Lecture 4:

Deep Neural Networks

Tomas Kulvicius

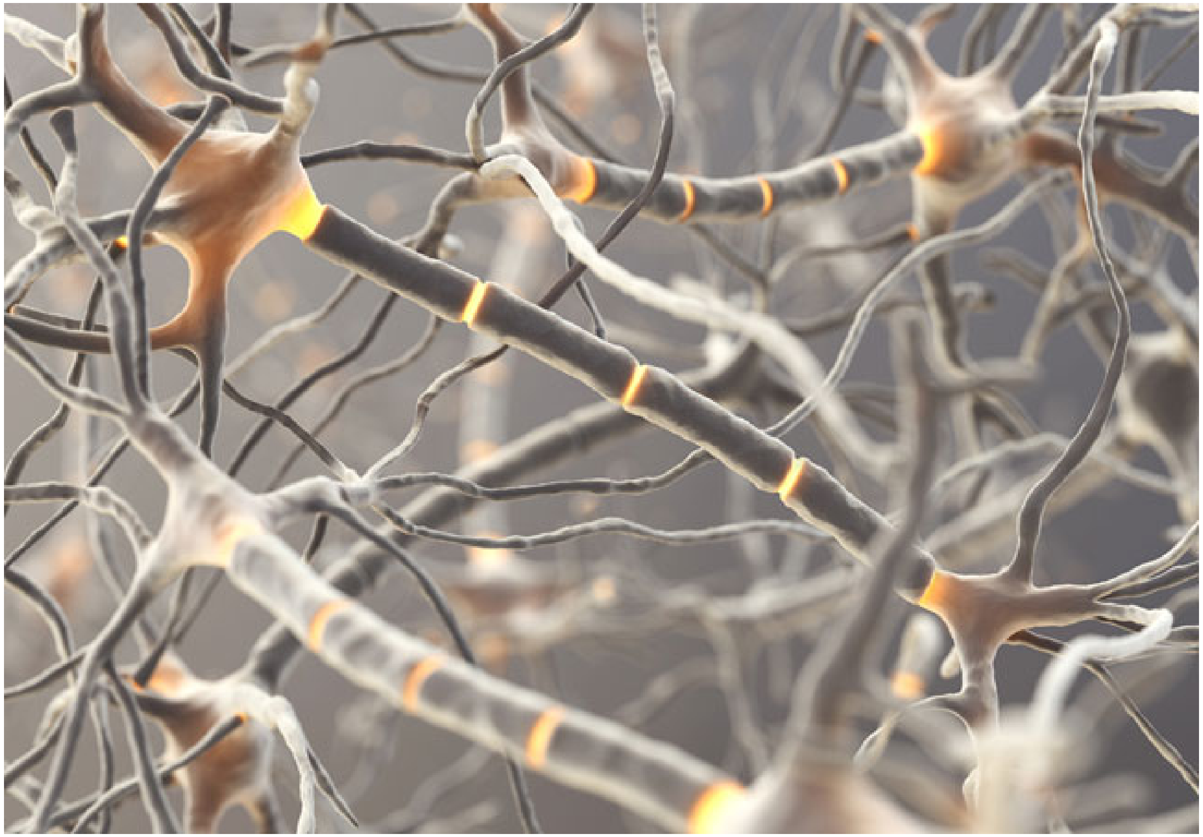
Large portion of slides are adopted from Nasrullah Memon, Florentin Wörgötter, Honglak Lee, Geoffrey Hinton, Yann LeCun and Marc'Aurelio Ranzato

Outline of the Course

- Introduction to AI and Learning Algorithms (week 5)
- Artificial Neural Networks and Perceptrons (week 6)
- Multilayer Perceptrons (week 7)
- Reexams – no lecture (week 8)
- Deep Learning (week 9)
- Genetic Algorithms (week 10; 09.03)
- Self-Organizing Maps (week 10; 10.03)
- Correlation-based Learning (week 11)
- Easter holiday – no lecture (week 12)
- Reinforcement Learning I (week 13)
- Reinforcement Learning II (week 14)
- Working on Assignments (week 15 onward)

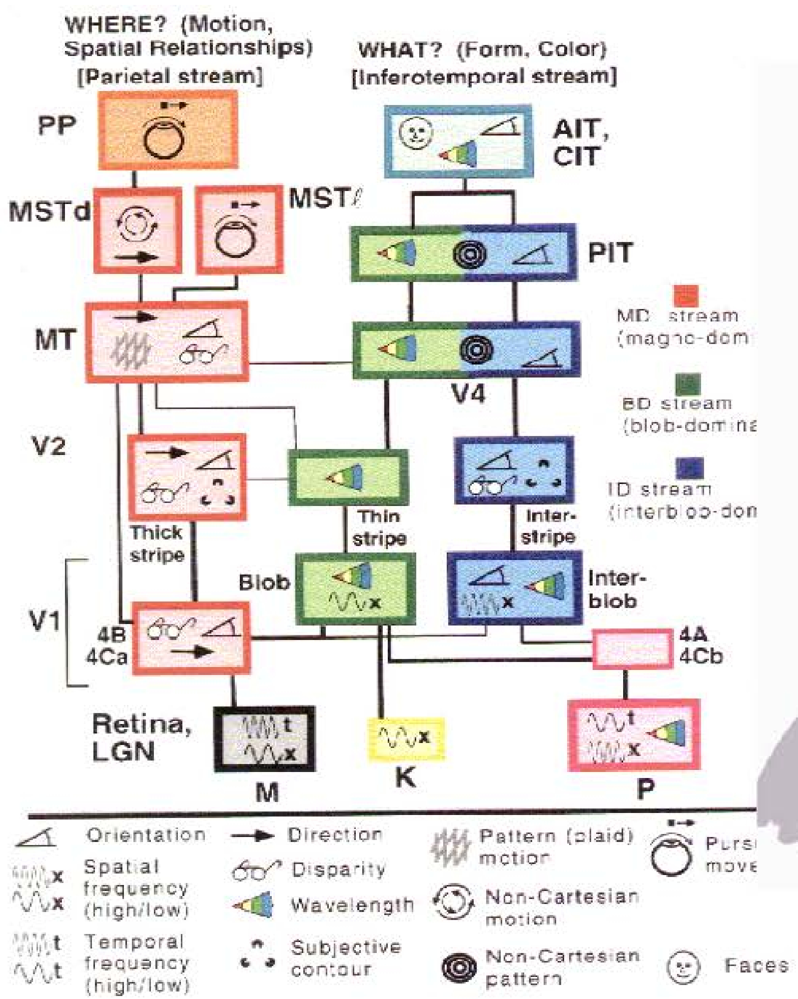
References

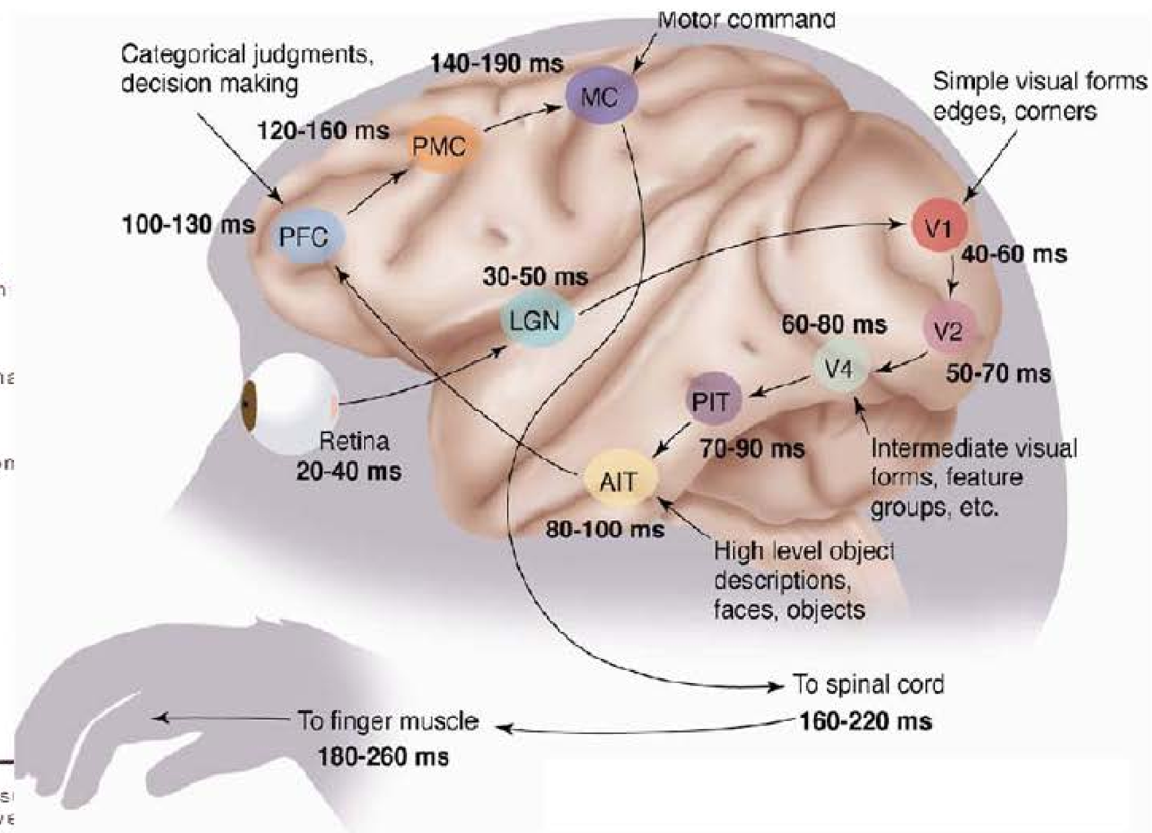
- http://deeplearning.net/
- Deep Learning, Yoshua Bengio, Ian Goodfellow, Aaron Courville, MIT Press, In preparation (http://www.deeplearningbook.org/)



Motivation: Mammalian Visual Cortex

- The recognition pathway in the visual cortex has multiple stages

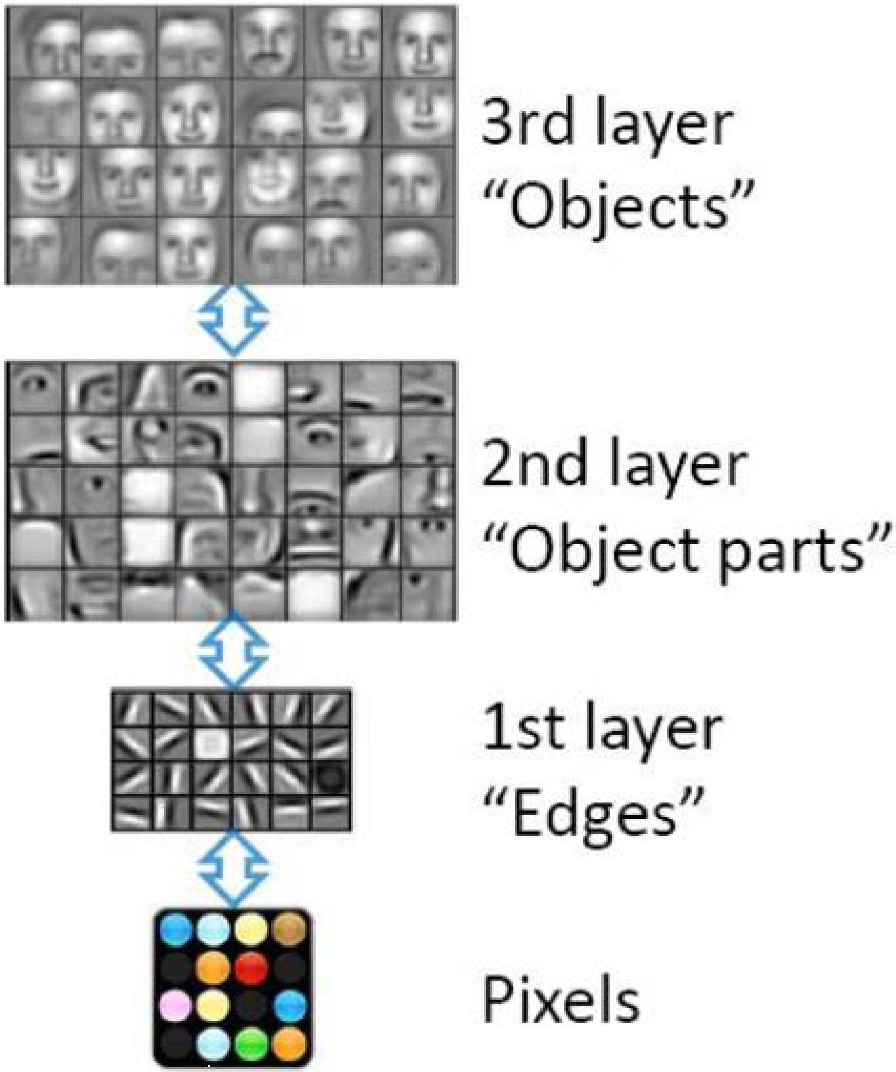




[Gallant & Van Essen]

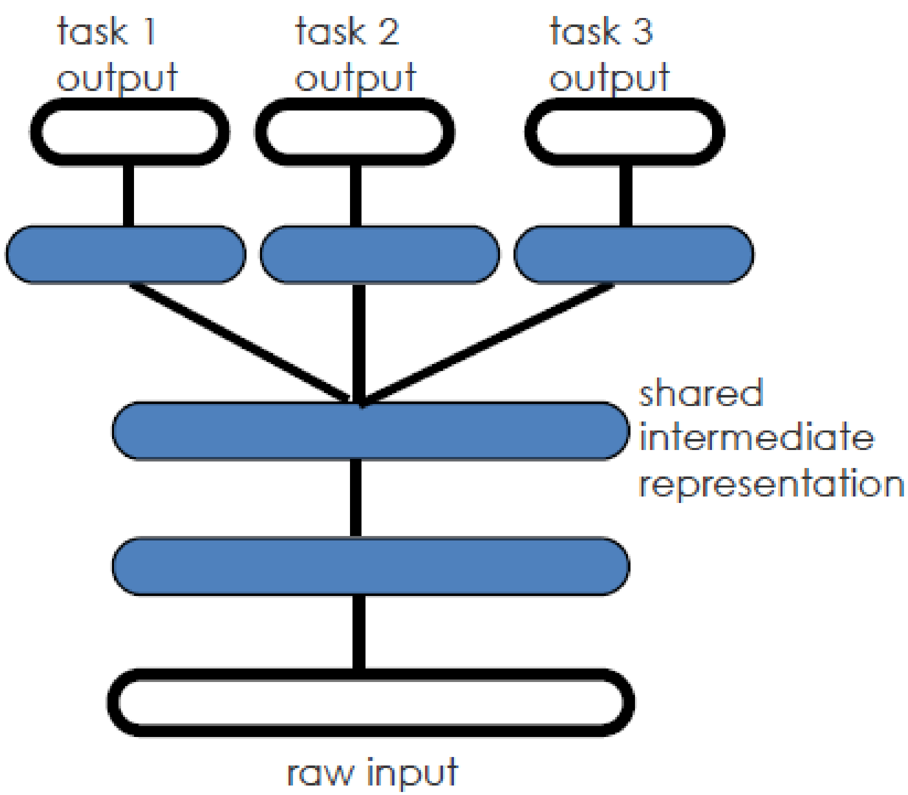
Learning Hierarchical Representations

- Natural progression from low level to high level structures
- Each module transforms its input representation into a higher-level one
- High-level features are more global and more invariant
- Low-level features are shared among categories
- A good lower level representation can be used for many distinct tasks

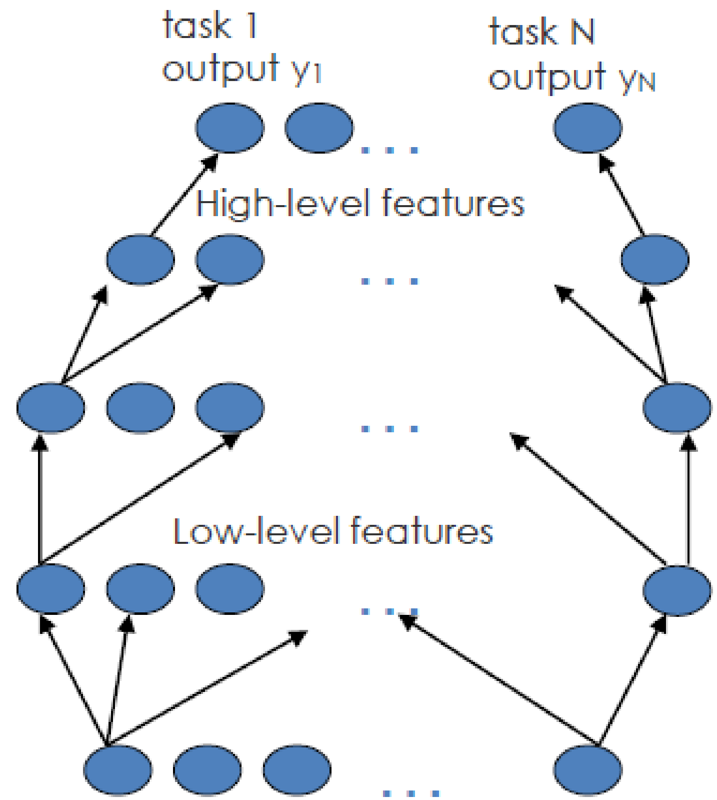


Generalisable Learning

Shared low level representations



Partial feature sharing



Examples of Hierarchical Representations

- **Hierarchy of representations with increasing level of abstraction**
- **Each stage is a kind of trainable feature transform**
- **Image recognition**
  - Pixel → edge → texton → motif → part → object
- **Text**
  - Character → word → word group → clause → sentence → story
- **Speech**
  - Sample → spectral band → sound → … → phone → phoneme →word

Learning Hierachical Representations

- **Purely Supervised**
  - Initialize parameters randomly
  - Train in supervised mode using backpropagation
  - Used in most practical systems for speech and image recognition

Learning Hierachical Representations (cont.)

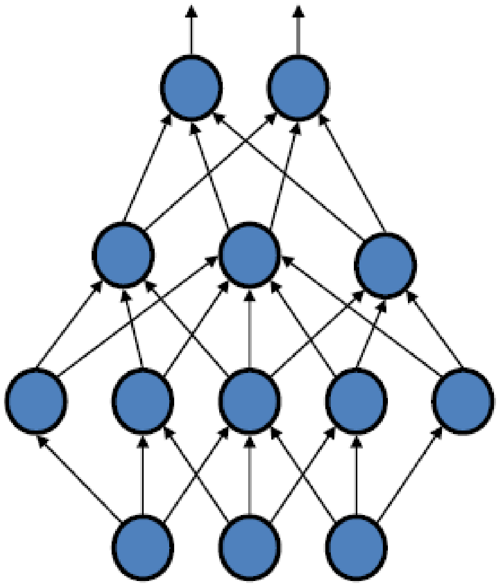
- **Unsupervised, layerwise + supervised classifier on top**
  - Train each layer unsupervised, one after the other
  - Train a supervised classifier on top, keeping the other layers fixed
  - Good when very few labeled samples are available

Learning Hierachical Representations (cont.)

- **Unsupervised, layerwise + global supervised fine-tuning**
  - Train each layer unsupervised, one after the other
  - Add a classifier layer, and retrain the whole network supervised
  - Good when label set is poor (e.g., pedestrian detection)

Deep Neural Networks and Backpropagation

- **Simple to construct**
  - Sigmoid nonlinearity for hidden layers
  - Softmax for the output layer



- **But, backpropagation does not work well (if randomly initialized)**
  - Deep networks trained with backpropagation (no pretraining) perform worse than shallow nets

	train.	valid.	test
DBN, unsupervised pre-training	0%	1.2%	1.2%
Deep net, auto-associator pre-training	0%	1.4%	1.4%
Deep net, supervised pre-training	0%	1.7%	2.0%
Deep net, no pre-training	.004%	2.1%	2.4%
Shallow net, no pre-training	.004%	1.8%	1.9%

Bengio et al., NIPS 2007

Problems with Backpropagation

- **Gradient is progressively getting more dilute**
  - Below top few layers, correction signal is minimal
  - Gets stuck in local minima
  - Especially if they start out far from ‘good’ regions (i.e., random initialization)

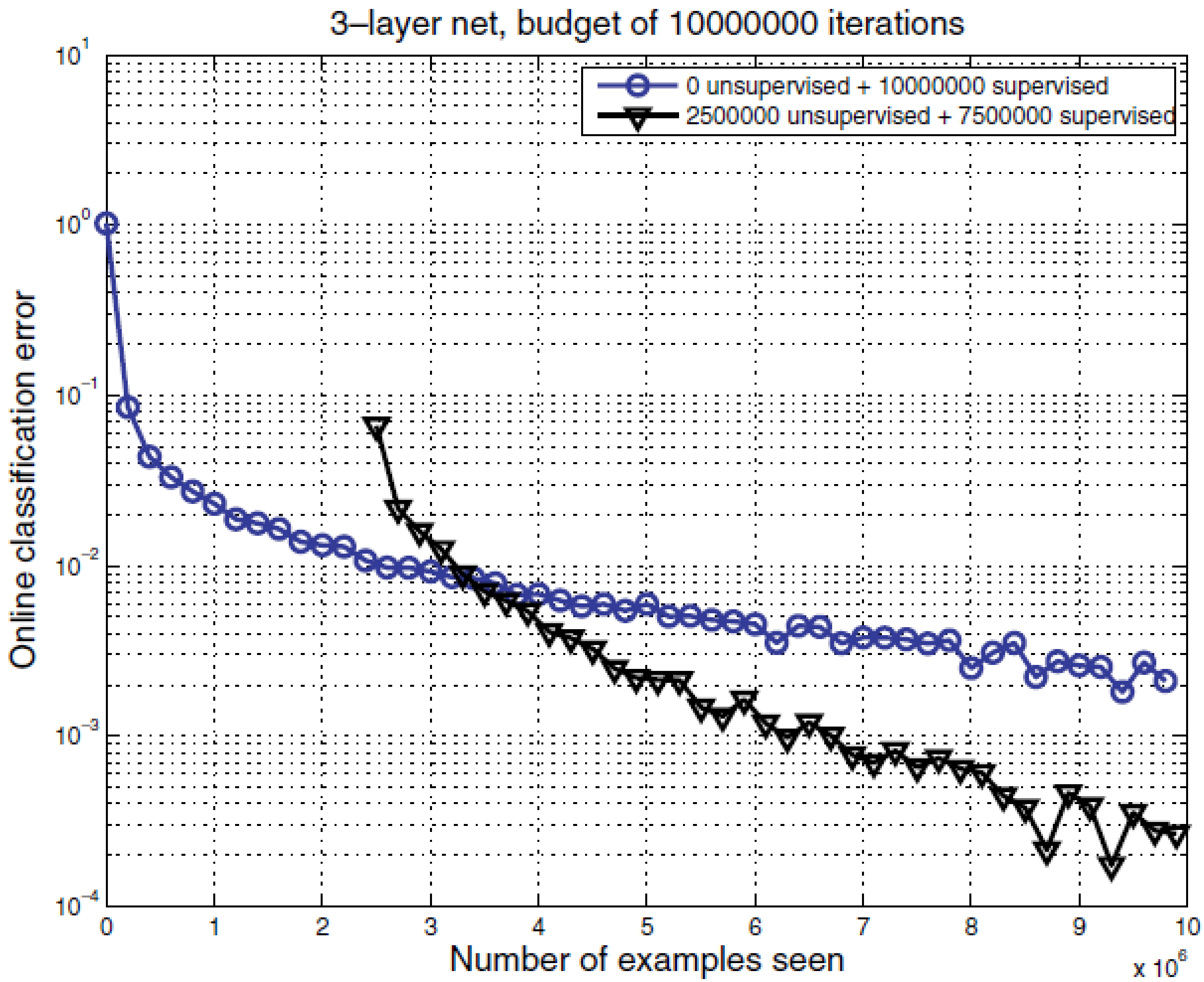
- **In usual settings, we can use only labeled data**
  - Almost all data is unlabeled
  - The brain can learn from unlabeled data

Deep Network Training

- **Use unsupervised learning (greedy layer-wise training)**
  - Allows abstraction to develop naturally from one layer to another
  - Help the network initialize with good parameters

- **Perform supervised top-down training as final step**
  - Refine the features (intermediate layers) so that they become more relevant for the task

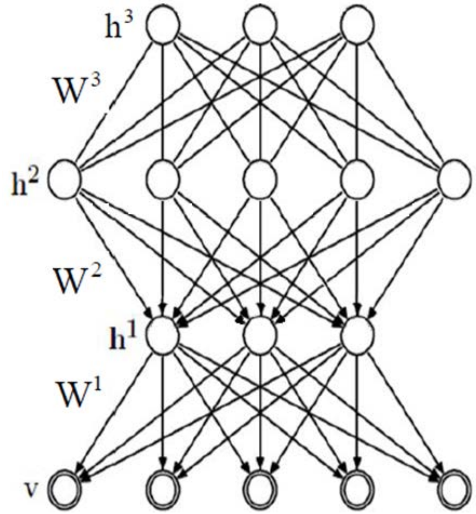
Deep Network Training (cont.)



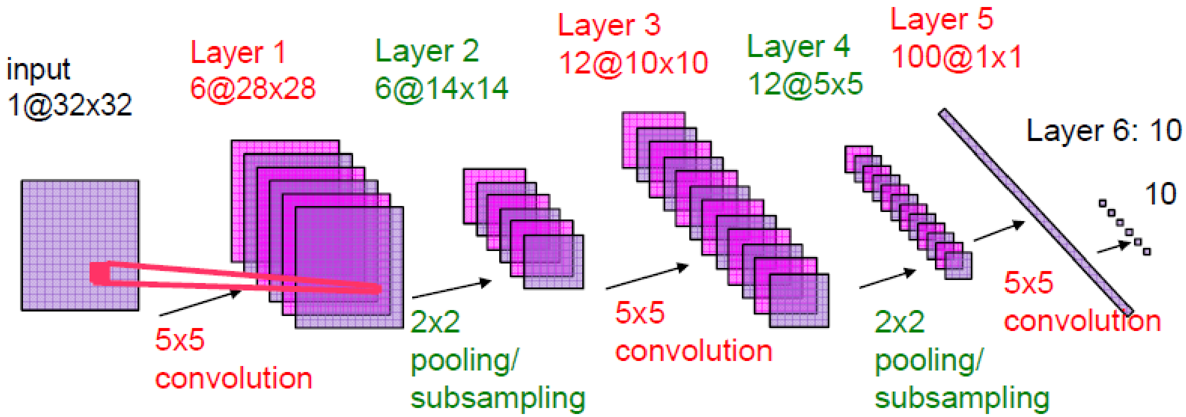
Bengio, 2009

Deep Neural Networks

- **Deep Believe Networks (DBNs)**



- **Convolutional Neural Networks (CNNs)**

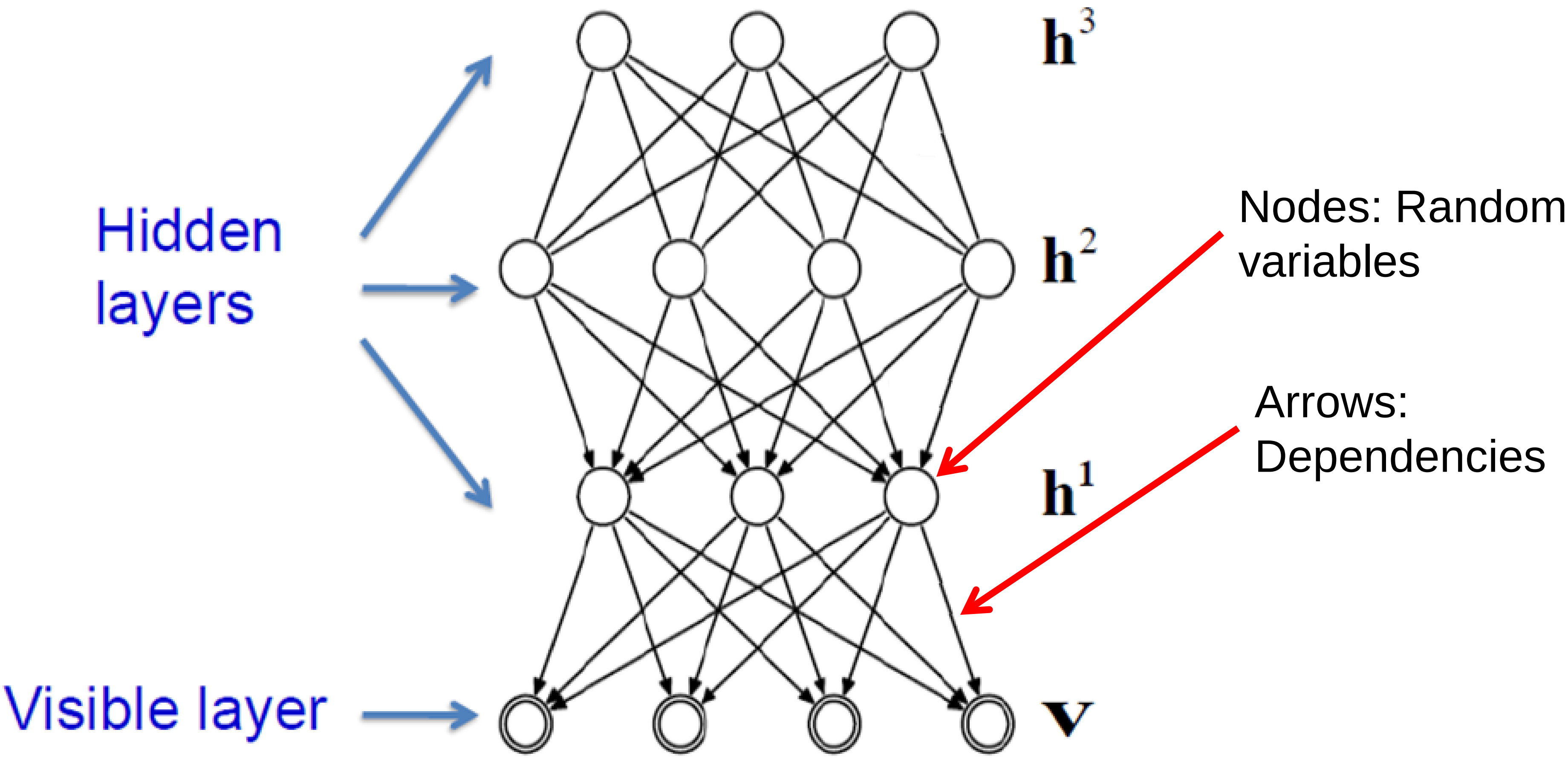


Deep Belief Networks (DBNs)

- **Deep architecture – multiple layers**
- **Unsupervised pre-learning provides a good initialization of the network**
- **Supervised fine-tuning**

Hinton et al., 2006

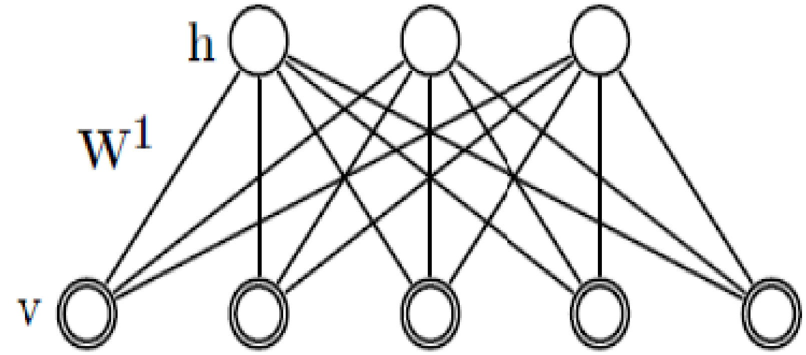
DBN Structure



Hinton et al., 2006

DBN Greedy Learning

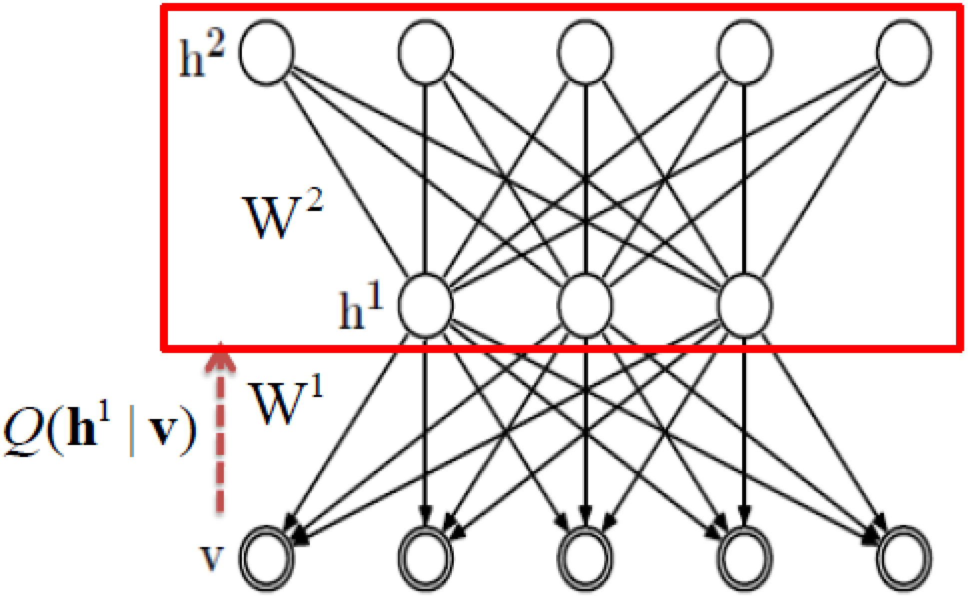
- **First step:**
  - Construct an Restricted Bolzmann Machine (RBM) with an input layer **v** and a hidden layer **h**
  - Train the RBM



Hinton et al., 2006

DBN Greedy Learning

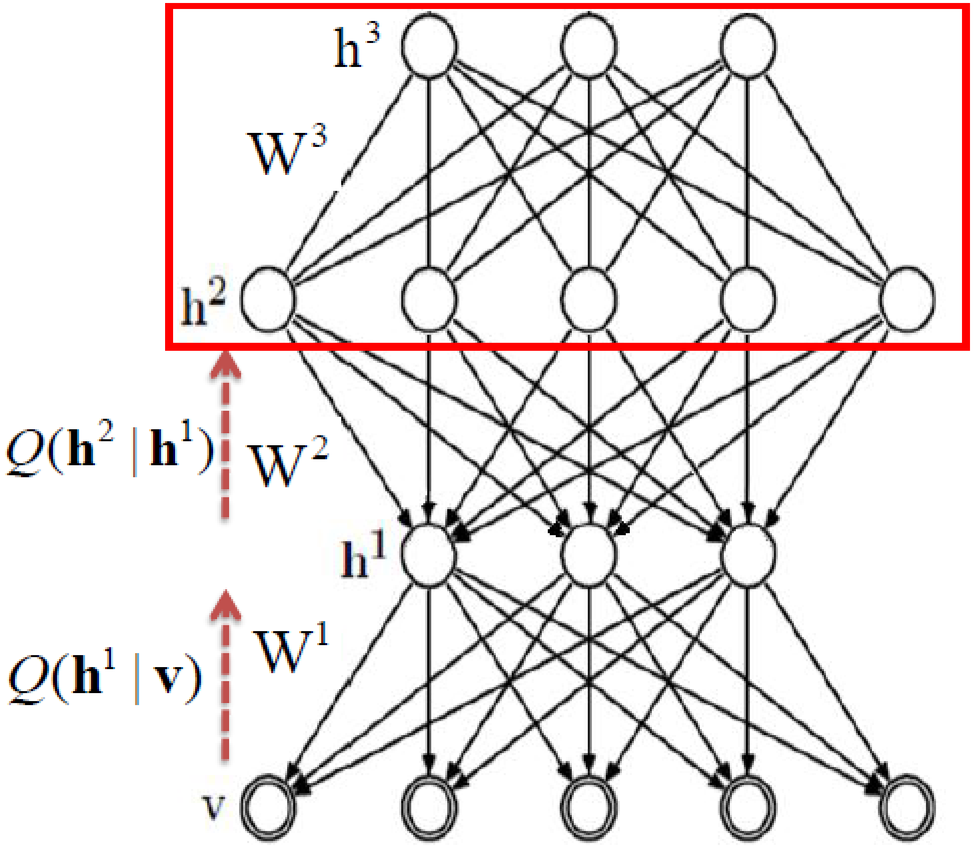
- **Second step:**
  - Stack another hidden layer on top of the RBM to form a new RBM
  - Fix $\mathbf{W^1}$, sample $\mathbf{h^1}$ from $\mathbf{Q(h^1|v)}$ as input. Train $\mathbf{W^2}$ as RBM



Hinton et al., 2006

DBN Greedy Learning

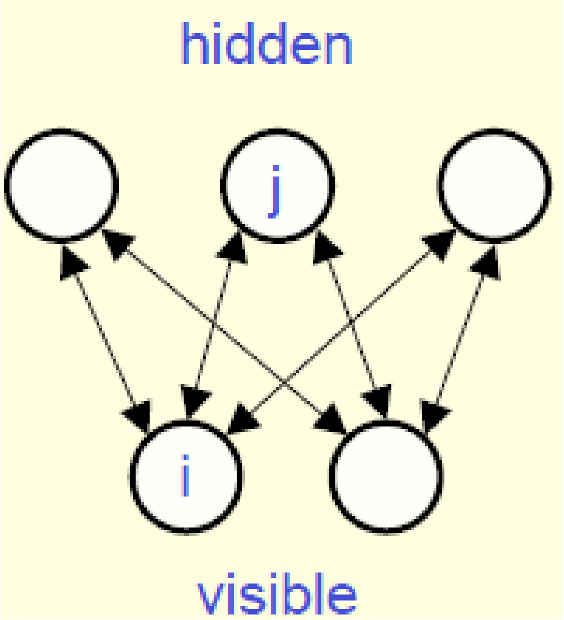
- **Third step:**
  - Continue to stack layers on top of the network, train it as previous step, with sample sampled from **$Q(h^2|h^1)$**
- **And so on …**



Hinton et al., 2006

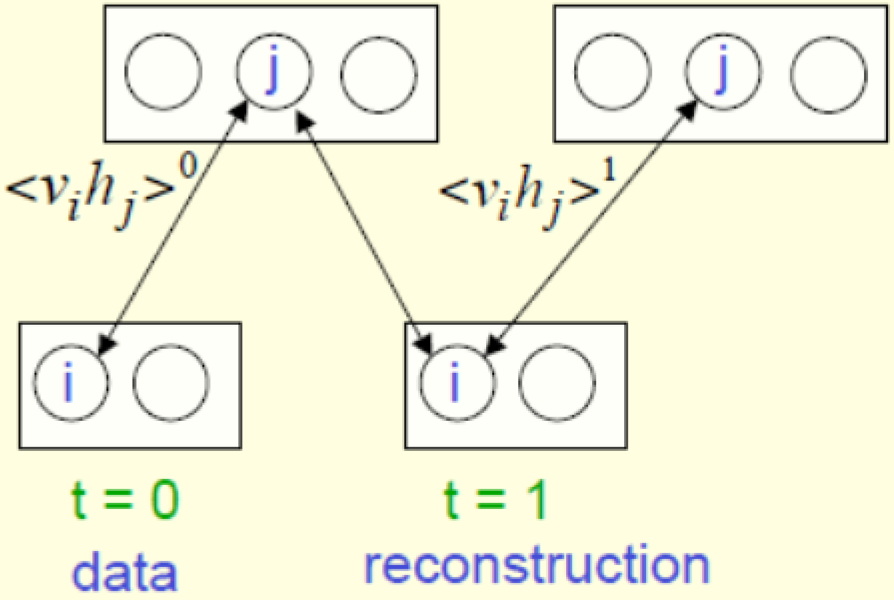
Restricted Bolzmann Machine

- **We restrict the connectivity to make learning easier**
  - Only one layer of hidden units
  - No connections between hidden units



Smolensky, 1986

Learning an RBM



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.

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

RBM Weight Update

Procedure

for all hidden units i **do**

- compute $Q(\mathbf{h}_{1i}=1|\mathbf{x}_1)$ (for binomial units, $\text{sigm}(\mathbf{c}_i+\sum_j W_{ij}\mathbf{x}_{1j})$)
- sample $\mathbf{h}_{1i} \in \{0,1\}$ from $Q(\mathbf{h}_{1i}|\mathbf{x}_1)$

end for

for all visible units j **do**

- compute $P(\mathbf{x}_{2j}=1|\mathbf{h}_1)$ (for binomial units, $\text{sigm}(\mathbf{b}_j+\sum_i W_{ij}\mathbf{h}_{1i})$)
- sample $\mathbf{x}_{2j} \in \{0,1\}$ from $P(\mathbf{x}_{2j}=1|\mathbf{h}_1)$

end for

for all hidden units i **do**

- compute $Q(\mathbf{h}_{2i}=1|\mathbf{x}_2)$ (for binomial units, $\text{sigm}(\mathbf{c}_i+\sum_j W_{ij}\mathbf{x}_{2j})$)

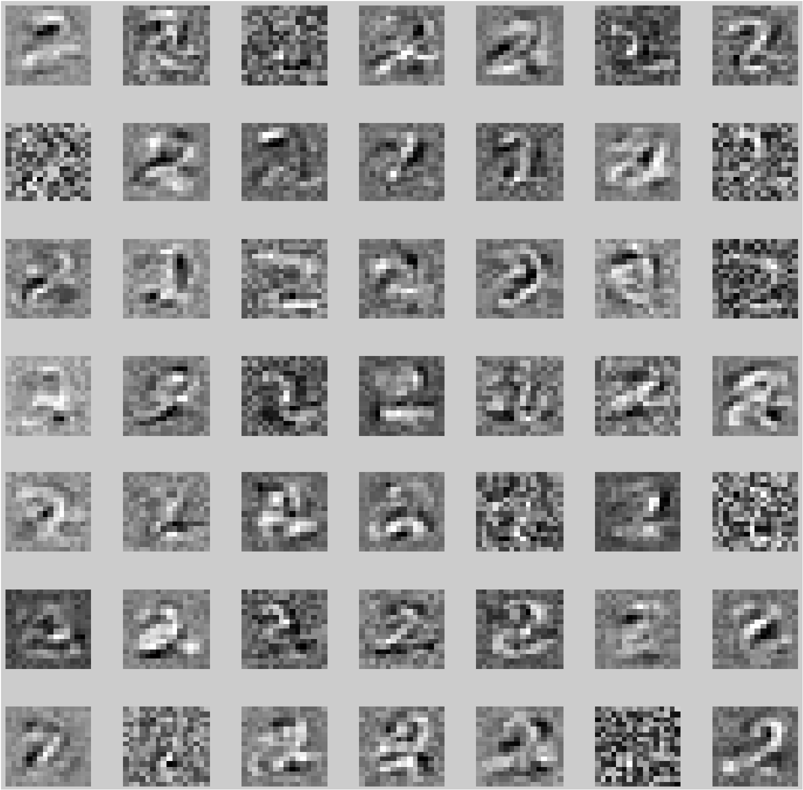
end for

- $W \leftarrow W + \epsilon(\mathbf{h}_1\mathbf{x}_1' - Q(\mathbf{h}_{2\cdot}=1|\mathbf{x}_2)\mathbf{x}_2')$
- $\mathbf{b} \leftarrow \mathbf{b} + \epsilon(\mathbf{x}_1 - \mathbf{x}_2)$
- $\mathbf{c} \leftarrow \mathbf{c} + \epsilon(\mathbf{h}_1 - Q(\mathbf{h}_{2\cdot}=1|\mathbf{x}_2))$

$\mathbf{x}_1$ is a sample
ϵ is a learning rate
W is the RBM weight matrix
$\mathbf{b}$ is the RBM offset vector for input units
$\mathbf{c}$ is the RBM offset vector for hidden units
$Q(\mathbf{h}_{2\cdot}=1|\mathbf{x}_2)$ is the vector with elements $Q(\mathbf{h}_{2i}=1|\mathbf{x}_2)$

Demo: RBM for Handwritten Digits

Weights 49 x 16x16 obtained from training on digit „2“ (16x16px)

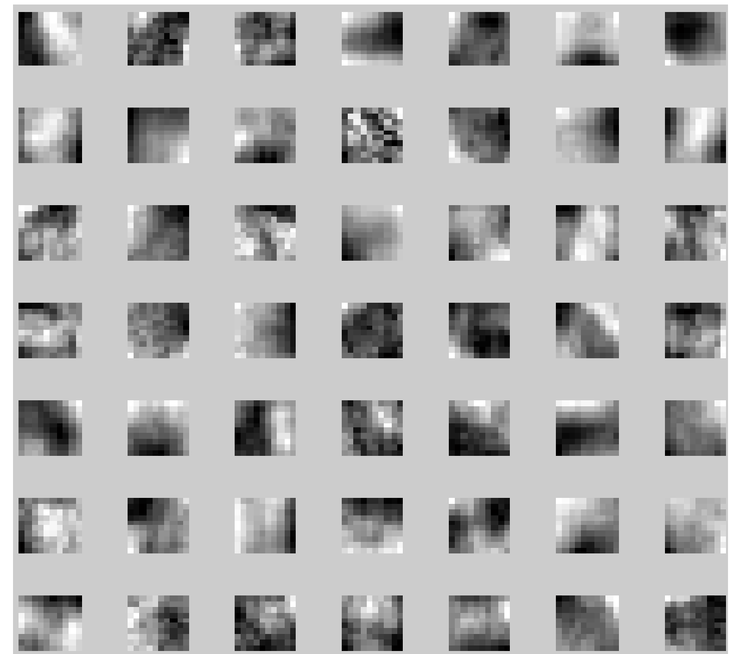


Demo: RBM for an Image

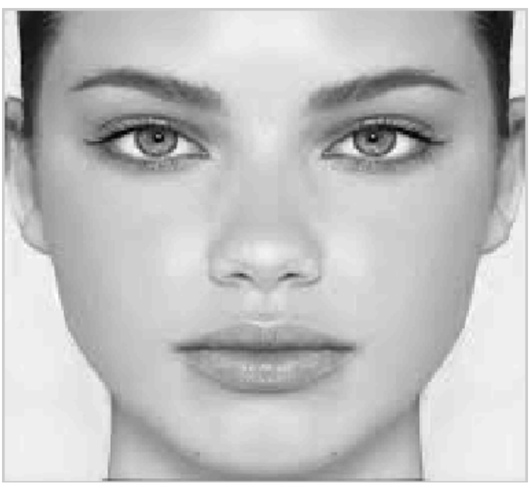
Original image

Reconstruction from binary features

Weights 49 x 10x10 obtained from training on image patches 10x10px



Reconstruction of unseen image

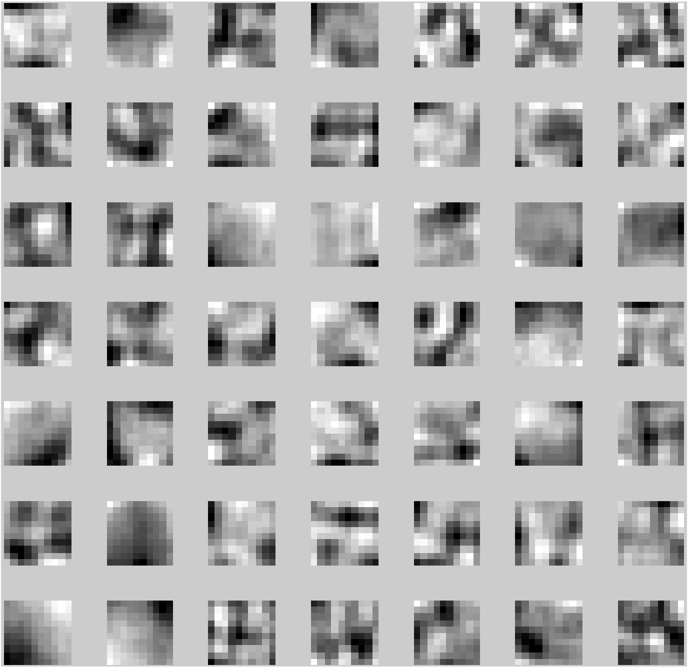


Demo: RBM for an Image

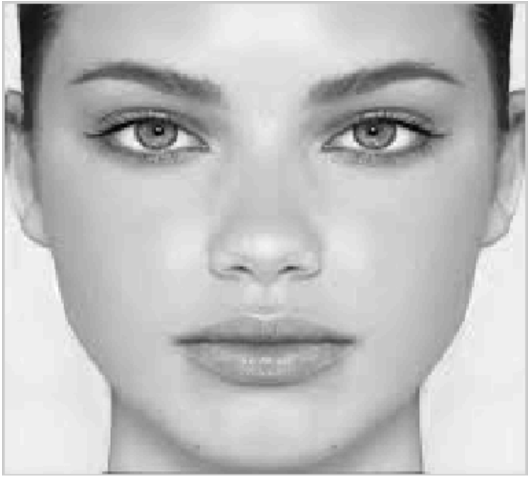
Original image

Reconstruction from non-binary features

Weights 49 x 10x10 obtained from training on image patches 10x10px

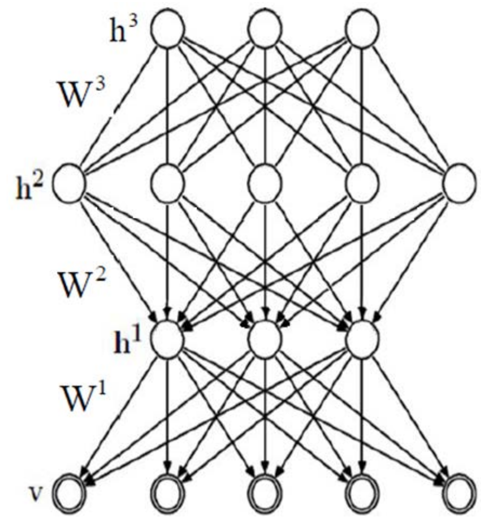


Reconstruction of unseen image

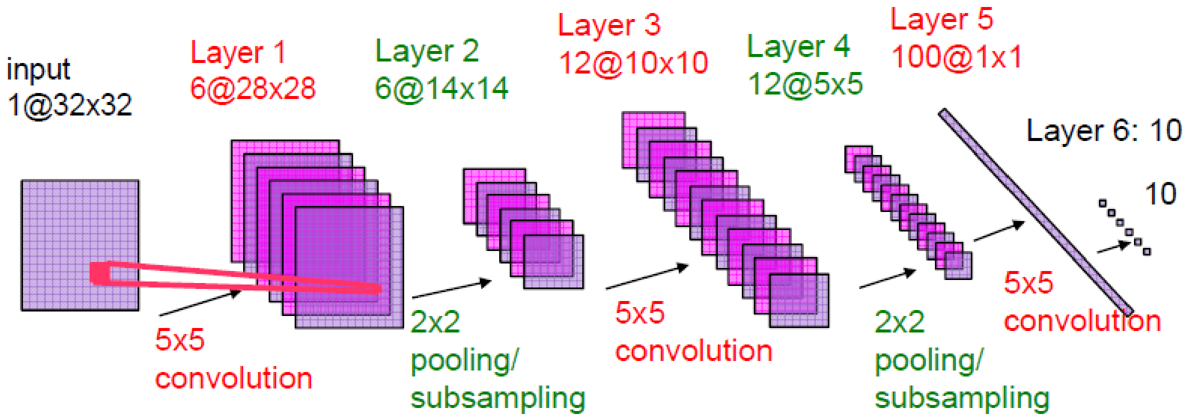


Deep Neural Networks

- **Deep Believe Networks (DBNs)**



- **Convolutional Neural Networks (CNNs)**



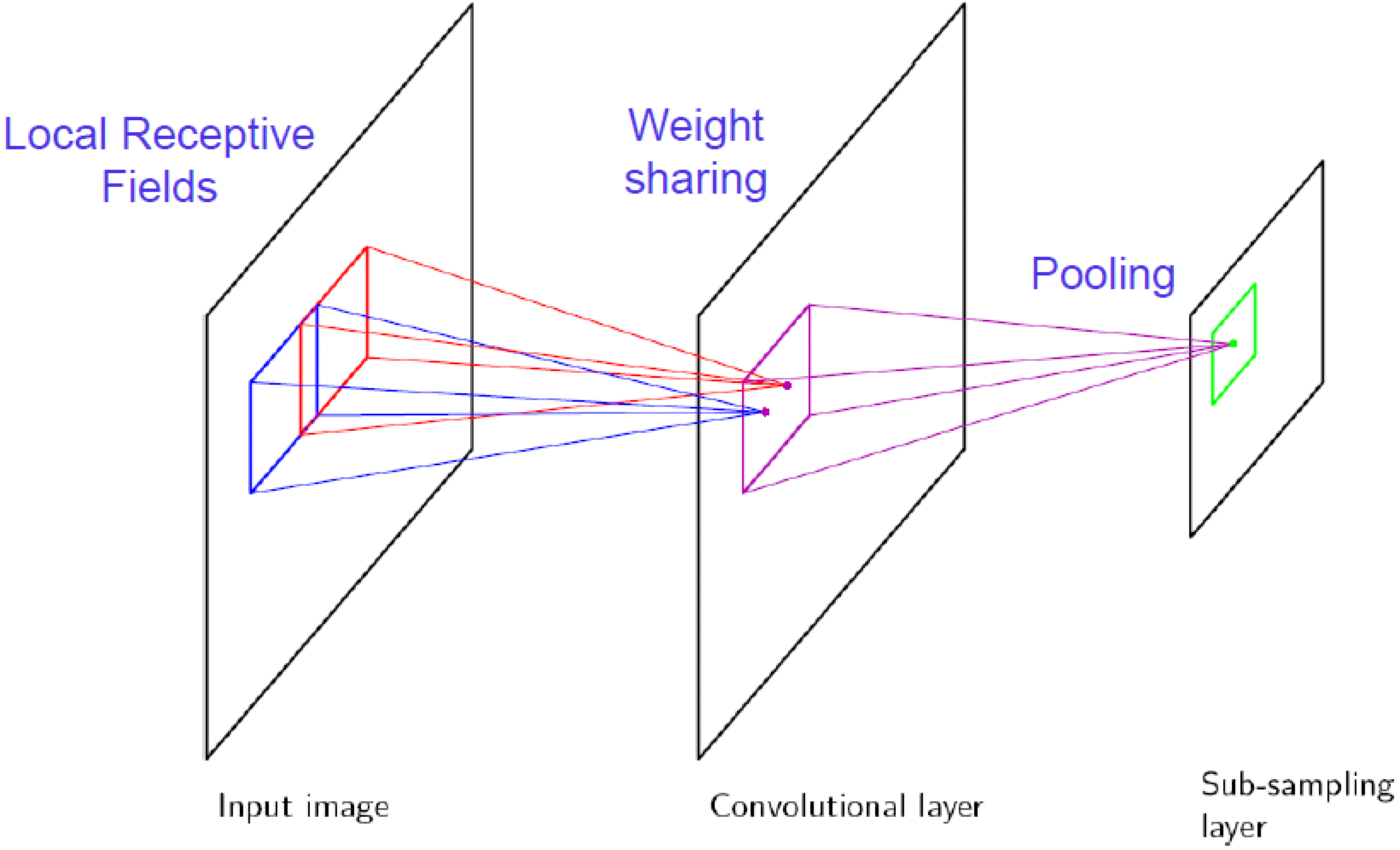
Convolutional Neural Networks

- **Are deployed in many practical applications**
  - Image recognition, speech recognition, Google's and Baidu's photo taggers

- **Have won several competitions**
  - ImageNet, Kaggle Facial Expression, Kaggle Multimodal Learning, German Traffic Signs, Connectomics, Handwriting....

- **Are applicable to array data where nearby values are correlated**
  - Images, sound, time-frequency representations, video, volumetric images, RGB-Depth images,.....

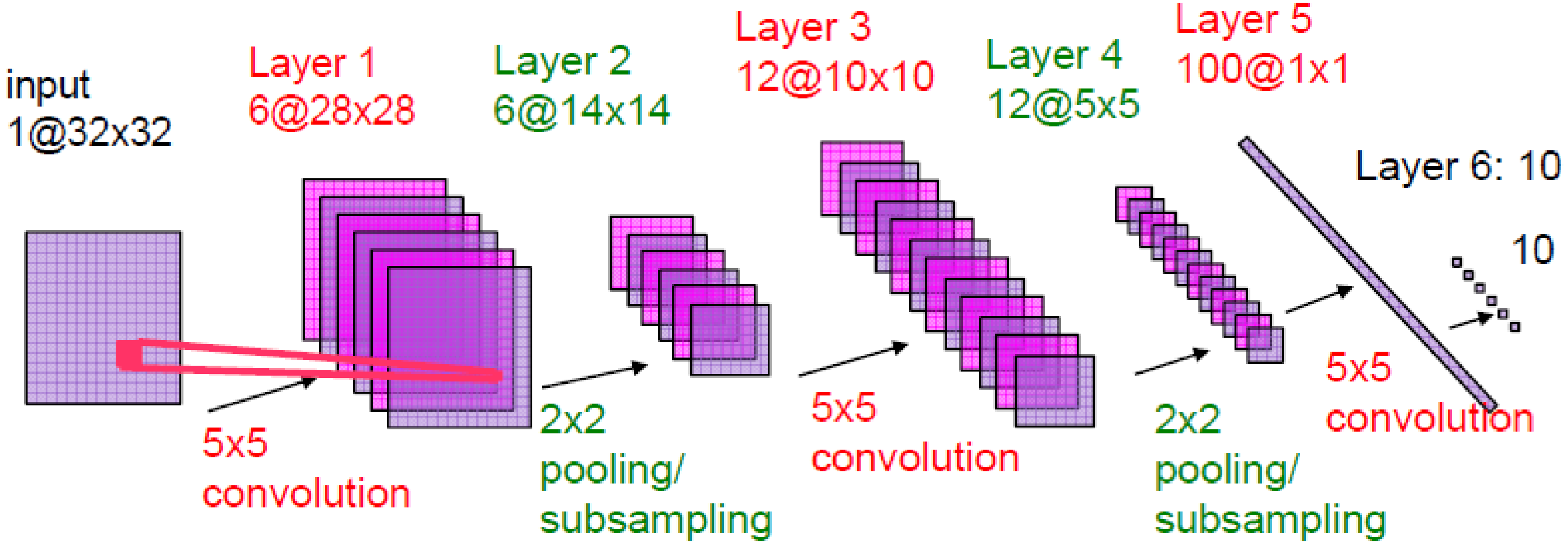
CNN Structure



LeCun at al., 1989

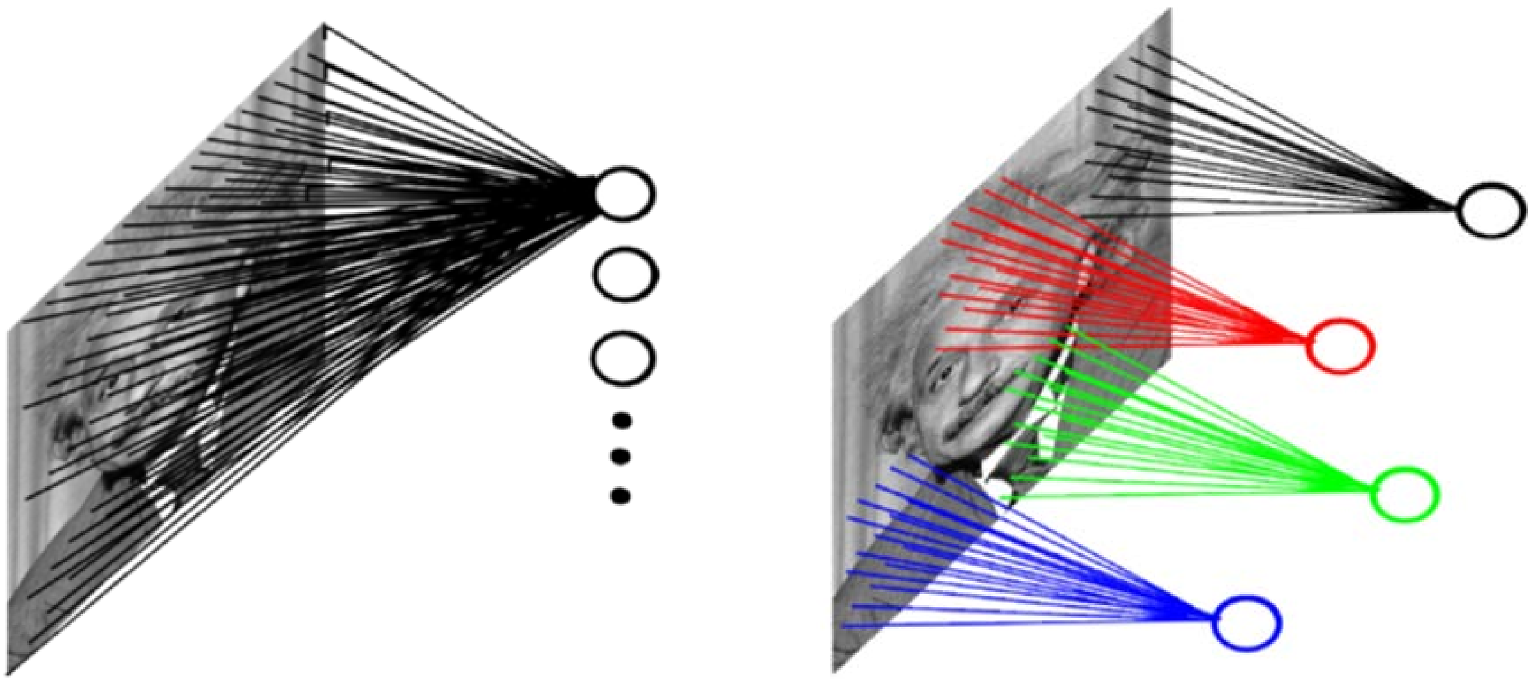
CNN Structure (cont.)

- Stacking multiple stages of convolution and pooling/subsampling



Fully Connected vs. Convolutional Net

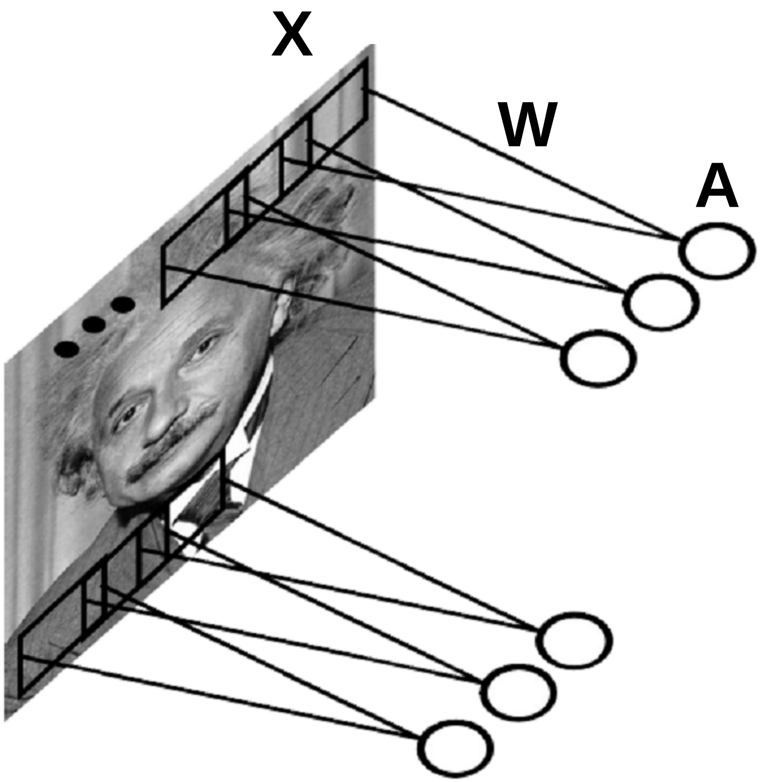
- **Example: 200x200 image**
  - Fully-connected, 400,000 hidden units = 16 billion parameters
  - Locally-connected, 400,000 hidden units 10x10 fields = 40 million parameters
  - Only few inputs per neuron: helps gradients to propagate through so many layers without diffusing so much



Convolution

- Convolution with a kernel (filter):

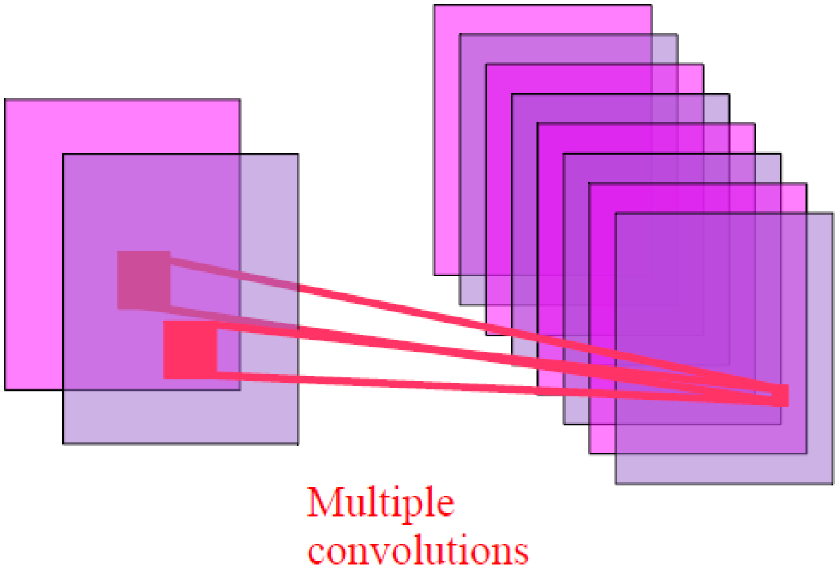
$$A_{ij} = \sum_{kl} W_{kl} X_{i+k,j+l}$$

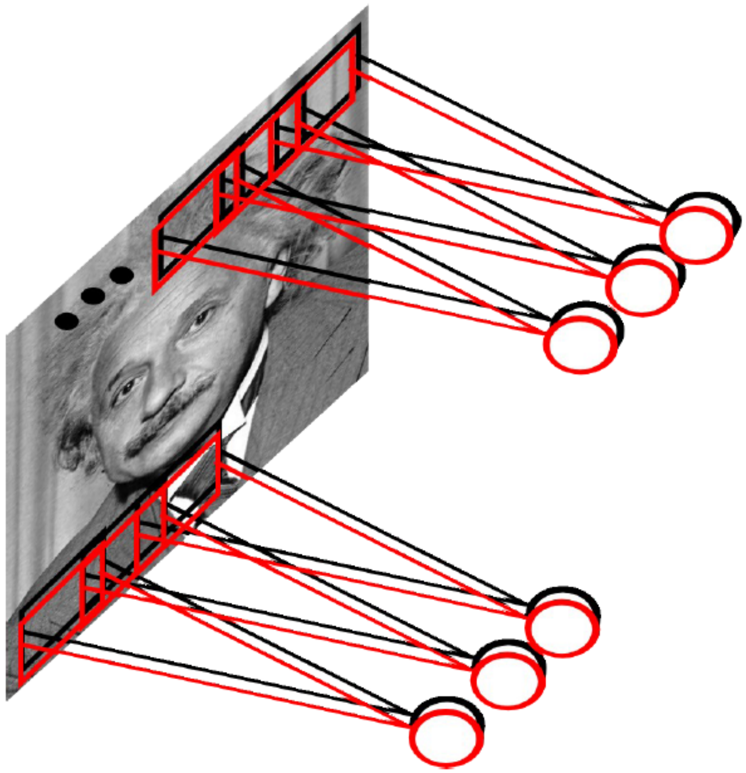


Note: All nodes share the same weight matrix

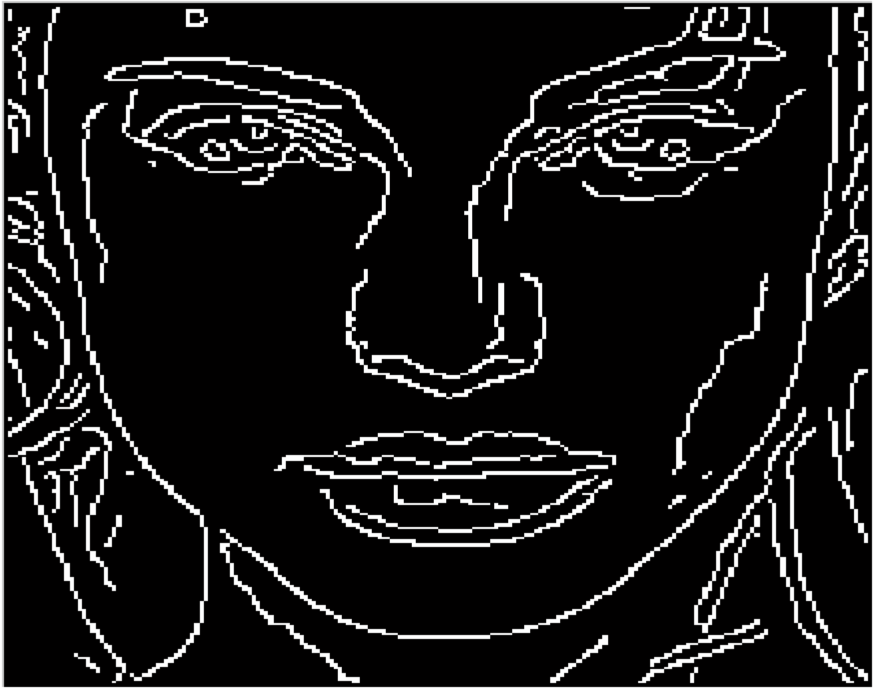
Multiple Convolutions with Different Kernels

- Detects multiple motifs at each location
- The result is a 3D array, where each slice is a feature map



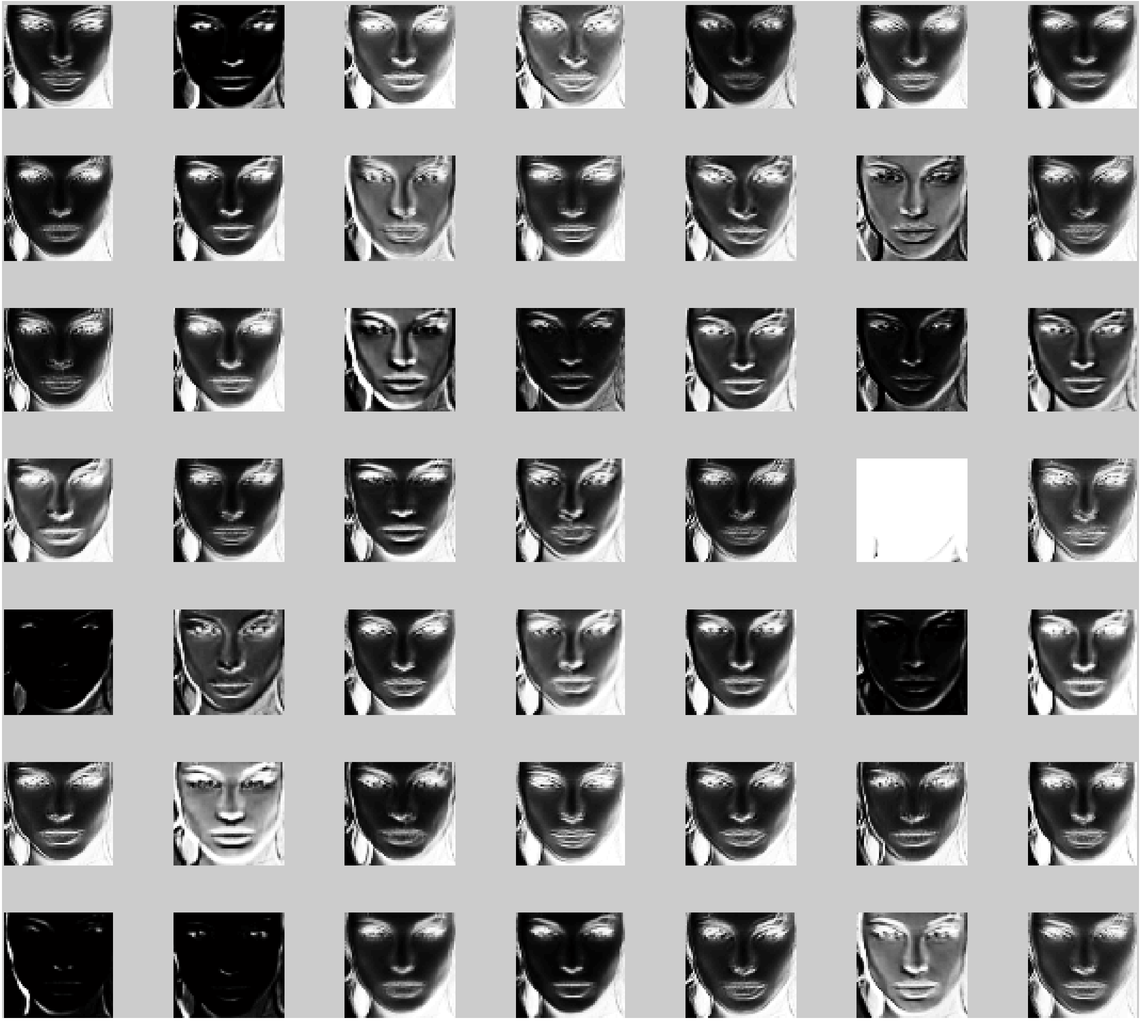


Demo: Convolution

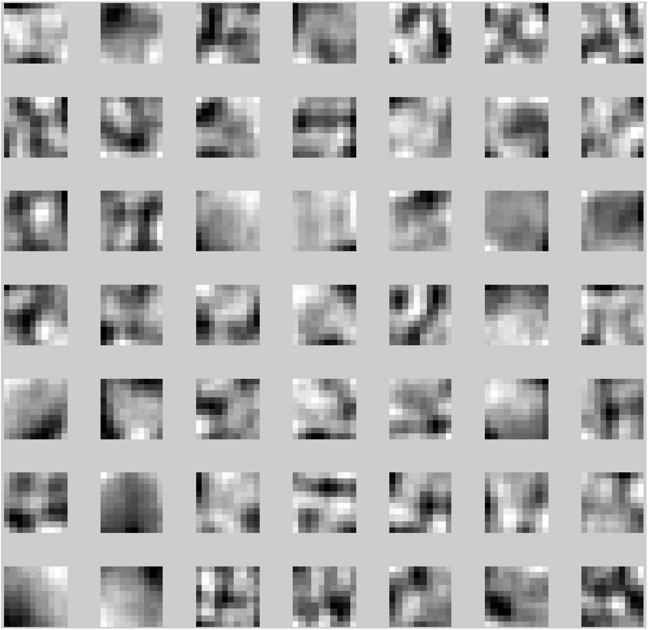


Feature Maps for an Image Using RBM

Feature maps 49 x 166x166px

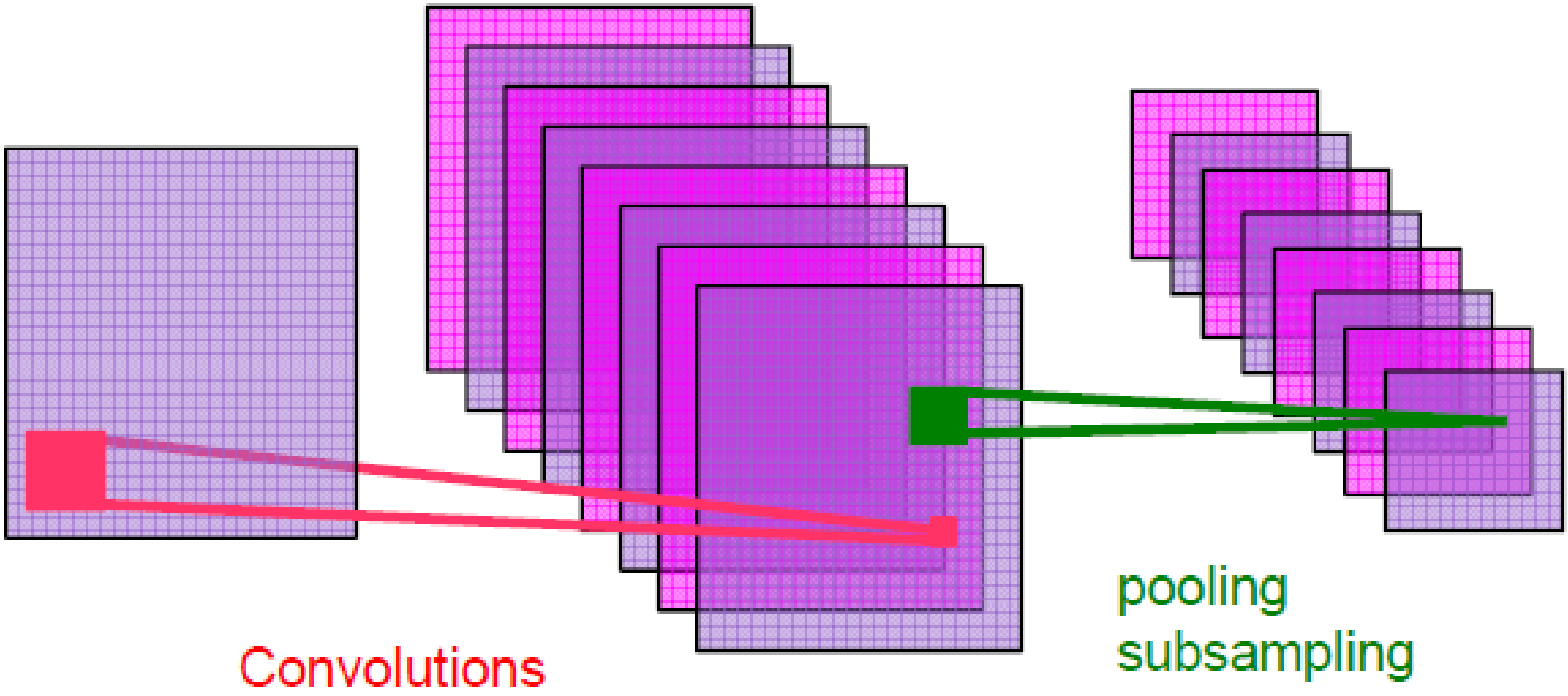


Weights 49 x 10x10px



Pooling/Subsampling

- Small rectangular blocks from the convolutional layer are subsampled to produce single outputs from those blocks

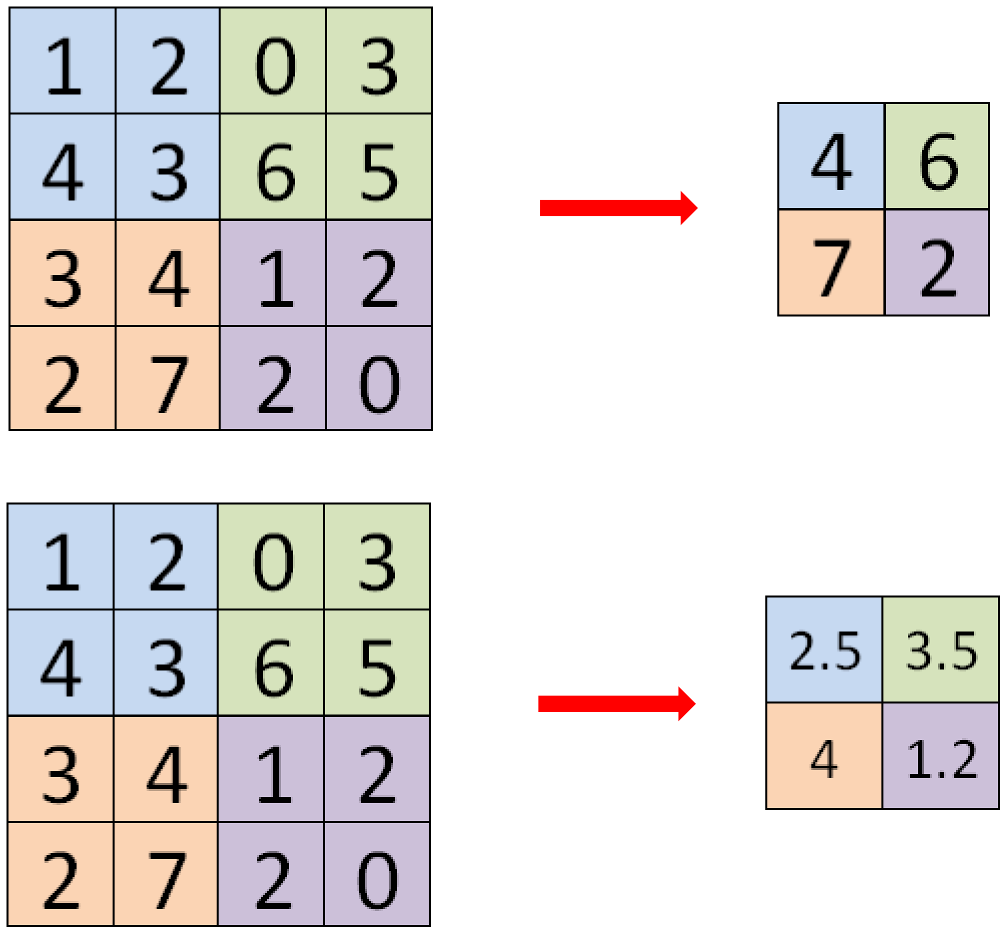


Pooling/Subsampling (cont.)

- Subsampling with 2x2 filter and stride 2

Max pooling

Averaging pooling



Pooling/Subsampling (cont.)

- Subsampling with 4x4 filter and stride 4

200x200px

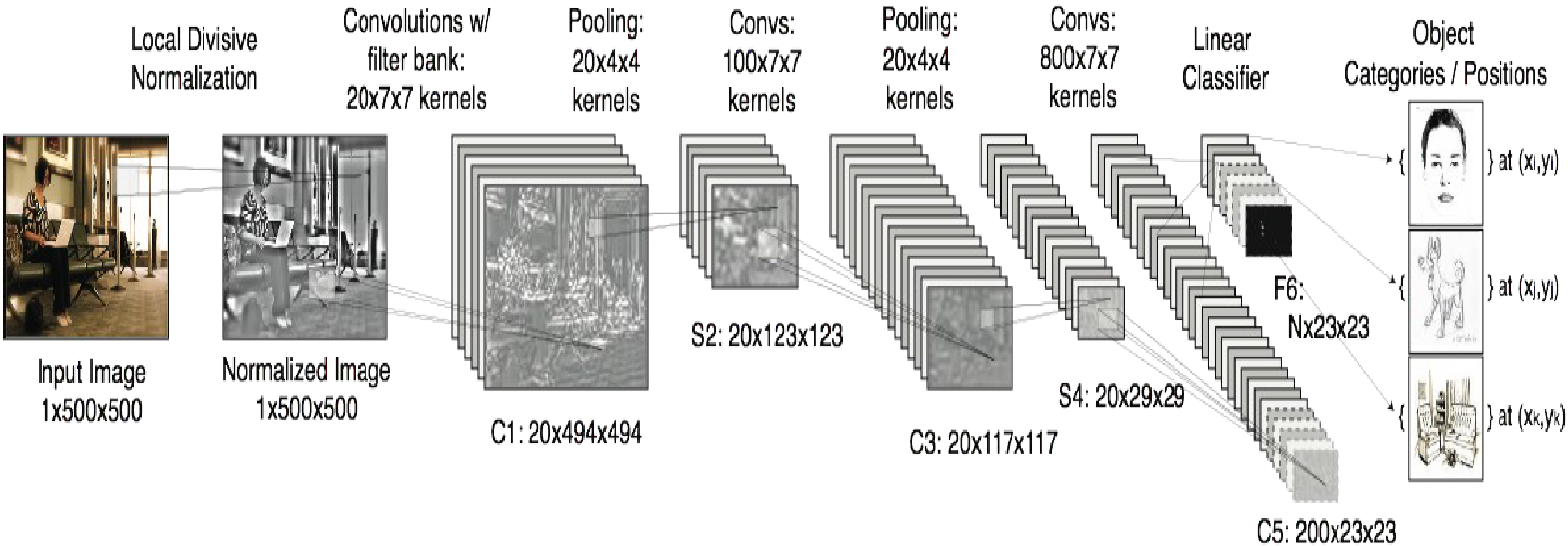
Max pooling

50x50px

Averaging pooling

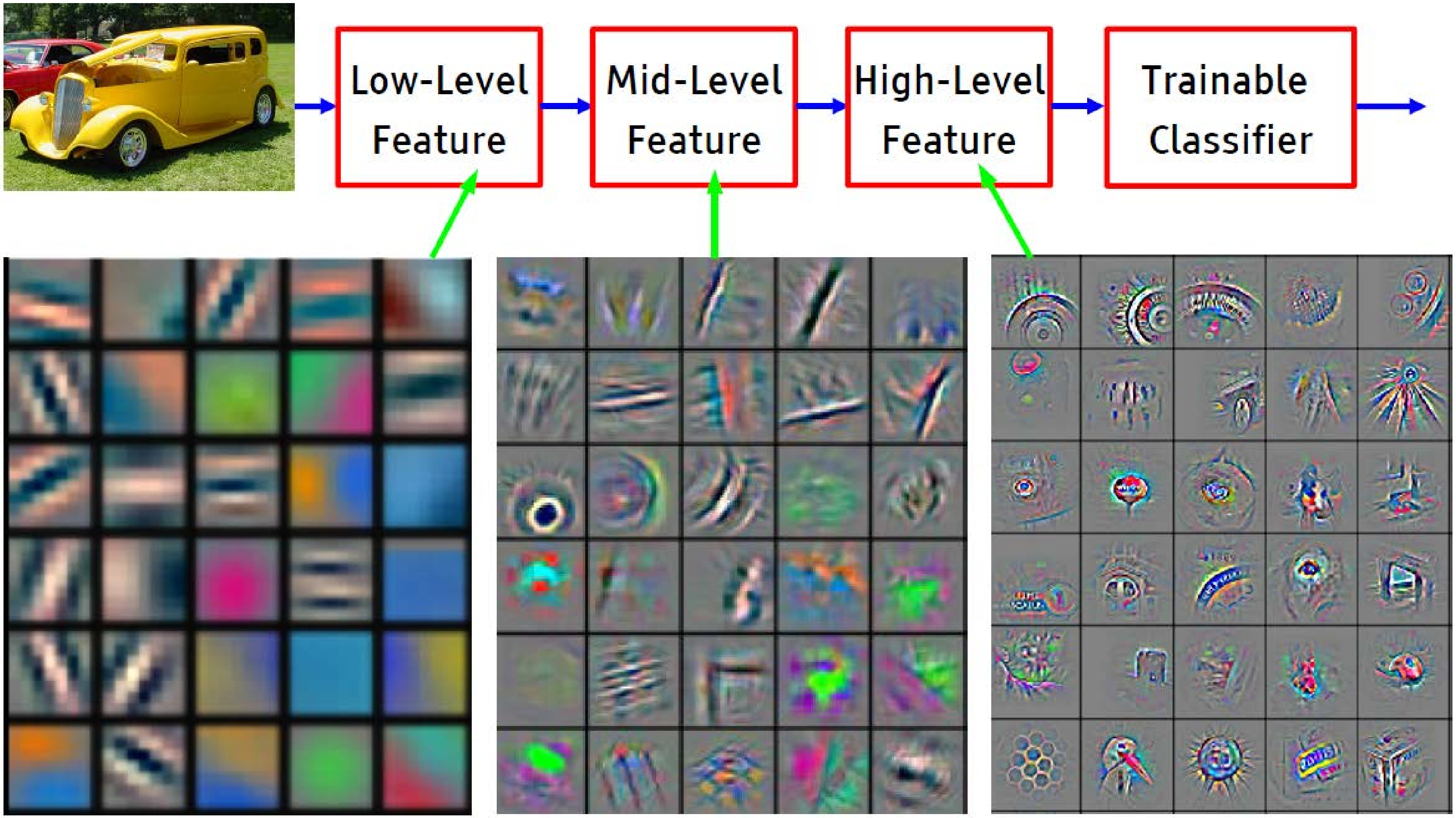
50x50px

Example of Convolutional Net



Multistage Hubel-Wiesel system (LeCun et al., 89; LeCun et al., 98)

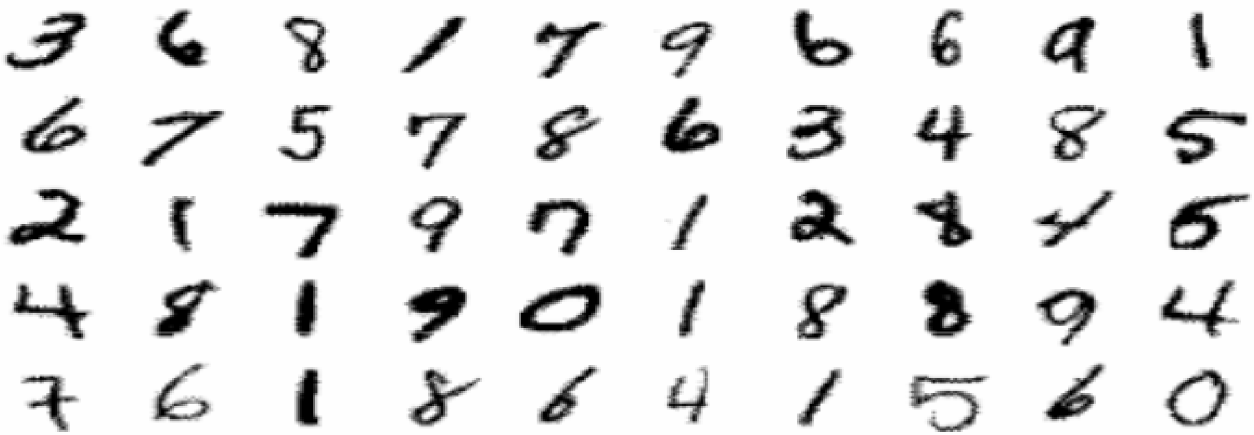
Example of Convolutional Net (cont.)



Feature visualization of convolutional net trained on ImageNet
(Zeiler & Fergus 2013)

Comparison of Classifiers on MNIST Dataset

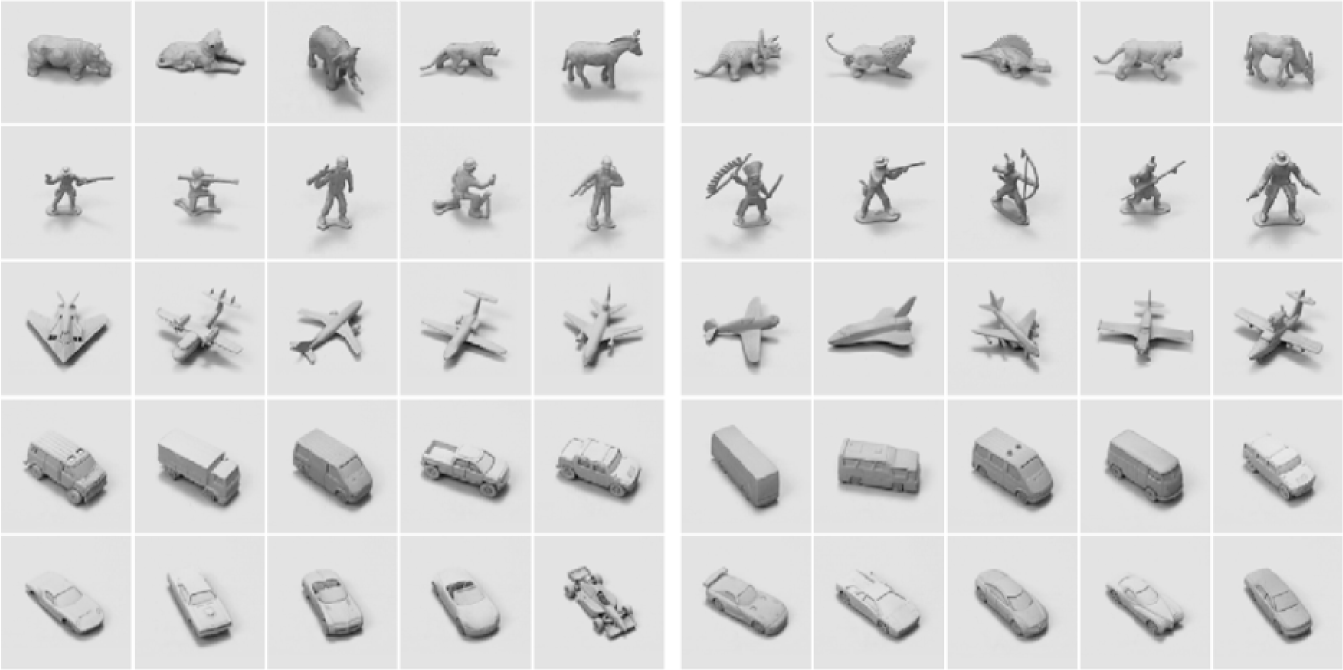
CLASSIFIER	ERROR
Knowledge-free methods	
2-layer NN, 800 HU, CE	1.60
3-layer NN, 500+300 HU, CE, reg	1.53
SVM, Gaussian Kernel	1.40
Unsupervised Stacked RBM + backprop	0.95
Convolutional nets	
Convolutional net LeNet-5,	0.80
Convolutional net LeNet-6,	0.70
Conv. net LeNet-6- + unsup learning	0.60



Yann LeCun

Comparison of Classifiers on Object Recognition

Linear Classifier on raw stereo images:	30.2% error
K-Nearest-Neighbors on raw stereo images:	18.4% error
K-Nearest-Neighbors on PCA-95:	16.6% error
Pairwise SVM on 96x96 stereo images:	11.6% error
Pairwise SVM on 95 Principal Components:	13.3% error
Convolutional Net on 96x96 stereo images:	5.8% error



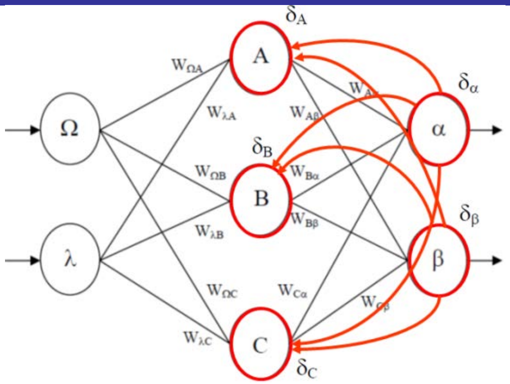
Training instances

Test instances

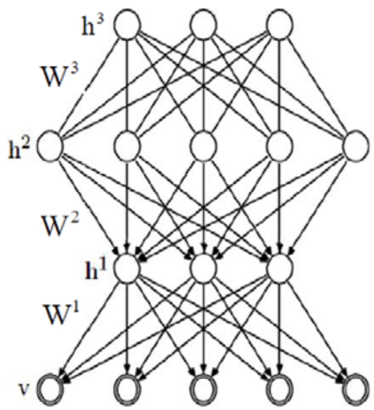
Yann LeCun

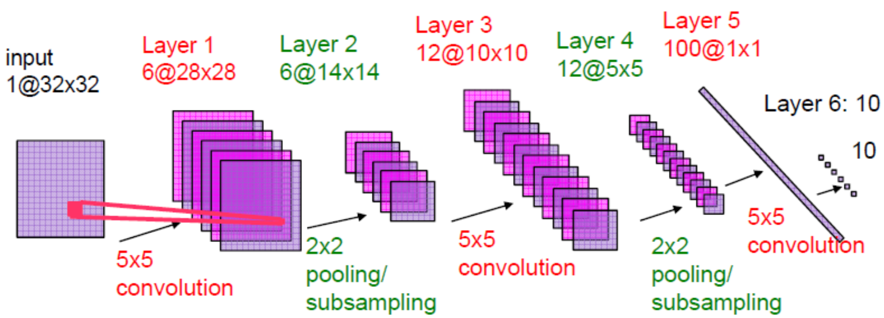
Summary

- **Backpropagation for MLP learning**
  - Easy to implement
  - Becomes slow when using many hidden layers due to diluting gradient
  - Can get stuck in local optima when starting with random initialization



- **Deep neural networks**
  - Hierarchical learning
  - Unsupervised layerwise learning
  - Supervised learning for final tuning of weights and classification
  - Deep Belief Networks
  - Convolutional Neural Networks





Homework

- Task 1
  - Train RBM on digits from 1 to 5
  - Do reconstruction on digits from 1 to 9

- Task 2
  - Train RBM on an image from one category (e.g., dog)
  - Do reconstruction on an image from other category (e.g., cat)