Introduction to Machine Learning CMU-10701

Deep Learning

Barnabás Póczos & Aarti Singh

Credits

Many of the pictures, results, and other materials are taken from:

Ruslan Salakhutdinov
Joshua Bengio
Geoffrey Hinton
Yann LeCun

Contents

- Definition and Motivation
- History of Deep architectures
- Deep architectures
  - Convolutional networks
  - Deep Belief networks
- Applications

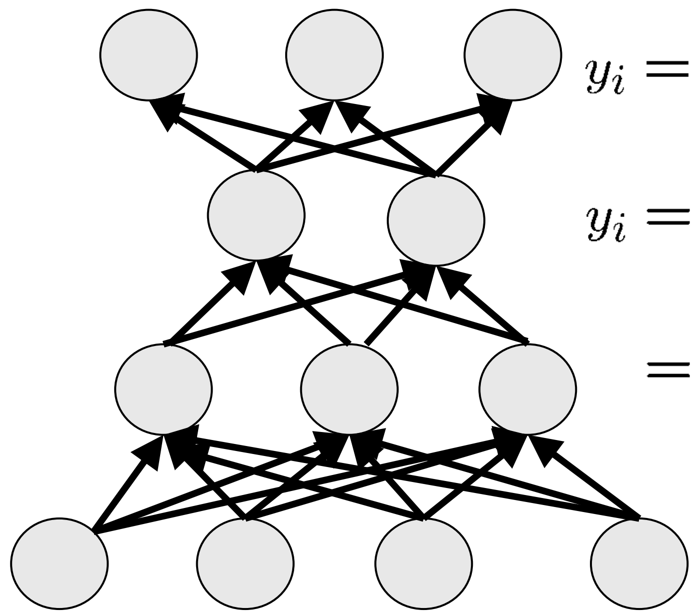
Deep architectures

Defintion: Deep architectures are composed of *multiple levels* of non-linear operations, such as neural nets with many hidden layers.

Output layer

Hidden layers

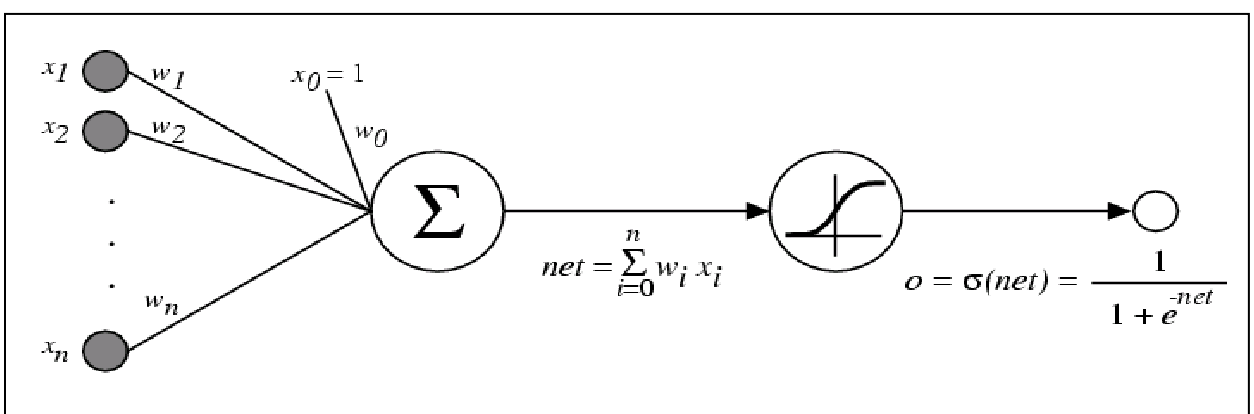
Input layer



$$y_i = \tanh\left(\sum_j W_{ij} x_j + W_{i0}\right)$$

$$y_i = \sigma\left(\sum_j W_{ij} x_j + W_{i0}\right)$$

$$= \frac{1}{1 + \exp\left[-\left(\sum_j W_{ij} x_j + W_{i0}\right)\right]}$$



Goal of Deep architectures

Goal: Deep learning methods aim at

- learning *feature hierarchies*
- where features from higher levels of the hierarchy are formed by lower level features.

edges, local shapes, object parts

Low level representation

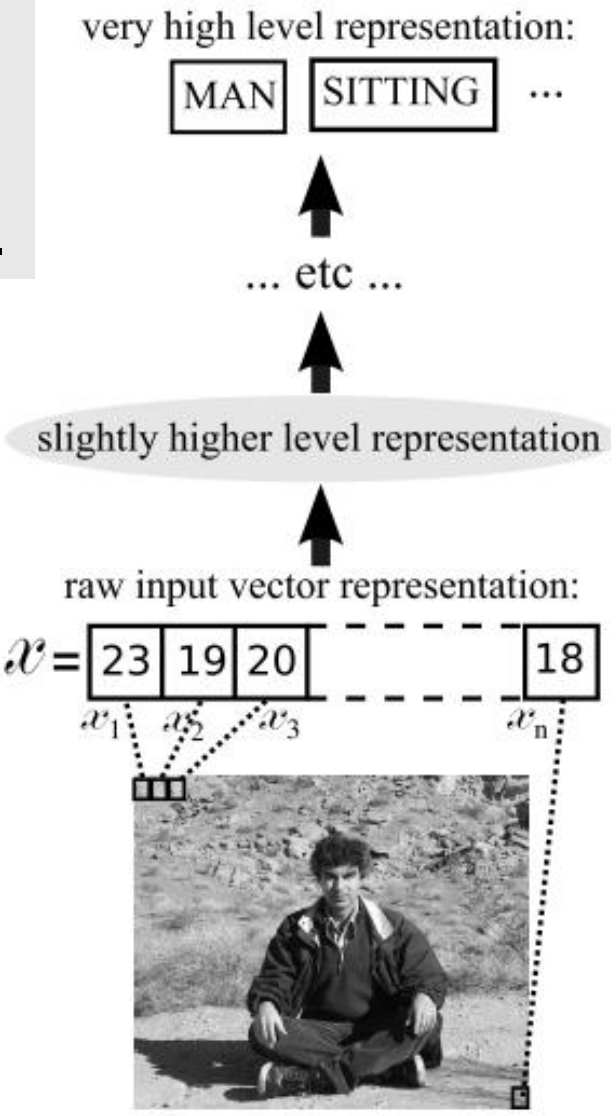
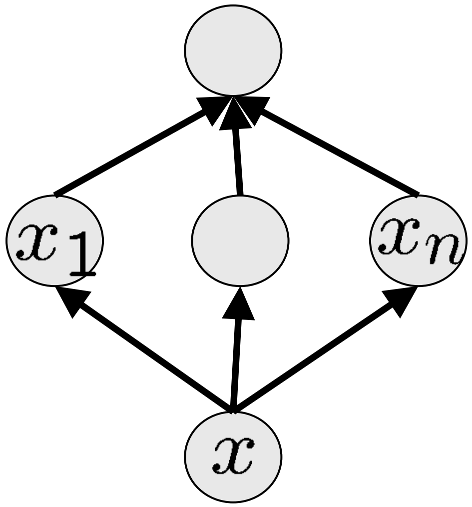


Figure is from Yoshua Bengio

Neurobiological Motivation

- Most current learning algorithms are shallow architectures (1-3 levels) (SVM, kNN, MoG, KDE, Parzen Kernel regression, PCA, Perceptron,…)

SVM: $\hat{f}(\mathbf{x}) = \mathrm{sign}(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}))$



- The mammal brain is organized in a deep architecture (Serre, Kreiman, Kouh, Cadieu, Knoblich, & Poggio, 2007)
  (E.g. visual system has 5 to 10 levels)

Deep Learning History

- **Inspired** by the architectural depth of the brain, researchers wanted for decades to train deep multi-layer neural networks.
- **No successful** attempts were reported before 2006 …

  Researchers reported positive experimental results with typically two or three levels (i.e. one or two hidden layers), but training deeper networks consistently yielded poorer results.
- **Exception**: convolutional neural networks, LeCun 1998
- **SVM**: Vapnik and his co-workers developed the Support Vector Machine (1993). It is a shallow architecture.
- **Digression**: In the 1990's, many researchers abandoned neural networks with multiple adaptive hidden layers because SVMs worked better, and there was no successful attempts to train deep networks.
- **Breakthrough in 2006**

Breakthrough

Deep Belief Networks (DBN)

Hinton, G. E, Osindero, S., and Teh, Y. W. (2006).
A fast learning algorithm for deep belief nets.
Neural Computation, 18:1527-1554.

Autoencoders

Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007).
Greedy Layer-Wise Training of Deep Networks,
Advances in Neural Information Processing Systems 19

Theoretical Advantages of Deep Architectures

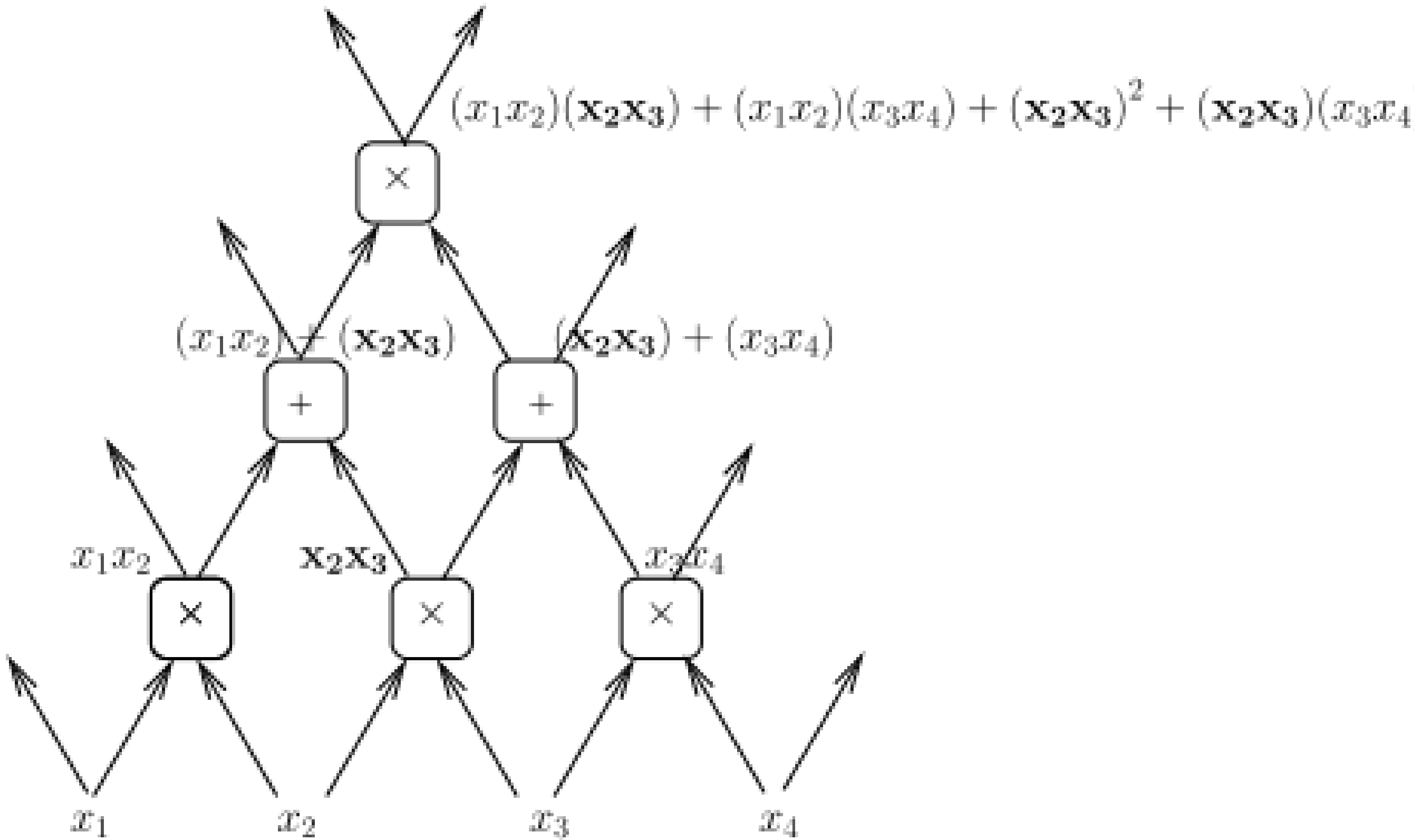
- Some functions cannot be efficiently represented (in terms of number of tunable elements) by architectures that are too shallow.
- Deep architectures might be able to represent some functions otherwise not efficiently representable.
- **More formally**:

  Functions that can be compactly represented by a depth k architecture might require an exponential number of computational elements to be represented by a depth k – 1 architecture

- The consequences are
  - **Computational**: We don’t need exponentially many elements in the layers
  - **Statistical**: poor generalization may be expected when using an insufficiently deep architecture for representing some functions.

Theoretical Advantages of Deep Architectures

The Polynoimal circuit:



Deep Convolutional Networks

Deep Convolutional Networks

- Deep supervised neural networks are generally too difficult to train.

- **One notable exception**: convolutional neural networks (CNN)

- Convolutional nets were inspired by the visual system's structure

- They typically have five, six or seven layers, a number of layers which makes fully-connected neural networks almost impossible to train properly when initialized randomly.

Deep Convolutional Networks

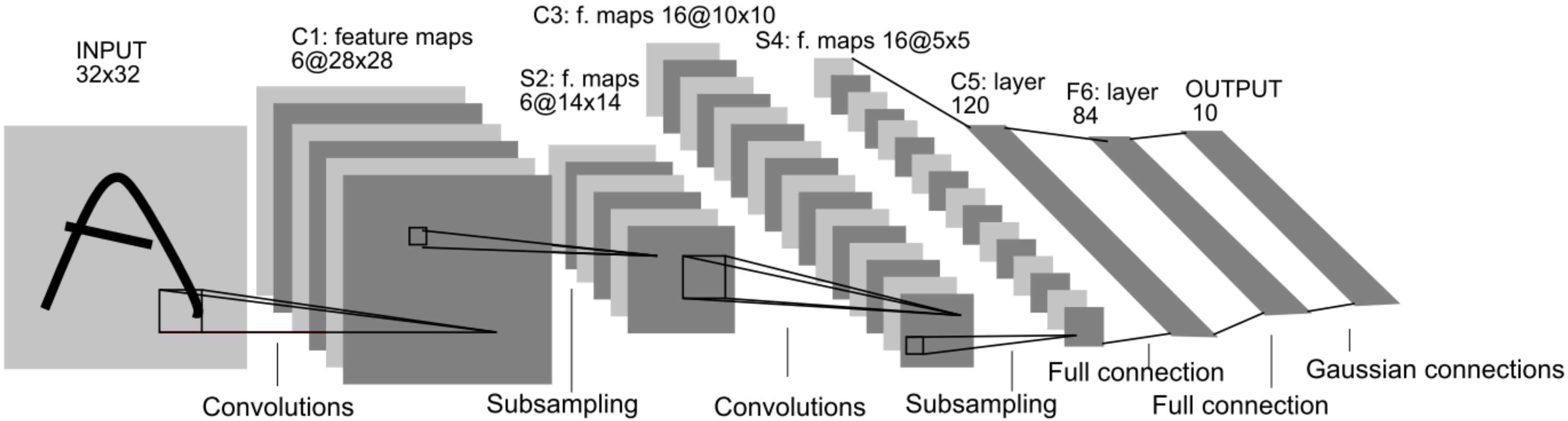
Compared to standard feedforward neural networks with similarly-sized layers,

- CNNs have much fewer connections and parameters
- and so they are easier to train,
- while their theoretically-best performance is likely to be only slightly worse.

LeNet 5

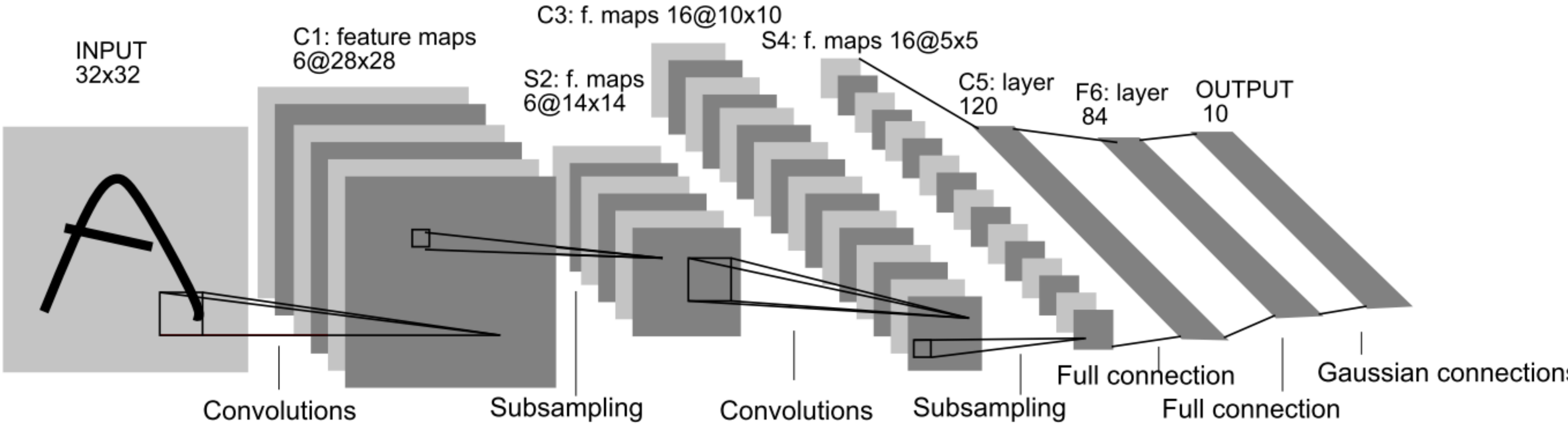
Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: **Gradient-Based Learning Applied to Document Recognition**, *Proceedings of the IEEE, 86(11):2278-2324, November* ***1998***

LeNet 5, LeCun 1998



- Input: 32x32 pixel image. Largest character is 20x20
(All important info should be in the center of the receptive field of the highest level feature detectors)
- Cx: Convolutional layer
- Sx: Subsample layer
- Fx: Fully connected layer
- Black and White pixel values are normalized:
  E.g. White = -0.1, Black =1.175 (Mean of pixels = 0, Std of pixels =1)

LeNet 5, Layer C1



C1: Convolutional layer with 6 feature maps of size 28x28. $C1_k$ (k=1…6)

Each unit of C1 has a 5x5 receptive field in the input layer.

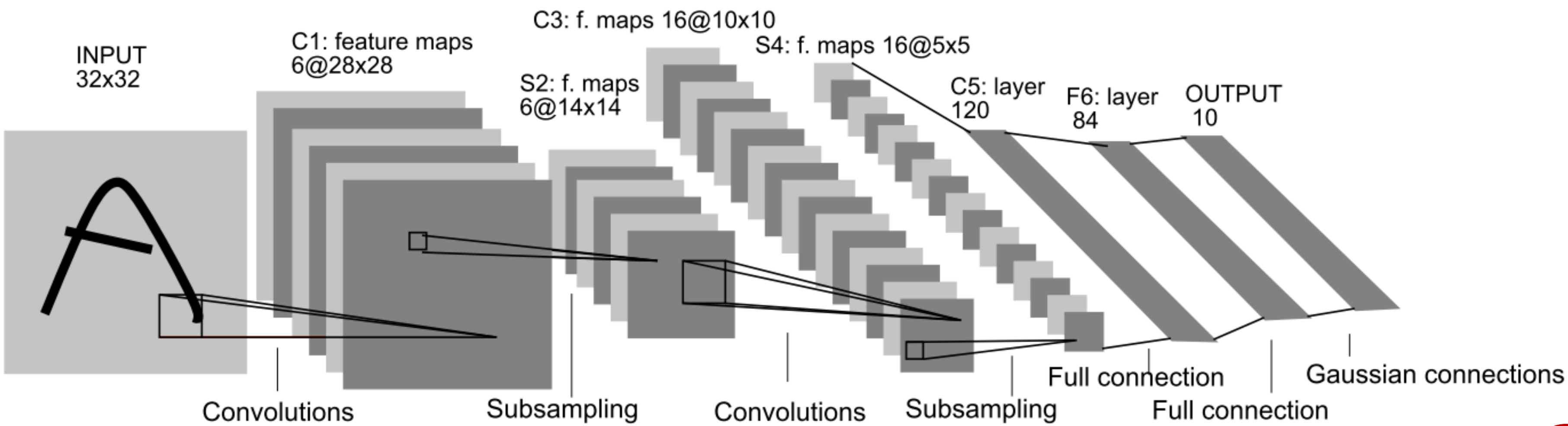
- Topological structure
- Sparse connections
- Shared weights

(5*5+1)*6=156 parameters to learn

Connections: 28*28*(5*5+1)*6=122304

If it was fully connected we had (32*32+1)*(28*28)*6 parameters

LeNet 5, Layer S2



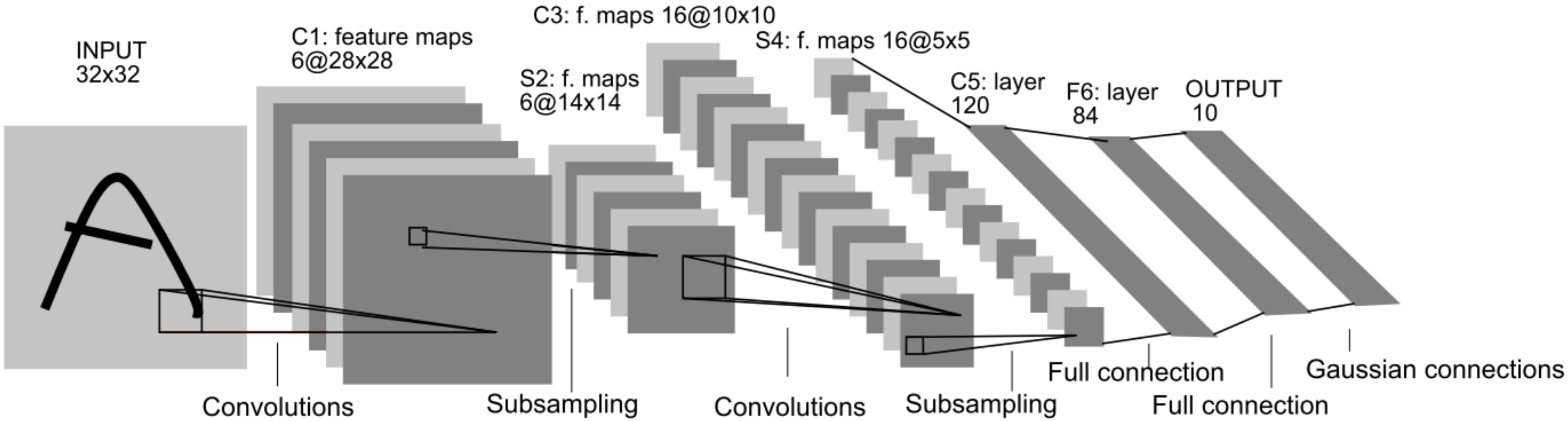
S2: Subsampling layer with 6 feature maps of size 14x14

2x2 nonoverlapping receptive fields in C1

Layer S2: 6*2=12 trainable parameters.

Connections: 14*14*(2*2+1)*6=5880

LeNet 5, Layer C3



- C3: Convolutional layer with 16 feature maps of size 10x10
- Each unit in C3 is connected to several! 5x5 receptive fields at identical locations in S2

Layer C3:

1516 trainable parameters.

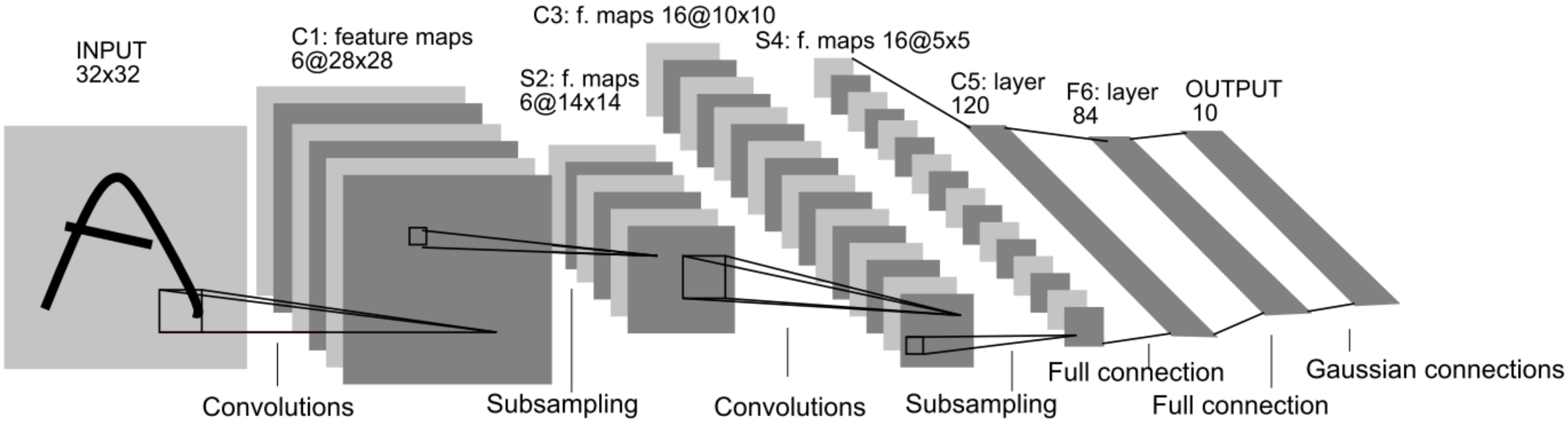
Connections: 151600

	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.

LeNet 5, Layer S4

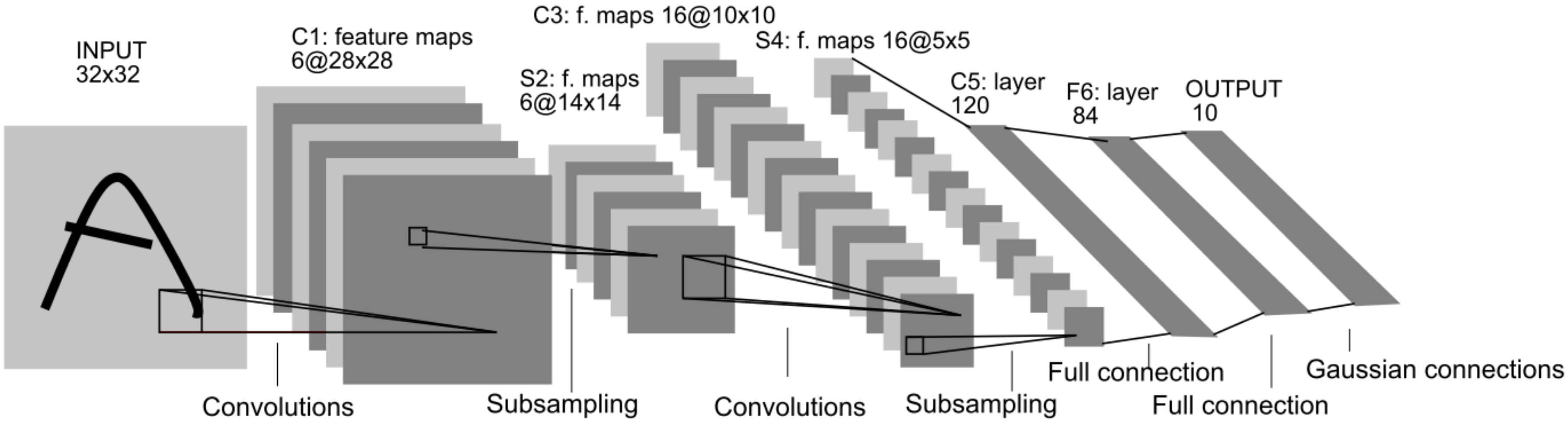


- S4: Subsampling layer with 16 feature maps of size 5x5
- Each unit in S4 is connected to the corresponding 2x2 receptive field at C3

Layer S4: 16*2=32 trainable parameters.

Connections: 5*5*(2*2+1)*16=2000

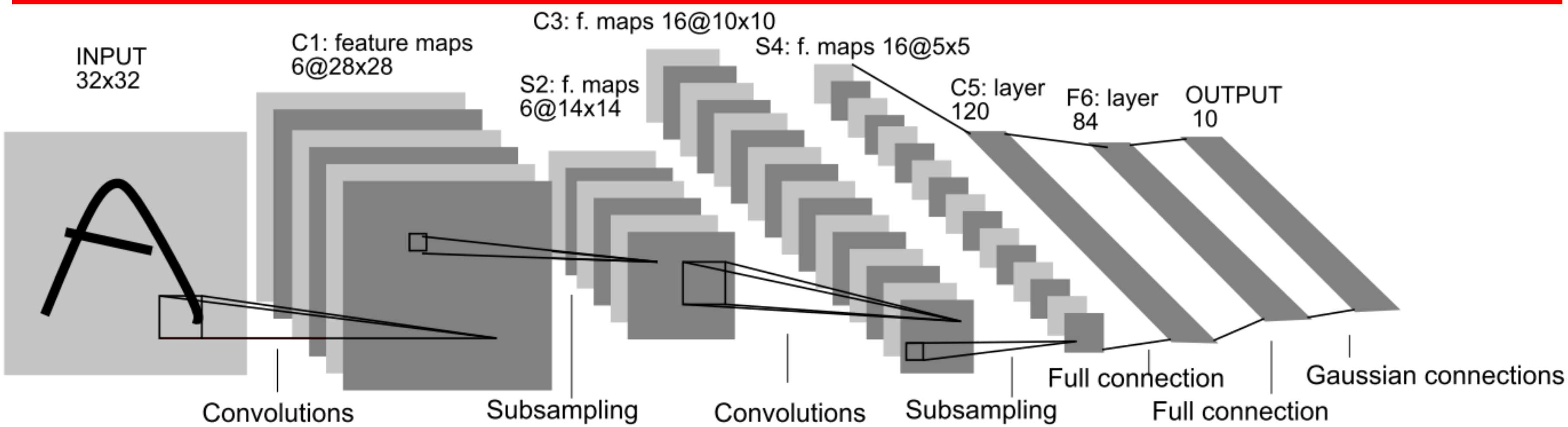
LeNet 5, Layer C5



- C5: Convolutional layer with 120 feature maps of size 1x1
- Each unit in C5 is connected to all 16 5x5 receptive fields in S4

Layer C5: 120*(16*25+1) = 48120 trainable parameters and connections
(Fully connected)

LeNet 5, Layer C5



Layer F6: 84 fully connected units. 84*(120+1)=10164 trainable parameters and connections.

Output layer: 10RBF (One for each digit)

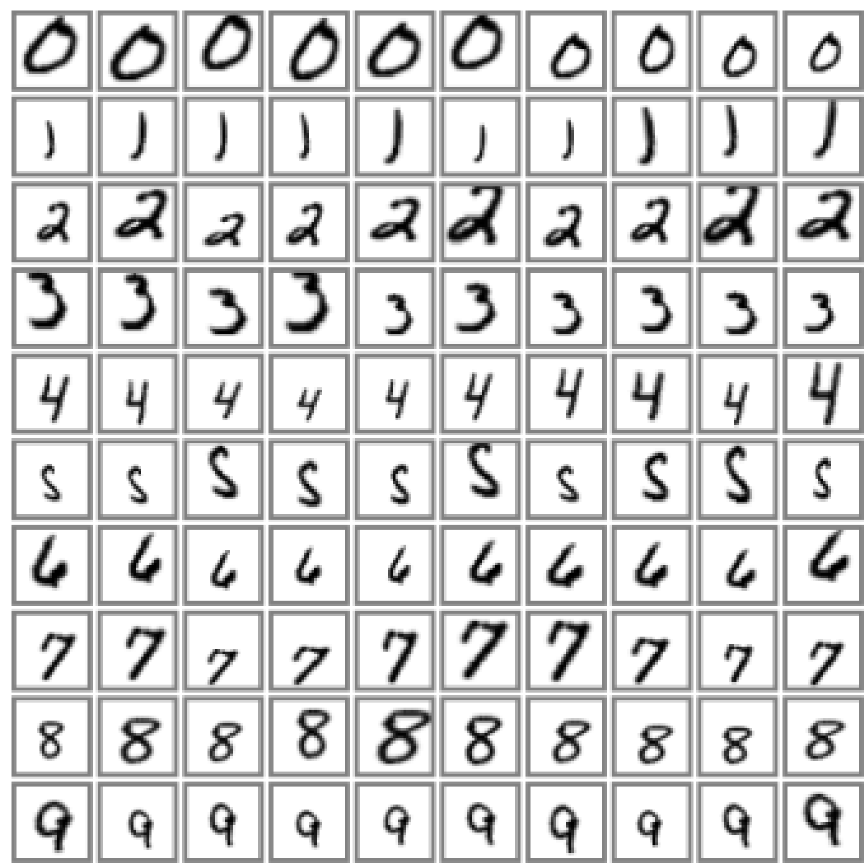
84=7x12, stylized image

Weight update: Backpropagation

MINIST Dataset

60,000 original datasets

Test error: 0.95%



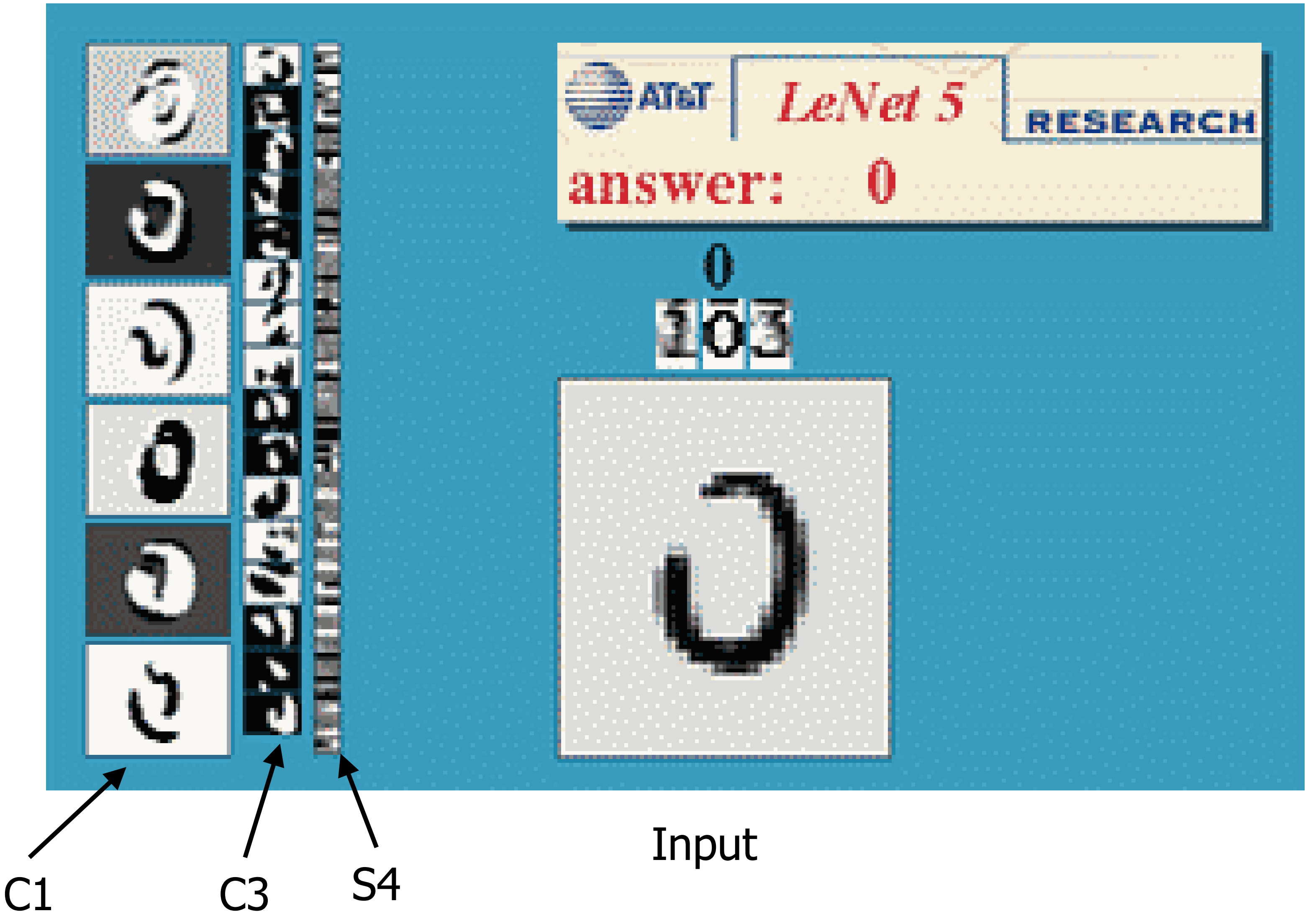
540,000 artificial distortions

+ 60,000 original

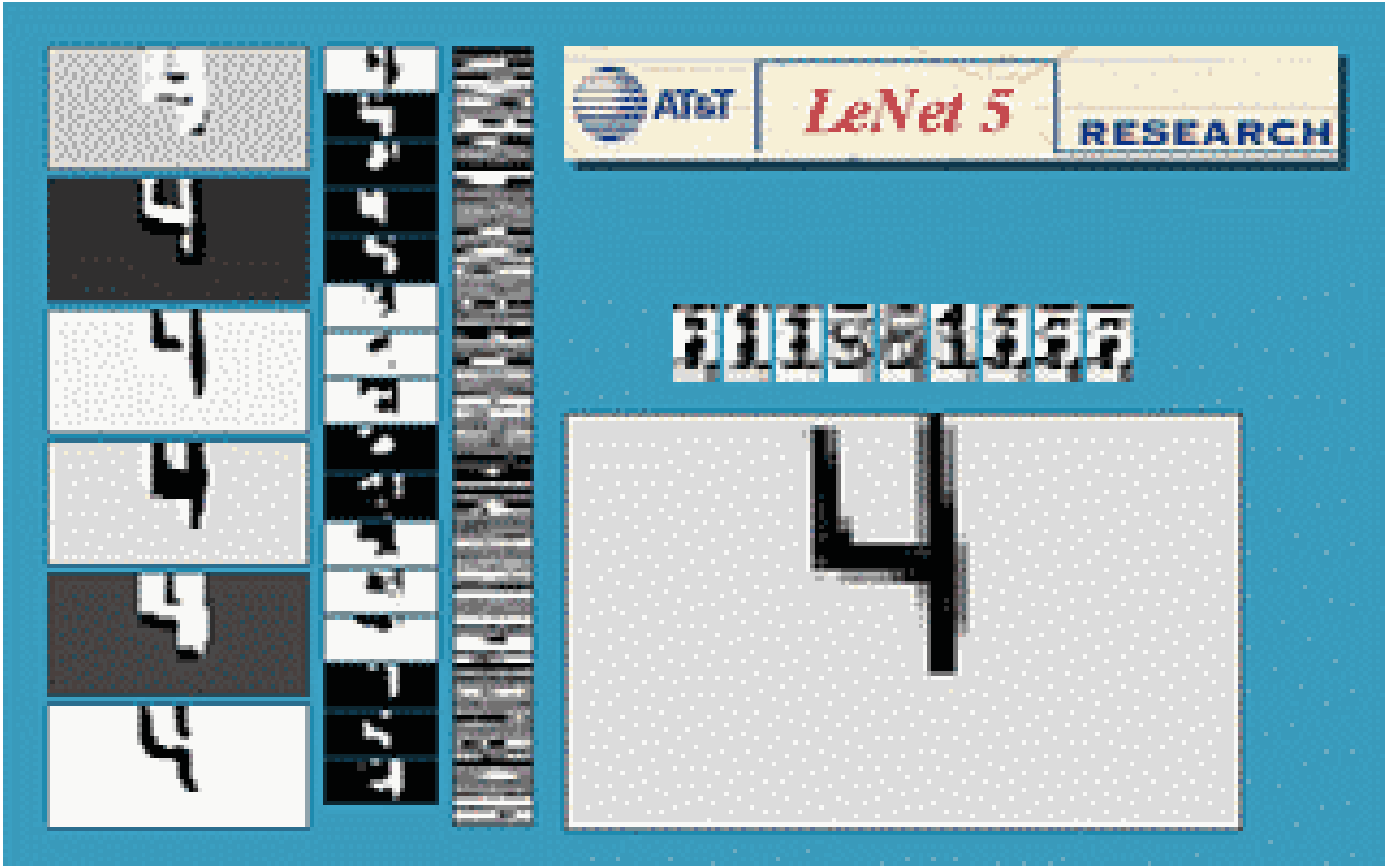
Test error: 0.8%

Misclassified examples

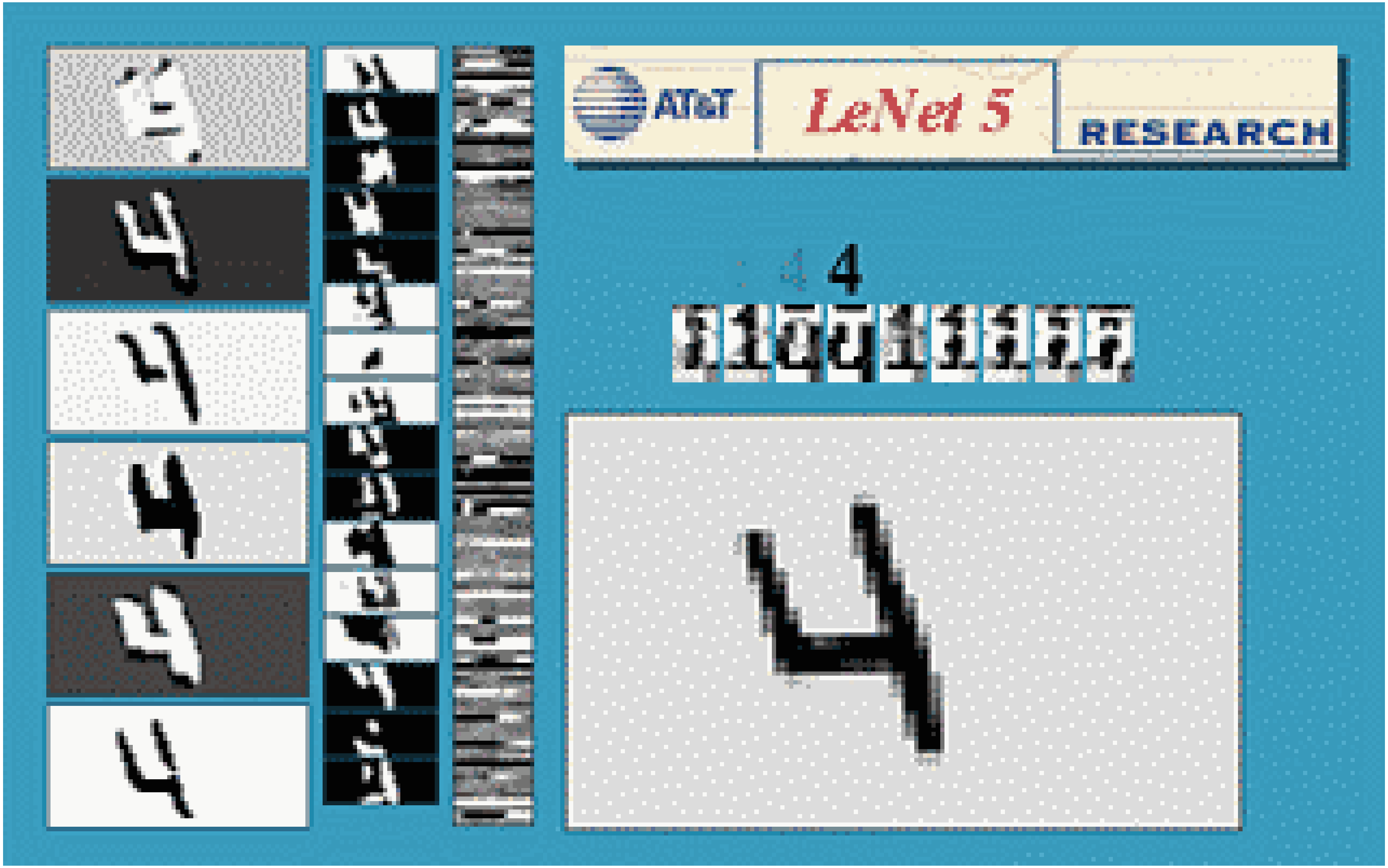
LeNet 5 in Action



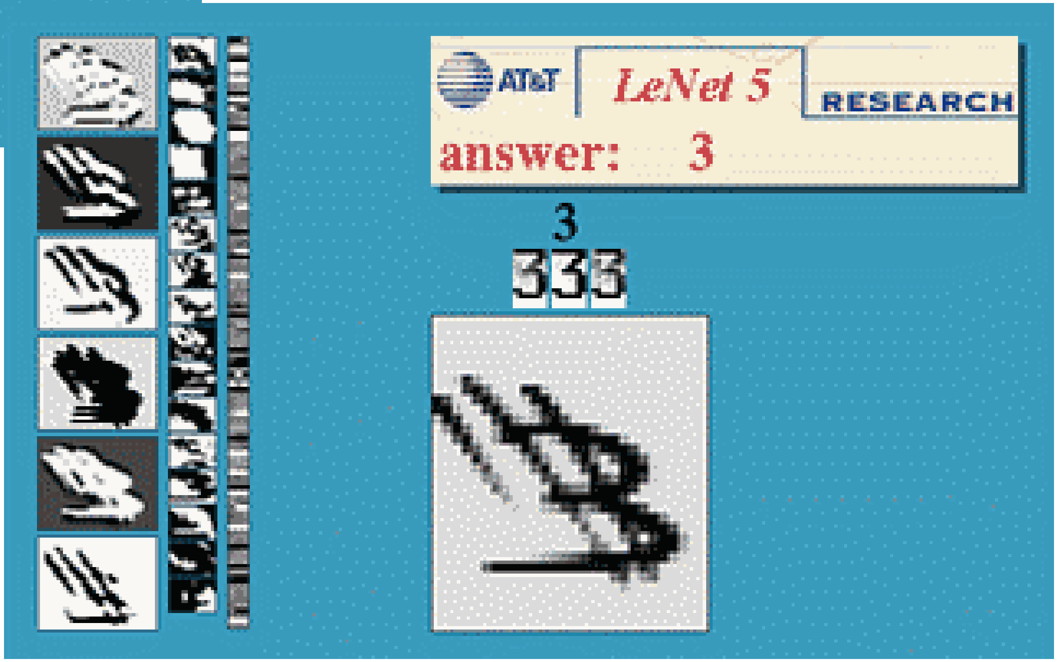
LeNet 5, Shift invariance



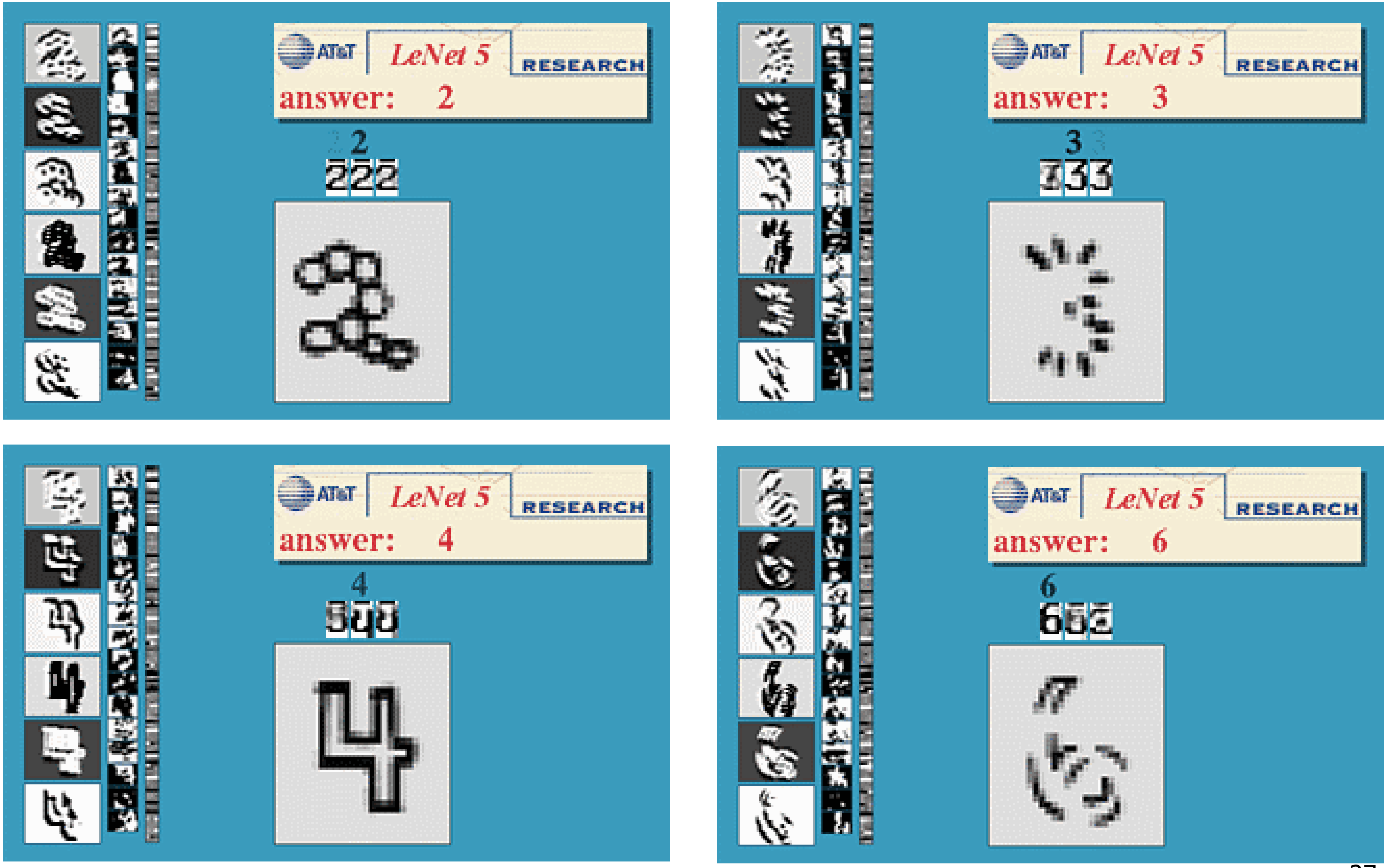
LeNet 5, Rotation invariance



LeNet 5, Nosie resistance



LeNet 5, Unusual Patterns



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton,

Advances in Neural Information Processing Systems 2012

ImageNet

- 15M images
- 22K categories
- Images collected from Web
- Human labelers (Amazon’s Mechanical Turk crowd-sourcing)
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)
    - 1K categories
    - 1.2M training images (~1000 per category)
    - 50,000 validation images
    - 150,000 testing images
- RGB images
- Variable-resolution, but this architecture scales them to 256x256 size

ImageNet

Classification goals:

- ❑ Make 1 guess about the label (Top-1 error)
- ❑ make 5 guesses about the label (Top-5 error)

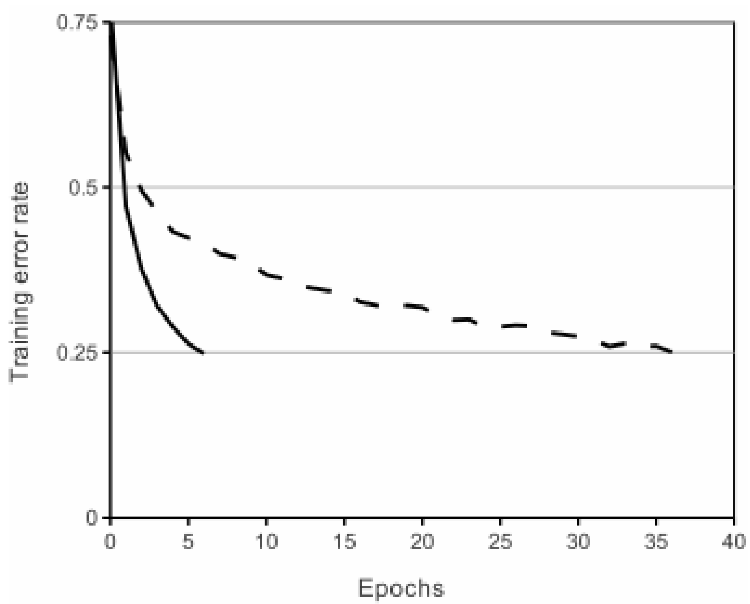
The Architecture

Typical nonlinearities: $f(x) = \tanh(x)$

$$f(x) = (1 + e^{-x})^{-1}$$

Here, however, Rectified Linear Units (ReLU) are used: $f(x) = \max(0, x)$

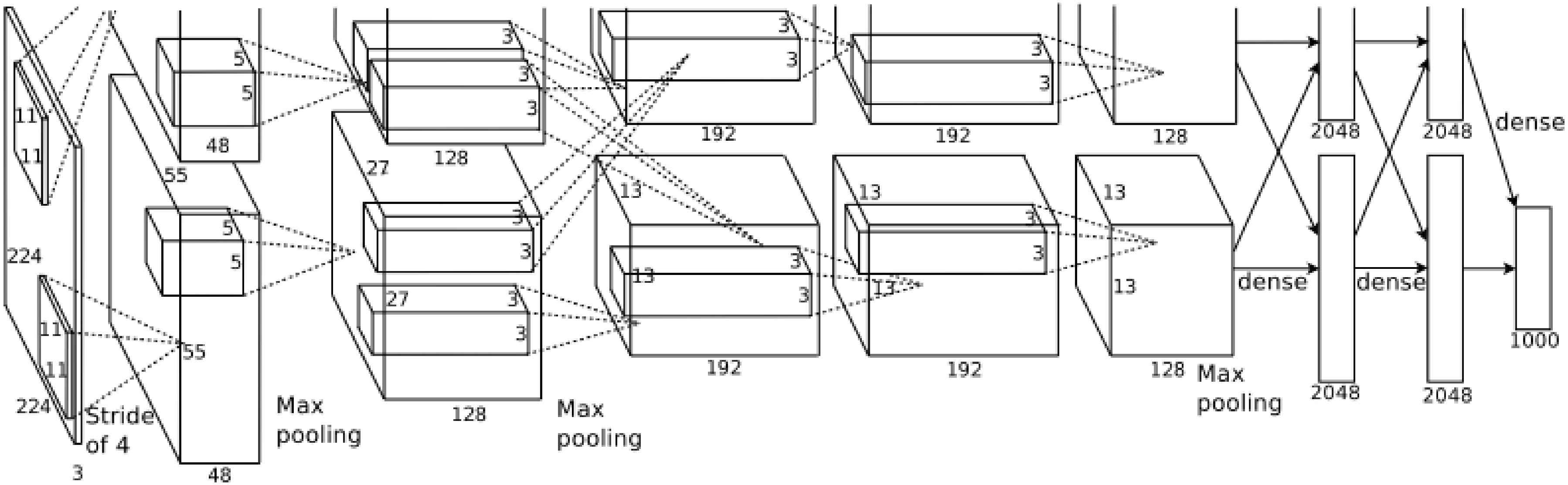
Empirical observation: Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units



A four-layer convolutional neural network with ReLUs (solid line) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons

(dashed line)

The Architecture



The first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in the kernel map. 224/4=56

The pooling layer: form of non-linear down-sampling. Max-pooling partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum value

The Architecture

- Trained with stochastic gradient descent
- on two NVIDIA GTX 580 3GB GPUs
- for about a week

- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- 5 convolutional layer, 3 fully connected layer
- Final feature layer: 4096-dimensional

Data Augmentation

The easiest and most common method to **reduce overfitting** on image data is to artificially **enlarge the dataset** using label-preserving transformations.

We employ two distinct forms of data augmentation:

- image translation
- horizontal reflections
- changing RGB intensities

Dropout

- We know that combining different models can be very useful (Mixture of experts, majority voting, boosting, etc)

- Training many different models, however, is very time consuming.

The solution:

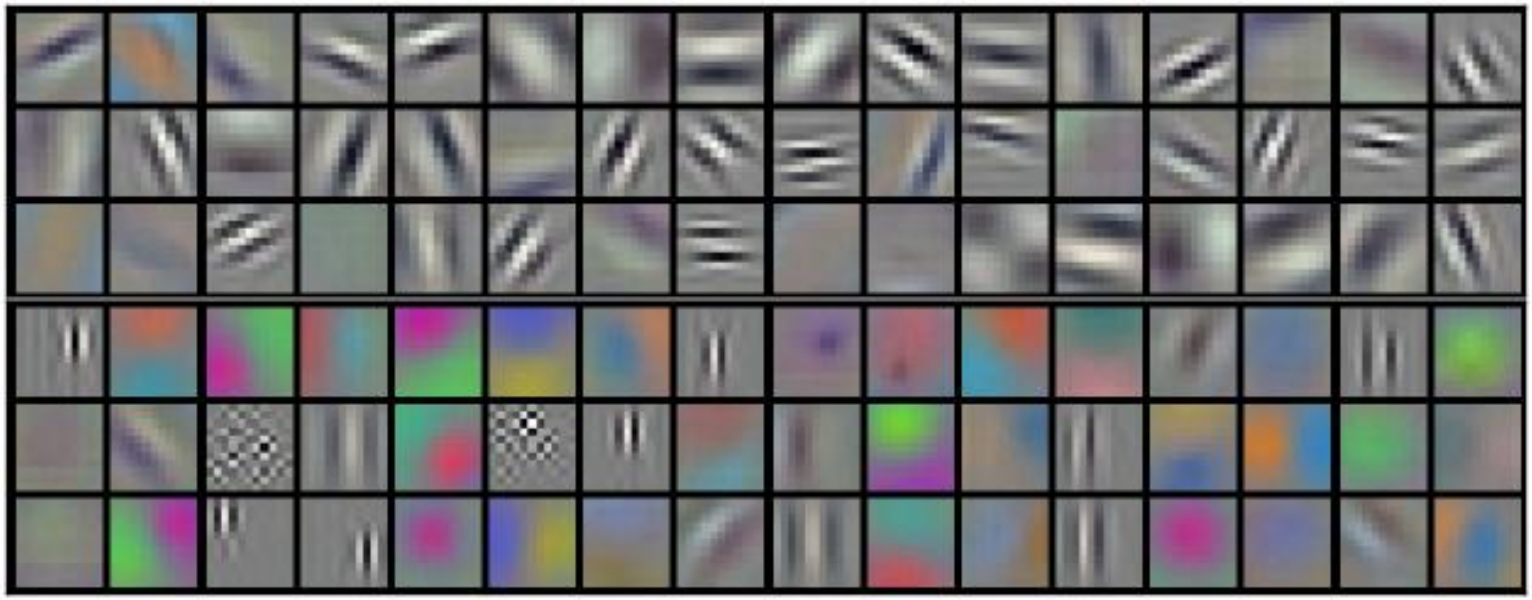
Dropout: set the output of each hidden neuron to zero w.p. 0.5.

Dropout

Dropout: set the output of each hidden neuron to zero w.p. 0.5.

- The neurons which are "dropped out" in this way do not contribute to the forward pass and do not participate in backpropagation.
- So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights.
- This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons.
- It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.
- Without dropout, our network exhibits substantial overfitting.
- Dropout roughly doubles the number of iterations required to converge.

The first convolutional layer



96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images.

The top 48 kernels were learned on GPU1 while the bottom 48 kernels were learned on GPU2

Looks like Gabor wavelets, ICA filters…

Results

Results on the test data:

top-1 error rate: 37.5%

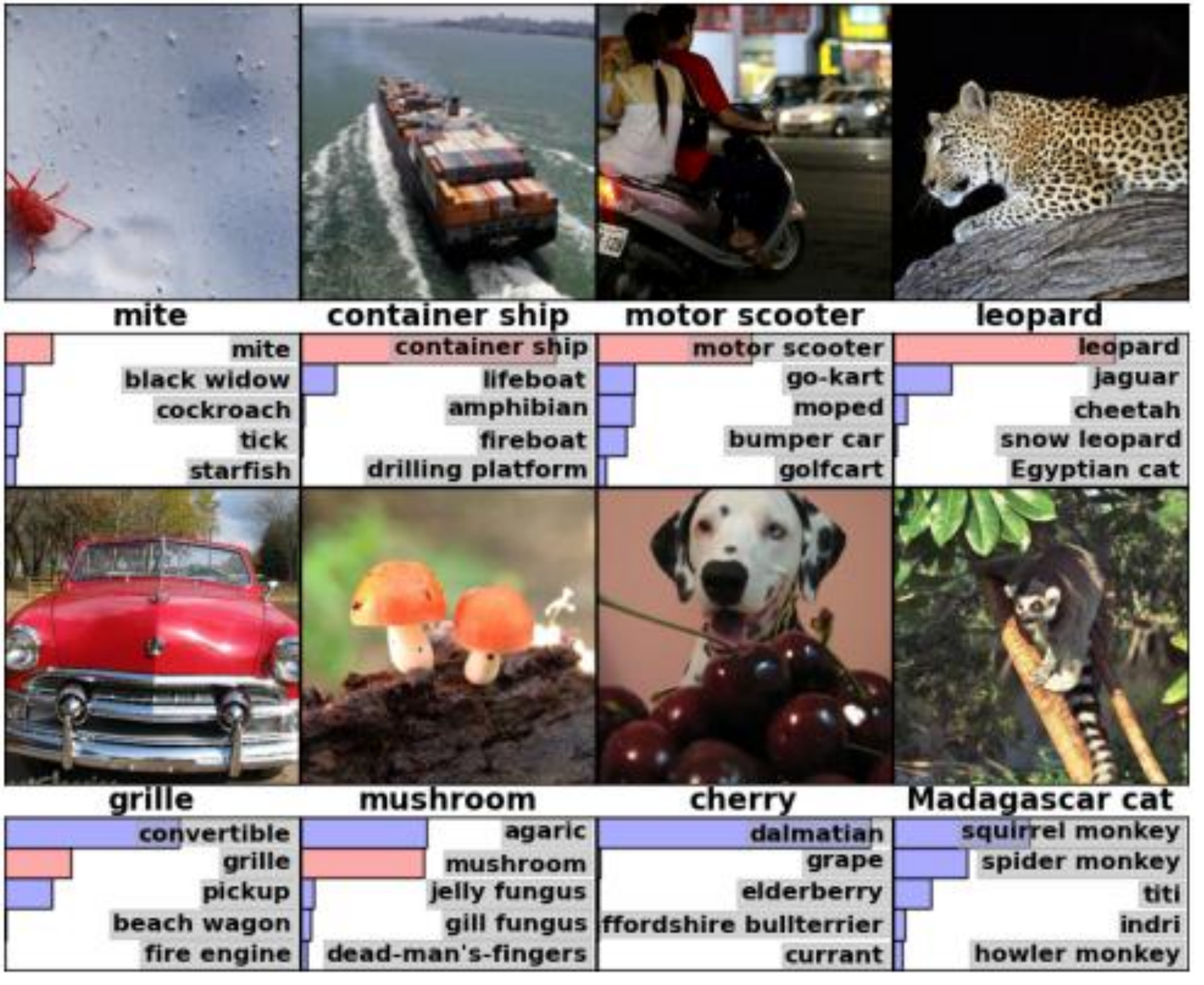
top-5 error rate: 17.0%

ILSVRC-2012 competition:

15.3% accuracy

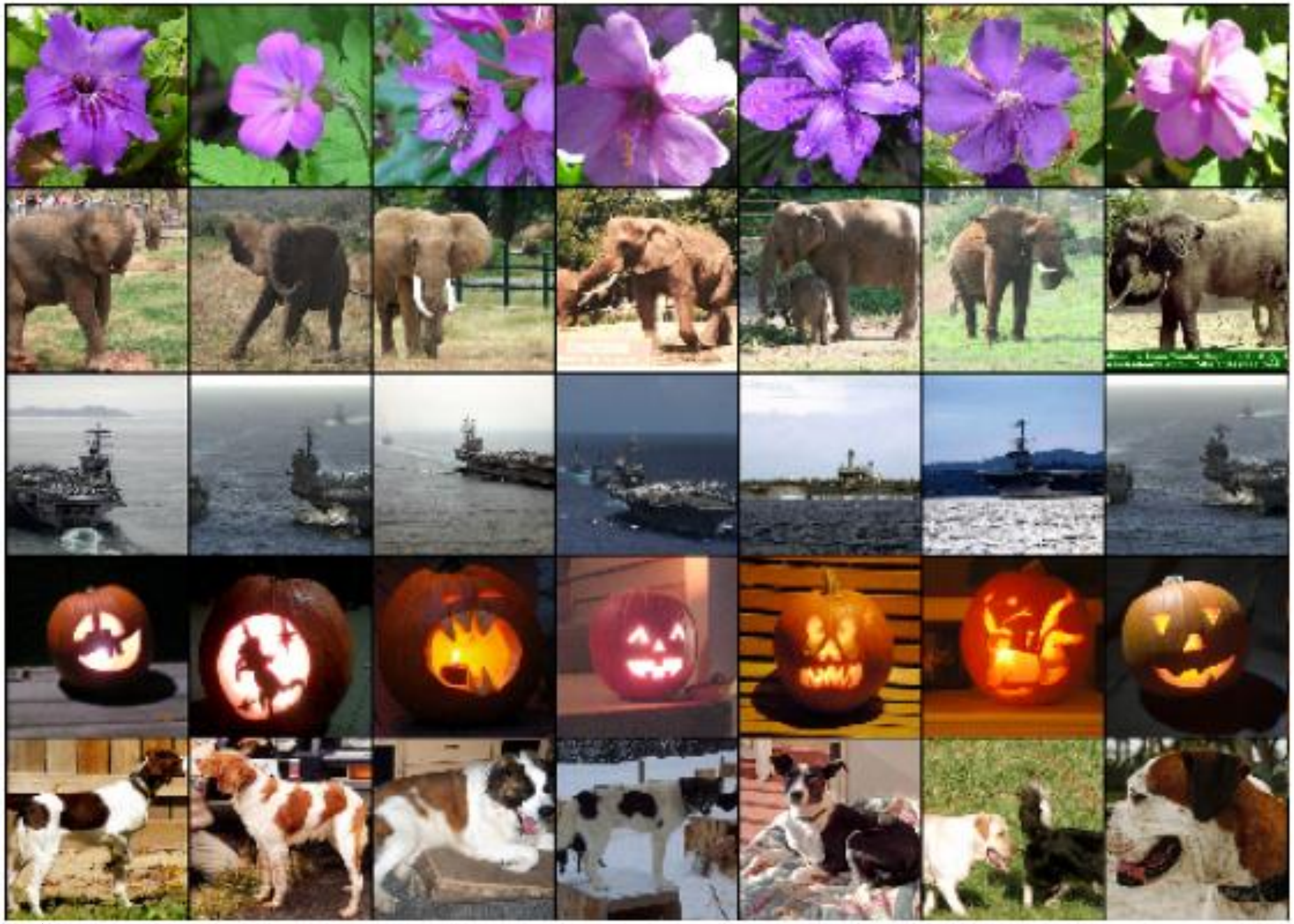
2nd best team: 26.2% accuracy

Results



mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Results: Image similarity



Test column

six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

Deep Belief Networks

What is wrong with back propagation?

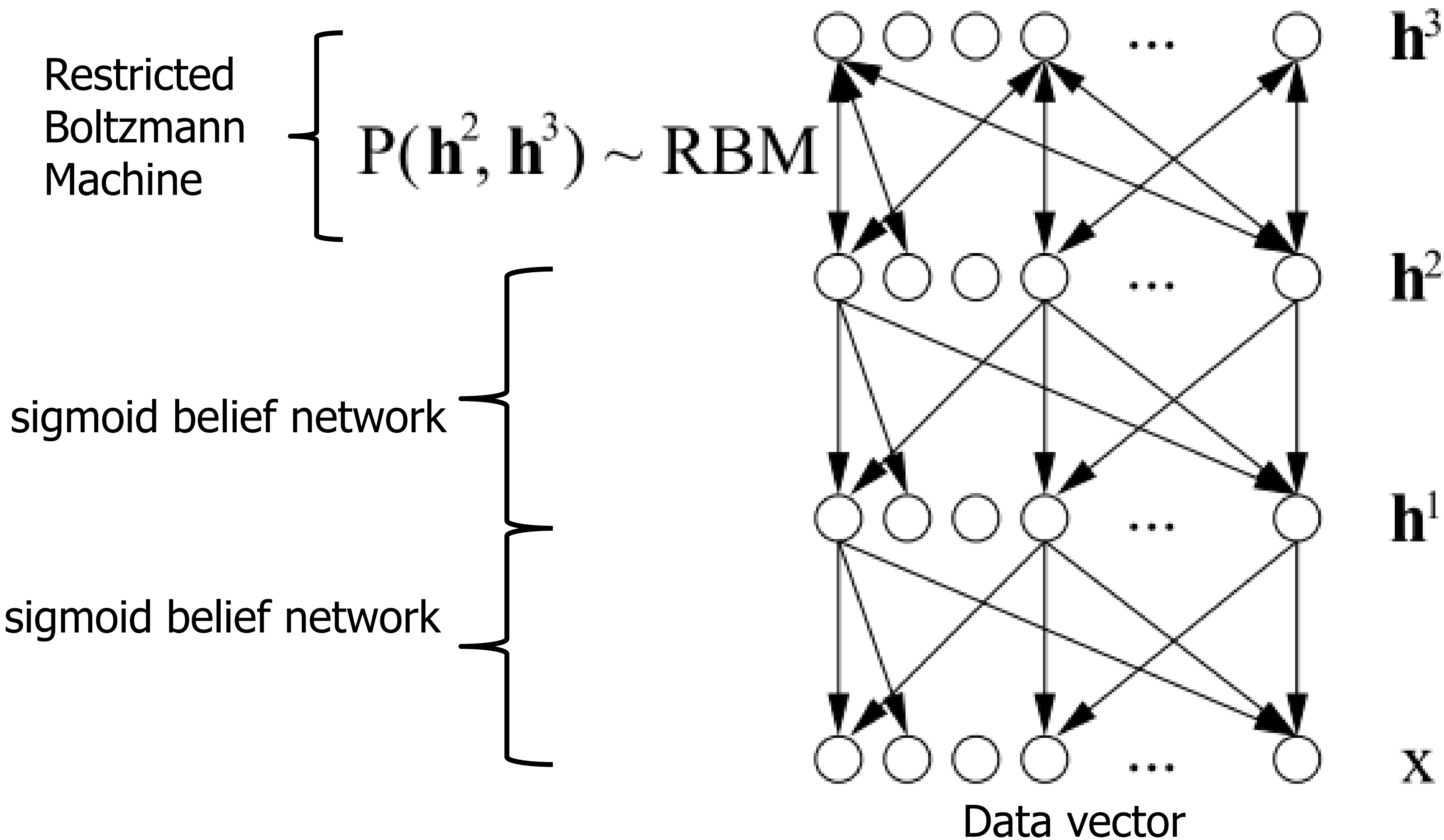
- It requires labeled training data.
  - Almost all data is unlabeled.
- The learning time does not scale well.
  - It is very slow in networks with multiple hidden layers.
- It can get stuck in poor local optima.
  - Usually in deep nets they are far from optimal.
- MLP is not a generative model, it only focuses on P(Y|X).
  We would like a generative approach that could learn P(X) as well.
- **Solution**: ***Deep Belief Networks***, a generative graphical model

Deep Belief Network

Deep Belief Networks (DBN's)

- are probabilistic generative models
- contain many layers of hidden variables
- each layer captures high-order correlations between the activities of hidden features in the layer below
- the top two layers of the DBN form an undirected bipartite graph called Restricted Boltzmann Machine
- the lower layers forming a directed sigmoid belief network

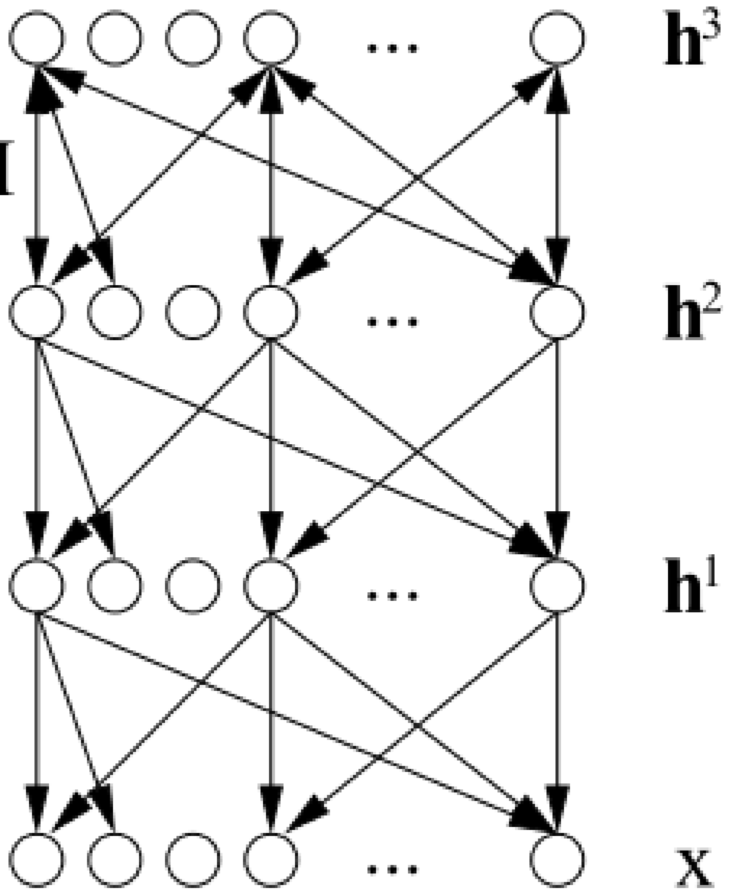
Deep Belief Network



Deep Belief Network

$$P(\mathbf{h}^l, \mathbf{h}^{l-1}) \propto \exp(\mathbf{b}^T \mathbf{h}^{l-1} + \mathbf{c}^T \mathbf{h}^l + \mathbf{h}^{l^T} W \mathbf{h}^{l-1})$$

$$P(\mathbf{h}^2, \mathbf{h}^3) \sim \text{RBM}$$



$$\sigma(x) = (1 + e^{-x})^{-1}$$

$$P(h_i^k = 1 | \mathbf{h}^{k+1}) = \sigma(b_i^{k+1} + \sum_j W_{i,j}^{k+1} h_j^{k+1})$$

$$P(x_i = 1 | \mathbf{h}^1) = \sigma(b_i^1 + \sum_j W_{i,j}^1 h_j^1)$$

Joint likelihood:

$$P(\mathbf{x}, \mathbf{h}^1, \dots, \mathbf{h}^l) = P(\mathbf{h}^l, \mathbf{h}^{l-1}) \left(\prod_{k=1}^{l-2} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \right) P(\mathbf{x} | \mathbf{h}^1)$$

Boltzmann Machines

Boltzmann Machines

Boltzmann machine: a network of symmetrically coupled stochastic binary units {0,1}

Parameters:

$\theta = \{\mathbf{W}, \mathbf{L}, \mathbf{J}\}$

W: visible-to-hidden

L: visible-to-visible, diag(L)=0

J: hidden-to-hidden, diag(J)=0

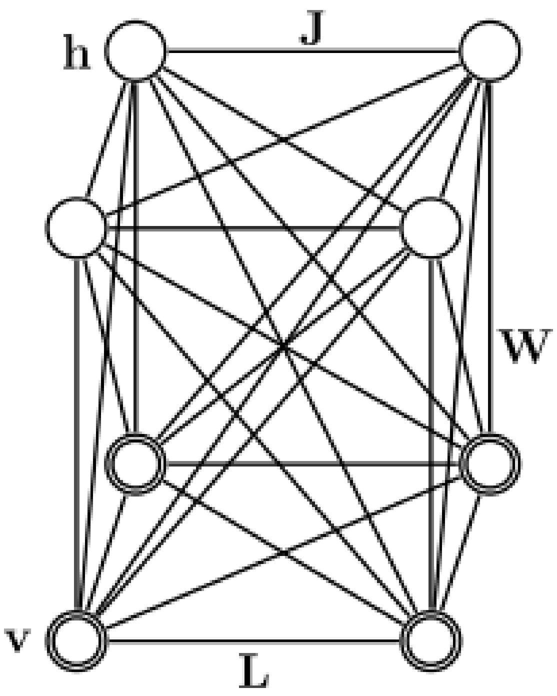
Hidden layer

$\mathbf{h} \in \{0,1\}^p$

Visible layer

$\mathbf{v} \in \{0,1\}^d$

General Boltzmann Machine



Energy of the Boltzmann machine:

$$E(\mathrm{v}, \mathrm{h}|\theta) = -\frac{1}{2}\mathrm{v}^T\mathbf{L}\mathrm{v} - \frac{1}{2}\mathrm{h}^T\mathbf{J}\mathrm{h} - \mathrm{v}^T\mathbf{W}\mathrm{h}$$

Boltzmann Machines

Energy of the Boltzmann machine:

$$E(\mathrm{v}, \mathrm{h}|\theta) = -\frac{1}{2}\mathrm{v}^T \mathbf{L}\mathrm{v} - \frac{1}{2}\mathrm{h}^T \mathbf{J}\mathrm{h} - \mathrm{v}^T \mathbf{W}\mathrm{h}$$

Generative model:

Joint likelihood: $P(\mathbf{v}, \mathbf{h}|\theta) \propto \exp(-E(\mathbf{v}, \mathbf{h}; \theta))$

Probability of a visible vector v:

$$P(\mathbf{v}|\theta) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}|\theta)$$

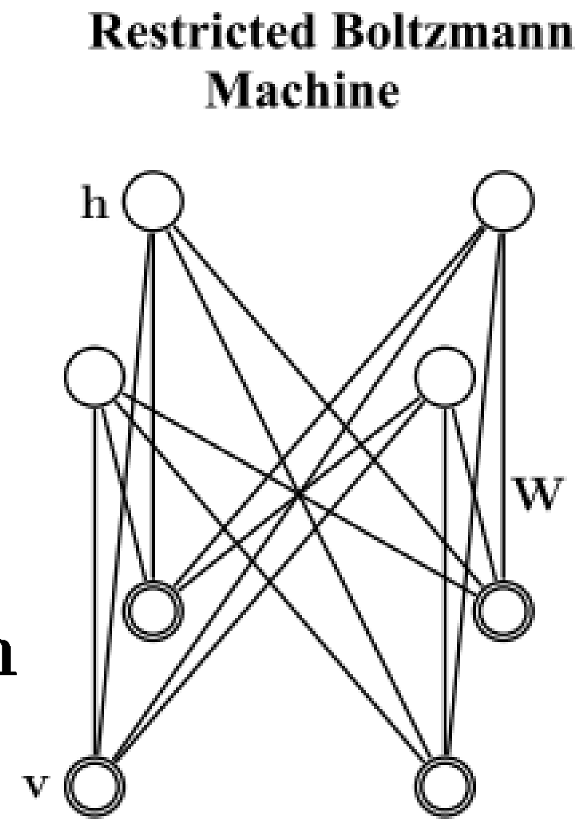
Exponentially large set

$$= \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}; \theta))}{Z(\theta)}$$

$$Z(\theta) = \sum_{\mathrm{v}} \sum_{\mathrm{h}} \exp(-E(\mathrm{v}, \mathrm{h}; \theta))$$

Restricted Boltzmann Machines

No hidden-to-hidden and no visible-to-visible connections.



W: visible-to-hidden

L = 0: visible-to-visible

J = 0: hidden-to-hidden

Hidden layer

Visible layer

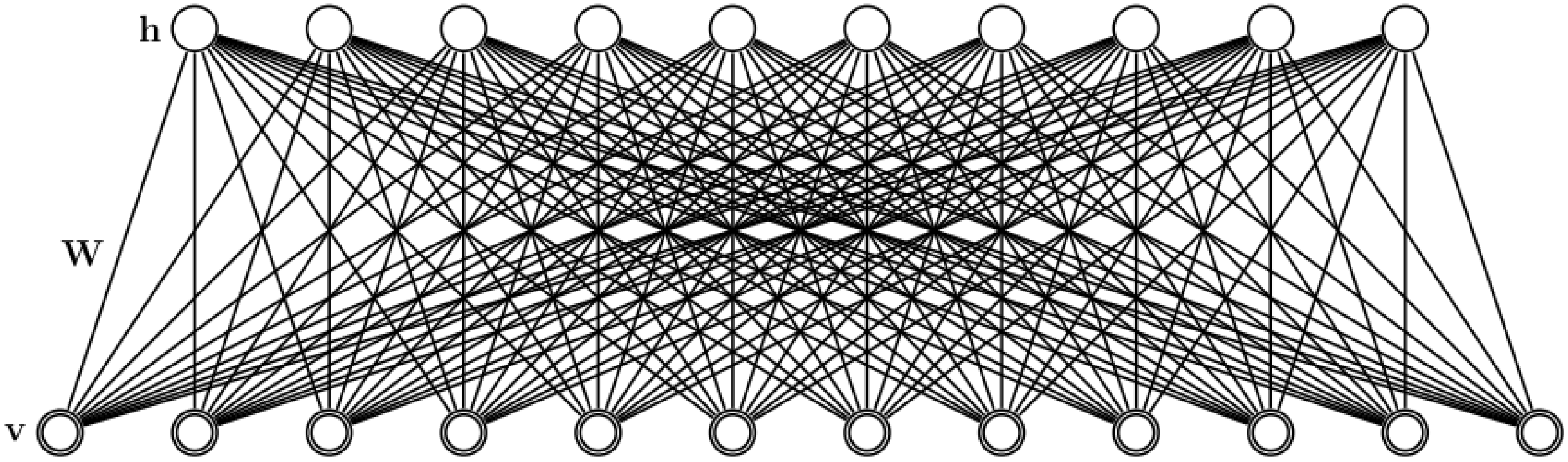
Energy of RBM:

$$E(\mathbf{v},\mathbf{h}|\theta) = -\mathbf{v}^T W \mathbf{h} - \mathbf{b}^T \mathbf{v} - \mathbf{a}^T \mathbf{h}$$

Joint likelihood:

$$P(\mathbf{v},\mathbf{h}|\theta) = \frac{1}{Z(\theta)} \exp(-E(\mathbf{v},\mathbf{h};\theta))$$

Restricted Boltzmann Machines



Top layer: vector of stochastic binary hidden units **h**
Bottom layer: a vector of stochastic binary visible variables **v**.

Figure is taken from R. Salakhutdinov

Training RBM

Due to the special bipartite structure of RBM's, the hidden units can be explicitly marginalized out:

$$P(\mathbf{v};\theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h};\theta)\right).$$

$$\begin{aligned} P(\mathbf{v};\theta) &= \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(\mathbf{v}^\top W \mathbf{h} + \mathbf{b}^\top \mathbf{v} + \mathbf{a}^\top \mathbf{h}\right) \\ &= \frac{1}{\mathcal{Z}(\theta)} \exp(\mathbf{b}^\top \mathbf{v}) \prod_{j=1}^{F} \sum_{h_j \in \{0,1\}} \exp\left(a_j h_j + \sum_{i=1}^{D} W_{ij} v_i h_j\right) \\ &= \frac{1}{\mathcal{Z}(\theta)} \exp(\mathbf{b}^\top \mathbf{v}) \prod_{j=1}^{F} \left(1 + \exp\left(a_j + \sum_{i=1}^{D} W_{ij} v_i\right)\right). \end{aligned}$$

Training RBM

$$P(\mathbf{v};\theta) = \frac{1}{\mathcal{Z}(\theta)} \exp(\mathbf{b}^\top \mathbf{v}) \prod_{j=1}^{F} \left(1 + \exp\left(a_j + \sum_{i=1}^{D} W_{ij} v_i\right)\right)$$

Gradient descent:

$$\frac{\partial \log P(\mathbf{v};\theta)}{\partial W} = \mathrm{E}_{P_{\text{data}}}[\mathbf{v}\mathbf{h}^\top] - \mathrm{E}_{P_{\text{Model}}}[\mathbf{v}\mathbf{h}^\top],$$

$$\frac{\partial \log P(\mathbf{v};\theta)}{\partial \mathbf{a}} = \mathrm{E}_{P_{\text{data}}}[\mathbf{h}] - \mathrm{E}_{P_{\text{Model}}}[\mathbf{h}],$$

$$\frac{\partial \log P(\mathbf{v};\theta)}{\partial \mathbf{b}} = \mathrm{E}_{P_{\text{data}}}[\mathbf{v}] - \mathrm{E}_{P_{\text{Model}}}[\mathbf{v}].$$

The exact calculations are intractable because the expectation operator in E_P_Model takes exponential time in min(D,F)

Efficient Gibbs sampling based approximation exists (Contrastive divergence)

Inference in RBM

Inference is simple in RBM:

$$P(\mathbf{h}|\mathbf{v};\theta) = \prod_j p(h_j|\mathbf{v}), \quad P(\mathbf{v}|\mathbf{h};\theta) = \prod_i p(v_i|\mathbf{h}),$$

$$p(h_j = 1|\mathbf{v}) = g\left(\sum_i W_{ij} v_i + a_j\right),$$

$$p(v_i = 1|\mathbf{h}) = g\left(\sum_j W_{ij} h_j + b_i\right),$$

where $g(x) = 1/(1+\exp(-x))$ is the logistic function.

Training Deep Belief Networks

Training Deep Belief Networks

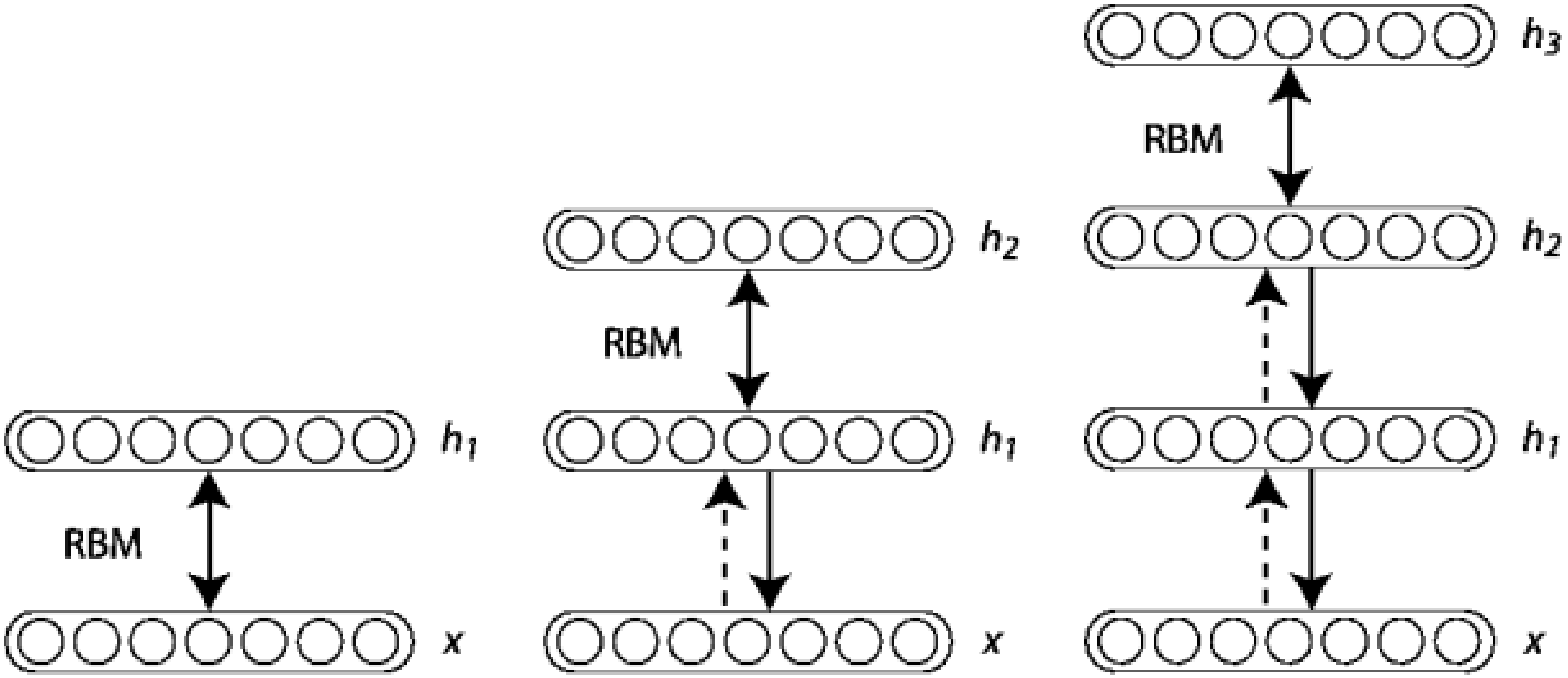
Greedy layer-wise unsupervised learning:

Much better results could be achieved when pre-training each layer with an unsupervised learning algorithm, one layer after the other, starting with the first layer (that directly takes in the observed x as input).

- The initial experiments used the RBM generative model for each layer.
- Later variants: auto-encoders for training each layer (Bengio et al., 2007; Ranzato et al., 2007; Vincent et al., 2008
- After having initialized a number of layers, the whole neural network can be fine-tuned with respect to a supervised training criterion as usual

Training Deep Belief Networks

The unsupervised greedy layer-wise training serves as **initialization**, **replacing** the traditional **random initialization** of multi-layer networks.

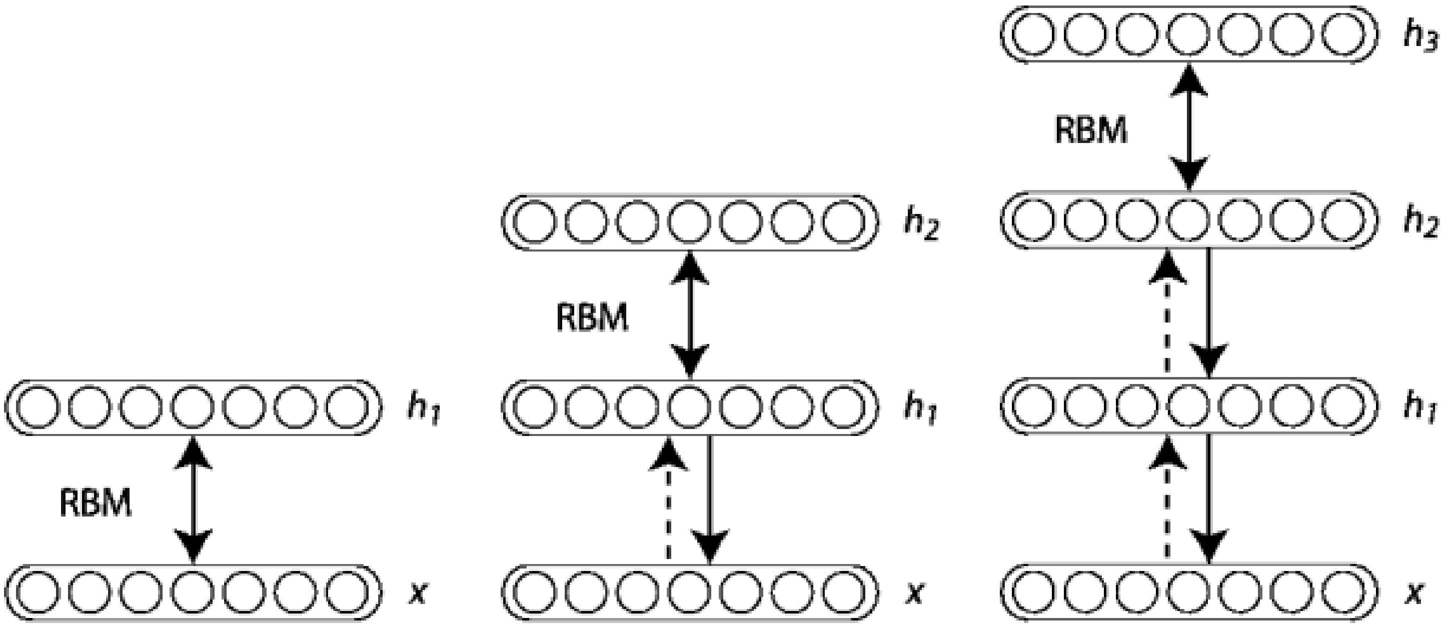


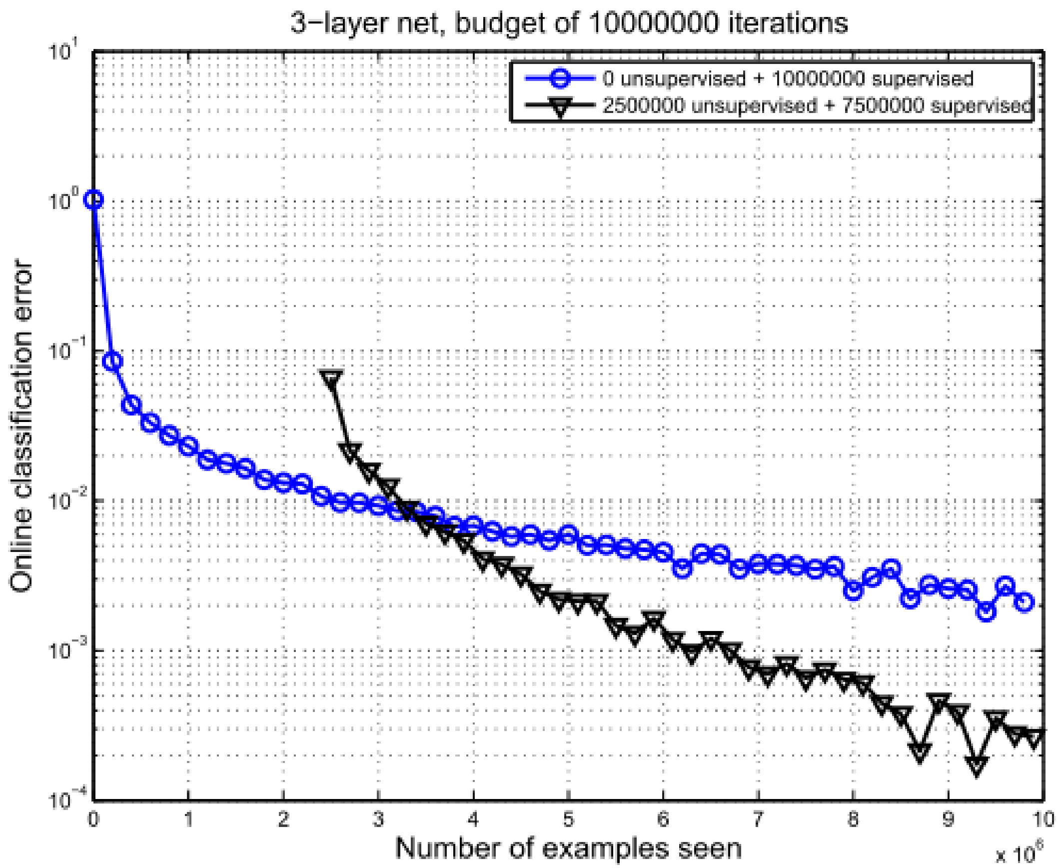
Data

Training Deep Belief Networks

Algorithm 1 Recursive Greedy Learning Procedure for the DBN.

1: Fit parameters W^1 of the 1^{st} layer RBM to data.
2: Freeze the parameter vector W^1 and use samples $\mathbf{h}^1$ from $Q(\mathbf{h}^1|\mathbf{v}) = P(\mathbf{h}^1|\mathbf{v}, W^1)$ as the data for training the next layer of binary features with an RBM.
3: Freeze the parameters W^2 that define the 2^{nd} layer of features and use the samples $\mathbf{h}^2$ from $Q(\mathbf{h}^2|\mathbf{h}^1) = P(\mathbf{h}^2|\mathbf{h}^1, W^2)$ as the data for training the 3^{rd} layer of binary features.
4: Proceed recursively for the next layers.



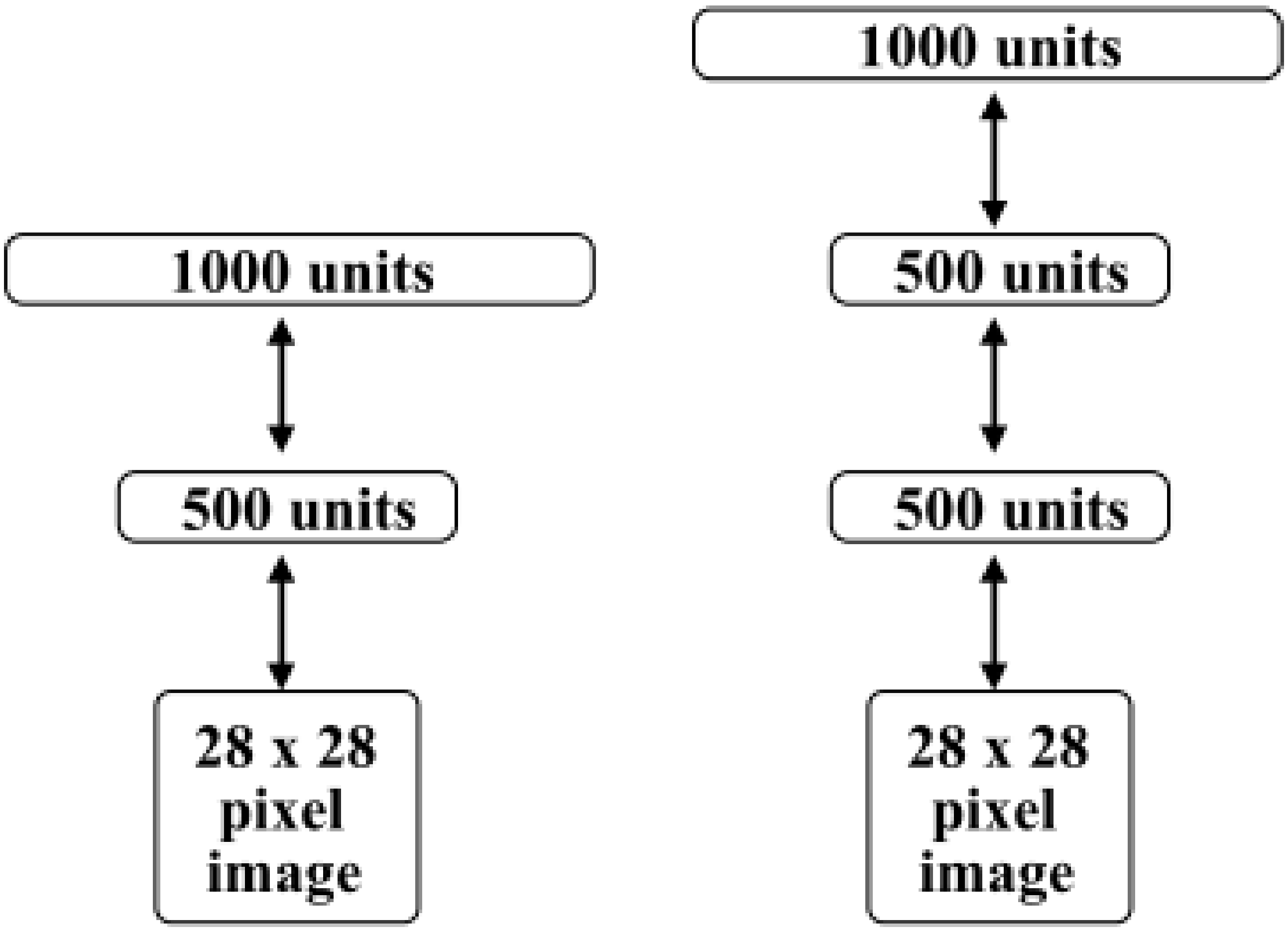


- Deep architecture trained online with 10 million examples of digit images, either with pre-training (triangles) or without (circles).
- The first 2.5 million examples are used for unsupervised pre-training.
- One can see that without pre-training, training converges to a poorer apparent local minimum: unsupervised pre-training helps to find a better minimum of the online error.

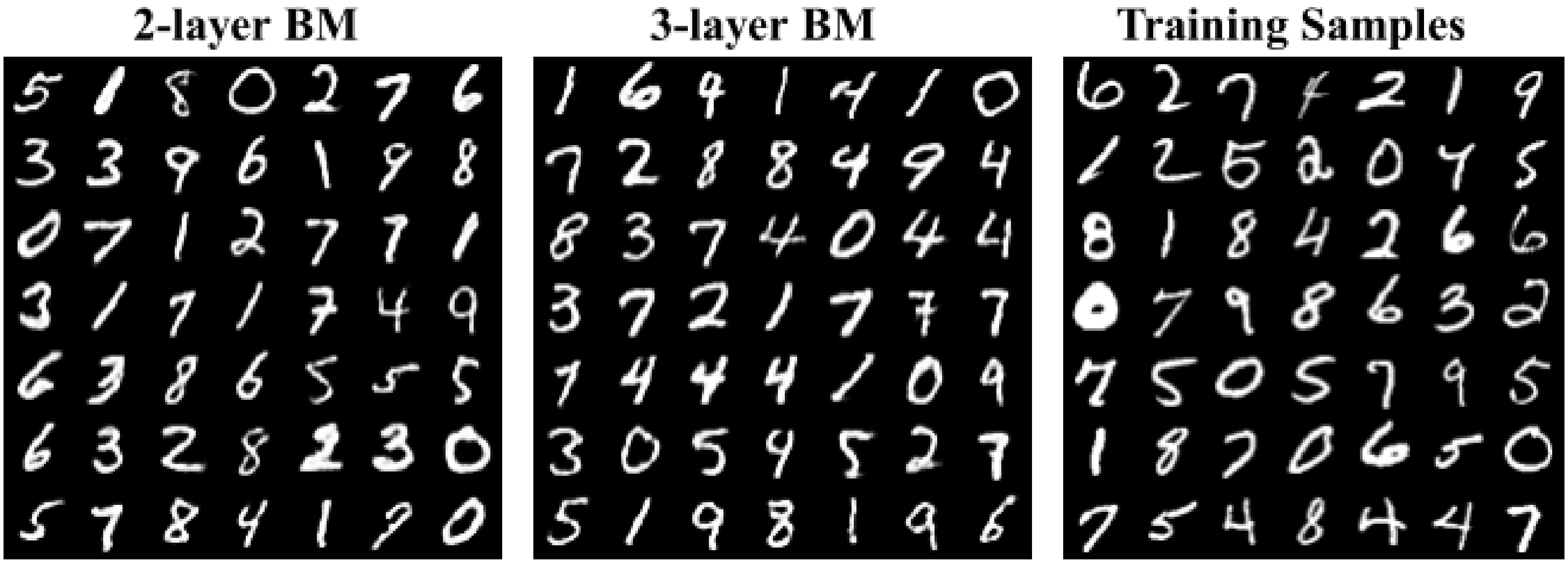
Experiments performed by Dumitru Erhan.

Results

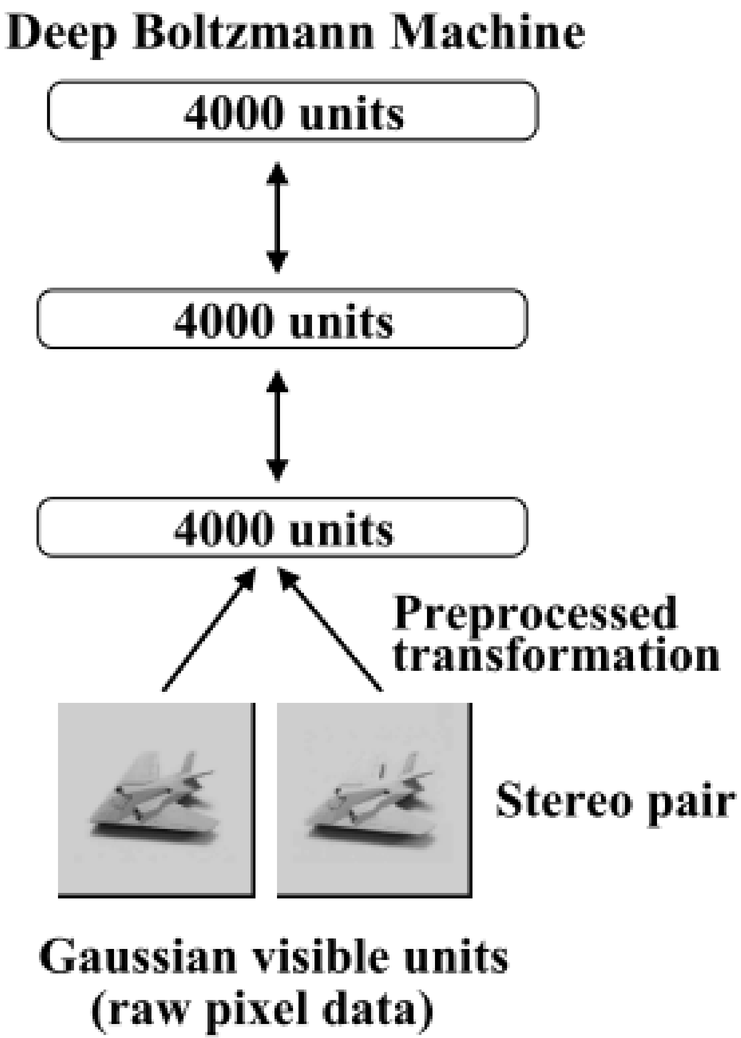
Deep Boltzmann Machines Results



Deep Boltzmann Machines Results

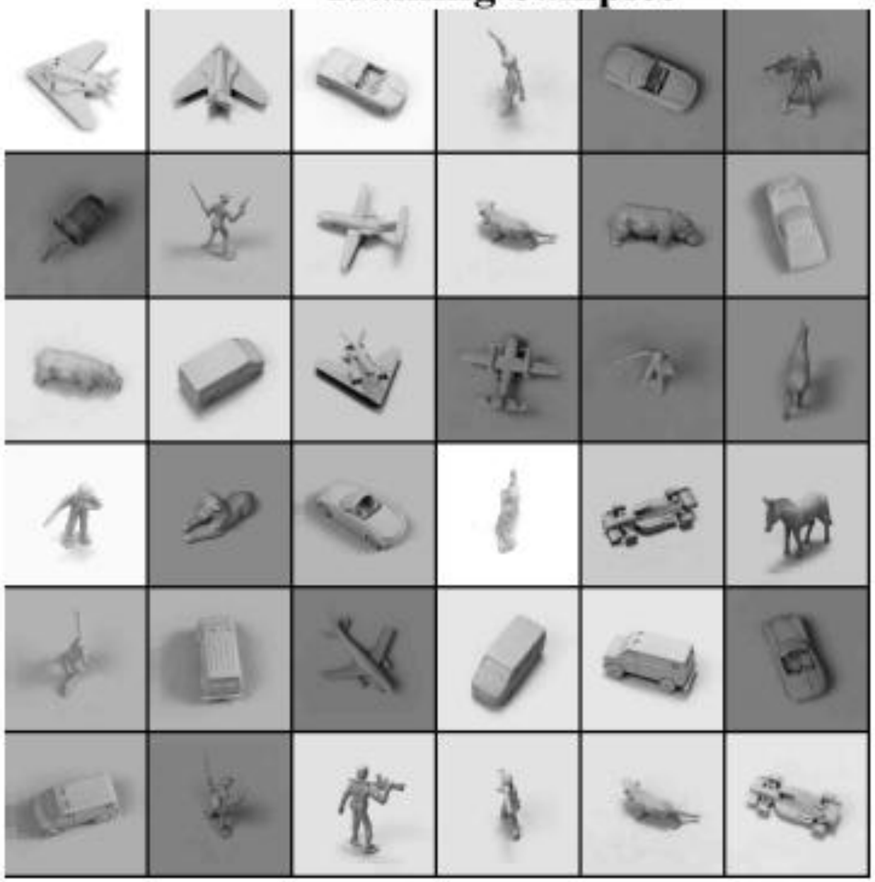


Deep Boltzmann Machines Results

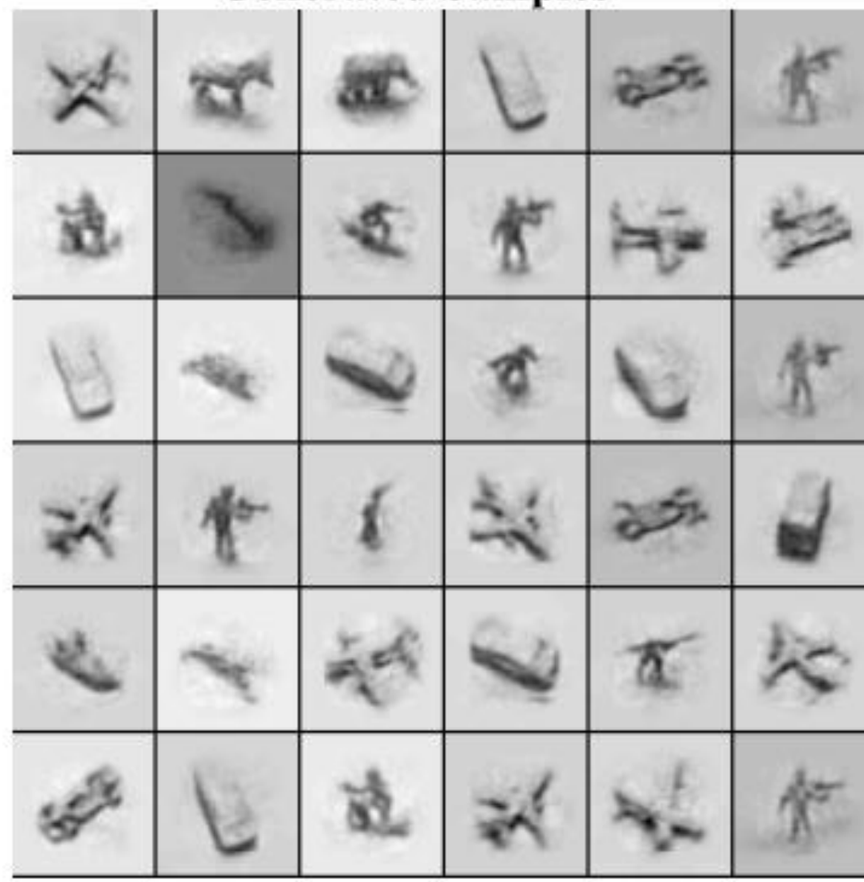


Deep Boltzmann Machines Results

Training Samples



Generated Samples



Thanks for your Attention! ☺