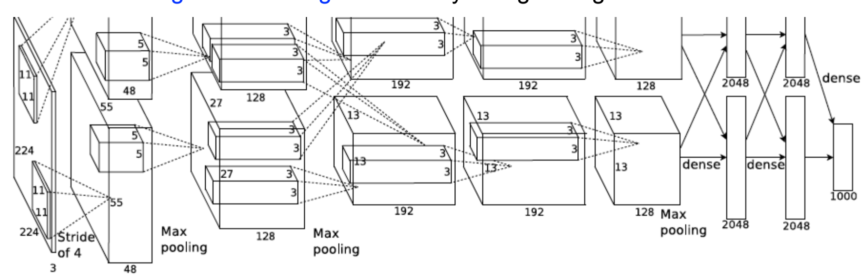
[Deng et al. CVPR 2009]

IMAGENET

- 14+ million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- The challenge: 1.2 million training images, 1000 classes

ImageNet Challenge 2012

- Similar to LeCun'98 with some differences:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images) — ImageNet dataset [Deng et al.]
  - GPU implementation (50x speedup over CPU) ~ 2 weeks to train
  - Some twists: Dropout regularization, ReLU max(0,x)
- Won the ImageNet challenge in 2012 by a large margin!

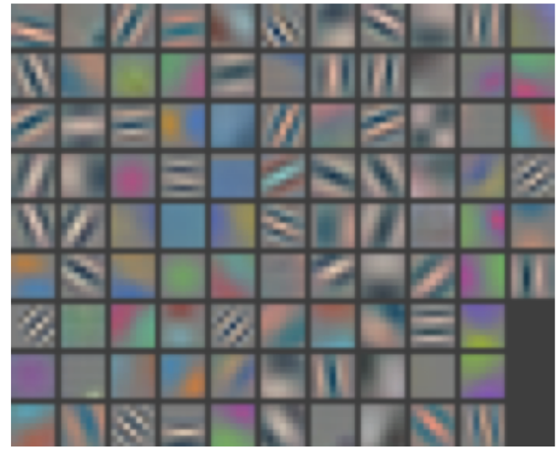


Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

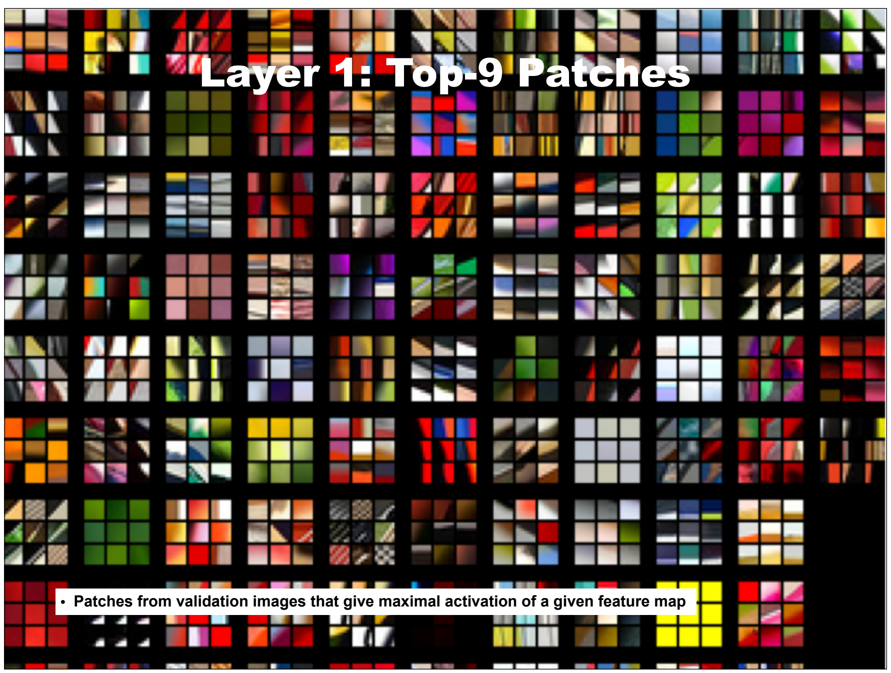
What do these networks learn?

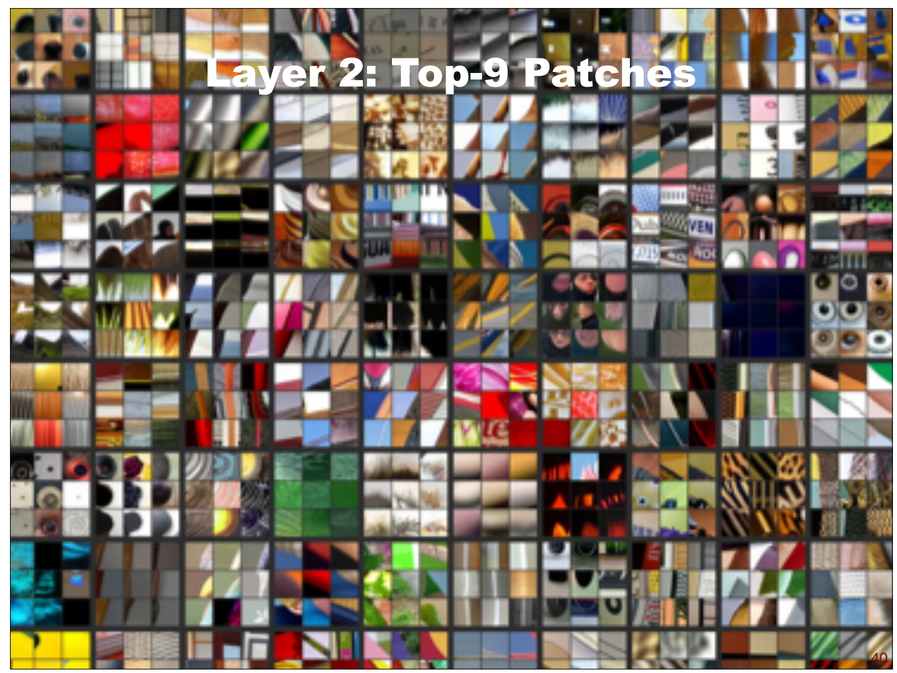
- How do we visualize a complicated, non-linear function?
- Good paper: Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler, Rob Fergus, ECCV 2014
- Good toolbox: Understanding Neural Networks Through Deep Visualization, Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, ICML Deep Learning Workshop, 2015
  - http://yosinski.com/deepvis
- Many other resources online (search for visualizing deep networks)

Layer 1: Learned filters



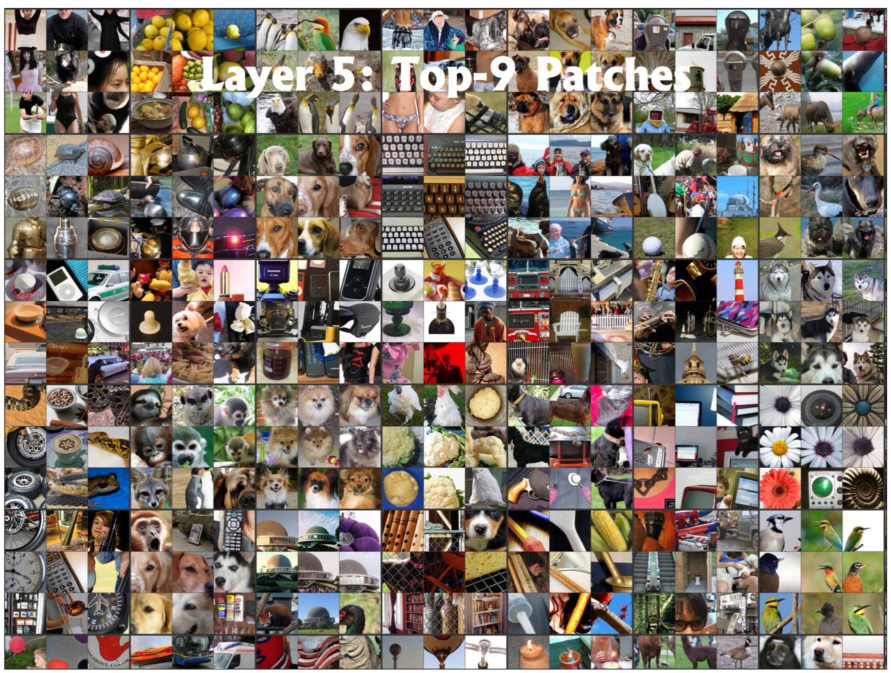
similar to “edge” and “blob” detectors

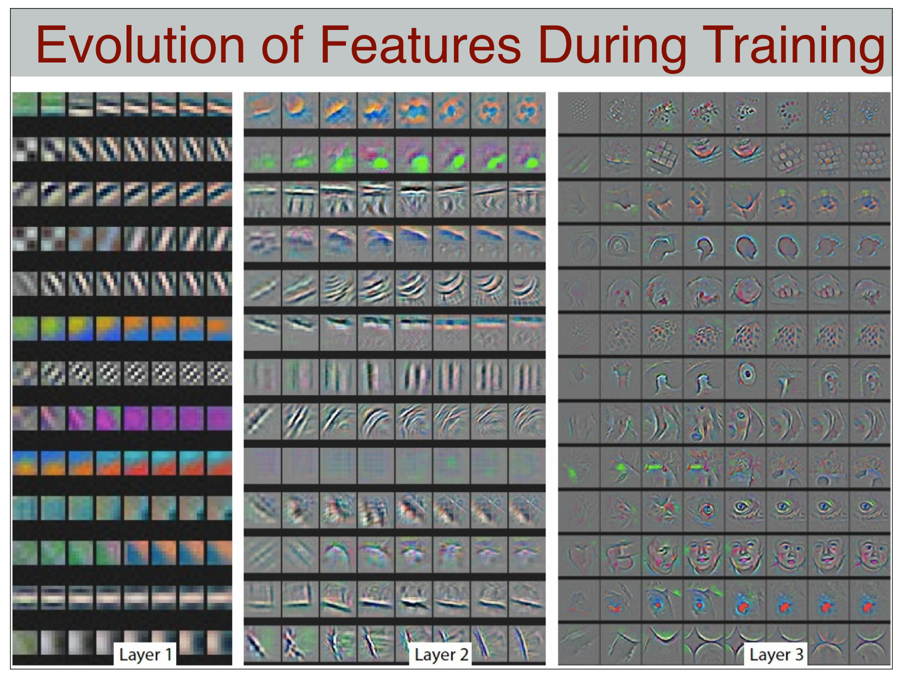




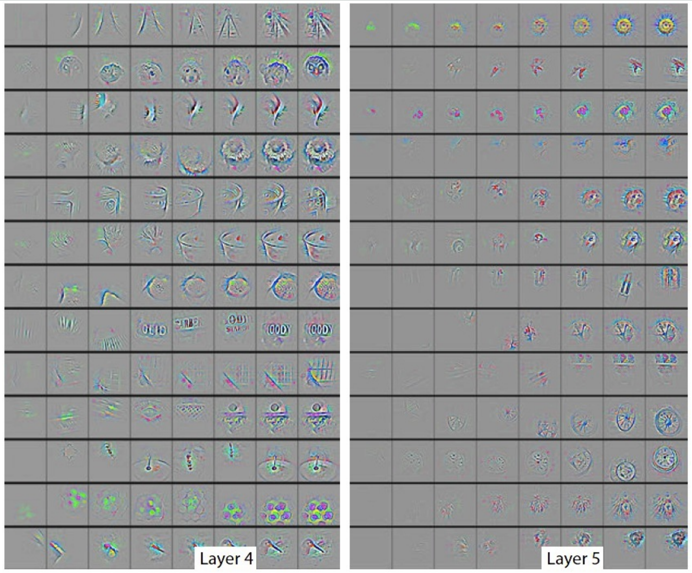

Layer 3: Top-9 Patches


Layer 4: Top-9 Patches


Layer 5: Top-9 Patches


Evolution of Features During Training
Layer 1
Layer 2
Layer 3

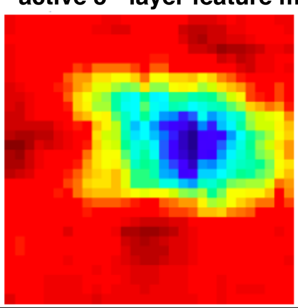
Evolution of Features During Training



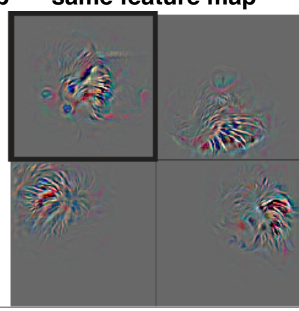
Occlusion Experiment

- Mask parts of input with occluding square
- Monitor output (class probability)

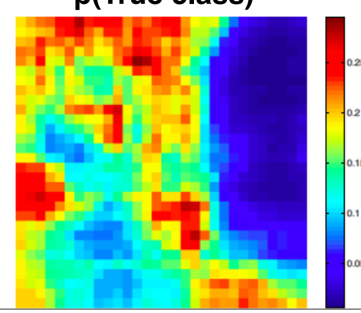
Total activation in most active 5th layer feature map



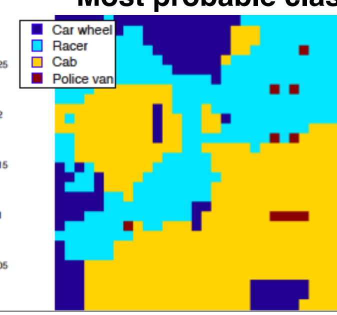
Other activations from same feature map



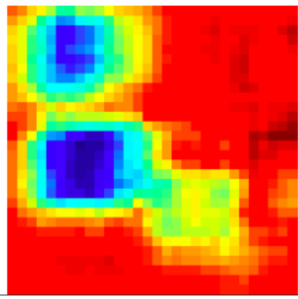
p(True class)

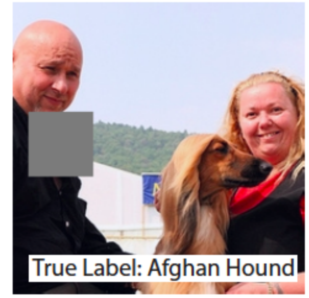


Most probable class

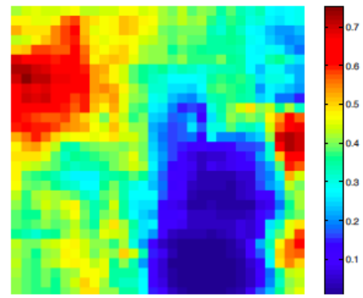


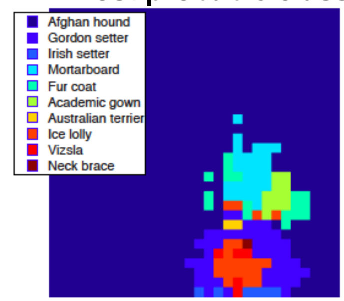
Total activation in most active 5th layer feature map | **Other activations from same feature map**

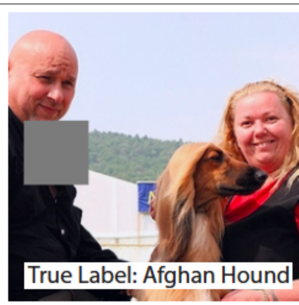




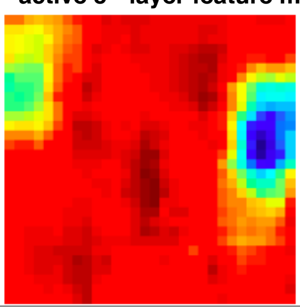
p(True class) | **Most probable class**





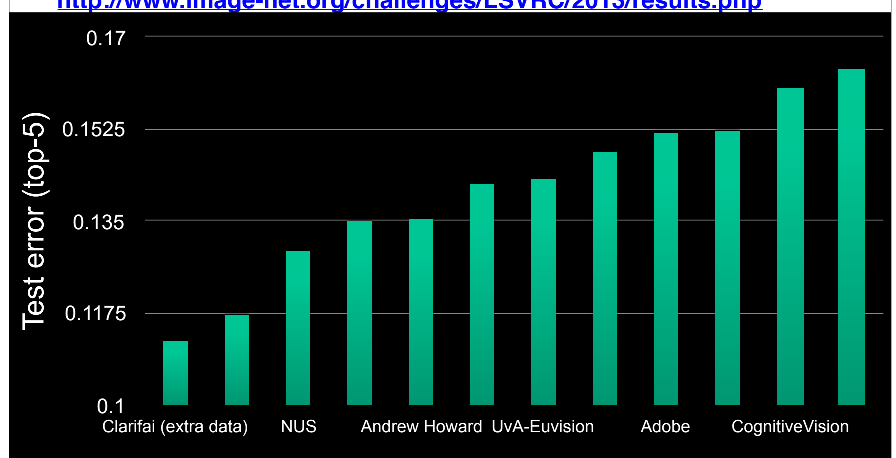


Total activation in most active 5th layer feature map | **Other activations from same feature map**



ImageNet Classification 2013 Results

http://www.image-net.org/challenges/LSVRC/2013/results.php



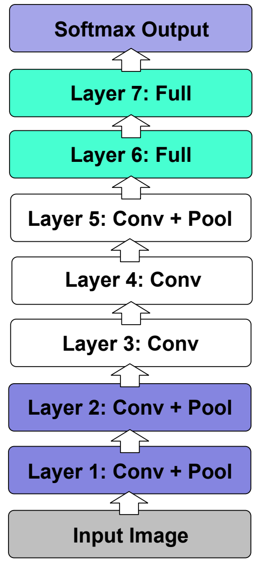
ImageNet 2014 - Test error at 0.07 (Google & Oxford groups)

http://image-net.org/challenges/LSVRC/2014/results

How important is depth?

Architecture of Krizhevsky et al.

- 8 layers total
- Trained on ImageNet
- 18.1% top-5 error



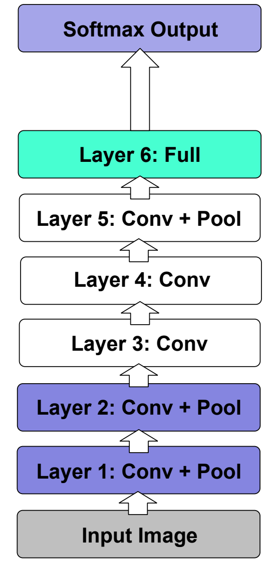
How important is depth?

Remove top fully connected layer

- Layer 7

Drop 16 million parameters

Only 1.1% drop in performance!



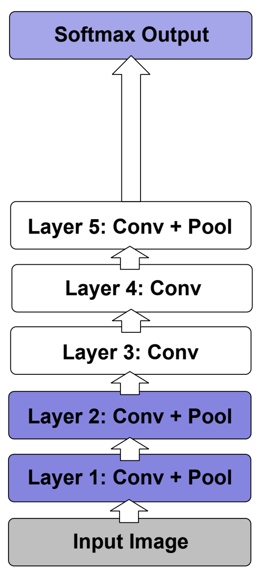
How important is depth?

Remove both fully connected layers

- Layer 6 & 7

Drop ~50 million parameters

5.7% drop in performance



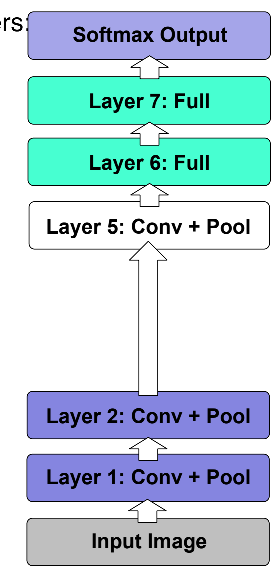
How important is depth?

Now try removing upper feature extractor layers:

- Layers 3 & 4

Drop ~1 million parameters

3.0% drop in performance



How important is depth?

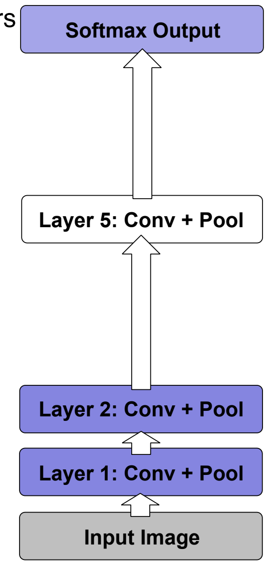
Now try removing upper feature extractor layers & fully connected:

- Layers 3, 4, 6 ,7

Now only 4 layers

33.5% drop in performance

→ Depth of network is key



Can we go deeper?

- Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, ECCV 2016

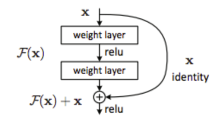
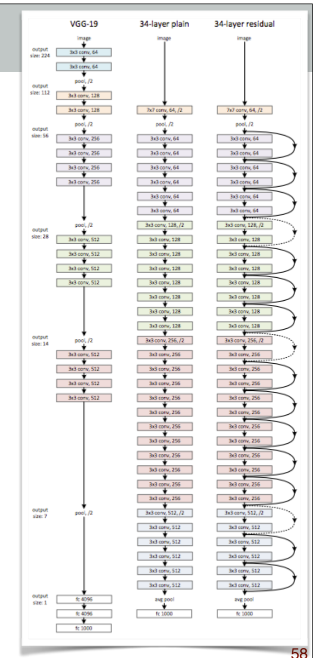


Figure 2. Residual learning: a building block.

- Winner of ImageNet challenge 2015
  - 5.7% top5 error
- VGG-19 (3x more layers than AlexNet)
- **ResNet** (2-20x more layers than VGG-19)



CNNs for small datasets

- Take model trained on ImageNet
- Take outputs of 6th or 7th layer before or after nonlinearity as features
- Train linear classifiers on these features (like retraining the last layer of the network)
- Optionally back-propagate: fine-tune features and/or classifier on new dataset

- Transfer learning
  - Techniques to generalize from one task to another
  - Training and testing distributions may be different
    - Will driving in Amherst help driving in Boston?

Tapping off features at each Layer

Plug features from each layer into linear classifier

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$

Higher layers are better

Results on benchmarks

[1] Caltech-101 (30 samples per class)

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

[1] SUN 397 dataset (DeCAF)

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

[1] Caltech-UCSD Birds (DeCAF)

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

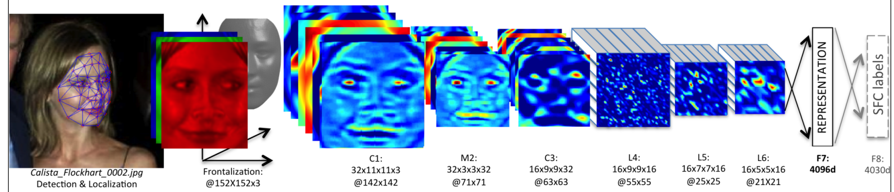
[2] MIT-67 Indoor Scenes dataset (OverFeat)

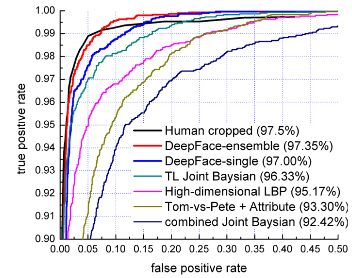
Method	mean Accuracy
ROI + Gist[36]	26.05
DPM[30]	30.40
Object Bank[25]	37.60
RBow[31]	37.93
BoP[22]	46.10
miSVM[26]	46.40
D-Parts[40]	51.40
IFV[22]	60.77
MLrep[11]	**64.03**
CNN-SVM	58.44

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, **DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition**, arXiv preprint, 2014

[2] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, **CNN Features off-the-shelf: an Astounding Baseline for Recognition**, arXiv preprint, 2014

CNN features for face verification





Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR 2014

Open-source CNN software

- **Cuda-convnet** (Alex Krizhevsky, Google)
  - High speed convolutions on the GPU
- **Caffe** (Y. Jia and others, Berkeley)
  - High performance CNNs
  - Flexible CPU/GPU computations
- **Overfeat** (NYU)
- **MatConvNet** (Andrea Vedaldi, Oxford)
  - An easy to use toolbox for CNNs from MATLAB
  - Comparable performance/features with Caffe
- **TensorFlow** (Google)
- **Torch** (Facebook, Google, academia, etc.)
- Many others ….

Summary

- Motivation: non-linearity
- Ingredients of a neural network
  - hidden units, link functions
- Training by back-propagation
  - random initialization, chain rule, stochastic gradients, momentum
  - Practical issues: learning, network architecture
- Theoretical properties:
  - A two-layer network is a universal function approximator
  - However, deeper networks can be more efficient at approximating certain functions
- Convolutional neural networks:
  - Good for vision problems where inputs have local structure
  - Shared structure of weights leads to significantly fewer parameters

Slides credit

- Multilayer neural network figure source:
  - http://www.ibimapublishing.com/journals/CIBIMA/2012/525995/525995.html
- Cat image: http://www.playbuzz.com/abbeymcneill10/which-cat-breed-are-you
- More about the structure of the visual processing system
  - http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/lgn-V1.html
- ImageNet visualization slides are by Rob Fergus @ NYU/Facebook http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf
- LeNet5 figure from: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf
- Chain rule of derivatives: http://en.wikipedia.org/wiki/Chain_rule