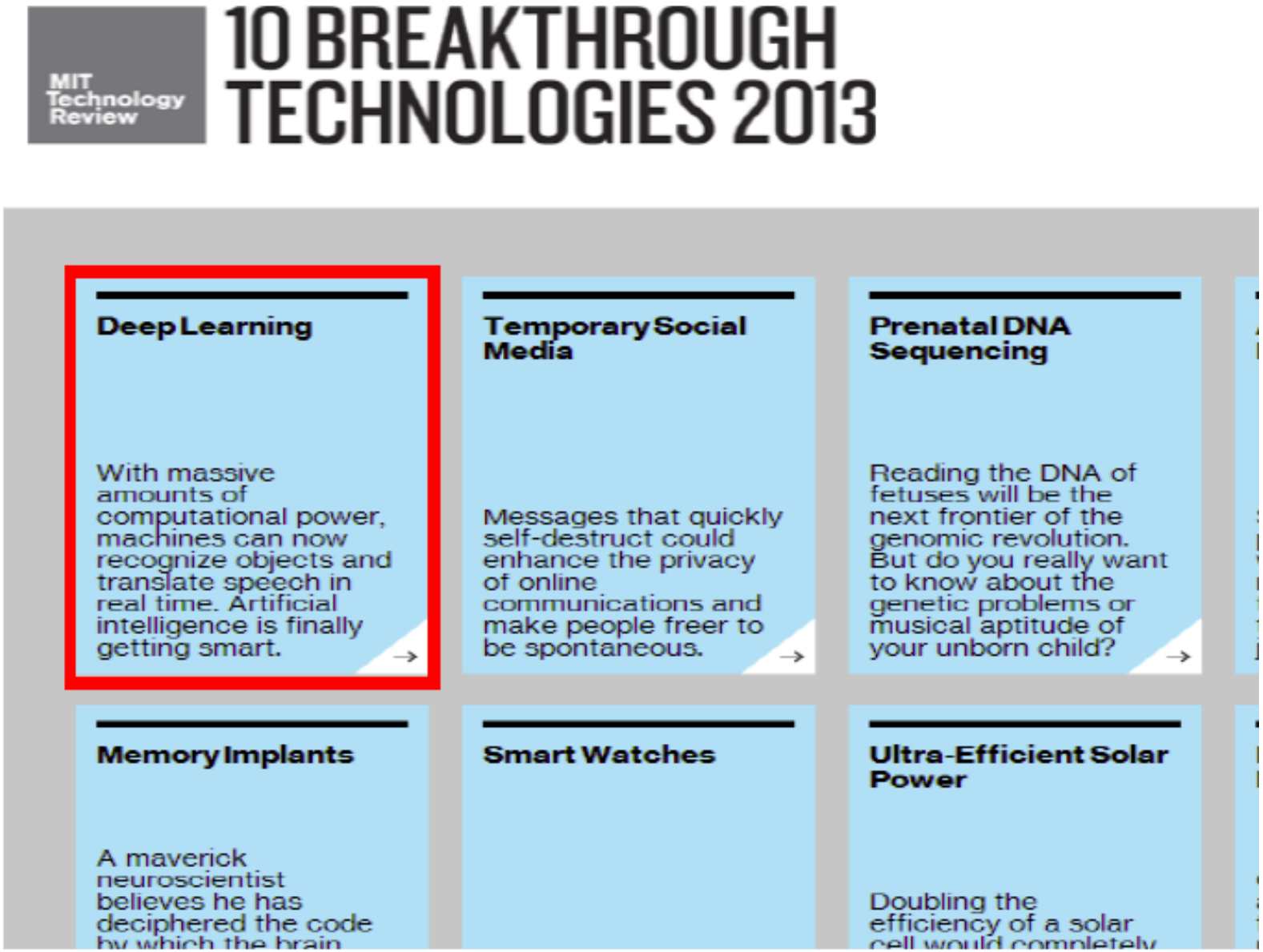
Introduction to Deep Learning

Yu Cheng

IBM T.J. Watson Research Center
chengyu@us.ibm.com

March 31th, 2016

Success of Deep Learning



Success of Deep Learning

MIT Technology Review

Facebook Launches Advanced AI Effort to Find Meaning in Your Posts

A technique called deep learning could help Facebook understand its users and their data better.

By Tom Simonite on September 20, 2013

Facebook is set to get an even better understanding of the 700 million people who use the social network to share details of their personal lives each day.

A new research group within the company is working on an emerging and powerful approach to artificial intelligence known as deep learning, which uses simulated networks of brain cells to process data. Applying this method to data shared on Facebook could allow for novel features and perhaps boost the company's ad targeting.

Deep learning has shown potential as the basis for software that could work out the emotions or events described in text even if they aren't explicitly referenced, recognize objects in photos, and make sophisticated predictions about people's likely future behavior.

WHY IT MATTERS

Facebook's piles of data on people's lives could allow it to push the boundaries of what can be done with the emerging AI technique

Success of Deep Learning

The New York Times

Scientists See Promise in Deep-Learning Programs

Hao Zhang/The New York Times

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese.

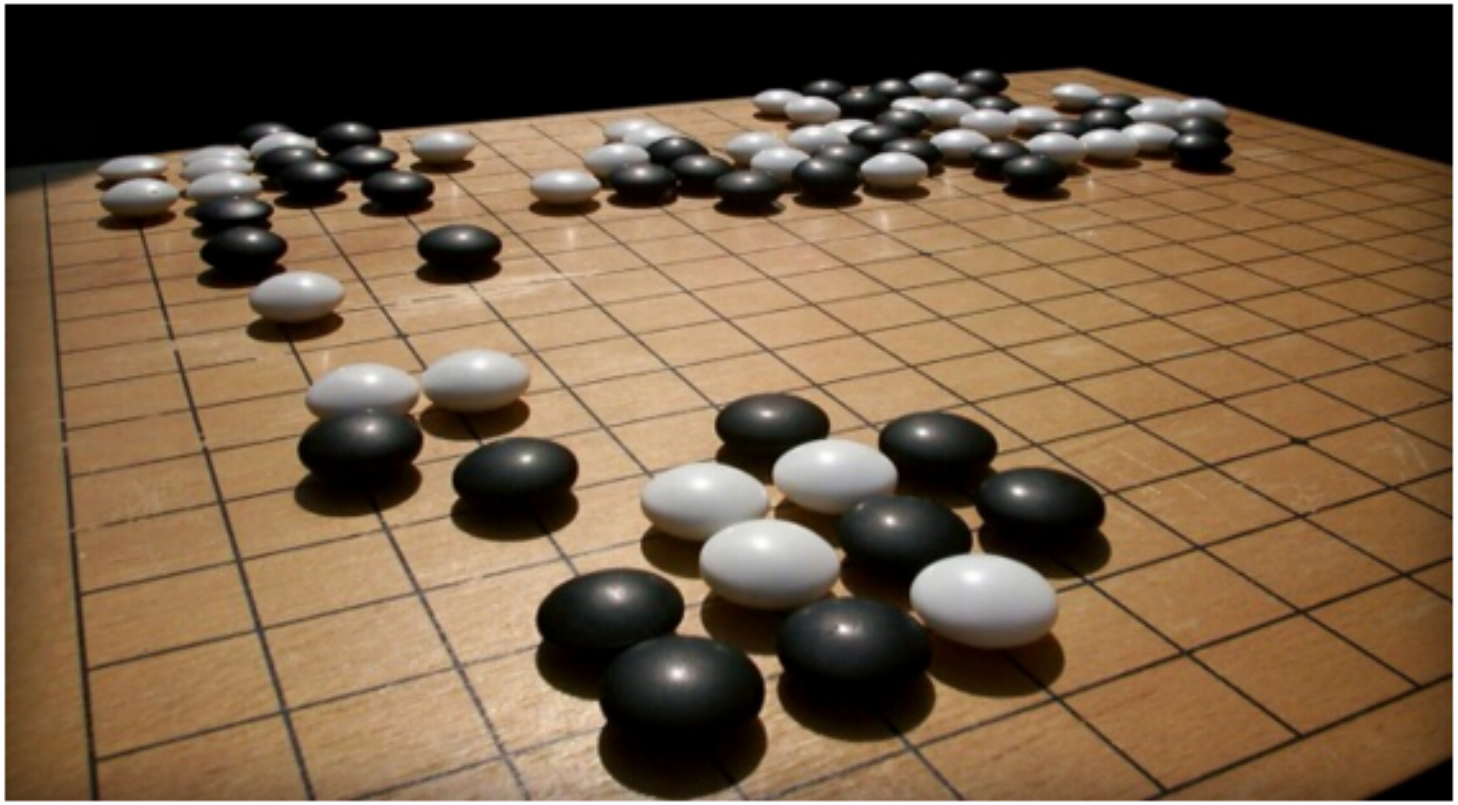
By JOHN MARKOFF
Published: November 23, 2012

Using an artificial intelligence technique inspired by theories about how the brain recognizes patterns, technology companies are reporting startling gains in fields as diverse as computer vision, speech recognition and the identification of promising new molecules for designing drugs.

Success of Deep Learning

Google's DeepMind AI beats humans at the massively complex game Go

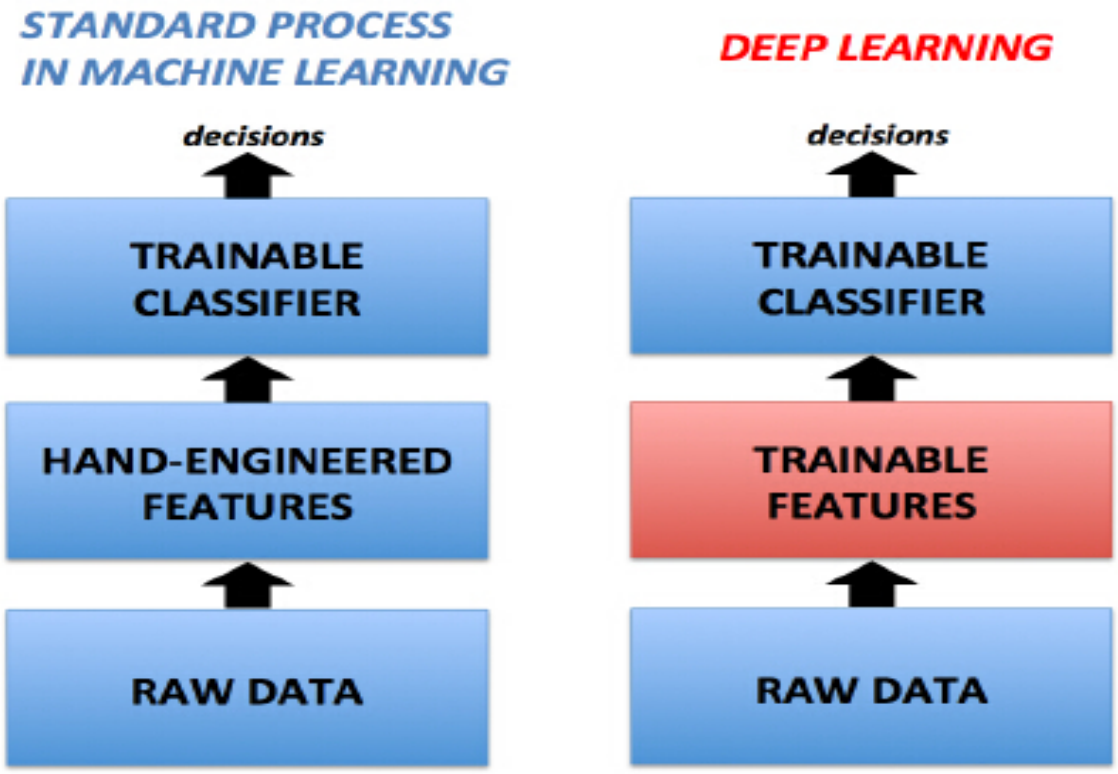
By Ryan Whitwam on January 27, 2016 at 4:00 pm | 11 Comments



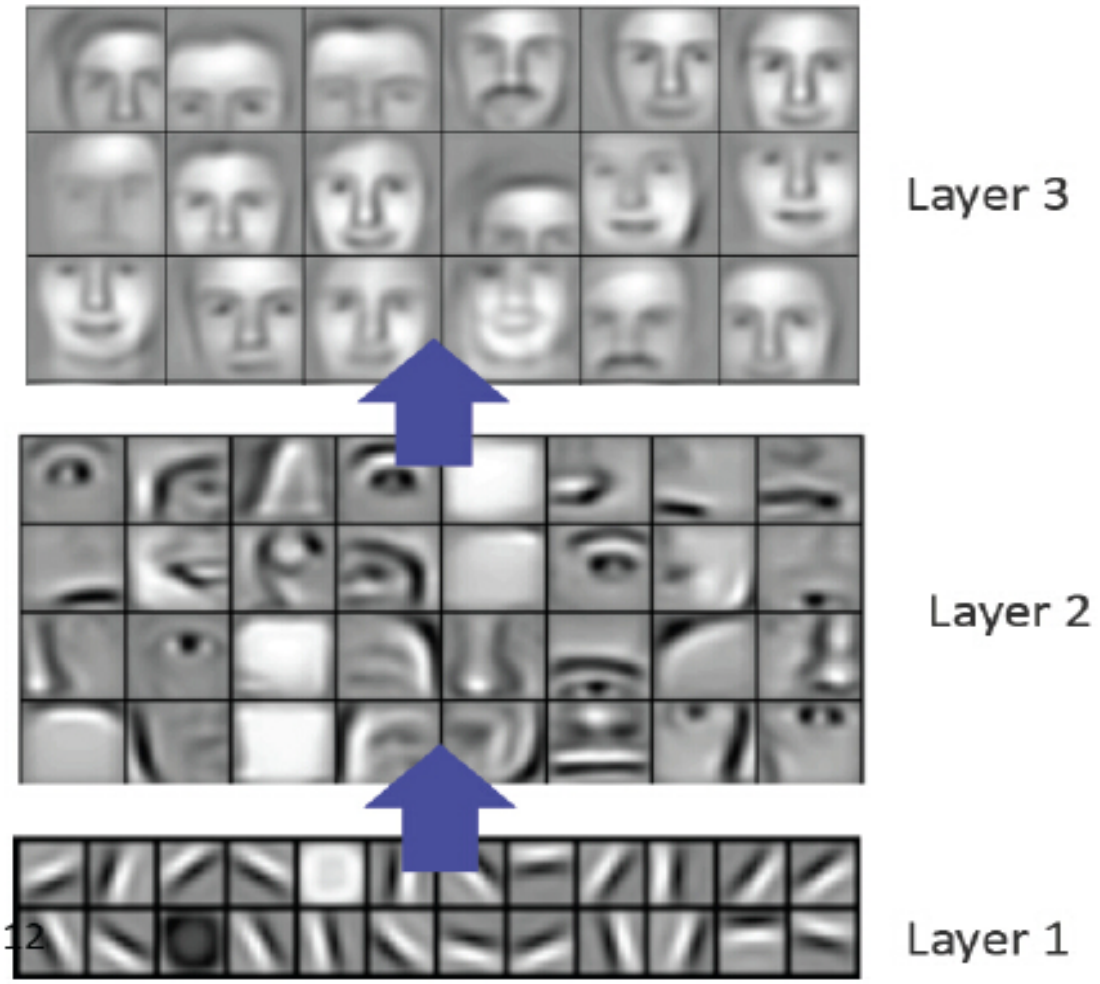
What is Deep Learning?

Definition

A family of methods that uses deep architectures to learn high-level feature representations.



Example of trainable features

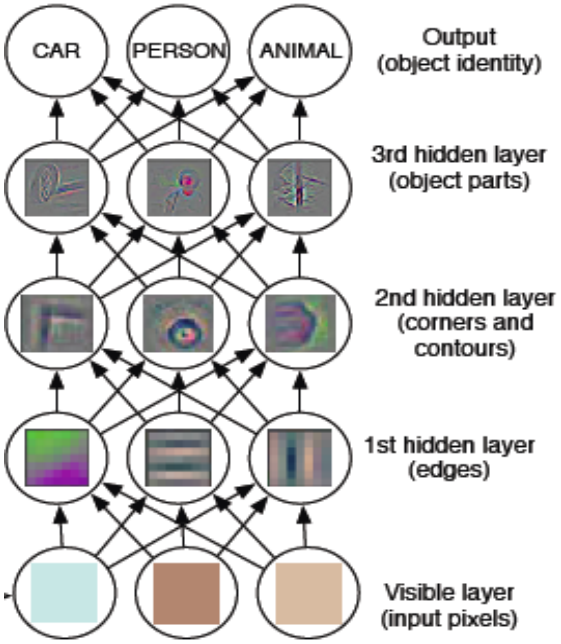


Why do we need deep model?

Why Deep Model?

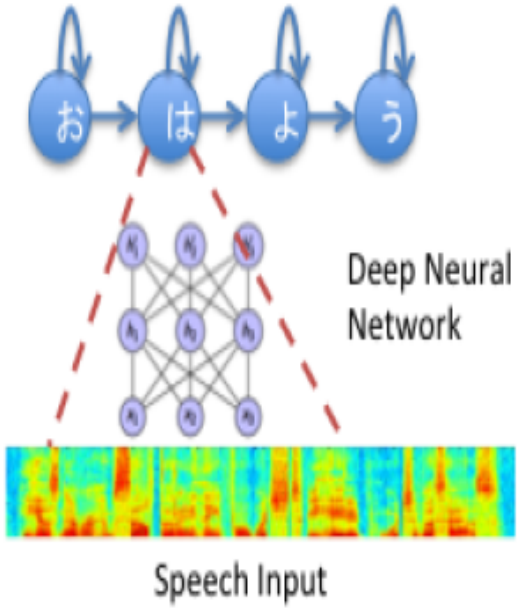
1) Fill in the gap between low-level feature and semantic meaning
2) Learn useful high-level abstraction with less variant/noise

object



pixels

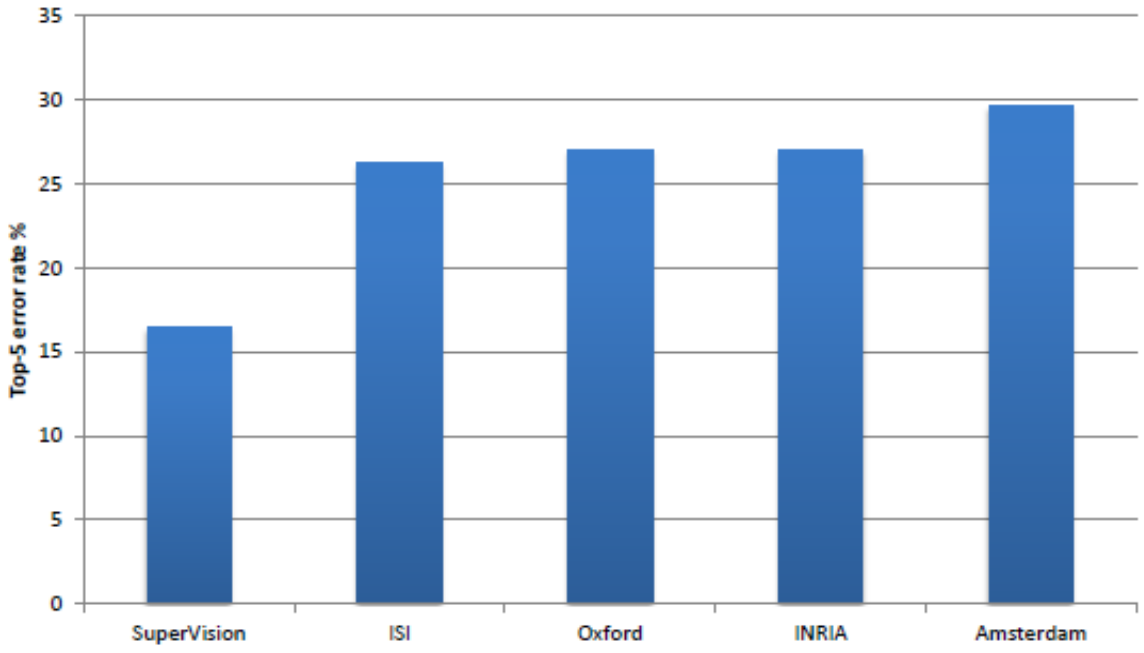
sentence



signals

Success in Computer Vision

- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error



Success in Speech Recognition

Number of hidden layers	Word Error Rate %
1	16
2	12.8
4	11.4
8	10.9
12	11.1

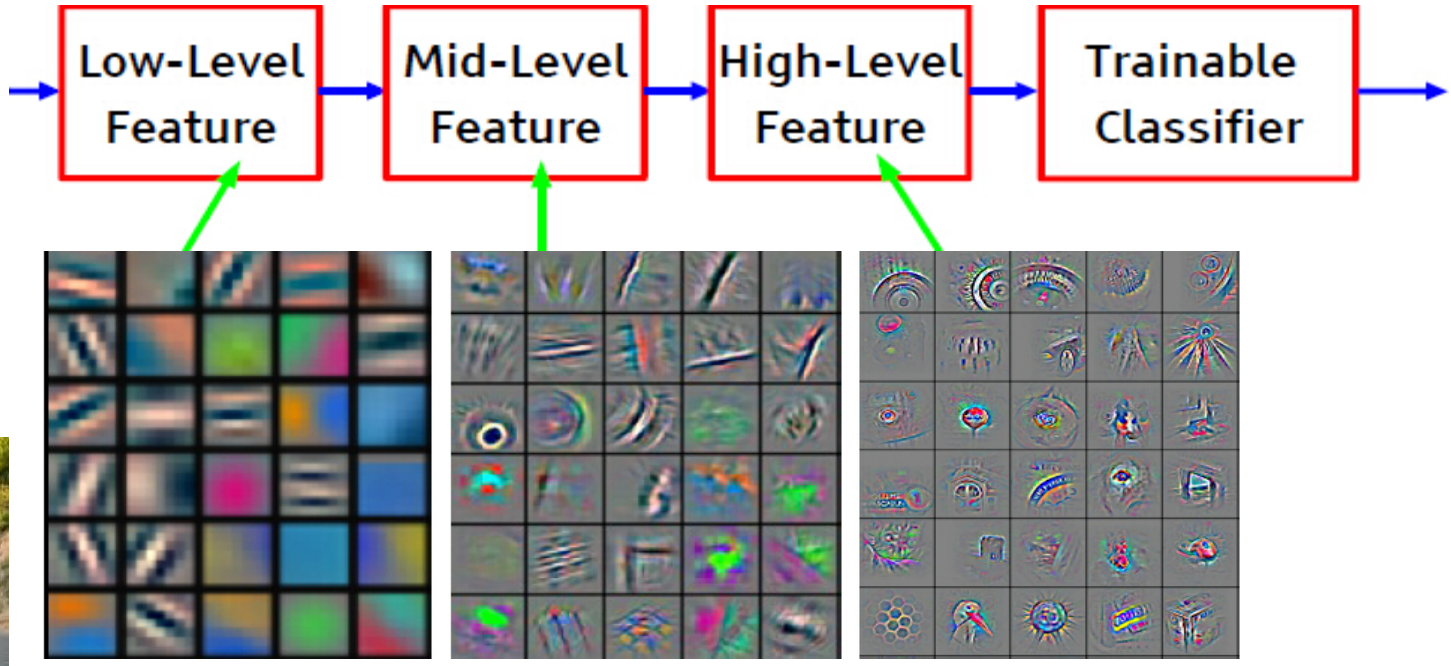
GMM baseline: 15.4%

Zeiler et al. “On rectified linear units for speech recognition” ICASSP 2013

Why do we need deep model?

Why Deep Model?

1) Fill in the gap between low-level feature and semantic meaning
2) Learn useful high-level abstractions with less variant/noise



Course Outline

- Goal:

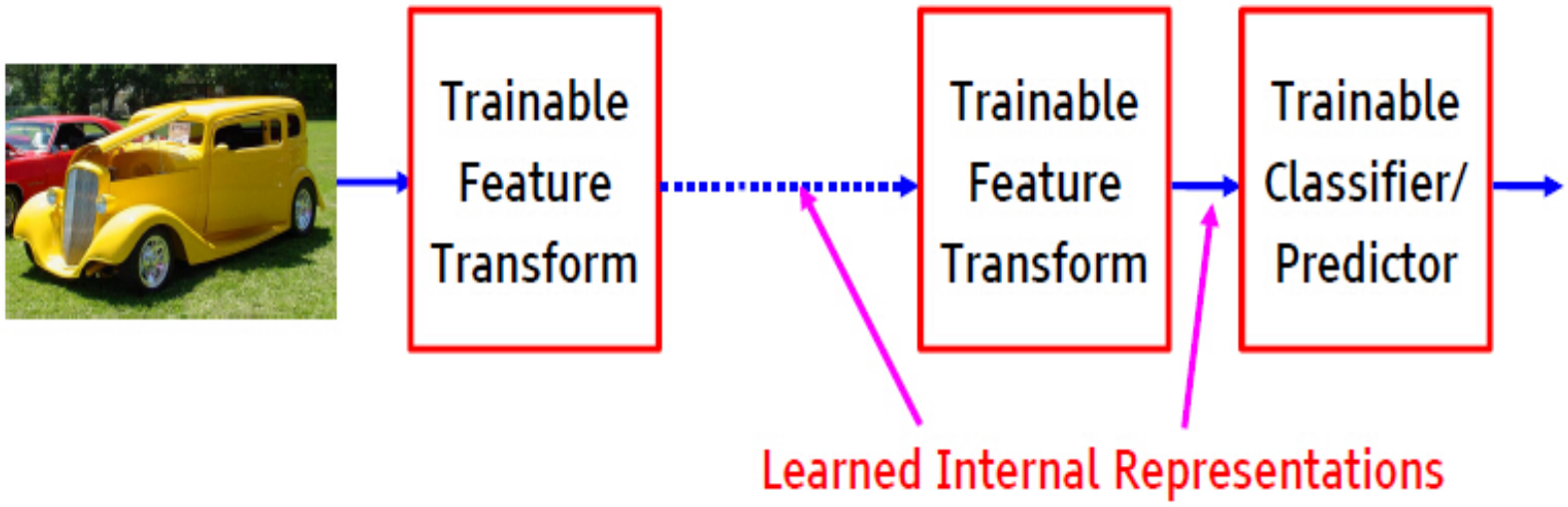
  To understand the foundations of neural networks and deep learning, at a level sufficient for reading recent research papers

- Schedule:

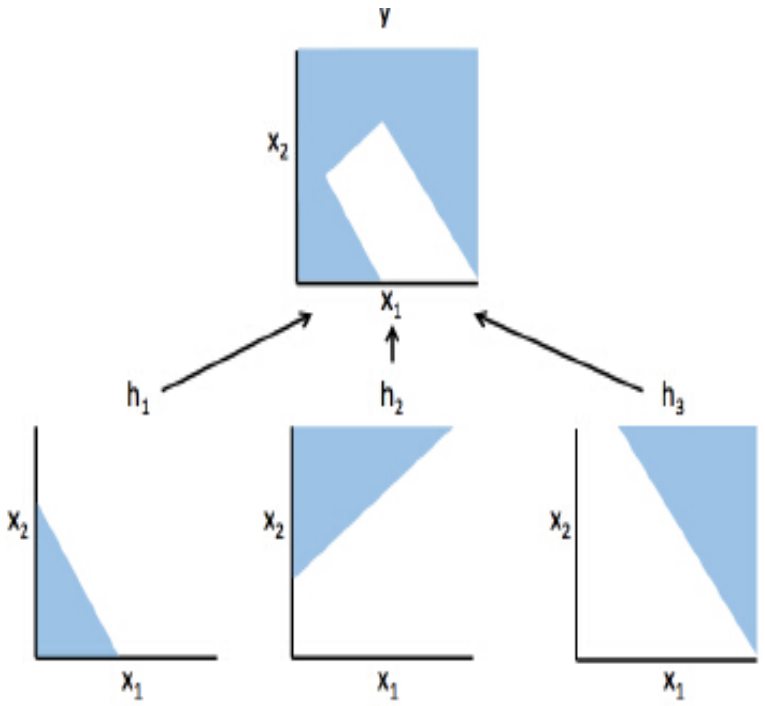
  - Basic of DNN
  - Different types of DNN
  - Advanced topics in DNN

Which structure is deep?

- Structures with more than one (>=2) stage of nonlinear, trainable feature transform



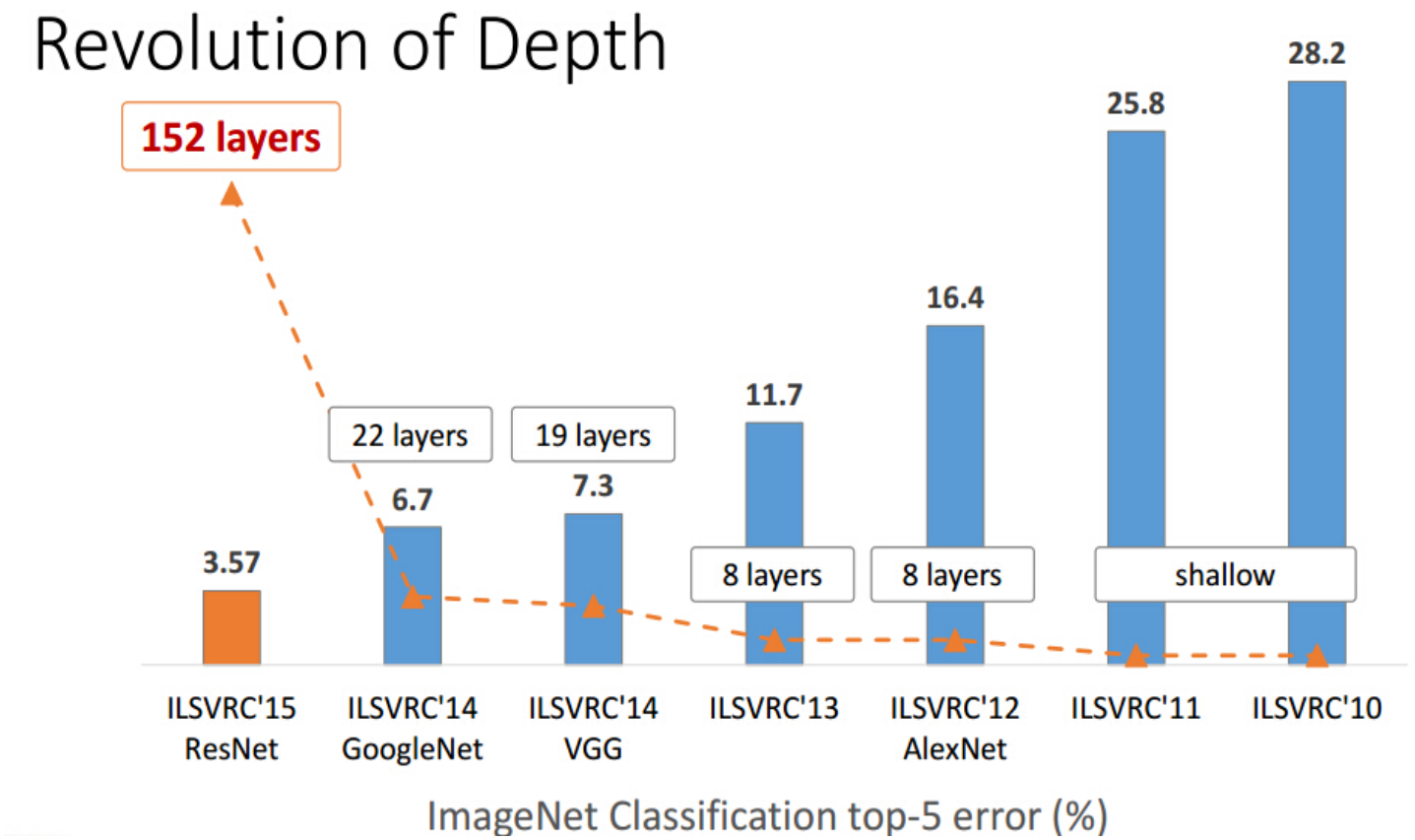
Why is deep structure powerful?



- 1-layer only model liner hyperplanes
- 2-layer with infinite hidden nodes can express any continuous functions
- >2-layer can do this with less nodes

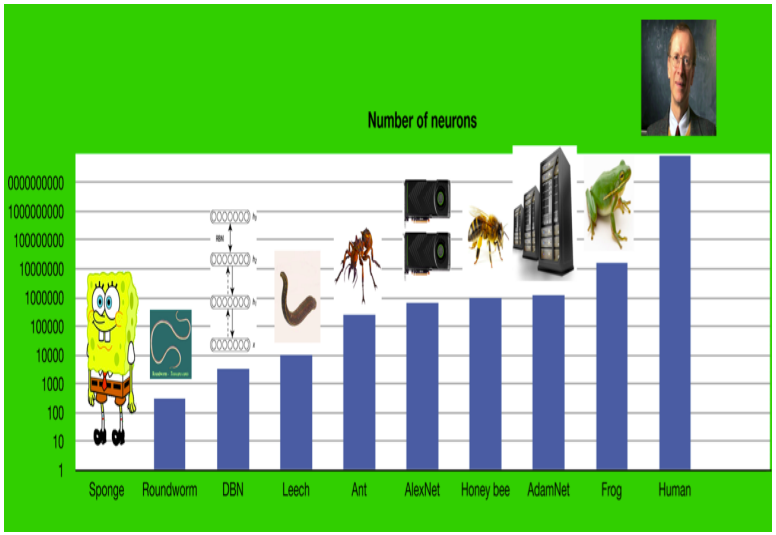
Deeper = Better?

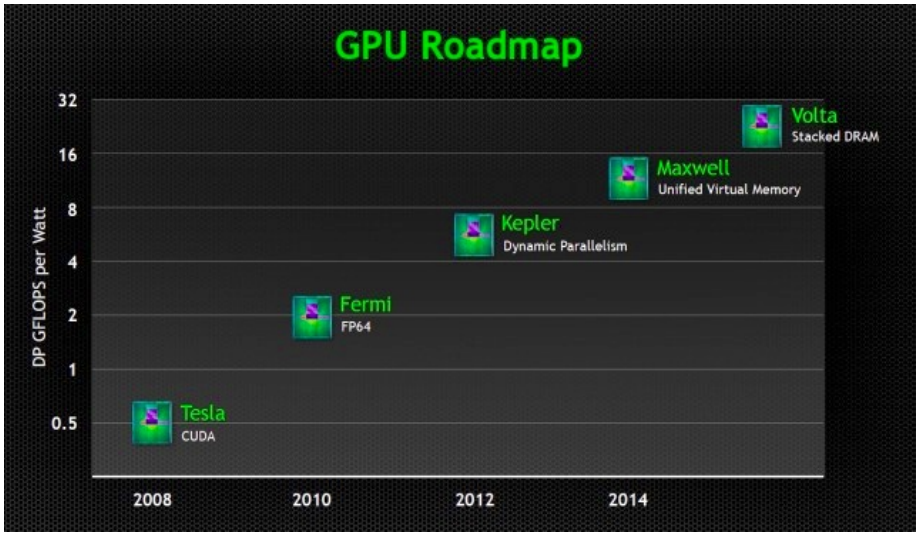
- Yes (for performance)



Deeper = Better?

- No ☺
- You need to consider the deep net structure, the sufficiency of training set, and balance of effective vs. efficiency etc.
- Generalization and Optimization

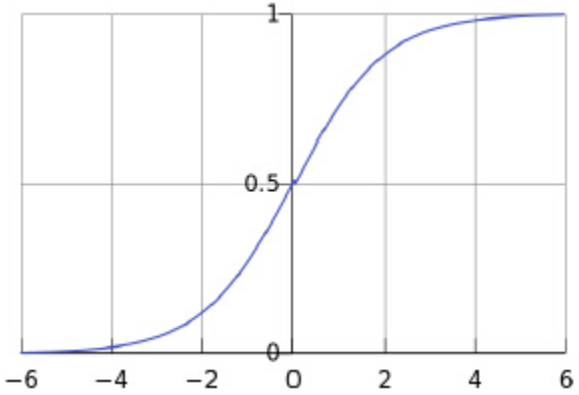




Basic of DL I

Function model: $f(x) = \sigma(w^T \cdot x + b)$

- Parameters: vector $w \in R^d$, b is scalar bias term
- σ is a non-linearity, e.g. sigmoid: $\sigma(z) = 1/(1 + \exp(-z))$
- For simplicity, sometimes write $f(x) = \sigma(w^T x)$ where $w = [w; b]$ and $x = [x; 1]$

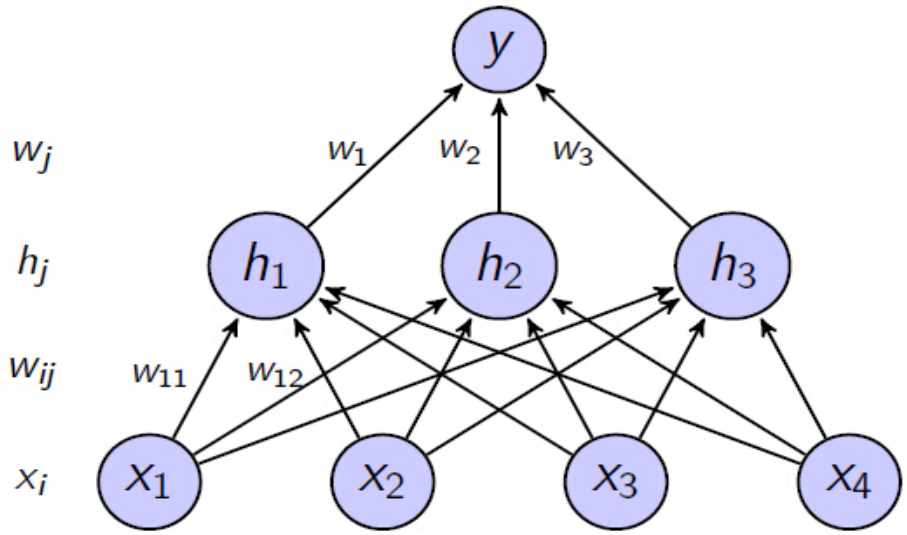


Assume Squared-Error* $Loss(w) = \frac{1}{2}\sum_m (\sigma(w^T x^{(m)}) - y^{(m)})^2$

Gradient: $\nabla_w Loss = \sum_m \left[\sigma(w^T x^{(m)}) - y^{(m)}\right] \sigma'(w^T x^{(m)}) x^{(m)}$

- General form of gradient: $\sum_m Error^{(m)} * \sigma'(in^{(m)}) * x^{(m)}$

Basic of DL II



$$Loss = \sum_k \frac{1}{2} [\sigma(in_k) - y_k]^2$$

$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i))$$

$$\frac{\partial Loss}{\partial w_{jk}} = \frac{\partial Loss}{\partial in_k} \frac{\partial in_k}{\partial w_{jk}} = \delta_k \frac{\partial(\sum_j w_{jk} h_j)}{\partial w_{jk}} = \delta_k h_j$$

$$\frac{\partial Loss}{\partial w_{ij}} = \frac{\partial Loss}{\partial in_j} \frac{\partial in_j}{\partial w_{ij}} = \delta_j \frac{\partial(\sum_j w_{ij} x_i)}{\partial w_{ij}} = \delta_j x_i$$

$$\delta_k = \frac{\partial}{\partial in_k} \left(\sum_k \frac{1}{2} [\sigma(in_k) - y_k]^2 \right) = [\sigma(in_k) - y_k] \, \sigma'(in_k)$$

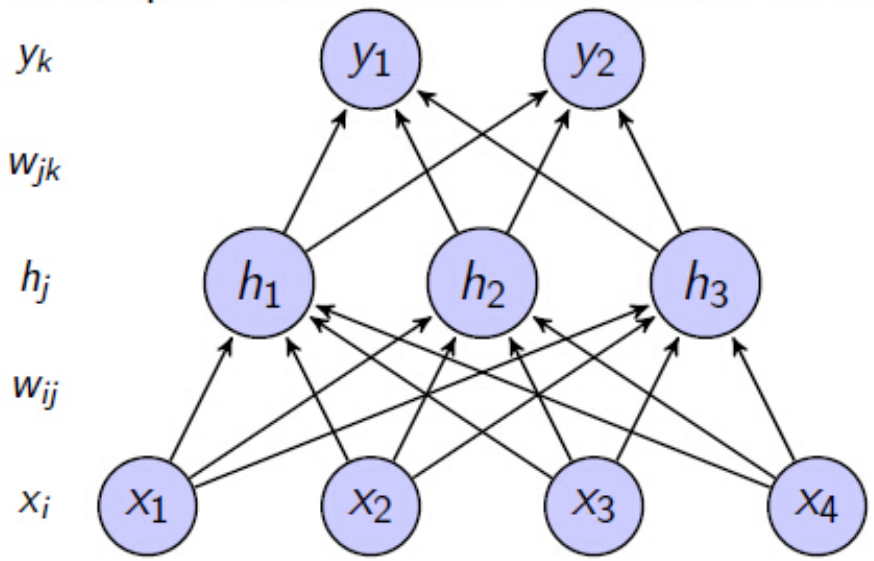
$$\delta_j = \sum_k \frac{\partial Loss}{\partial in_k} \frac{\partial in_k}{\partial in_i} = \sum_k \delta_k \cdot \frac{\partial}{\partial in_i} \left(\sum_j w_{jk} \sigma(in_j) \right) = [\sum_k \delta_k w_{jk}] \, \sigma'(in_j)$$

Basic of DL III

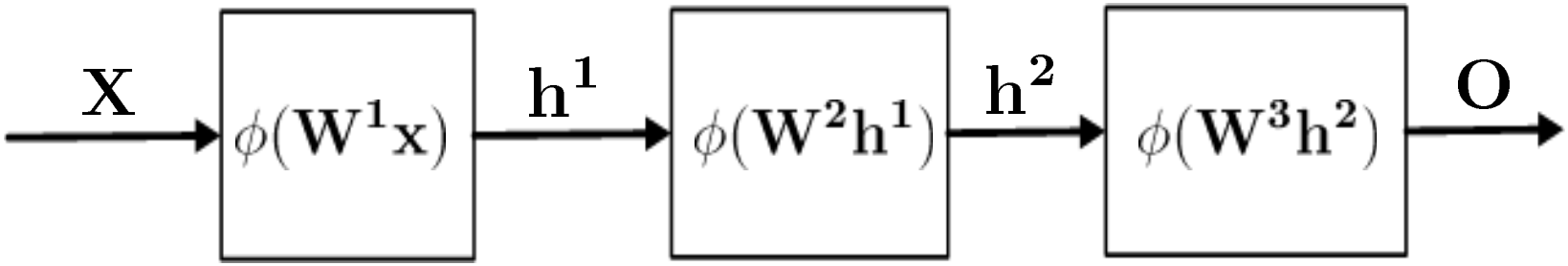
All updates involve some scaled error from output $*$ input feature:

- $\frac{\partial Loss}{\partial w_{jk}} = \delta_k h_j$ where $\delta_k = [\sigma(in_k) - y_k]\,\sigma'(in_k)$
- $\frac{\partial Loss}{\partial w_{ij}} = \delta_j x_i$ where $\delta_j = [\sum_k \delta_k w_{jk}]\,\sigma'(in_j)$

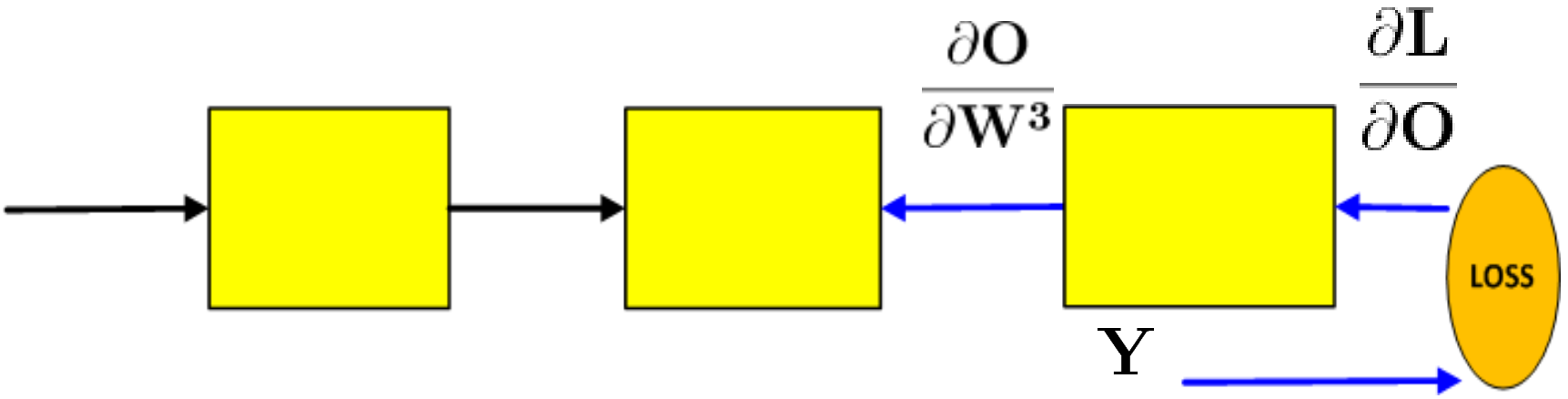
After forward pass, compute δ_k from final layer, then δ_j for previous layer. For deeper nets, iterate backwards further.



Forward-backward propagation

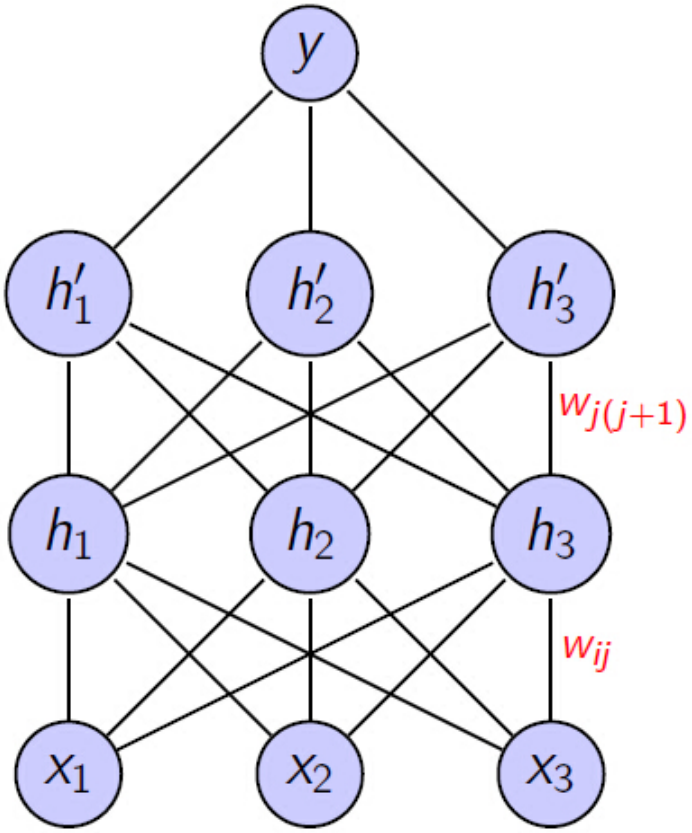


Forward propagation: $h(\mathbf{x}) = \phi(\mathbf{W}\mathbf{x})$



Backward propagation: $\frac{\partial \mathbf{L}}{\partial \mathbf{W^3}} = \frac{\partial \mathbf{L}}{\partial \mathbf{O}} \frac{\partial \mathbf{O}}{\partial \mathbf{W^3}}$ **(Chain Rule)**

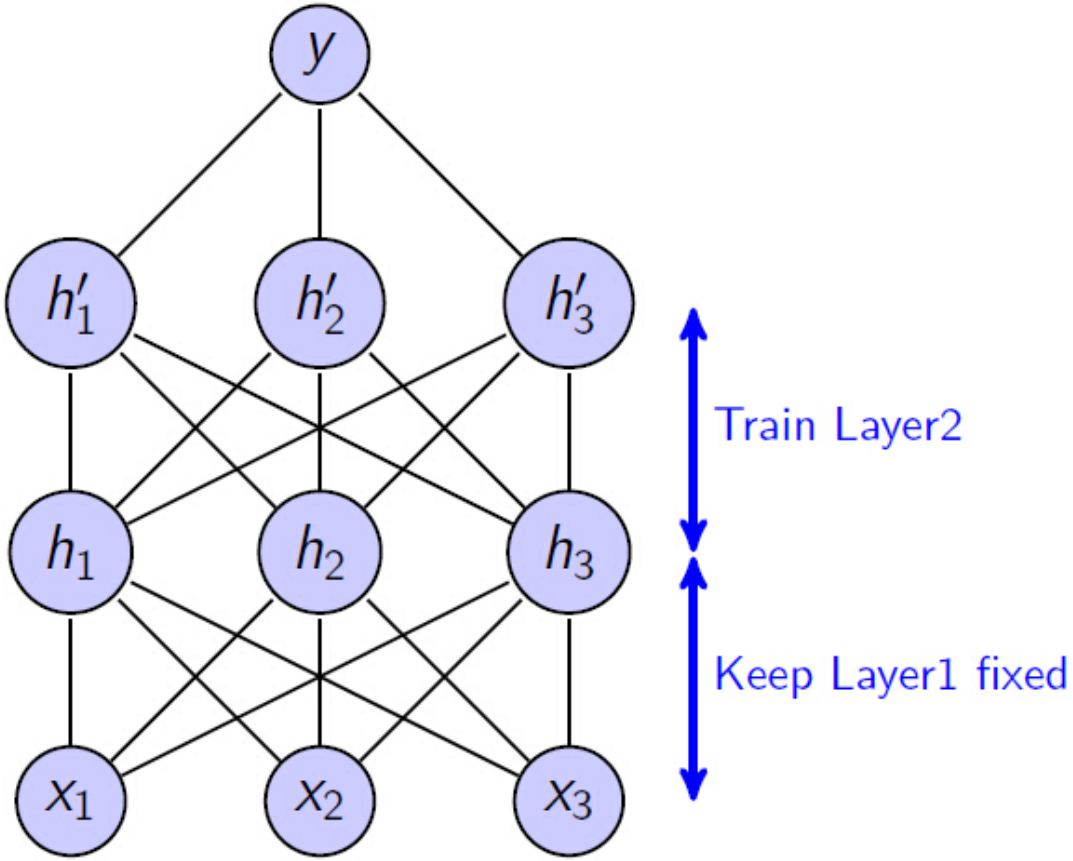
Why are deep architectures hard to train?



- Vanishing gradient problem in backward propagation
- Insufficient training data

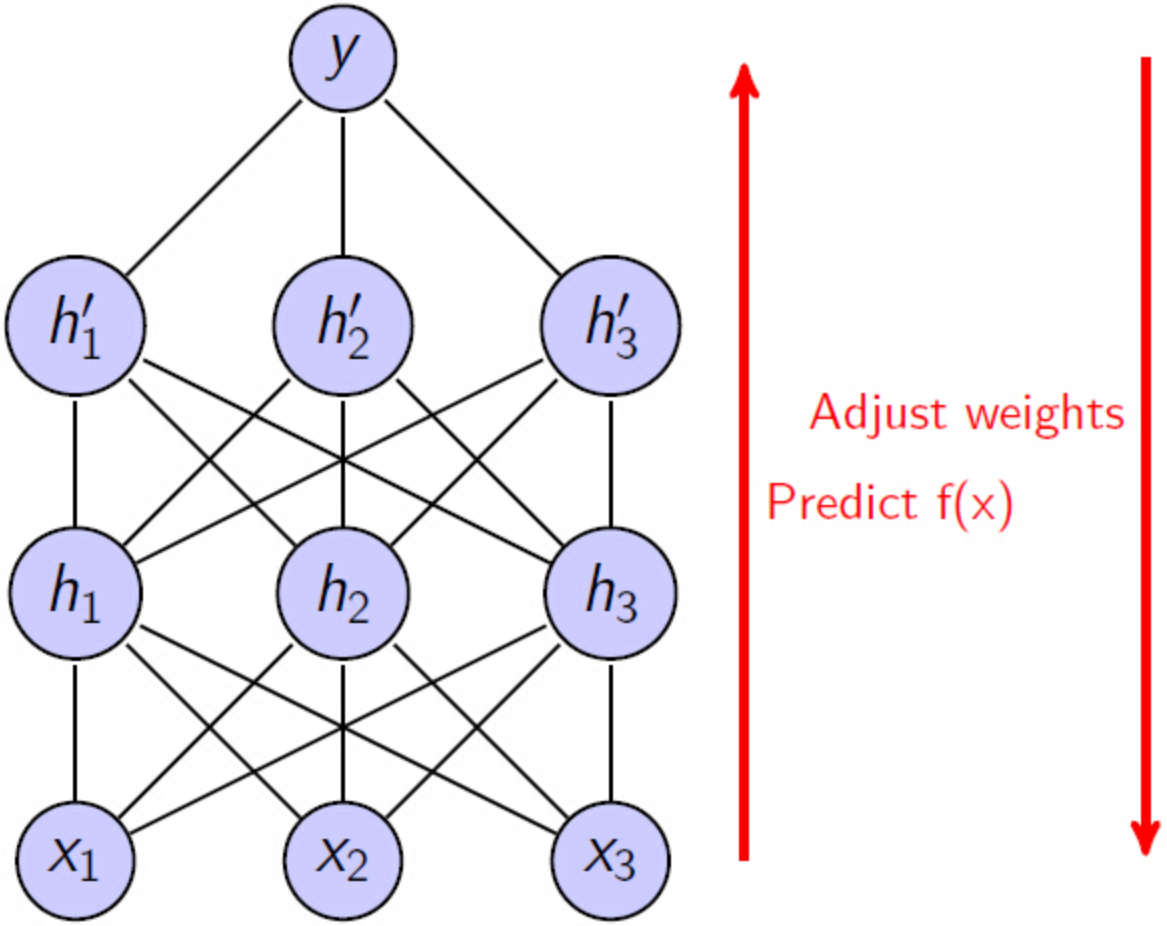
Layer-wise pre-training [Hinton et al. 2006]

First, train one layer at a time, optimizing data-likelihood objective $P(x)$



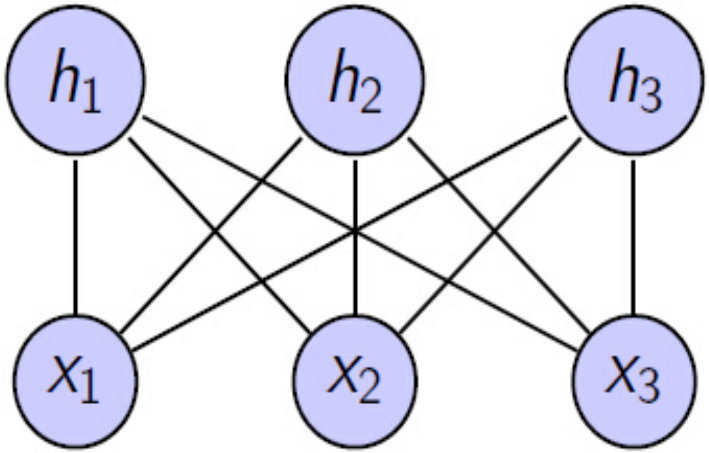
Layer-wise pre-training [Hinton et al. 2006]

Finally, fine-tune labeled objective $P(y|x)$ by Backpropagation

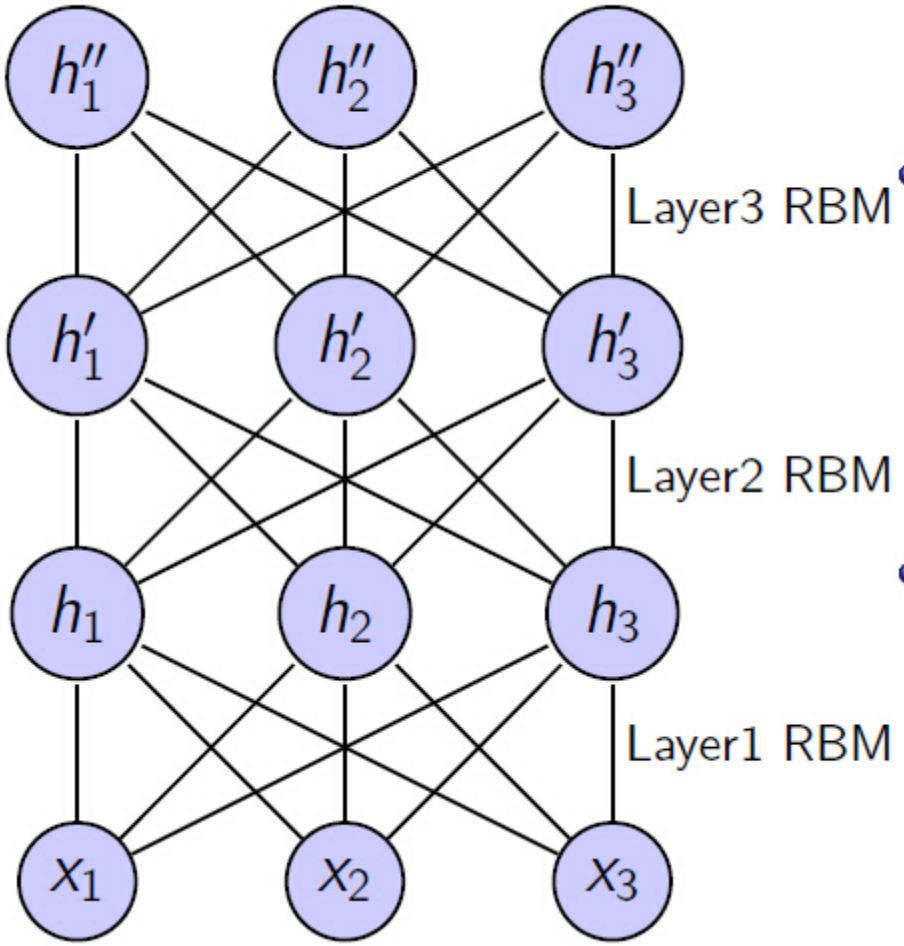


Restricted Boltzmann Machine (RBM)

- RBM is a simple energy-based model: $p(x, h) = \frac{1}{Z_\theta} \exp\left(-E_\theta(x, h)\right)$
  - with only h-x interactions: $E_\theta(x, h) = -x^T W h - b^T x - d^T h$
  - here, we assume h_j and x_i are binary variables
  - normalizer: $Z_\theta = \sum_{(x,h)} \exp(-E_\theta(x, h))$ is called partition function

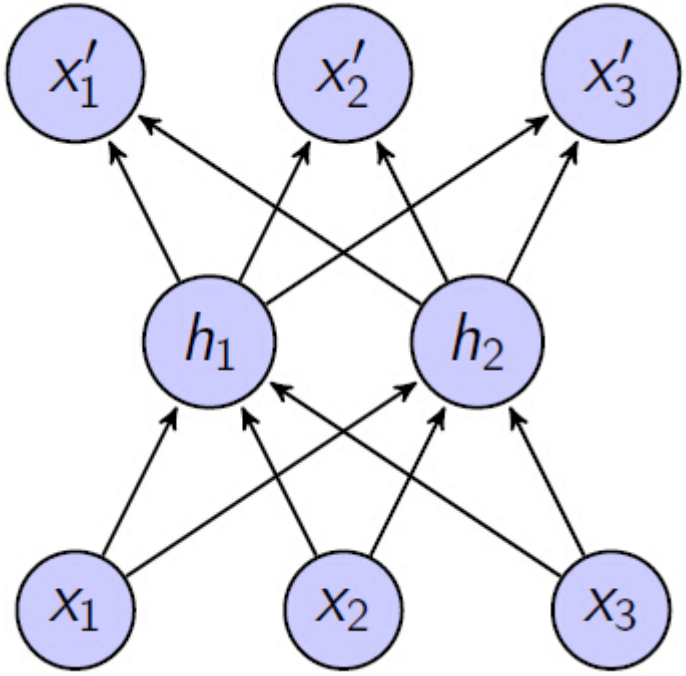


Deep Belief Nets (DBN) = Stacked RBM



- DBN defines a probabilistic generative model $p(x) = \sum_{h,h',h''} p(x|h)p(h|h')p(h',h'')$ (top 2 layers is interpreted as a RBM; lower layers are directed sigmoids)
- Stacked RBMs can also be used to initialize a Deep Neural Network (DNN)

Auto-Encoders: simpler alternatives to RBMs



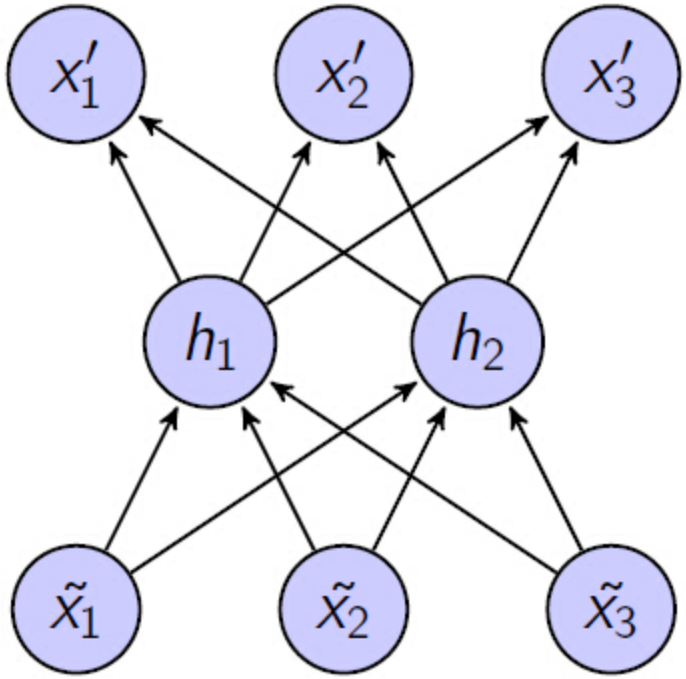
Decoder: $x' = \sigma(W'h + d)$

Encoder: $h = \sigma(Wx + b)$

Encourage h to give small reconstruction error:

- e.g. $Loss = \sum_m ||x^{(m)} - DECODER(ENCODER(x^{(m)}))||^2$
- Reconstruction: $x' = \sigma(W'\sigma(Wx + b) + d)$
- This can be trained with the same Backpropagation algorithm for 2-layer nets, with $x^{(m)}$ as both input and output

Denoising Auto-Encoders



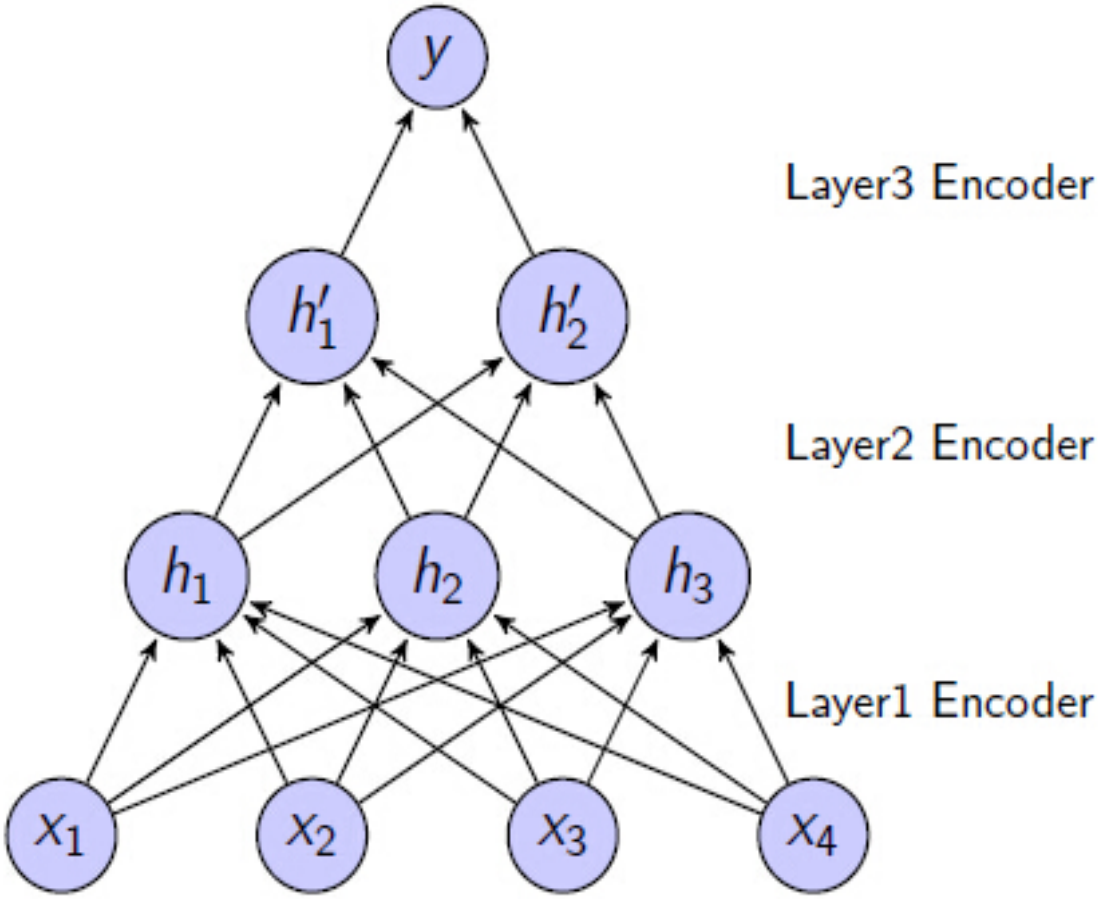
Decoder: $x' = \sigma(W'h + d)$

Encoder: $h = \sigma(W\tilde{x} + b)$

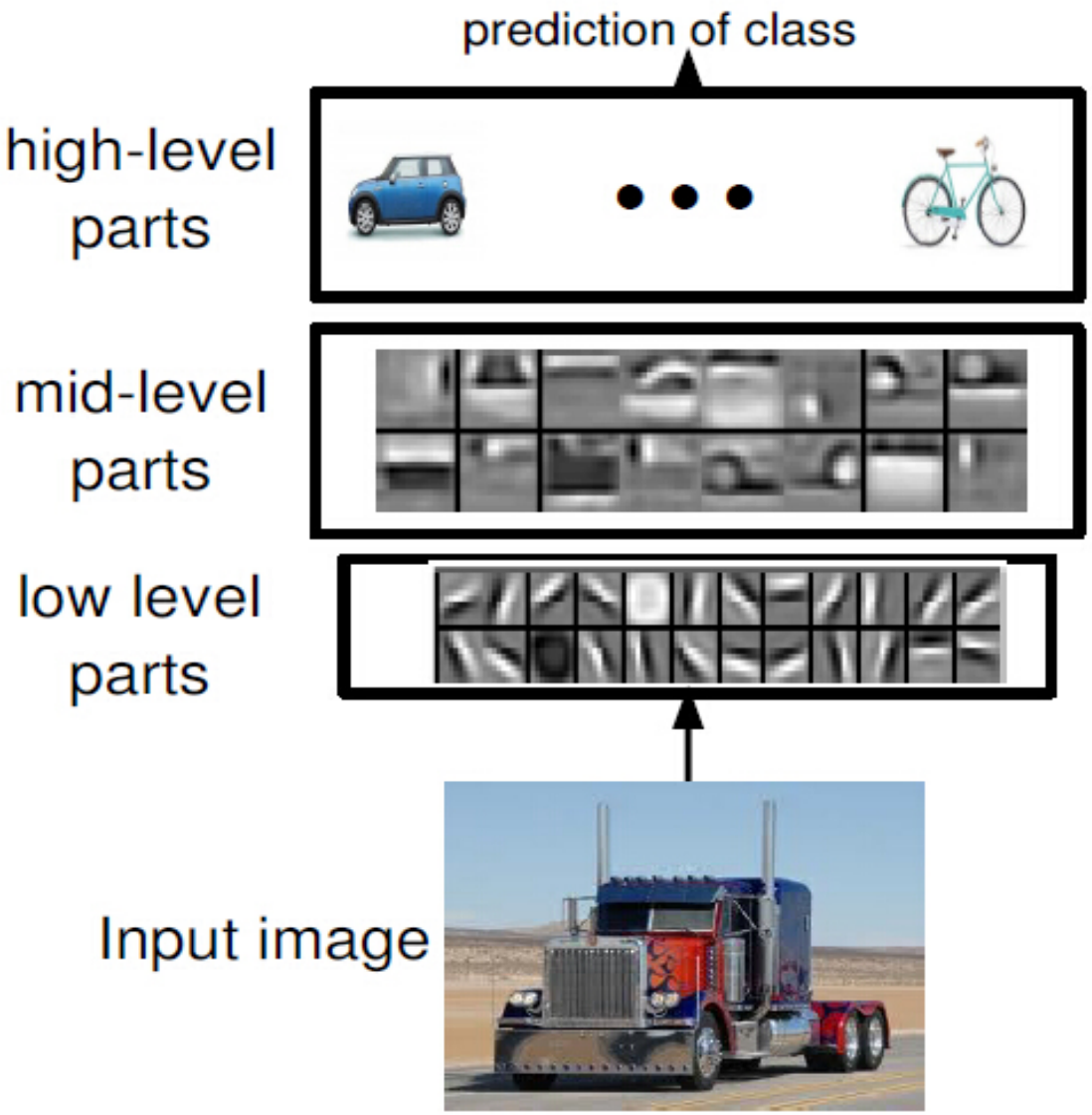
$\tilde{x} = x +$ noise

1. Perturb input data x to $\tilde{x}$ using invariance from domain knowledge.
2. Train weights to reduce reconstruction error with respect to original input: $||x - x'||$

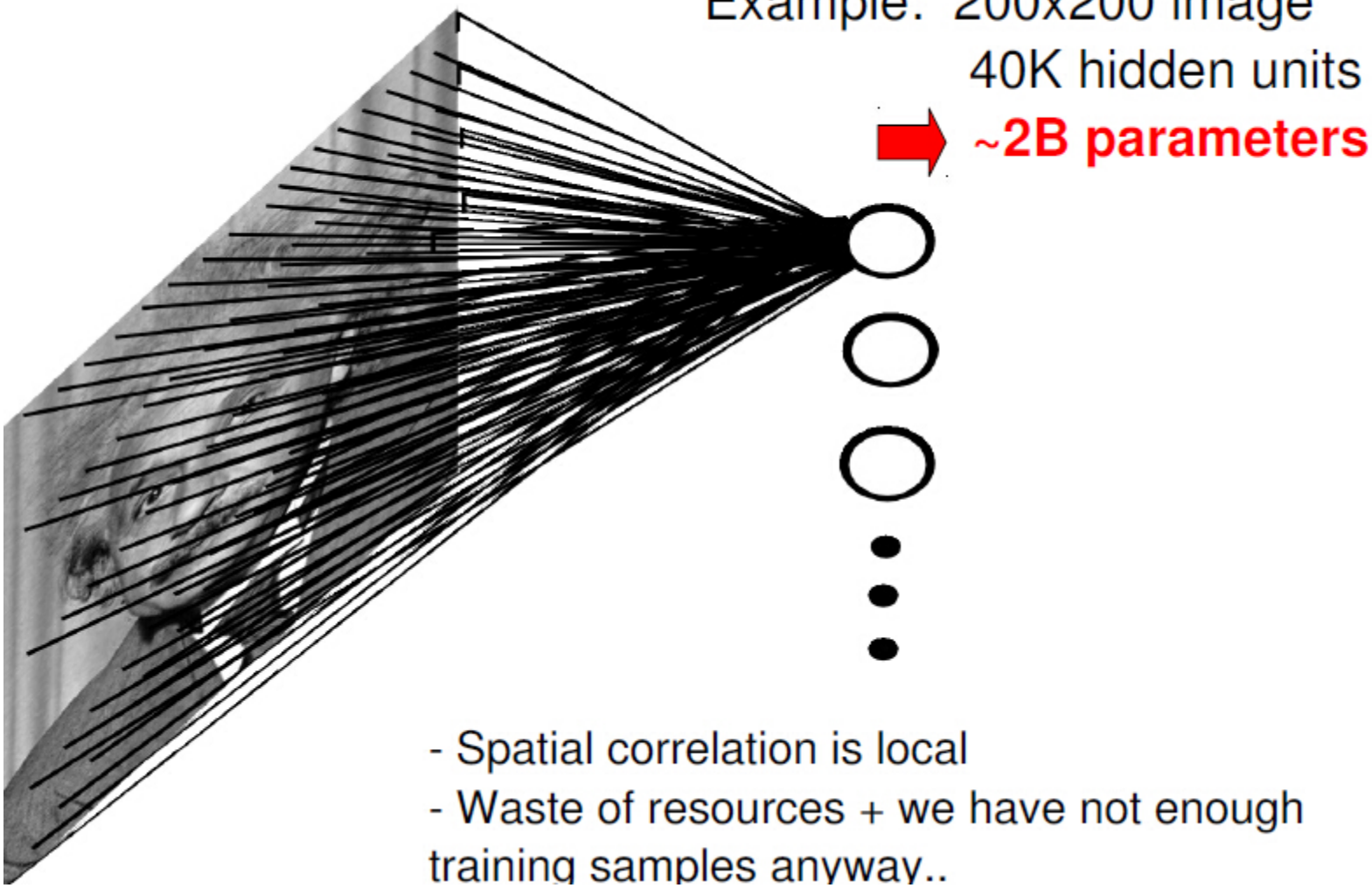
Stacked Auto-Encoders (SAE)



Convolutional Neural Networks

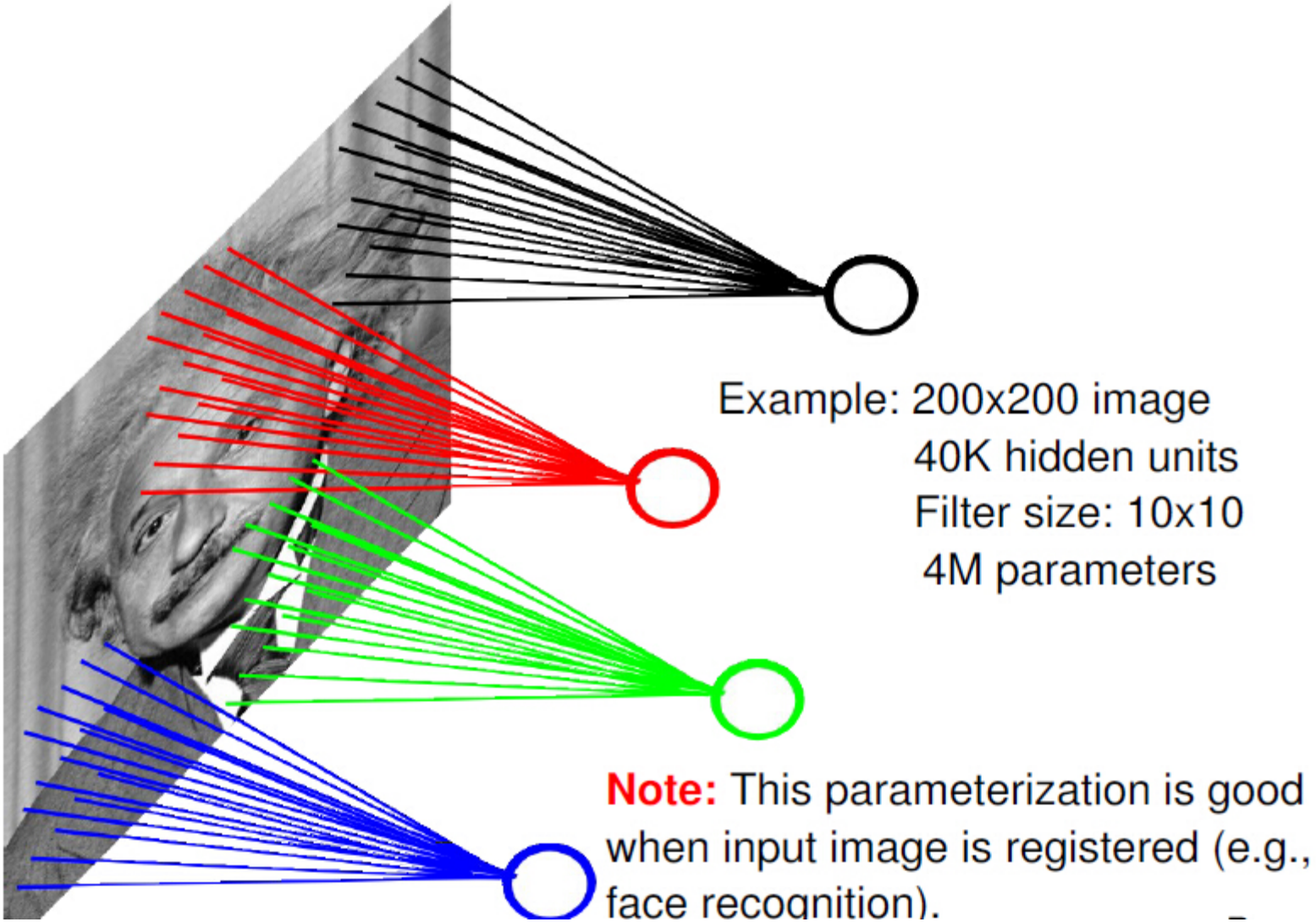


Fully Connected Layer

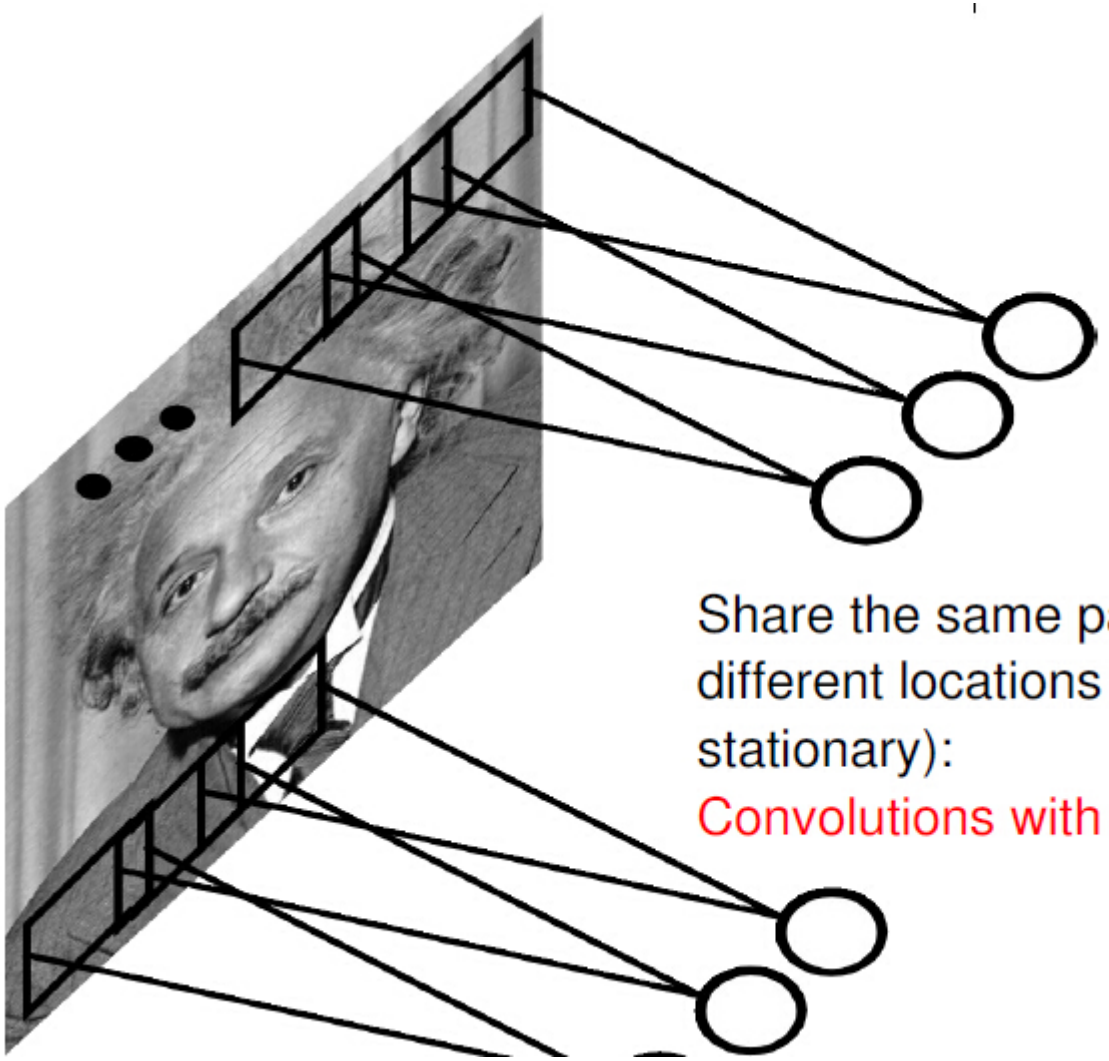


- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Locally Connected Layer



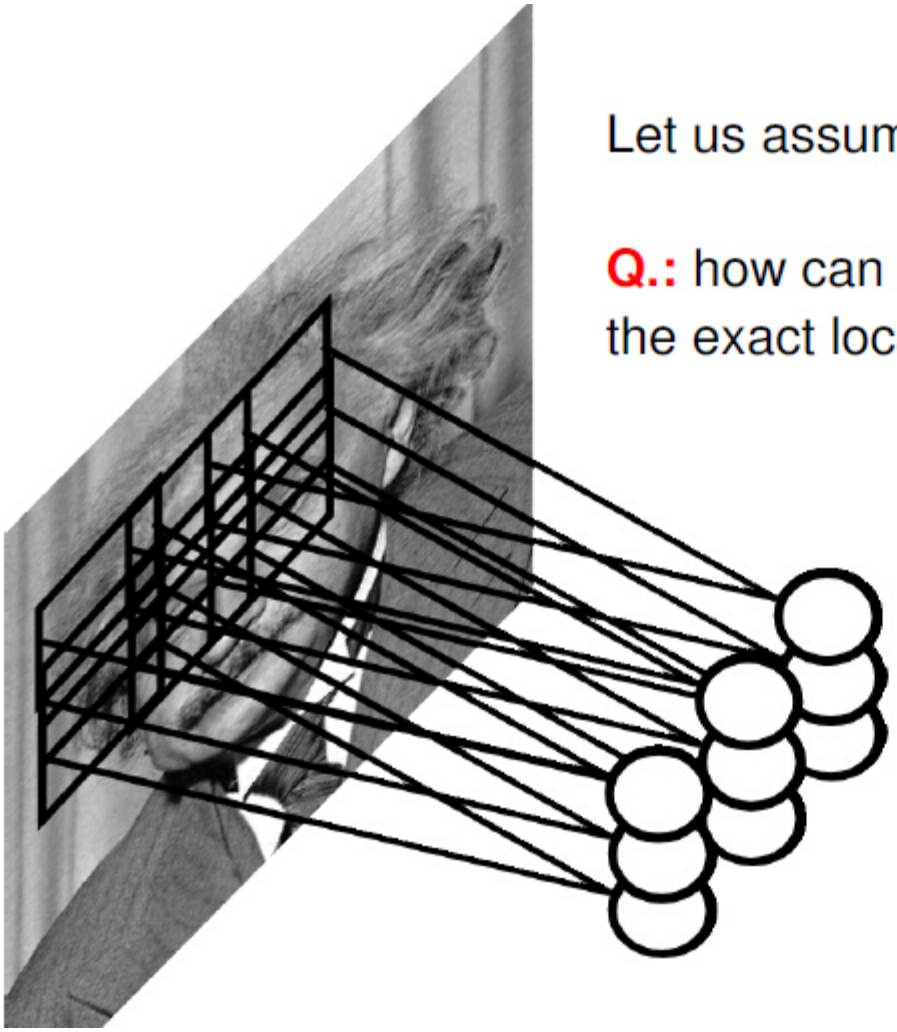
Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels

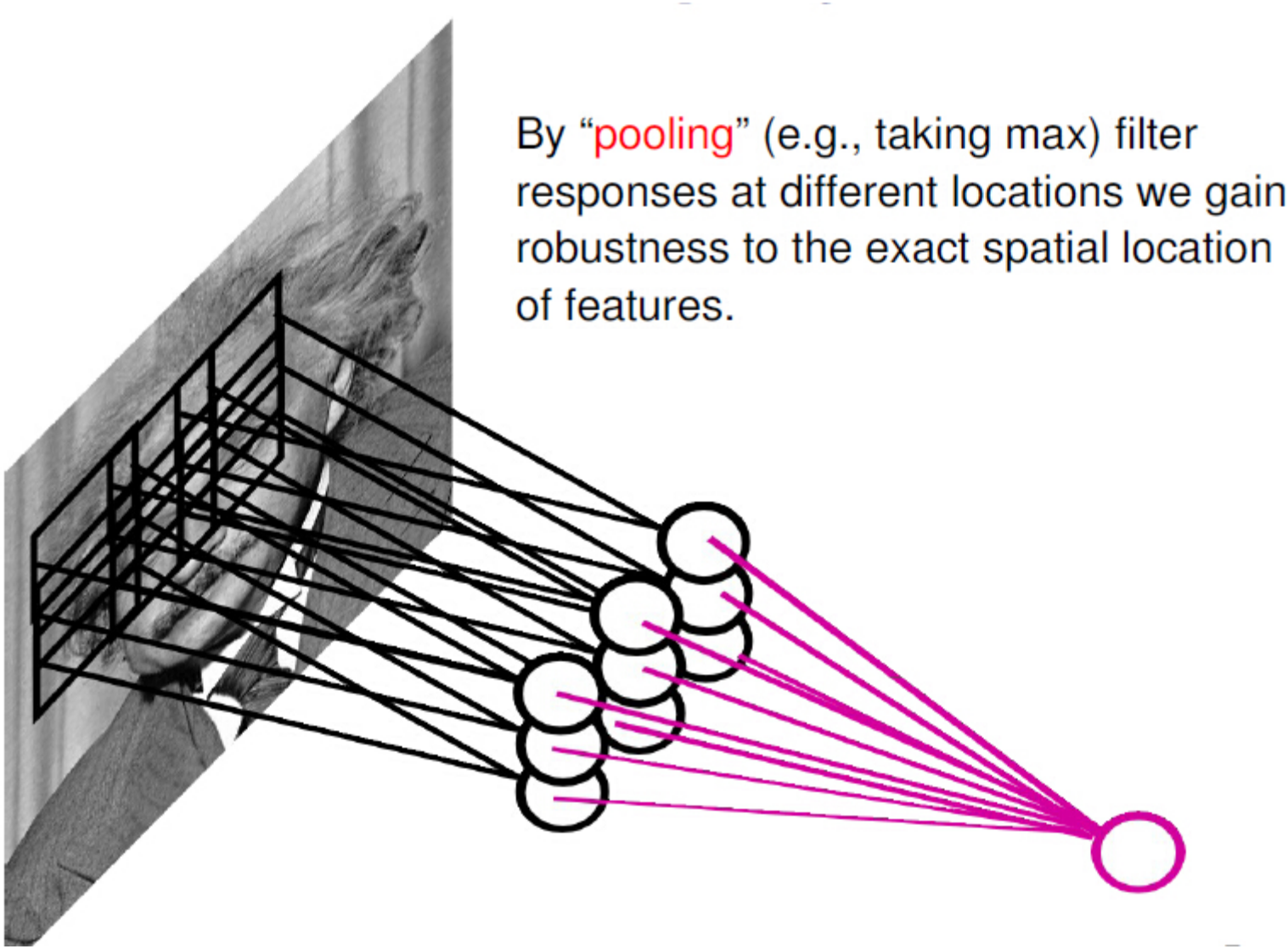
Pooling Layer



Let us assume filter is an “eye” detector.

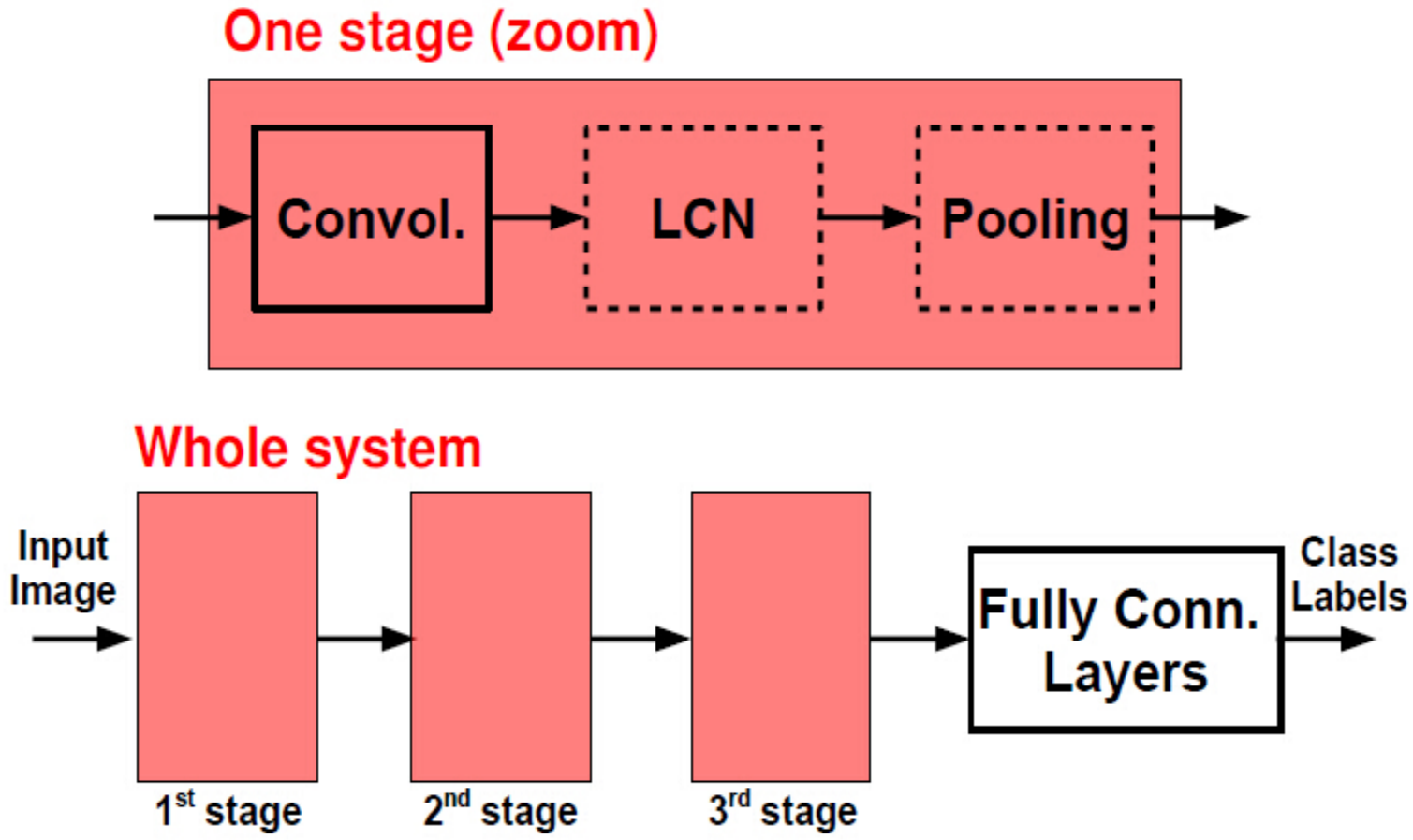
Q.: how can we make the detection robust to the exact location of the eye?

Pooling Layer

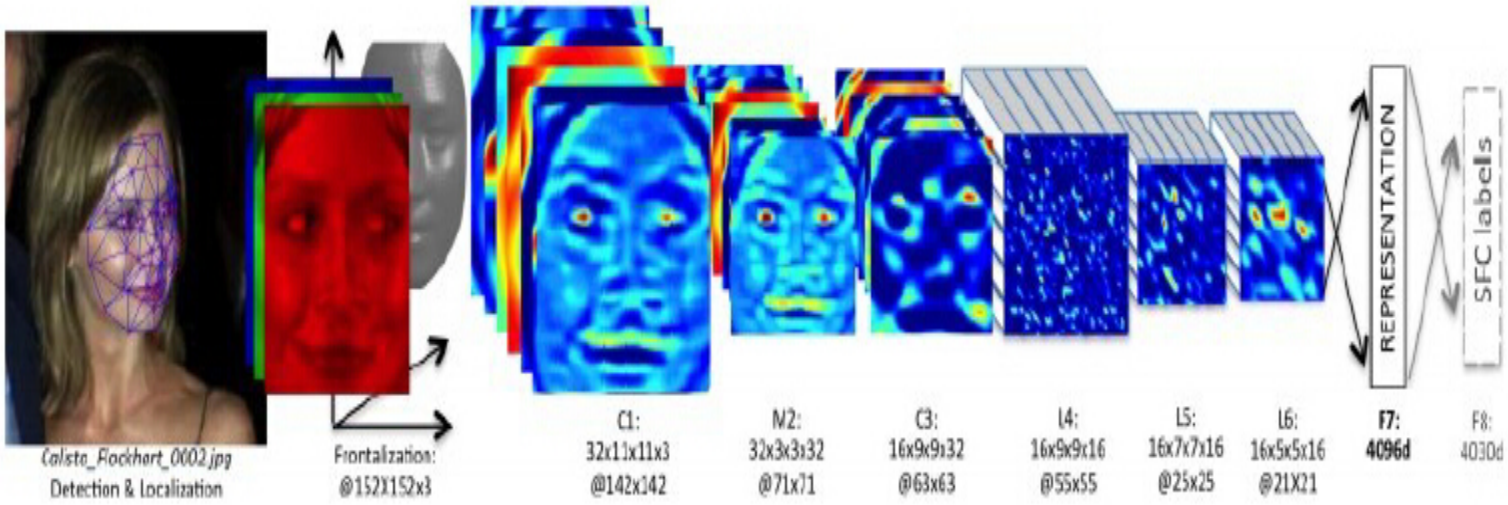


By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

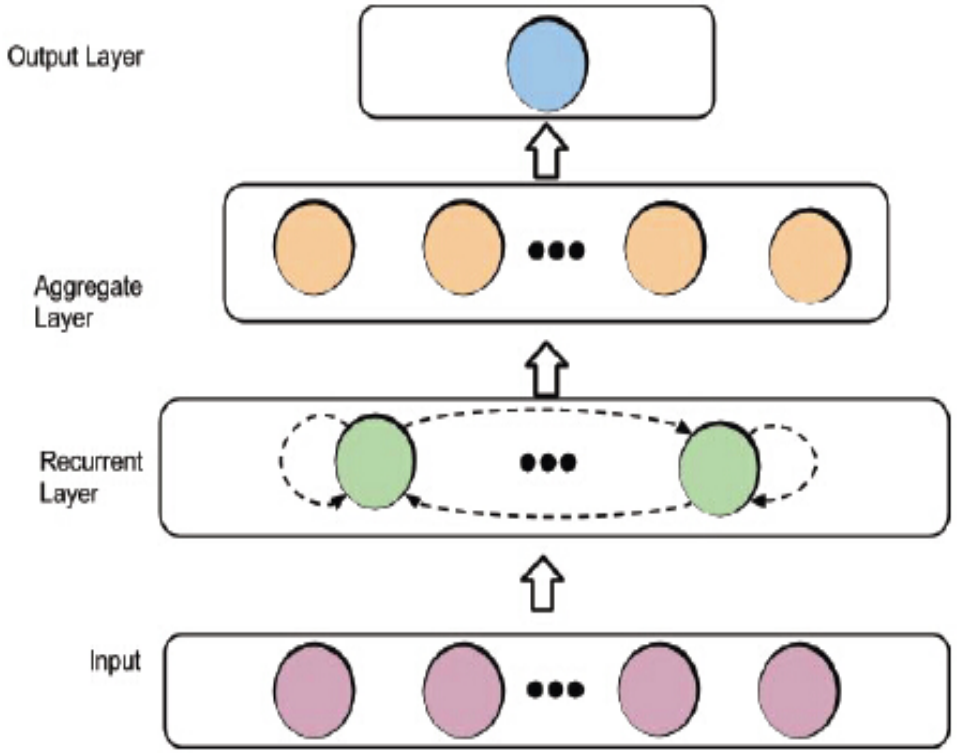
ConvNets: Typical Architecture



Examples – DeepFace

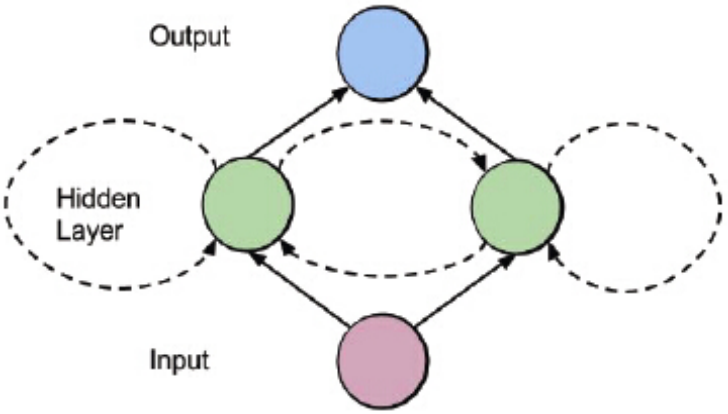


Recurrent Neural Networks



Input: sequential data/time series

Simple Recurrent Neural Net



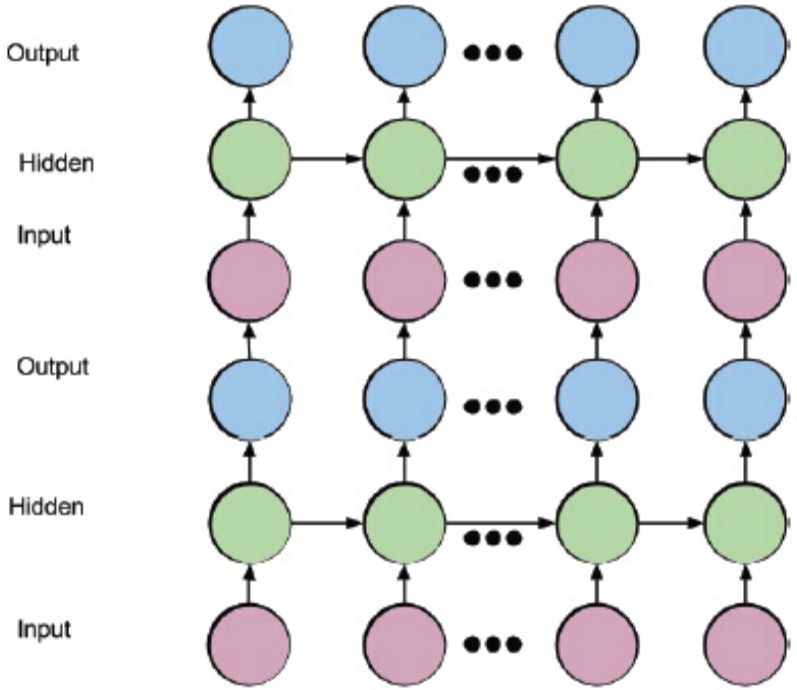
$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y)$$

- Recurrent edges may form cycles, including self-connections
- Optimization is especially difficult due to long-range dependencies
- Vanishing and exploding gradients occur when propagating errors across many time steps[Lipton, 2015]

Stacking Recurrent Layer: deep RNN models

- Common RNN do not accommodate temporal hierarchy [Hermans and Schrauwen, 2013]
- A deep hidden-to-output function can be useful to disentangle the factors of variations in the hidden state



DNN Tools

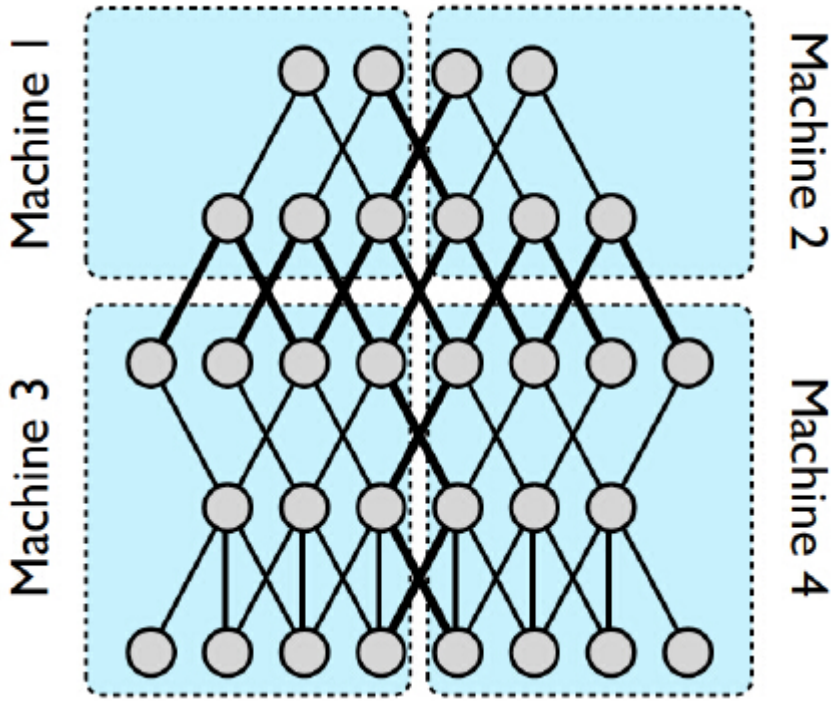
- Caffe: http://caffe.berkeleyvision.org/
- Torch: http:// www.torch.ch
- Theano: http://deeplearning.net/software/theano/
- TensorFlow: https://www.tensorflow.org/
- DeepLearnToolbox: https://github.com/rasmusbergpalm/DeepLearnToolbox
- Pylearn2: https://github.com/lisa-lab/pylearn2

Advanced Topics in DL

- There are many exciting new works out of ICML/NIPS. I just pick 3 that I think is most interesting.

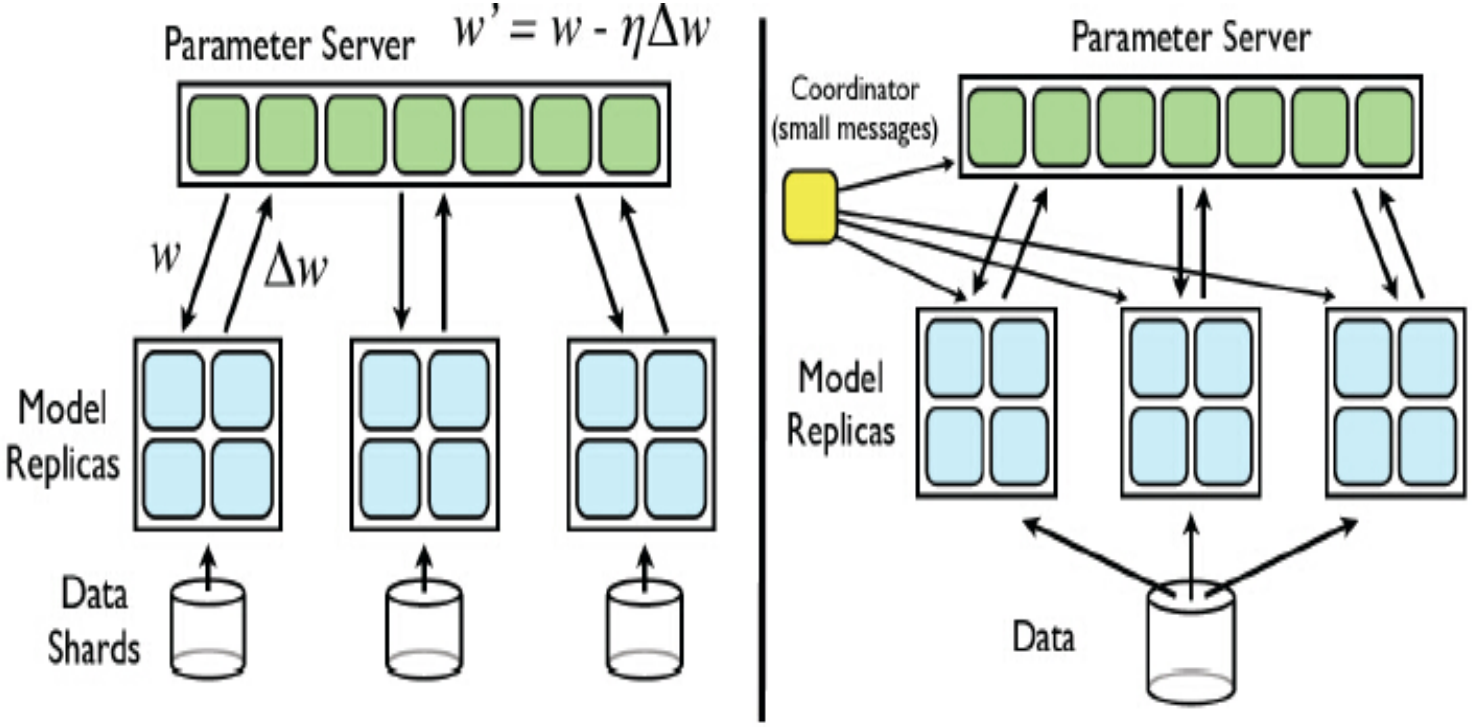
- Scaling to large data
- Parameters reduction
- Deep generative model

Model Parallelism



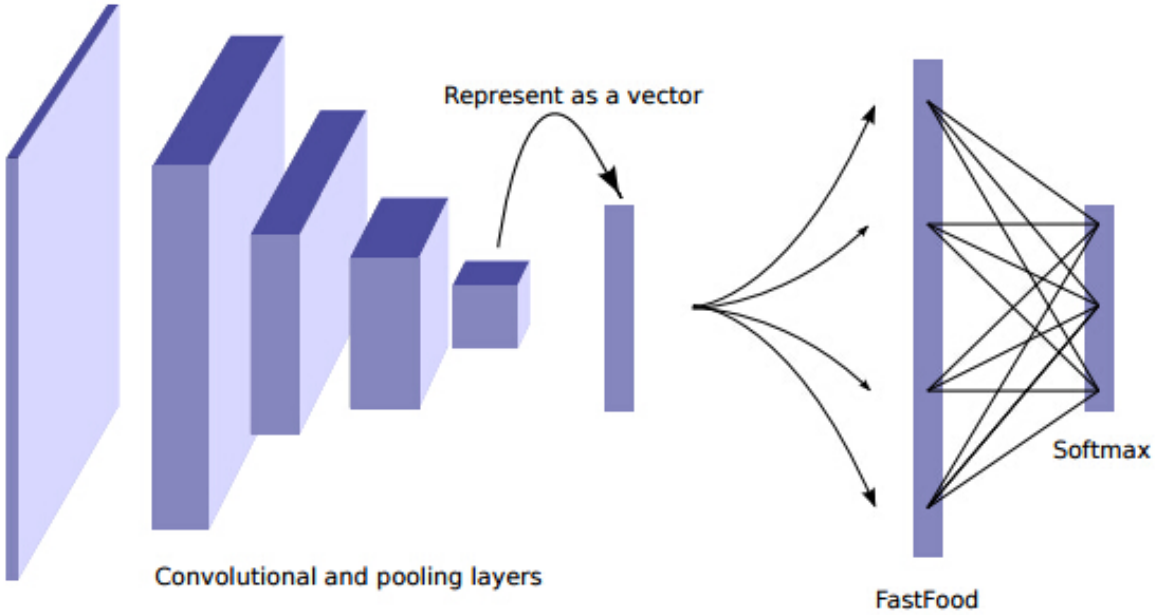
- Deep net is stored and processed on multiple cores (multi-thread) or machines (message passing)
- Performance benefit depends on connectivity structure vs. computational demand

Data Parallelism



- Left: Asynchronous SGD
- Right: Distributed L-BFGS

Parameters Reduction

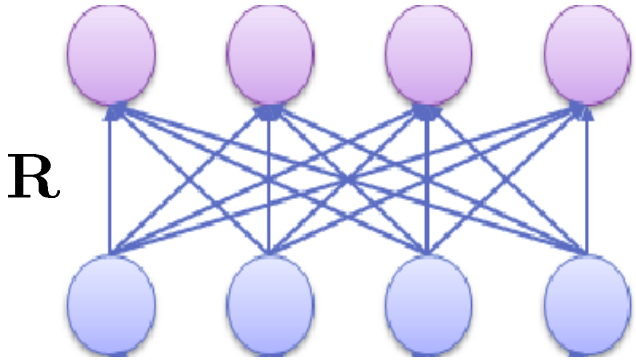


$$\mathbf{h}_{l+1} = \mathbf{W}\mathbf{h}_l \qquad \mathbf{h}_{l+1} = (\mathbf{SHG\Pi HB})\,\mathbf{h}_l = \widehat{\mathbf{W}}\mathbf{h}_l$$

- Time complexity O(nlogd), space complexity O(n)

Parameters Reduction

$h(\mathbf{x})$



$$h(\mathbf{x}) = \phi(\mathbf{R}\mathbf{x})$$

$$\mathbf{r} = (r_0, r_1, ..., r_{d-1})$$

$$\mathbf{R} = \mathrm{circ}(\mathbf{r}) := \begin{bmatrix} r_0 & r_{-1} & \cdots & r_{-(d-2)} & r_{-(d-1)} \\ r_1 & r_0 & r_{-1} & & r_{-(d-2)} \\ \vdots & r_1 & r_0 & \ddots & \vdots \\ r_{d-2} & & \ddots & \ddots & r_{-1} \\ r_{d-1} & r_{d-2} & \cdots & r_1 & r_0 \end{bmatrix}.$$

- Time complexity O(dlogd), space complexity O(d)