Deep Learning & Neural Networks

Machine Learning – CSE4546
Sham Kakade
University of Washington
November 29, 2016

Announcements:

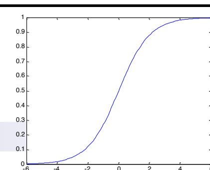
- HW4 posted
- Poster Session Thurs, Dec 8

- Today:
  - Review: EM
  - Neural nets and deep learning

Poster Session

- Thursday Dec 8, 9-11:30am
  - Please arrive 20 mins early to set up
- Everyone is expected to attend
- Prepare a poster
  - We provide poster board and pins
  - Both one large poster (recommended) and several pinned pages are OK.
- Capture
  - Problem you are solving
  - Data you used
  - ML methodology
  - Results
- ***Prepare a 1-minute speech about your project***
- Two instructors will visit your poster separately
- Project Grading: scope, depth, data

Logistic regression



- P(Y|X) represented by:

$$P(Y=1 \mid x, W) = \frac{1}{1+e^{-(w_0+\sum_i w_i x_i)}}$$
$$= g(w_0 + \sum_i w_i x_i)$$

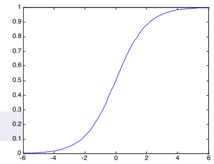
- Learning rule – MLE:

$$\frac{\partial \ell(W)}{\partial w_i} = \sum_j x_i^j [y^j - P(Y^j = 1 \mid x^j, W)]$$
$$= \sum_j x_i^j [y^j - g(w_0 + \sum_i w_i x_i^j)]$$

$$w_i \leftarrow w_i + \eta \sum_j x_i^j \delta^j$$

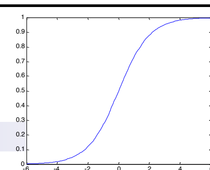
$$\delta^j = y^j - g(w_0 + \sum_i w_i x_i^j)$$

Perceptron as a graph



$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}$$

Linear perceptron classification region



$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}$$

Percepton, linear classification, Boolean functions

- Can learn x_1 AND x_2
- Can learn x_1 OR x_2
- Can learn any conjunction or disjunction

Percepton, linear classification, Boolean functions

- Can learn majority
- Can perceptrons do everything?

Going beyond linear classification

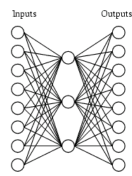
- Solving the XOR problem

Hidden layer

- Perceptron: $$out(\mathbf{x}) = g(w_0 + \sum_i w_i x_i)$$
- 1-hidden layer:

$$out(\mathbf{x}) = g\left(w_0 + \sum_k w_k g(w_0^k + \sum_i w_i^k x_i)\right)$$

Example data for NN with hidden layer



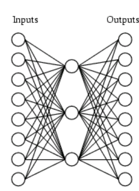
A target function:

Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Can this be learned??

Learned weights for hidden layer

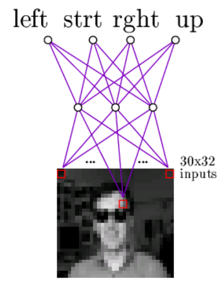
A network:



Learned hidden layer representation:

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

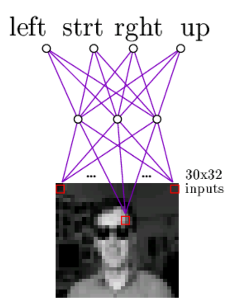
NN for images

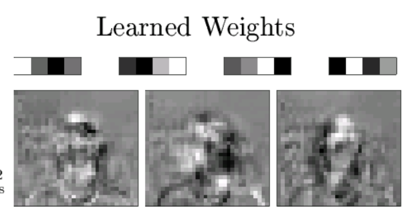


Typical input images

90% accurate learning head pose, and recognizing 1-of-20 faces

Weights in NN for images





Typical input images

Forward propagation for 1-hidden layer - Prediction

- 1-hidden layer:

$$out(\mathbf{x}) = g\left(w_0 + \sum_k w_k g(w_0^k + \sum_i w_i^k x_i)\right)$$

©Sham Kakade 15

Gradient descent for 1-hidden layer – Back-propagation: Computing $\frac{\partial \ell(W)}{\partial w_k}$

Dropped w_0 to make derivation simpler

$$\ell(W) = \frac{1}{2}\sum_j [y^j - out(\mathbf{x}^j)]^2$$

$$out(\mathbf{x}) = g\left(\sum_{k'} w_{k'} g(\sum_{i'} w_{i'}^{k'} x_{i'})\right)$$

$$\frac{\partial \ell(W)}{\partial w_k} = \sum_{j=1}^m -[y^j - out(\mathbf{x}^j)]\frac{\partial out(\mathbf{x}^j)}{\partial w_k}$$

©Sham Kakade 16

Gradient descent for 1-hidden layer – Back-propagation: Computing $\frac{\partial \ell(W)}{\partial w_i^k}$

Dropped w_0 to make derivation simpler

$$\ell(W) = \frac{1}{2}\sum_j [y^j - out(\mathbf{x}^j)]^2$$

$$out(\mathbf{x}) = g\left(\sum_{k'} w_{k'} g(\sum_{i'} w_{i'}^{k'} x_{i'})\right)$$

$$\frac{\partial \ell(W)}{\partial w_i^k} = \sum_{j=1}^{m} -[y - out(\mathbf{x}^j)] \frac{\partial out(\mathbf{x}^j)}{\partial w_i^k}$$

Multilayer neural networks

Forward propagation – prediction

- Recursive algorithm
- Start from input layer
- Output of node V_k with parents U_1,U_2,…:

$$V_k = g\left(\sum_i w_i^k U_i\right)$$

Back-propagation – learning

- Just stochastic gradient descent!!!
- Recursive algorithm for computing gradient
- For each example
  - Perform forward propagation
  - Start from output layer
  - Compute gradient of node V_k with parents U_1,U_2,…
  - Update weight w_i^k

Many possible response/link functions

- Sigmoid
- Linear
- Exponential
- Gaussian
- Hinge
- Max
- …

Convolutional Neural Networks & Application to Computer Vision

Machine Learning – CSE4546
Sham Kakade
University of Washington

November 29, 2016

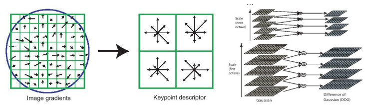
Contains slides from…

- LeCun & Ranzato
- Russ Salakhutdinov
- Honglak Lee

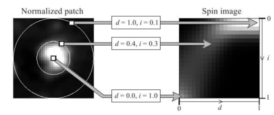
Neural Networks in Computer Vision

- Neural nets have made an amazing come back
  - Used to engineer high-level features of images
- Image features:

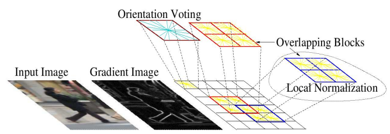
Some hand-created image features



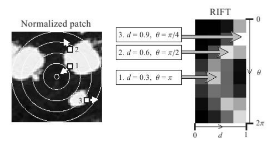
SIFT



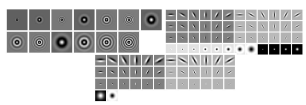
Spin image



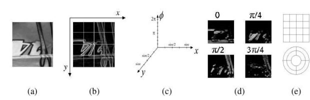
HoG



RIFT



Textons



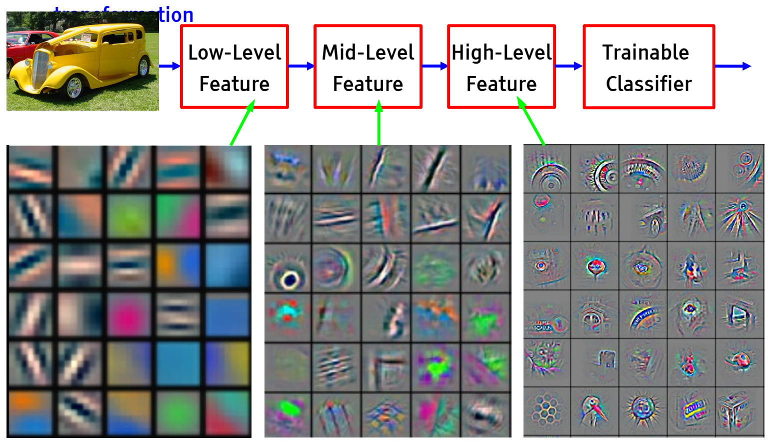
GLOH

Slide Credit: Honglak Lee

Scanning an image with a detector

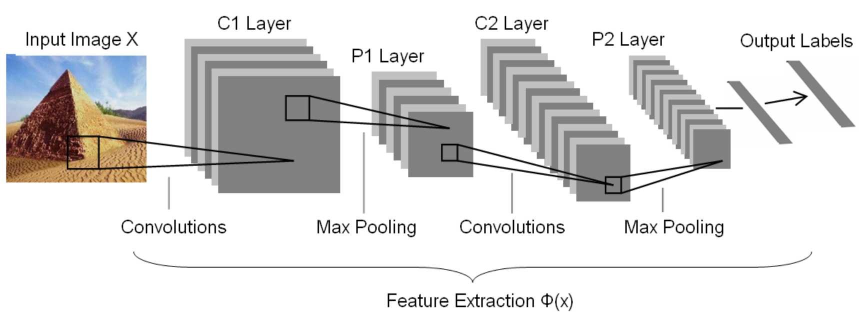
- Detector = Classifier from image patches:
- Typically scan image with detector:

Using neural nets to learn non-linear features



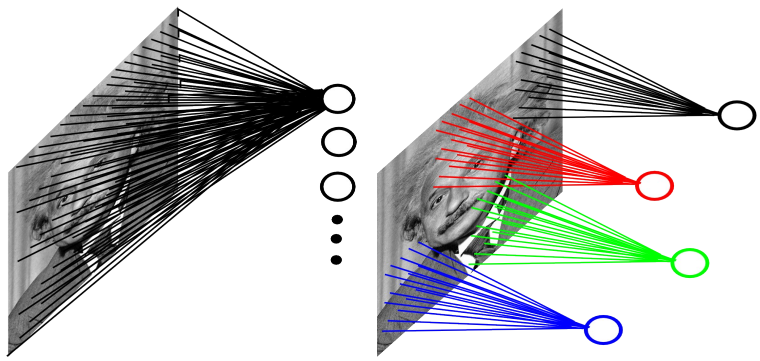
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

But, many tricks needed to work well…



Convolution Layer

- **Example: 200x200 image**
  - Fully-connected, 400,000 hidden units = 16 billion parameters
  - Locally-connected, 400,000 hidden units 10x10 fields = 40 million params
  - Local connections capture local dependencies



Parameter sharing

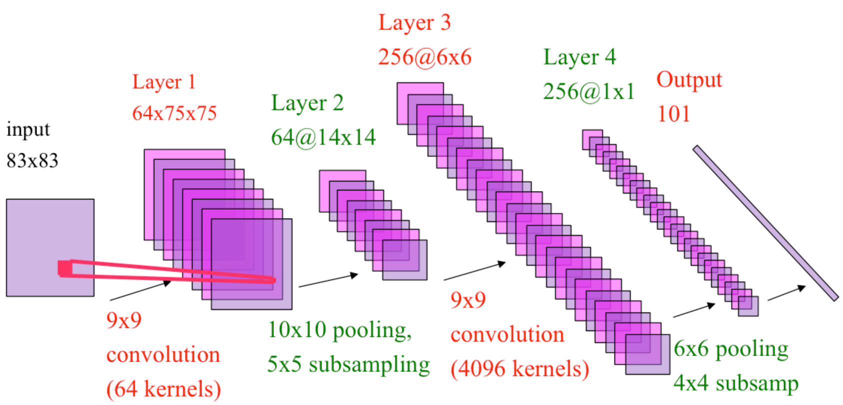
- Fundamental technique used throughout ML
- Neural net without parameter sharing:
- Sharing parameters:

Pooling/Subsampling

- Convolutions act like detectors:

- But we don't expect true detections in every patch
- Pooling/subsampling nodes:

Example neural net architecture



Sample results

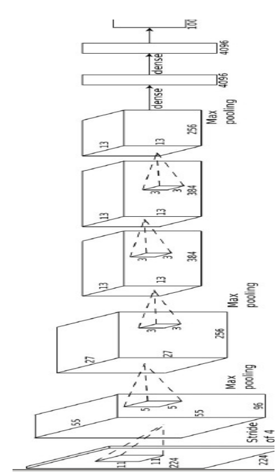
- Traffic Sign Recognition (GTSRB)
  - German Traffic Sign Reco Bench
  - 99.2% accuracy

- House Number Recognition (Google)
  - Street View House Numbers
  - 94.3 % accuracy

Example from Krizhevsky, Sutskever, Hinton 2012

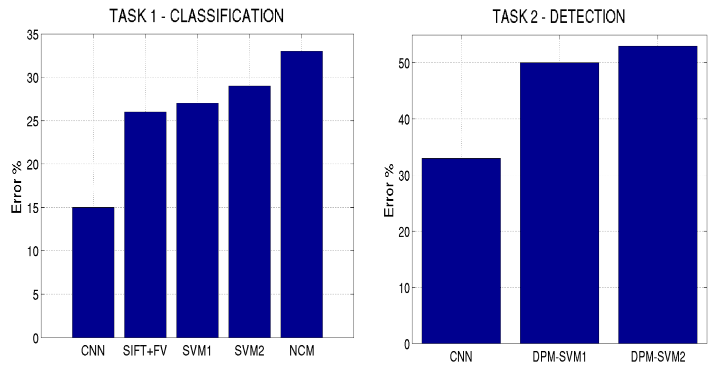
- Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops

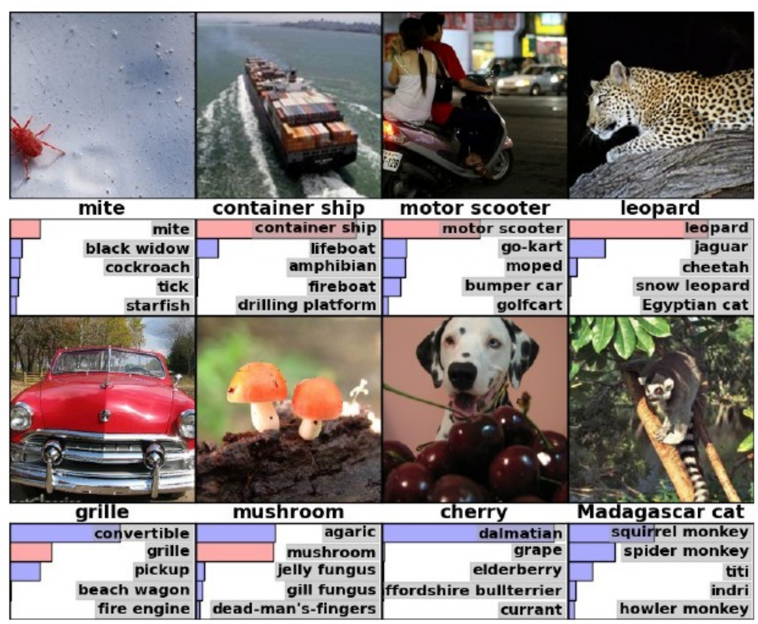
4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3ReLU 384fm	224M
884K	CONV 3x3/ReLU 384fm	149M
	MAX POOLING 2x2sub	
	LOCAL CONTRAST NORM	
307K	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
	LOCAL CONTRAST NORM	
35K	CONV 11x11/ReLU 96fm	105M



Results by Krizhevsky, Sutskever, Hinton 2012

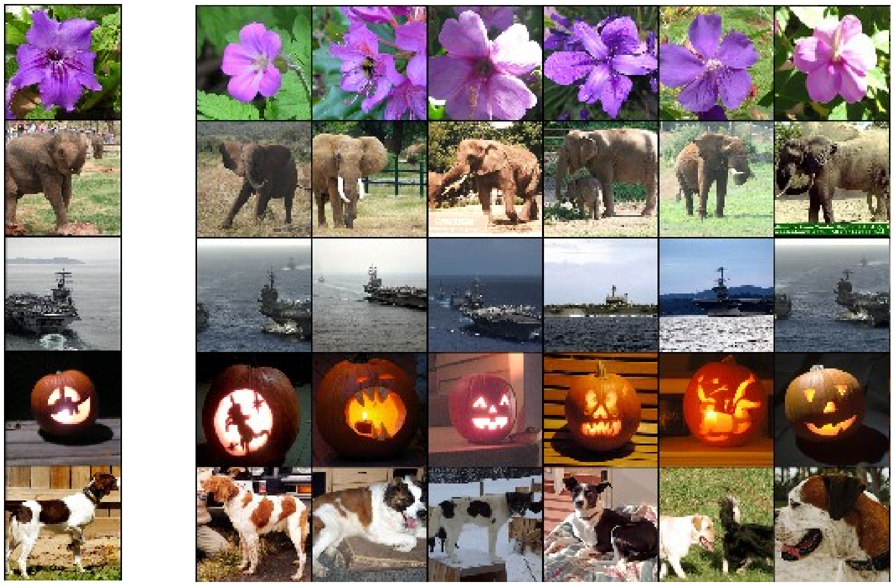
- ImageNet Large Scale Visual Recognition Challenge
- 1000 categories, 1.5 Million labeled training samples



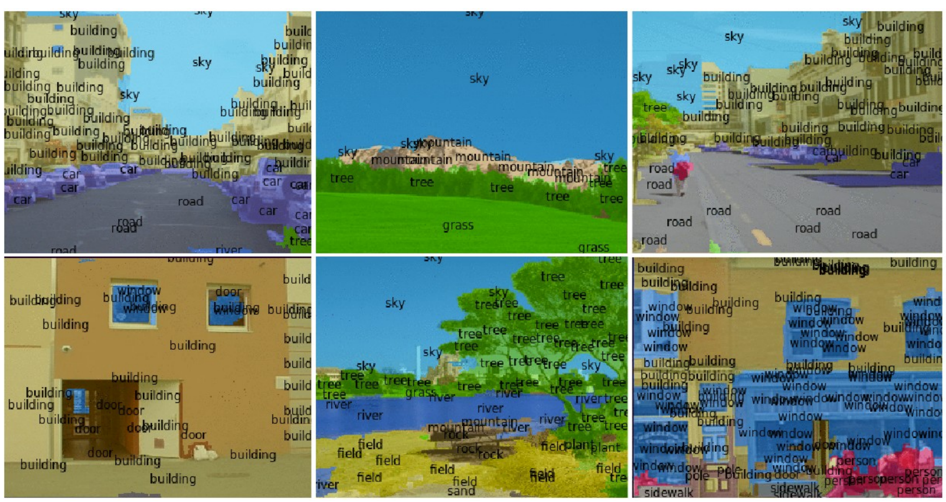


TEST IMAGE

RETRIEVED IMAGES

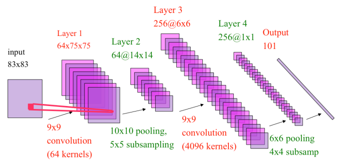


Application to scene parsing



[Farabet et al. ICML 2012, PAMI 2013]

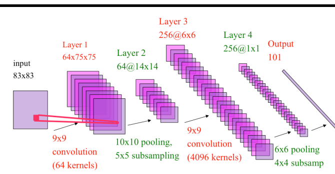
Learning challenges for neural nets



- Choosing architecture
- Slow per iteration and convergence

- Gradient "diffusion" across layers
- Many local optima

Random dropouts



- Standard backprop:

$$w_i \leftarrow w_i + \eta \sum_j x_i^j \delta^j$$

- Random dropouts: randomly choose edges not to update:

- Functions as a type of "regularization"… helps avoid "diffusion" of gradient

Revival of neural networks

- Neural networks fell into disfavor in mid 90s -early 2000s
  - Many methods have now been rediscovered ☺
- Exciting new results using modifications to optimization techniques and GPUs
- Challenges still remain:
  - Architecture selection feels like a black art
  - Optimization can be very sensitive to parameters
  - Requires a significant amount of expertise to get good results