Deep Learning & Neural Networks

Machine Learning – CSE4546
Sham Kakade
University of Washington
December 1, 2016

Announcements:

- HW4 posted
- Poster Session Thurs, Dec 8
  - TAs (or your CSE friends) can help with printing

- Today:
  - Review: Deep Learning
  - Convolutional Neural Nets (+ RNNs?)
  - Start: RL
  - Also: MusicNet is out!

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~~ 2-min summary about your project***
- Two ~~instructors~~ people will visit your poster separately
- Project Grading: scope, depth, data

Review

Machine Learning – CSE4546
Sham Kakade
University of Washington
December 1, 2016

Hidden layer

1-st layer output, k, L

- 1-hidden layer:

2nd output

act

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

input

output

Forward propagation for 1-hidden layer - Prediction

- 1-hidden layer:

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

compute / cache

output / activation at all

layers.

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

Dropped w_0 to make derivation simpler

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

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

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

1st layer

g'(2nd layer activation) $\times$ output

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

Dropped w_0 to make derivation simpler

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

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

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

Theory, Optimization, and Debugging

Machine Learning – CSE4546
Sham Kakade
University of Washington

December 1, 2016

Architecture Selection

- Feed-forward nets
  - These are fully interconnected nets
  - Try wider nets
  - (empirical question) When do deeper nets help?
  - (empirical question) Do feed-forward nets perform better than random features?
- Structured Nets
  - ConvNets are a great idea
    - Some idea of how to choose architecture
  - Recurrent nets
    - Architecture chosen optimization

Optimization Issues

- Initialization
  - Want non-zero gradients
  - Init with a 'sensitivity analysis'
  - Want to start with a point not to far from to some local opt
- Needs lots of Training data?
- Learning rates
  - Set by hand
  - Turn down when learning slows down
- Tensor Flow Defaults?

Regularization

- Needs lots of Training data?
  - Sometimes
  - (briefly) Share MusicNet case study
- Regularization (sometimes important?)
  - L2?
  - Dropout?

very similar to weight decay / L2

"Theory" $w \leftarrow w - \eta \nabla L(w)$

- L(w) is out total loss on N data points
- Suppose L(w) is R-smooth (and L is bounded)
- Let's do batch gradient descent.
- What can we say?

find w s.t. $\|\nabla L(w)\|^2 \le \varepsilon$

in $O(\frac{1}{\varepsilon})$ steps.

Convolutional Neural Networks & Application to Computer Vision

Machine Learning – CSE4546
Sham Kakade
University of Washington

December 1, 2016

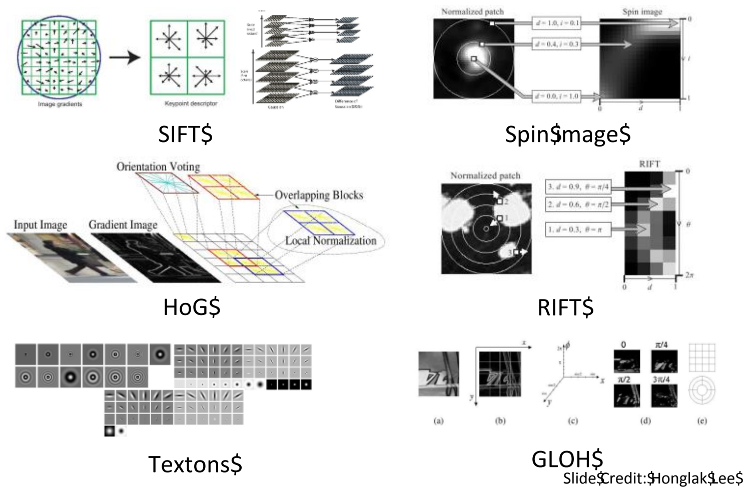
Contains slides from…

- 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



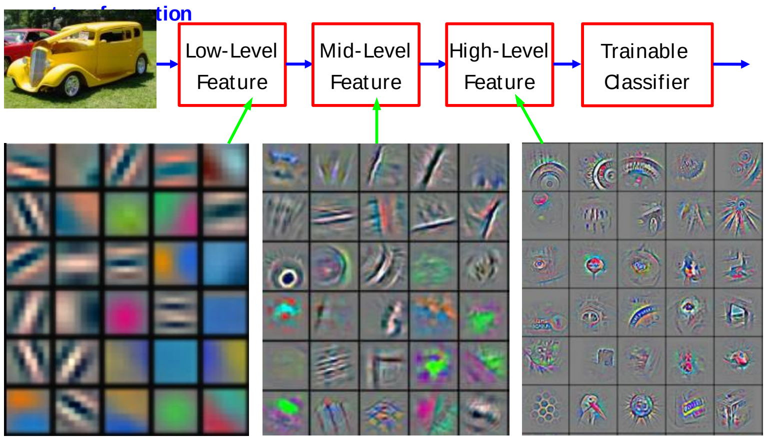
Slide$Credit:$HonglakLee

Scanning an image with a detector

- Detector = Classifier from image patches:

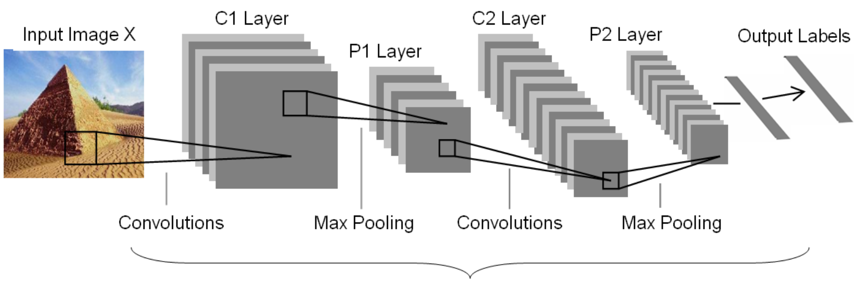
- Typically scan image with detector:

Using neural nets to learn non-linear features



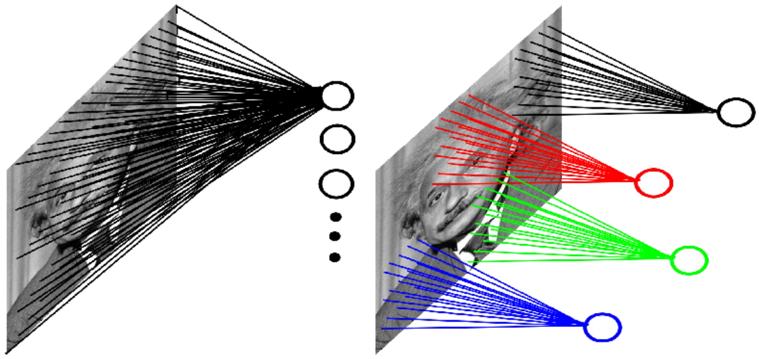
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

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:

(implementing convolutions).

- Sharing parameters:

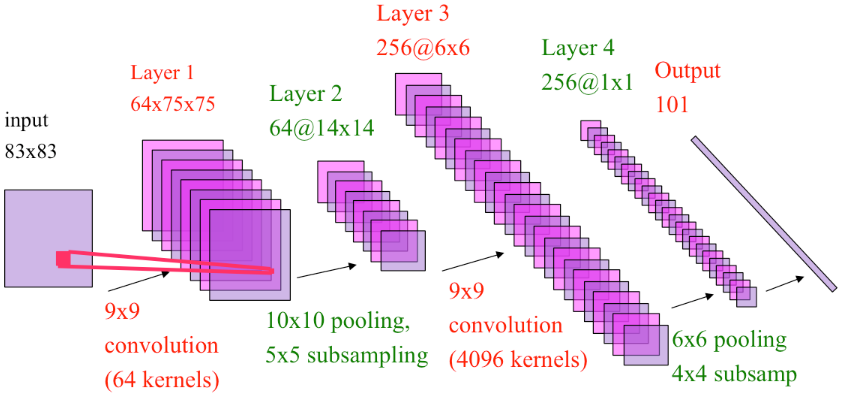
nodes have the same weights.

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

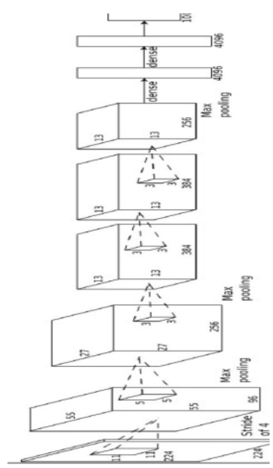
- Traffic Sign Recognition (GTSRB)
  - German Traffic Sign Reco Bench
  - 99.2% accuracy

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

Example from Krizhevsky, Sutskever, Hinton 2012

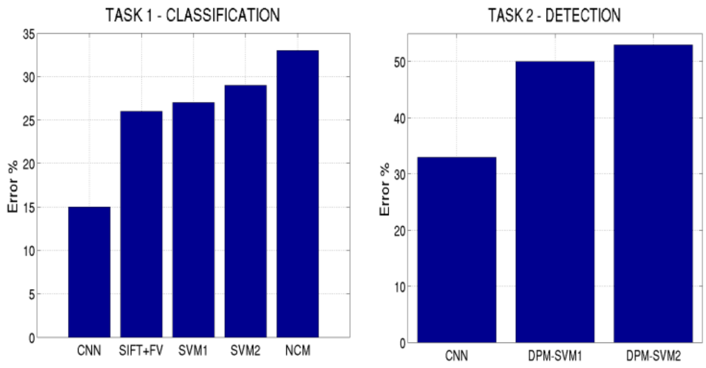
- Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops

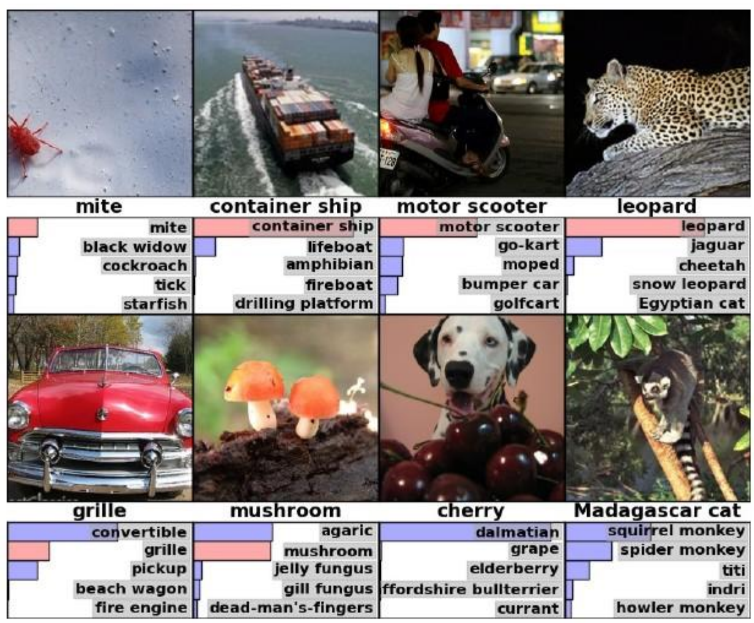
4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3ReLU 384fm	224M
884K	CONV 3x3/ReLU 384fm	149M
	MAX POOLING 2x2sub	
	LOCAL CONTRAST NORM	
307K	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
	LOCAL CONTRAST NORM	
35K	CONV 11x11/ReLU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



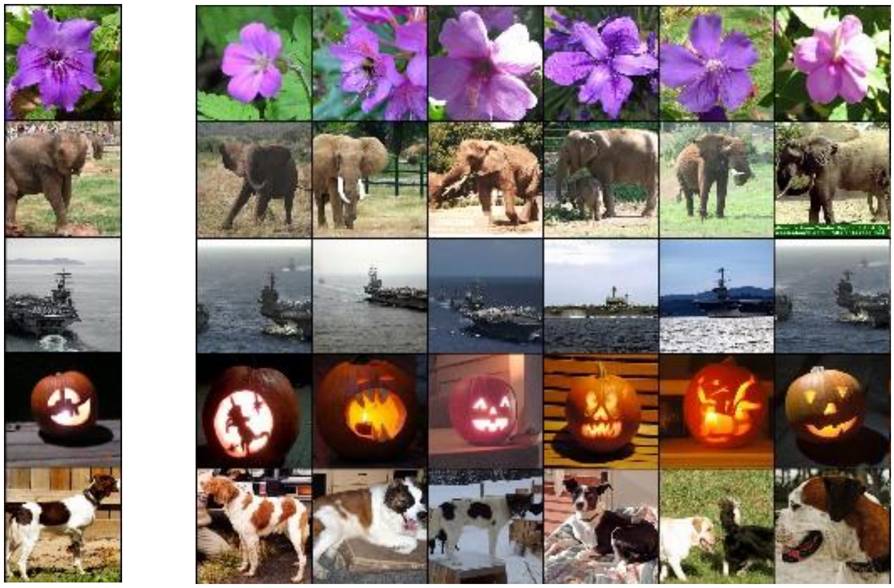
Results by Krizhevsky, Sutskever, Hinton 2012

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





TEST IMAGE **RETRIEVED IMAGES**



RNNs and LSTMs