Introduction to Deep Learning

胡晓林
Dept. of Computer Science and Technology
Tsinghua University

April 23, 2015

Deep learning in the industry

Driverless car

Face identification

Speech recognition

Web search

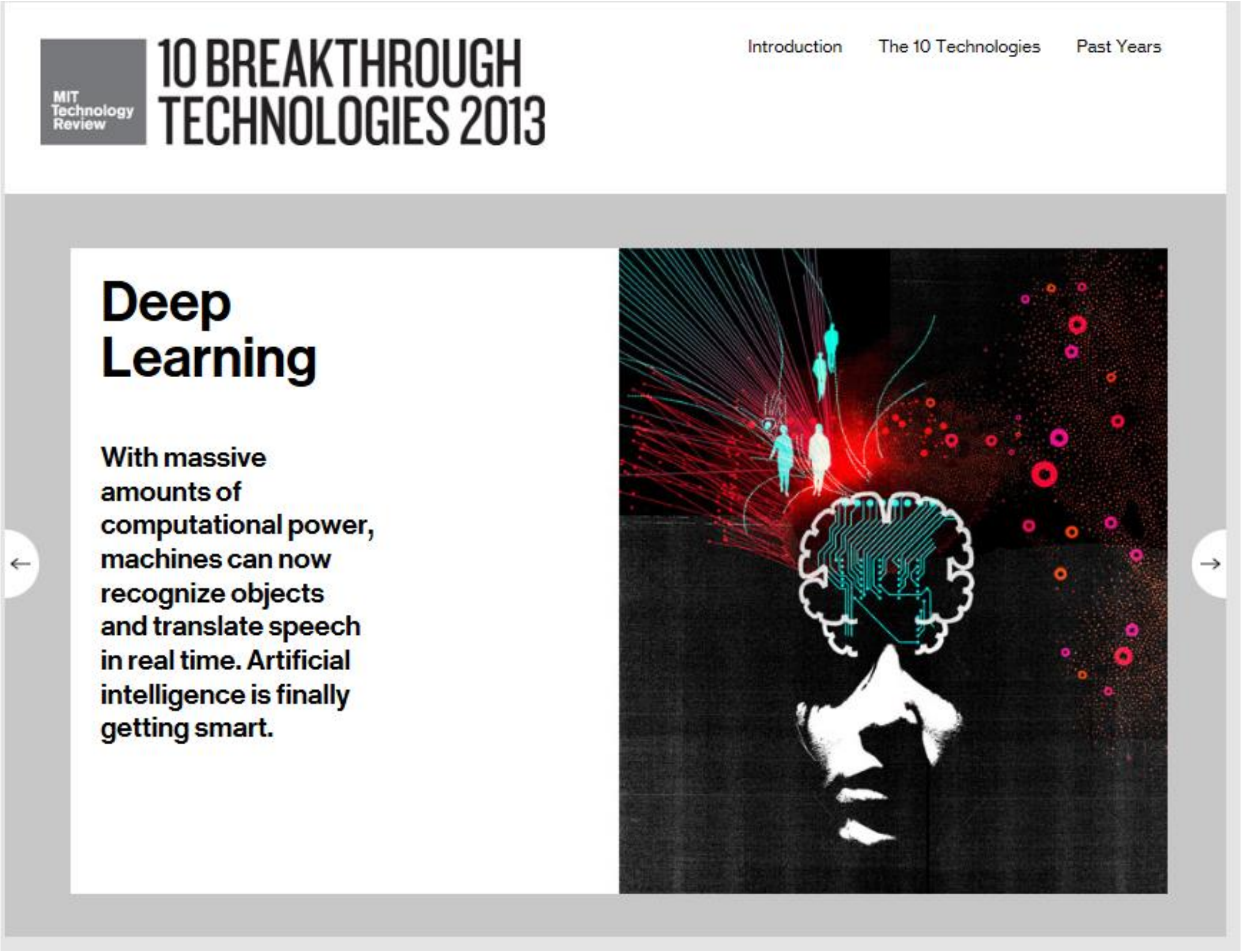
...

Google
f
阿里巴巴 Alibaba.com
HUAWEI
intel

...

Microsoft
Bai du 百度
Tencent 腾讯
AMD

MIT 10 Breakthrough Tech 2013



http://www.technologyreview.com/featuredstory/513696/deep-learning/

Outline

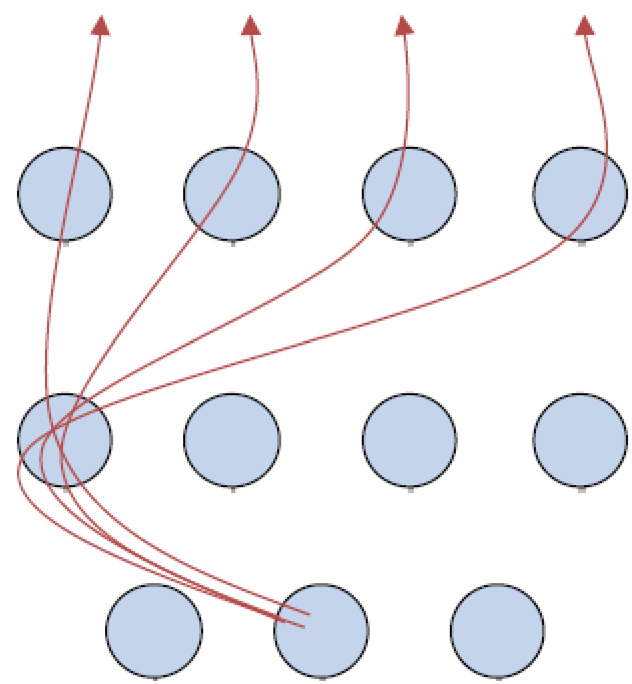
- Why go deep
- Multi-layer perceptron (review)
- Restricted Boltzmann machine
- Deep belief network
- Deep auto-encoder
- Convolutional neural network

Why go deep?

- Data are often high-dimensional
- There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model
- Insufficient depth can require more computational elements than architectures whose depth matches the task
- Deep nets provide simpler but more descriptive model of many problems.

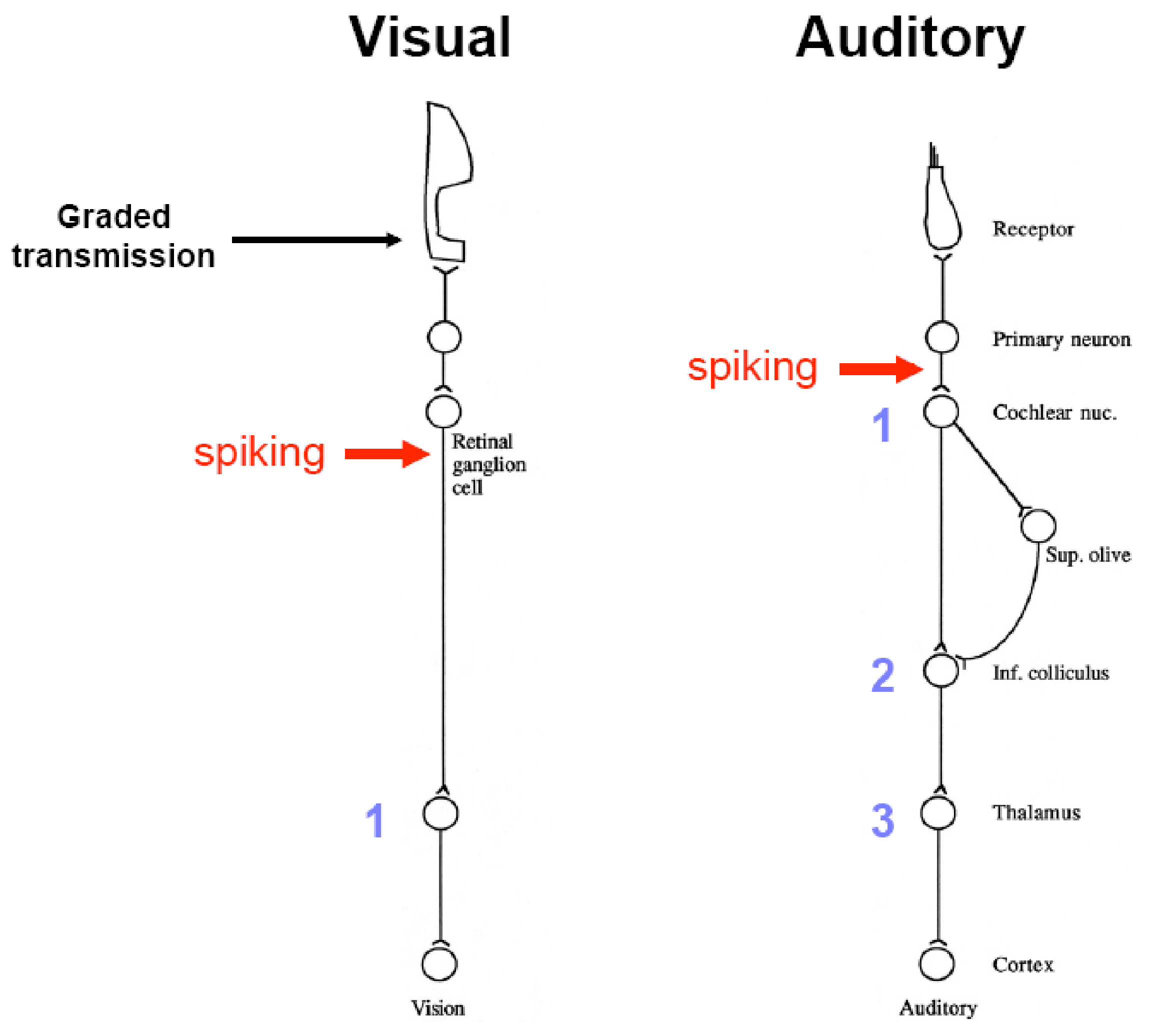
Representation ability

- In a deep model, the no. of paths from an input node to an output node increases exponentially
  - On each path there are a number of nonlinear operations
- The representation ability (nonlinear mapping from input to output) increases dramatically
- It is more powerful than a shallow model with the same number of nodes and nonlinear operations



图片摘自胡晓林，朱军，中国计算机协会通讯2013,9(7)

Sensory information processing in human brain

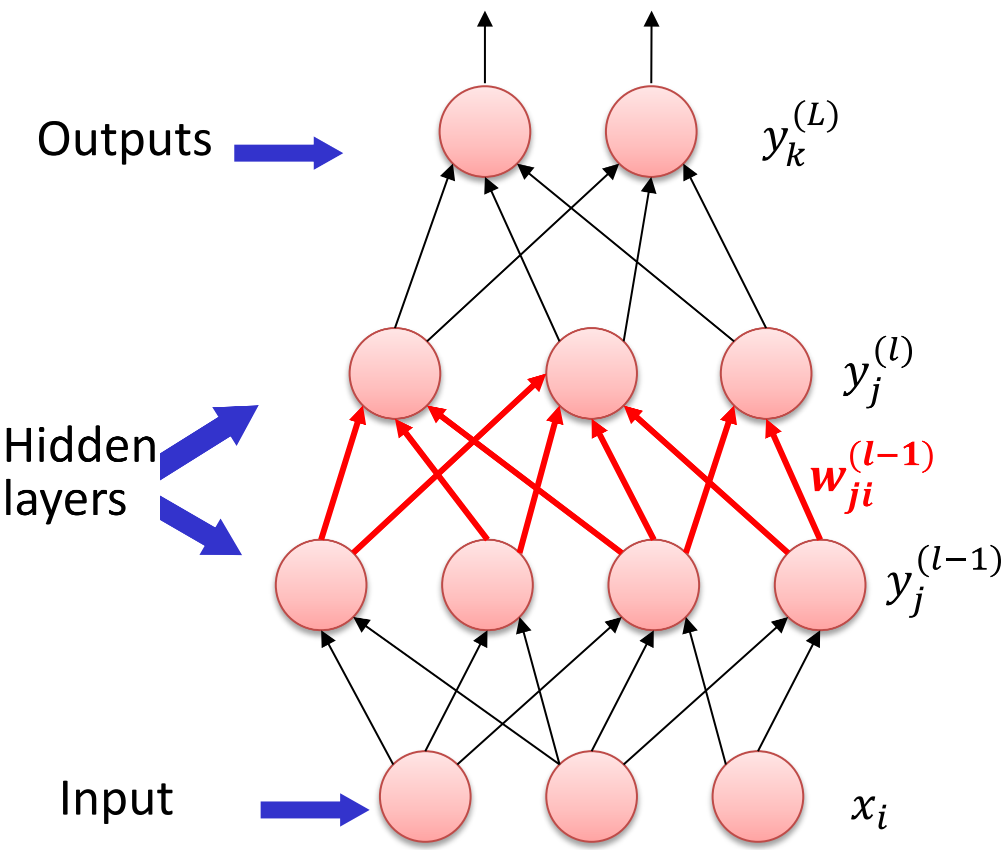


Courtesy of Xiaoqin Wang

Outline

- Why go deep
- Multi-layer perceptron (review)
- Restricted Boltzmann machine
- Deep belief network
- Deep auto-encoder
- Convolutional neural network

Multi-layer Perceptron (MLP)



For $l = 1, \ldots, L$ calculate the input to neuron j in the l-th layer

$$u_j^{(l)} = \sum_i w_{ji}^{(l-1)} y_i^{(l-1)} + b_j^{(l-1)}$$

and its output

$$y_j^{(l)} = f(u_j^{(l)}),$$

where $f(\cdot)$ is activation function

- Note $y^{(0)} = x$
- There are desired outputs t_k for each input sample in the form $(0, 0, \ldots, 1, 0, 0)^T$

Activation functions

- Logistic function

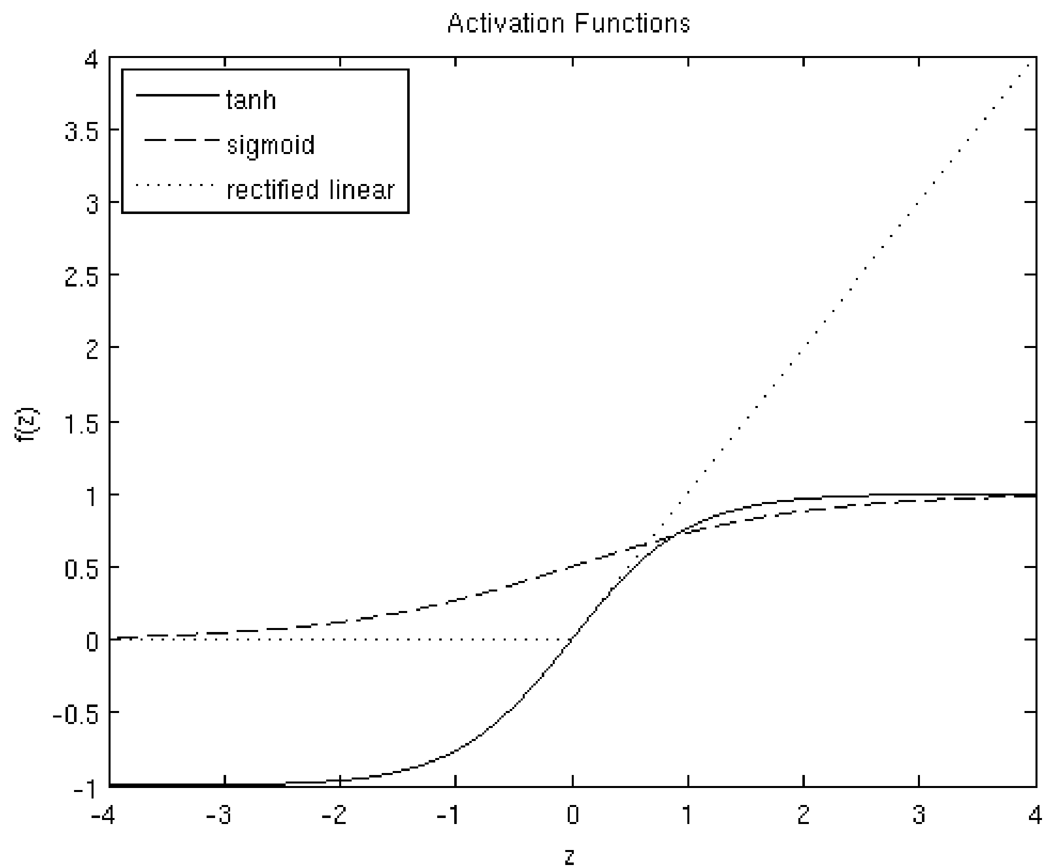
$$f(z) = \frac{1}{1 + \exp(-z)}$$

- Hyperbolic tangent, or tanh, function

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- Rectified linear activation function

$$f(z) = \max(0, x)$$



Error functions for BP

- Error function $$E = \sum_{n=1}^{N} E^{(n)}$$

  where $E^{(n)}$ is the error function for each input sample n

  – Least square error

sigmoid

$$E^{(n)} = \frac{1}{2}\sum_{k=1}^{K}(t_k - y_k^{(L)})^2,\ y_k^{(L)} = \frac{1}{1+\exp(-w_k^{(L-1)\top}y^{(L-1)}) - b_k^{(L-1)})}$$

  – Cross-entropy error

softmax

$$E^{(n)} = -\sum_{k=1}^{K} t_k \ln y_k^{(L)},\quad y_k^{(L)} = \frac{\exp(w_k^{(L-1)\top}y^{(L-1)} + b_k^{(L-1)})}{\sum_{j=1}^{K}\exp(w_j^{(L-1)\top}y^{(L-1)} + b_j^{(L-1)})}$$

- Weight adjustment

Learning rate

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \alpha\frac{\partial E}{\partial w_{ji}^{(l)}} \qquad b_j^{(l)} = b_j^{(l)} - \alpha\frac{\partial E}{\partial b_j^{(l)}}$$

Historically…

- Including more layers was not proved to be useful, sometimes even harmful
- A two-layer MLP was often used in practice

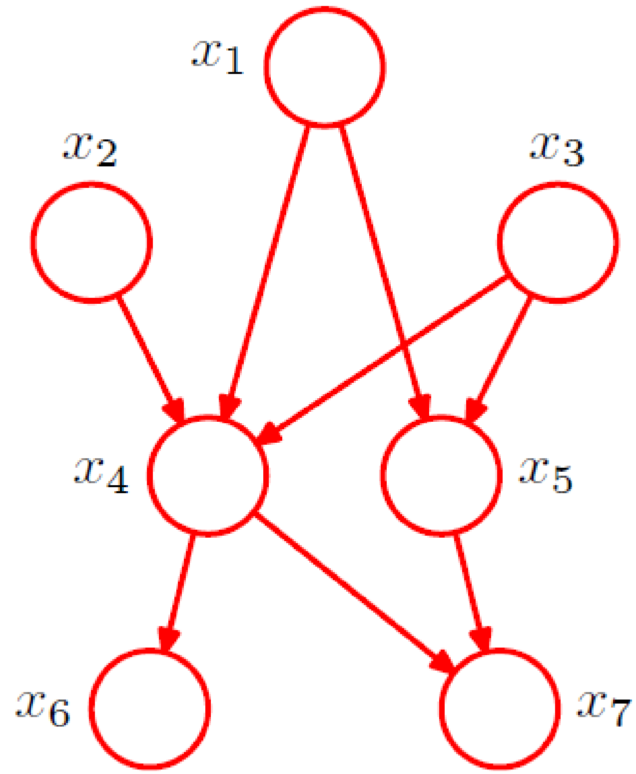
A two-layer MLP can approximate any function with arbitrary precision

Outline

- Why go deep
- Multi-layer perceptron (review)
- Restricted Boltzmann machine
- Deep belief network
- Deep auto-encoder
- Convolutional neural network

Probabilistic graphic model

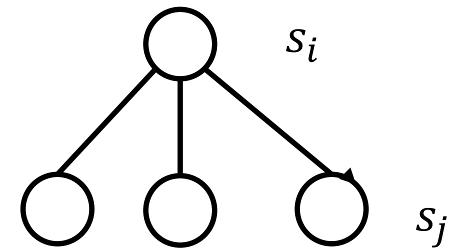
- Use a graph $G(V, E)$ to represent the joint distribution of a set of variables
- Directed graph (right)
  - Bayesian networks
- Undirected graph
  - Markov random fields



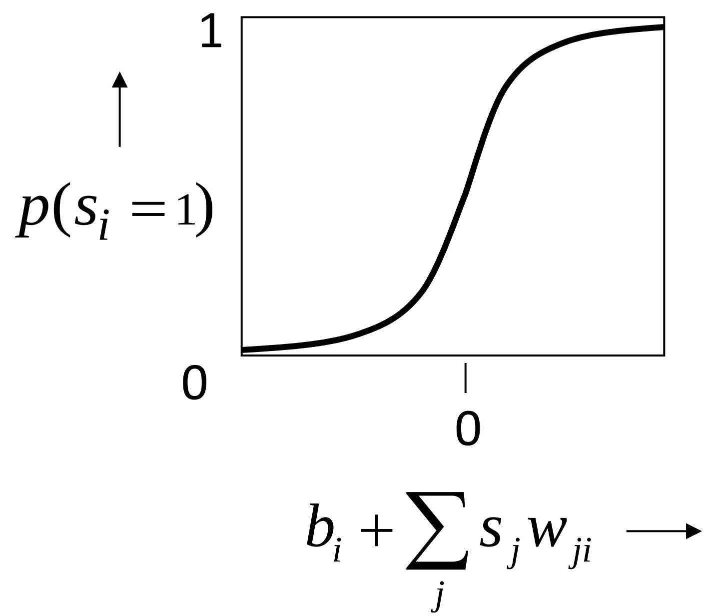
$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

Stochastic binary units

- Each unit has a state of 0 or 1
- The probability of turning on is determined by

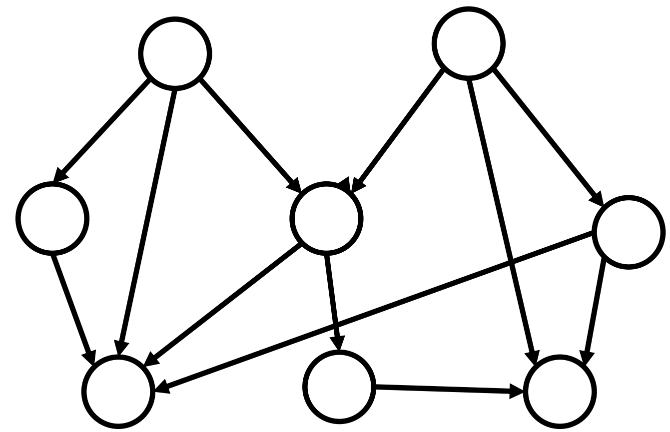


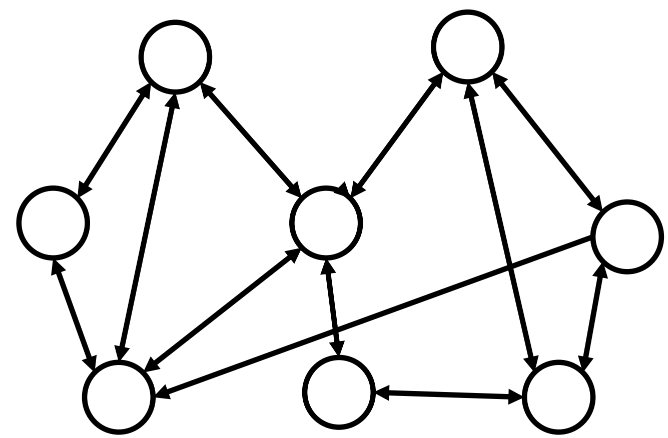
$$p(s_i = 1) = \frac{1}{1 + \exp(-b_i - \sum_j s_j w_{ji})}$$



Generative models

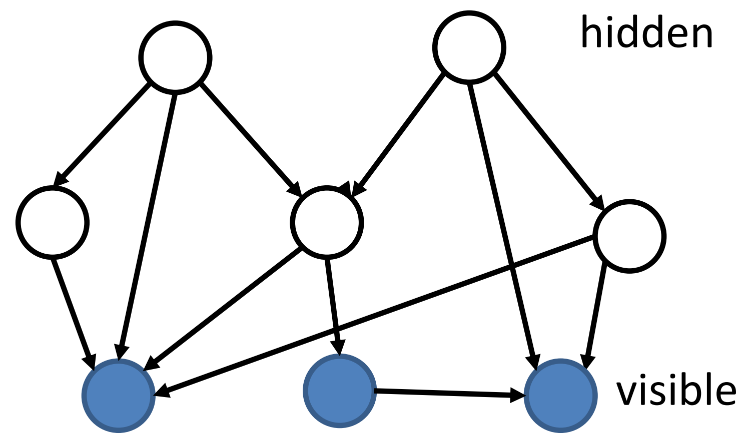
- Directed acyclic graph with stochastic binary units is termed Sigmoid Belief Net
  – Radford Neal 1992
- Undirected graph with stochastic binary units is termed Boltzmann Machine
  – Hinton & Sejnowski, 1983

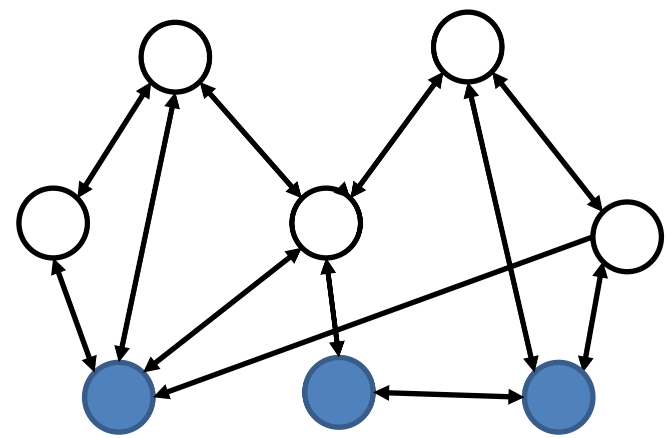




Generative models

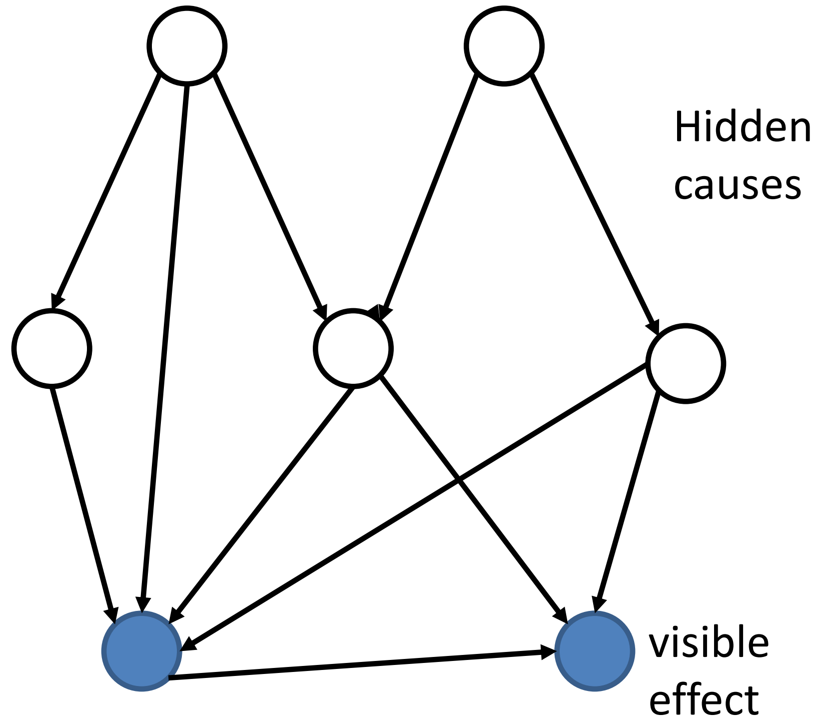
- Learning: Adjust the interactions between variables to make the network more likely to generate the observed data
- Inference: Infer the states of the unobserved variables
- Generate: Generate the observed data





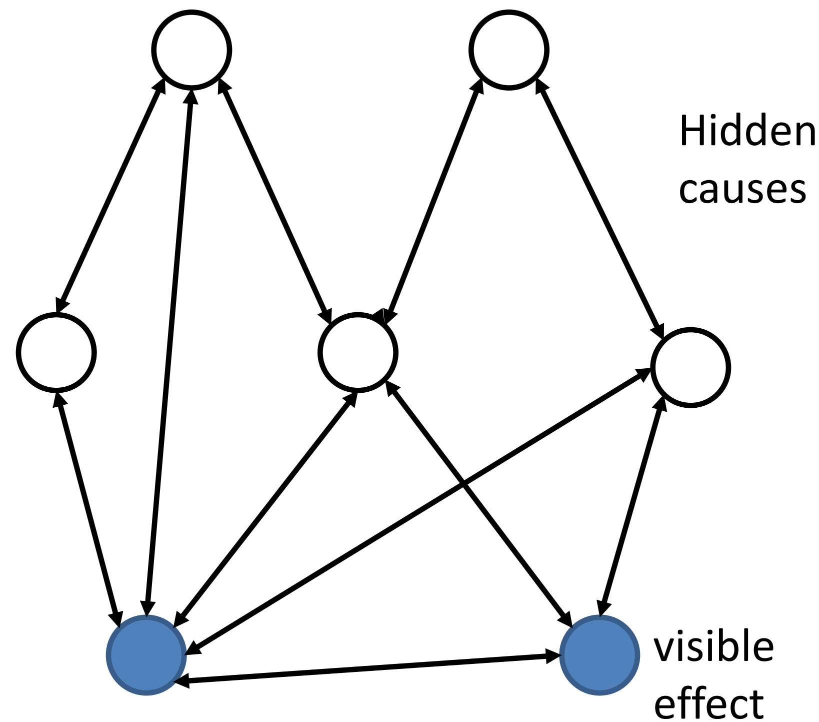
Learning deep belief nets

- Easy to generate an unbiased example at the leaf nodes
- Hard to infer the posterior distribution over all possible configurations of hidden causes
  - Hard to even get a sample from the posterior



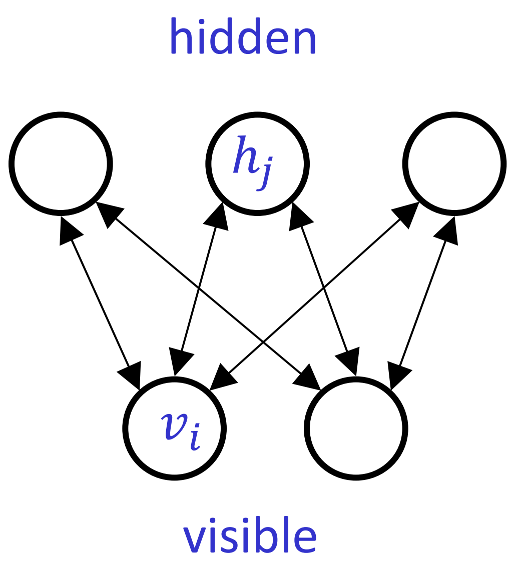
Learning Boltzmann machine

- Hard to generate an unbiased example for the visible units
- Hard to infer the posterior distribution over all possible configurations of hidden causes
  – Hard to even get a sample from the posterior

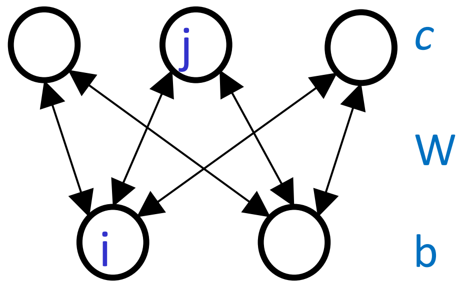


Restricted Boltzmann machines

- Restrict the connectivity to make learning easier.
  - Only one layer of hidden units.
  - No connections between hidden units.
  - Every unit can take only 1 or 0 stochastically
- In an RBM, the visible units are conditionally independent given the hidden states
  - We can quickly get an unbiased sample from the distribution $P(v|h)$ when given a hidden vector h
- In an RBM, the hidden units are conditionally independent given the visible states
  - We can quickly get an unbiased sample from the posterior distribution $P(h|v)$ when given a data v



Energy model



- Joint distribution

partition function

$$P[v,h] = \frac{\exp(-E(v,h))}{Z} \quad \text{where} \quad Z = \sum_{v,h} \exp(-E(v,h)).$$

- The energy function

$$E(v,h) = -v \cdot W \cdot h - b \cdot v - c \cdot h$$

- The probability distribution of data

$$P[v;\mathcal{G}] = \sum_h P[v,h;\mathcal{G}] = \frac{1}{Z} \sum_h \exp(-E(v,h)).$$

where $\mathcal{G} \triangleq (W,b,c)$

Maximum data log likelihood

- The primary goal

$$P[v;\mathcal{G}] = \frac{1}{Z}\sum_h \exp(-E(v,h)).$$

$$\mathcal{G}^* = \arg\max \langle \ln P[v;\mathcal{G}] \rangle$$

- The gradient

$$E(v,h) = -v \cdot W \cdot h - b \cdot v - c \cdot h$$

$$\frac{\partial \ln P[v;\mathcal{G}]}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \left(\ln \sum_h \exp(-E(v,h)) - \ln \sum_{v,h} \exp(-E(v,h)) \right)$$

$$= \sum_h \frac{\exp(-E)}{\sum_h \exp(-E)} v_i h_j - \sum_{v,h} \frac{\exp(-E)}{\sum_{v,h} \exp(-E)} v_i h_j$$

$$= \sum_h P[h|v;\mathcal{G}] h_j v_i - \sum_{v,h} P[v,h;\mathcal{G}] h_j v_i.$$

$$\frac{\partial \ln P[v;\mathcal{G}]}{\partial b_i} = \sum_h P[h|v;\mathcal{G}] v_i - \sum_{v,h} P[v,h;\mathcal{G}] v_i$$

$$\frac{\partial \ln P[v;\mathcal{G}]}{\partial c_j} = \sum_h P[h|v;\mathcal{G}] h_j - \sum_{v,h} P[v,h;\mathcal{G}] h_j$$

Approximate avg with one sample

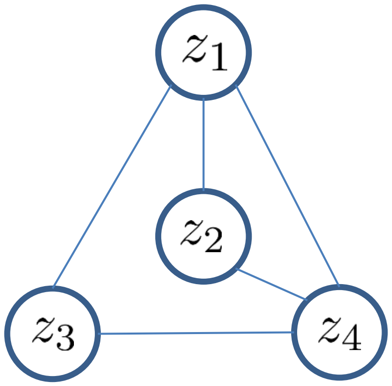
Learning rule

- Stochastic gradient ascent (*n* is the sample index)

$$W_{ij} = W_{ij} + \epsilon_W(h_j(v^n)v_i^n - h_j(t \to \infty)v_i(t \to \infty))$$
$$b_i = b_i + \epsilon_b(v_i^n - v_i(t \to \infty))$$
$$c_j = c_j + \epsilon_c(h_j(v^n) - h_j(t \to \infty))$$

- Wake phase: Gibbs sampling is used to calculate $h(v^n)$
- Sleep phase: Gibbs sampling is used to calculate $h(t \to \infty)$ and $v(t \to \infty)$

Gibbs sampling



- Draw a sample from

$$p(z) = p(z_1, \ldots, z_M)$$

1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.

     $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.

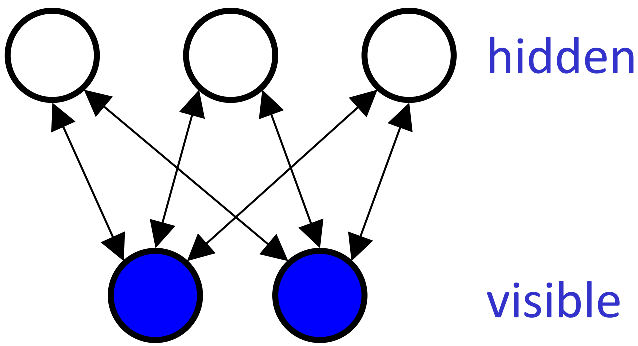
     $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.

Gibbs sampling

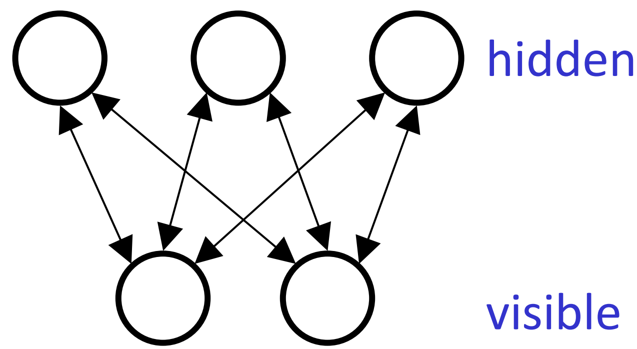
- It can be proved that this procedure draw a sample from the joint distribution $p(z)$
- Gibbs sampling is a special case of the Markov Chain Monte Carlo (MCMC) algorithm
- It applies to both directed and undirected probabilistic graphical models
- It also applies to models other than graphical models

Tasks

- Draw a sample from the conditional distribution $P(h|v)$



  – Draw a sample from the joint distribution $P(h, v)$

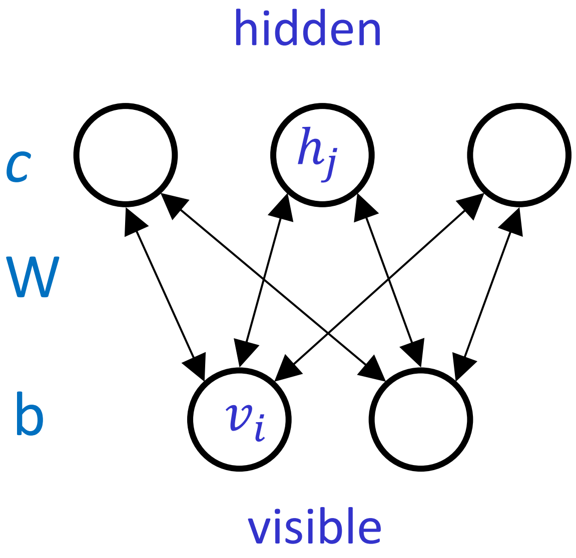


Both by (block) Gibbs sampling!

What we only need are $P(h_j = 1|v)$ and $P(v_i = 1|h)$

Conditional distributions

- With the energy function defined before, it can be shown that (details are skipped here)

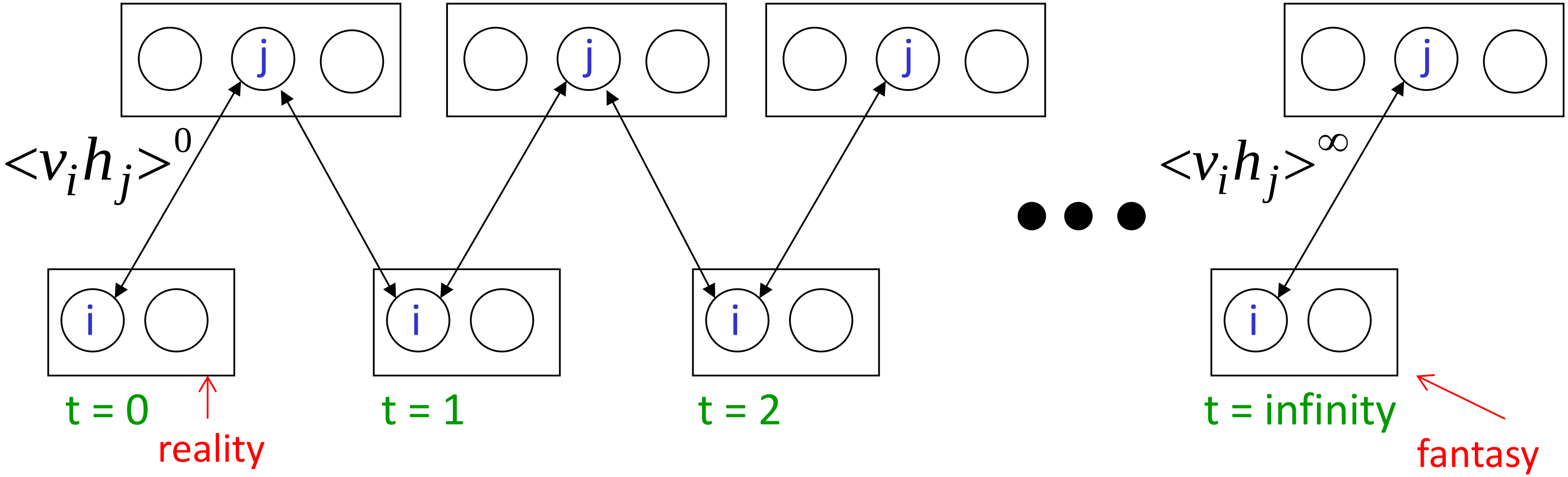


$$P(v_i = 1|h) = sigmoid(\sum_j w_{ji} h_j + b_i)$$

$$P(h_j = 1|v) = sigmoid(\sum_i w_{ij} v_i + c_j)$$

Illustration of learning

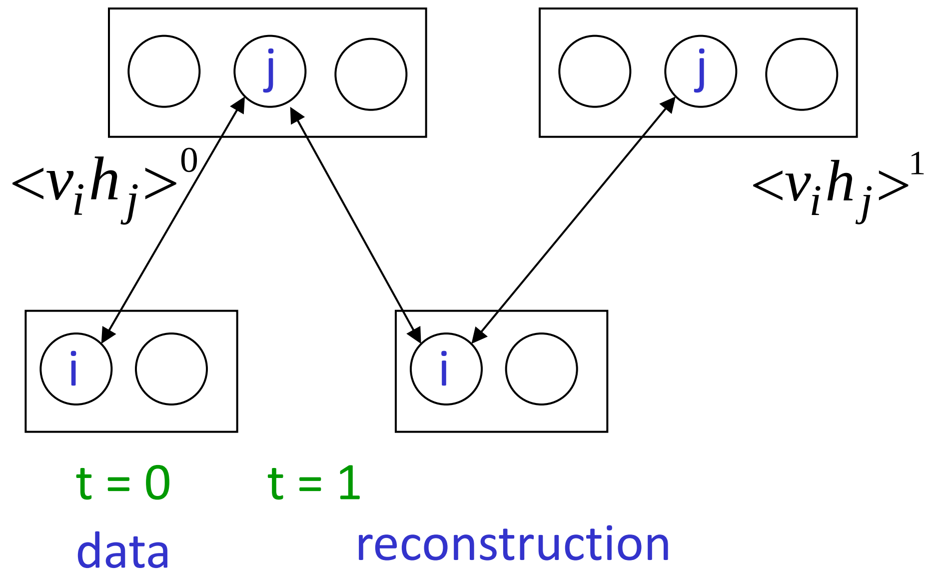
$$W_{ij} = W_{ij} + \epsilon_W(\langle h_j v_i \rangle^0 - \langle h_j v_i \rangle^\infty)$$
$$b_i = b_i + \epsilon_b(\langle v_i \rangle - \langle v_i \rangle^\infty)$$
$$c_j = c_j + \epsilon_c(\langle h_j \rangle - \langle h_j \rangle^\infty)$$



Alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

Contrastive divergence learning

- CD-1
  - Start with a training vector on the visible units.
  - Update all the hidden units in parallel
  - Update the all the visible units in parallel to get a "reconstruction"
  - Update the hidden units again



$$\Delta w_{ij} = \varepsilon \, (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^1)$$

- CD-*n*
  - Keep running for *n* steps

$$\Delta w_{ij} = \varepsilon \, (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^n)$$

Outline

- Why go deep
- Multi-layer perceptron (review)
- Restricted Boltzmann machine
- Deep belief network
- Deep auto-encoder
- Convolutional neural network

What’s wrong with BP network?
-Hinton’s opinion in 2006

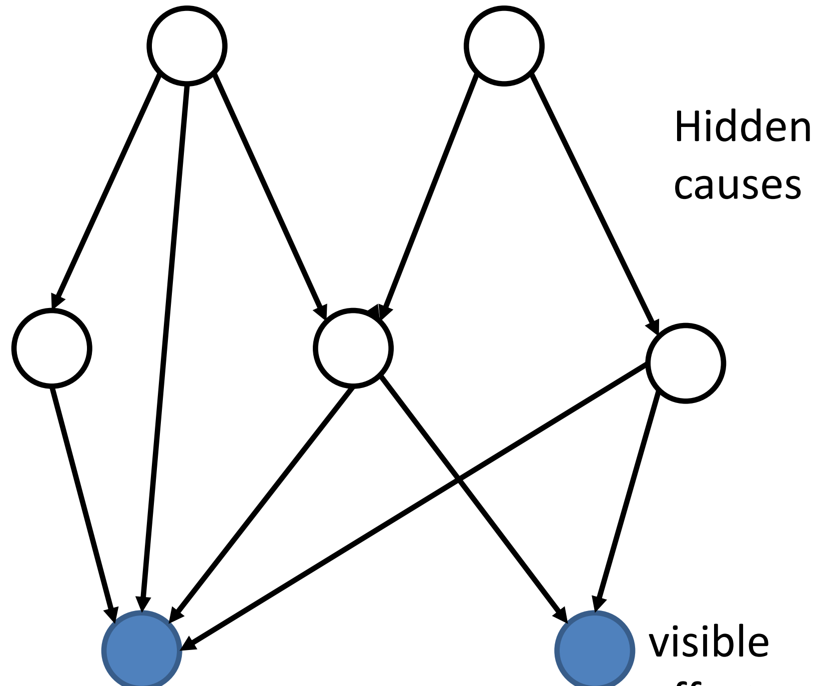
- It requires labeled training data
  - Almost all data is unlabeled
- The learning time does not scale well
  - It is very slow in networks with multiple hidden layers
- It can get stuck in poor local optima
  - These are often quite good, but for deep nets they are far from optimal

Overcoming the limitations of BP

- Different purpose: modeling the structure of the sensory input
  - Adjust the weights to maximize the probability that a generative model would have produced the sensory input
  - Learn p(image) not p(label | image)
- What kind of generative model shall we learn?

Belief networks

- Easy to generate an unbiased example at the leaf nodes
- Hard to infer the posterior distribution over all possible configurations of hidden causes
- So how can we learn deep belief nets that have millions of parameters?



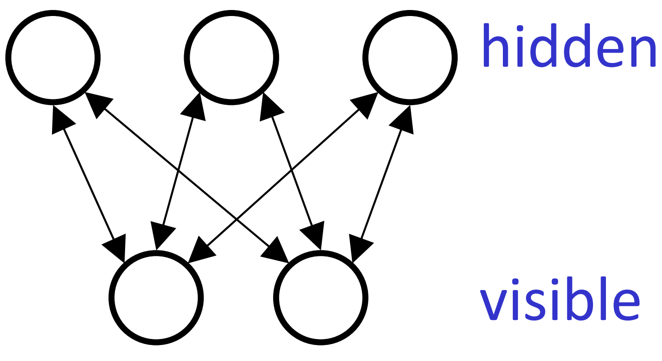
$$p(s_i = 1) = \frac{1}{1 + \exp(-b_i - \sum_j s_j w_{ji})}$$

Some methods for learning deep belief nets

- Monte Carlo methods can be used to sample from the posterior.
  - But its painfully slow for large, deep models.
- In the 1990's people developed variational methods for learning deep belief nets
  - These only get approximate samples from the posterior.
  - Nevertheless, the learning is still guaranteed to improve a variational bound on the log probability of generating the observed data.

The breakthrough

- To learn deep nets efficiently, we need to learn one layer of features at a time.
- We need a way of learning one layer at a time that takes into account the fact that we will be learning more hidden layers later.
  - We solve this problem by using RBM.



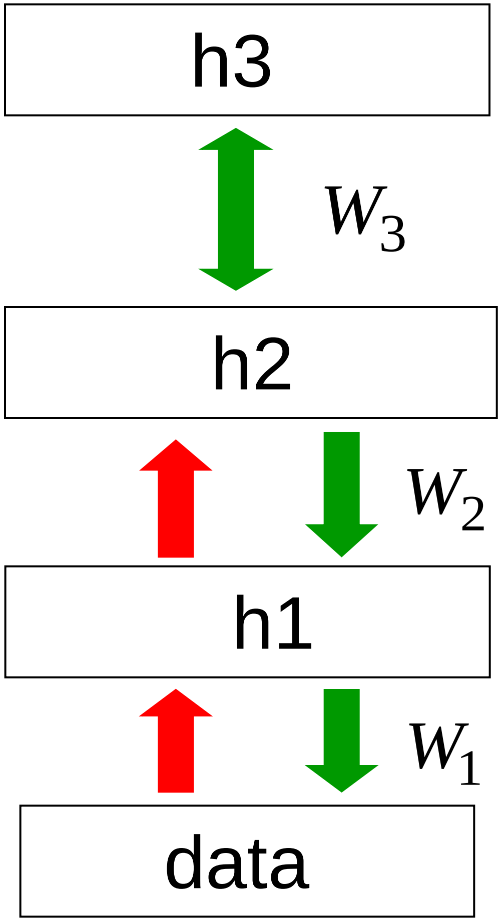
Training a deep network

- Stacking RBMs to form deep architecture
  - First train an RBM that receives input directly from the pixels
  - Then treat the activations of the hidden layer as if they were pixels and train a second hidden layer
  - Repeat the process
- Each time we add another layer of features we improve a variational lower bound on the log probability of the training data
  - Proof is a little bit complicated

The generative model after learning a 3-layer model

- To generate data:
  1. Get an equilibrium sample from the top-level RBM by performing alternating Gibbs sampling for a long time.
  2. Perform a top-down pass to get states for all the other layers.

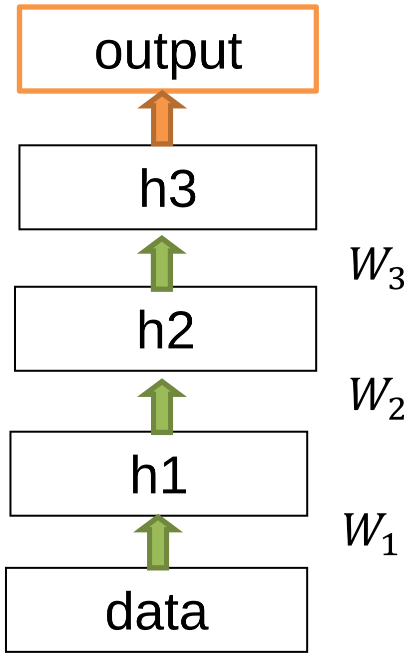
So the bottom-up connections are not part of the generative model. They are just used for inference.



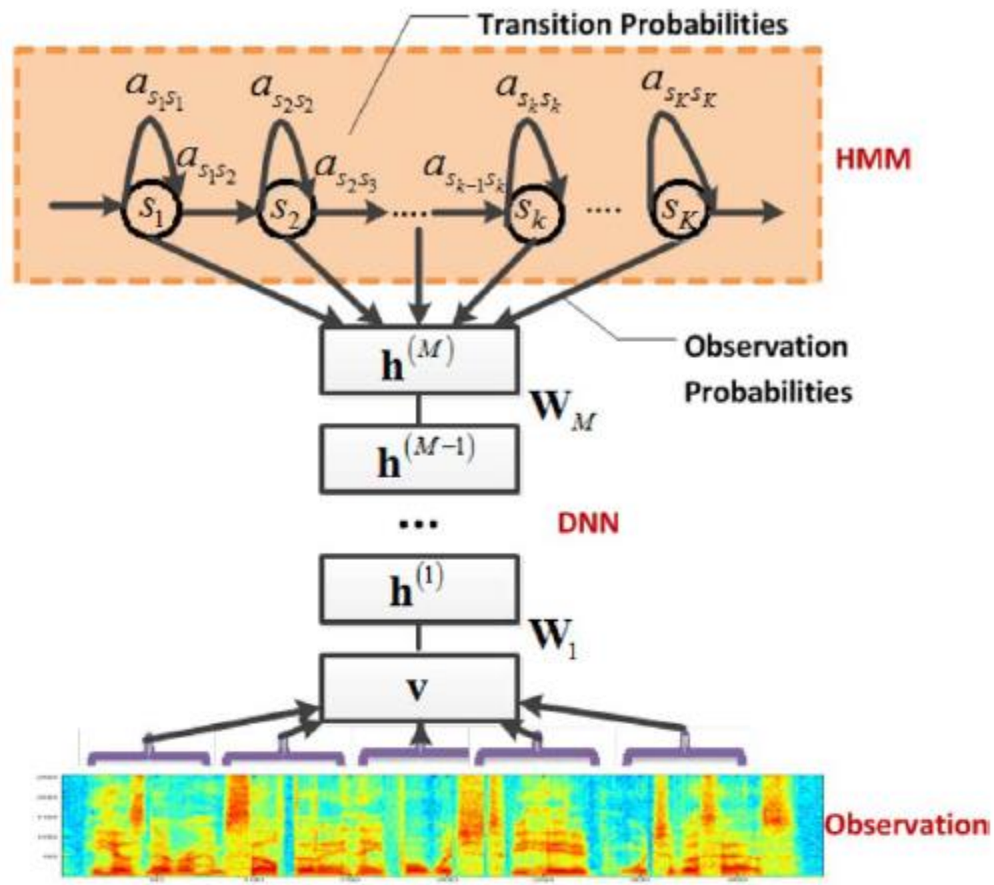
How to use the pre-trained DBN?

Method 1: Add a layer on top

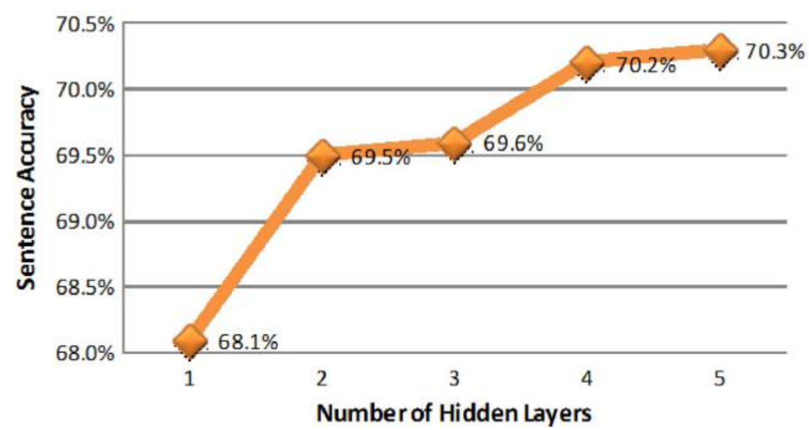
- Add a softmax layer on top, then perform BP training with the pretrained weights as initial weights
  - Dahl, Yu, Deng, Acero, IEEE TASLP, 2012
- Add an SVM on top
  - Lee, et al, ICML 2009



Speech Recognition



Compared with CD-GMM-HMMs， CD-DNN-HMMs improved 5.8% and 9.2% accuracy using the minimum phone error rate (MPE) and maximum-likelihood (ML) criteria



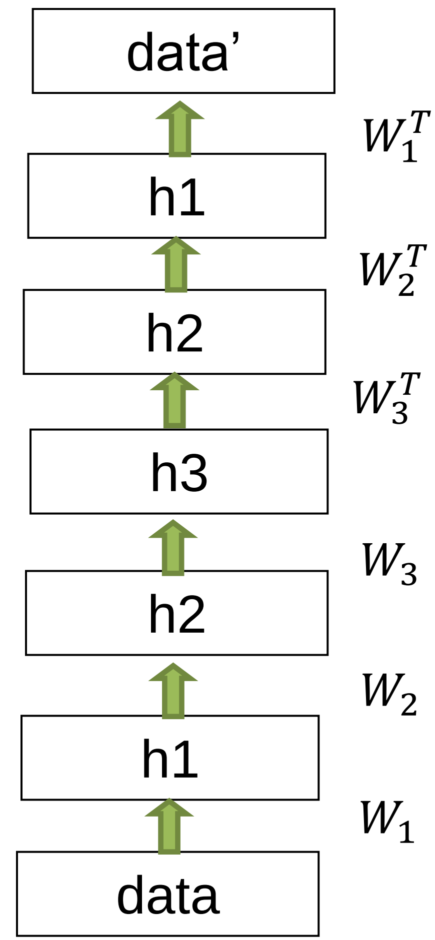
Dahl, Yu, Deng, Acero, IEEE TASLP, 2012

Speech Translation by Microsoft Research

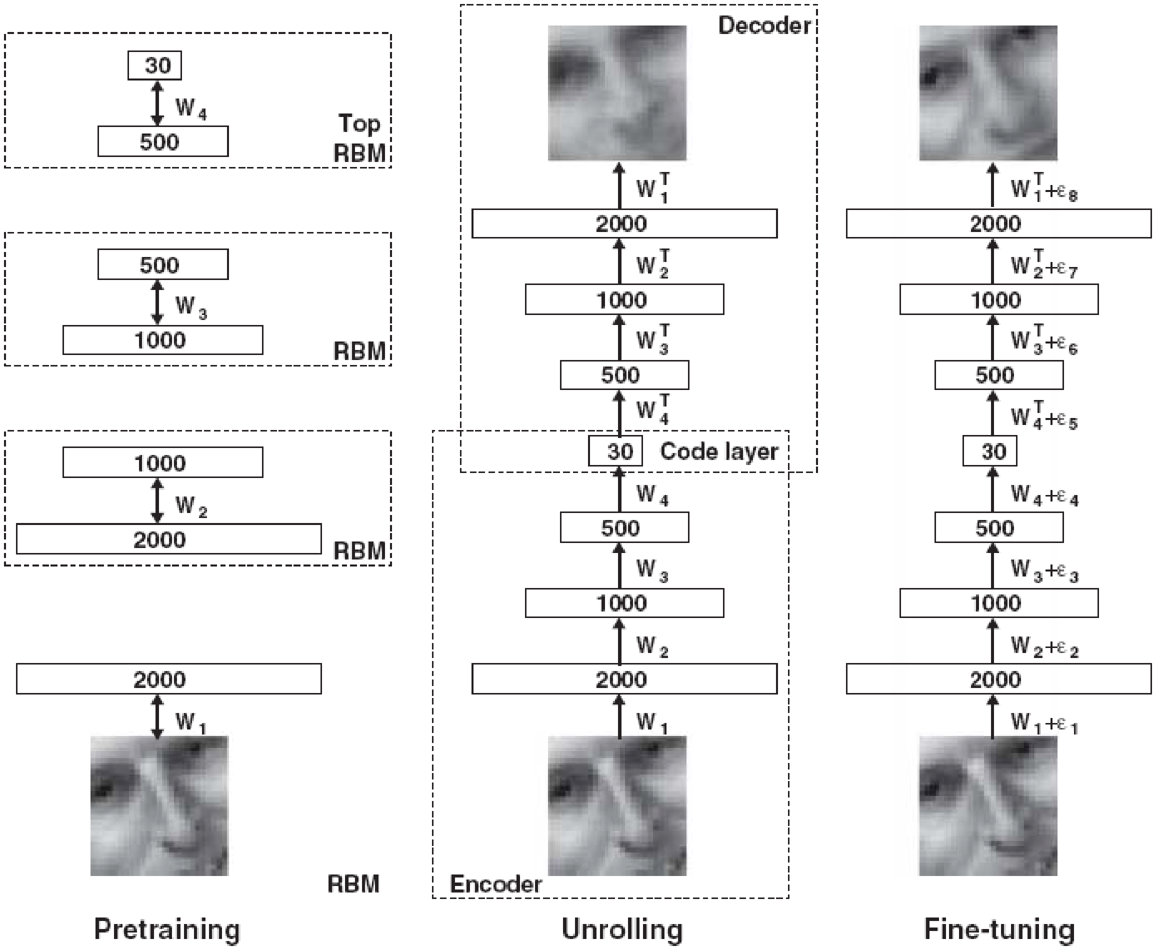
- See youku: http://v.youku.com/v_show/id_XNDc0MDY4ODI0.html

Method 2: Unroll the architecture and fine-tune with BP

- The target is the data itself
- If the number of units in layer h3 is small, then it performs data compression
  - Hinton, Salakhutdinov, Science, 2006

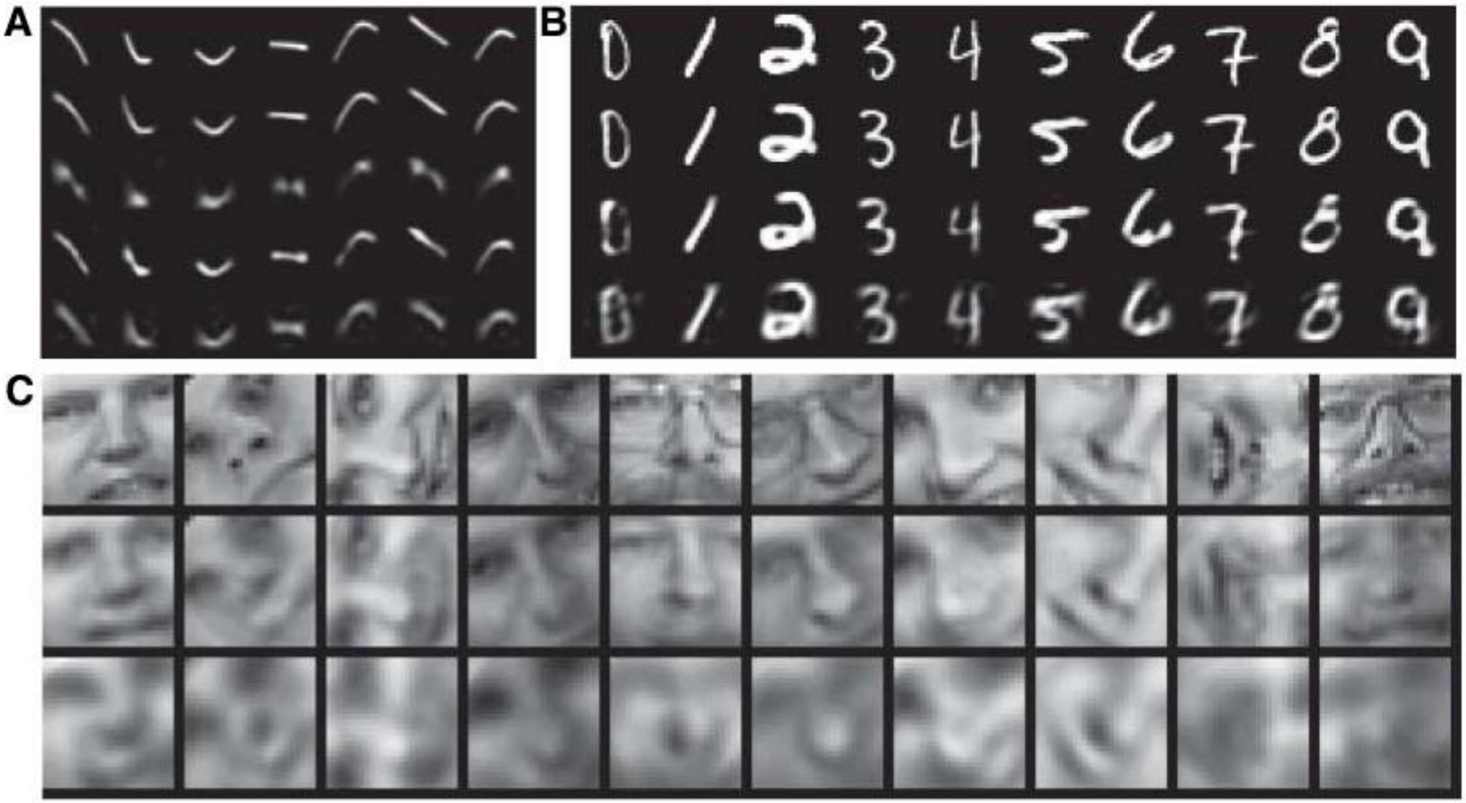


Data compression



Hinton, Salakhutdinov, Science, 2006

Reconstruction Results



real data

30-D deep auto

30-D logistic PCA

30-D PCA

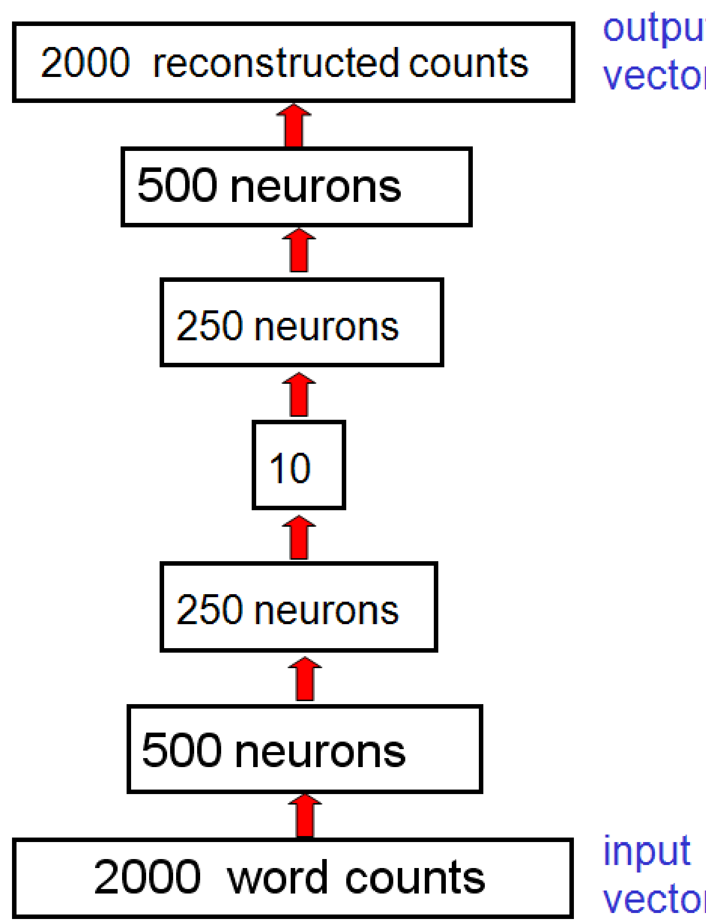
real data

30-D deep auto

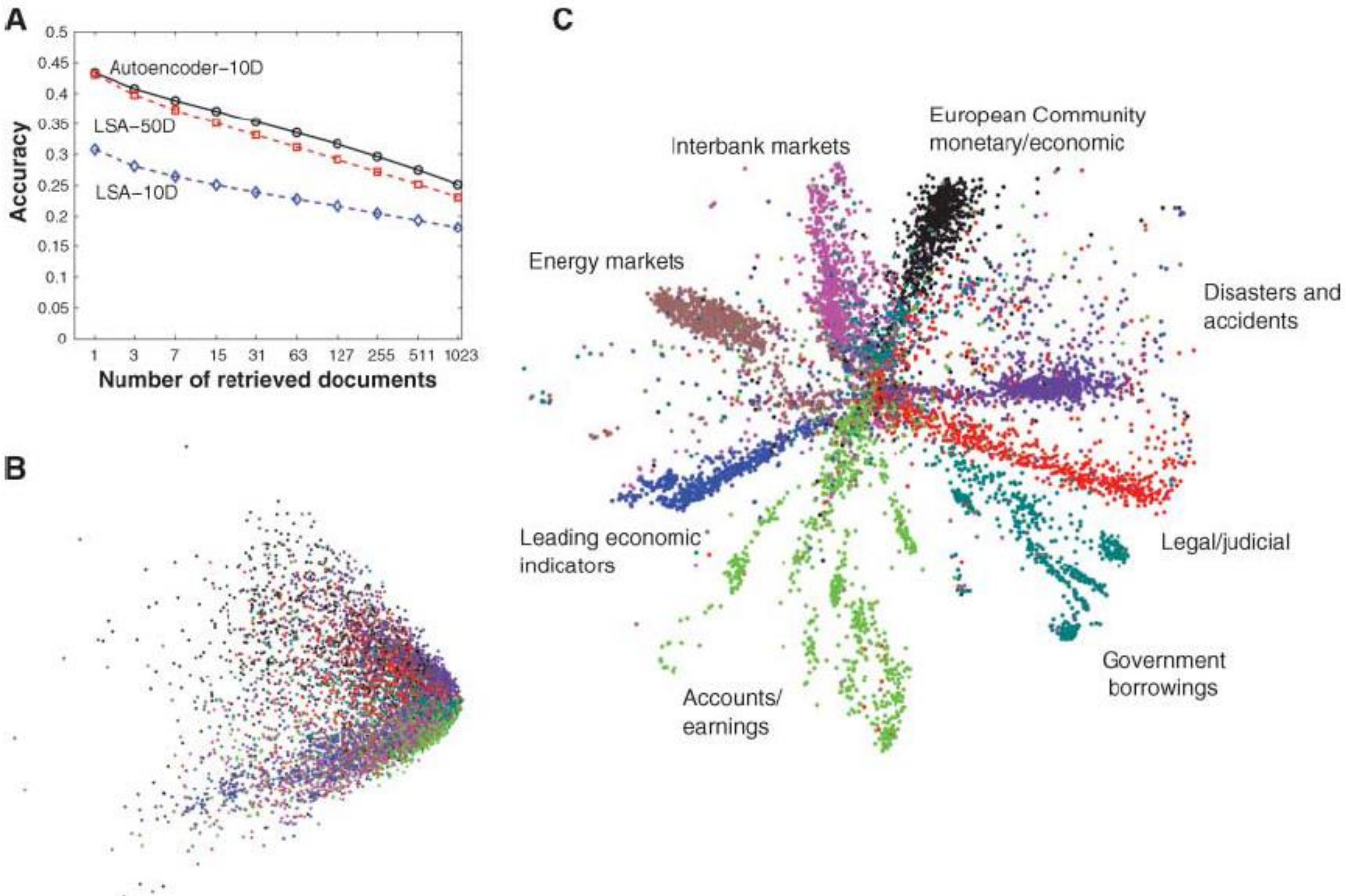
30-D PCA

Retrieving Documents

- Convert each document into a "bag of words".
  - This a 2000D vector
- Compress them to 10D vectors
- Compare documents based on these 10D vectors



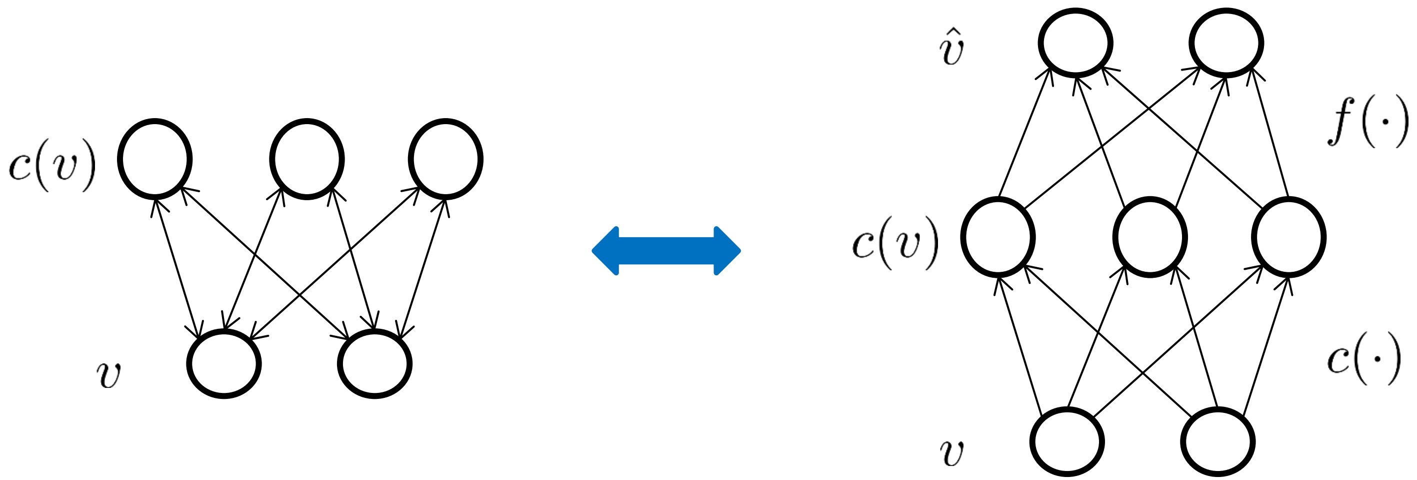
Results on 804,414 Newswire Stories



Outline

- Why go deep
- Multi-layer perceptron (review)
- Restricted Boltzmann machine
- Deep belief network
- Deep auto-encoder
- Convolutional neural network

Auto-encoder



- Encode the input *v* into some representation *c(v)* so that the input can be reconstructed from that representation
  - Encoding function *c(v)*
  - Decoding function *f(c(v))*

- Nonlinear function

$$c(v) = sigmoid(W_1 v + \theta)$$

$$f(c) = sigmoid(W_2 c + \eta)$$

It can be constrained $W_1 = W_2^T$ or not

- The functions can be used as probabilities for binary variables

Learning Goal

- Minimize the reconstruction error or the negative data log-likelihood

$$RE = -\langle \ln P(v|c(v)) \rangle$$

  - Gaussian probability (*v* is real)

$$P(v|c(v)) \propto \exp\left(\frac{-\|v-f(c(v))\|^2}{2\sigma^2}\right)$$

  then $RE = \langle \|v - f(c(v))\|^2 \rangle$

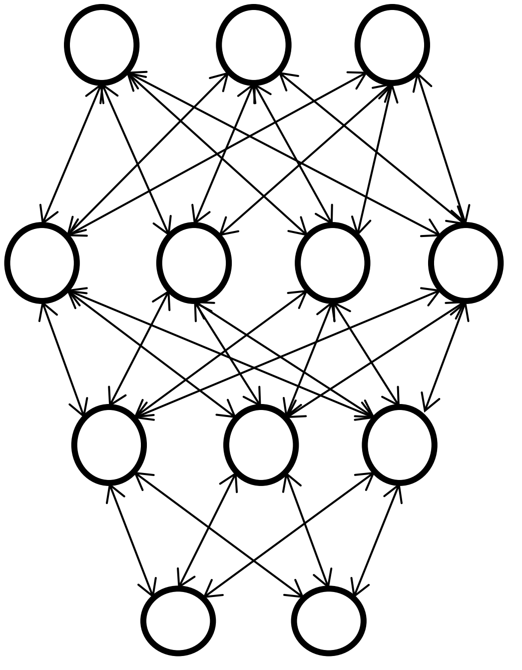
  - Binomial probability (*v* is binary)

$$P(v|c(v)) \propto \Pi_i f_i(c(v))^{v_i} (1 - f_i(c(v)))^{1-v_i}$$

  then $RE = -\langle \sum_i \left(v_i \ln f_i(c(v)) + (1 - v_i) \ln(1 - f_i(c(v)))\right) \rangle$

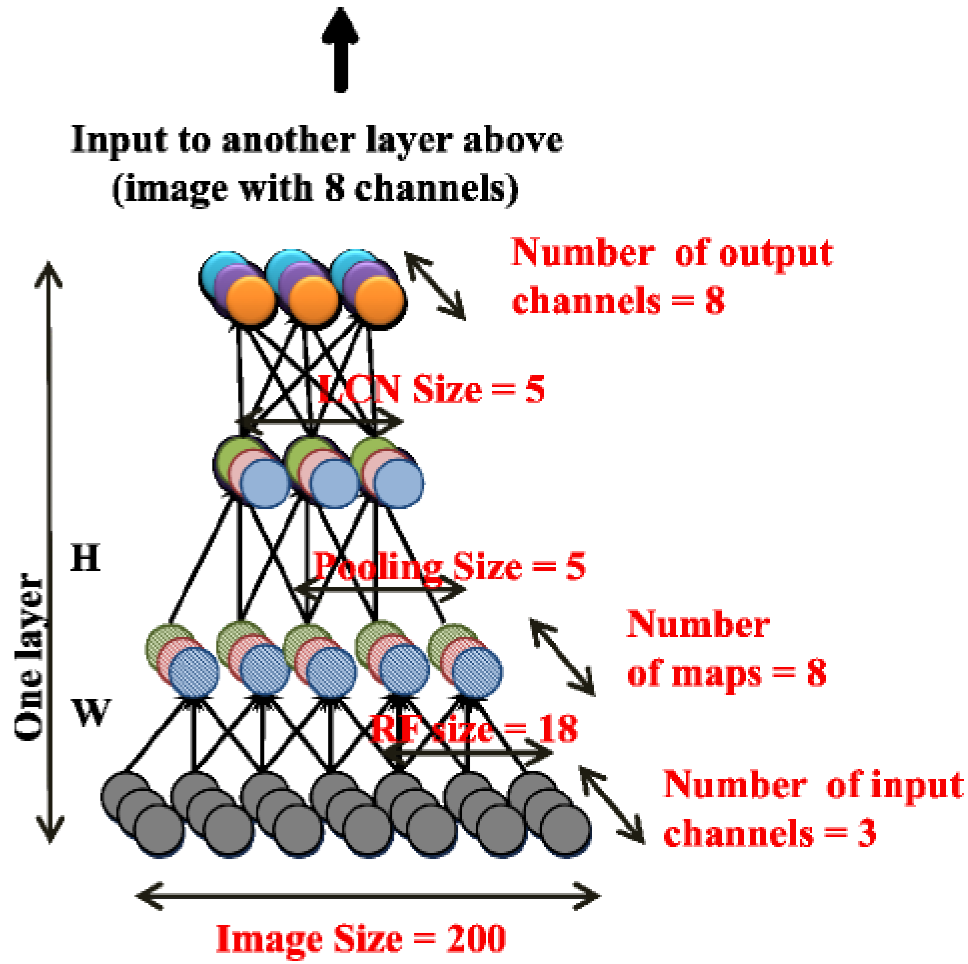
Deep Auto-encoder

- Stack auto-encoders on top of each other
- Train layers one by one
- Sparsity or other regularizations can be used

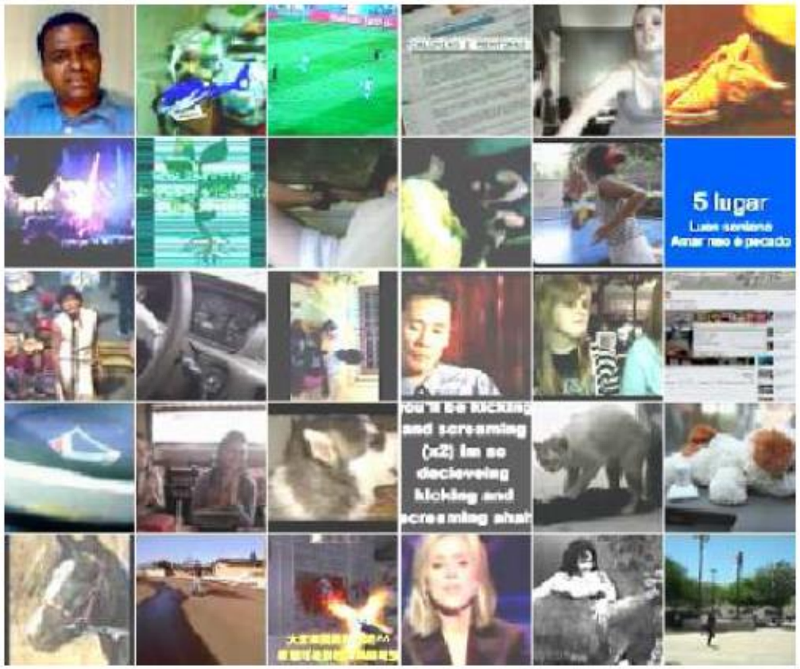


A interesting application

- Combined with
  - Local receptive field
  - L2 pooling
  - Local contrast normalization
- The overall network replicate this architecture 3 times
- Over 1 billion parameters
- Three days on a cluster with 1,000 machines (16,000 cores).



Le, et al., ICML 2012

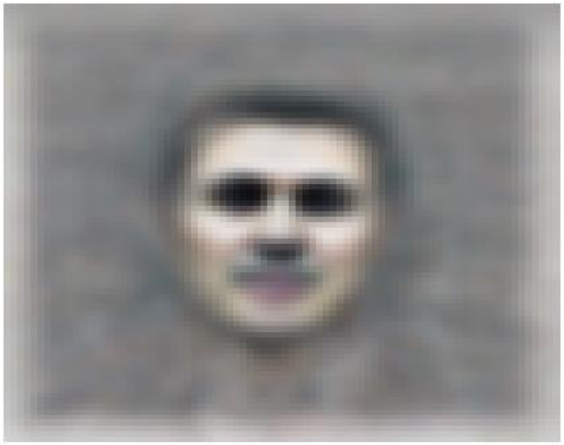


Trained on Youtube images

Tested on a mixture of Labeled Faces in The Wild and ImageNet

- “face neuron”

Images with strongest responses



Optimal stimulus

- “cat neuron”

Images with strongest responses

Optimal stimulus

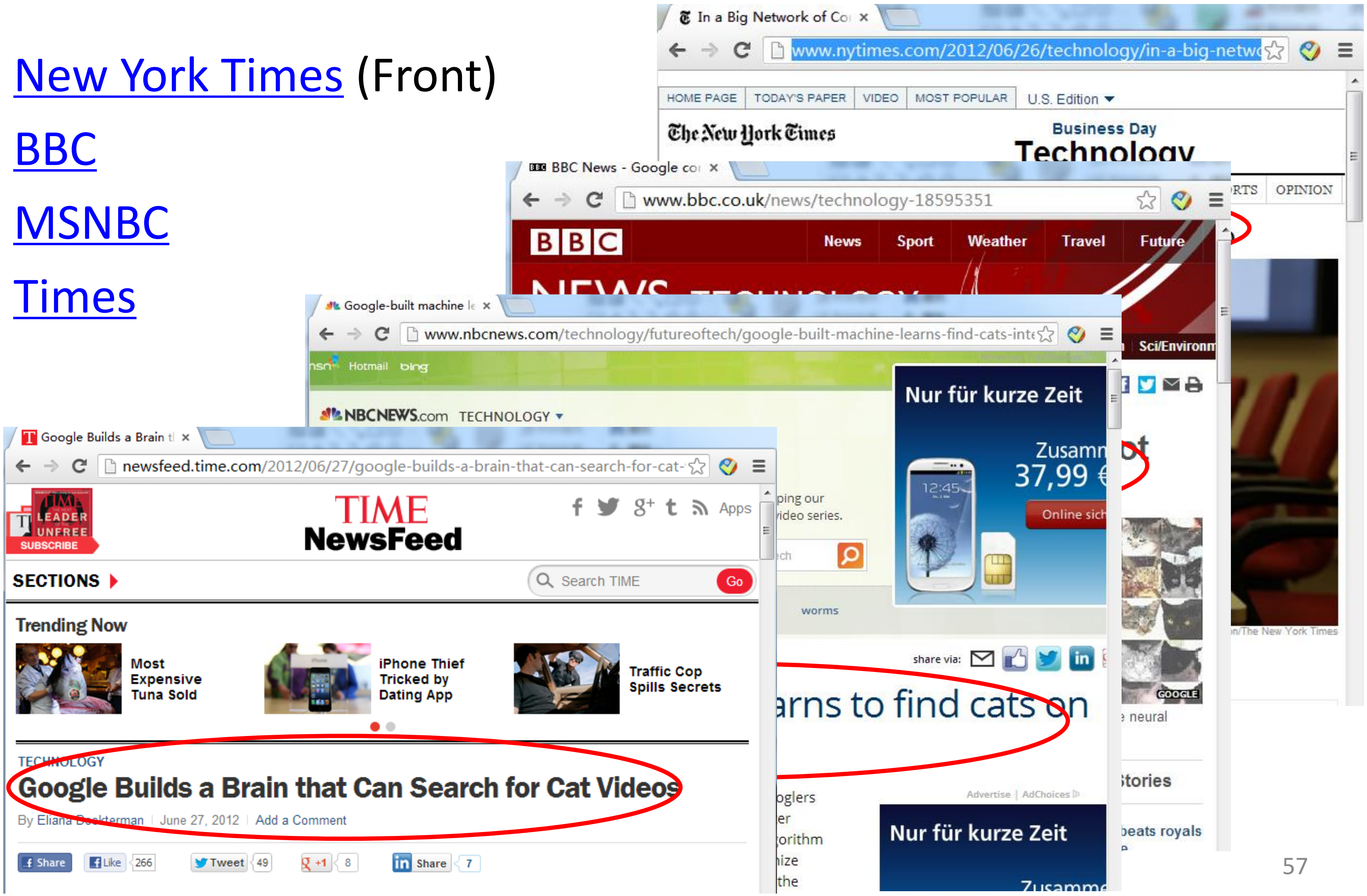
- “body neuron”

Images with strongest responses

Optimal stimulus

News in the Media

- [New York Times](#) (Front)
- [BBC](#)
- [MSNBC](#)
- [Times](#)



Summary so far

Principle: learn a representation first, then do task-relevant job

Unsupervised learning

- RBM
- DBN
- Auto-encoder

Supervised learning

- A classifier on top (w/o fine tuning)
- MLP (fine tuning with BP)

Is unsupervised learning really necessary?

Recall Hinton's opinion about BP network

It requires labeled training data

- Almost all data is unlabeled

What if in some applications there are enough labeled data?

The learning time does not scale well

- It is very slow in networks with multiple hidden layers

It can get stuck in poor local optima

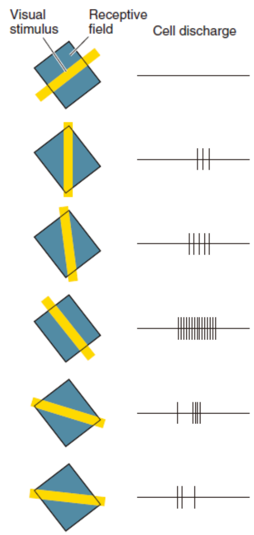
What if we have faster computing hardware and better model?

Things have changed since the end of 2012

Outline

- Why go deep
- Multi-layer perceptron (review)
- Restricted Boltzmann machine
- Deep belief network
- Deep auto-encoder
- Convolutional neural network

Motivation



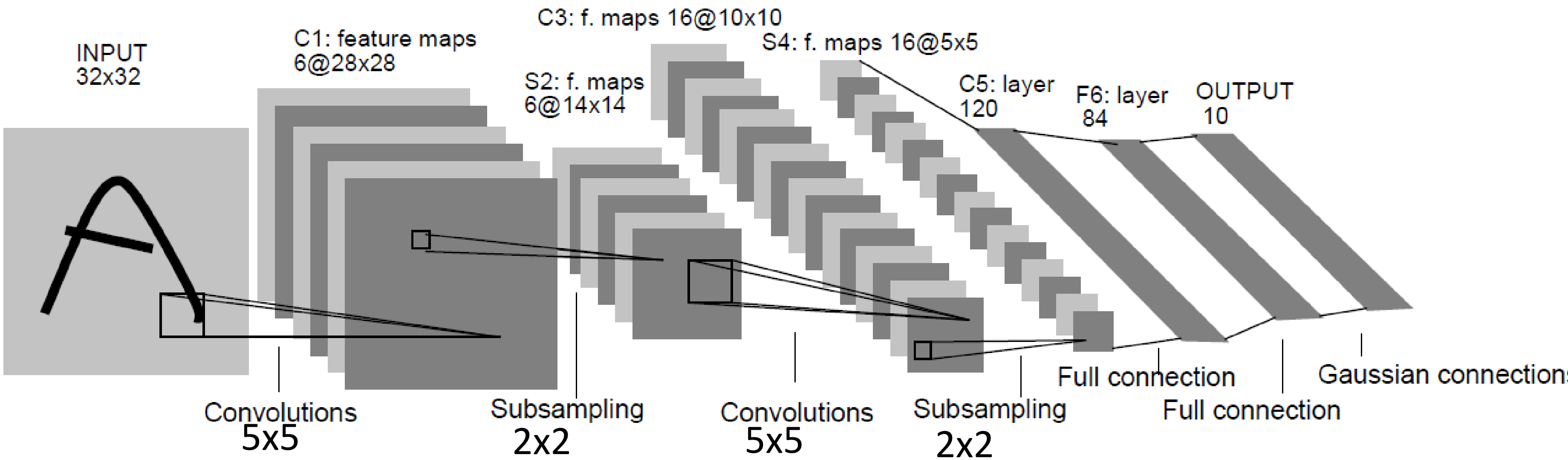
David Hubel
(1926-2013)

Torsten Wiesel
(1924-)

Nobel Prize in Physiology or Medicine, 1981

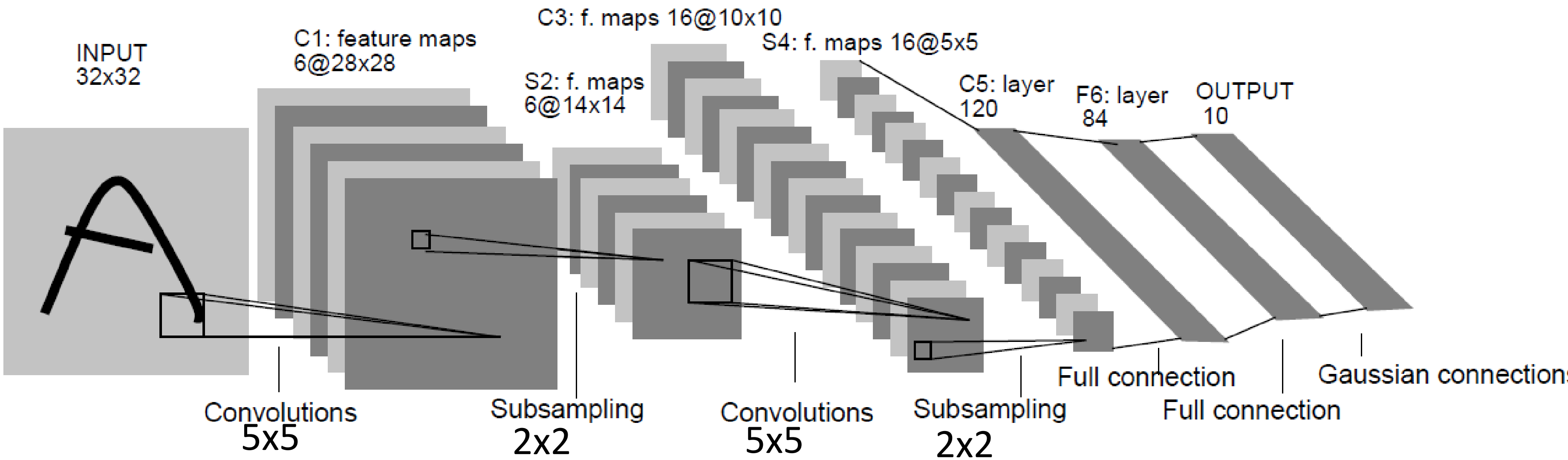
- Hierarchical organization of the visual system
  - Inspired deep learning
- Local receptive field
- Simple cell and complex cell
  - Template matching and pooling
- It inspired Neocognitron (1980), then CNN (late 1980s-1990s)

Convolutional neural network



- Local connections and weight sharing
- C layers: convolution
  - Output $y_i = f(\sum_{\Omega} w_j x_j + b)$ where Ω is the patch size, $f(\cdot)$ is the sigmoid function, w and b are parameters
- S layers: subsampling (avg pooling)
  - Output $y_i = f(w \sum_{\Omega} x_j + b)$ where Ω is the pooling size

Convolutional neural network



- Full connection layers: same as MLP
- The last layer can be either the sigmoid or softmax function

BP algorithm

- Error function $$E = \sum_{n=1}^{N} E^{(n)}$$

where $E^{(n)}$ is the error function for each input sample n

  - Least square error

$$E^{(n)} = \frac{1}{2}\sum_{k=1}^{K}(t_k - y_k^{(L)})^2,\ y_k^{(L)} = \frac{1}{1 + \exp(-w_k^{(L-1)\top} y^{(L-1)}) - b_k^{(L-1)})}$$

  - Cross-entropy error

$$E^{(n)} = -\sum_{k=1}^{K} t_k \ln y_k^{(L)},\quad y_k^{(L)} = \frac{\exp(w_k^{(L-1)\top} y^{(L-1)} + b_k^{(L-1)})}{\sum_{j=1}^{K} \exp(w_j^{(L-1)\top} y^{(L-1)} + b_j^{(L-1)})}$$

- Weight adjustment

Learning rate

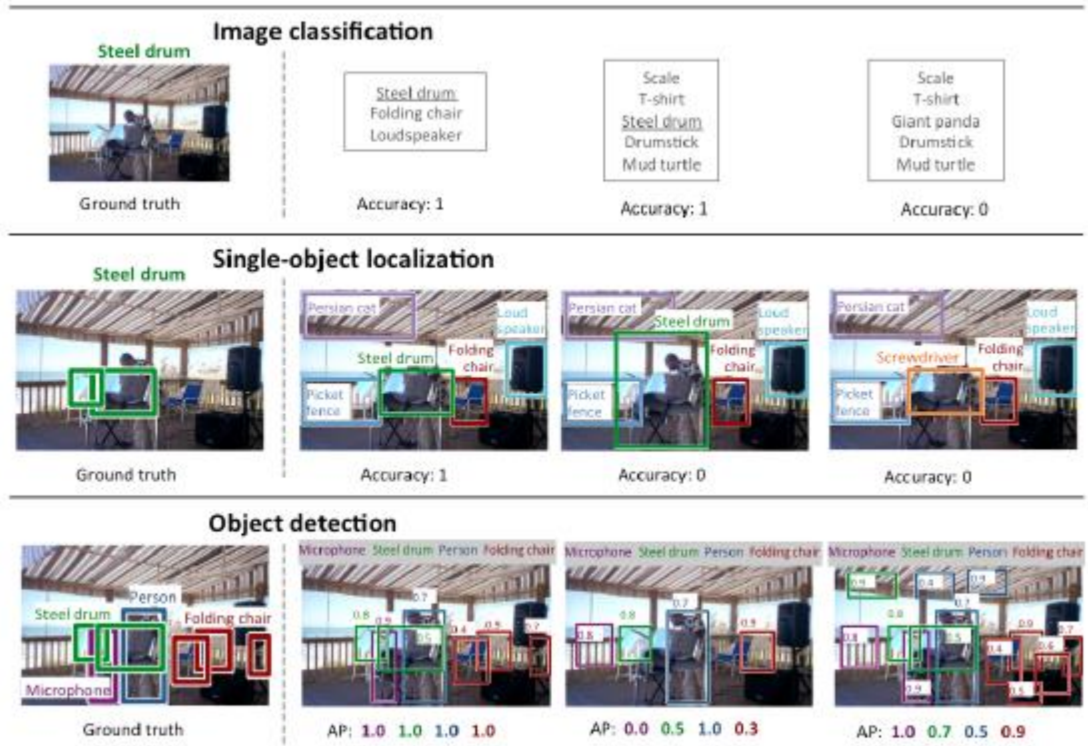
$$w_{ji}^{(l)} = w_{ji}^{(l)} - \alpha \frac{\partial E}{\partial w_{ji}^{(l)}} \qquad b_j^{(l)} = b_j^{(l)} - \alpha \frac{\partial E}{\partial b_j^{(l)}}$$

New trends

- Use GPU for acceleration
- Do not use parameters in pooling layers
- Activation function: rectified linear function is preferred
- Convolutional layers, pooling layers and full connection layers can be arbitrarily placed
  – E.g., not every convolutional layer requires a subsequent pooling layer
  – E.g., full connection layers may be unnecessary

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

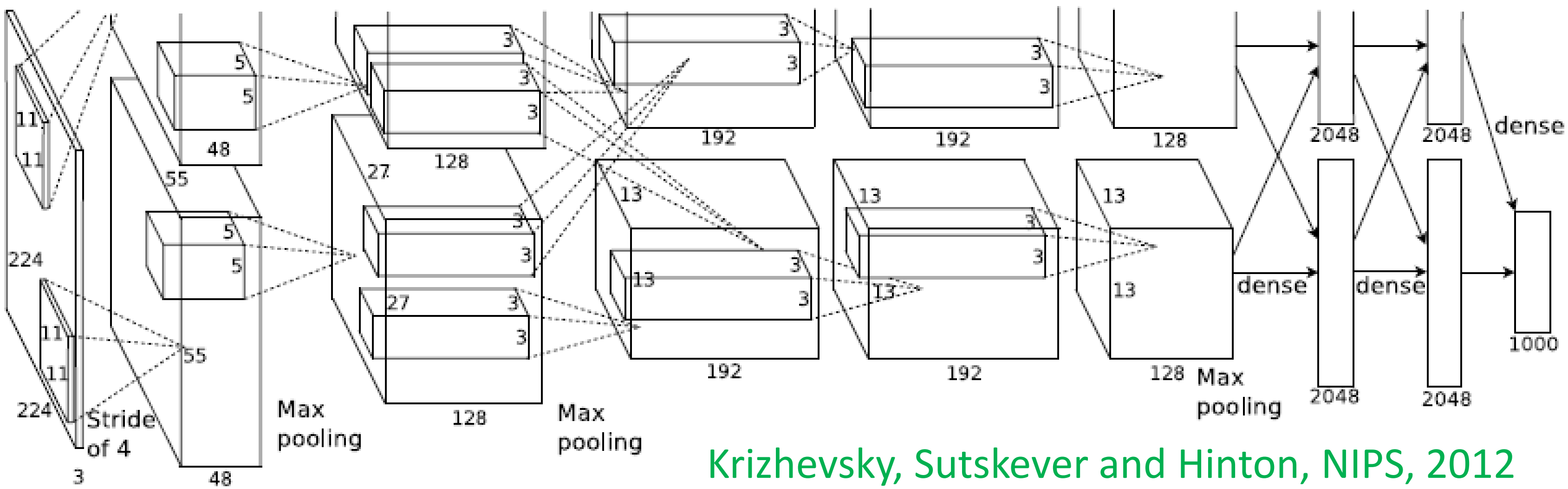
Tasks



The first column shows the ground truth label on an example image, and the next three show three sample outputs with the corresponding evaluation score.

Russakovsky, et al., 2014

CNN for image classification

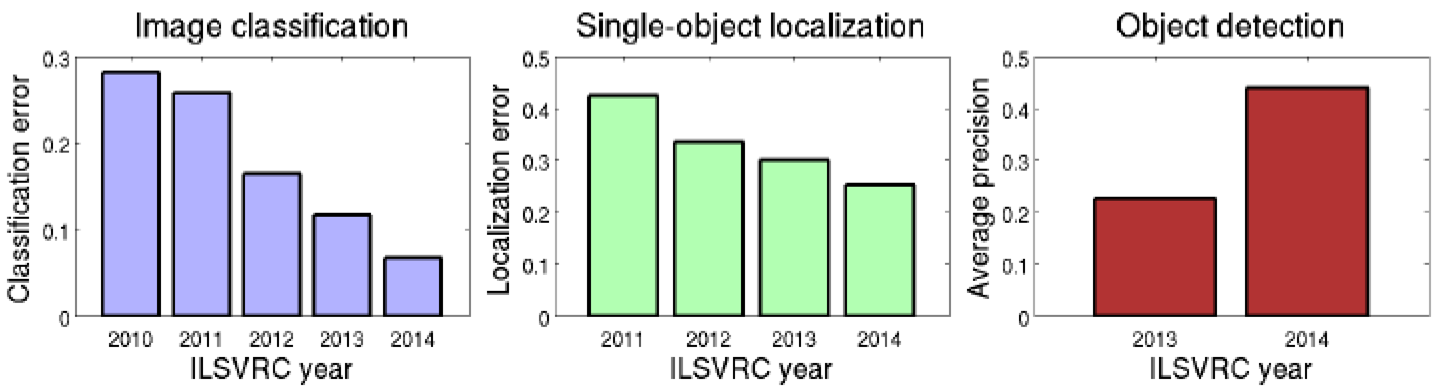


Krizhevsky, Sutskever and Hinton, NIPS, 2012

- Network dimension: 150,528(input)-253,440–186,624–64,896–64,896–43,264–4096–4096–1000(output)
- In total: 60 million parameters
- Task: classify 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes
- Results: Beat all previous models

Results

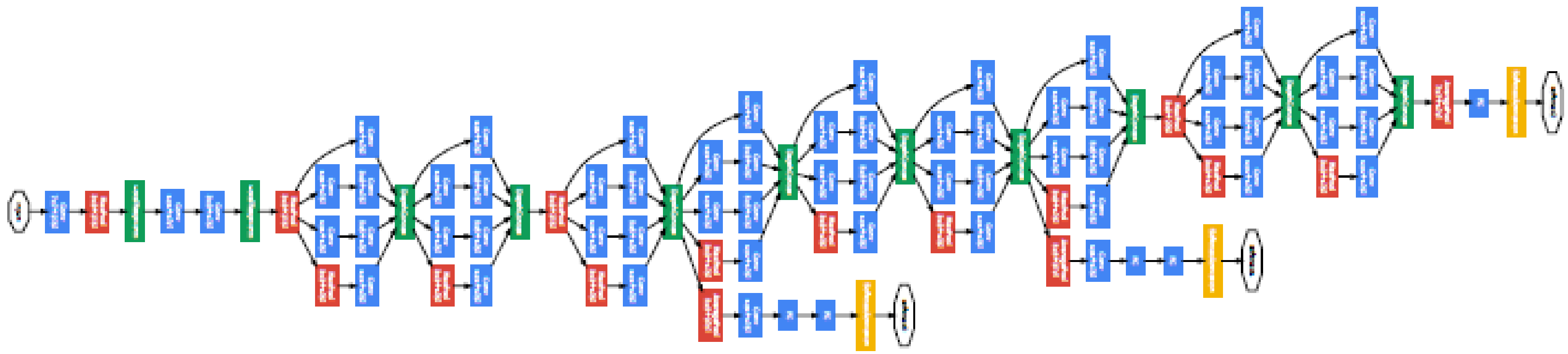
In 2013, the vast majority of teams used CNN.
In 2014, almost all teams used convolutional neural networks.



Relative Confusion	A1	A2
Human succeeds, GoogLeNet succeeds	1352	219
Human succeeds, GoogLeNet fails	72	8
Human fails, GoogLeNet succeeds	46	24
Human fails, GoogLeNet fails	30	7
Total number of images	1500	258
Estimated GoogLeNet classification error	6.8%	5.8%
Estimated human classification error	5.1%	12.0%

Human classication results on the ILSVRC2012-2014 classification test set, for two expert annotators A1 and A2. Top-5 classification error is reported

GoogLeNet

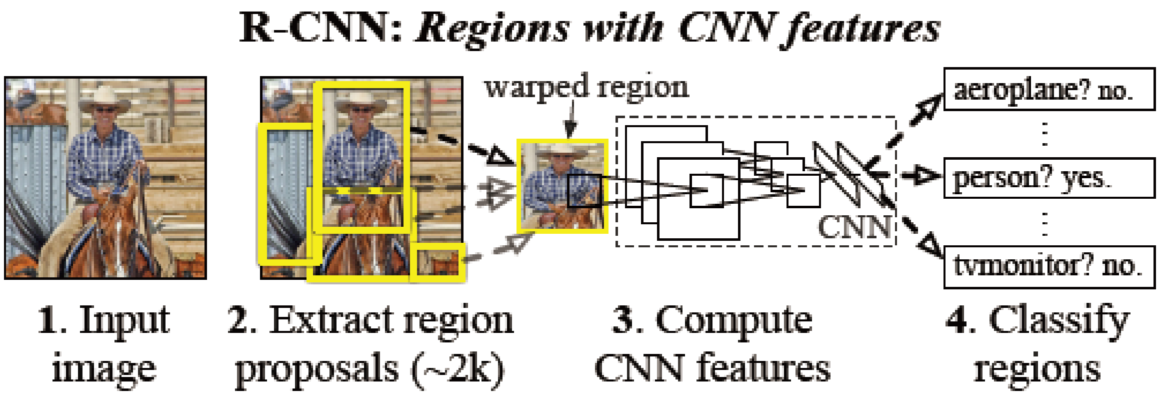


- The network is 22 layers deep when counting only layers with parameters (or 27 layers if we also count pooling)
- Small filters are used (1x1, 3x3, 5x5)
- Two auxiliary classifiers connected to intermediate layers are used to increase the gradient signal for BP algorithm

Szegedy, et al., 2014

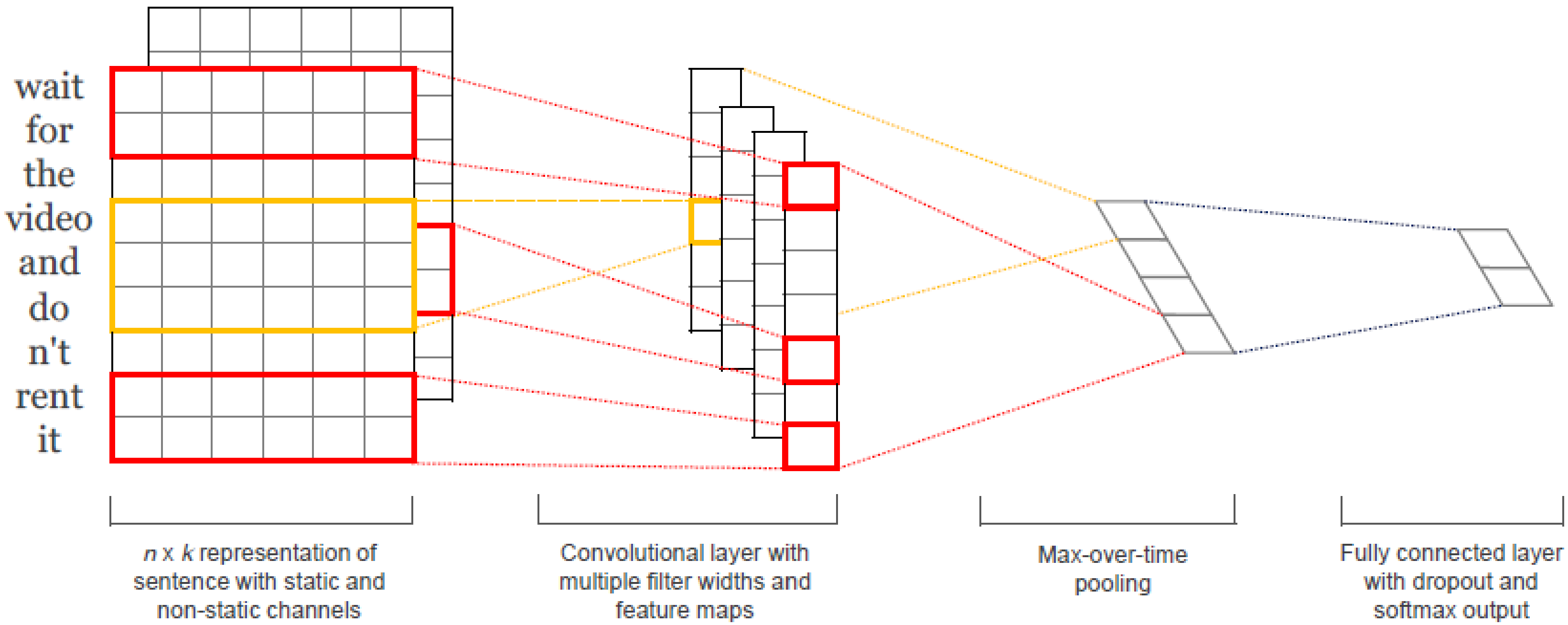
Generic features for computer vision

- The features trained on 1.2M images in ImageNet are generic
  - They have led to state-of-the-art accuracies on other image classification benchmark datasets such as Caltech-101, CIFAR-10
  - They have led to state-of-the-art accuracies in object detection tasks



Girshick, et al, 2013

CNN for sentence classification



- A convolutional layer (multiple filters with different lengths), a global max pooling layer and softmax layer
- Every word is represented by a vector (using word2vec tech)
- This simple model beats other models on some benchmark datasets

Concluding remarks

- Deep learning has achieved exciting results on many real-world problems
- It seems to be a good model for processing big data
- Large models seems to be critical
  - Parallel computing
- Theoretical foundations are lacked
- Relation to neuroscience
  - Inspired by neuroscience
  - Many neuroscience findings are not incorporated

Online resource

- Website: http://deeplearning.net/
  - A reading list
  - Software
  - Datasets
  - Tutorials and demos