Deep Learning for Vision

Marc'Aurelio Ranzato

Facebook, AI Group

www.cs.toronto.edu/~ranzato

Berkeley – 24 Jansuary 2014

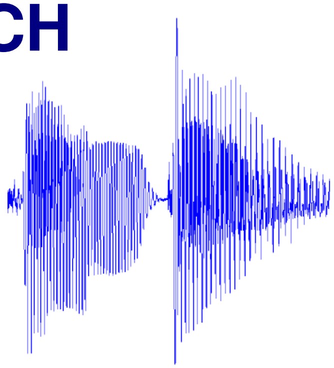
Traditional Pattern Recognition

VISION

→ SIFT/HOG → K-Means/ pooling → classifier → "car"

fixed | unsupervised | supervised

SPEECH



→ MFCC → Mixture of Gaussians → classifier → \'d ē p\

fixed | unsupervised | supervised

NLP

This burrito place is yummy and fun! → Parse Tree Syntactic → n-grams → classifier → "+"

fixed | unsupervised | supervised

Hierarchical Compositionality (DEEP)

VISION

pixels → edge → texton → motif → part → object

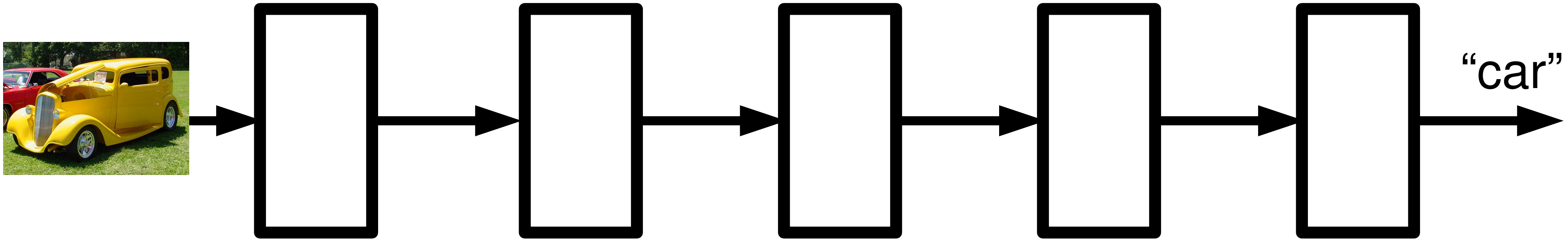
SPEECH

sample → spectral band → formant → motif → phone → word

NLP

character → word → NP/VP/.. → clause → sentence → story

Deep Learning



What is Deep Learning

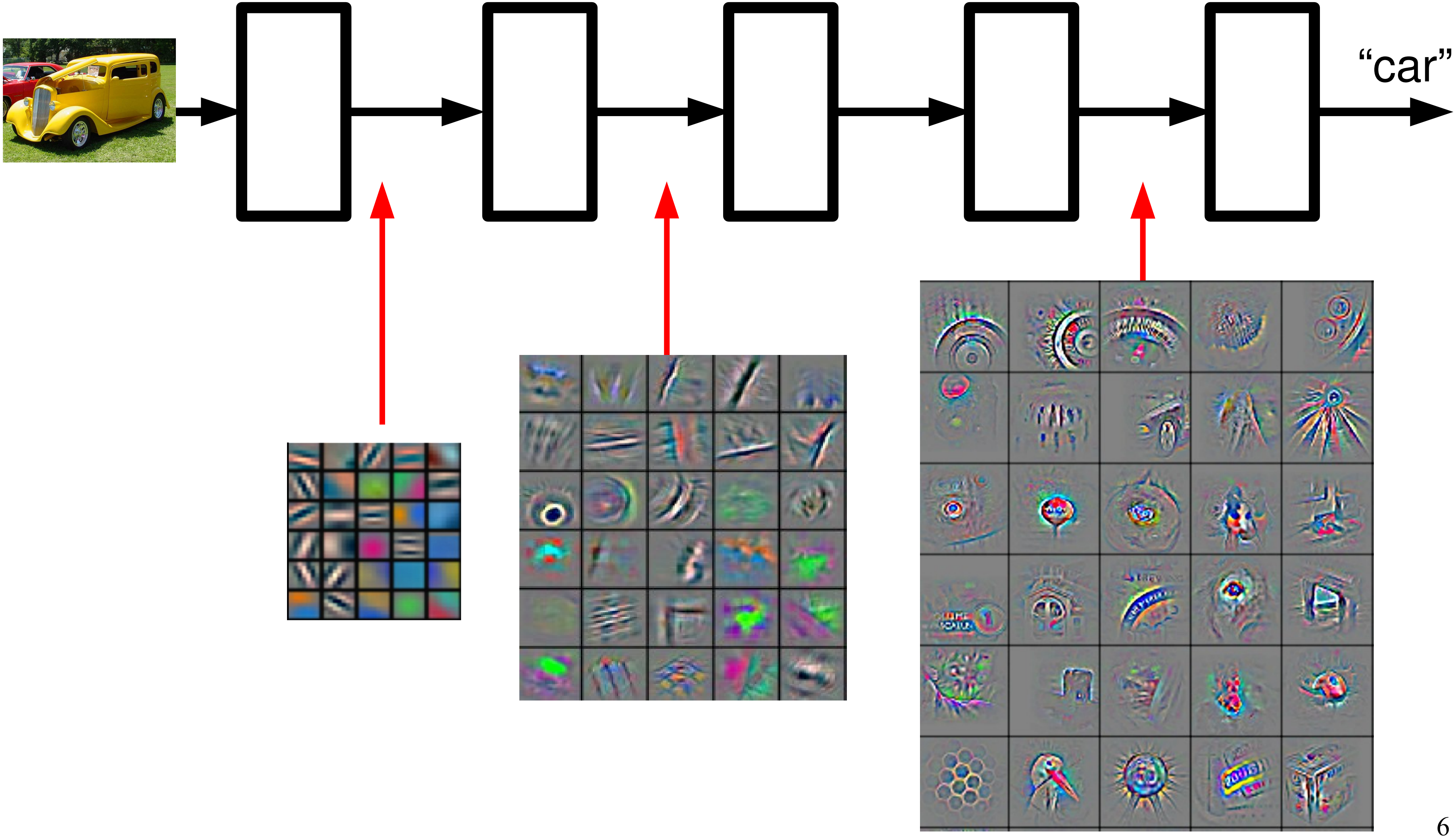
- Cascade of non-linear transformations
- End to end learning
- General framework (any hierarchical model is deep)

Deep Learning VS Shallow Learning

- Structure of the system naturally matches the problem which is inherently hierarchical.

pixels → edge → texton → motif → part → object

Deep Learning



Zeiler et al. "Visualizing and Understanding ConvNets" Arxiv 2013

Ranzato

Deep Learning VS Shallow Learning

- Structure of the system naturally matches the problem which is inherently hierarchical.

pixels → edge → texton → motif → part → object

- It is more efficient.

E.g.: Checking N-bit parity requires N-1 gates laid out on a tree of depth log(N-1). The same would require O(exp(N)) with a two layer architecture.

$$p = \sum_i \alpha_i f_i(x) \qquad \text{VS} \qquad p = \alpha_n f_n(\alpha_{n-1} f_{n-1}(\ldots \alpha_1 f_1(x) \ldots))$$

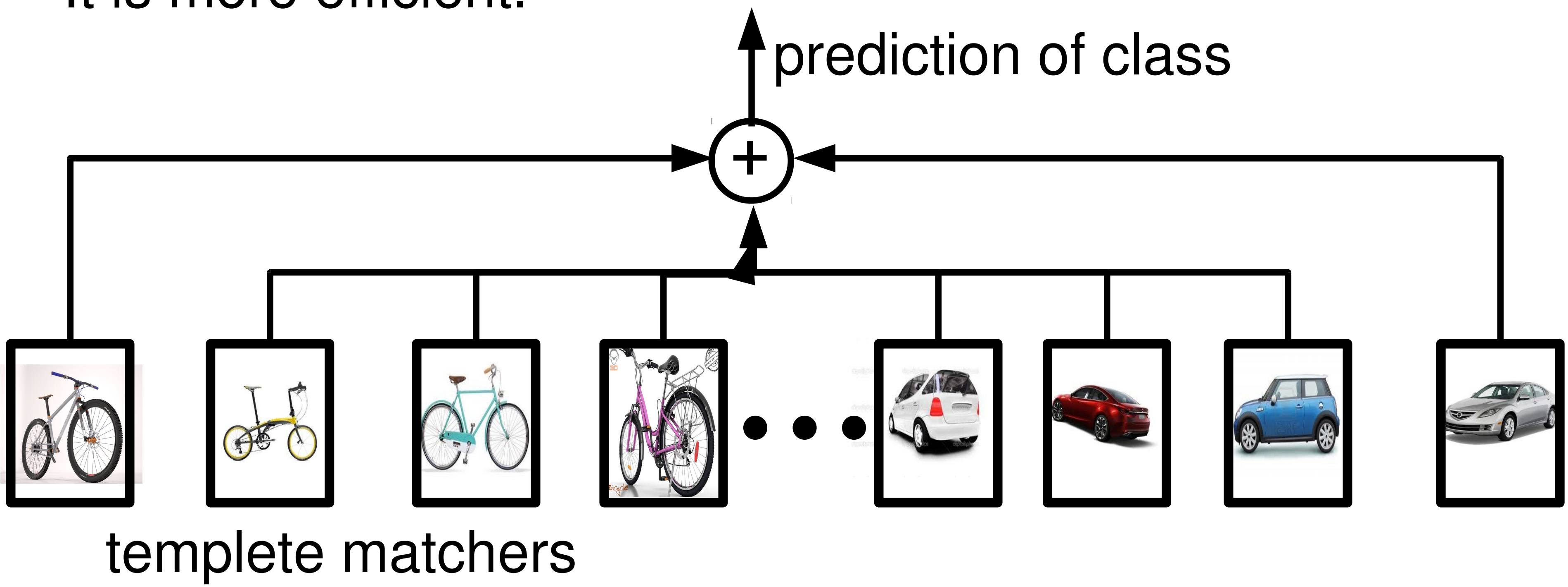
Shallow learner is often inefficient: it requires exponential number of templates (basis functions).

Deep Learning VS Shallow Learning

- Structure of the system naturally matches the problem which is inherently hierarchical.

pixels → edge → texton → motif → part → object

- It is more efficient.

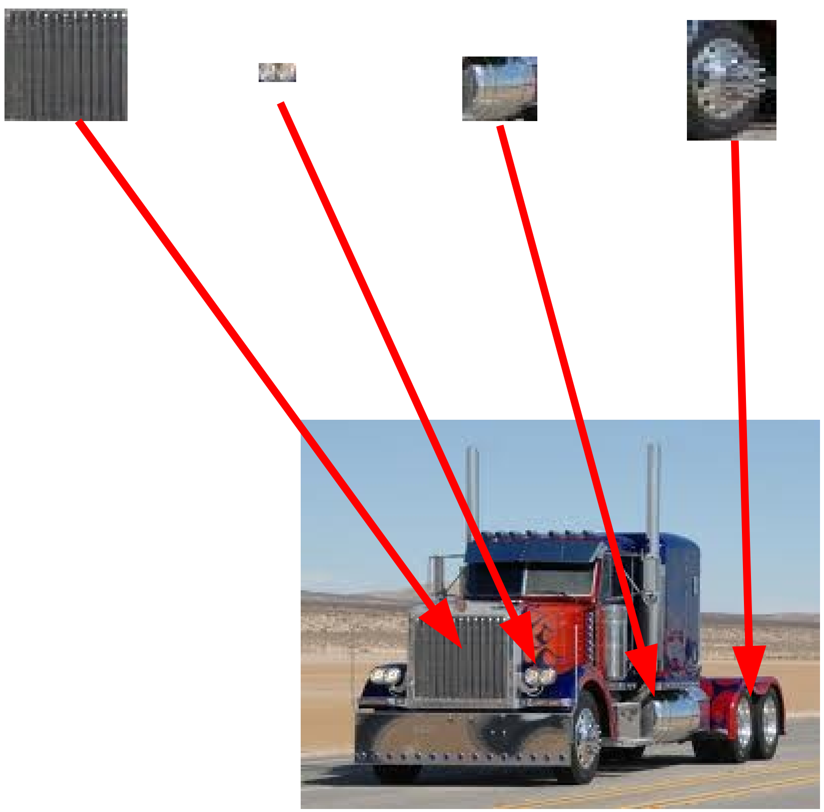


Shallow learner is inefficient.

Ranzato

Composition: distributed representations

[0 0 **1** 0 0 0 0 **1** 0 0 **1** **1** 0 0 **1** 0 …] truck feature

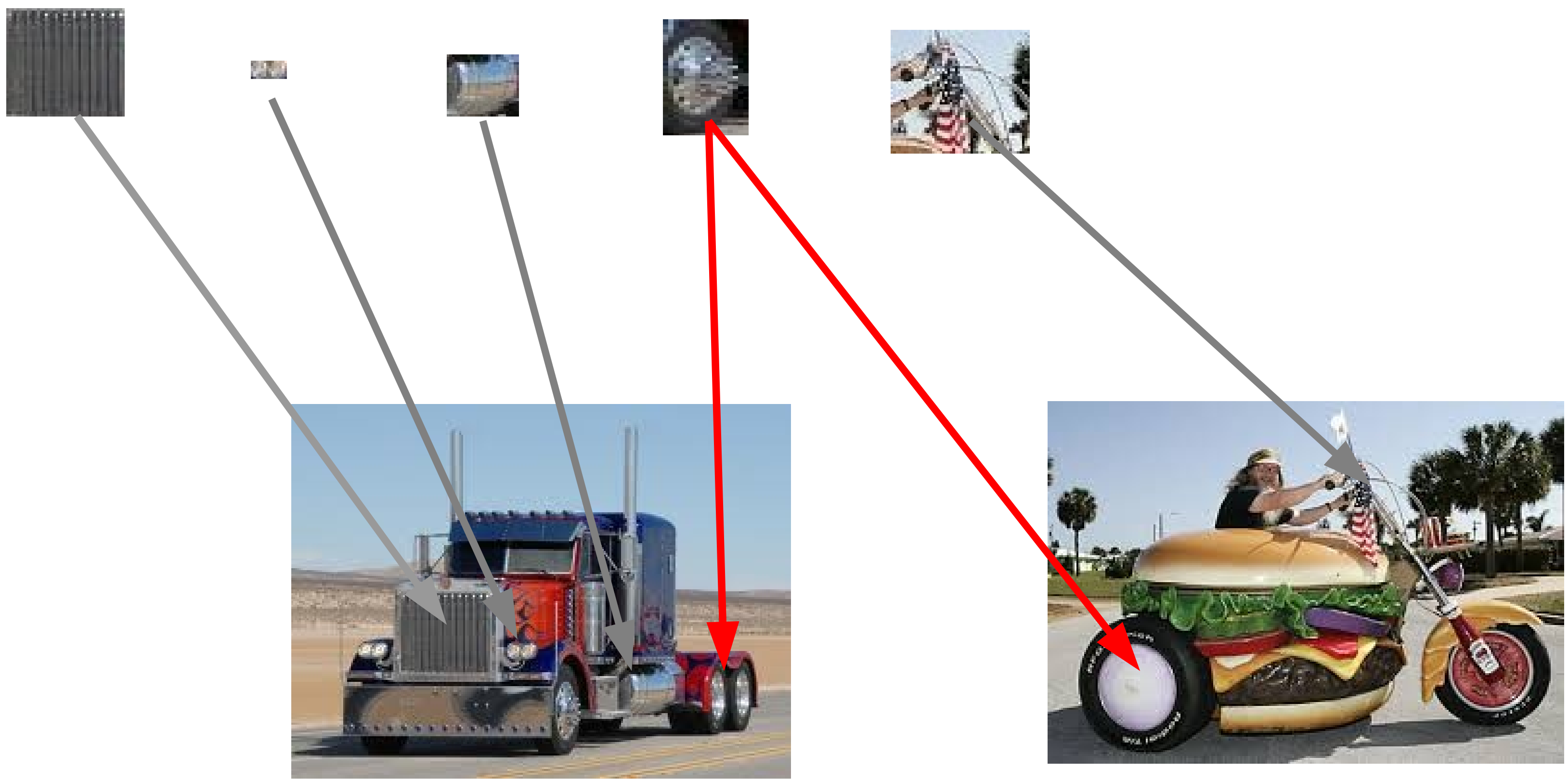


Exponentially more efficient than a 1-of-N representation (a la k-means)

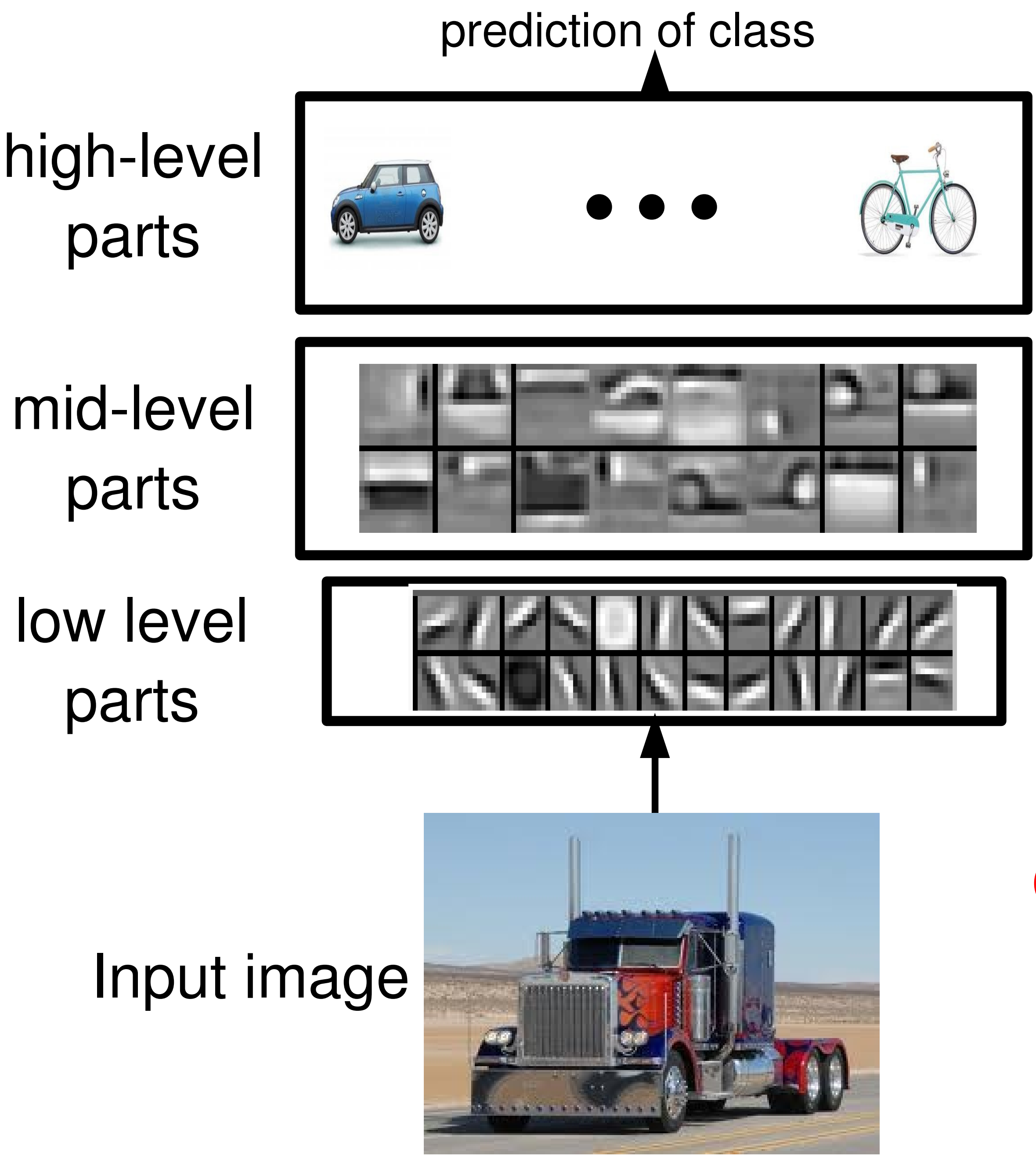
Composition: sharing

[1 1 0 0 0 1 0 **1** 0 0 0 0 1 1 0 1…] motorbike

[0 0 1 0 0 0 0 **1** 0 0 1 1 0 0 1 0 …] truck



Composition



- distributed representations
- feature sharing
- compositionality

GOOD: (exponentially) more efficient

Lee et al. “Convolutional DBN's ...” ICML 2009

Ranzato

Deep Learning

=

Representation Learning

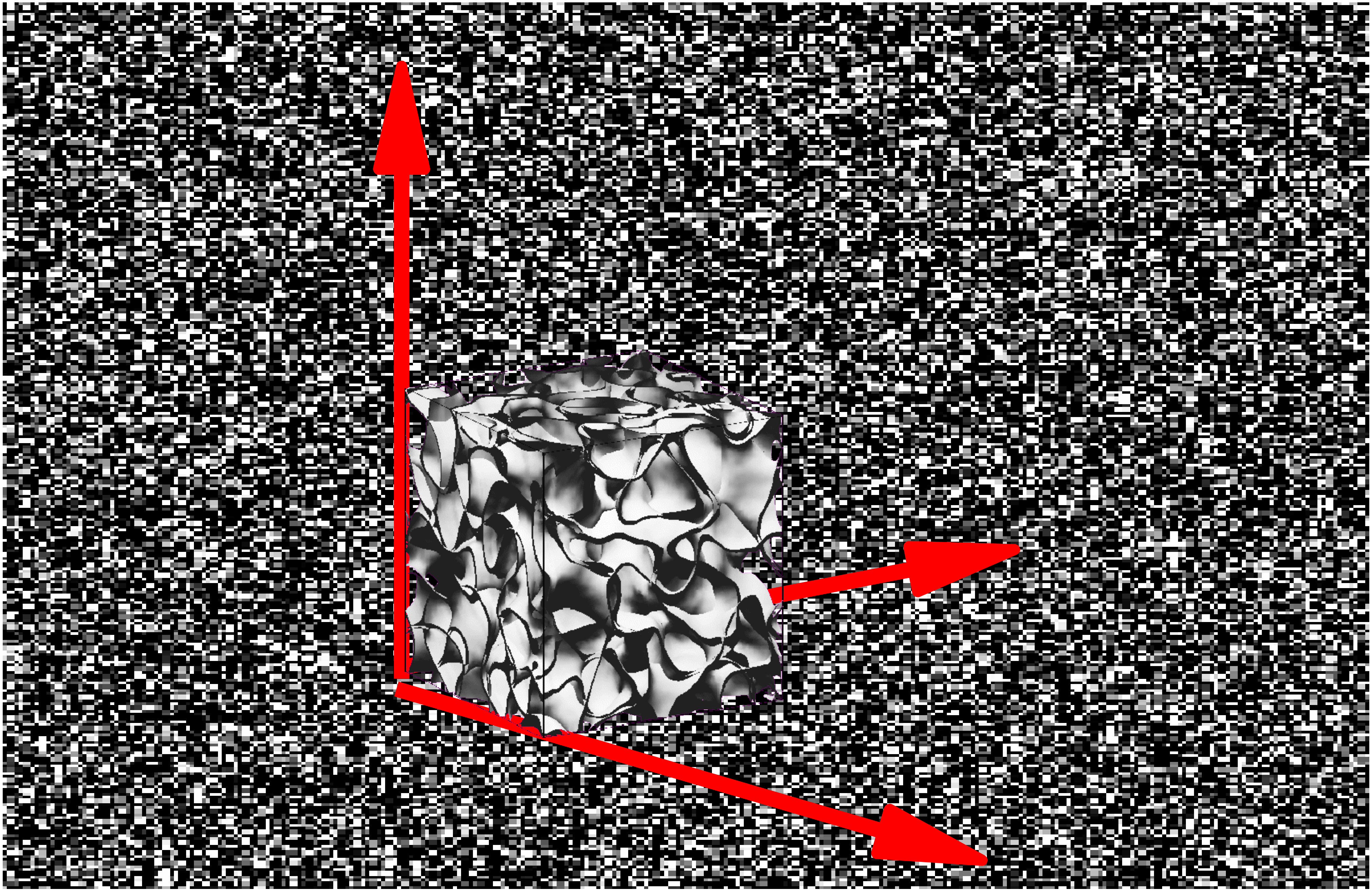
Ideal Features

→ **Ideal Feature Extractor** →

- window, right
- chair, left
- monitor, top of shelf
- carpet, bottom
- drums, corner
- ...
- pillows on couch

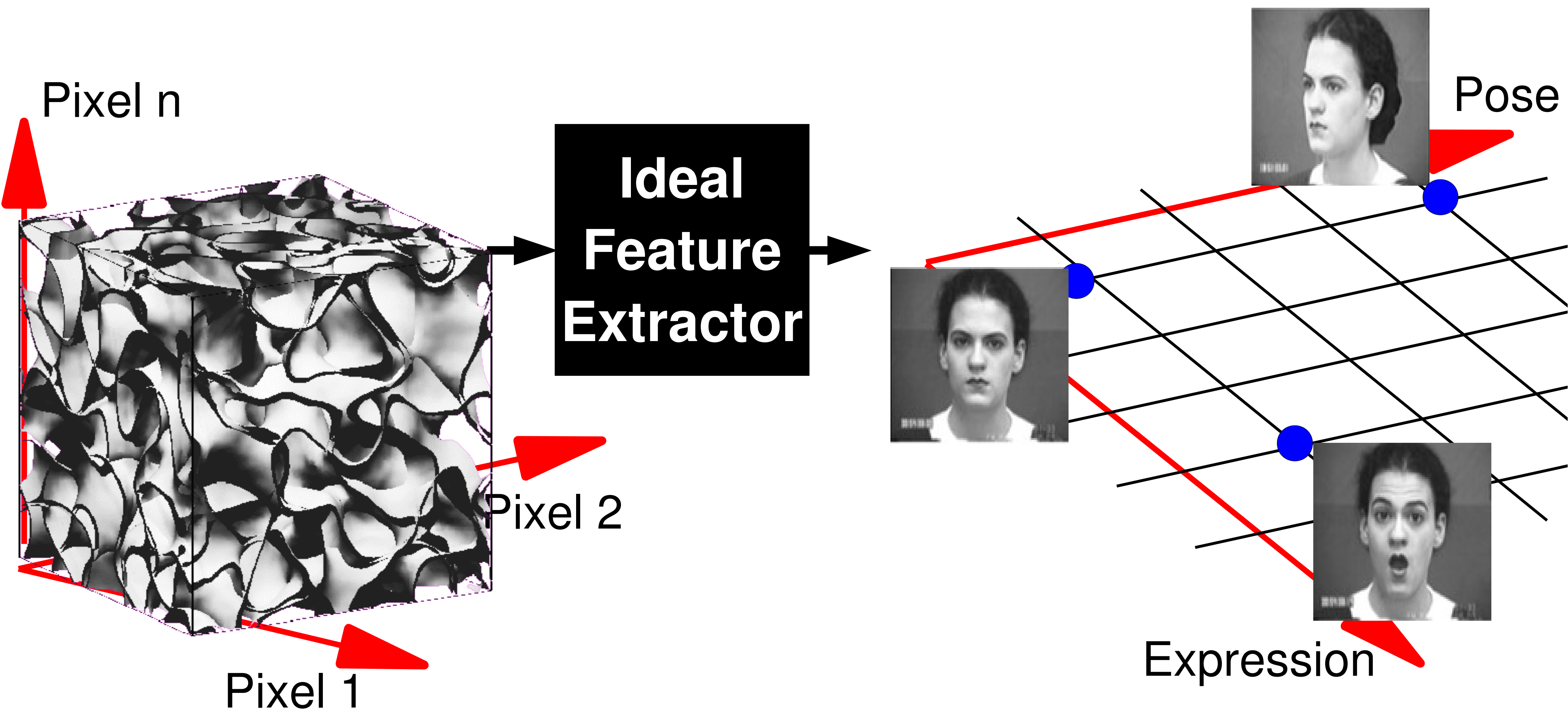
Q.: What objects are in the image? Where is the lamp? What is on the couch? ...

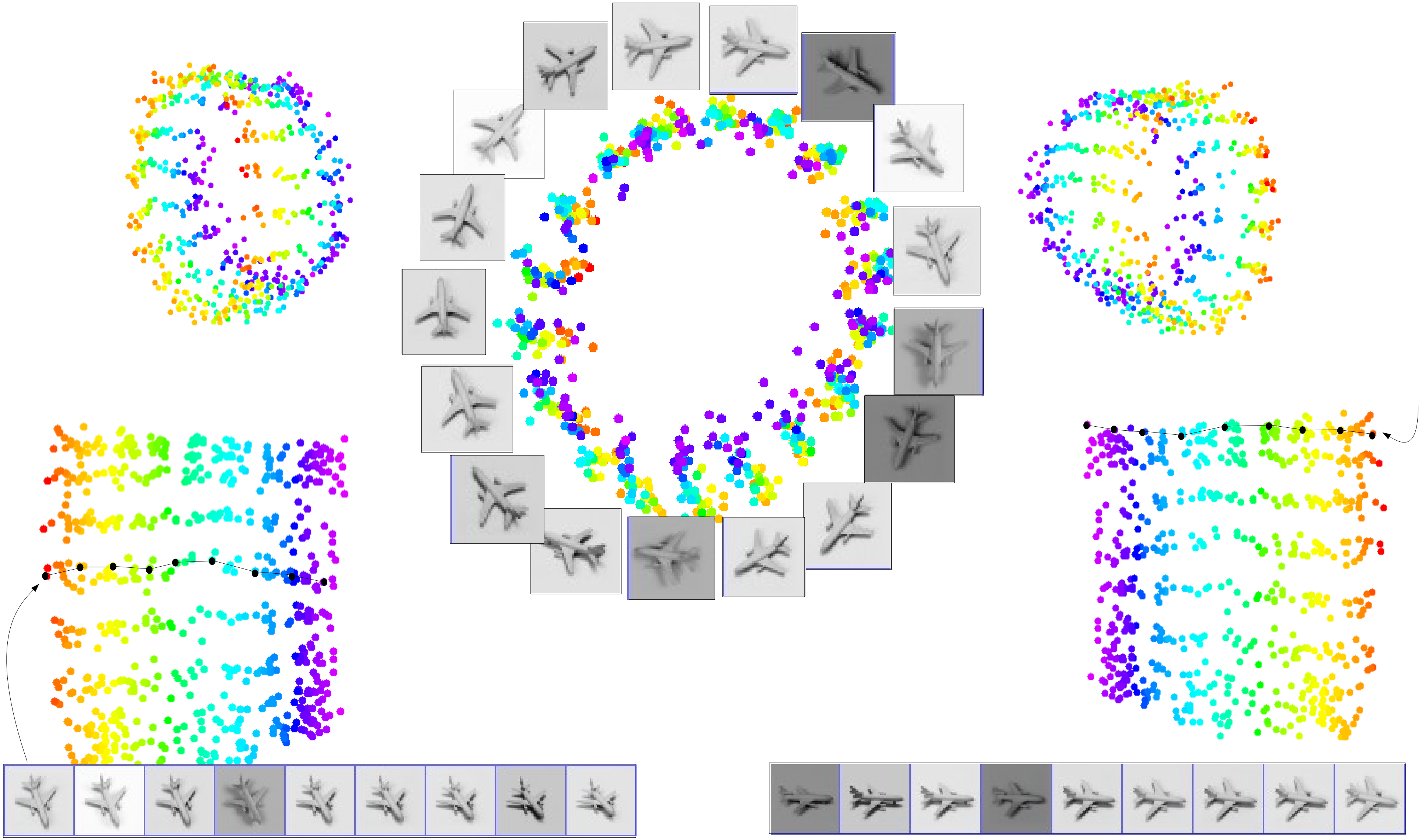
The Manifold of Natural Images



Ideal Feature Extraction

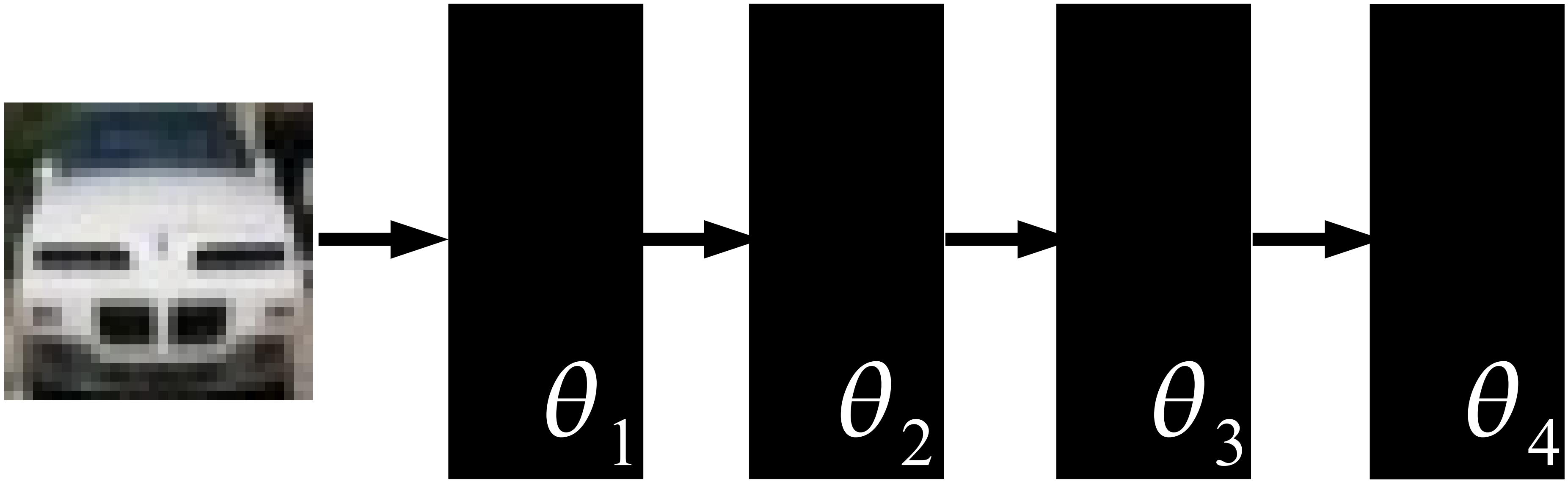
E.g.: face images live in about 60-D manifold (x,y,z, pitch, yaw, roll, 53 muscles).





Hadsell et al. “Dimensionality reduction by learning an invariant mapping” CVPR 2006

Deep Learning



Given lots of data, engineer less and learn more!!
Let the data find the structure (intrinsic dimensions).

Deep Learning in Practice

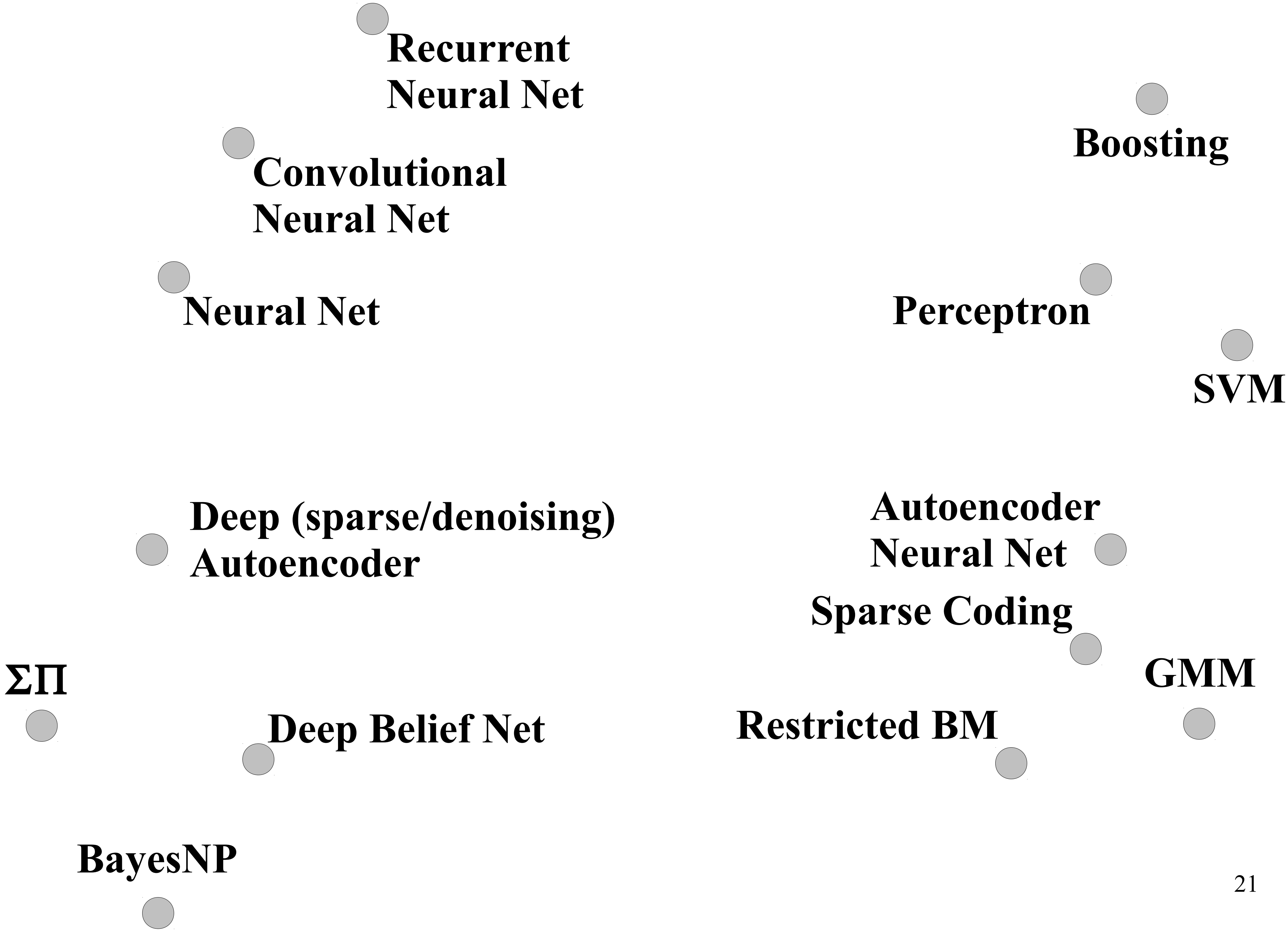
It works very well in practice:

KEY IDEAS OF DEEP LEARNING

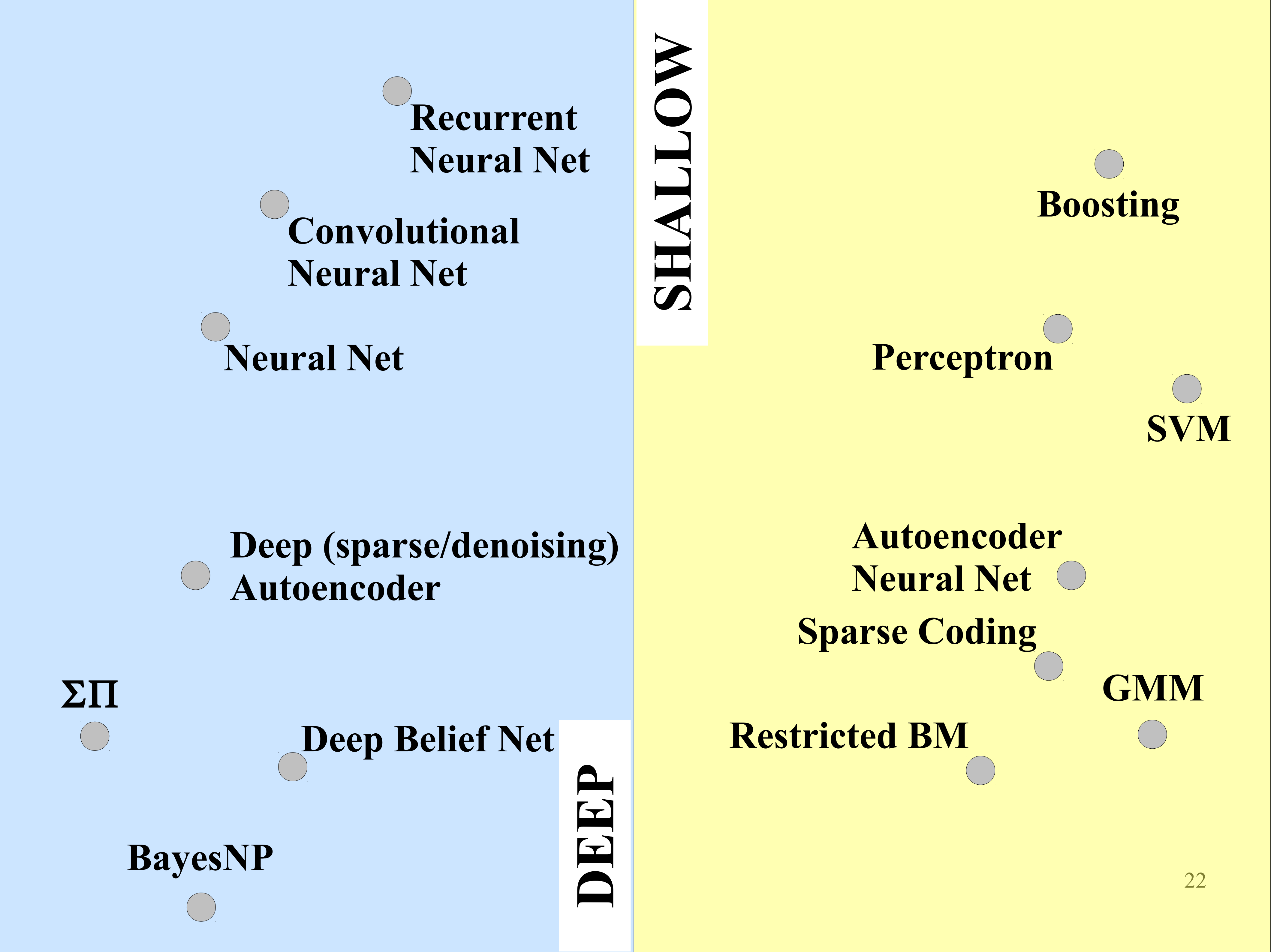
- Hierarchical non-linear system
  - Distributed representations
  - Sharing

- End-to-end learning
  - Joint optimization of features and classifier
  - Good features are learned as a side product of the learning process

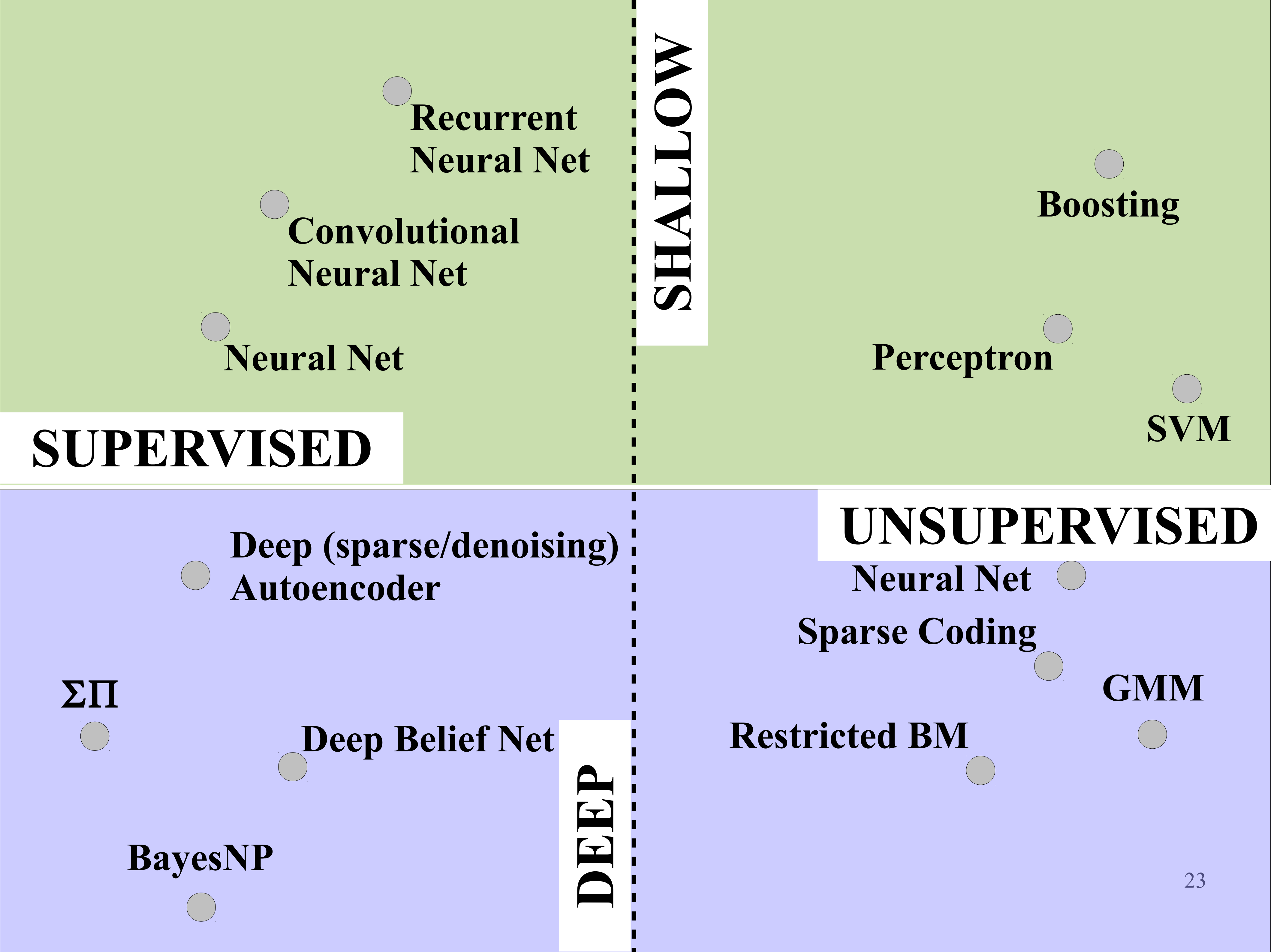
THE SPACE OF MACHINE LEARNING METHODS

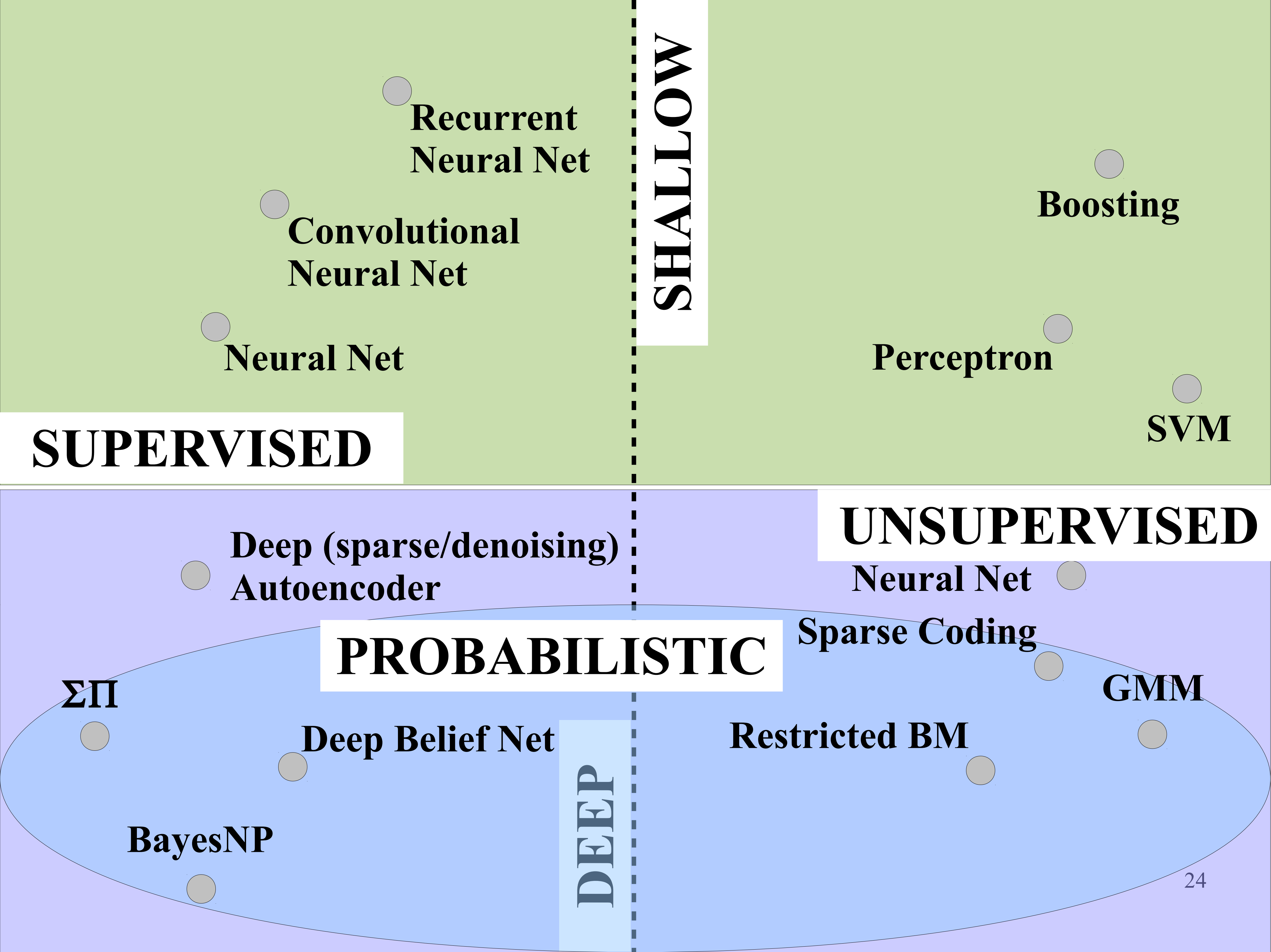


Recurrent Neural Net
Boosting
Convolutional Neural Net
Neural Net
Perceptron
SVM
Deep (sparse/denoising) Autoencoder
Autoencoder Neural Net
Sparse Coding
ΣΠ
GMM
Deep Belief Net
Restricted BM
BayesNP



SHALLOW
DEEP
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Deep (sparse/denoising) Autoencoder
ΣΠ
Deep Belief Net
BayesNP
Boosting
Perceptron
SVM
Autoencoder Neural Net
Sparse Coding
GMM
Restricted BM

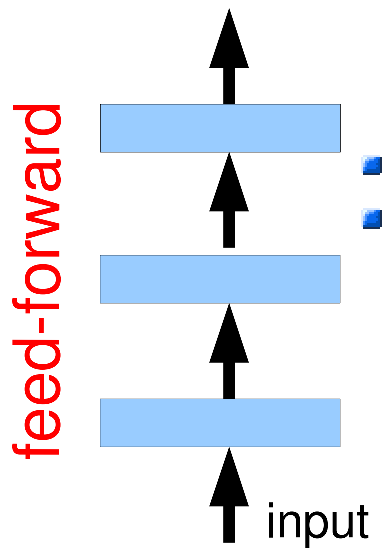

SUPERVISED
UNSUPERVISED
SHALLOW
DEEP
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Boosting
Perceptron
SVM
Deep (sparse/denoising) Autoencoder
ΣΠ
Deep Belief Net
BayesNP
Neural Net
Sparse Coding
GMM
Restricted BM



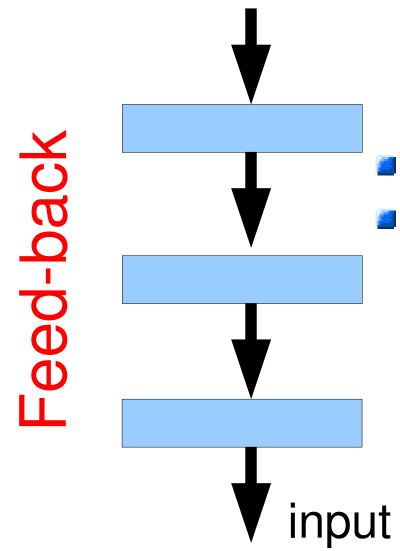
SHALLOW
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Boosting
Perceptron
SVM
SUPERVISED
UNSUPERVISED
Deep (sparse/denoising) Autoencoder
Neural Net
Sparse Coding
PROBABILISTIC
ΣΠ
GMM
Deep Belief Net
Restricted BM
BayesNP
DEEP

Deep Learning is B I G

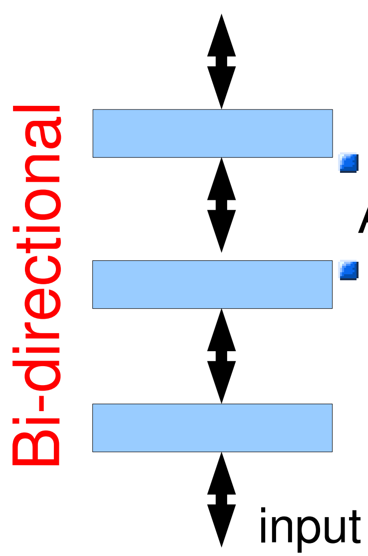
- Main types of deep architectures



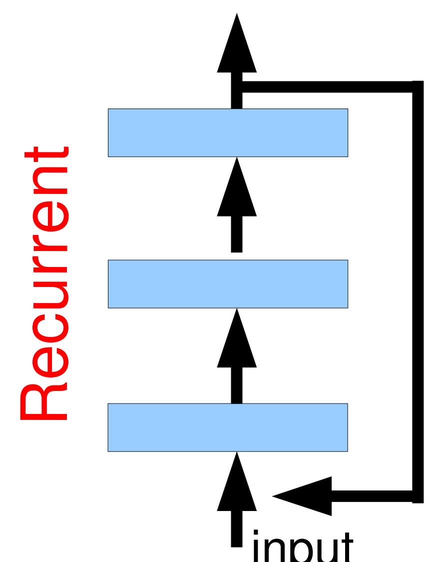
- Neural nets
- Conv Nets



- Hierar. Sparse Coding
- Deconv Nets



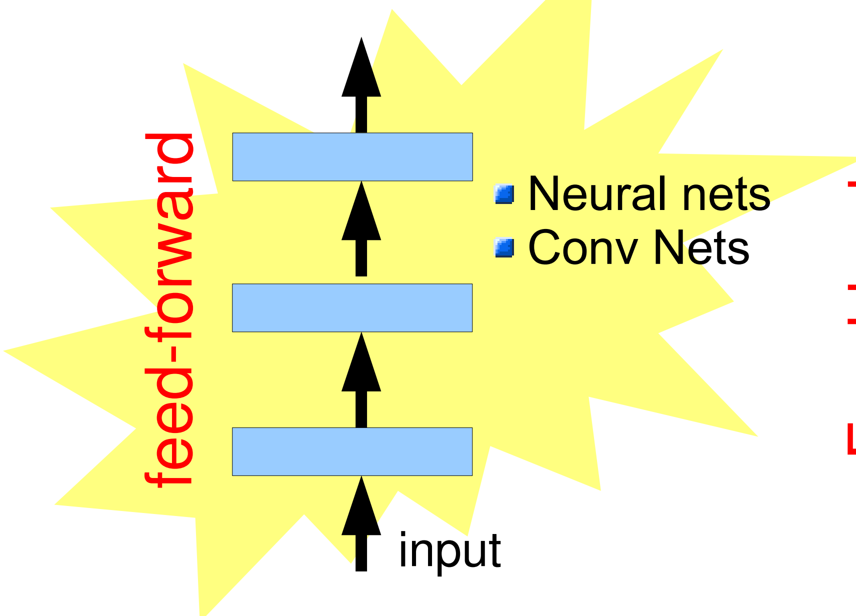
- Stacked Auto-encoders
- DBM



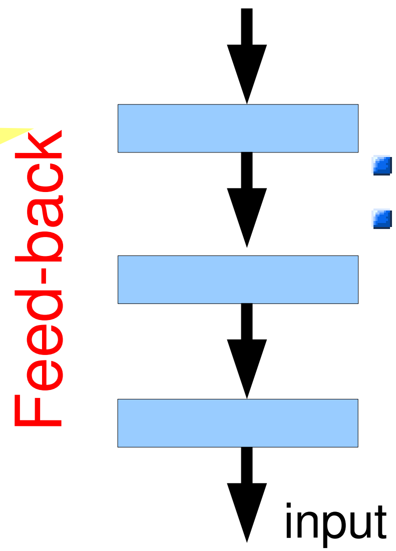
- Recurrent Neural nets
- Recursive Nets
- LISTA

Deep Learning is B I G

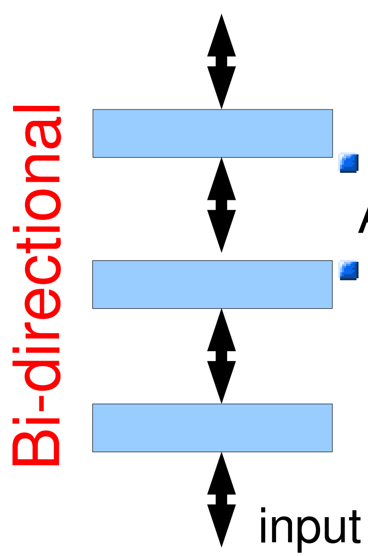
- Main types of deep architectures



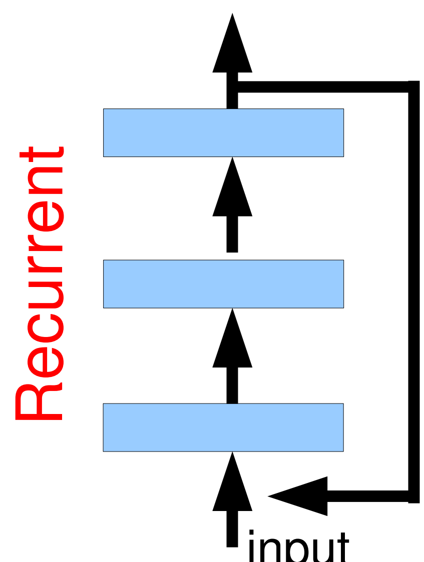
- Neural nets
- Conv Nets



- Hierar. Sparse Coding
- Deconv Nets



- Stacked Auto-encoders
- DBM



- Recurrent Neural nets
- Recursive Nets
- LISTA

Deep Learning is B I G

- Main types of learning protocols
  - Purely supervised
    - Backprop + SGD
    - Good when there is lots of labeled data.
  - Layer-wise unsupervised + superv. linear classifier
    - Train each layer in sequence using regularized auto-encoders or RBMs
    - Hold fix the feature extractor, train linear classifier on features
    - Good when labeled data is scarce but there is lots of unlabeled data.
  - Layer-wise unsupervised + supervised backprop
    - Train each layer in sequence
    - Backprop through the whole system
    - Good when learning problem is very difficult.

Deep Learning is B I G

- Main types of learning protocols
  - Purely supervised
    - Backprop + SGD
    - Good when there is lots of labeled data.
  - Layer-wise unsupervised + superv. linear classifier
    - Train each layer in sequence using regularized auto-encoders or RBMs
    - Hold fix the feature extractor, train linear classifier on features
    - Good when labeled data is scarce but there is lots of unlabeled data.
  - Layer-wise unsupervised + supervised backprop
    - Train each layer in sequence
    - Backprop through the whole system
    - Good when learning problem is very difficult.

Outline

- Theory: Energy-Based Models
  - Energy function
  - Loss function
- Examples:
    - Supervised learning: neural nets
    - Supervised learning: convnets
    - Unsupervised learning: sparse coding
    - Unsupervised learning: gated MRF
- Other examples
- Practical tricks

Energy-Based Models: Energy Function

- Energy:

$$E(y;\theta) \quad \text{or} \quad E(y;x,\theta)$$

unsupervised

supervised

LeCun et al. “Tutorial on Energy-based learning ...” Predicting Structure Data 2006

Ranzato et al. “A unified energy-based framework for unsup. learning” AISTATS 2007

Energy-Based Models: Energy Function

- Energy:

$$E(y;\theta) \quad \text{or} \quad E(y;x,\theta)$$

unsupervised   supervised

y can be $\begin{cases} \text{discrete} \\ \text{continuous} \end{cases}$

LeCun et al. “Tutorial on Energy-based learning ...” Predicting Structure Data 2006
Ranzato et al. “A unified energy-based framework for unsup. learning” AISTATS 2007

Energy-Based Models: Energy Function

- Energy:

$$E(y;\theta) \quad \text{or} \quad E(y;x,\theta)$$

unsupervised

supervised

y can be $\begin{cases} \text{discrete} \\ \text{continuous} \end{cases}$

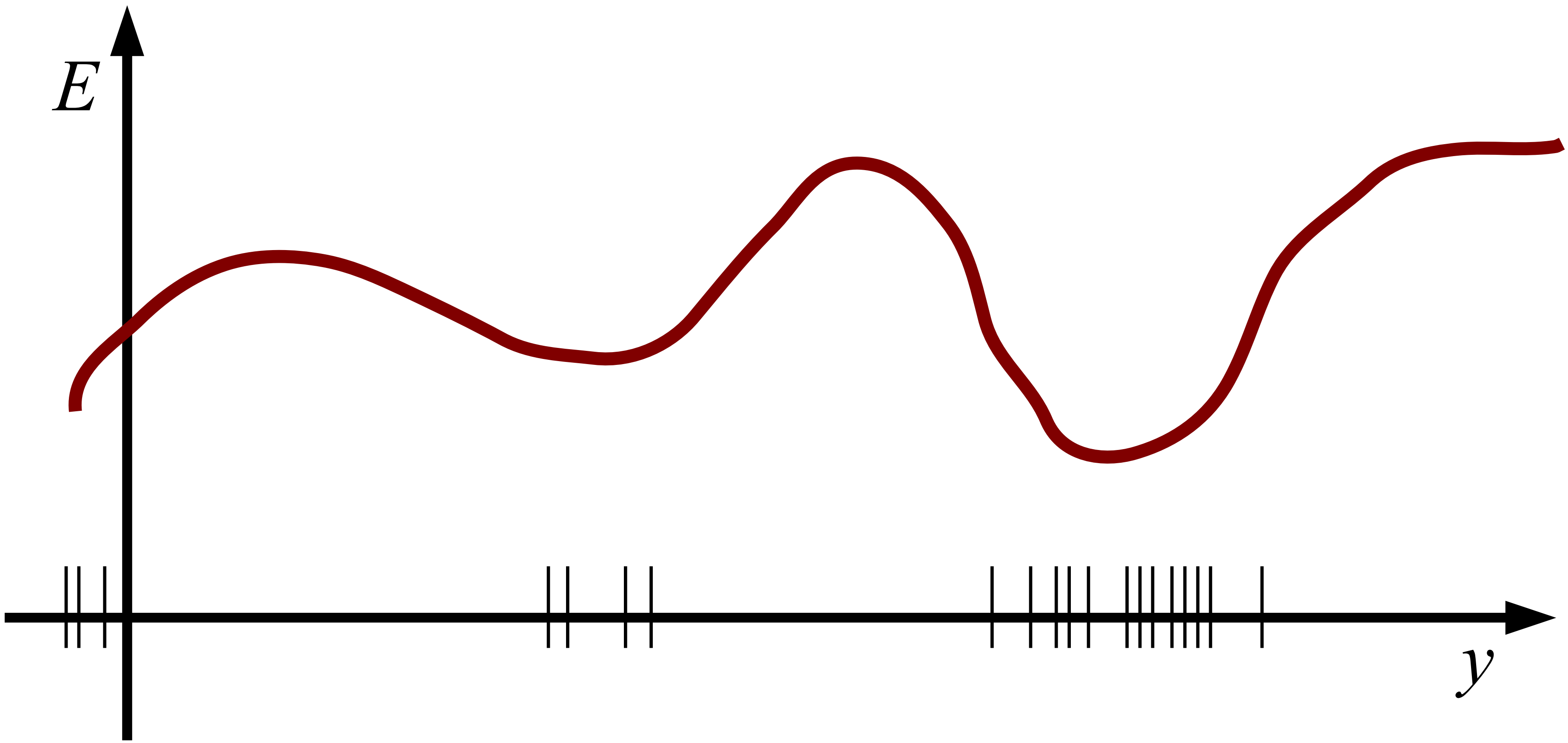
We will refer to the unsupervised/continous case, but much of the following applies to the other cases as well.

LeCun et al. “Tutorial on Energy-based learning ...” Predicting Structure Data 2006

Ranzato et al. “A unified energy-based framework for unsup. learning” AISTATS 2007

Energy-Based Models: Energy Function

- Energy should be lower for desired output



Ranzato

Energy-Based Models: Energy Function

- Make energy lower at the desired output

Energy-Based Models: Energy Function

- Make energy lower at the desired output

Energy-Based Models: Energy Function

- Examples of energy function:
  - PCA

$$E(y) = \| y - W W^T y \|_2^2$$

  - Linear binary classifier $y \in \{-1, +1\}$

$$E(y; x) = -y(W^T x)$$

  - Neural net binary classifier

$$E(y; x) = -y(W_2^T f(x; W_1))$$

Energy-Based Models: Loss Function

- Loss is a function of the energy
- Minimizing the loss over the training set yields the desired energy landscape.

$$\theta^* = min_\theta \sum_p L(E(y^p;\theta))$$

- Examples of loss function:
  - PCA

$$E(y) = \| y - W W^T y \|_2^2 \qquad L(E(y)) = E(y)$$

  - Logistic regression classifier

$$E(y;x) = -y(W^T x)$$
$$L(E(y;x)) = \log(1 + \exp(E(y;x)))$$

Energy-Based Models: Loss Function

- Loss is a function of the energy
- Minimizing the loss over the training set yields the desired energy landscape.

$$\theta^* = min_\theta \sum_p L(E(y^p; \theta))$$

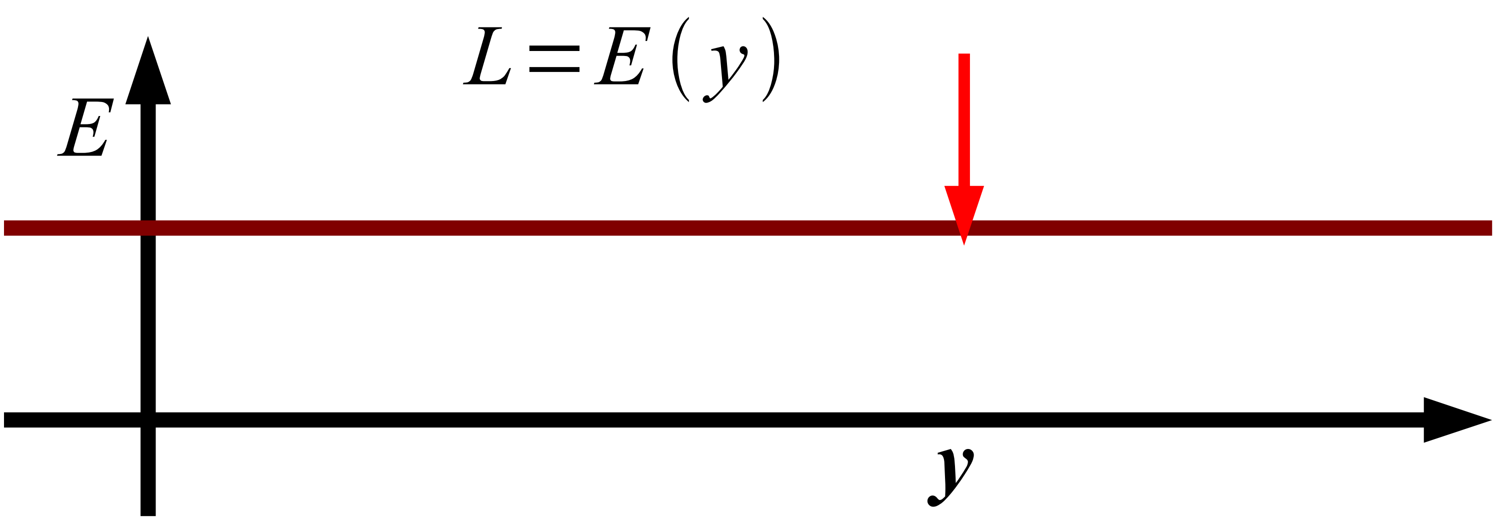
How to design loss good functions?

Energy-Based Models: Loss Function

- Loss is a function of the energy
- Minimizing the loss over the training set yields the desired energy landscape.

$$\theta^* = min_\theta \sum_p L(E(y^p;\theta))$$

How to design loss good functions?



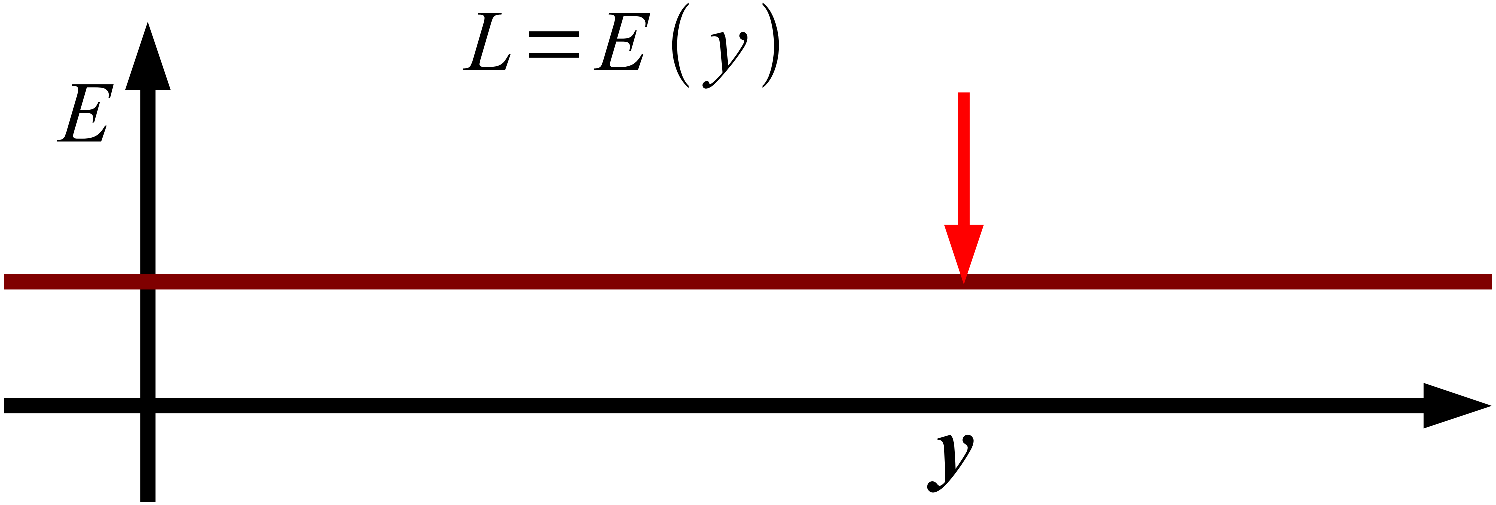
Energy-Based Models: Loss Function

- Loss is a function of the energy
- Minimizing the loss over the training set yields the desired energy landscape.

$$\theta^* = min_\theta \sum_p L(E(y^p;\theta))$$

How to design loss good functions?

$$L = E(y)$$



Energy-Based Models: Loss Function

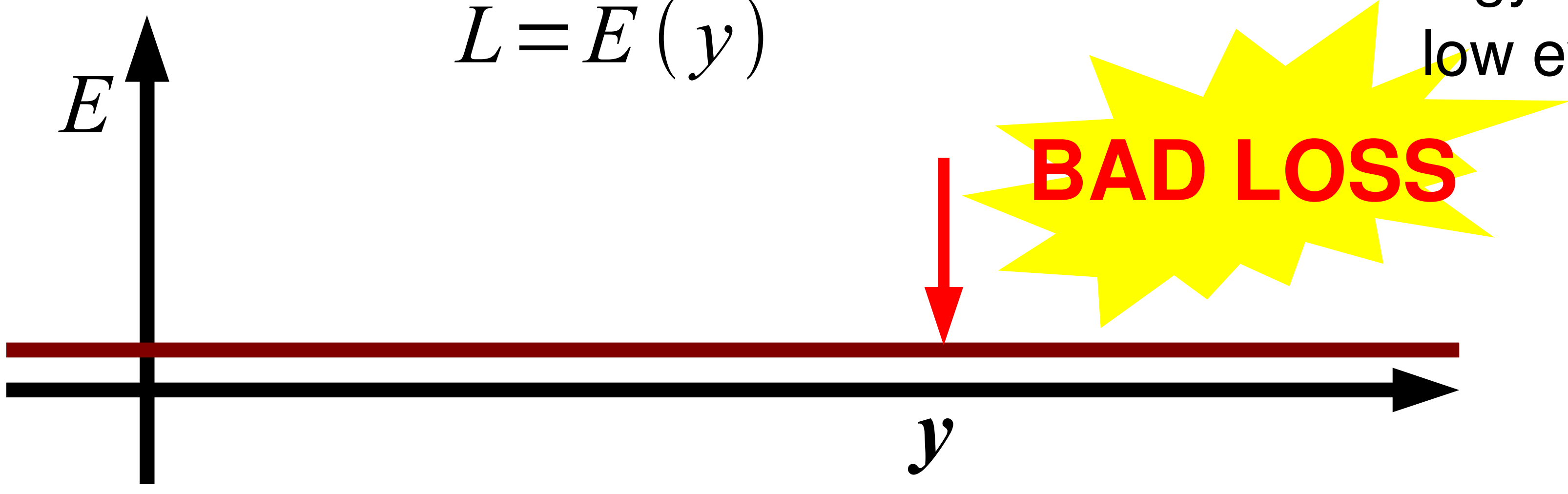
- Loss is a function of the energy
- Minimizing the loss over the training set yields the desired energy landscape.

$$\theta^* = min_\theta \sum_p L(E(y^p;\theta))$$

How to design loss good functions?

$$L = E(y)$$

Energy is degenerate:
low everywhere!



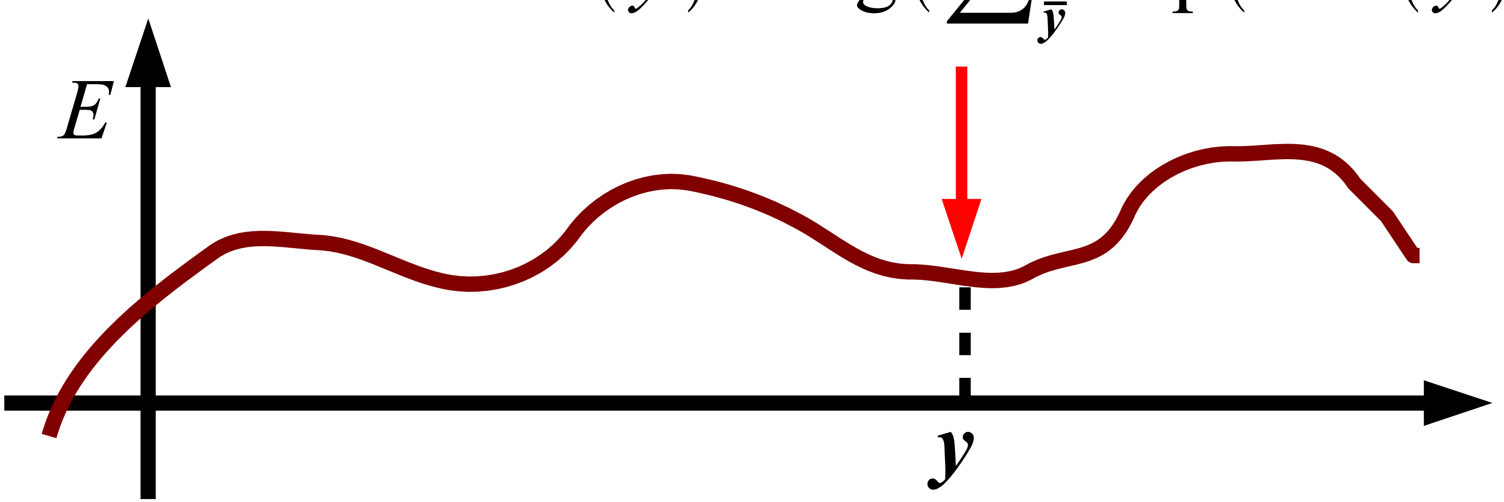
Energy-Based Models: Loss Function

- Loss is a function of the energy
- Minimizing the loss over the training set yields the desired energy landscape.

$$\theta^* = min_\theta \sum_p L(E(y^p;\theta))$$

How to design loss good functions?

$$L = E(y) + \log(\sum_{\bar{y}} \exp(-E(\bar{y})))$$



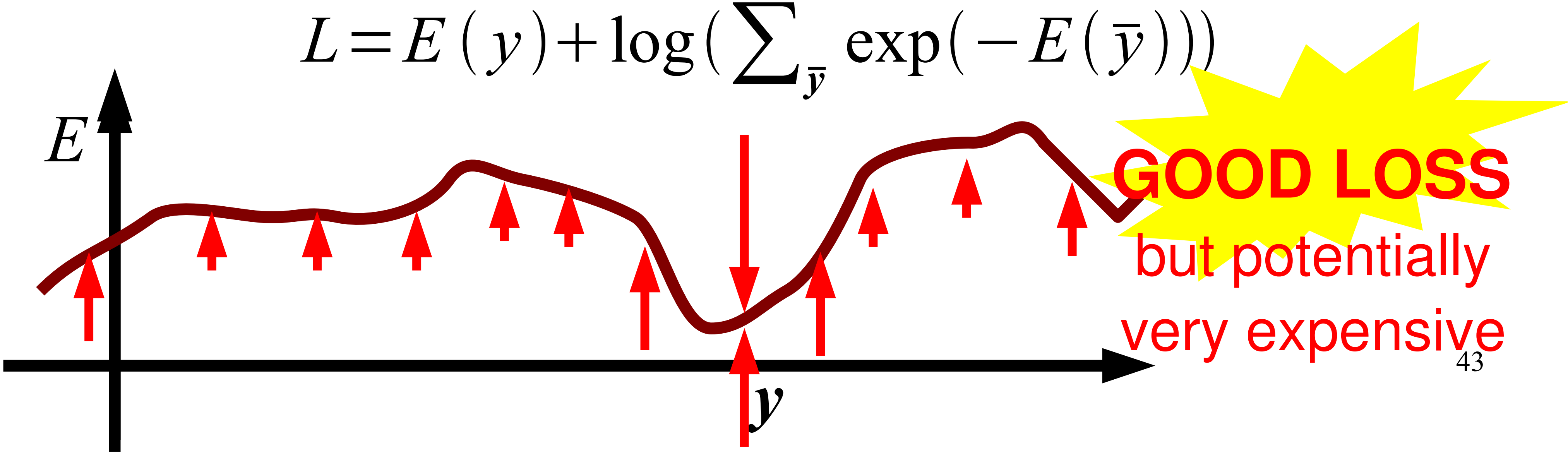
Energy-Based Models: Loss Function

- Loss is a function of the energy
- Minimizing the loss over the training set yields the desired energy landscape.

$$\theta^* = min_\theta \sum_p L(E(y^p; \theta))$$

How to design loss good functions?

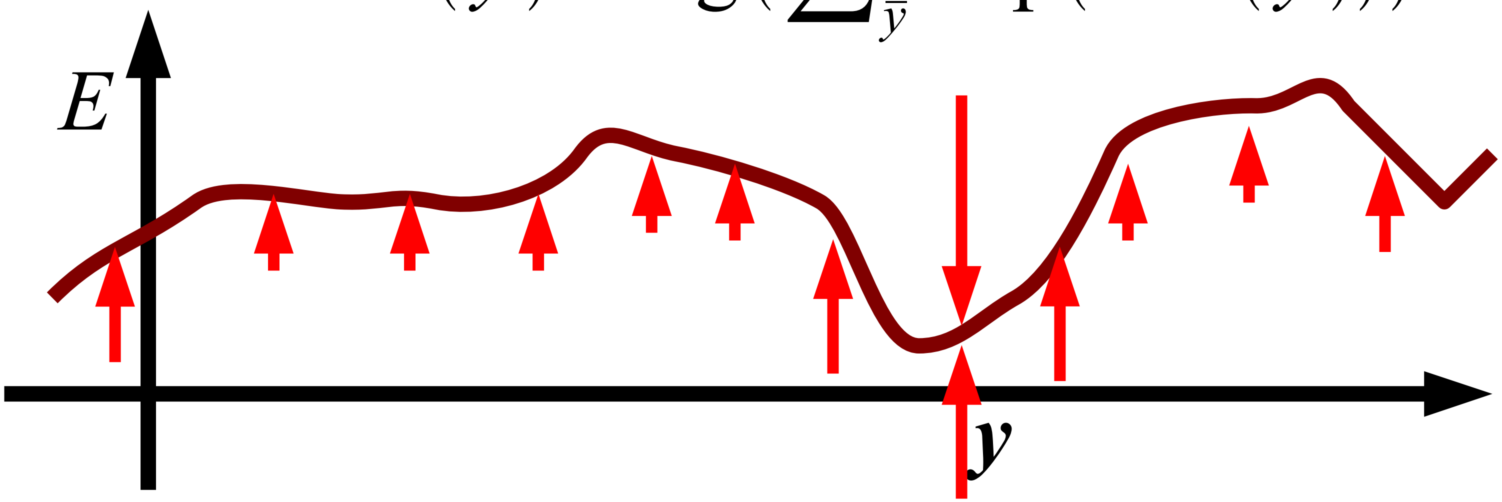
$$L = E(y) + \log\left(\sum_{\bar{y}} \exp(-E(\bar{y}))\right)$$



Strategies to Shape E: #1

- Pull down the correct answer and pull up everywhere else.

$$L = E(y) + \log\left(\sum_{\bar{y}} \exp(-E(\bar{y}))\right)$$



Negative Log-Likelihood

$$p(y) = \frac{e^{-E(y)}}{\sum_u e^{-E(u)}}$$

PROS

It produces calibrated probabilities.

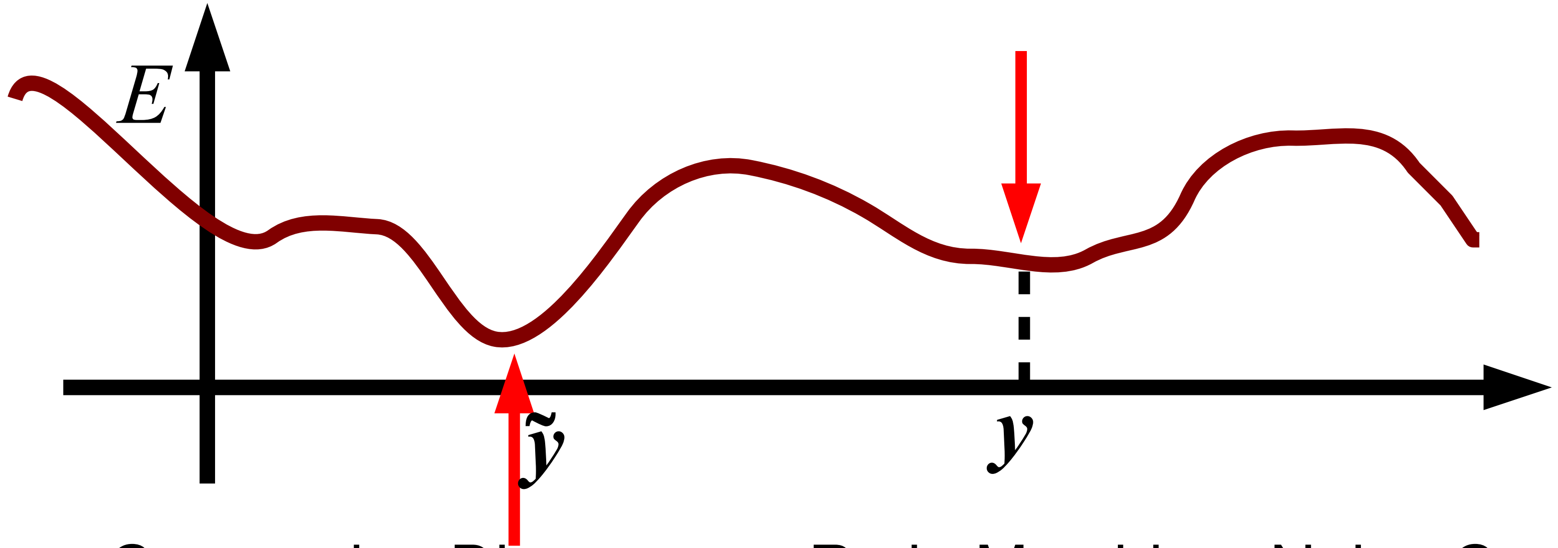
CONS

Expensive to compute when y is discrete and high dimensional.
Generally intractable when y is continuous.

Strategies to Shape E: #2

- Pull down the correct answer and pull up carefully chosen points.

$$L = max(0, m - (E(\bar{y}) - E(y))),\ \ e.g.\ \ \bar{y} = min_{\tilde{y} \neq y} E(\tilde{y})$$



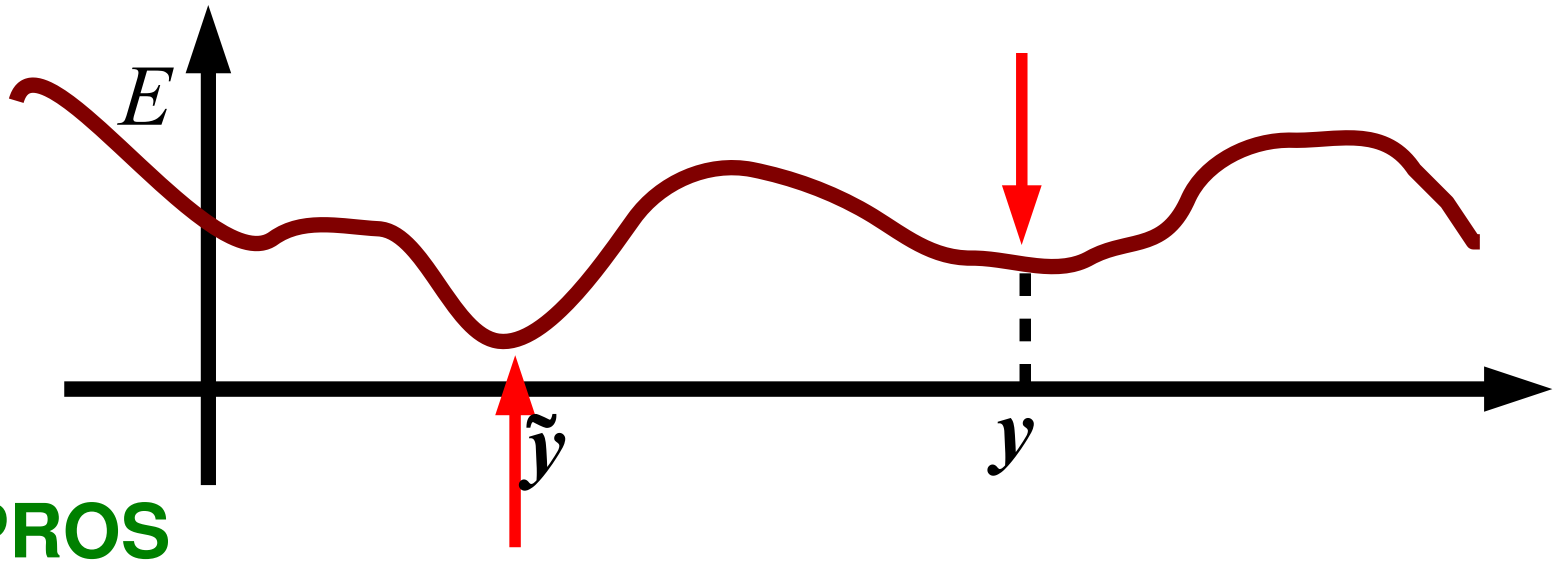
E.g.: Contrastive Divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow...

Hinton et al. "A fast learning algorithm for DBNs" Neural Comp. 2008
Hyvarinen "Some extensions of score matchine" Comp Stats 2007
Gutmann et al. "Noise contrastive estimation of unnormalized..." JMLR 2012
Sohl-Dickstein et al. "Minimum probability flow learning" ICML 2011

Strategies to Shape E: #2

- Pull down the correct answer and pull up carefully chosen points.

$$L = max(0, m - (E(\bar{y}) - E(y))),\ \ e.g.\ \ \bar{y} = min_{\tilde{y} \neq y} E(\tilde{y})$$



PROS

Efficient.

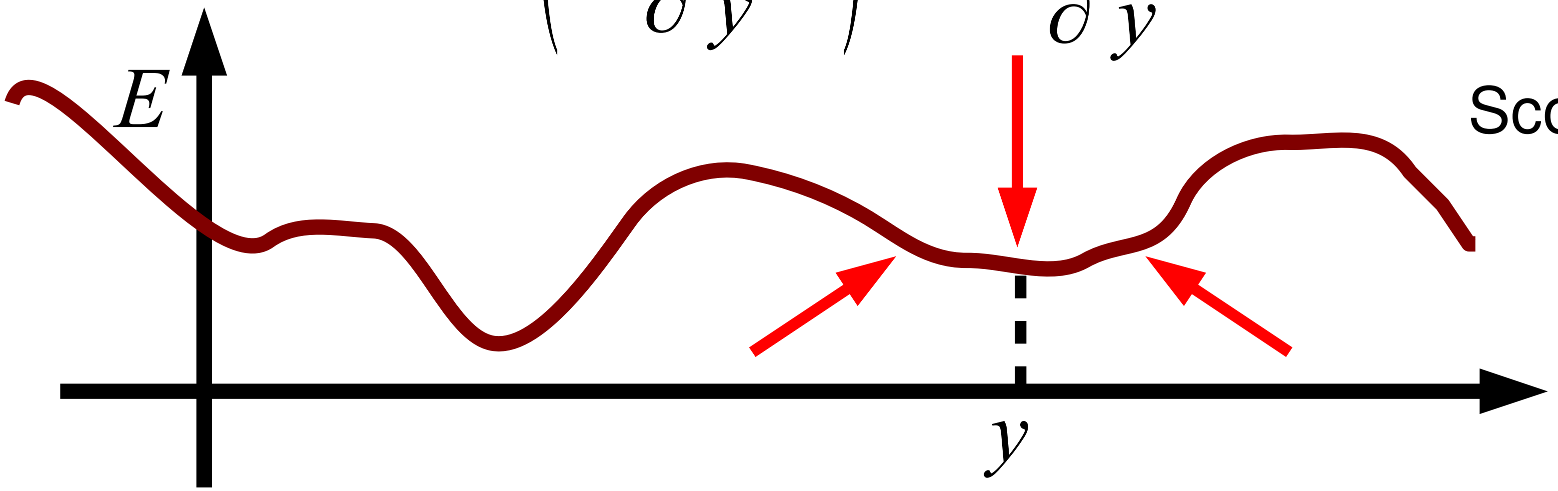
CONS

The criterion to pick where to pull up is tricky (overall in high dimensional spaces): trades-off computational and statistical efficiency.

Strategies to Shape E: #3

- Pull down the correct answer and increase local curvature.

$$L = \left(\frac{\partial E(y)}{\partial y} \right)^2 + \frac{\partial^2 E(y)}{\partial y^2}$$



Score Matching

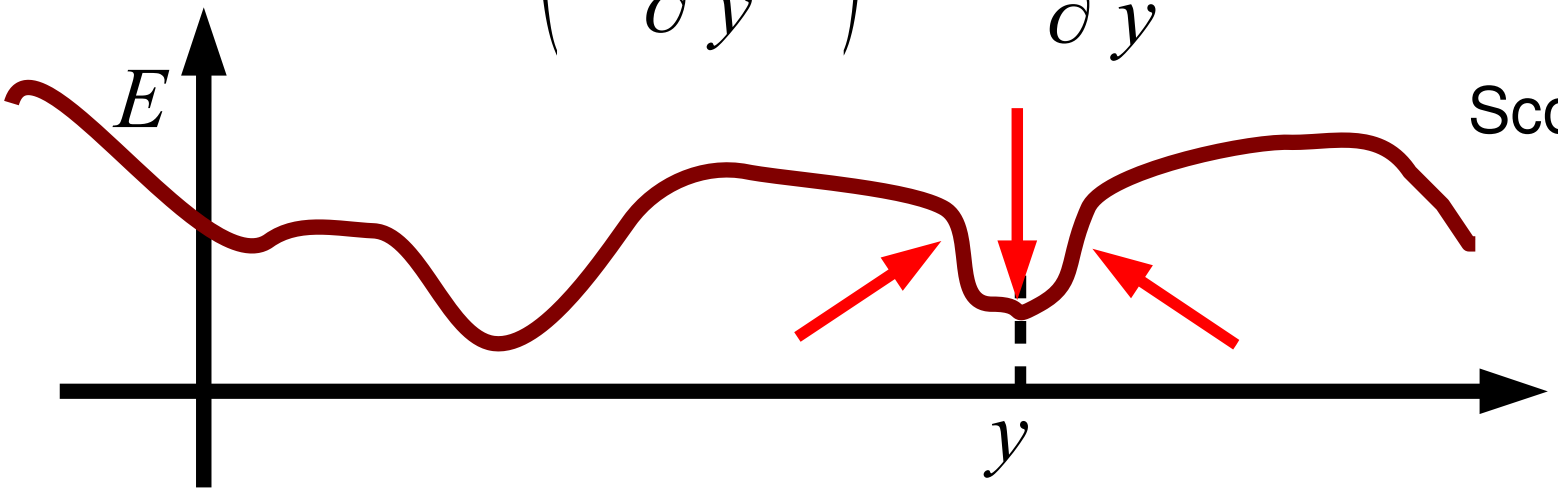
Hyvarinen “Estimation of non-normalized statistical models using score matching” JMLR 2005

Strategies to Shape E: #3

- Pull down the correct answer and increase local curvature.

$$L = \left(\frac{\partial E(y)}{\partial y} \right)^2 + \frac{\partial^2 E(y)}{\partial y^2}$$

Score Matching



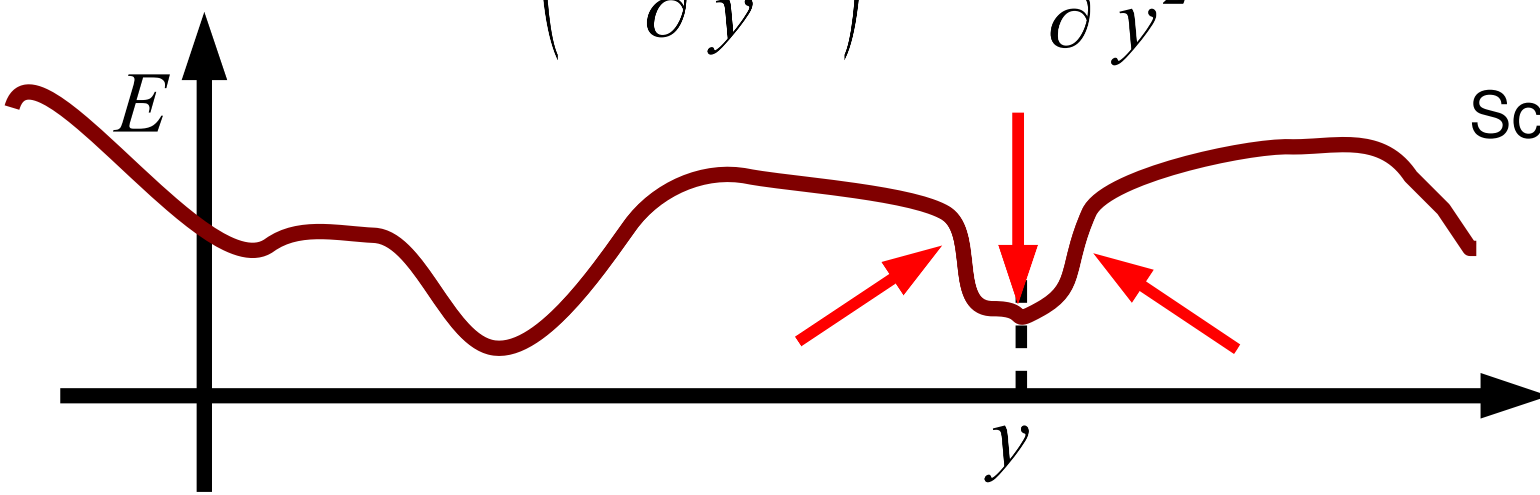
Hyvarinen "Estimation of non-normalized statistical models using score matching" JMLR 2005

Strategies to Shape E: #3

- Pull down the correct answer and increase local curvature.

$$L = \left(\frac{\partial E(y)}{\partial y} \right)^2 + \frac{\partial^2 E(y)}{\partial y^2}$$

Score Matching



PROS

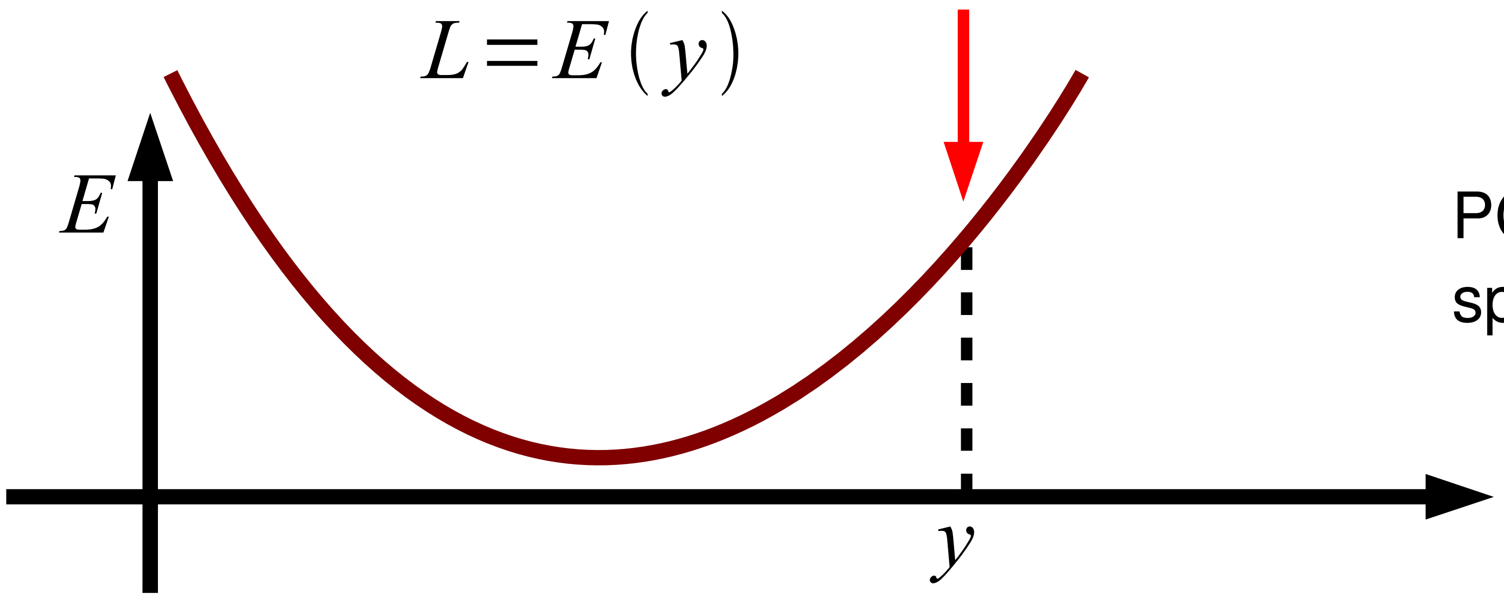
Efficient in continuous but not too high dimensional spaces.

CONS

Very complicated to compute and not practical in very high dimensional spaces. Not applicable in discrete spaces.

Strategies to Shape E: #4

- Pull down correct answer and have global constrain on the energy: only few minima exist



PCA, ICA,
sparse coding,...

PROS

Efficient in continuous, high dimensional spaces.

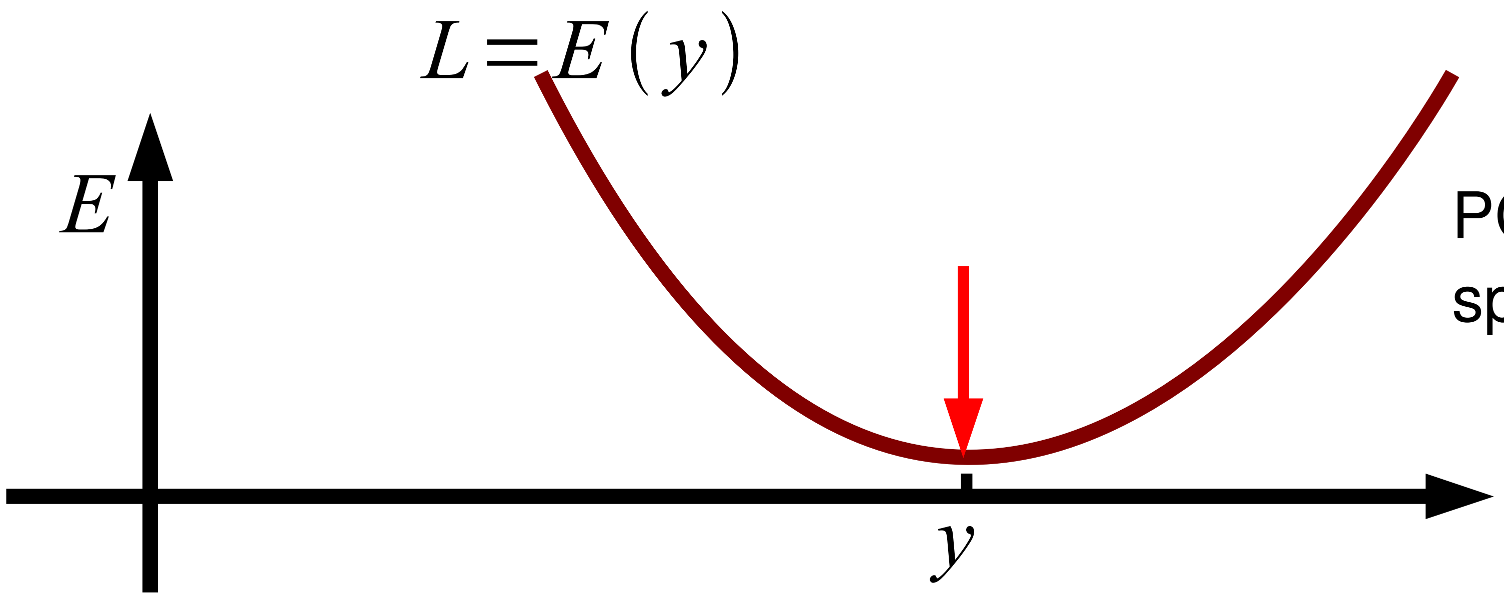
CONS

Need to design good global constraints. Used in unsup. learning only.

Ranzato et al. “A unified energy-based framework for unsup. learning” AISTATS 2007

Strategies to Shape E: #4

- Pull down correct answer and have global constrain on the energy: only few minima exist



PCA, ICA, sparse coding,...

PROS

Efficient in continuous, high dimensional spaces.

CONS

Need to design good global constraints. Used in unsup. learning only.

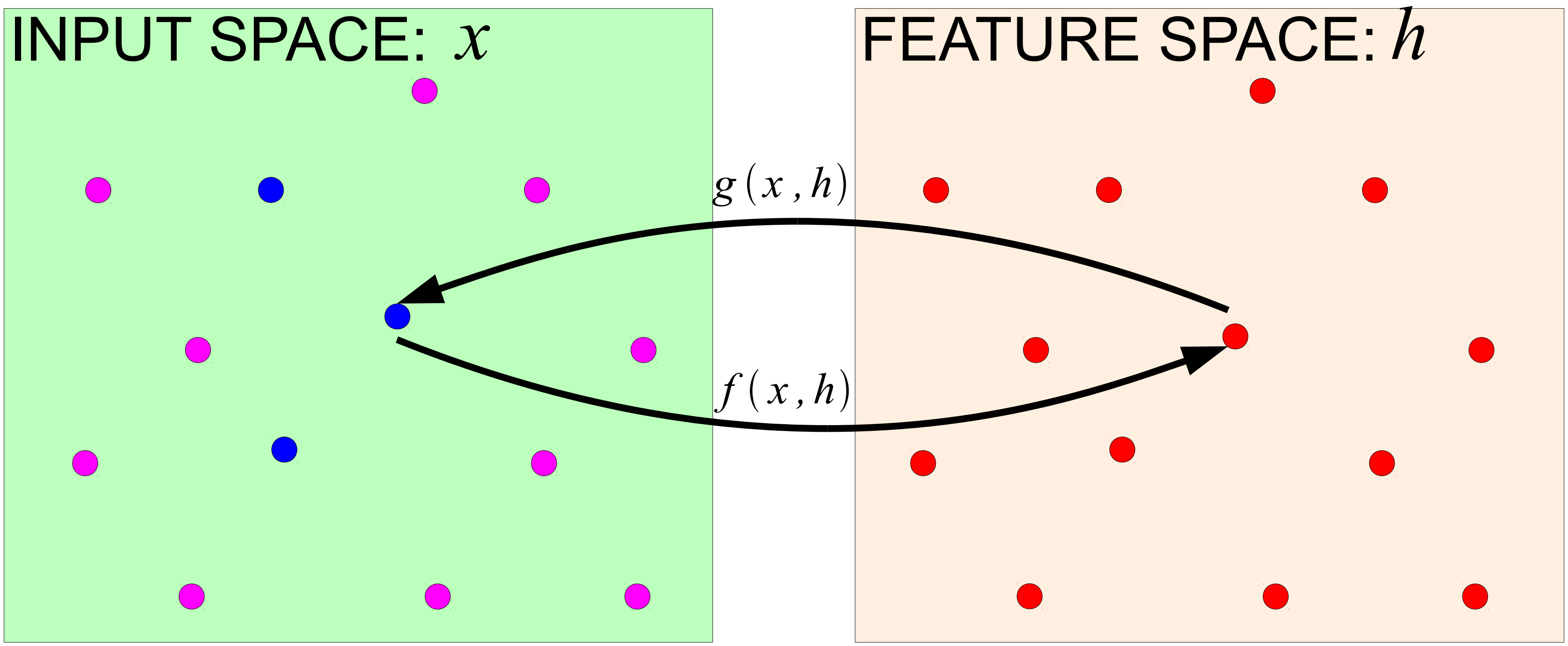
Ranzato et al. “A unified energy-based framework for unsup. learning” AISTATS 2007

Strategies to Shape E: #4

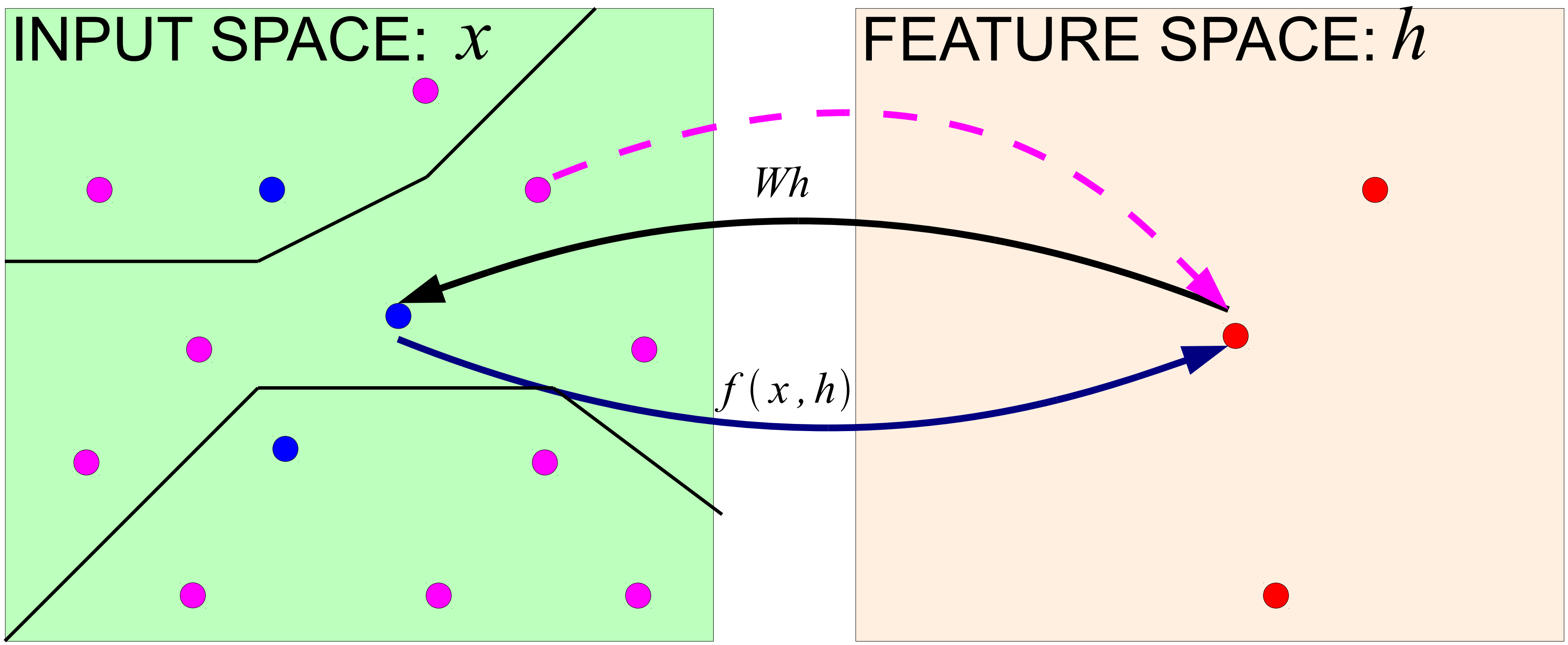
- Pull down correct answer and have global constrain on the energy: only few minima exist

- Typical methods (unsup. learning):
  - Use compact internal representation (PCA)
  - Have finite number of internal states (K-Means)
  - Use sparse codes (ICA, sparse coding)

- training sample
- input data point which is not a training sample
- feature (code)



E.g. K-means: $\boldsymbol{h}$ is 1-of-N.

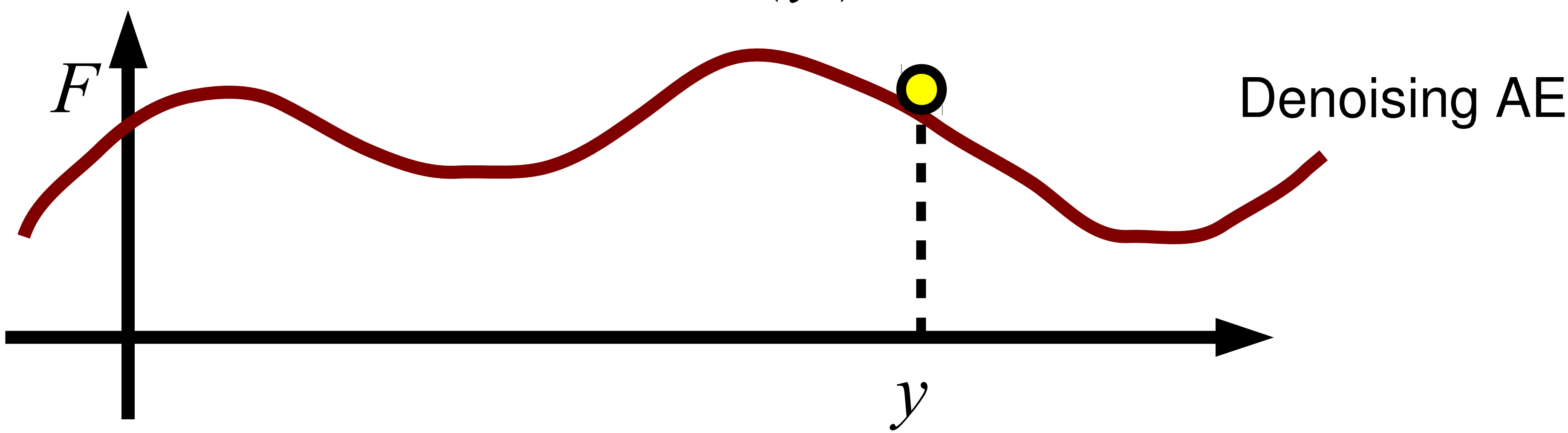


Since there are very few "codes" available and the energy (MSE) is minimized on the training set, the energy must be higher elsewhere.

Strategies to Shape E: #5

- Make the observed y an attractor state of some energy function. Need only to define a dynamics.

$$L = E(y)$$



E.g.:

$$L = \frac{1}{2} \| y - \hat{y} \|^2$$

Training samples must be stable points (local minima) and attractors.

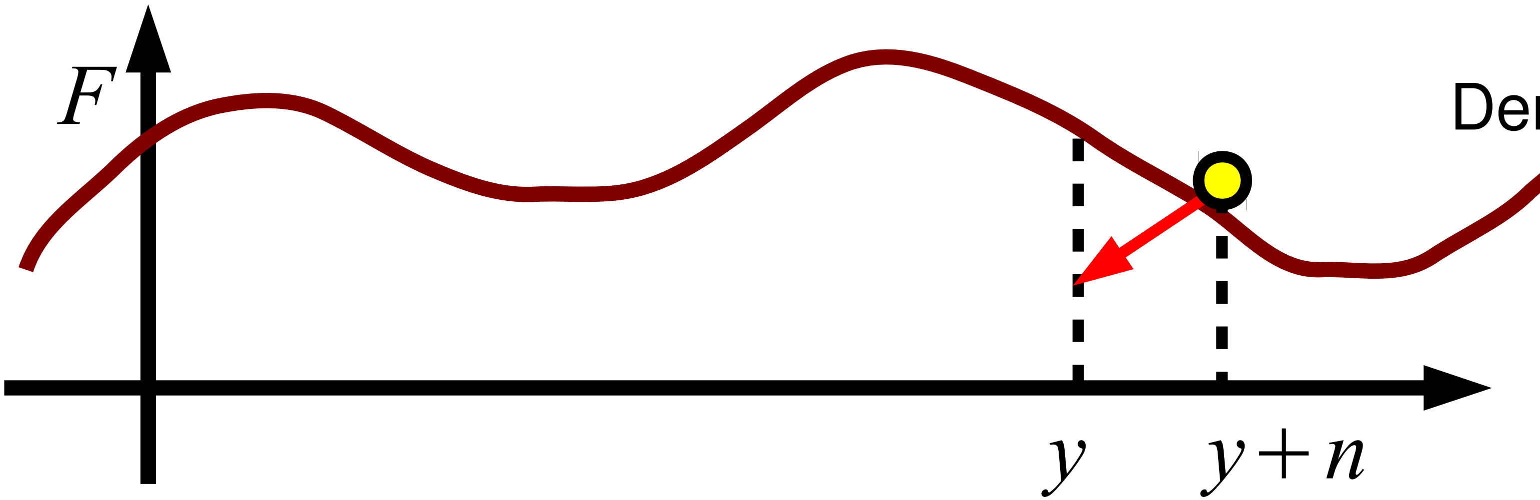
$$\hat{y} = W_2 \sigma(W_1 y + n), n \in N(0, I)$$

Vincent et al. “Extracting and composing robust features with denoising autoencoders” ICML 2008

Strategies to Shape E: #5

- Make the observed y an attractor state of some energy function. Need only to define a dynamics.

$$L = E(y)$$



E.g.:

$$L = \frac{1}{2} \|y - \hat{y}\|^2$$

$$\hat{y} = W_2 \sigma(W_1 y + n),\ n \in N(0, I)$$

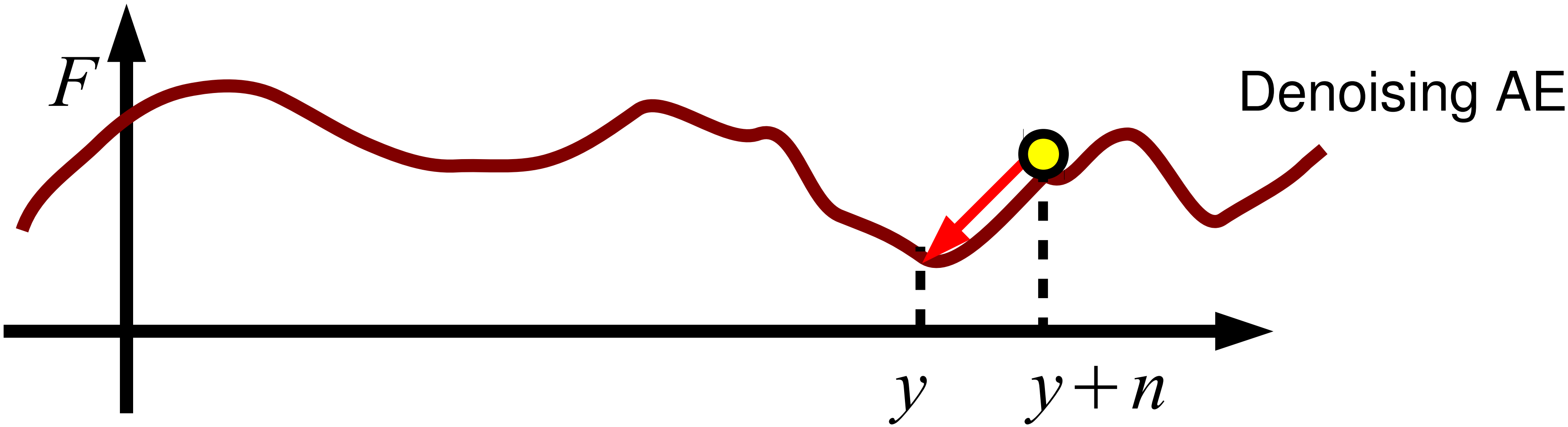
Training samples must be stable points (local minima) and attractors.

Kamyshanska et al. “On autoencoder scoring” ICML 2013

Strategies to Shape E: #5

- Make the observed y an attractor state of some energy function. Need only to define a dynamics.

$$L = E(y)$$



Training samples must be stable points (local minima) and attractors.

E.g.:

$$L = \frac{1}{2} \| y - \hat{y} \|^2$$

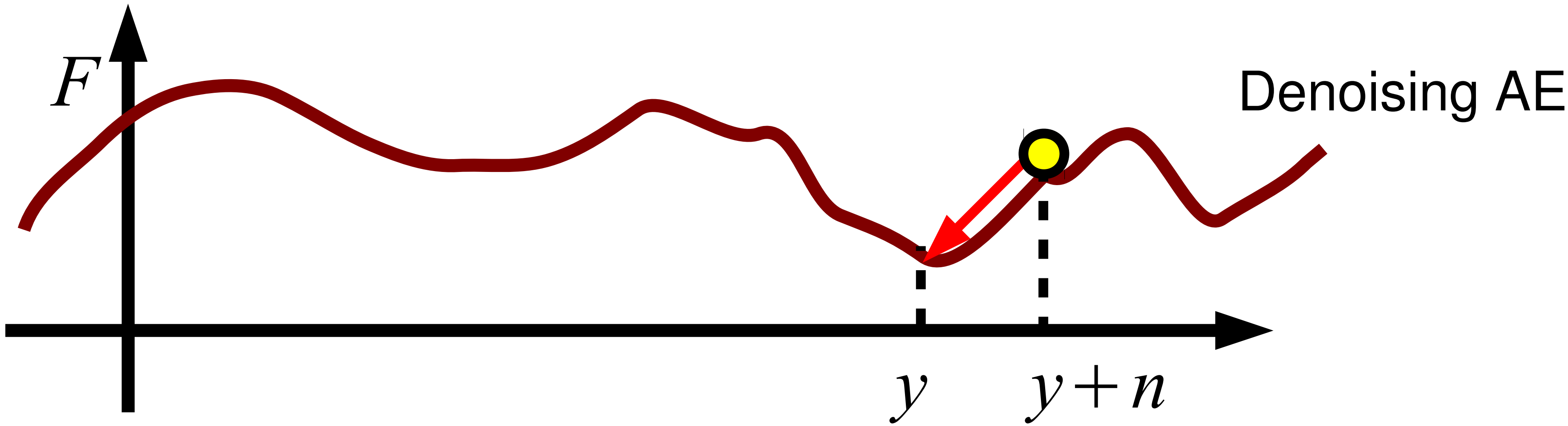
$$\hat{y} = W_2 \sigma (W_1 y + n), \; n \in N(0, I)$$

Kamyshanska et al. “On autoencoder scoring” ICML 2013

Strategies to Shape E: #5

- Make the observed y an attractor state of some energy function. Need only to define a dynamics.

$$L = E(y)$$



PROS

Efficient in high dimensional spaces.

CONS

Need to pick noise distribution. May need to pick many noisy points.

Kamyshanska et al. “On autoencoder scoring” ICML 2013

Loss: summary

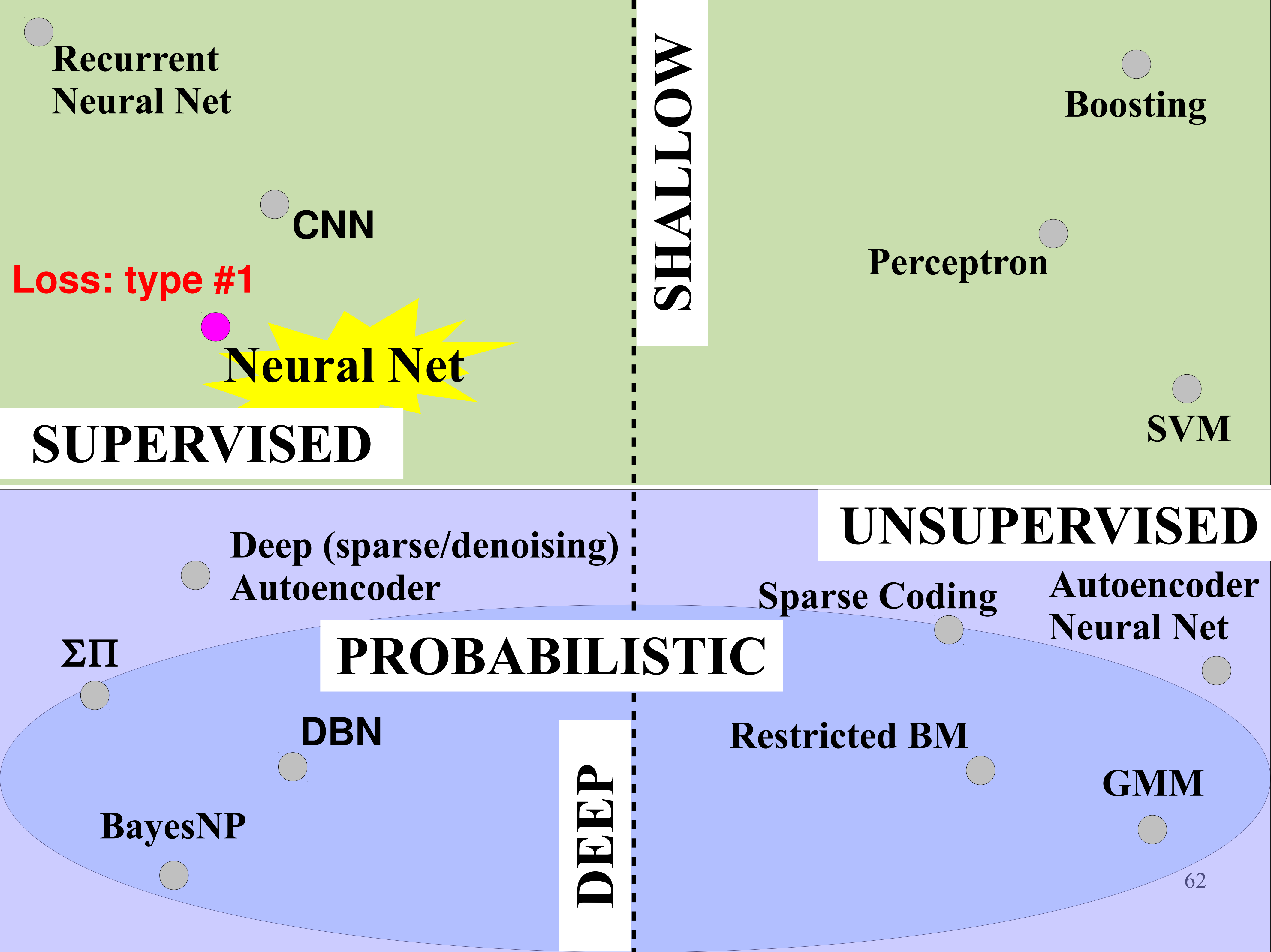
- Goal of loss: make energy lower for correct answer.
- Different losses choose differently how to “pull-up”
  - Pull-up all points
  - Pull up one or a few points only
  - Make observations minima & increase curvature
  - Add global constraints/penalties on internal states
  - Define a dynamics with observations at the minima
- Choice of loss depends on desired landscape, computational budget, domain of input (discrete/continouus), task, etc.

Final Notes on EBMs

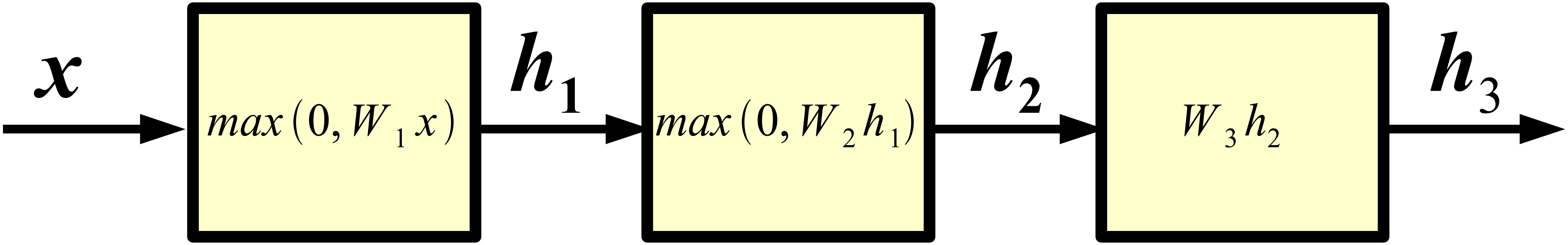
- EBMs apply to any predictor (shallow & deep).
- EBMs subsume graphical models (e.g., use strategy #1 or #2 to “pull-up”).
- EBM is general framework to design good loss functions.

Outline

- Theory: Energy-Based Models
  - Energy function
  - Loss function
- Examples:
  - Supervised learning: neural nets
  - Supervised learning: convnets
  - Unsupervised learning: sparse coding
  - Unsupervised learning: gated MRF
- Other examples
- Practical tricks

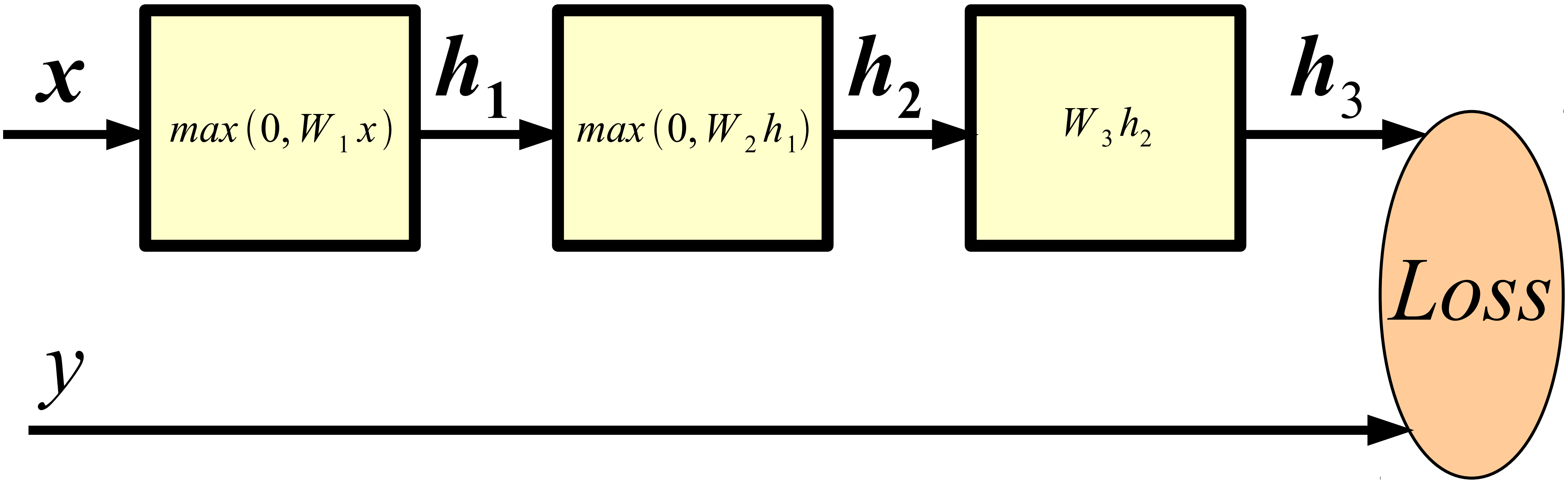

Recurrent Neural Net
CNN
Loss: type #1
Neural Net
SUPERVISED
SHALLOW
Boosting
Perceptron
SVM
UNSUPERVISED
Deep (sparse/denoising) Autoencoder
Sparse Coding
Autoencoder Neural Net
PROBABILISTIC
ΣΠ
DBN
Restricted BM
GMM
BayesNP
DEEP

Neural Nets



NOTE: In practice, any (a.e. differentiable) non-linear transformation can be used.

Loss

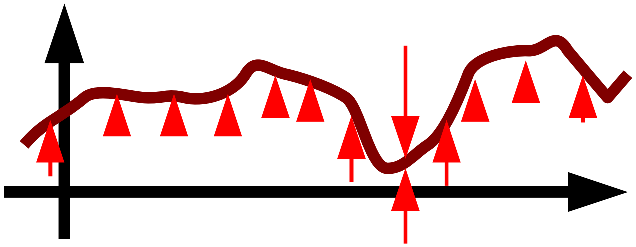


$$E(y;x) = -y\, h_3$$

$$L(E(y;x)) = \log(1 + \exp(E(y;x)))$$

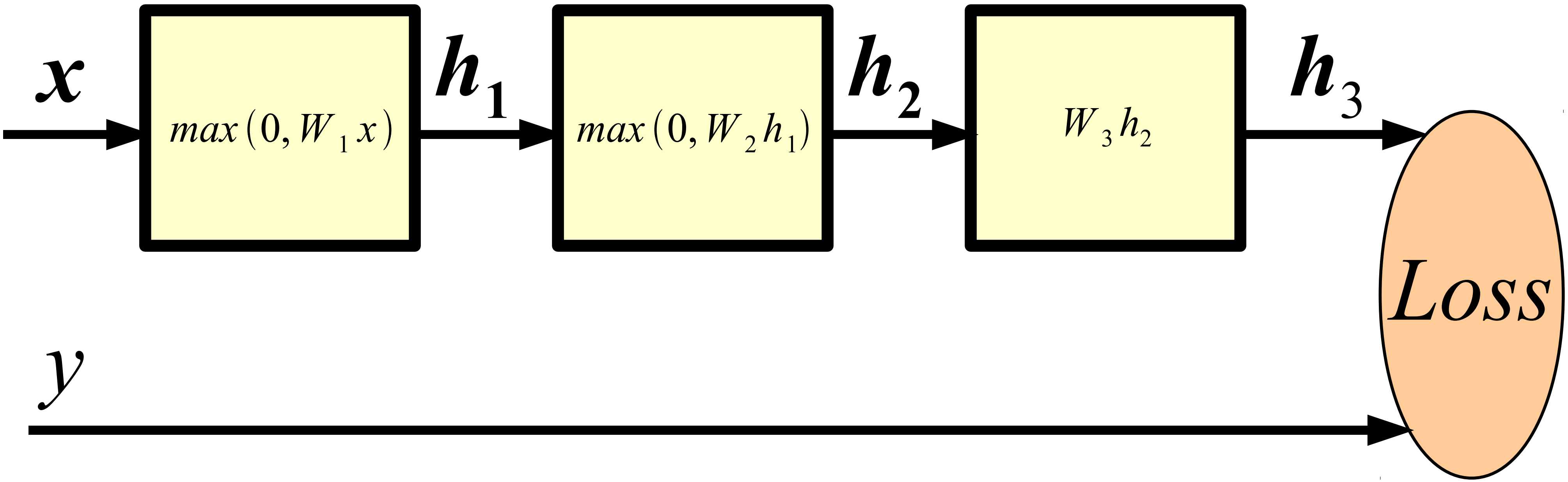
$$\{W_1^*, W_2^*, W_3^*\} = arg\ min_{W_1, W_2, W_3} L(E(y;x))$$

Loss: type #1



Ranzato

Loss



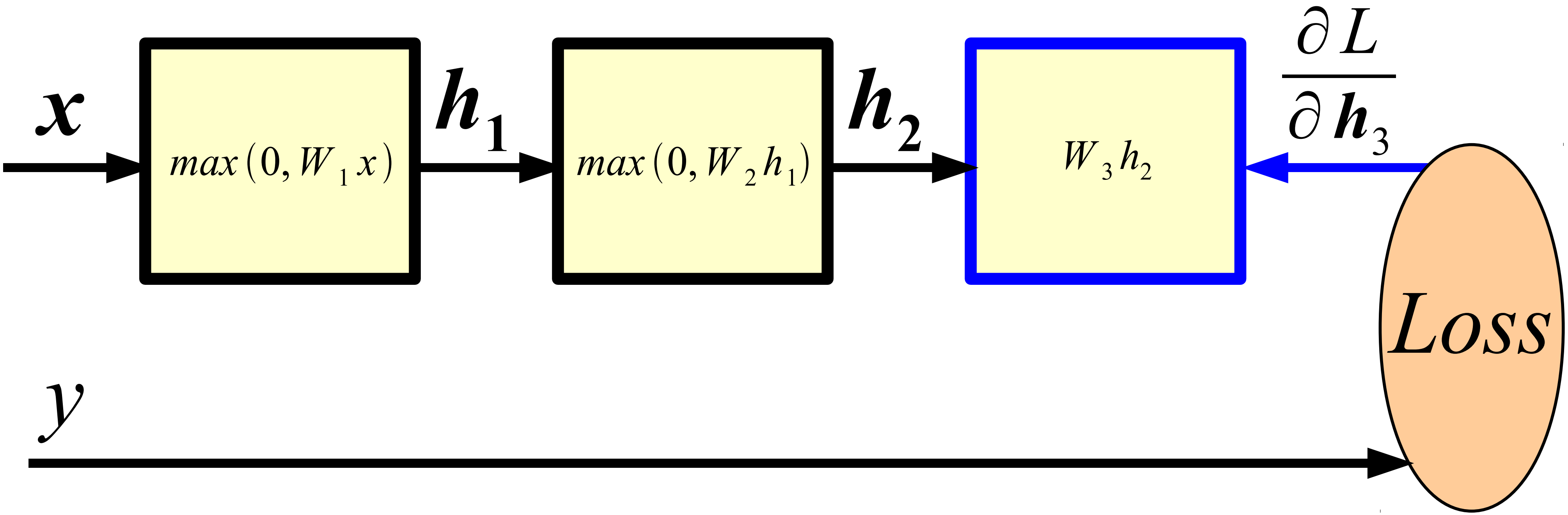
Q.: how to tune the parameters to decrease the loss?

If loss is (a.e.) differentiable we can compute gradients.

We can use chain-rule, a.k.a. **back-propagation**, to compute the gradients w.r.t. parameters at the lower layers.

Rumelhart et al. “Learning internal representations by back-propagating..” Nature 1986

Backward Propagation

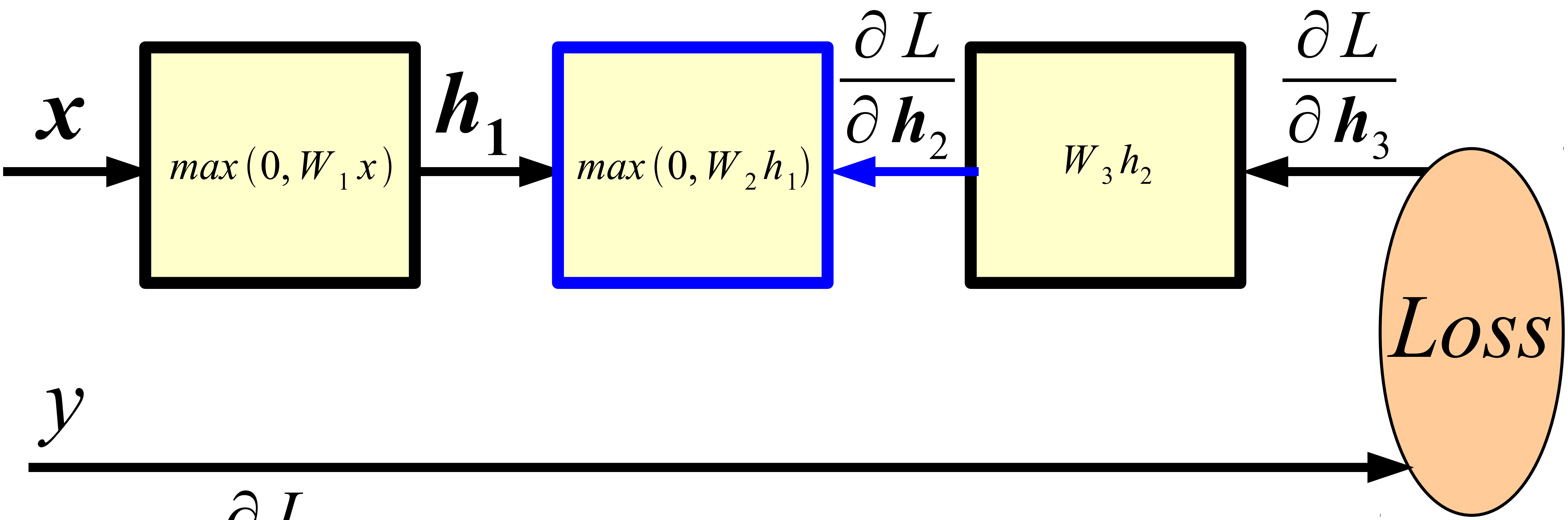


Given $\partial L / \partial \boldsymbol{h}_3$ and assuming we can easily compute the Jacobian of each module, we have:

$$\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial \boldsymbol{h}_3} \frac{\partial \boldsymbol{h}_3}{\partial W_3} \qquad \frac{\partial L}{\partial \boldsymbol{h}_2} = \frac{\partial L}{\partial \boldsymbol{h}_3} \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{h}_2}$$

$$\frac{\partial L}{\partial W_3} = (\sigma(\boldsymbol{h}_3) - y) \ \boldsymbol{h}_2^T \qquad \frac{\partial L}{\partial \boldsymbol{h}_2} = W_3^T (\sigma(\boldsymbol{h}_3) - y)$$

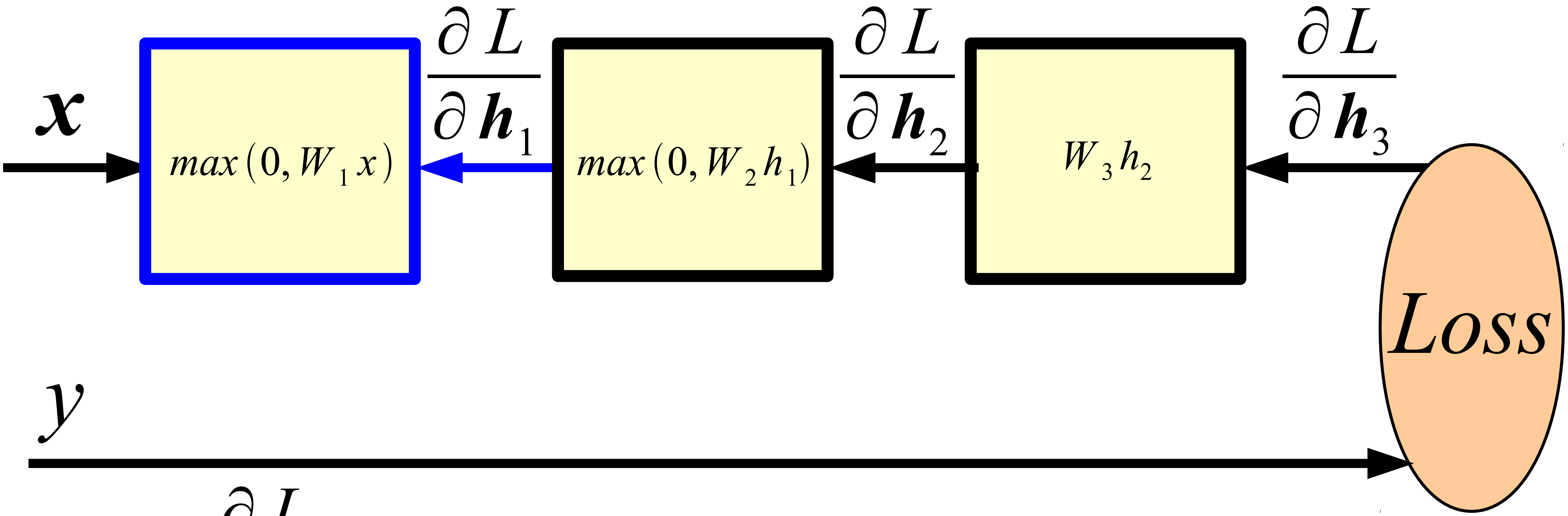
Backward Propagation



Given $\frac{\partial L}{\partial \boldsymbol{h}_2}$ we can compute now:

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial \boldsymbol{h}_2} \frac{\partial \boldsymbol{h}_2}{\partial W_2} \qquad \frac{\partial L}{\partial \boldsymbol{h}_1} = \frac{\partial L}{\partial \boldsymbol{h}_2} \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{h}_1}$$

Backward Propagation



Given $\frac{\partial L}{\partial \boldsymbol{h}_1}$ we can compute now:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial \boldsymbol{h}_1} \frac{\partial \boldsymbol{h}_1}{\partial W_1}$$

Optimization

Stochastic Gradient Descent (on mini-batches):

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}, \eta \in R$$

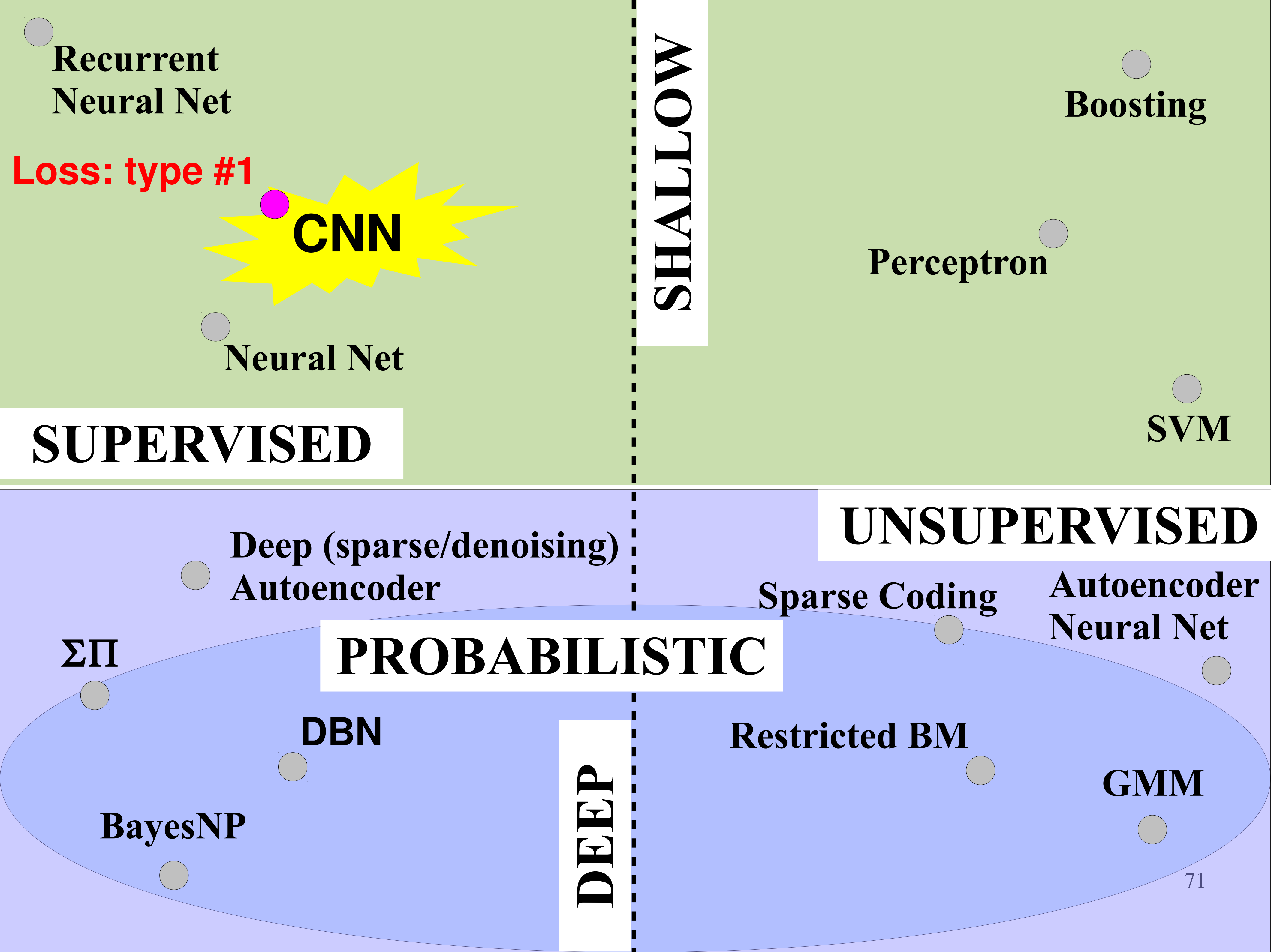
Stochastic Gradient Descent with Momentum:

$$\theta \leftarrow \theta - \eta \Delta$$

$$\Delta \leftarrow 0.9 \Delta + \frac{\partial L}{\partial \theta}$$

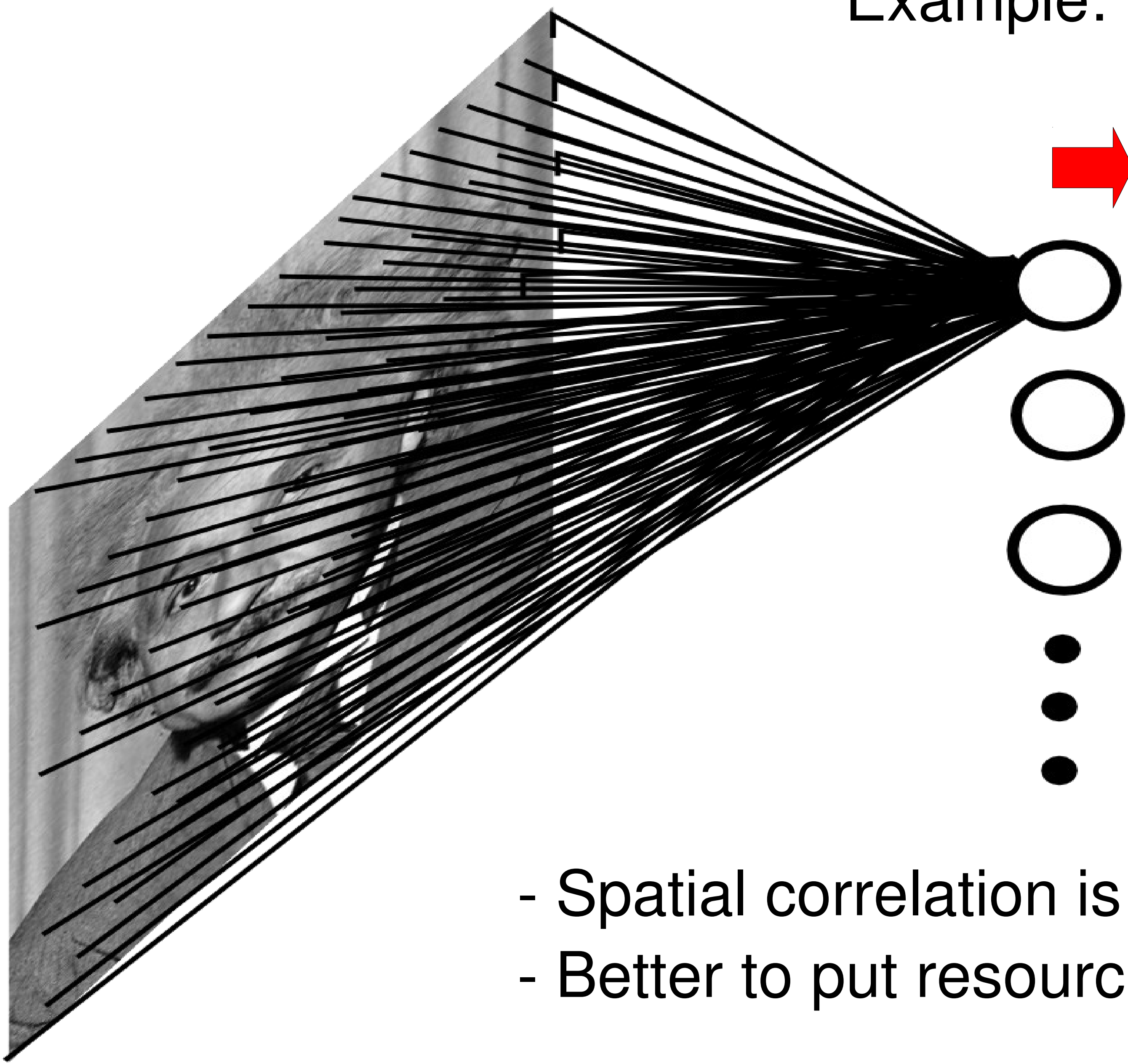
Toy Code (Matlab): Neural Net Trainer

```
% F-PROP
for i = 1 : nr_layers - 1
  [h{i}  jac{i}]  =  nonlinearity(W{i} * h{i-1} +  b{i});
end
h{nr_layers-1}  =  W{nr_layers-1} * h{nr_layers-2}  +   b{nr_layers-1};
prediction  =  softmax(h{l-1});


% CROSS ENTROPY LOSS
loss  =  -  sum(sum(log(prediction)  .*  target)) / batch_size;


% B-PROP
dh{l-1}  =  prediction  -  target;
for i = nr_layers - 1 : -1 : 1
  Wgrad{i}  =  dh{i} * h{i-1}';
  bgrad{i}  =  sum(dh{i}, 2);
  dh{i-1}  =  (W{i}' * dh{i})  .*  jac{i-1};
end


% UPDATE
for i = 1 : nr_layers - 1
  W{i}  =  W{i}  -  (lr / batch_size)  *  Wgrad{i};
  b{i}  =  b{i}  -  (lr / batch_size)  *  bgrad{i};
end
```



Recurrent
Neural Net
Loss: type #1
CNN
Neural Net
SUPERVISED
SHALLOW
Boosting
Perceptron
SVM
UNSUPERVISED
Deep (sparse/denoising)
Autoencoder
ΣΠ
PROBABILISTIC
DBN
BayesNP
DEEP
Sparse Coding
Autoencoder
Neural Net
Restricted BM
GMM

FULLY CONNECTED NEURAL NET

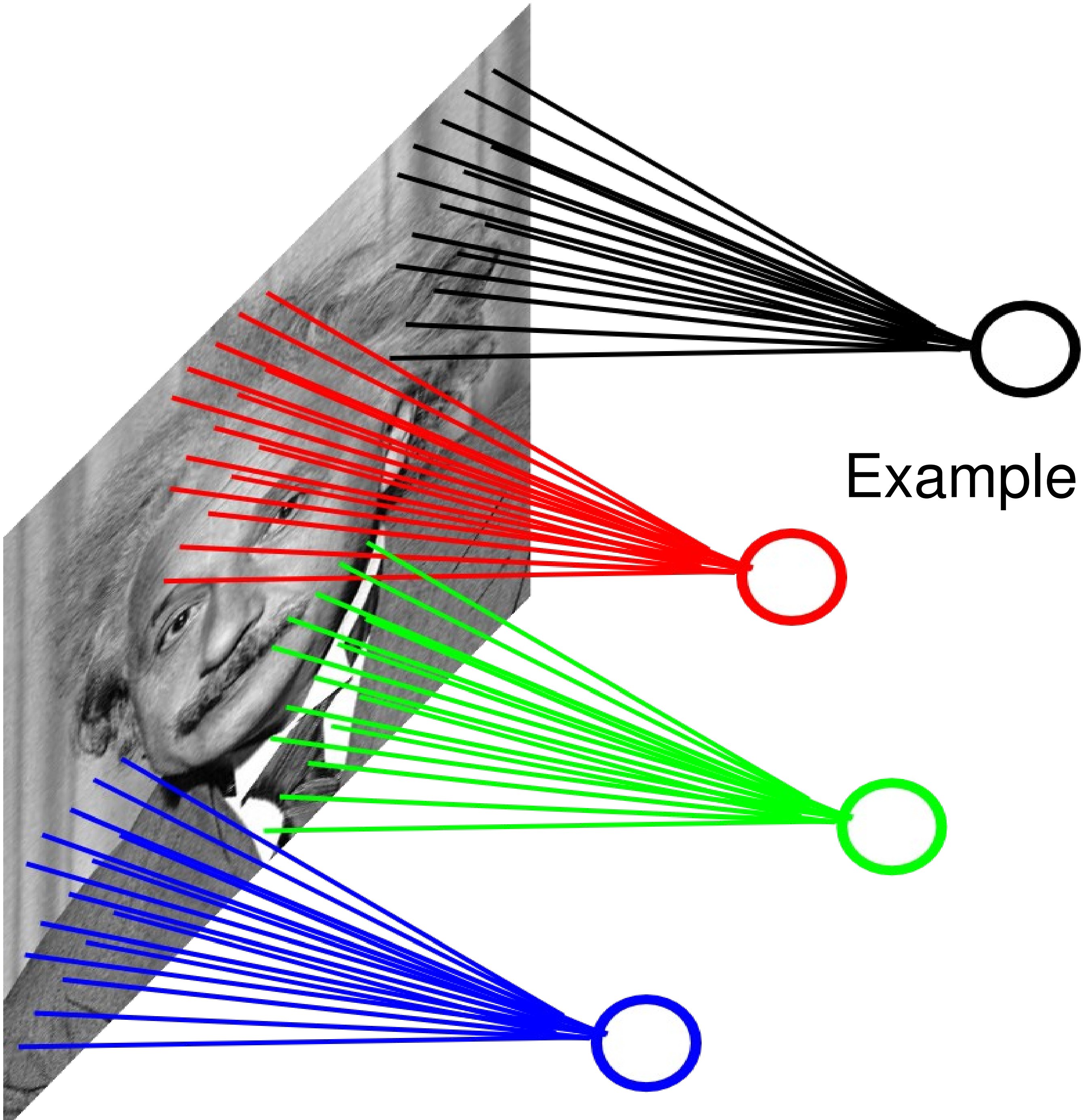


Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Better to put resources elsewhere!

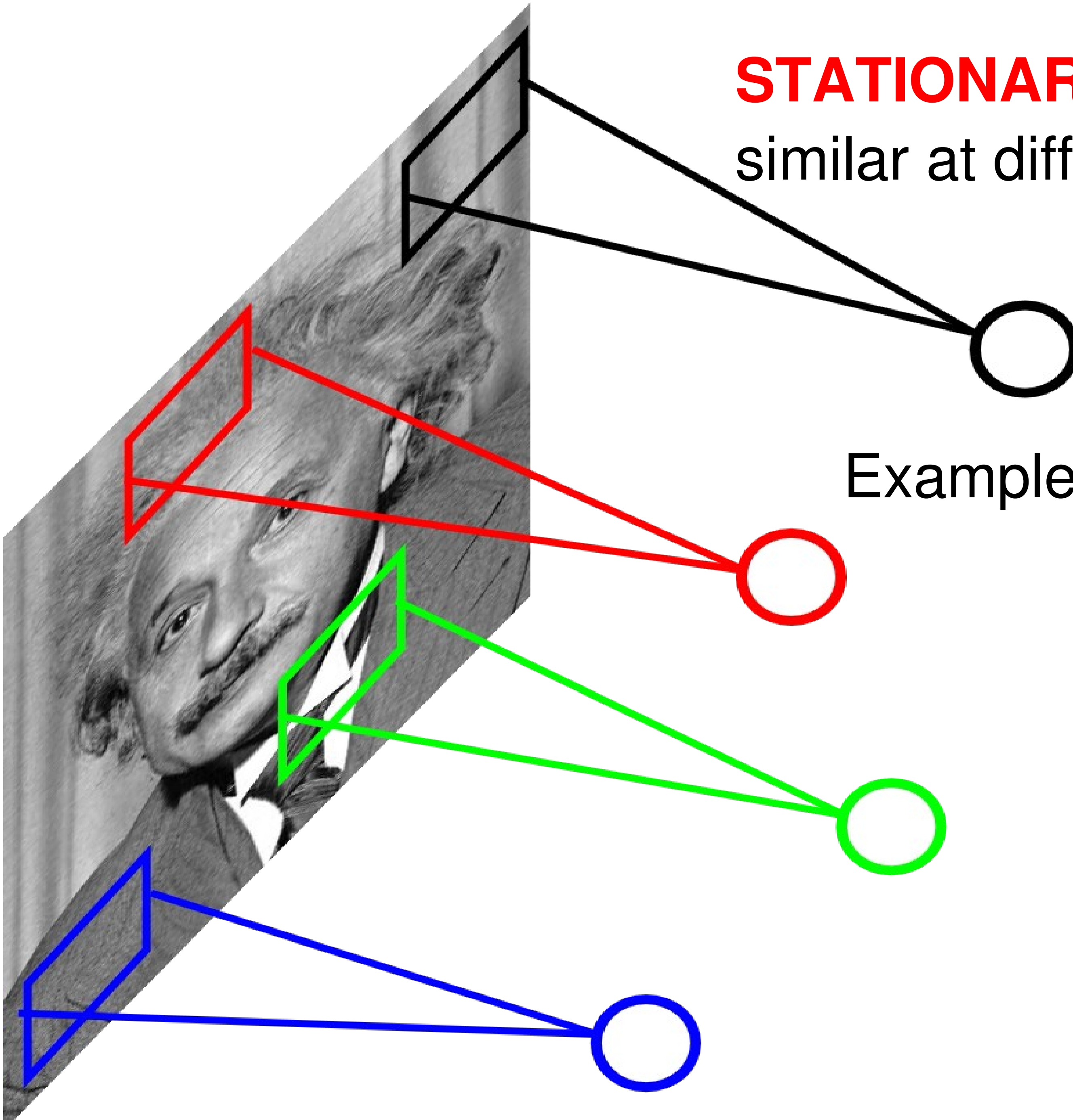
Ranzato

LOCALLY CONNECTED NEURAL NET



Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

LOCALLY CONNECTED NEURAL NET

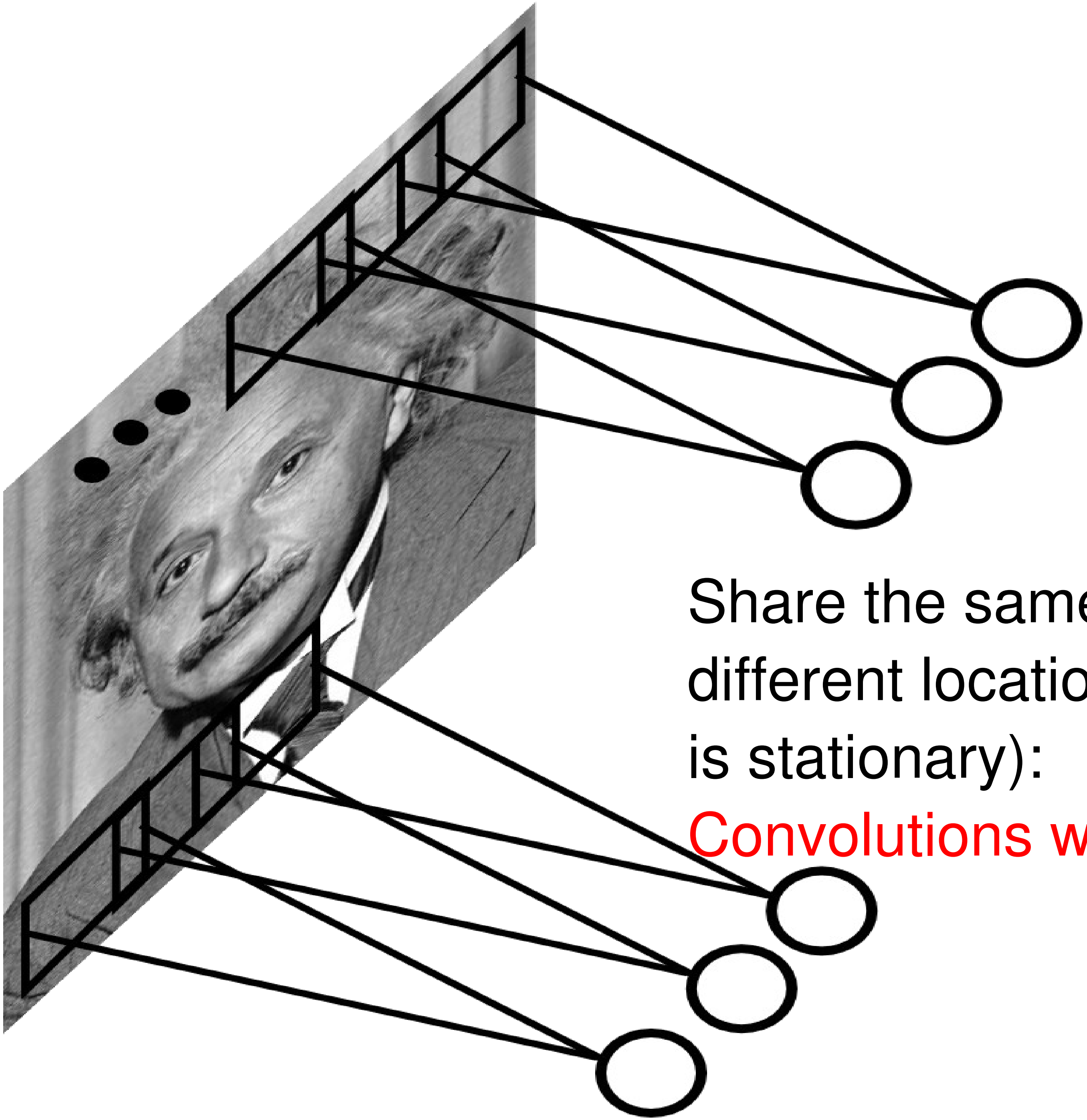


STATIONARITY? Statistics are similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Ranzato

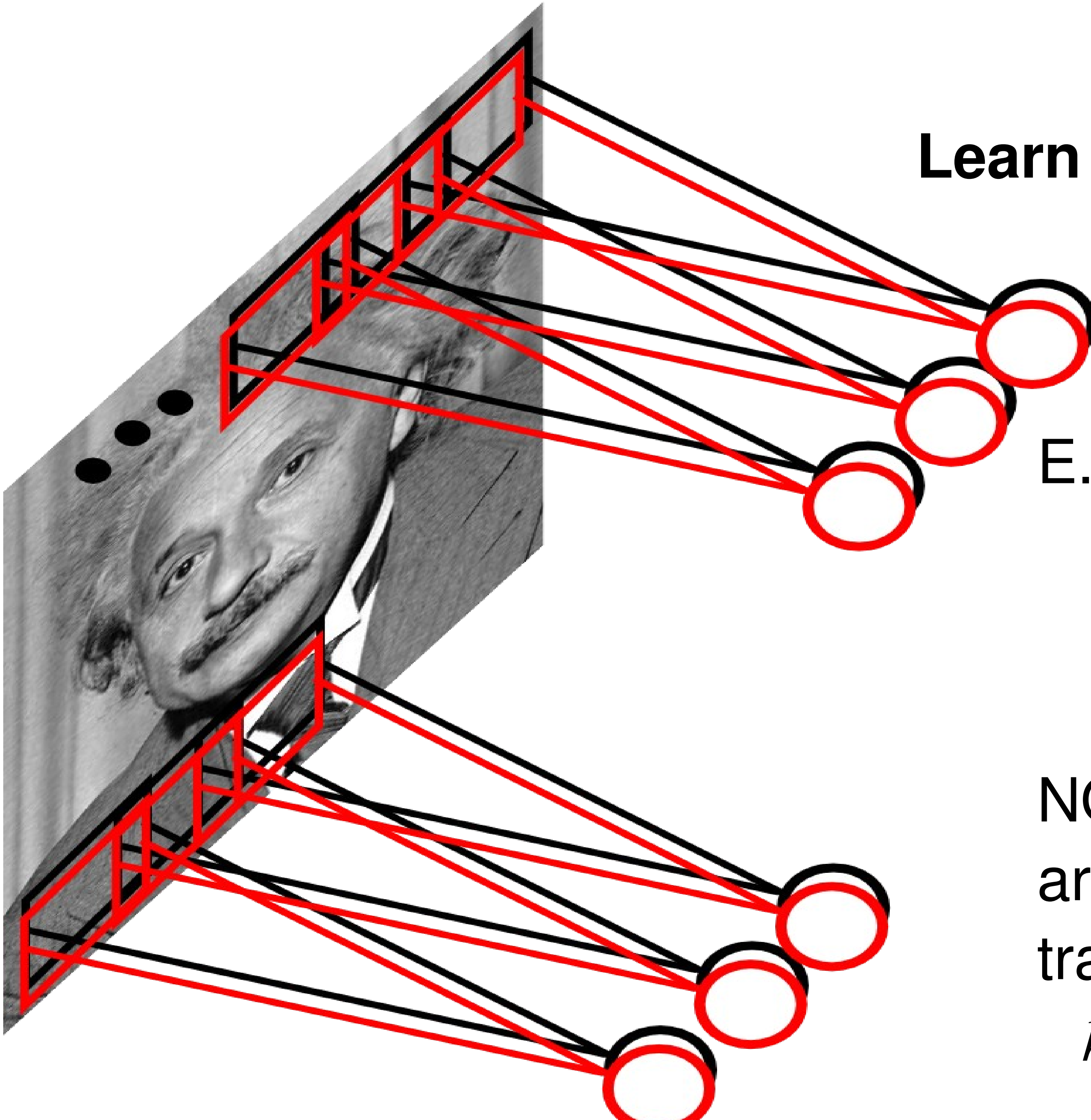
CONVOLUTIONAL NET



Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

Ranzato

CONVOLUTIONAL NET



Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

NOTE: filter responses are non-linearly transformed:

$$h = max(0, x * w)$$

KEY IDEAS

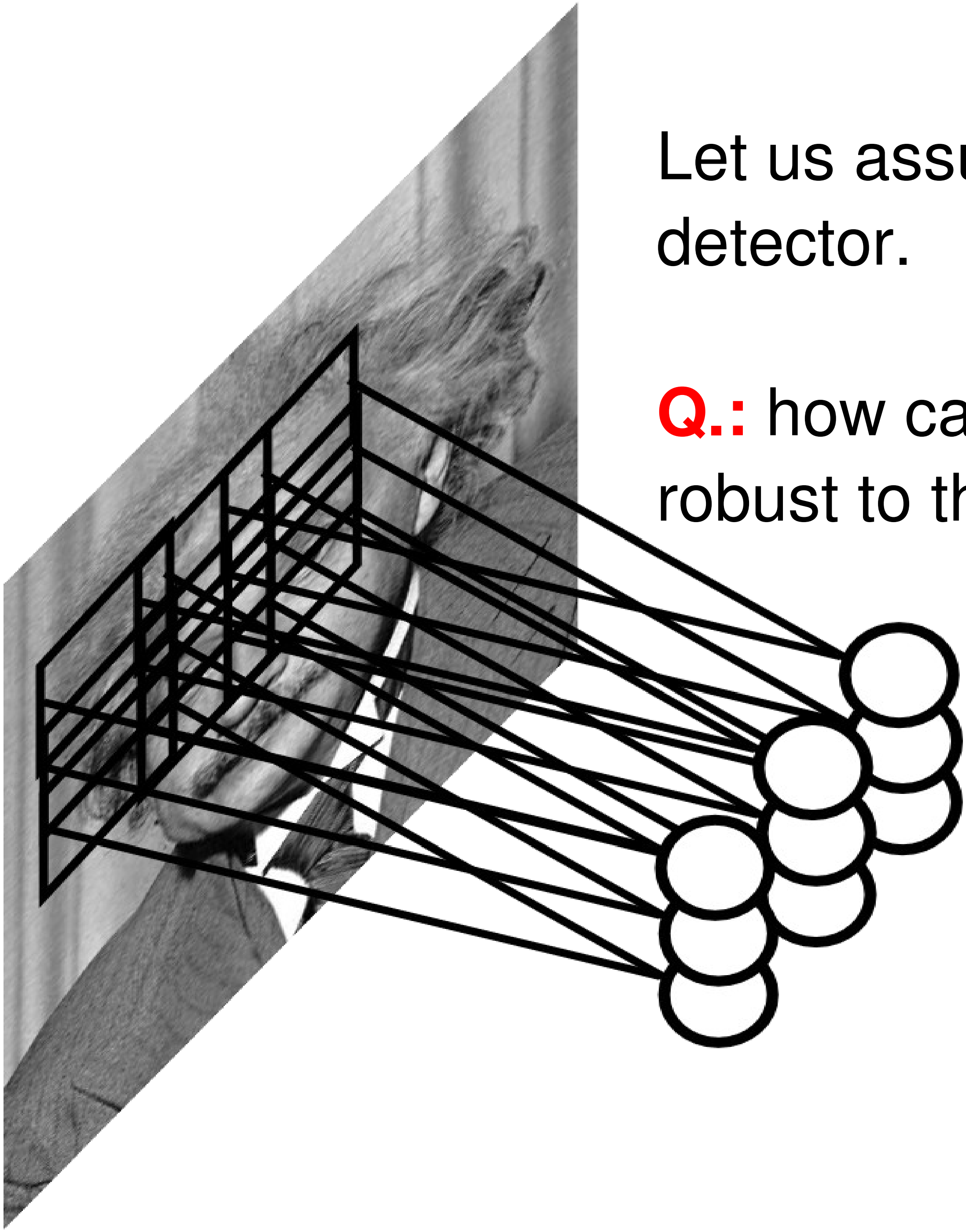
A standard neural net applied to images:
- scales quadratically with the size of the input
- does not leverage stationarity

Solution:
- connect each hidden unit to a small patch of the input
- share the weight across hidden units

This is called: **convolutional layer.**
A network with convolutional layers is called **convolutional network.**

LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998

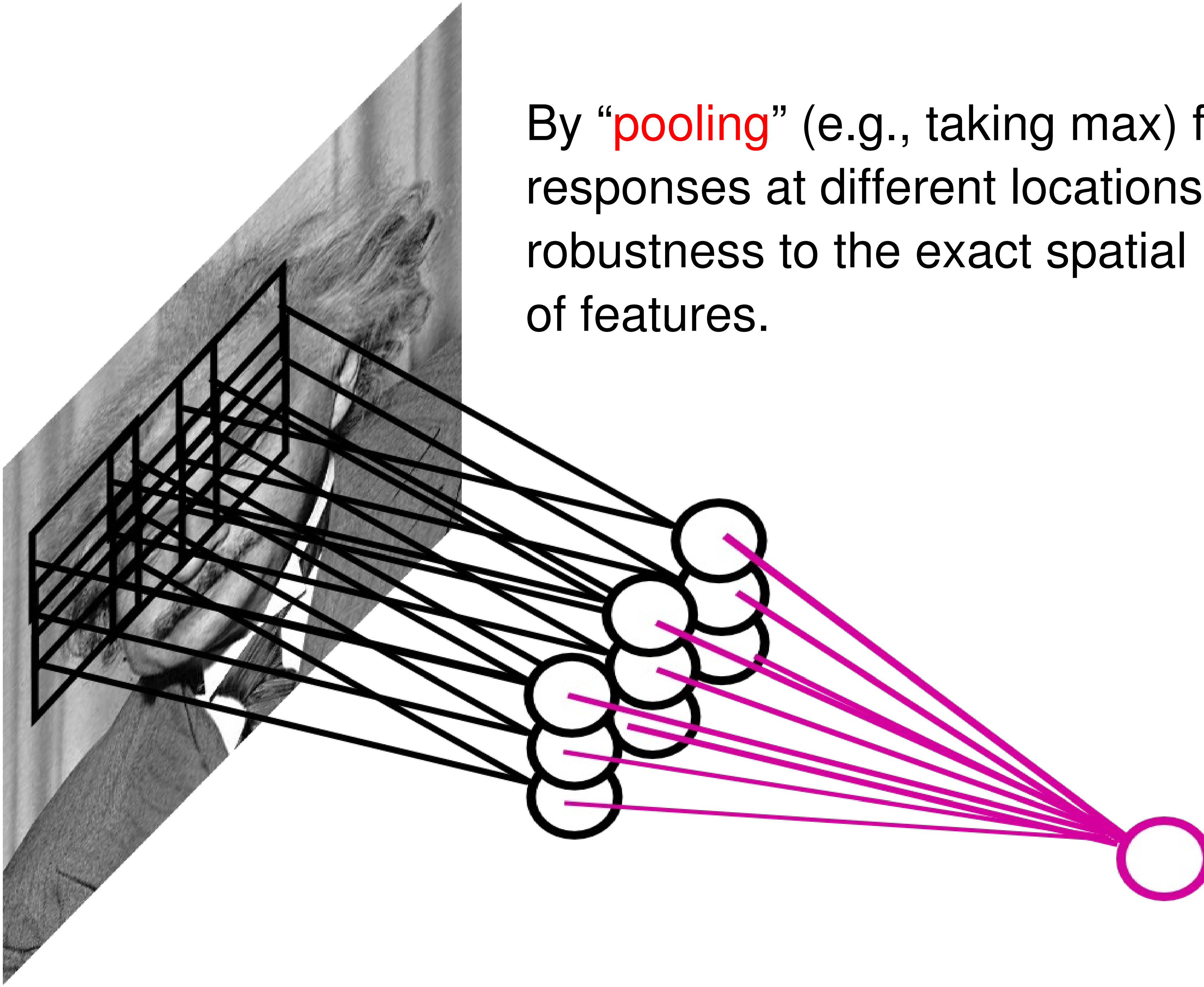
POOLING



Let us assume filter is an "eye" detector.

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

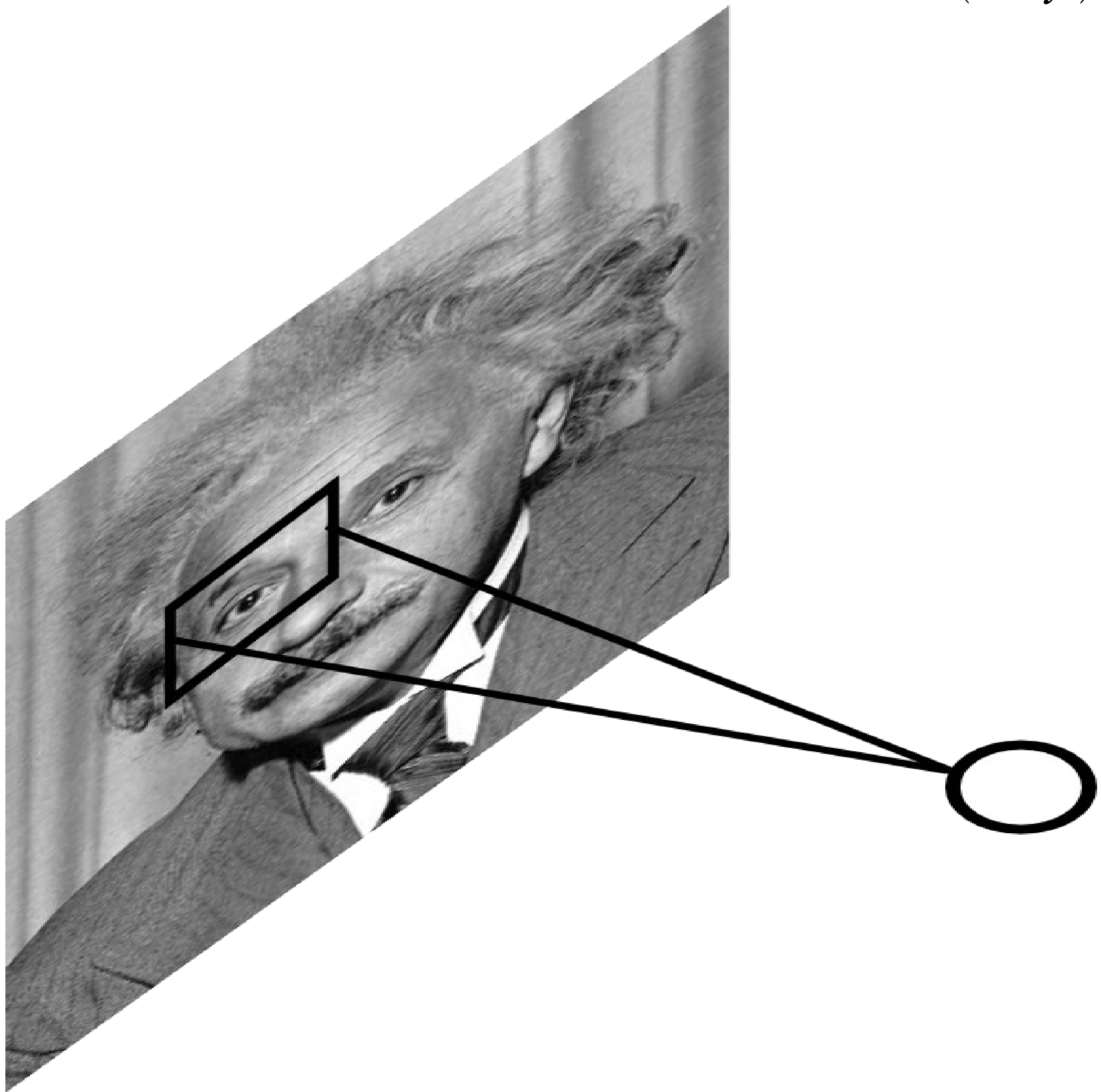
POOLING



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

LOCAL CONTRAST NORMALIZATION

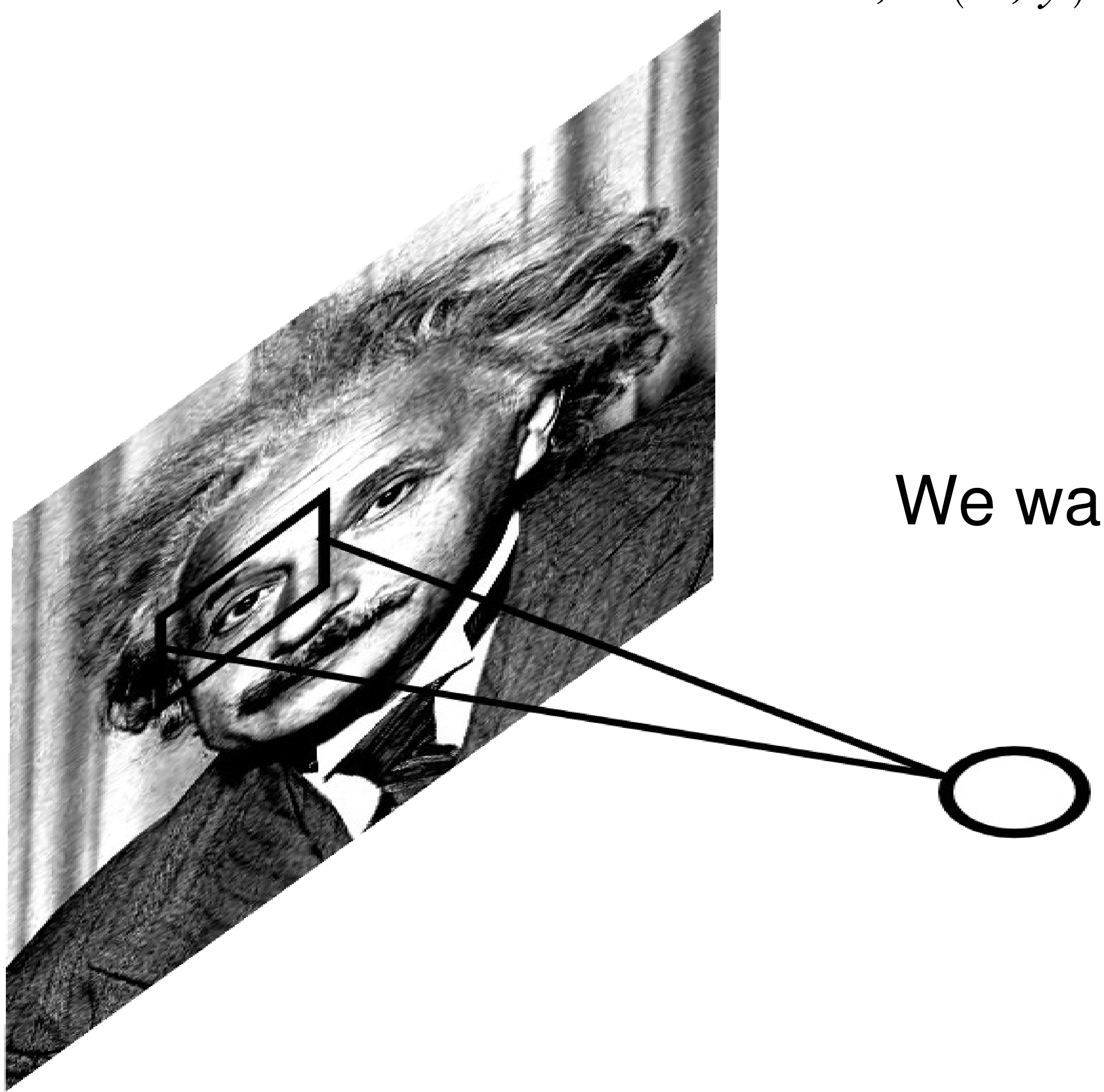
$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$



Ranzato

LOCAL CONTRAST NORMALIZATION

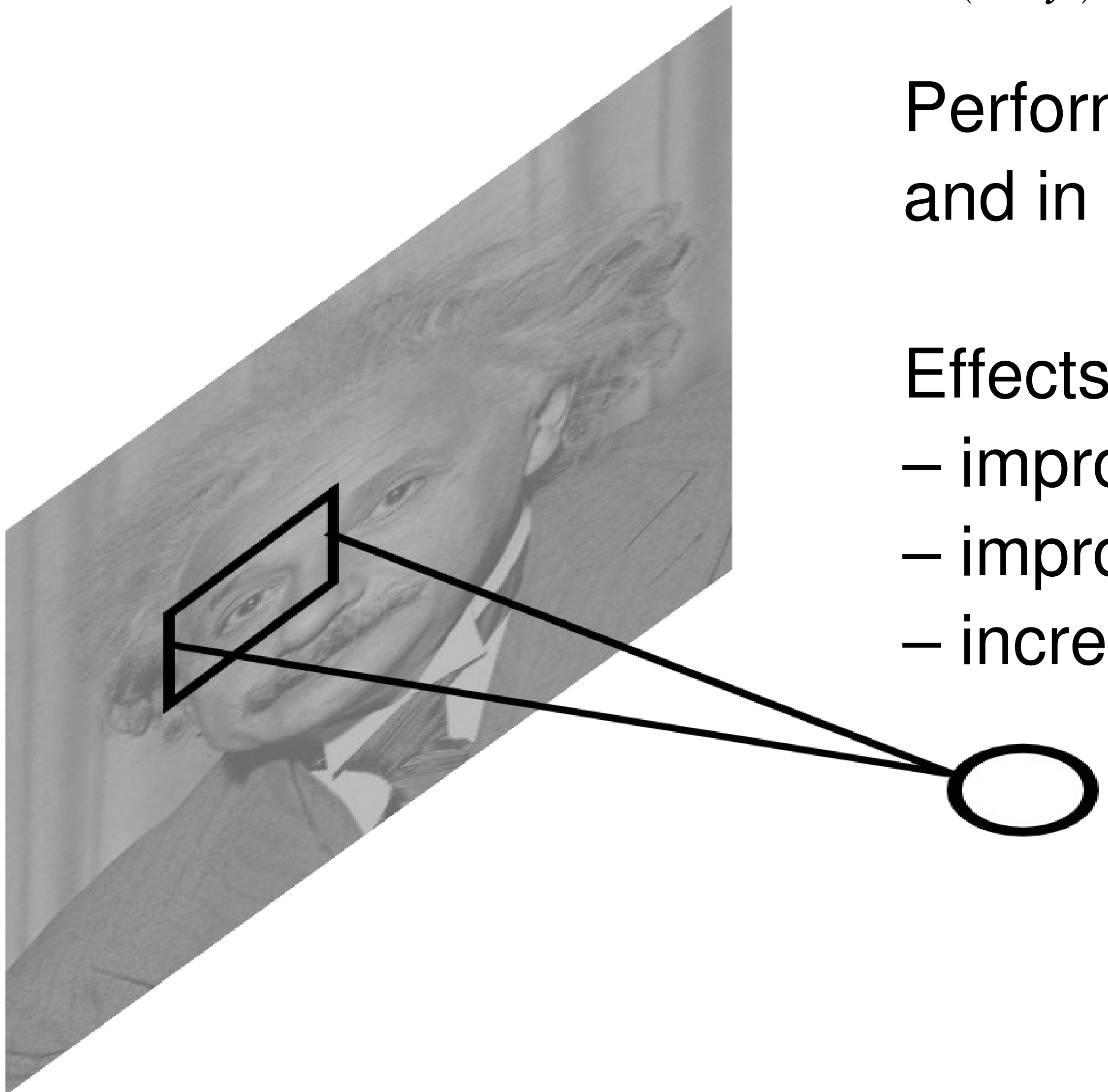
$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$



We want the same response.

LOCAL CONTRAST NORMALIZATION

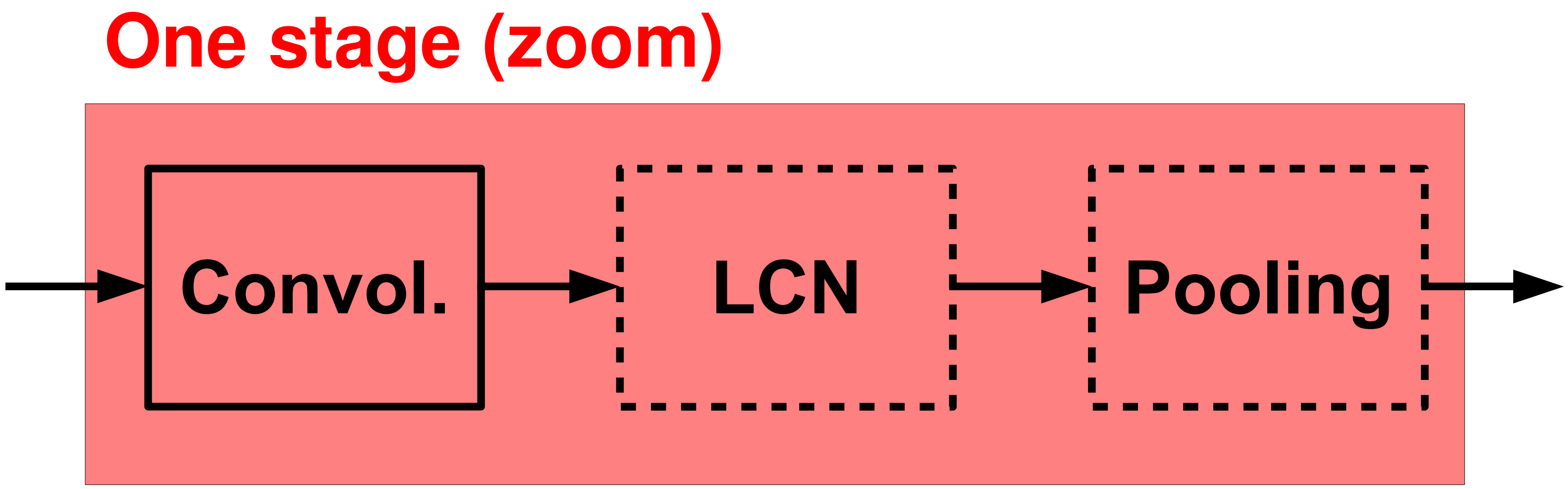
$$h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N(x,y)}}{\sigma_{i,N(x,y)}}$$

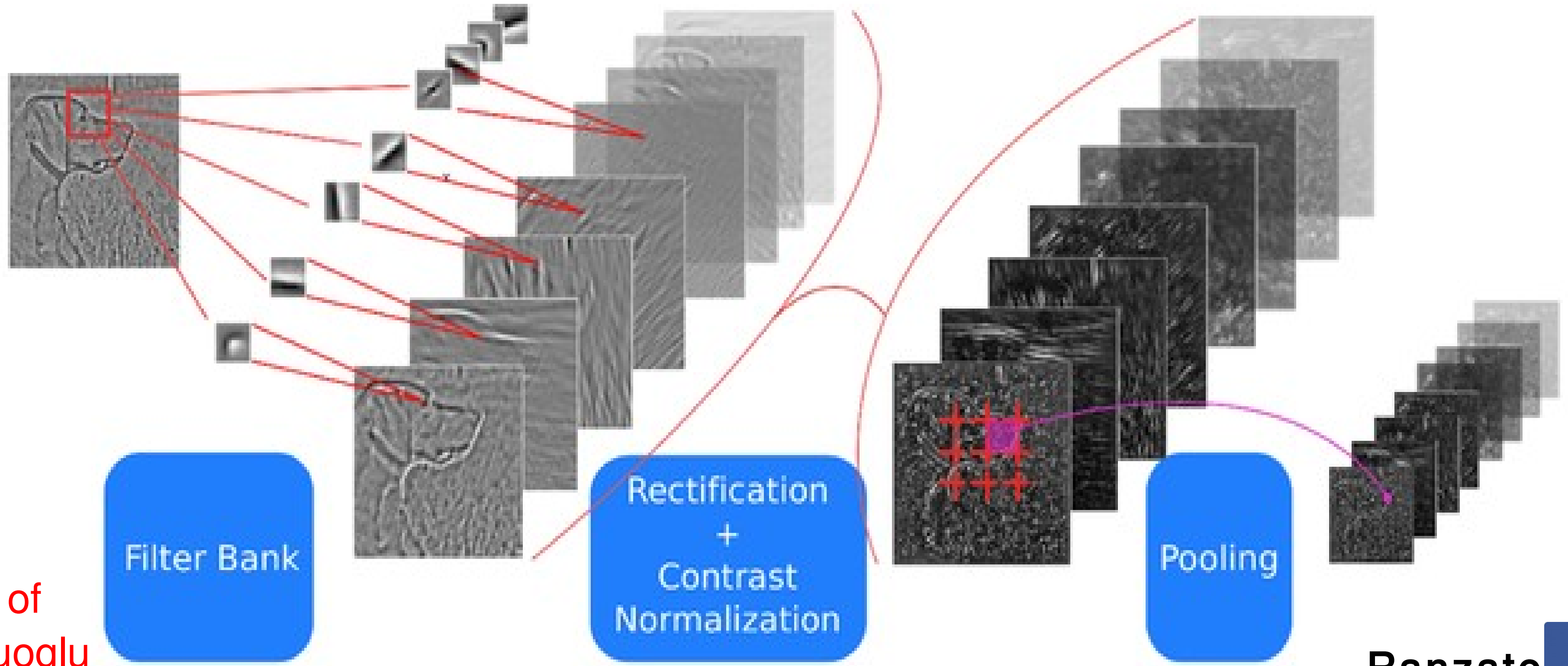


Performed also across features and in the higher layers.

Effects:
– improves invariance
– improves optimization
– increases sparsity

CONV NETS: TYPICAL ARCHITECTURE

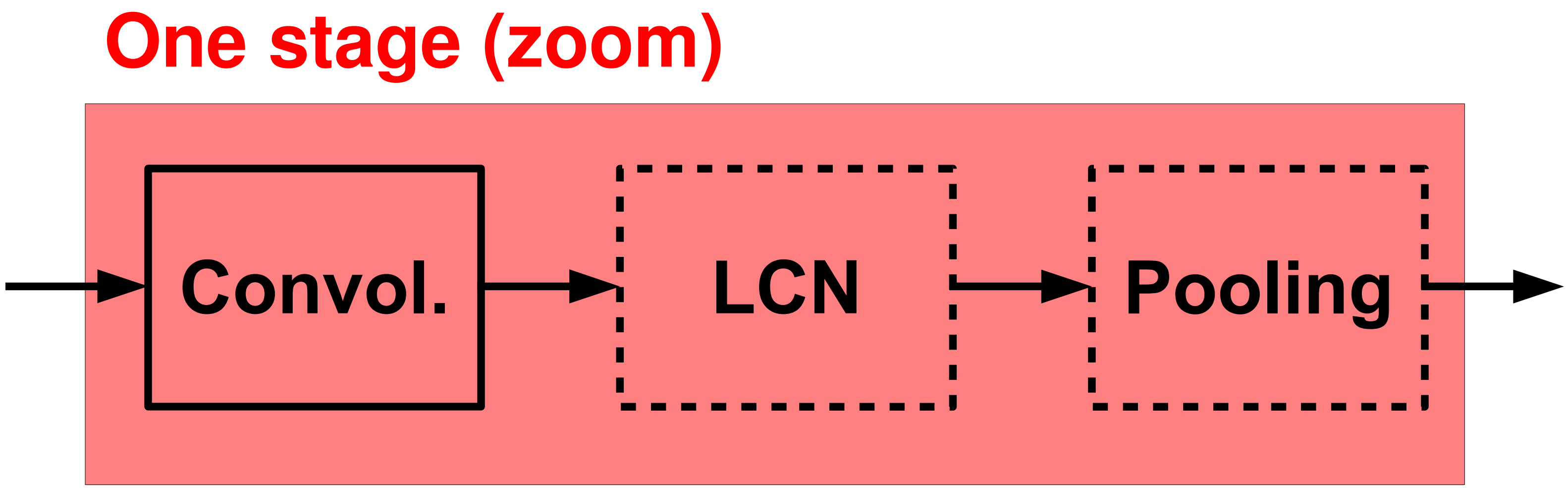




courtesy of
K. Kavukcuoglu

Ranzato

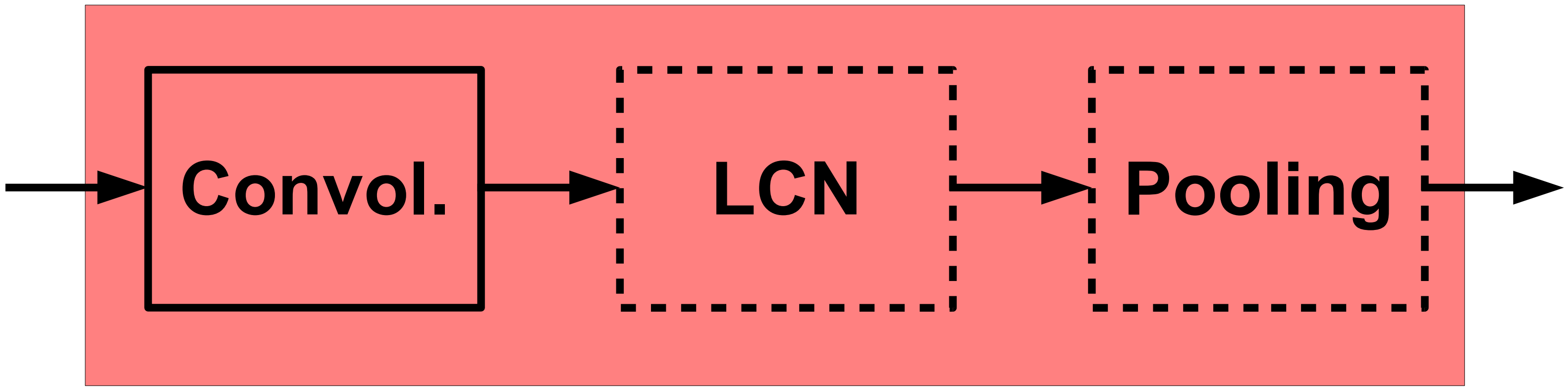
CONV NETS: TYPICAL ARCHITECTURE



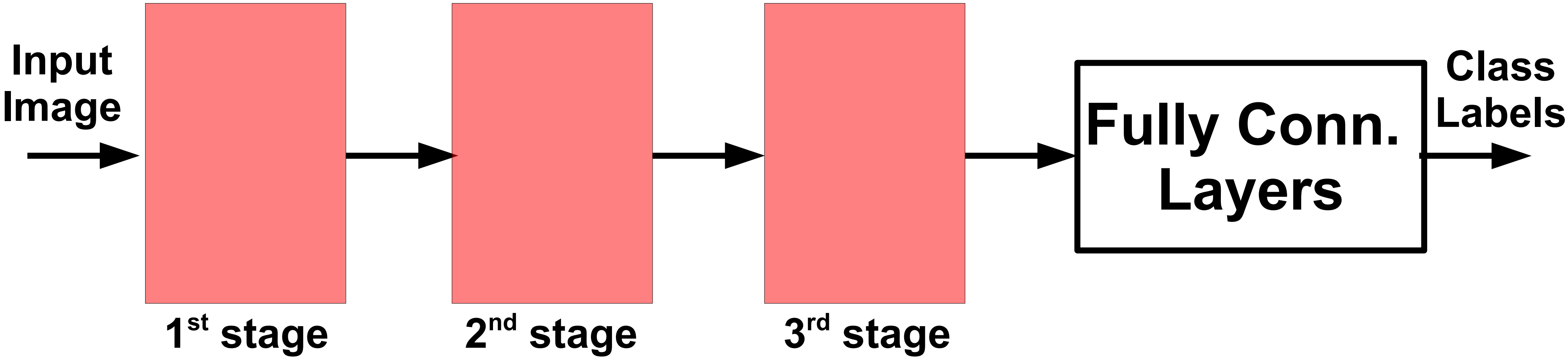
Conceptually similar to: SIFT, HoG, etc.

CONV NETS: TYPICAL ARCHITECTURE

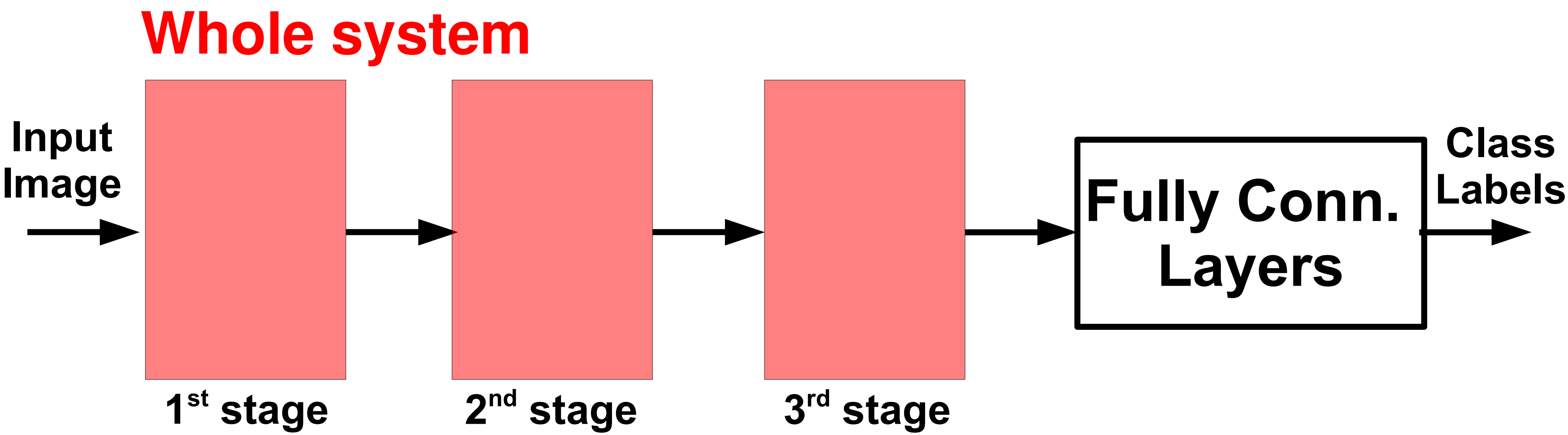
One stage (zoom)



Whole system



CONV NETS: TYPICAL ARCHITECTURE



Conceptually similar to:

SIFT → K-Means → Pyramid Pooling → SVM

Lazebnik et al. "...Spatial Pyramid Matching..." CVPR 2006

SIFT → Fisher Vect. → Pooling → SVM

Sanchez et al. "Image classifcation with F.V.: Theory and practice" IJCV 2012

CONV NETS: TRAINING

All layers are differentiable (a.e.).
We can use standard back-propagation.

Algorithm:
Given a small mini-batch
- F-PROP
- B-PROP
- PARAMETER UPDATE

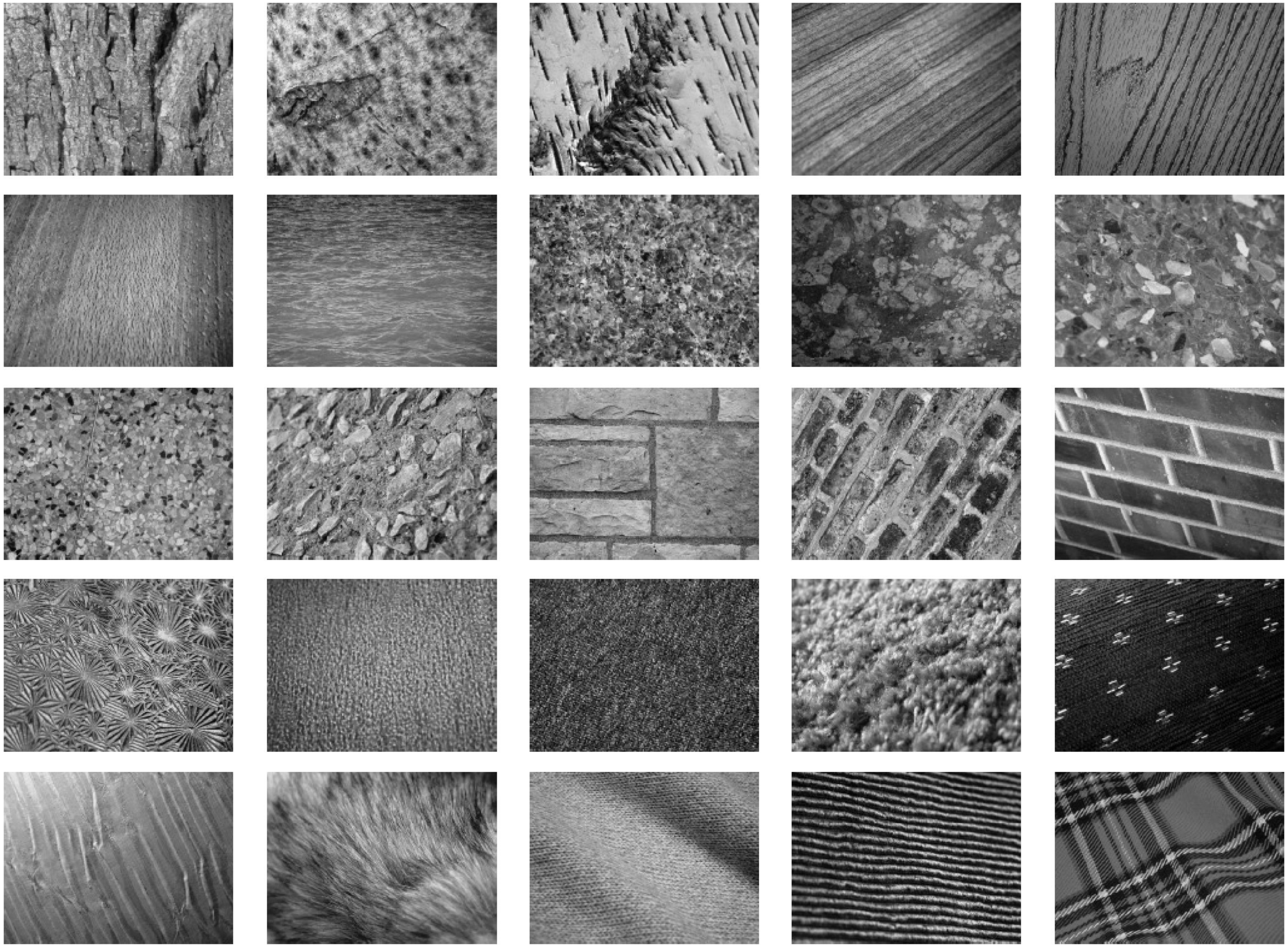
CONV NETS: EXAMPLES

- OCR / House number & Traffic sign classification

Ciresan et al. “MCDNN for image classification” CVPR 2012
Wan et al. “Regularization of neural networks using dropconnect” ICML 2013

CONV NETS: EXAMPLES

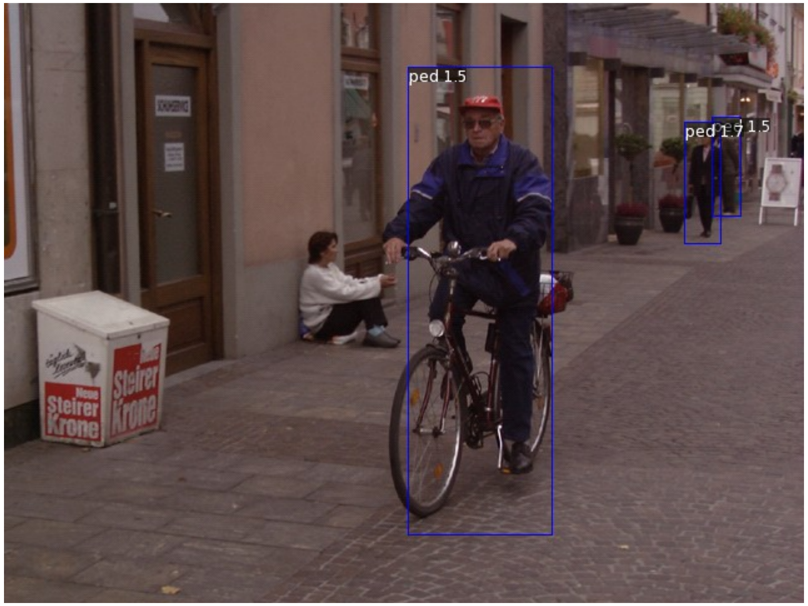
- **Texture classification**

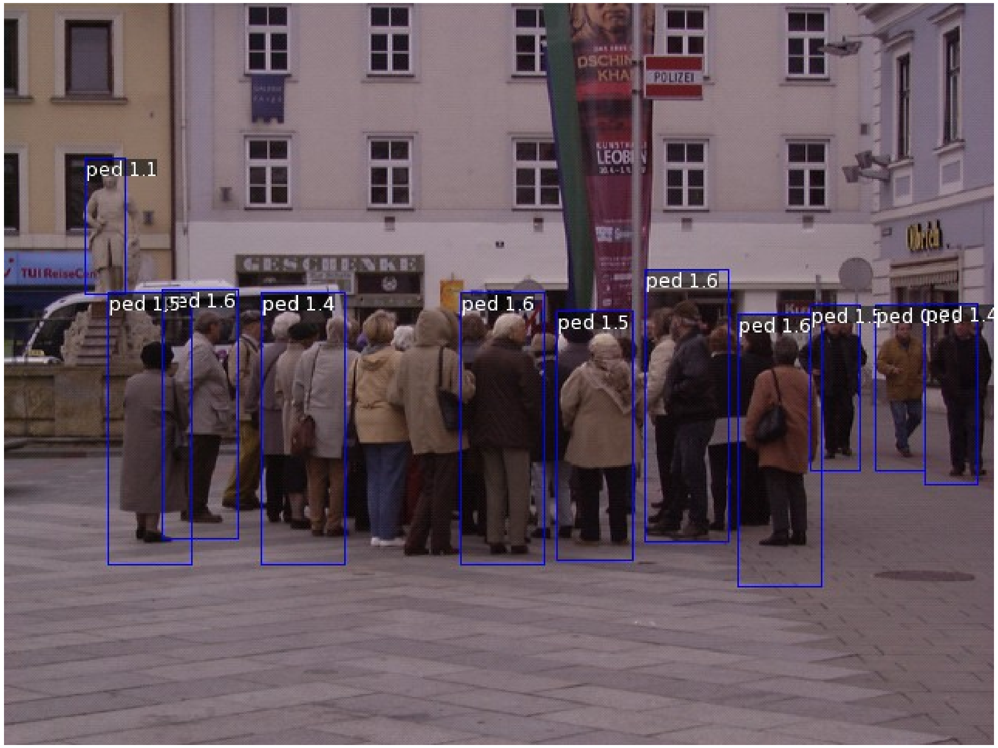


Sifre et al. “Rotation, scaling and deformation invariant scattering...” CVPR 2013

CONV NETS: EXAMPLES

- **Pedestrian detection**

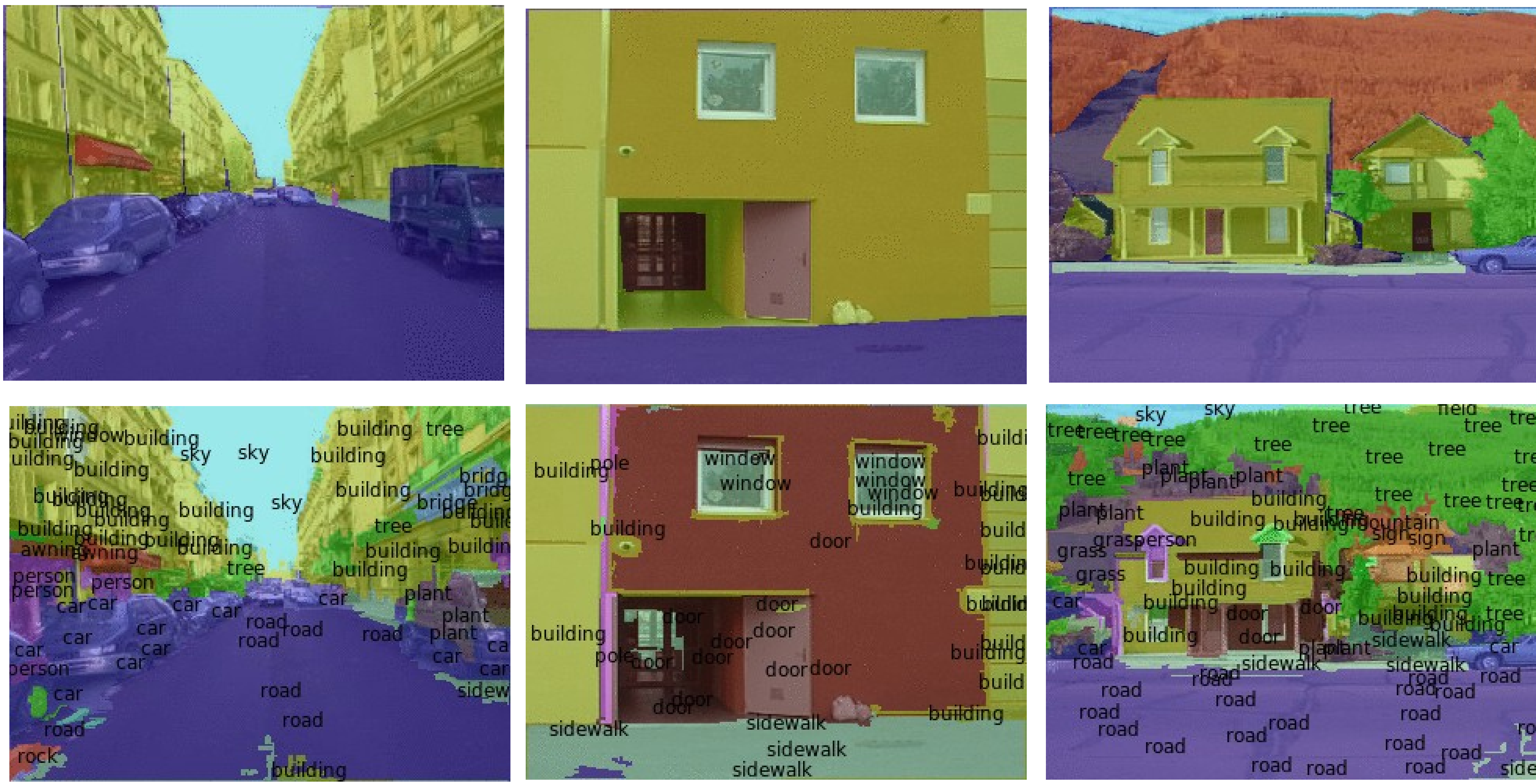




Sermanet et al. "Pedestrian detection with unsupervised multi-stage.." CVPR 2013

CONV NETS: EXAMPLES

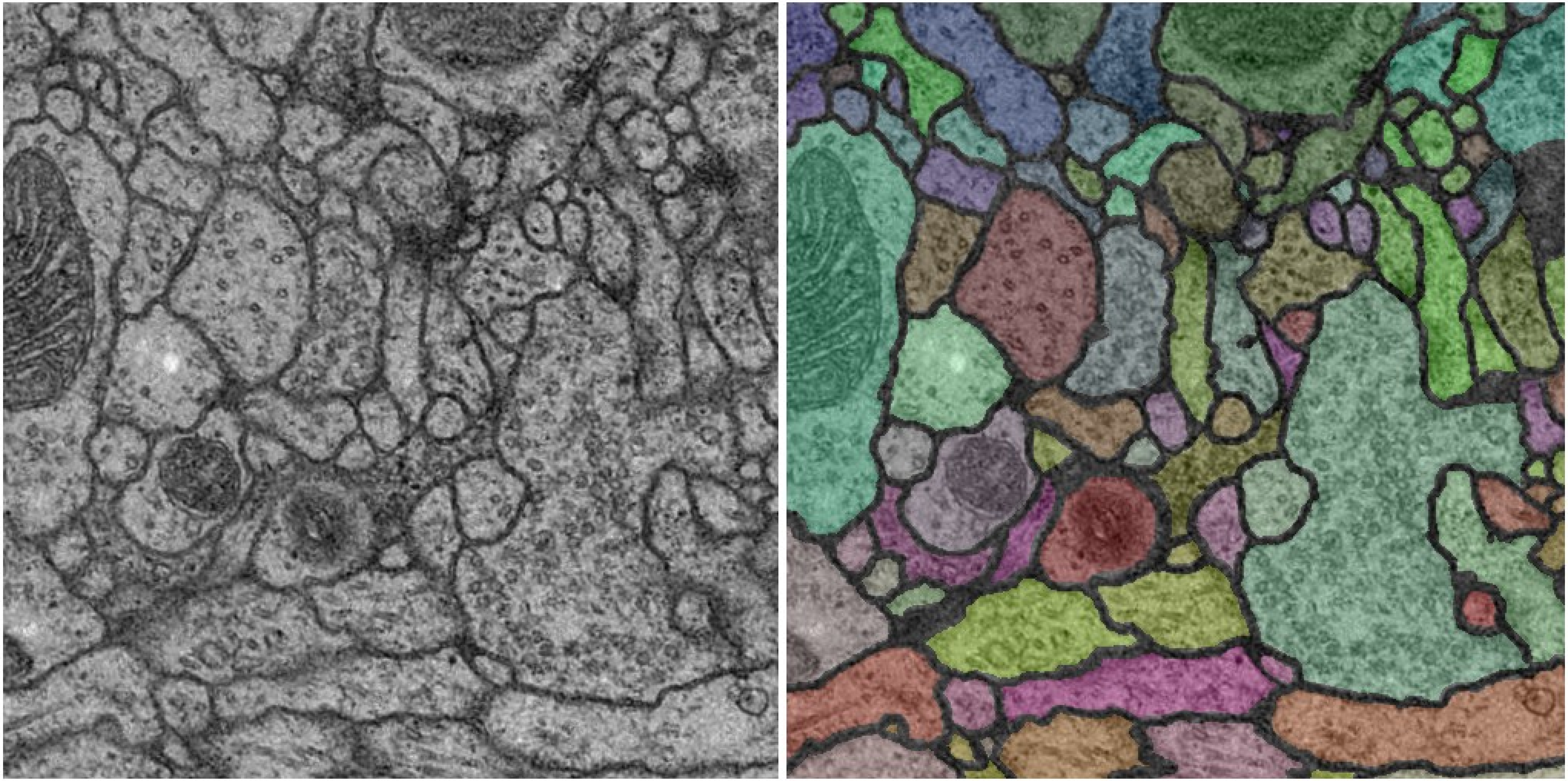
- **Scene Parsing**



Farabet et al. "Learning hierarchical features for scene labeling" PAMI 2013
Pinheiro et al. "Recurrent CNN for scene parsing" arxiv 2013

CONV NETS: EXAMPLES

- **Segmentation 3D volumetric images**



Ciresan et al. "DNN segment neuronal membranes..." NIPS 2012
Turaga et al. "Maximin learning of image segmentation" NIPS 2009

CONV NETS: EXAMPLES

- **Action recognition from videos**

Taylor et al. “Convolutional learning of spatio-temporal features” ECCV 2010

CONV NETS: EXAMPLES

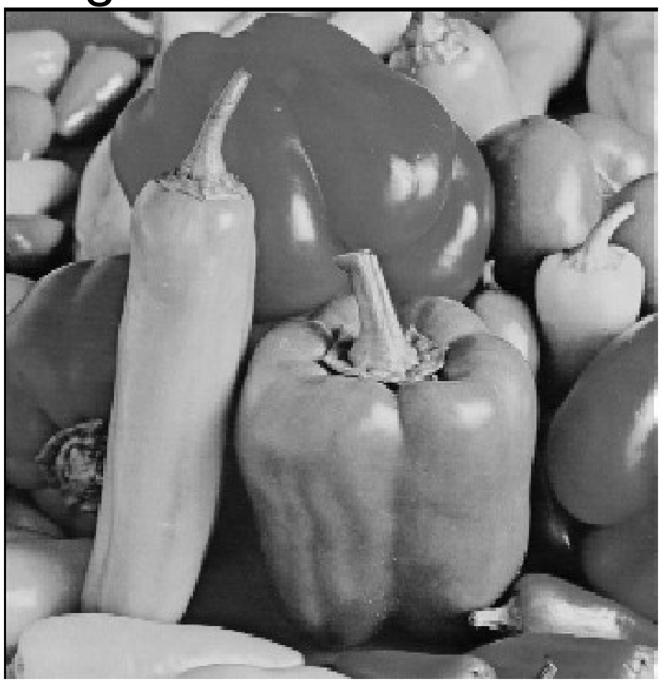
- **Robotics**

Sermanet et al. "Mapping and planning ...with long range perception" IROS 2008

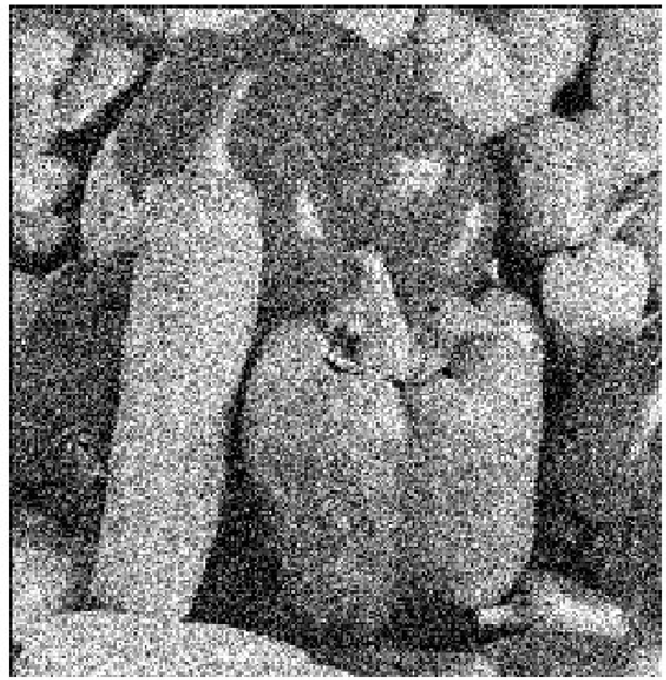
CONV NETS: EXAMPLES

- **Denoising**

original



noised

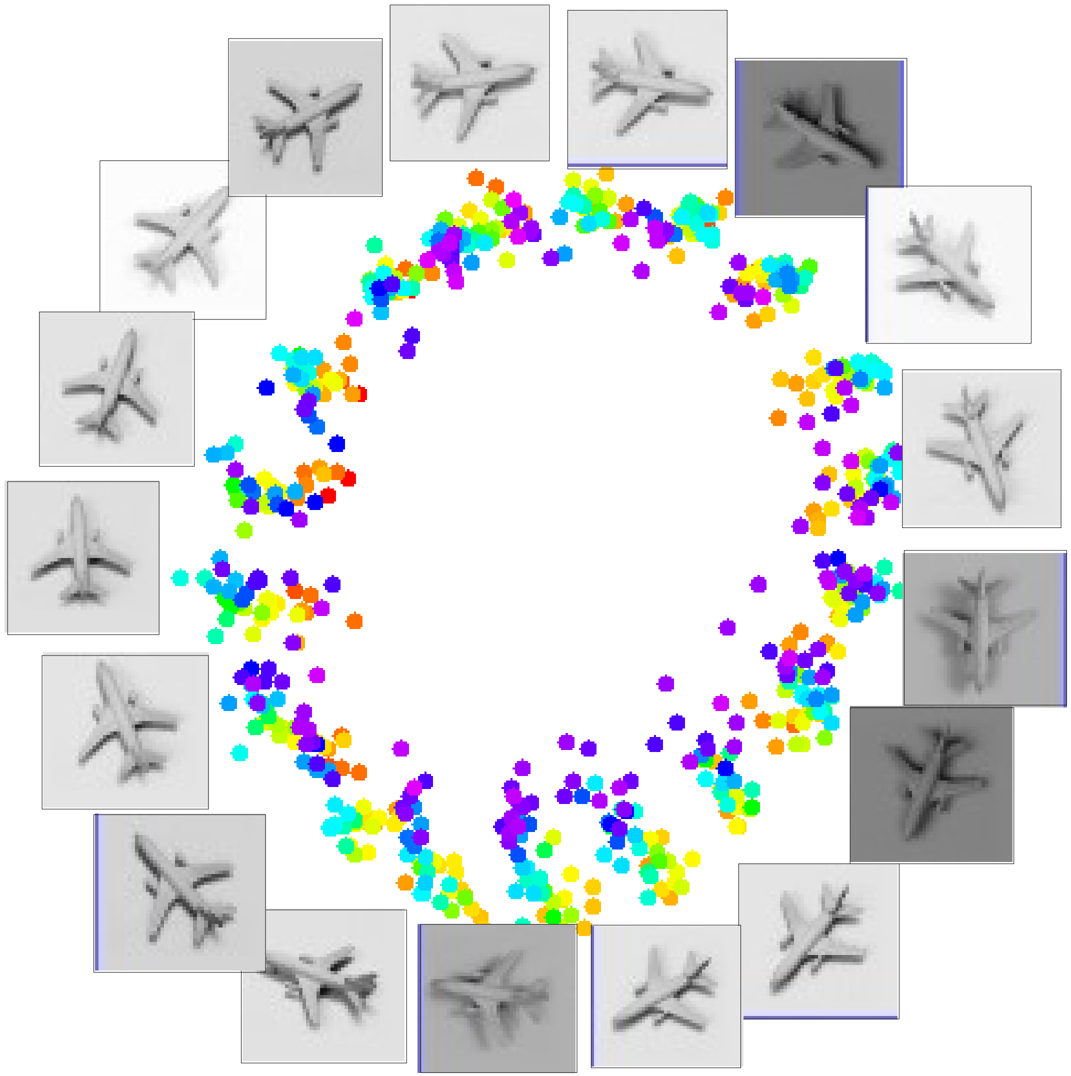


denoised

Burger et al. “Can plain NNs compete with BM3D?” CVPR 2012

CONV NETS: EXAMPLES

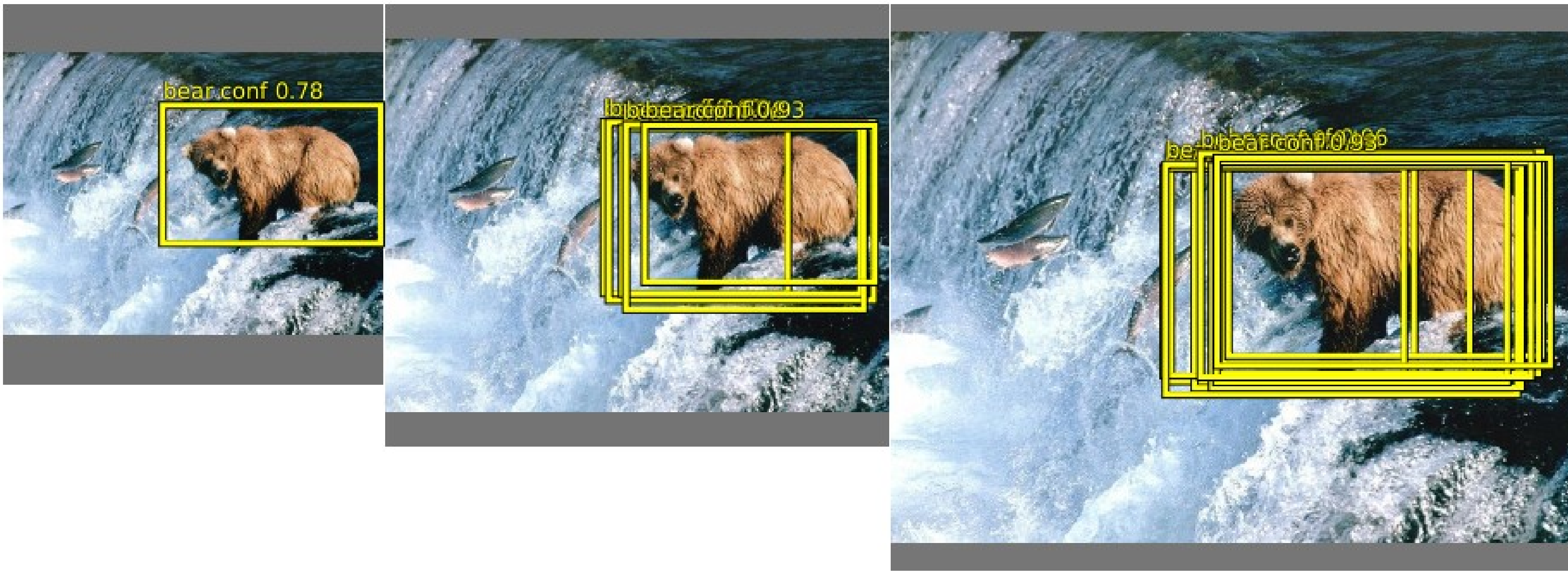
- **Dimensionality reduction / learning embeddings**



Hadsell et al. “Dimensionality reduction by learning an invariant mapping” CVPR 2006

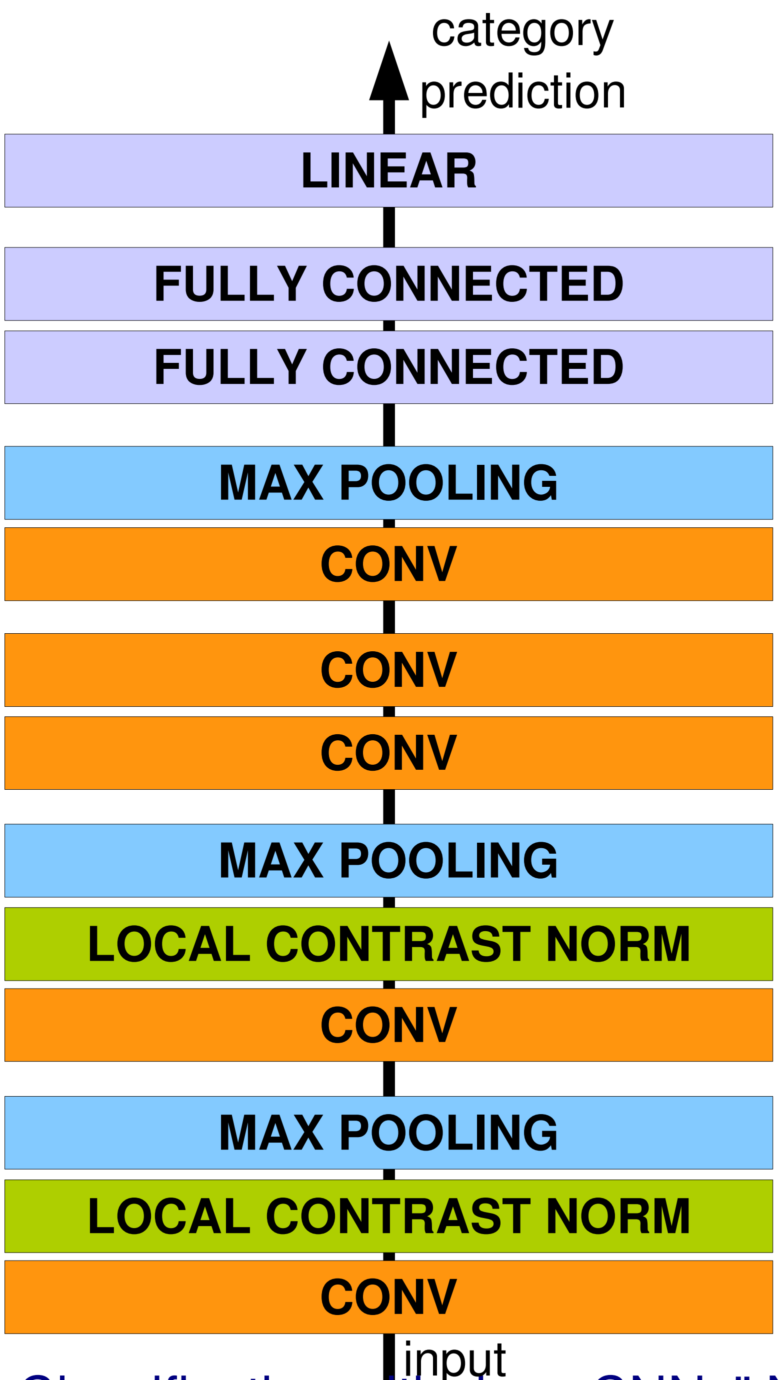
CONV NETS: EXAMPLES

- **Object detection**



Sermanet et al. "OverFeat: Integrated recognition, localization, ..." arxiv 2013
Girshick et al. "Rich feature hierarchies for accurate object detection..." arxiv 2013
Szegedy et al. "DNN for object detection" NIPS 2013

Architecture for Classification

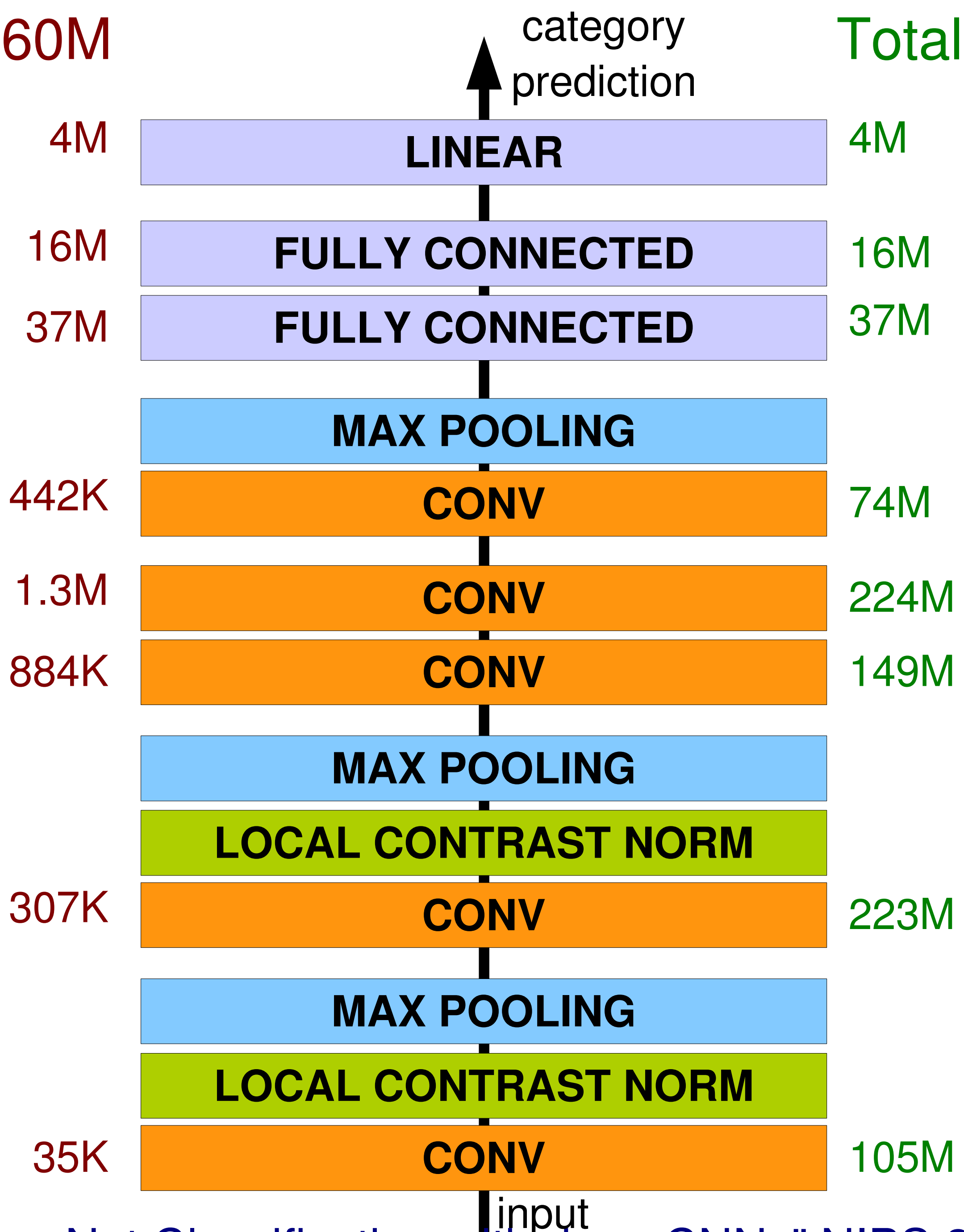


Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

Architecture for Classification

Total nr. params: 60M

Total nr. flops: 832M



Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

Optimization

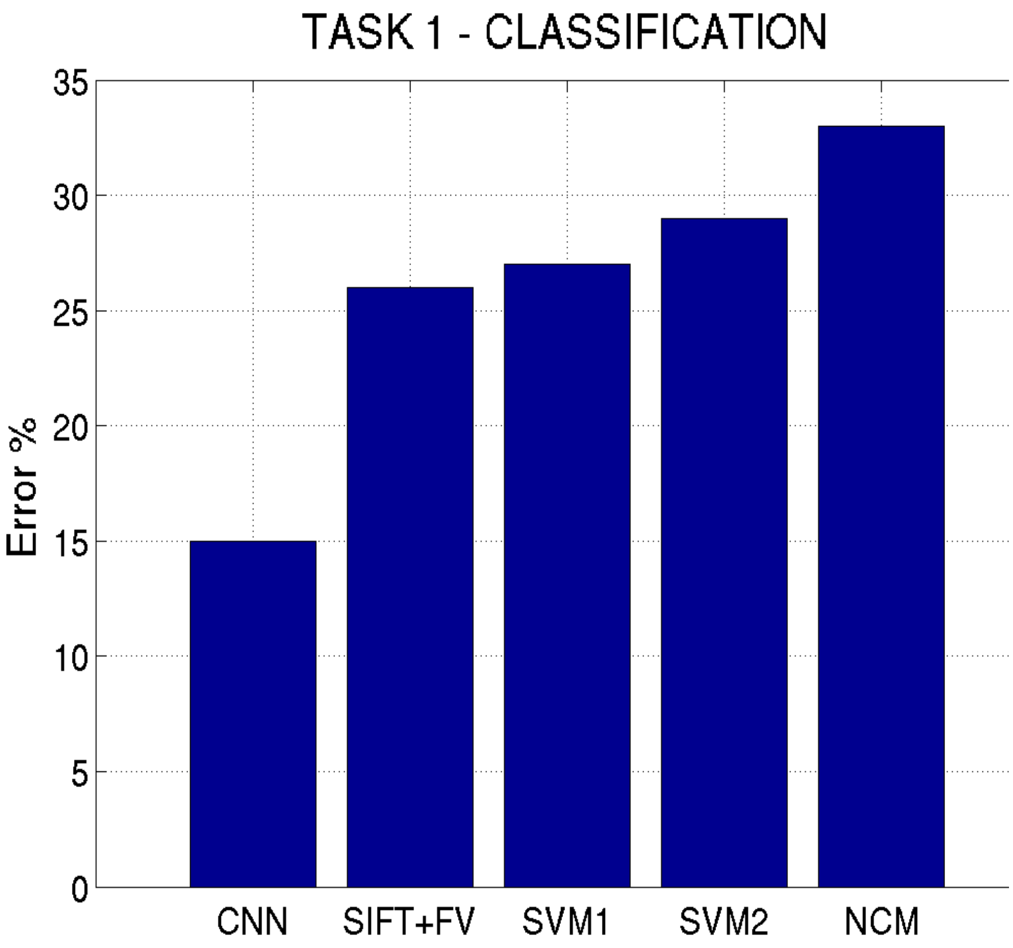
SGD with momentum:

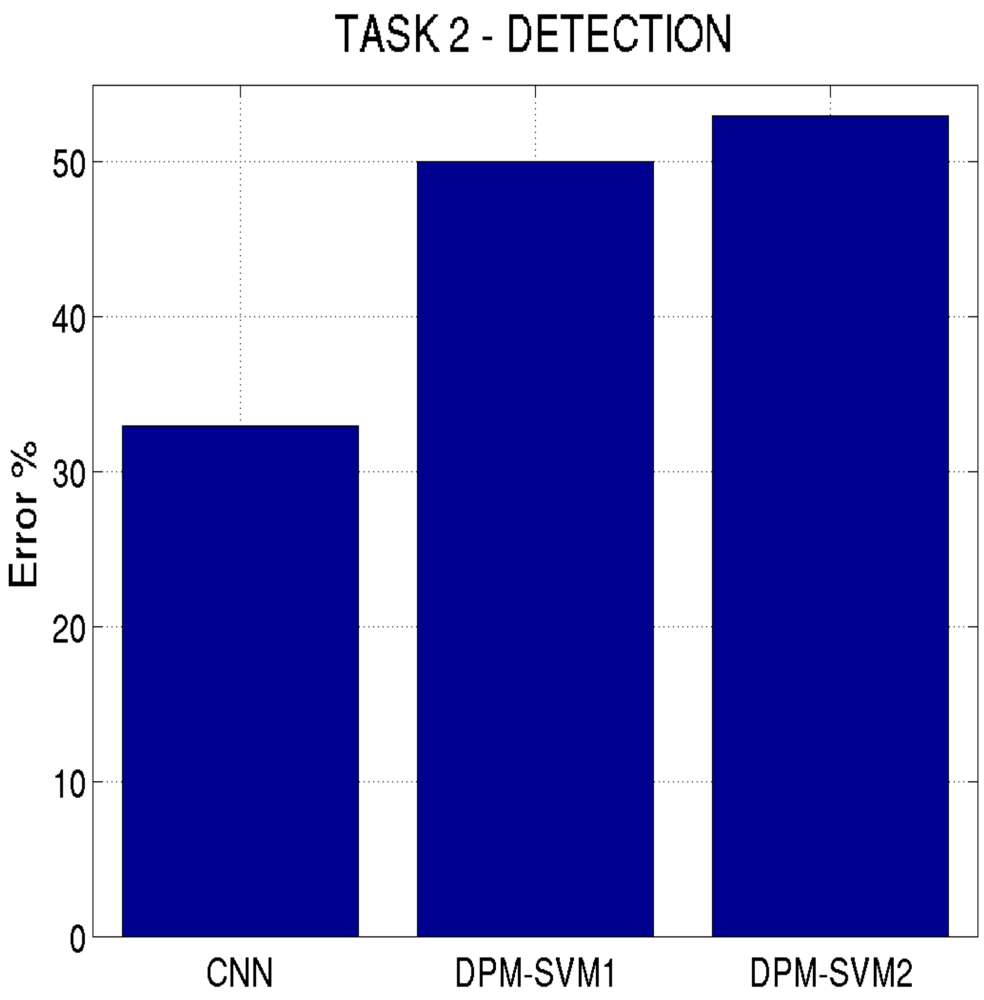
- Learning rate = 0.01
- Momentum = 0.9

Improving generalization by:

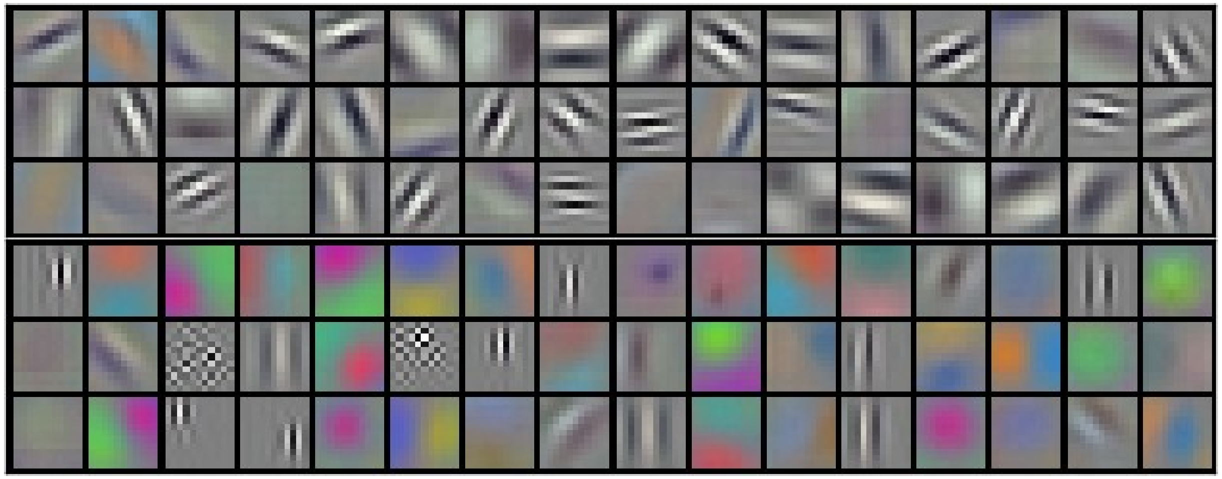
- Weight sharing (convolution)
- Input distortions
- Dropout = 0.5
- Weight decay = 0.0005

Results: ILSVRC 2012



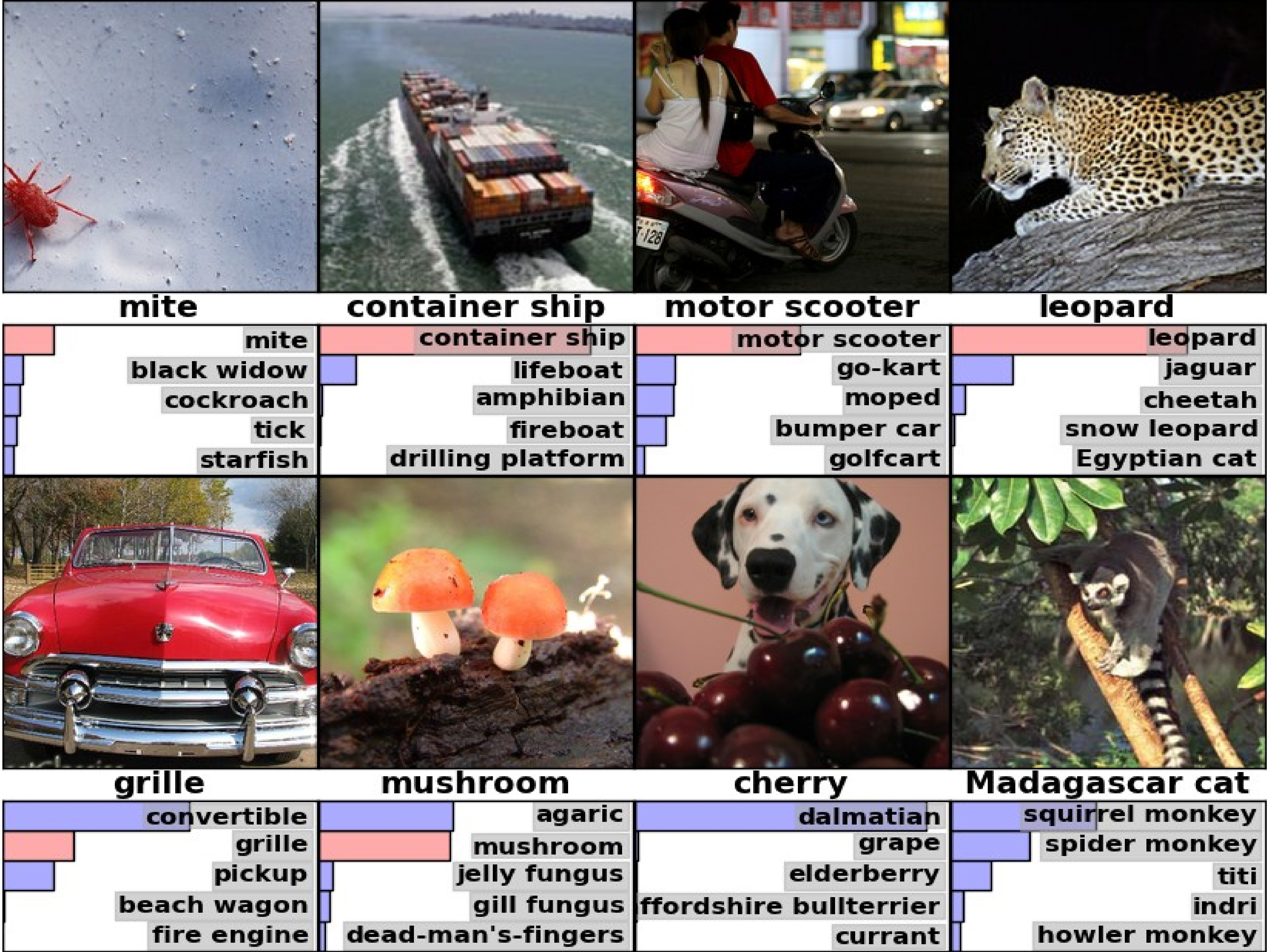


Results

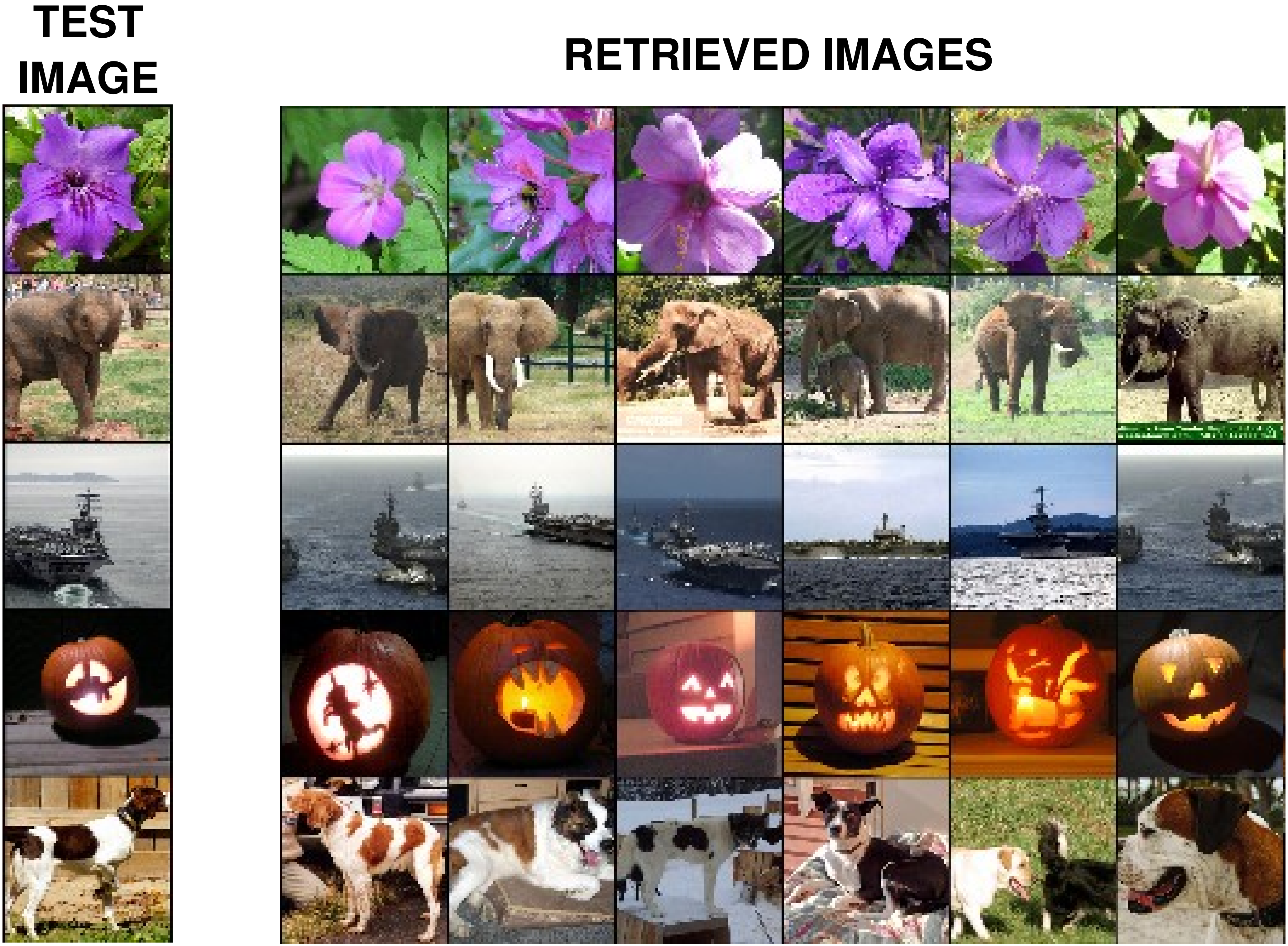


First layer learned filters (processing raw pixel values).

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

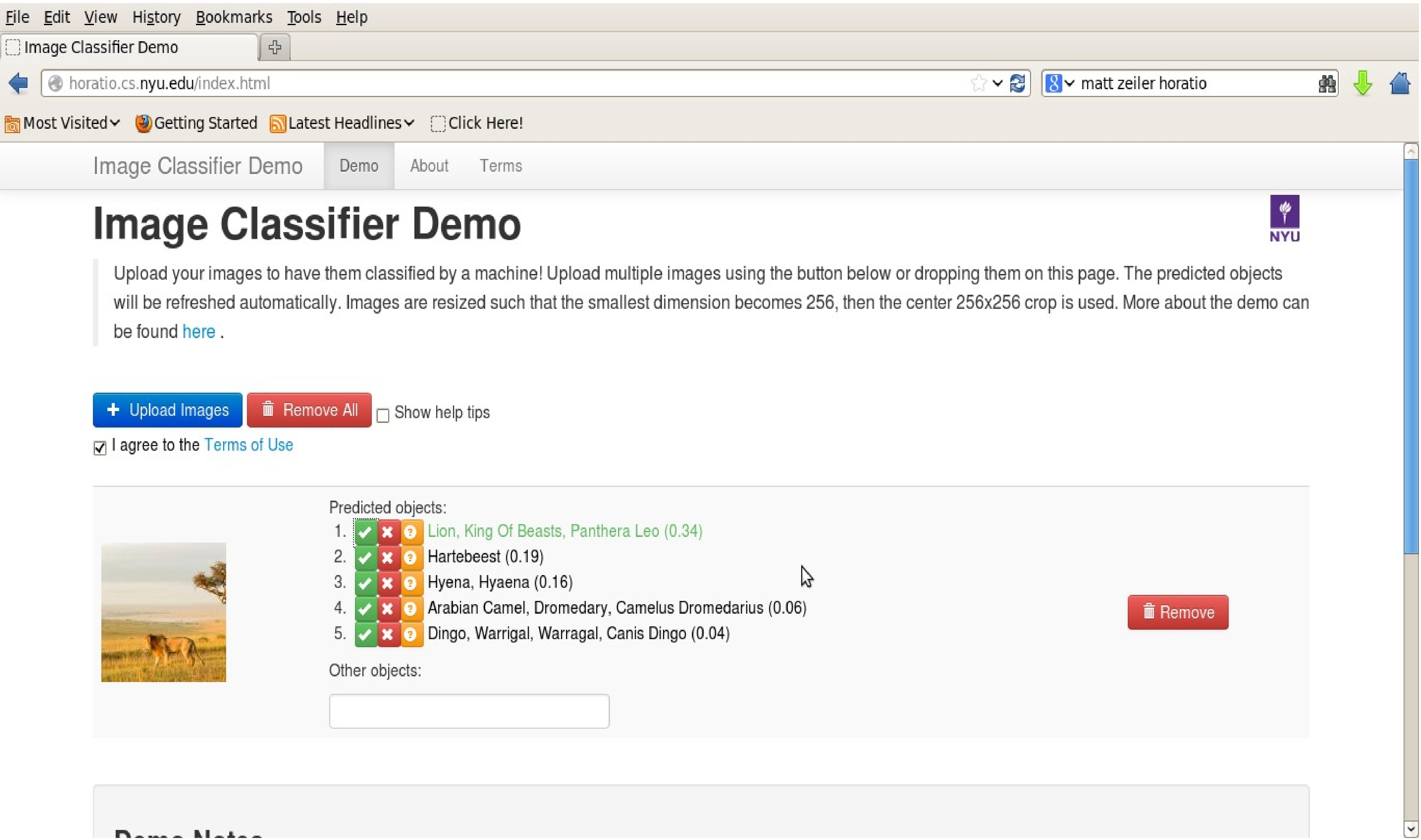


mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup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


TEST IMAGE
RETRIEVED IMAGES

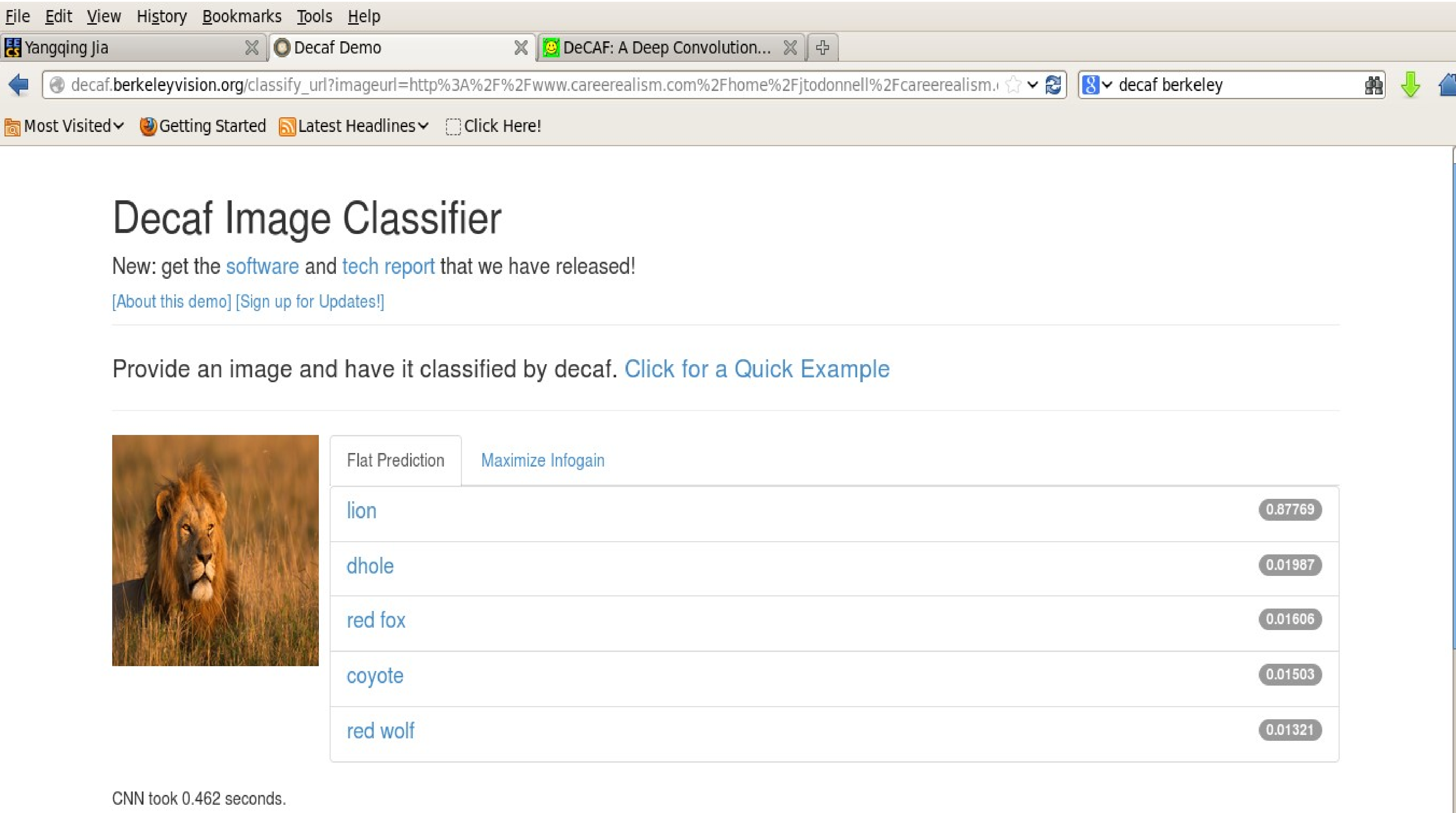
Demo of classifier by Matt Zeiler & Rob Fergus:

http://horatio.cs.nyu.edu/

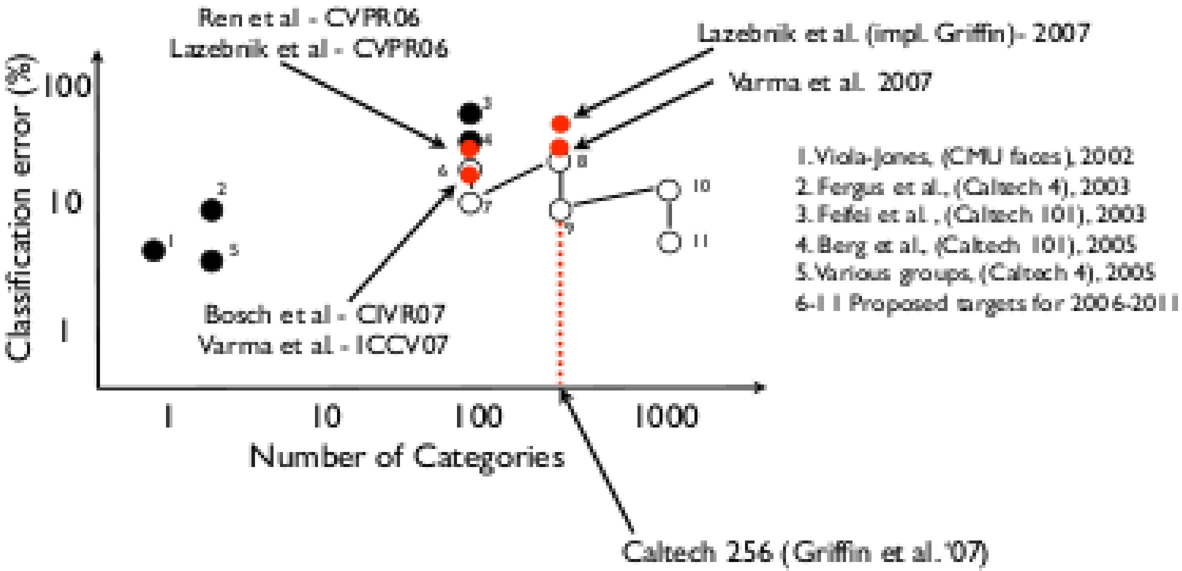


Demo of classifier by Yangqing Jia & Trevor Darrell:

http://decafberkeleyvision.org/



DeCAF arXiv 1310.1531 2013



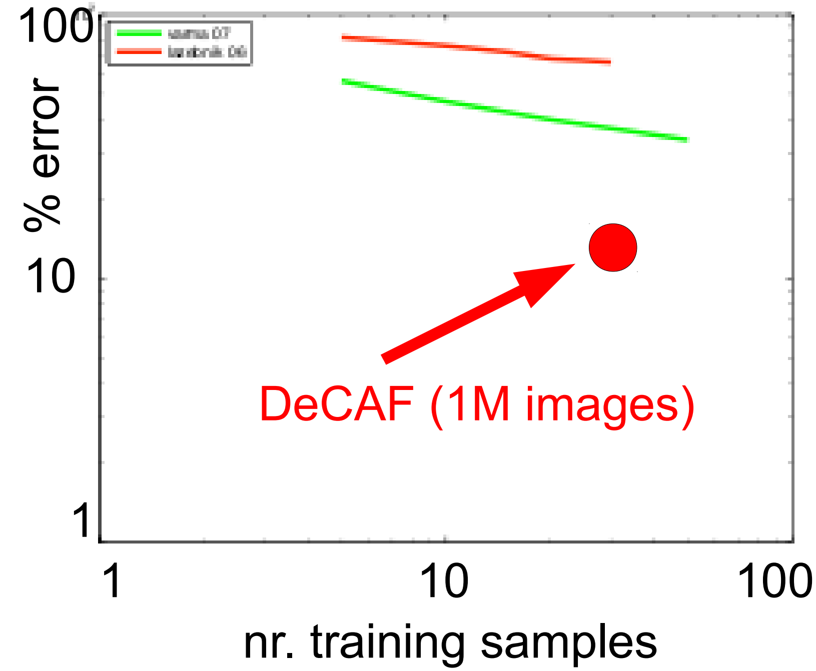


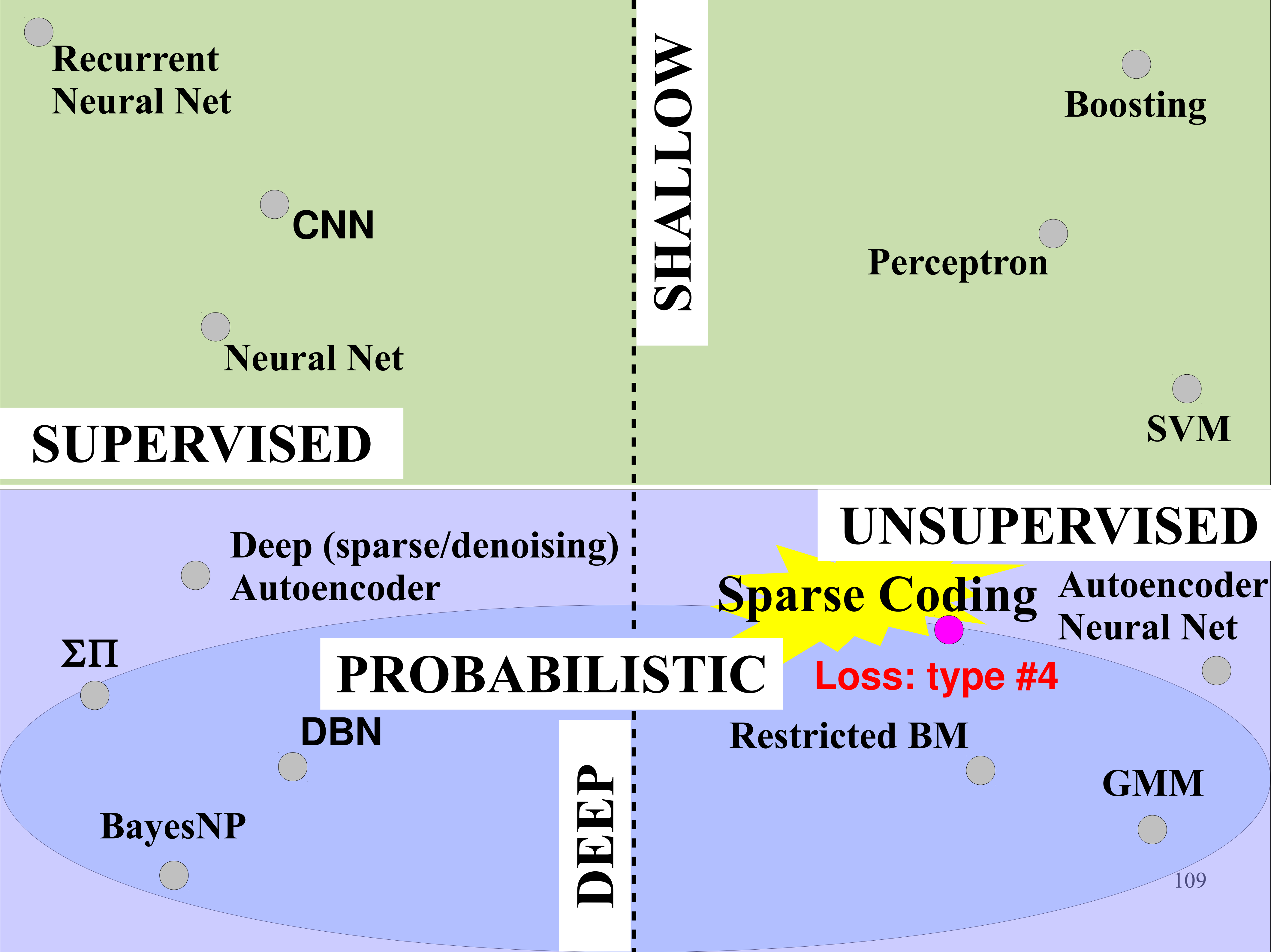
Figure 3: How well are we doing? (Left) Classification performance has seen steady improvement in the last few years, both in the number of categories on which algorithms are tested and in classification error rates. (Right) Performance of the best 2006 [Lazebnik et al., 2006] and the best 2007 algorithm [Varma, 2007] are compared here (classification error rates vs number of training examples). One may notice the significant year-on-year progress (see also left panel). Extrapolation enthusiasts may calculate that 10^8 training examples would be sufficient to achieve 1% error rates with current algorithms. Furthermore, if the pace of year-on-year progress is constant on this log scale chart, 1% error rates with 30 training examples will be achieved in 8-10 years.

Excerpt from Perona Visual Recognition 2007

Donahue, Jia et al. DeCAF arXiv 1310.1531 2013

Outline

- Theory: Energy-Based Models
  - Energy function
  - Loss function
- Examples:
    - Supervised learning: neural nets
    - Supervised learning: convnets
    - Unsupervised learning: sparse coding
    - Unsupervised learning: gated MRF
- Other examples
- Practical tricks



Recurrent
Neural Net
CNN
Neural Net
SHALLOW
Boosting
Perceptron
SVM
SUPERVISED
UNSUPERVISED
Deep (sparse/denoising)
Autoencoder
Sparse Coding
Autoencoder
Neural Net
ΣΠ
PROBABILISTIC
Loss: type #4
DBN
Restricted BM
GMM
BayesNP
DEEP

Energy & latent variables

- Energy may have latent variables.

- Two major approaches:
  - Marginalization (intractable if space is large)

$$\tilde{E}(\boldsymbol{y}) = \log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{y}, \boldsymbol{h}))$$

  - Minimization

$$\tilde{E}(\boldsymbol{y}) = min_{\boldsymbol{h}} E(\boldsymbol{y}, \boldsymbol{h})$$

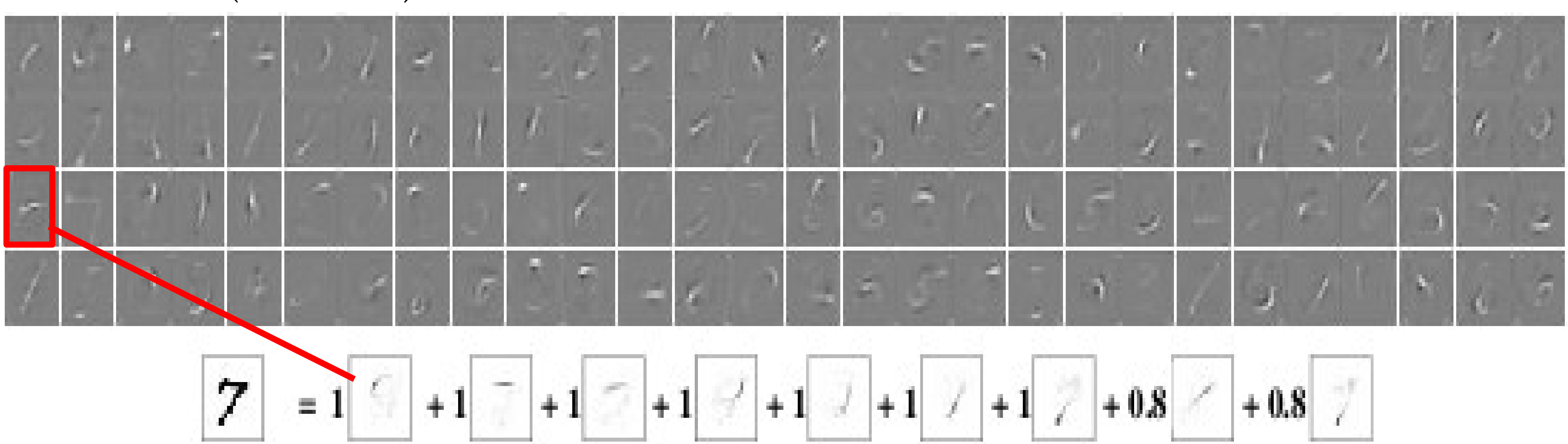
Sparse Coding

$$E(\boldsymbol{x},\boldsymbol{h};W)=\frac{1}{2}\|\boldsymbol{x}-W\boldsymbol{h}\|_2^2+\lambda\|\boldsymbol{h}\|_1$$

$$\tilde{E}(\boldsymbol{x};W)=min_{\boldsymbol{h}}E(\boldsymbol{x},\boldsymbol{h};W)$$

Loss type #4: energy loss with (sparsity) constraints.

$$L=\tilde{E}(\boldsymbol{x};W)$$



Olshausen & Field, Nature 1996

Ranzato et al. NIPS 2006

Inference

- Prediction of latent variables

$$E(\boldsymbol{x}, \boldsymbol{h}; W) = \frac{1}{2} \| \boldsymbol{x} - W \boldsymbol{h} \|_2^2 + \lambda \| \boldsymbol{h} \|_1$$

$$\boldsymbol{h}^* = arg\ min_{\boldsymbol{h}} E(\boldsymbol{x}, \boldsymbol{h}; W)$$

Inference is an iterative process.

Inference

- Prediction of latent variables

$$E(\boldsymbol{x},\boldsymbol{h};W)=\frac{1}{2}\|\boldsymbol{x}-W\boldsymbol{h}\|_2^2+\lambda\|\boldsymbol{h}\|_1$$

$$\boldsymbol{h}^*=arg\,min_{\boldsymbol{h}}E(\boldsymbol{x},\boldsymbol{h};W)$$

Inference is an iterative process.

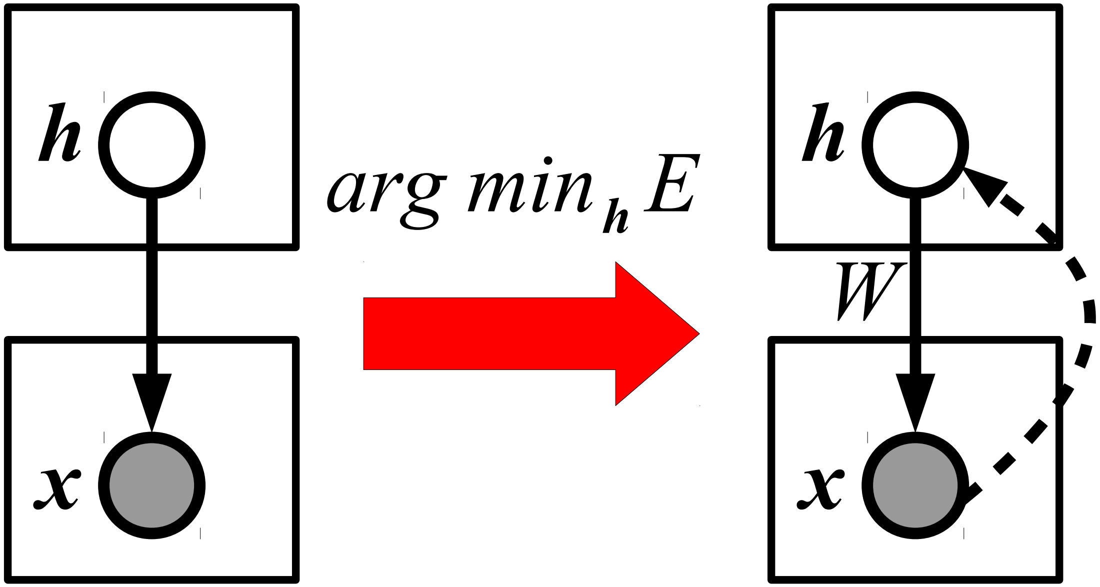
Q.: Is it possible to make inference more efficient?

A.: Yes, by training another module to directly predict

Kavukcuoglu et al. "Predictive Sparse Decomposition" ArXiv 2008

Inference in Sparse Coding

$$E(\boldsymbol{x},\boldsymbol{h})=\frac{1}{2}\|\boldsymbol{x}-W_2\boldsymbol{h}\|_2^2+\lambda\|\boldsymbol{h}\|_1$$

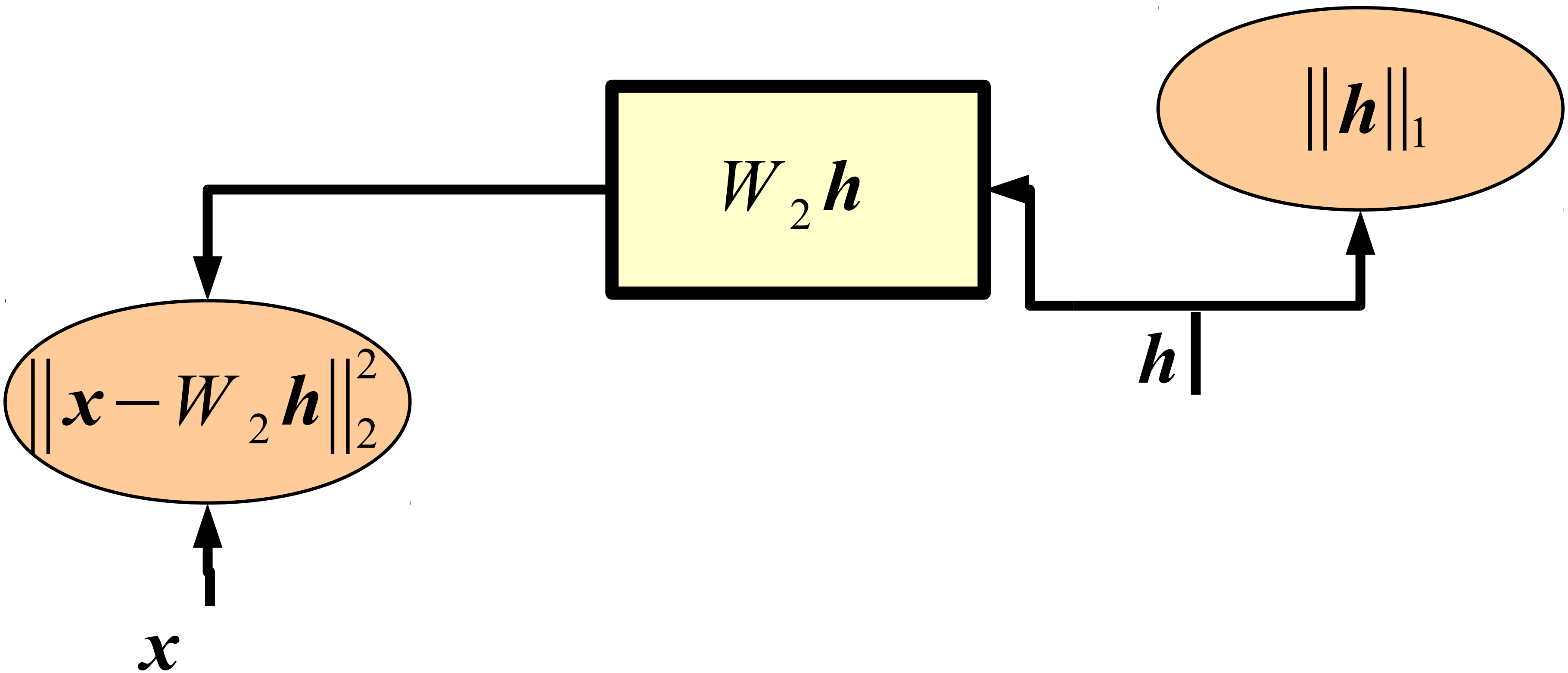


Kavukcuoglu et al. "Predictive Sparse Decomposition" ArXiv 2008

Ranzato

Inference in Sparse Coding

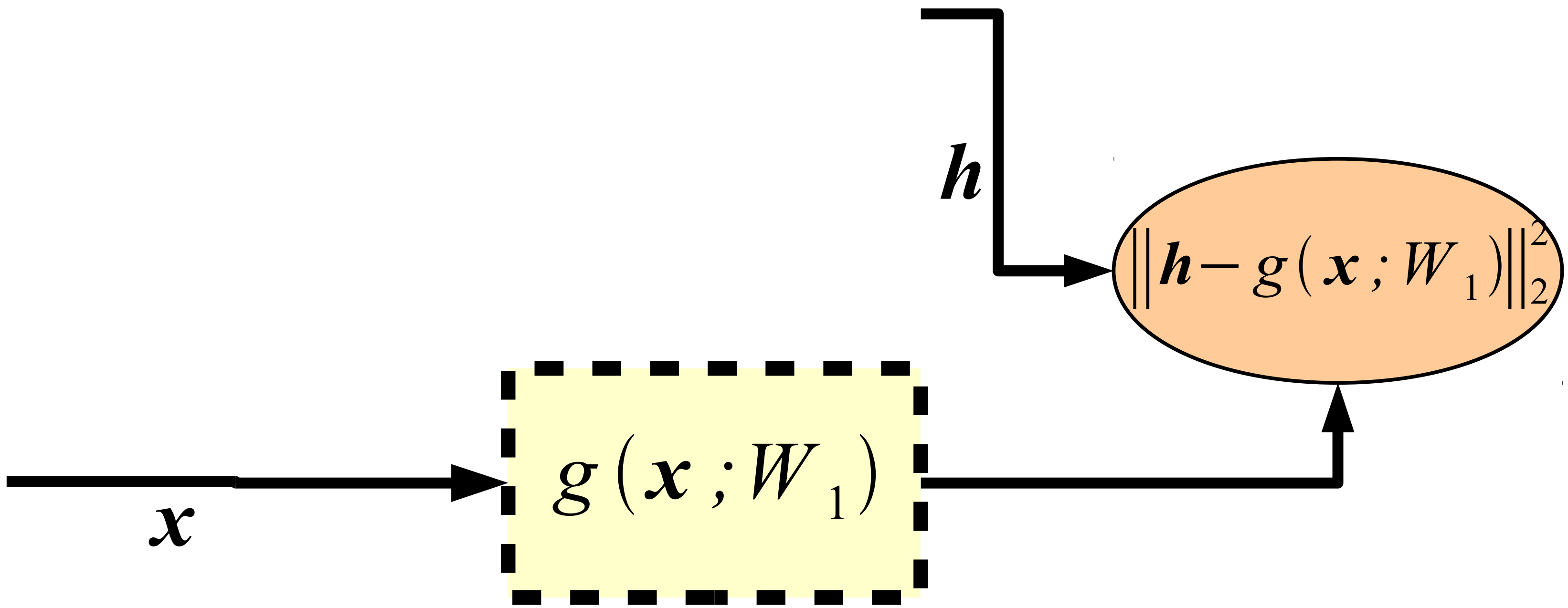
$$E(\boldsymbol{x},\boldsymbol{h})=\frac{1}{2}\|\boldsymbol{x}-W_2\boldsymbol{h}\|_2^2+\lambda\|\boldsymbol{h}\|_1$$



Kavukcuoglu et al. "Predictive Sparse Decomposition" ArXiv 2008

Learning To Perform Fast Inference

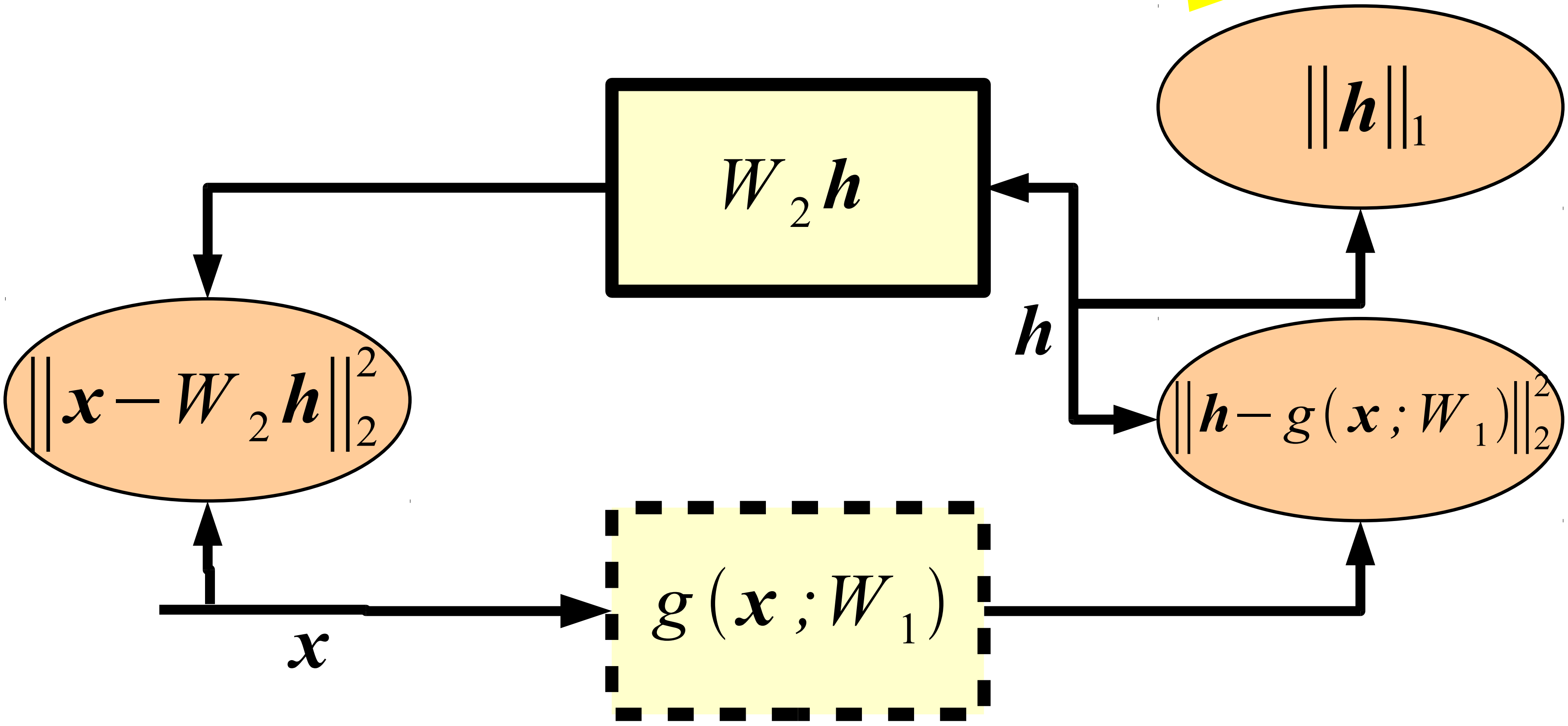
$$E(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{2}\|\boldsymbol{h} - g(\boldsymbol{x}; W_1)\|_2^2$$



Kavukcuoglu et al. “Predictive Sparse Decomposition” ArXiv 2008

Predictive Sparse Decomposition

$$E(\boldsymbol{x},\boldsymbol{h})=\frac{1}{2}\|\boldsymbol{x}-W_2\boldsymbol{h}\|_2^2+\lambda\|\boldsymbol{h}\|_1+\frac{1}{2}\|\boldsymbol{h}-g(\boldsymbol{x};W_1)\|_2^2$$



Kavukcuoglu et al. "Predictive Sparse Decomposition" ArXiv 2008

Sparse Auto-Encoders

- Example: Predictive Sparse Decomposition

$$E(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{2}\|\boldsymbol{x} - W_2 \boldsymbol{h}\|_2^2 + \lambda \|\boldsymbol{h}\|_1 + \frac{1}{2}\|\boldsymbol{h} - g(\boldsymbol{x}; W_1)\|_2^2$$

TRAINING:

For every sample:

Initialize $\boldsymbol{h} = g(\boldsymbol{x}; W_1)$

(E) Infer optimal latent variables: $\boldsymbol{h}^* = min_{\boldsymbol{h}} E(\boldsymbol{x}, \boldsymbol{h}; W)$

(M) Update parameters W_1, W_2

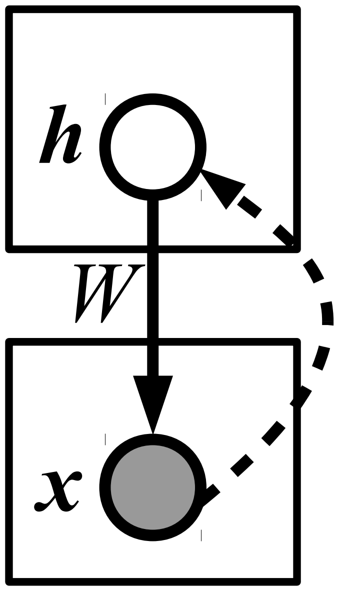
Inference at test time (fast):

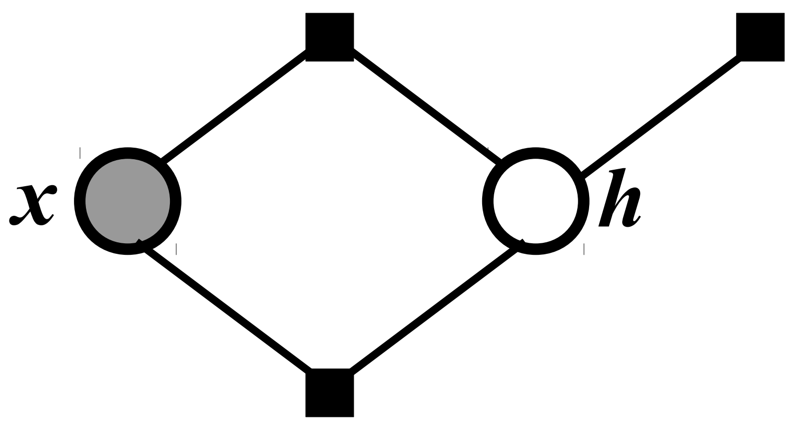
$$\boldsymbol{h}^* \approx g(\boldsymbol{x}; W_1)$$

Kavukcuoglu et al. "Predictive Sparse Decomposition" ArXiv 2008

Predictive Sparse Decomposition

$$E(\boldsymbol{x},\boldsymbol{h})=\frac{1}{2}\|\boldsymbol{x}-W_2\boldsymbol{h}\|_2^2+\lambda\|\boldsymbol{h}\|_1+\frac{1}{2}\|\boldsymbol{h}-g(\boldsymbol{x};W_1)\|_2^2$$

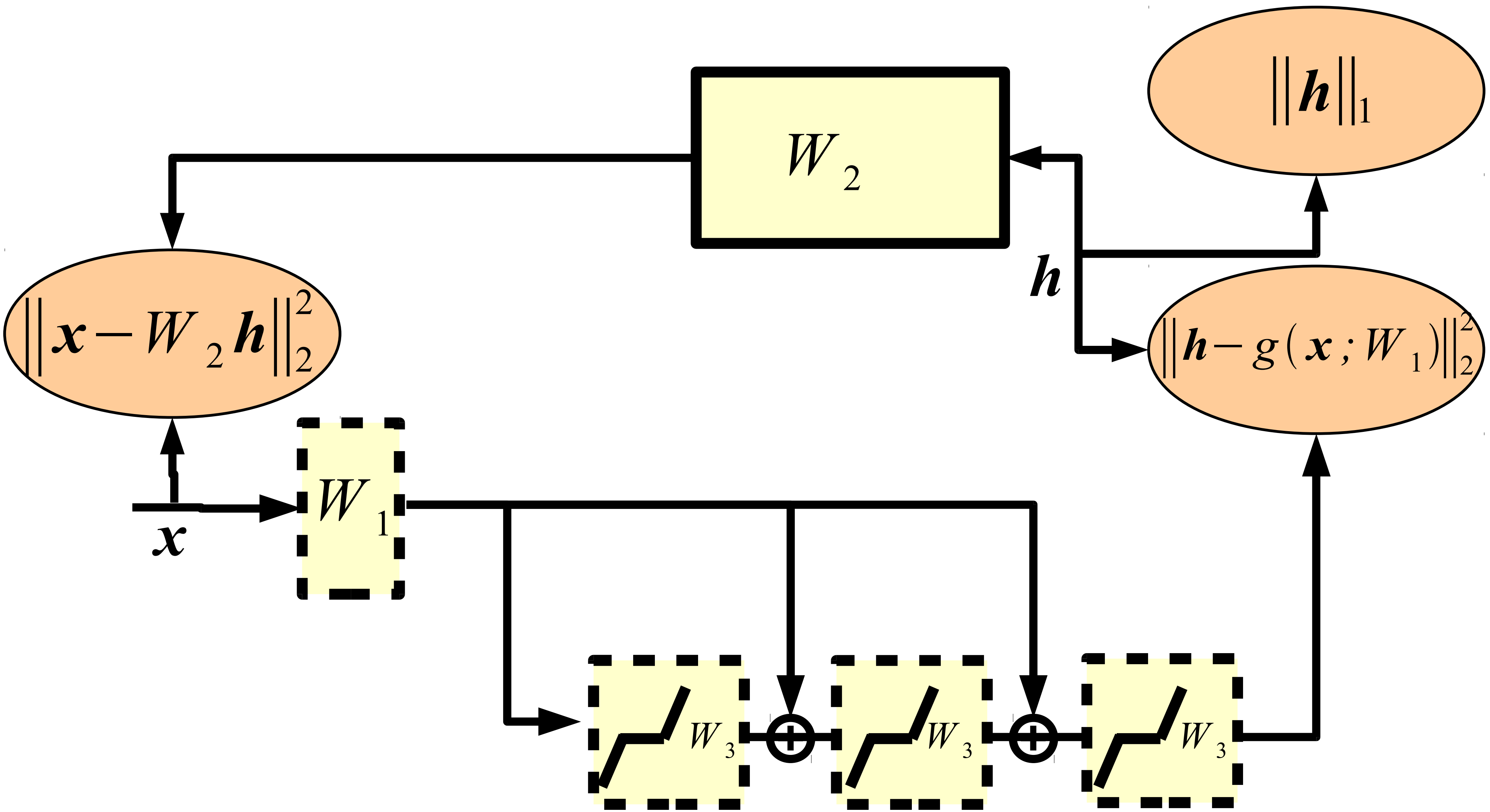




alternative graphical representations

Kavukcuoglu et al. "Predictive Sparse Decomposition" ArXiv 2008

LISTA



Gregor et al. “Learning fast approximations of sparse coding” ICML 2010

Ranzato

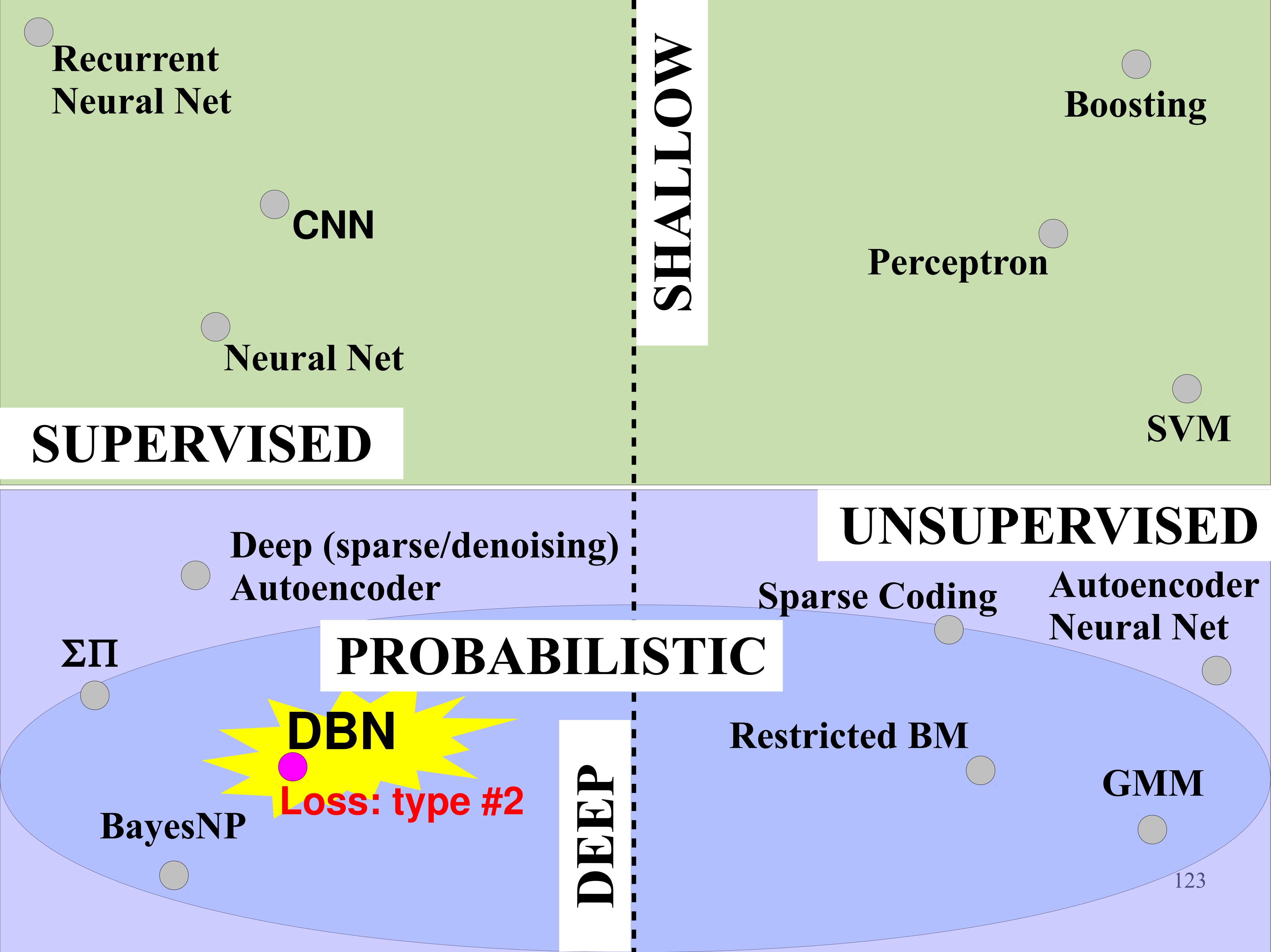
KEY IDEAS

- Inference can require expensive optimization
- We may approximate exact inference well by using a non-linear function (learn optimal approximation to perform fast inference)
- The original model and the fast predictor can be trained jointly

Kavukcuoglu et al. "Predictive Sparse Decomposition" ArXiv 2008
Kavukcuoglu et al. "Learning convolutonal feature hierarchies.." NIPS 2010
Gregor et al. "Structured sparse coding via lateral inhibition" NIPS 2011
Szlam et al. "Fast approximations to structured sparse coding..." ECCV 2012
Rolfe et al. "Discriminative Recurrent Sparse Autoencoders" ICLR 2013

Outline

- Theory: Energy-Based Models
  - Energy function
  - Loss function
- Examples:
  - Supervised learning: neural nets
  - Supervised learning: convnets
  - Unsupervised learning: sparse coding
  - Unsupervised learning: gated MRF
- Other examples
- Practical tricks

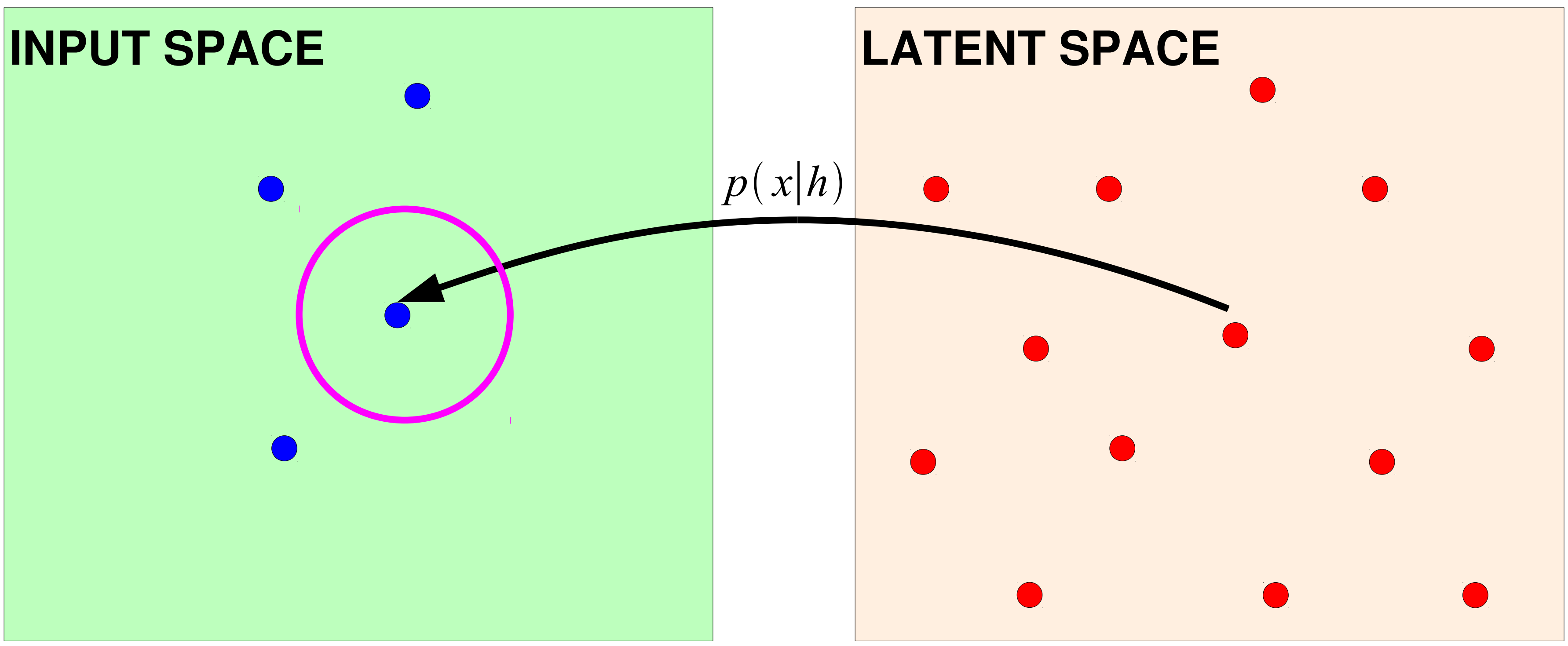


SHALLOW
Recurrent Neural Net
Boosting
CNN
Perceptron
Neural Net
SVM
SUPERVISED
UNSUPERVISED
Deep (sparse/denoising) Autoencoder
Sparse Coding
Autoencoder Neural Net
ΣΠ
PROBABILISTIC
DBN
Loss: type #2
Restricted BM
GMM
BayesNP
DEEP

Probabilistic Models of Natural Images

$$p(x|h) = N(mean(h), D)$$

- examples: PPCA, Factor Analysis, ICA, Gaussian RBM



● **Training sample** ● **Latent vector**

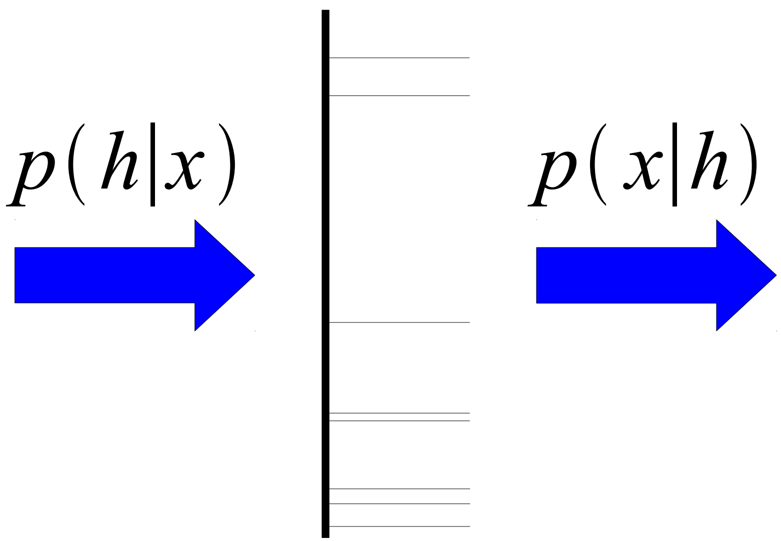
Ranzato

Probabilistic Models of Natural Images

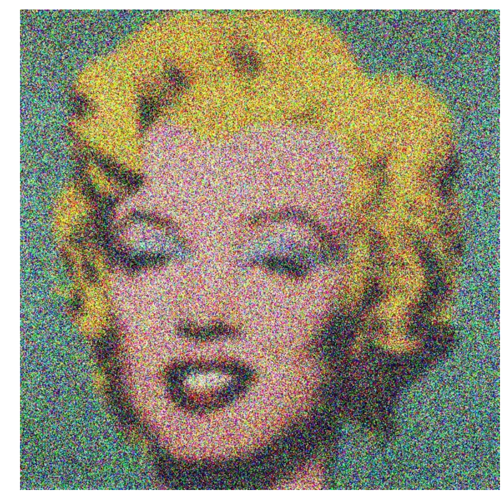
$$p(x|h) = N(mean(h), D)$$

- examples: PPCA, Factor Analysis, ICA, Gaussian RBM

input image



latent variables



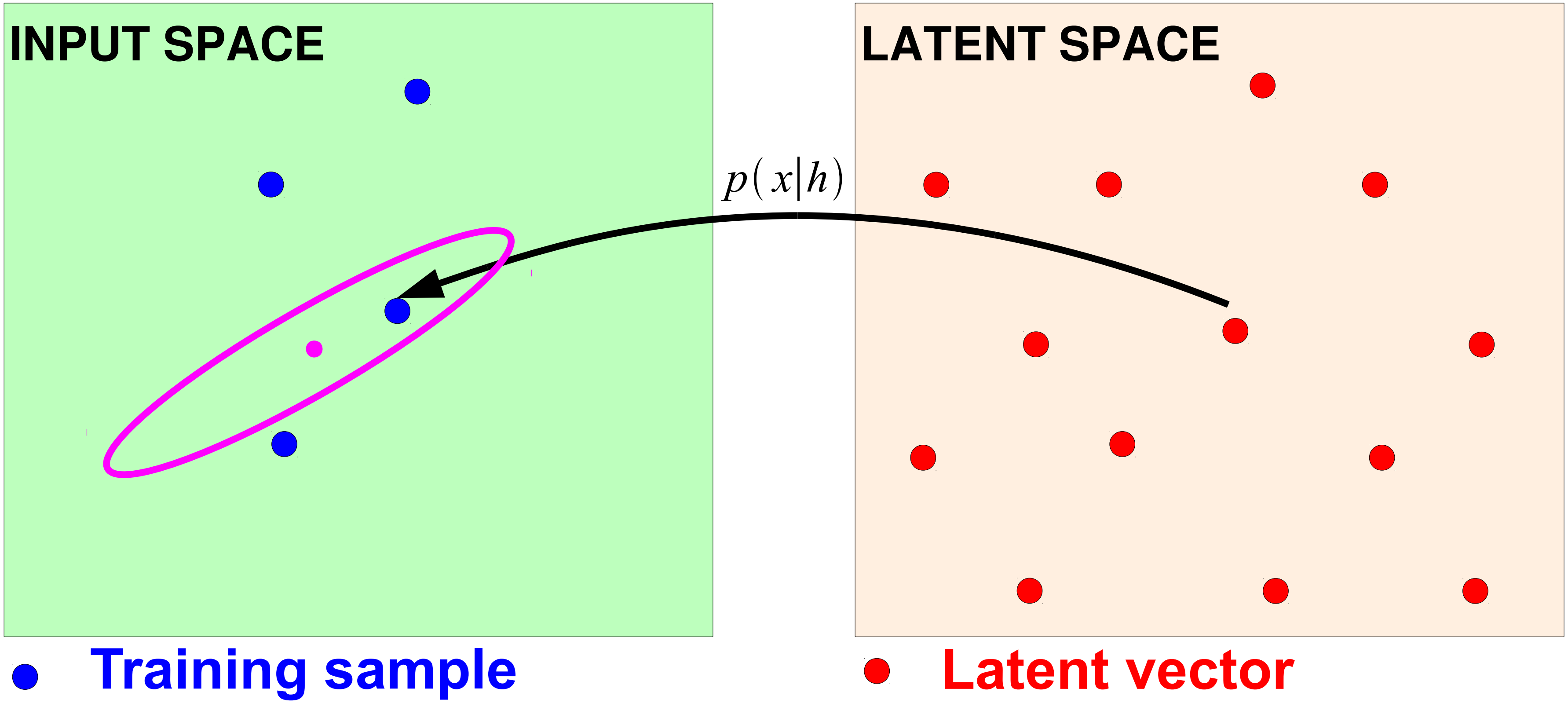
generated image

model does not represent well dependecies, only mean intensity

Probabilistic Models of Natural Images

$$p(x|h) = N(0, Covariance(h))$$

- examples: PoT, cRBM



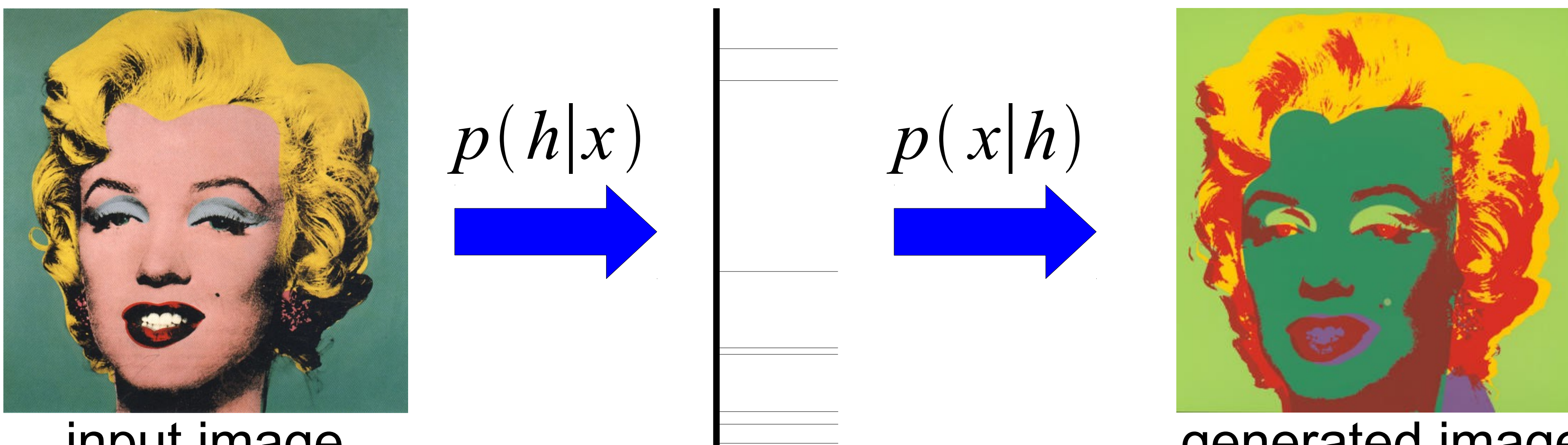
Welling et al. NIPS 2003, Ranzato et al. AISTATS 10

Ranzato

Probabilistic Models of Natural Images

$$p(x|h) = N(0, Covariance(h))$$

- examples: PoT, cRBM



input image

latent variables

generated image

model does not represent well mean intensity, only dependencies

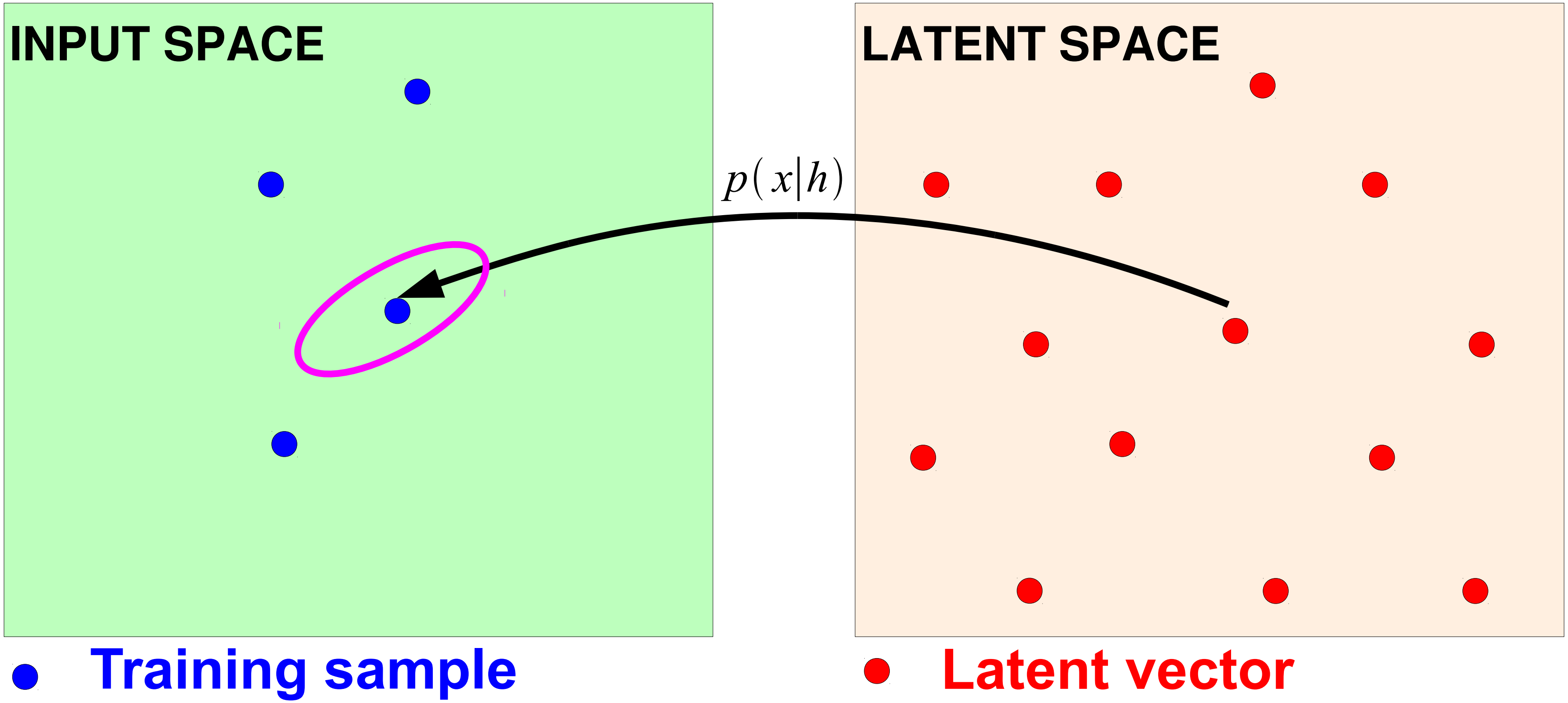
Welling et al. NIPS 2003, Ranzato et al. AISTATS 10

Andy Warhol 1960

Probabilistic Models of Natural Images

$$p(x|h) = N(mean(h), Covariance(h))$$

- this is what we propose: mcRBM, mPoT

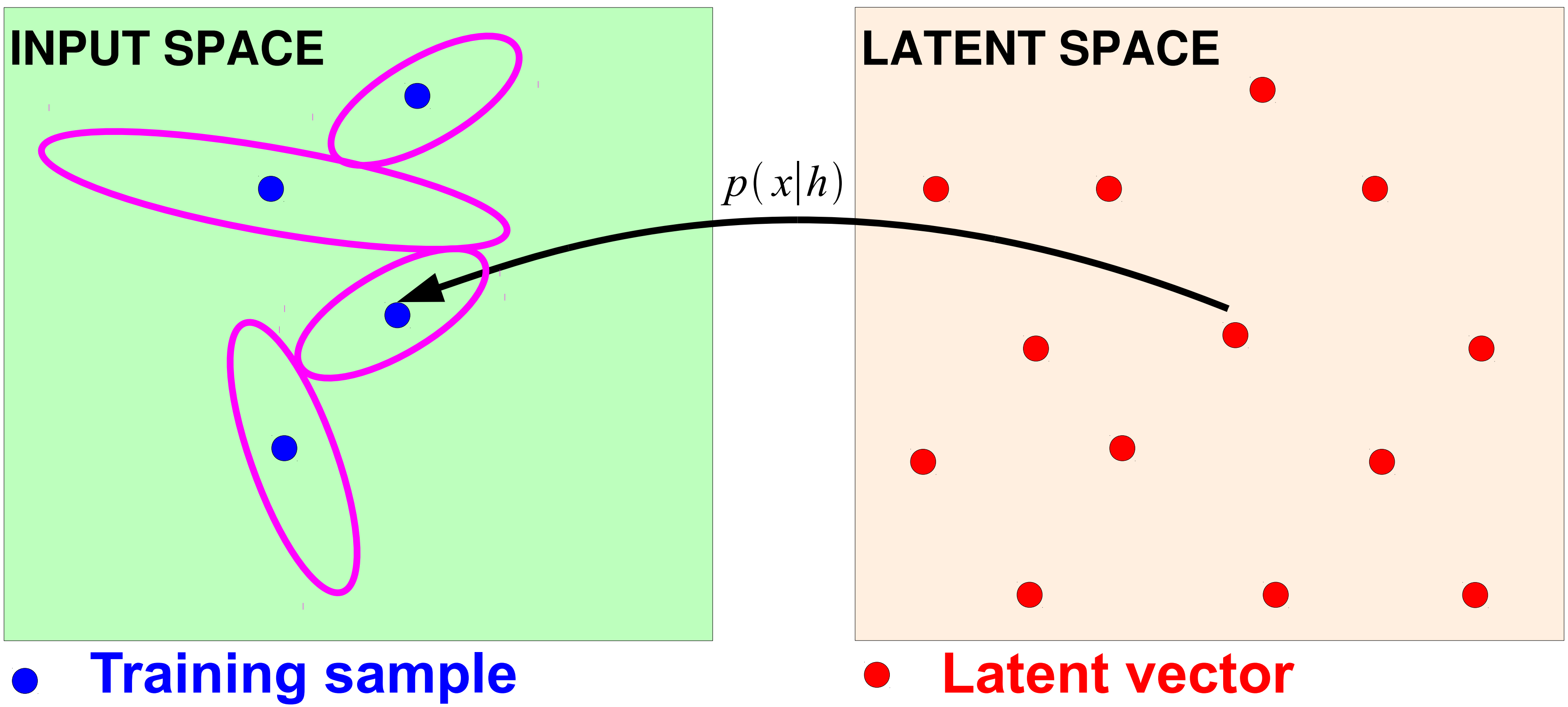


Ranzato et al. CVPR 10, Ranzato et al. NIPS 2010, Ranzato et al. CVPR 11

Probabilistic Models of Natural Images

$$p(x|h) = N(mean(h), Covariance(h))$$

- this is what we propose: mcRBM, mPoT



Ranzato et al. CVPR 10, Ranzato et al. NIPS 2010, Ranzato et al. CVPR 11

PoT

N(0,Σ)

PPCA

N(m,I)

Our model

N(m,Σ)

Deep Gated MRF

Layer 1:

$$E(x, h^c, h^m) = \frac{1}{2} x' \Sigma^{-1} x \qquad p(x, h^c, h^m) \alpha\, e^{-E(x, h^c, h^m)}$$

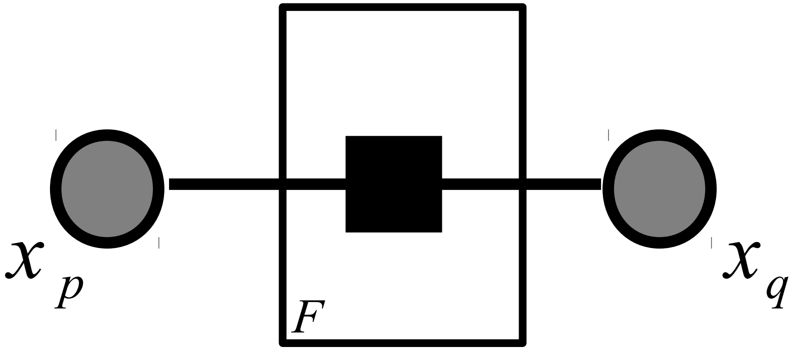
pair-wise MRF

Ranzato et al. “Modeling natural images with gated MRFs” PAMI 2013

Deep Gated MRF

Layer 1:

$$E(x, h^c, h^m) = \frac{1}{2} x' C C' x$$



pair-wise MRF

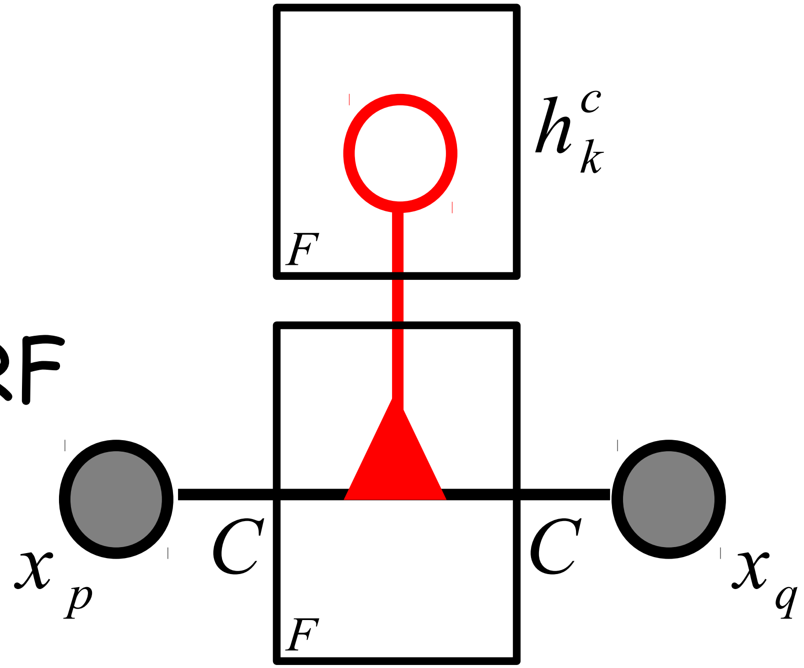
Ranzato et al. “Modeling natural images with gated MRFs” PAMI 2013

Deep Gated MRF

Layer 1:

$$E(x, h^c, h^m) = \frac{1}{2} x' C [\mathrm{diag}(h^c)] C' x$$

gated MRF



Ranzato et al. “Modeling natural images with gated MRFs” PAMI 2013

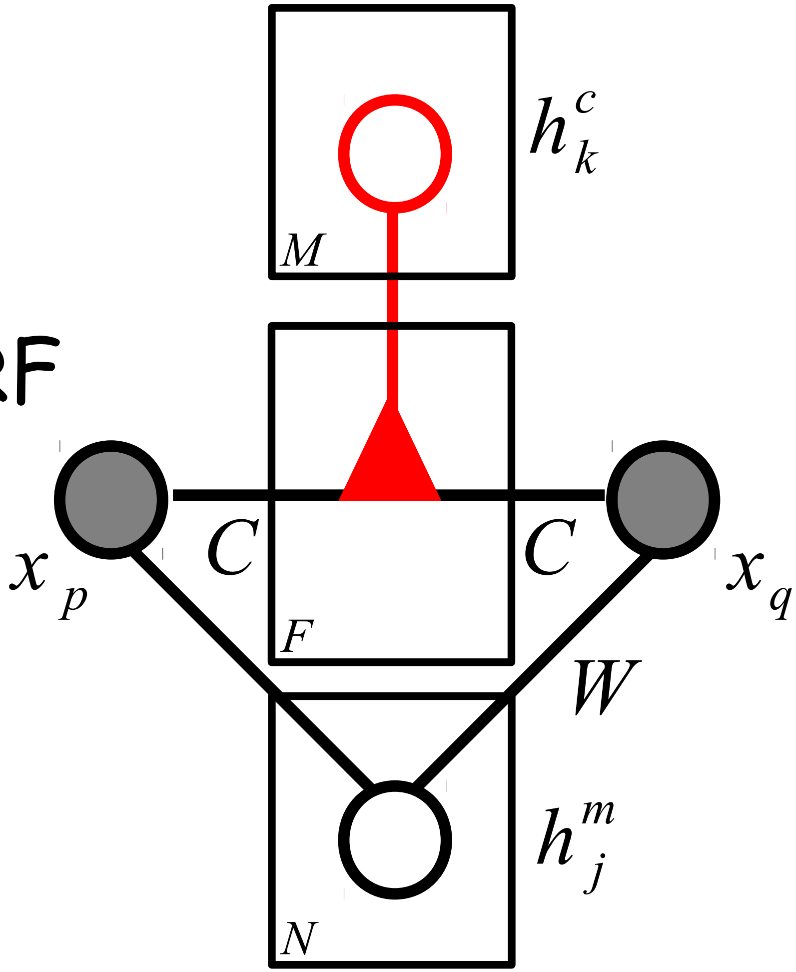
Ranzato

Deep Gated MRF

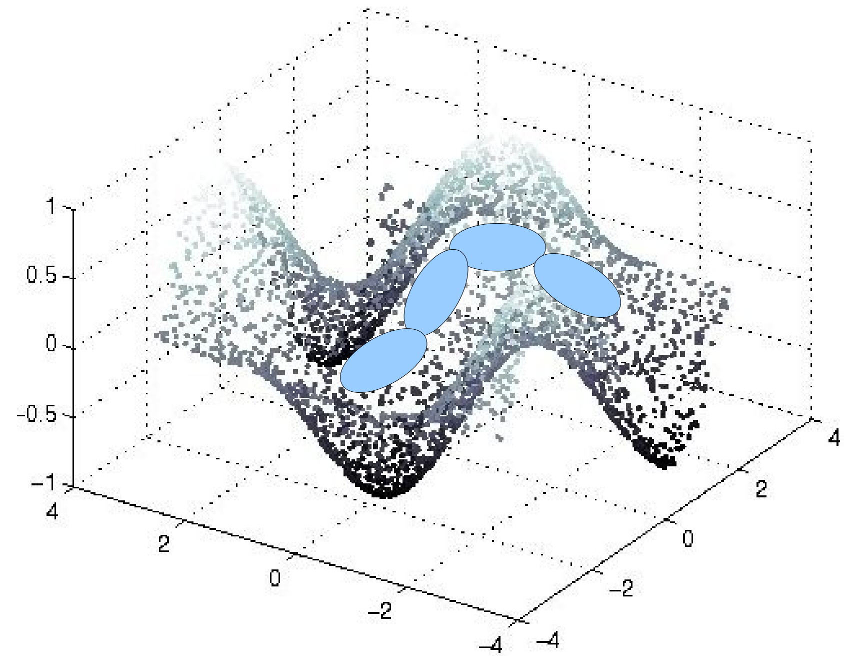
Layer 1:

$$E(x, h^c, h^m) = \frac{1}{2} x' C [diag(h^c)] C' x + \frac{1}{2} x' x - x' W h^m$$

gated MRF



$$p(x) \alpha \int_{h^c} \int_{h^m} e^{-E(x, h^c, h^m)}$$



Ranzato et al. “Modeling natural images with gated MRFs” PAMI 2013

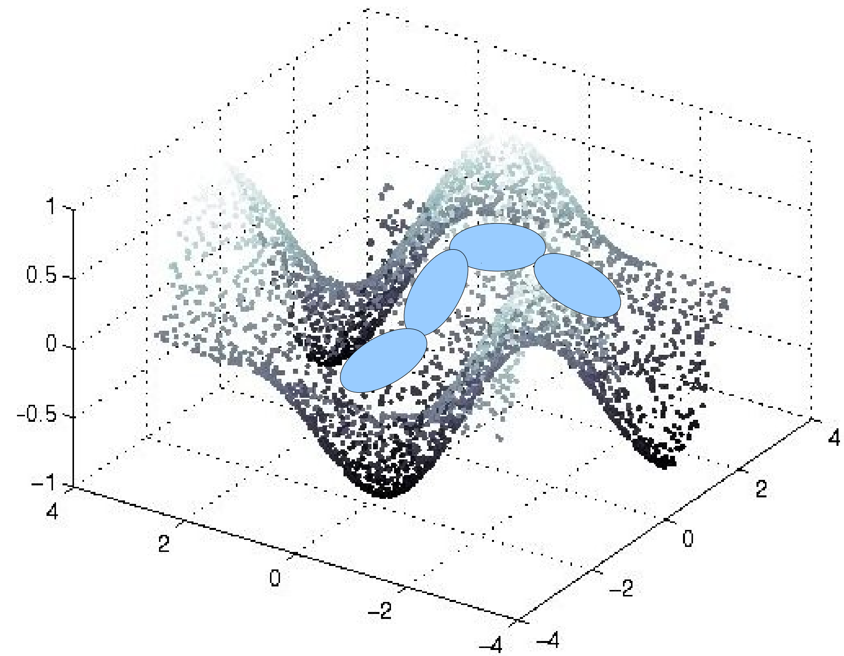
Deep Gated MRF

Layer 1:

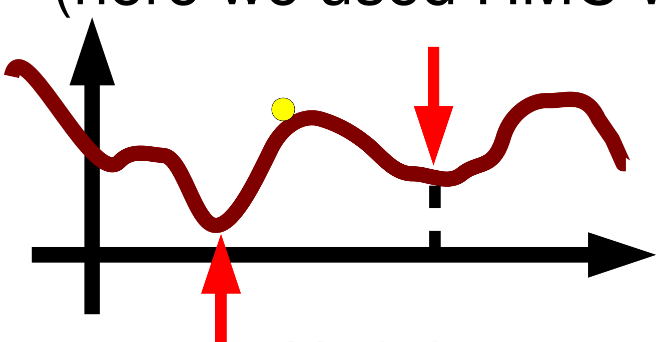
$$E(x, h^c, h^m) = \frac{1}{2} x' C [diag(h^c)] C' x + \frac{1}{2} x' x - x' W h^m$$

Inference of latent variables:
just a forward pass

$$p(x) \alpha \int_{h^c} \int_{h^m} e^{-E(x, h^c, h^m)}$$



Training:
requires approximations
(here we used HMC with PCD)



Loss: type #2

Ranzato et al. "Modeling natural images with gated MRFs" PAMI 2013

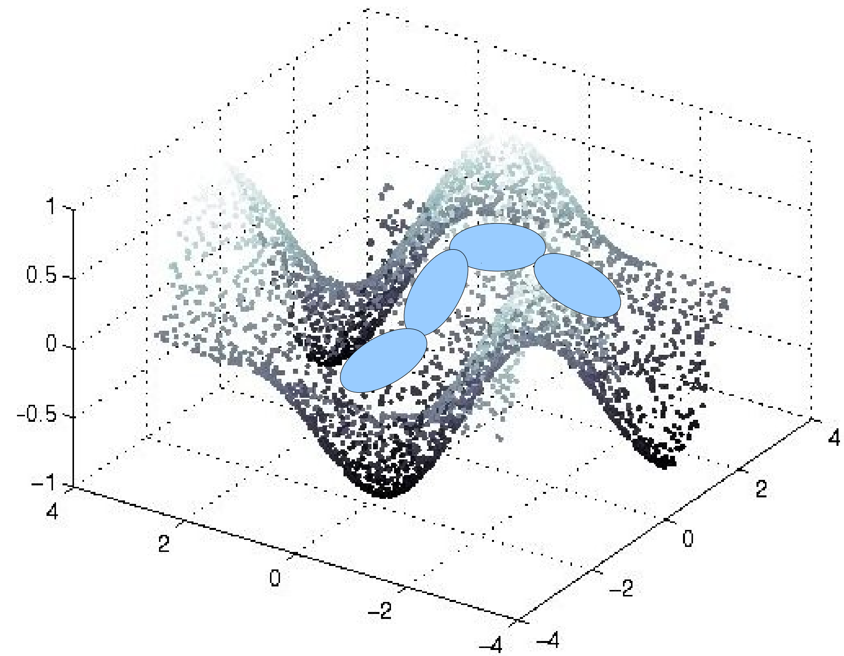
Deep Gated MRF

Layer 1:

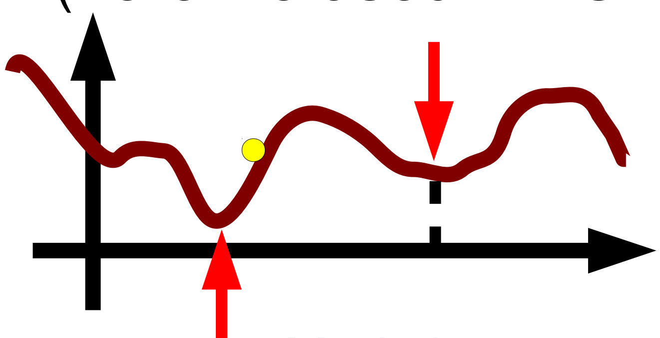
$$E(x, h^c, h^m) = \frac{1}{2} x' C [diag(h^c)] C' x + \frac{1}{2} x' x - x' W h^m$$

Inference of latent variables:
just a forward pass

$$p(x) \alpha \int_{h^c} \int_{h^m} e^{-E(x, h^c, h^m)}$$



Training:
requires approximations
(here we used HMC with PCD)



Loss: type #2

Ranzato et al. "Modeling natural images with gated MRFs" PAMI 2013

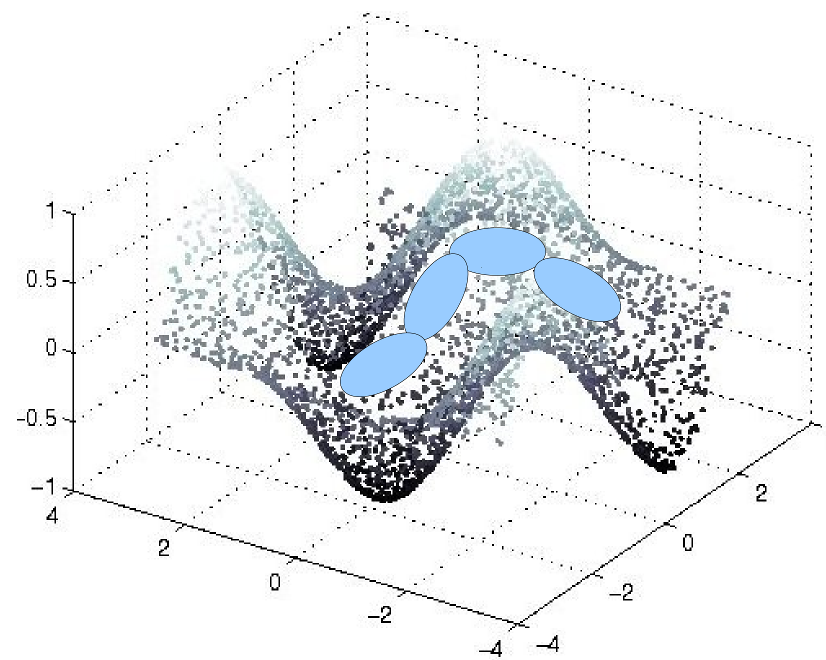
Deep Gated MRF

Layer 1:

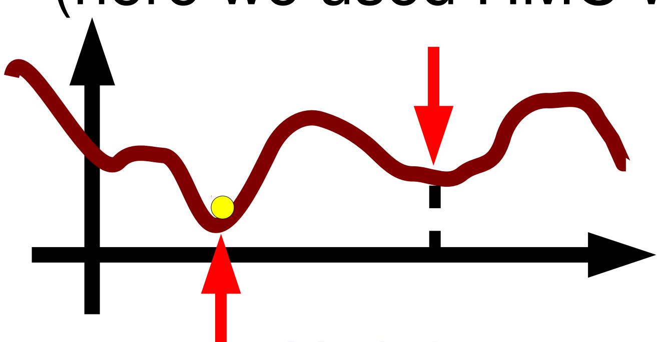
$$E(x, h^c, h^m) = \frac{1}{2} x' C [diag(h^c)] C' x + \frac{1}{2} x' x - x' W h^m$$

Inference of latent variables:
just a forward pass

$$p(x) \alpha \int_{h^c} \int_{h^m} e^{-E(x, h^c, h^m)}$$



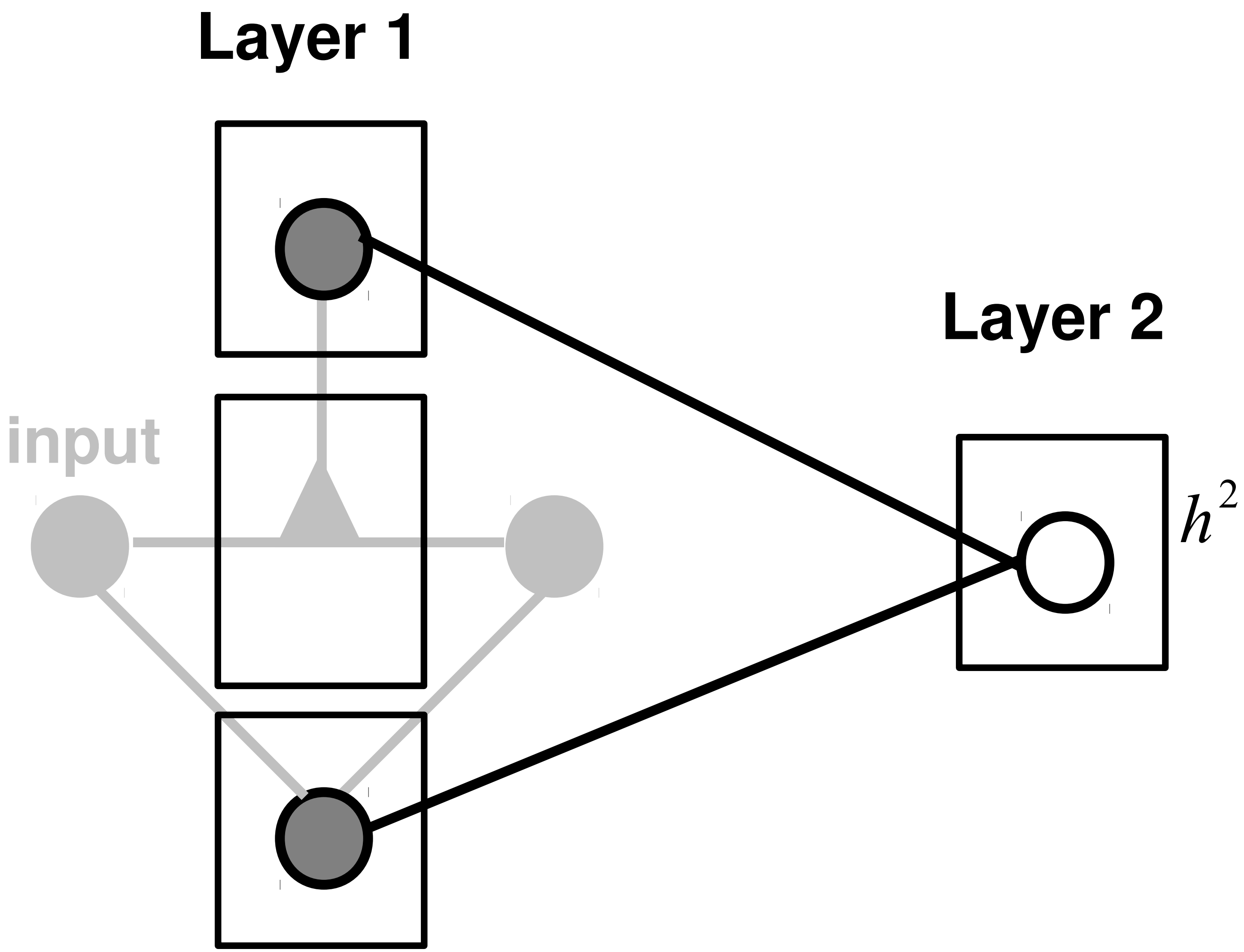
Training:
requires approximations
(here we used HMC with PCD)



Loss: type #2

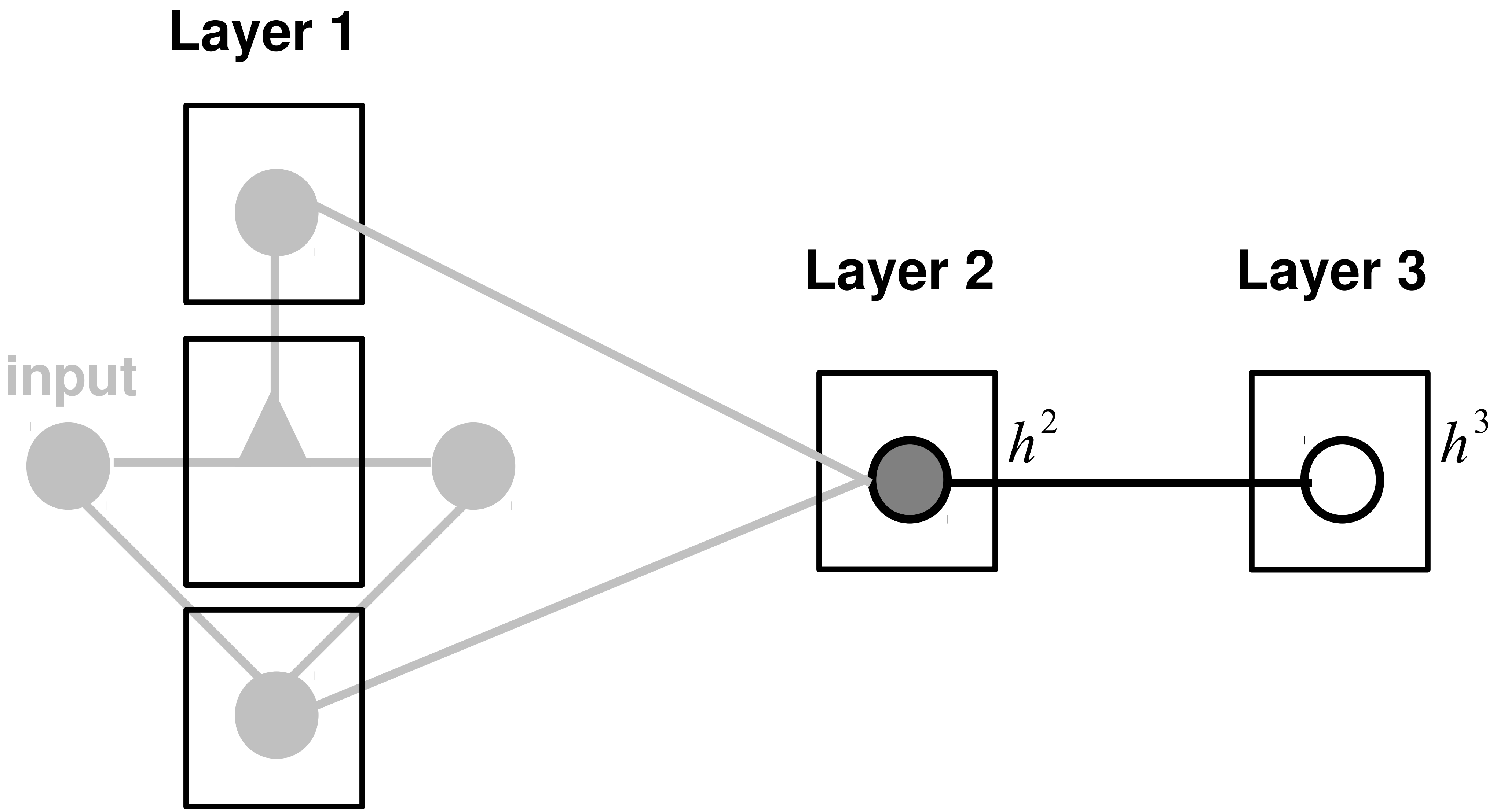
Ranzato et al. "Modeling natural images with gated MRFs" PAMI 2013

Deep Gated MRF



Ranzato et al. “Modeling natural images with gated MRFs” PAMI 2013

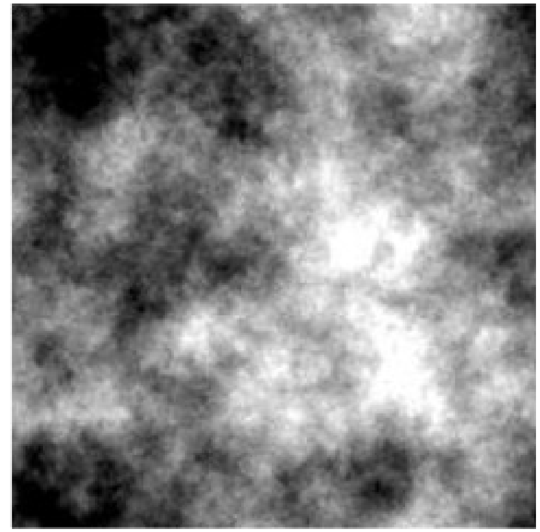
Deep Gated MRF



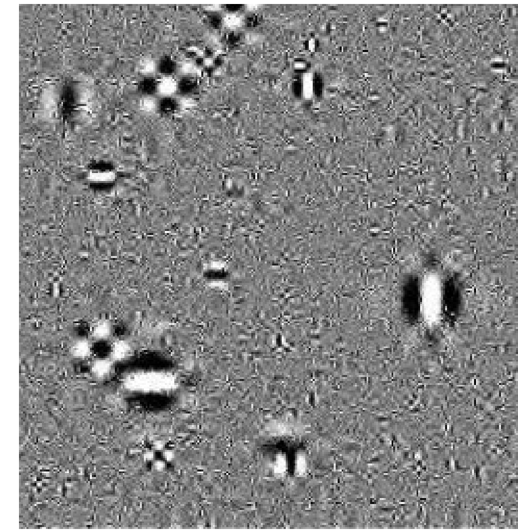
Ranzato et al. "Modeling natural images with gated MRFs" PAMI 2013

Sampling High-Resolution Images

Gaussian model

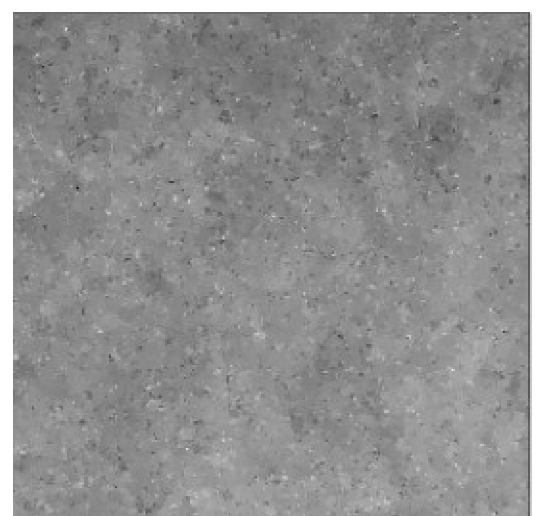


marginal wavelet

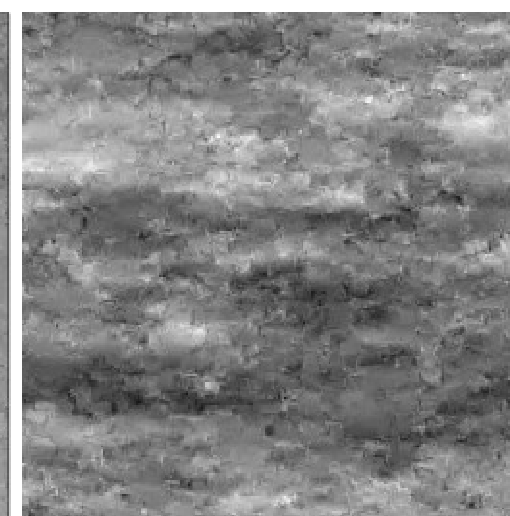


from Simoncelli 2005

Pair-wise MRF



FoE



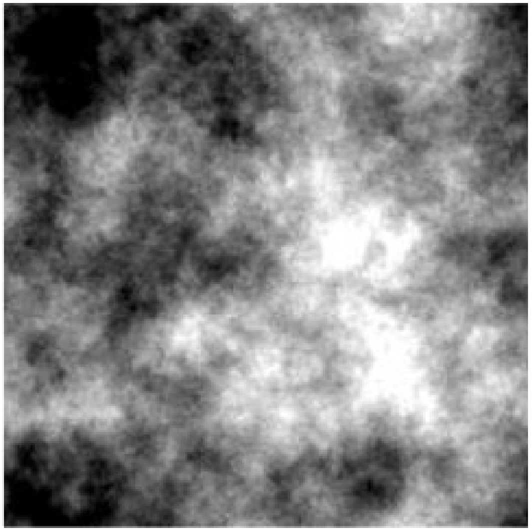
from Schmidt, Gao, Roth CVPR 2010

Sampling High-Resolution Images

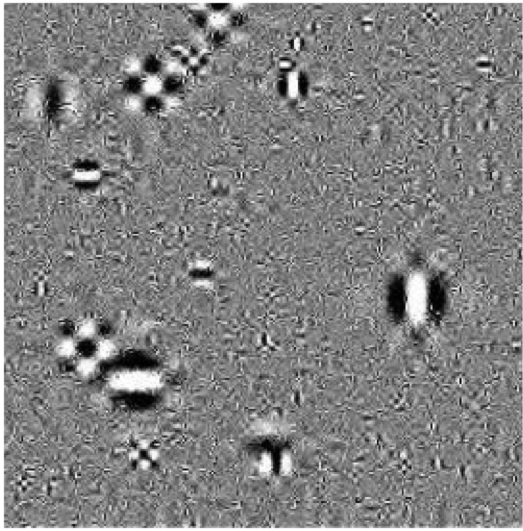
gMRF: **1 layer**

Ranzato et al. PAMI 2013

Gaussian model

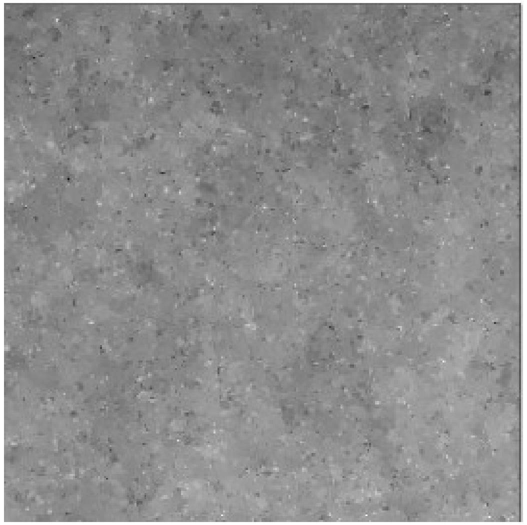


marginal wavelet



from Simoncelli 2005

Pair-wise MRF



FoE

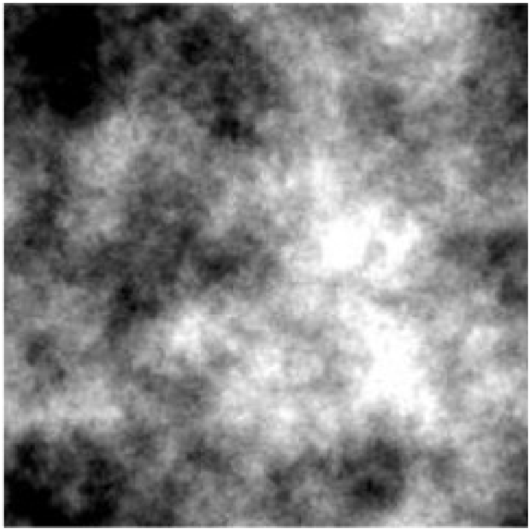
from Schmidt, Gao, Roth CVPR 2010

Sampling High-Resolution Images

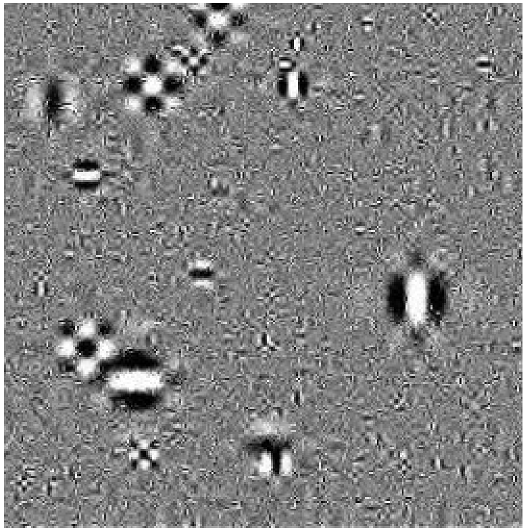
gMRF: **1 layer**

Ranzato et al. PAMI 2013

Gaussian model

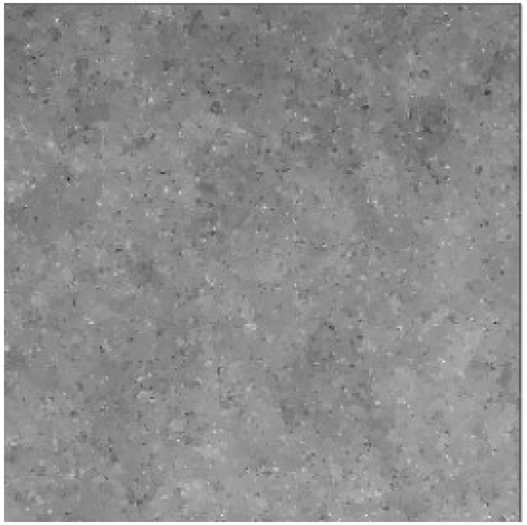


marginal wavelet



from Simoncelli 2005

Pair-wise MRF

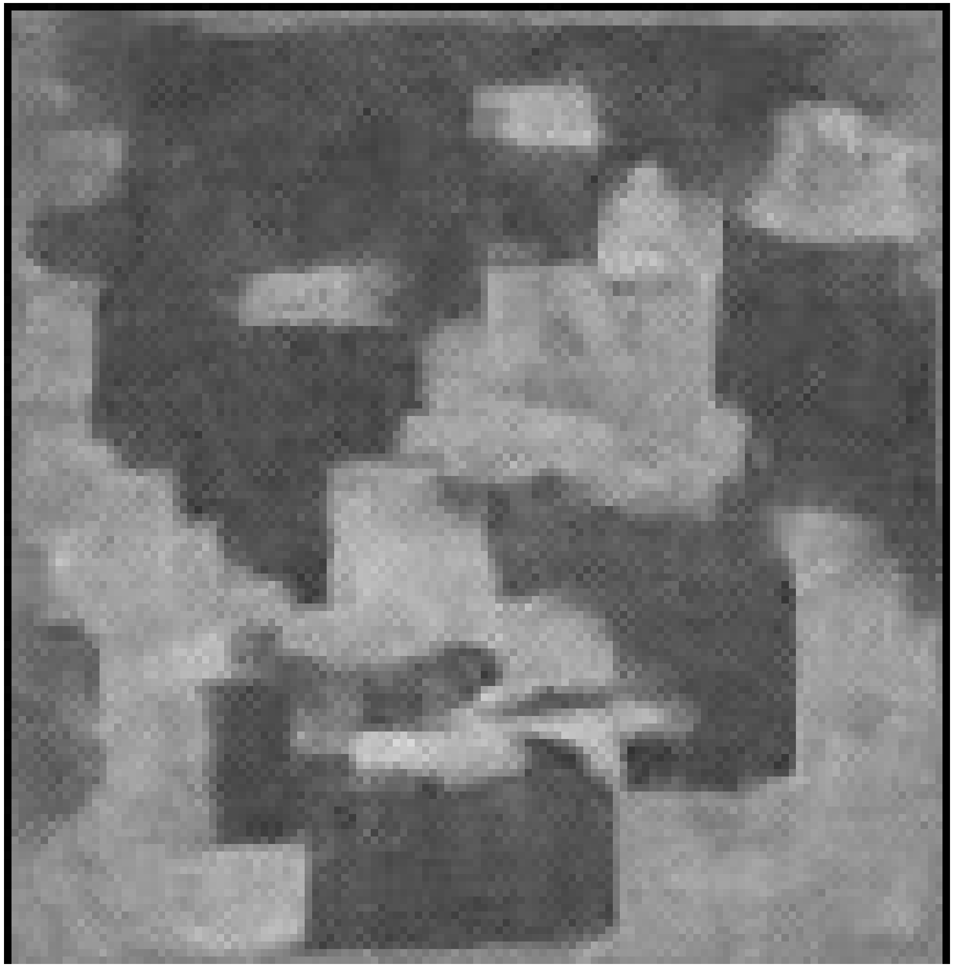


FoE

from Schmidt, Gao, Roth CVPR 2010

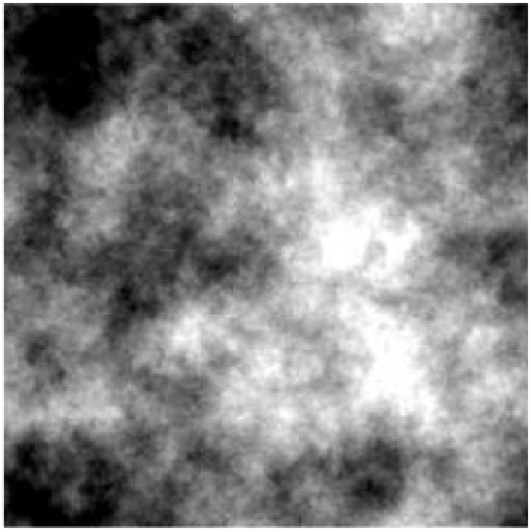
Sampling High-Resolution Images

gMRF: **1 layer**

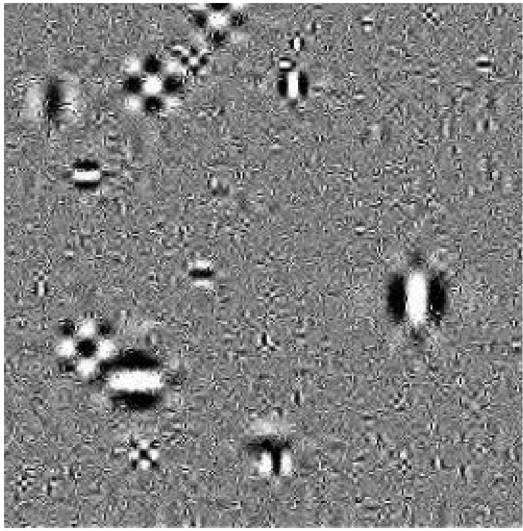


Ranzato et al. PAMI 2013

Gaussian model

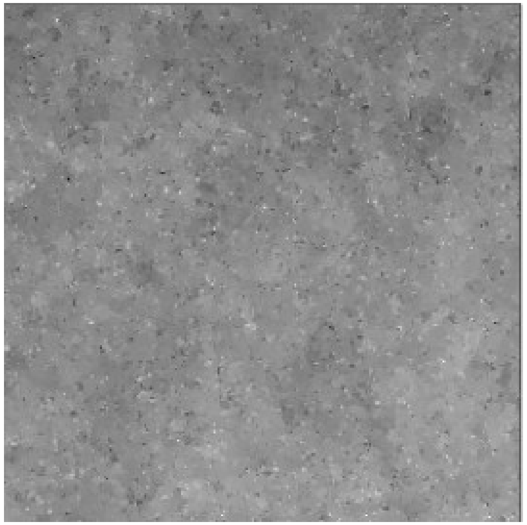


marginal wavelet



from Simoncelli 2005

Pair-wise MRF



FoE

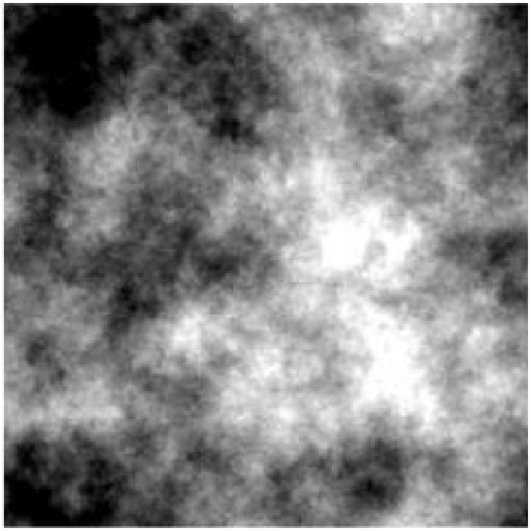
from Schmidt, Gao, Roth CVPR 2010

Sampling High-Resolution Images

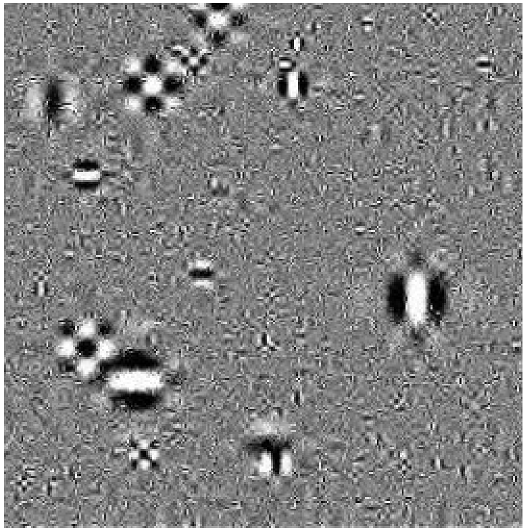
gMRF: **3 layer**

Ranzato et al. PAMI 2013

Gaussian model

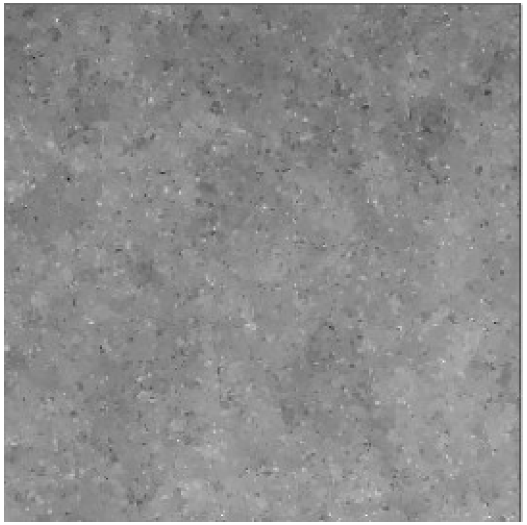


marginal wavelet



from Simoncelli 2005

Pair-wise MRF

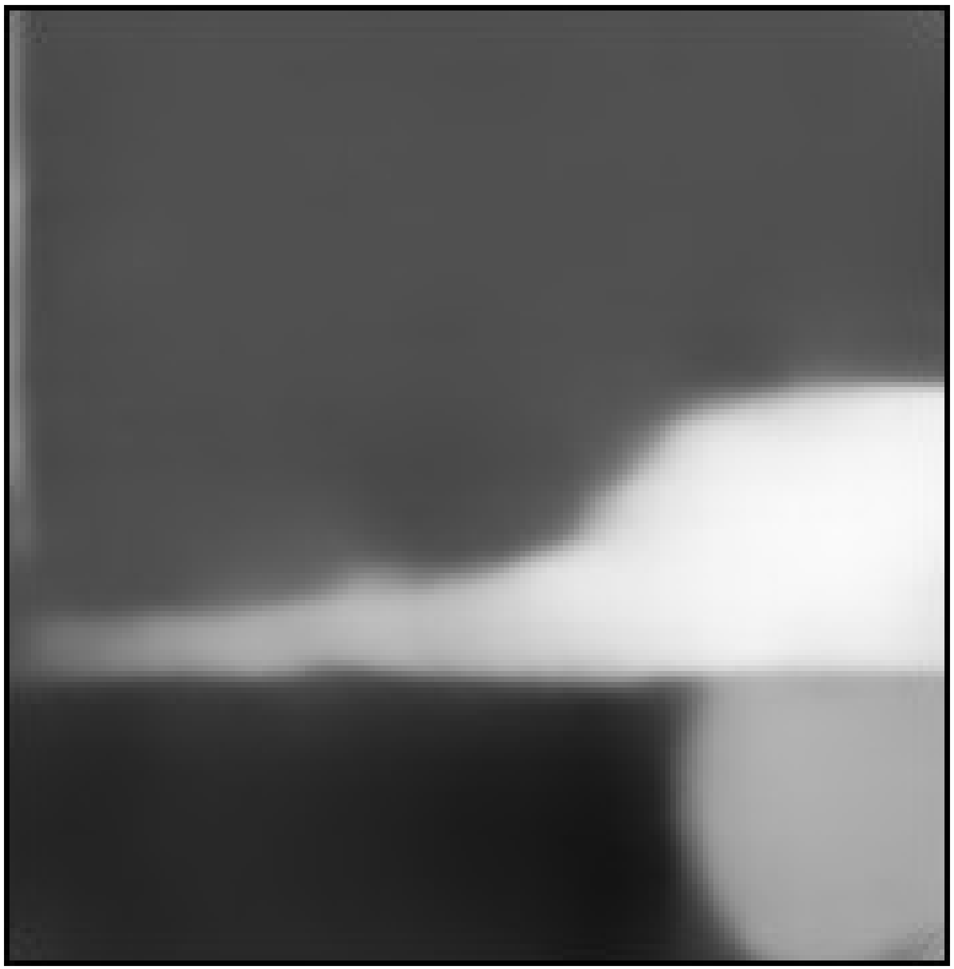


FoE

from Schmidt, Gao, Roth CVPR 2010

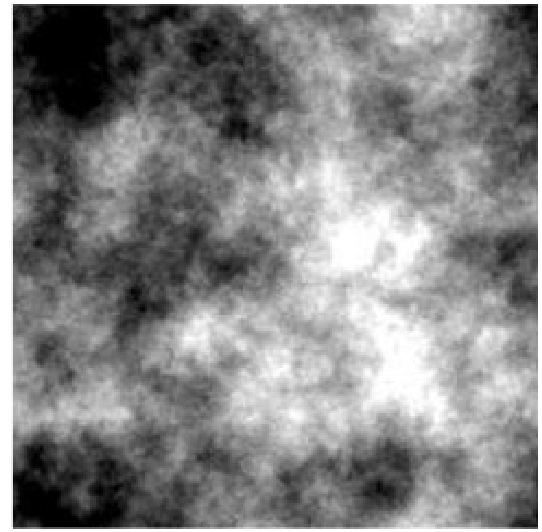
Sampling High-Resolution Images

gMRF: **3 layer**

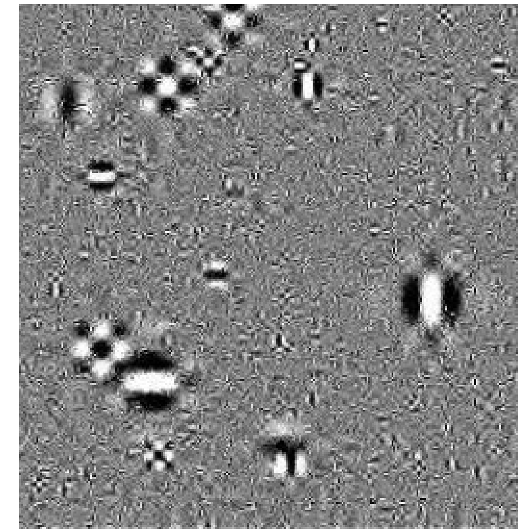


Ranzato et al. PAMI 2013

Gaussian model

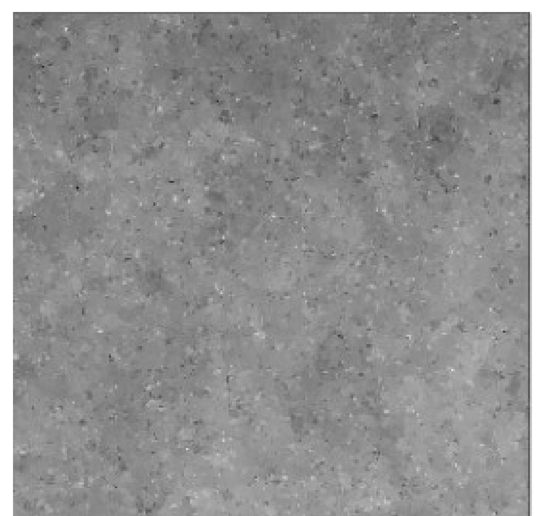


marginal wavelet

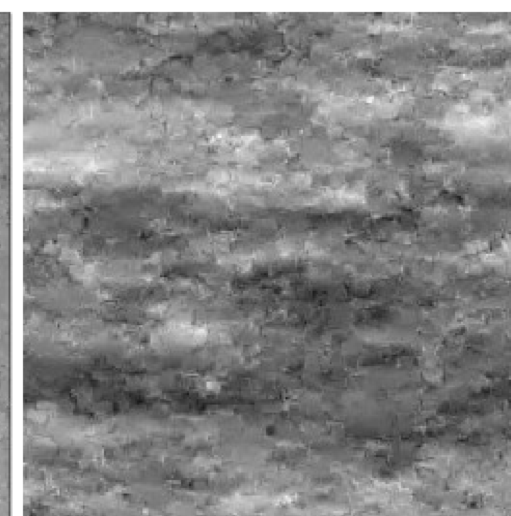


from Simoncelli 2005

Pair-wise MRF



FoE



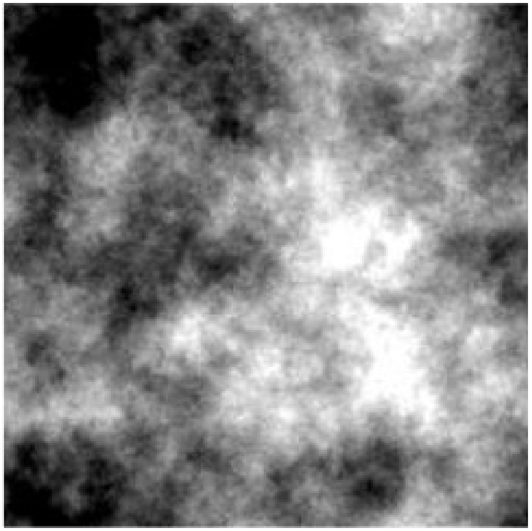
from Schmidt, Gao, Roth CVPR 2010

Sampling High-Resolution Images

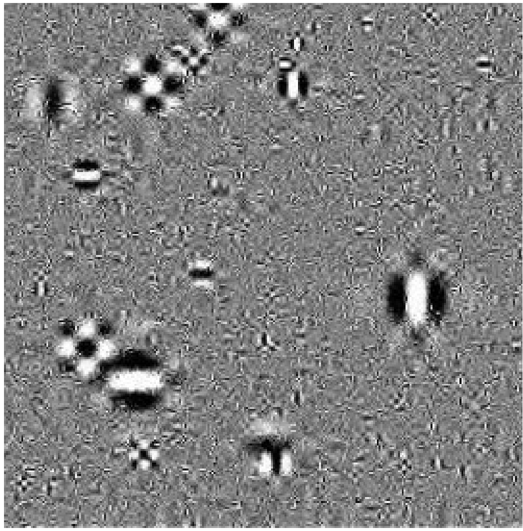
gMRF: **3 layer**

Ranzato et al. PAMI 2013

Gaussian model

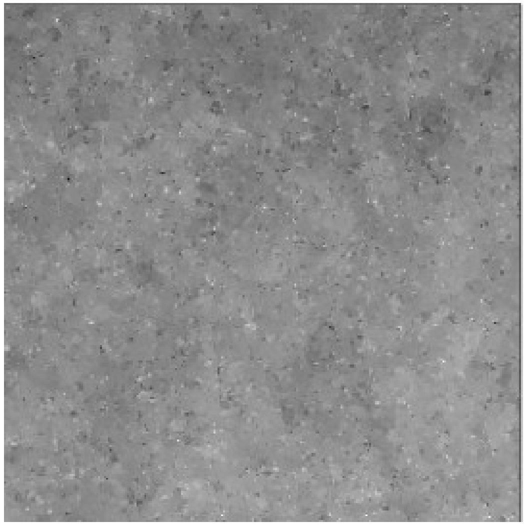


marginal wavelet



from Simoncelli 2005

Pair-wise MRF

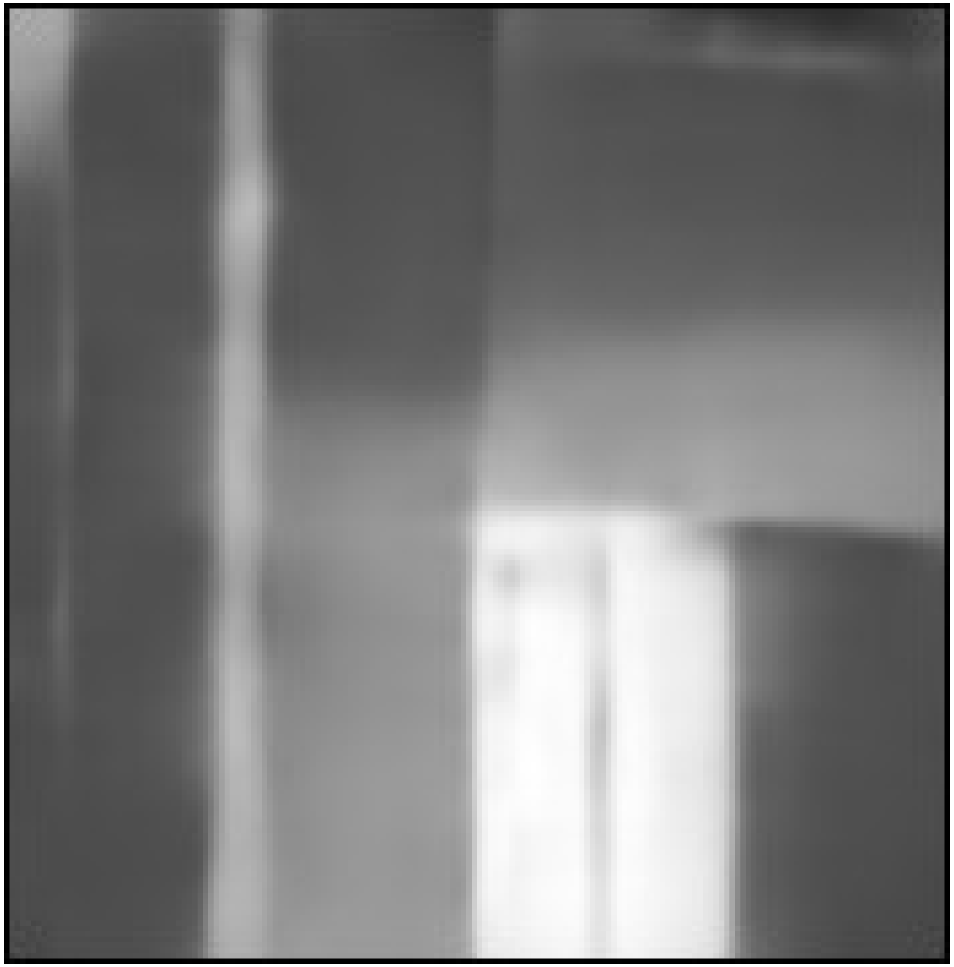


FoE

from Schmidt, Gao, Roth CVPR 2010

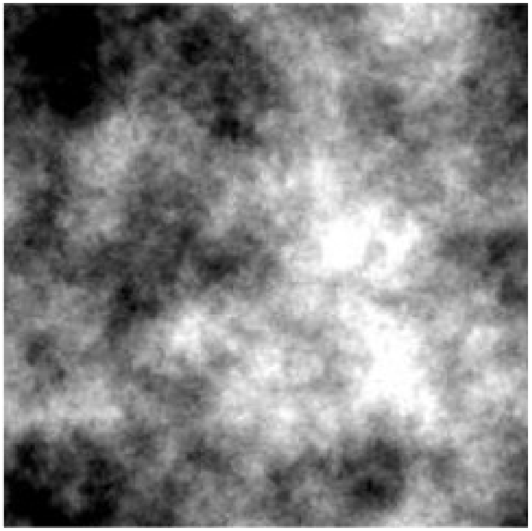
Sampling High-Resolution Images

gMRF: **3 layer**

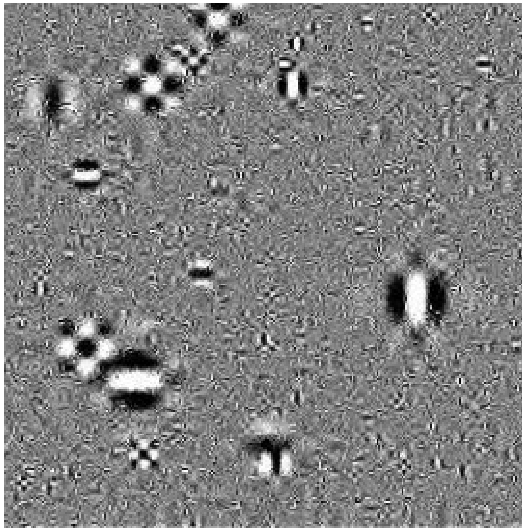


Ranzato et al. PAMI 2013

Gaussian model

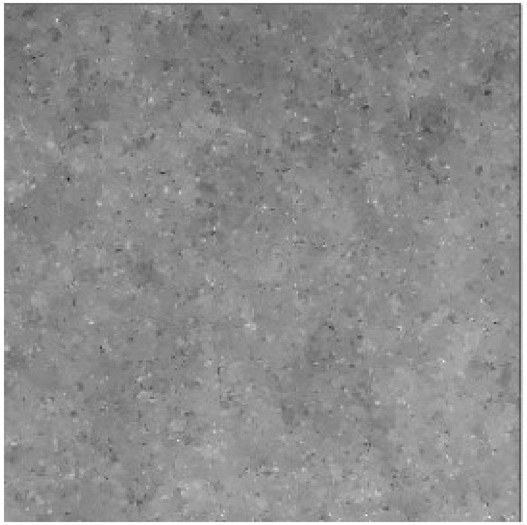


marginal wavelet



from Simoncelli 2005

Pair-wise MRF



FoE

from Schmidt, Gao, Roth CVPR 2010

Sampling After Training on Face Images

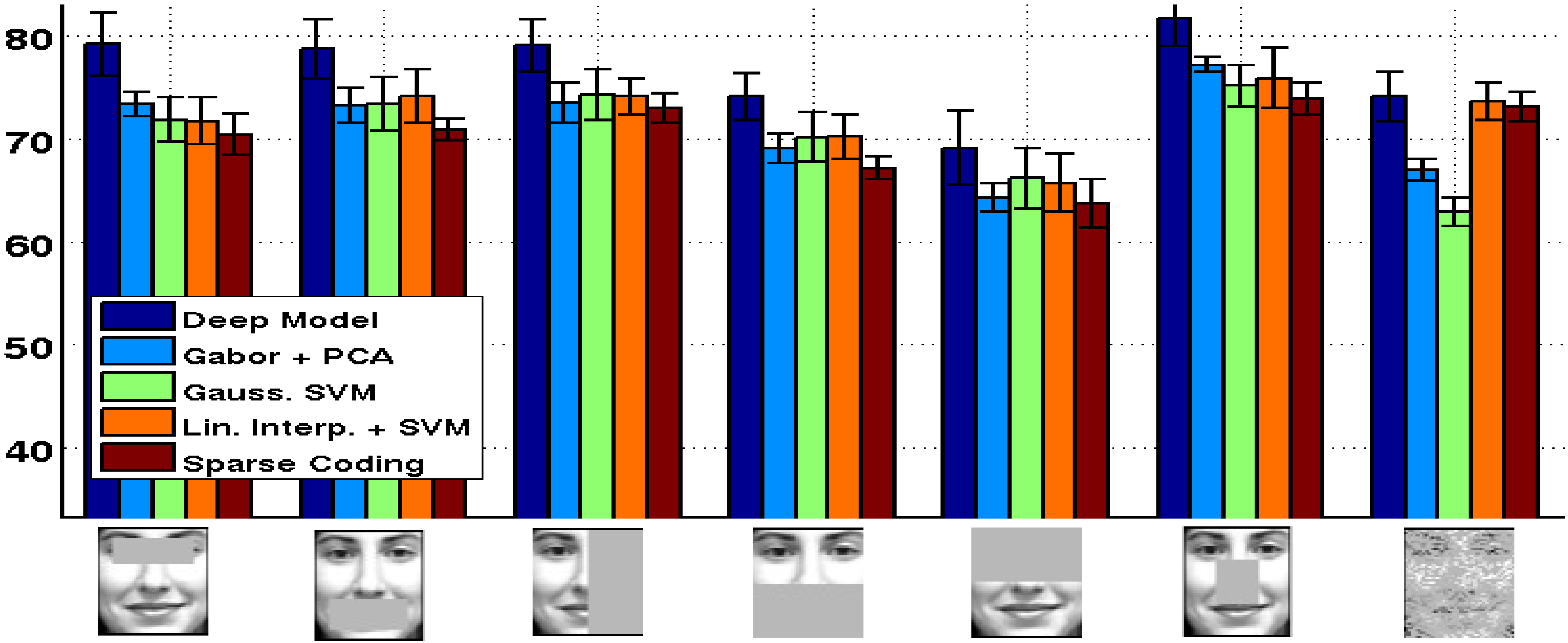
unconstrained samples

Original | Input | 1st layer | 2nd layer | 3rd layer | 4th layer | 10 times

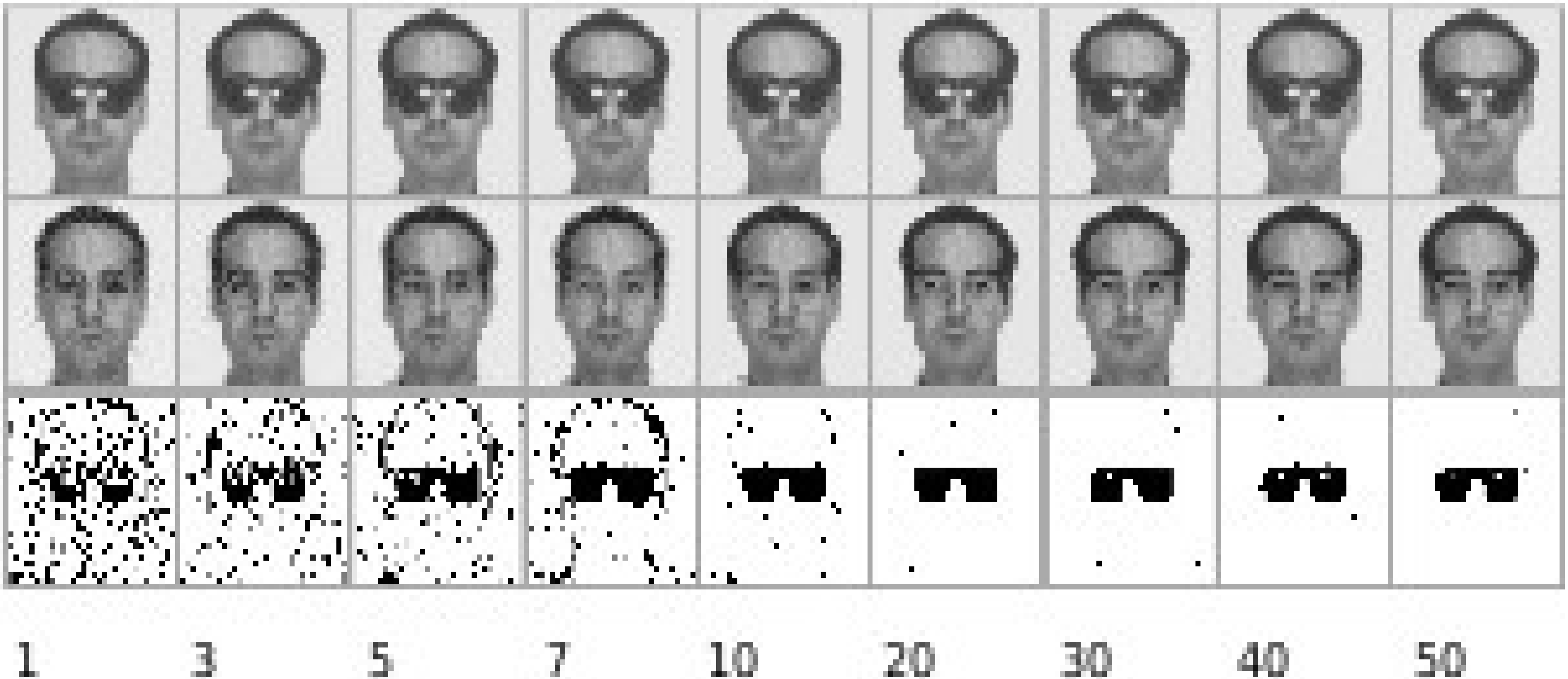
conditional (on the left part of the face) samples

Ranzato et al. PAMI 2013

Expression Recognition Under Occlusion



Ranzato et al. PAMI 2013



Tang et al. Robust BM for decognition and denoising CVPR 2012

Pros

- Feature extraction is fast
- Unprecedented generation quality
- Advances models of natural images
- Trains without labeled data

Cons

- Training is inefficient
  - Slow
  - Tricky
- Sampling scales badly with dimensionality
- What's the use case of generative models?

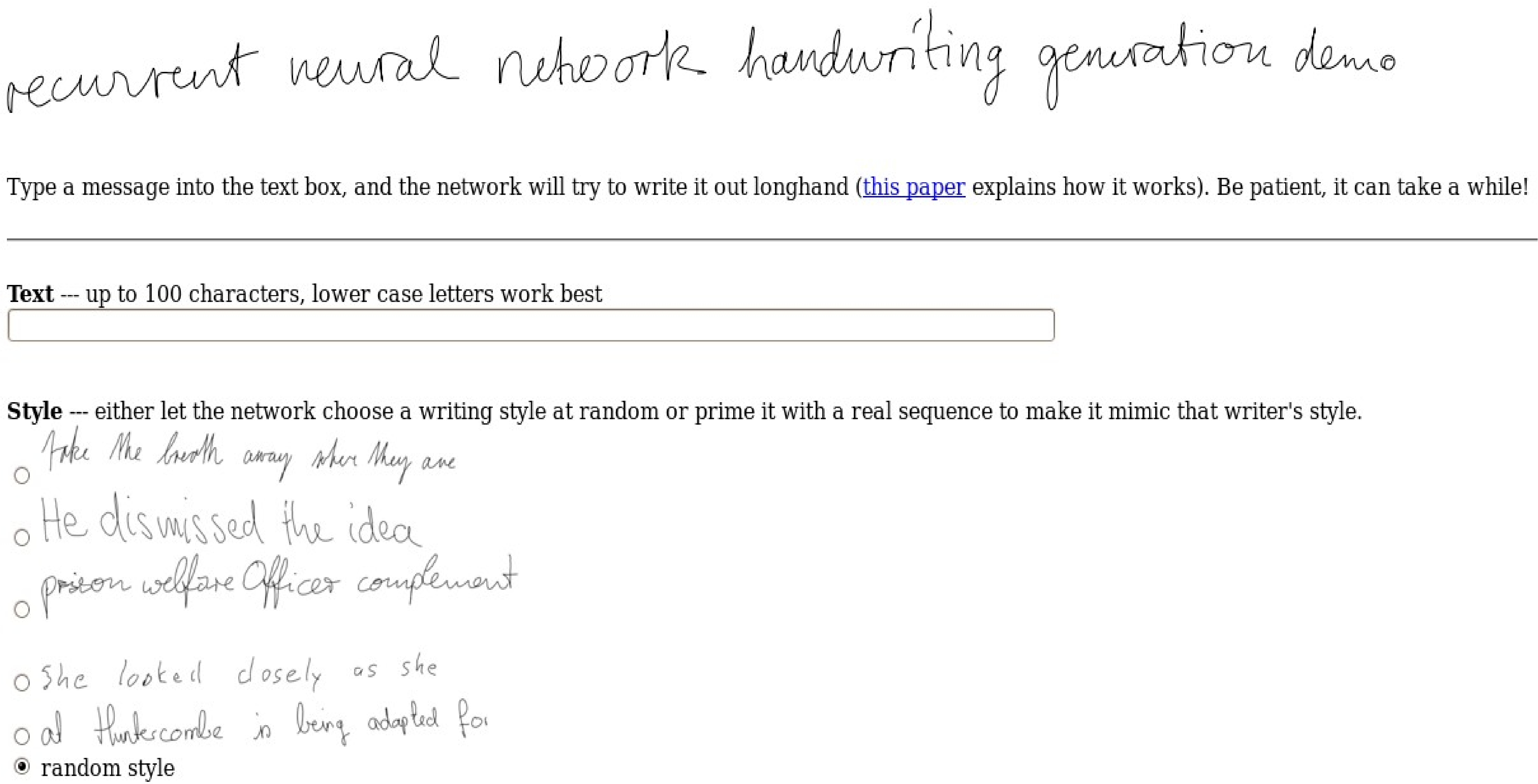
Conclusion

- If generation is not required, other feature learning methods are more efficient (e.g., sparse auto-encoders).
- What's the use case of generative models?
- Given enough labeled data, unsup. learning methods have not produced more useful features.

Outline

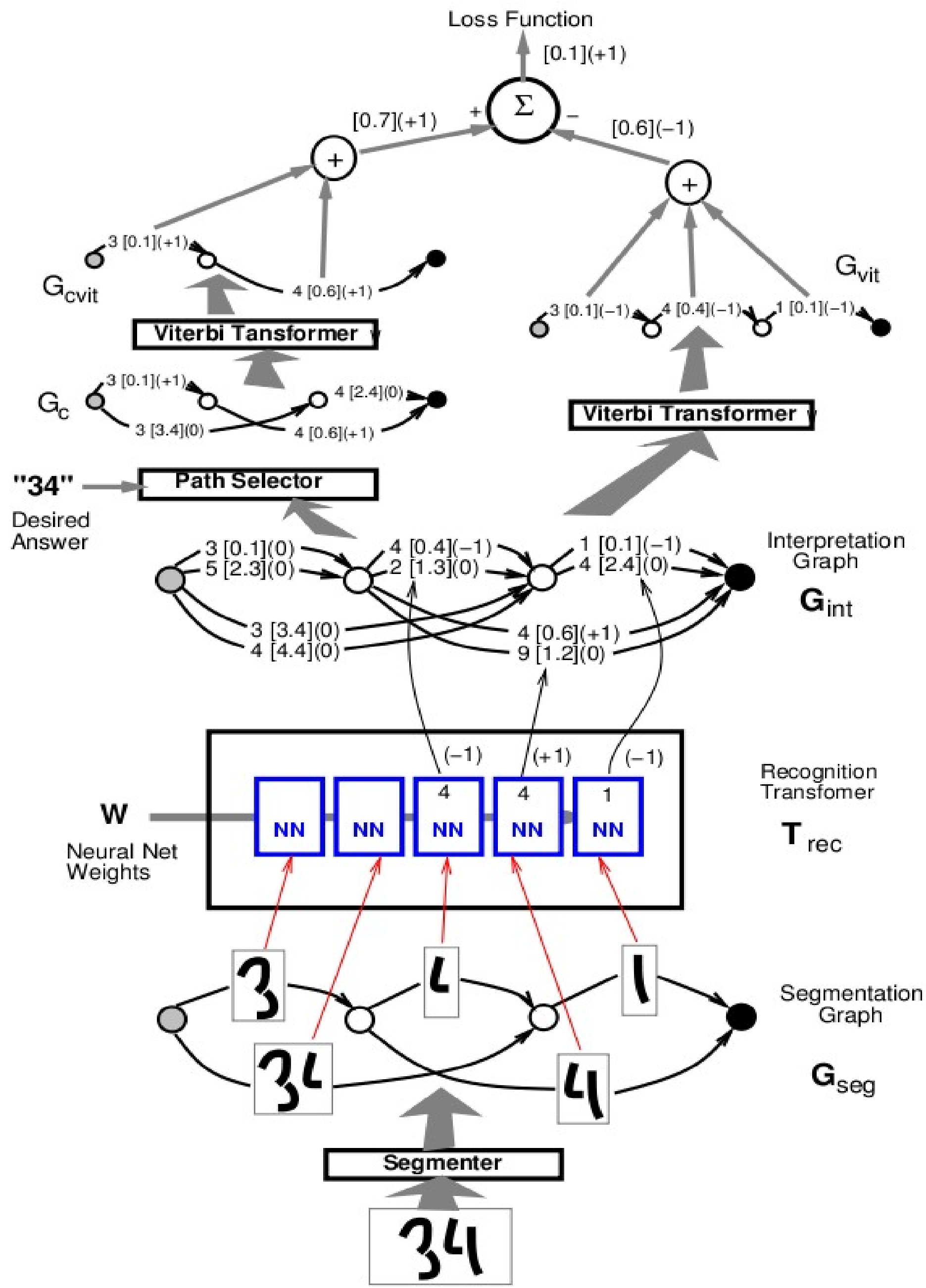
- Theory: Energy-Based Models
  - Energy function
  - Loss function
- Examples:
    - Supervised learning: neural nets
    - Supervised learning: convnets
    - Unsupervised learning: sparse coding
    - Unsupervised learning: gated MRF
- Other examples
- Practical tricks

RNNs



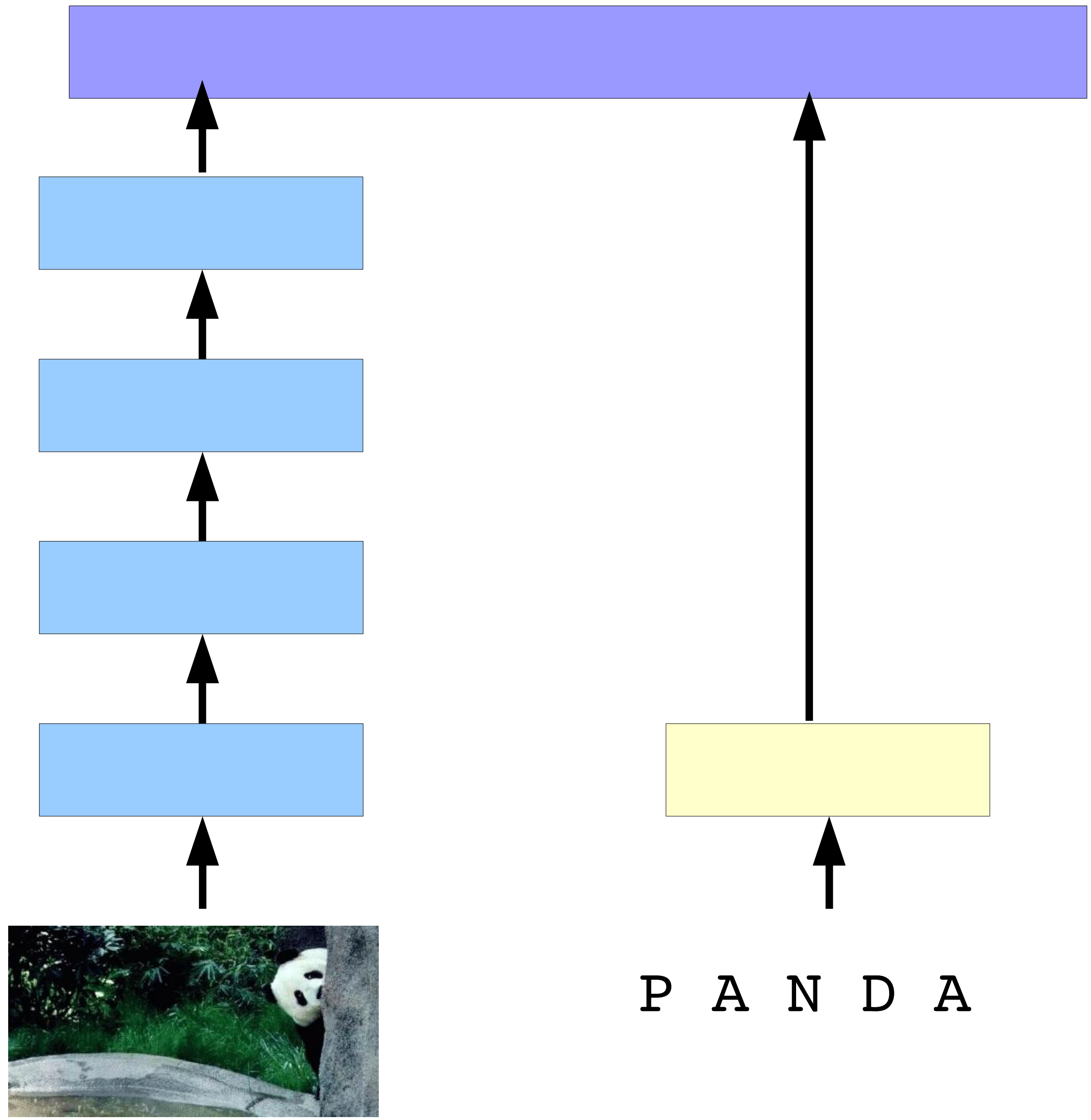
http://www.cs.toronto.edu/~graves/handwriting.html

Structured Prediction



LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

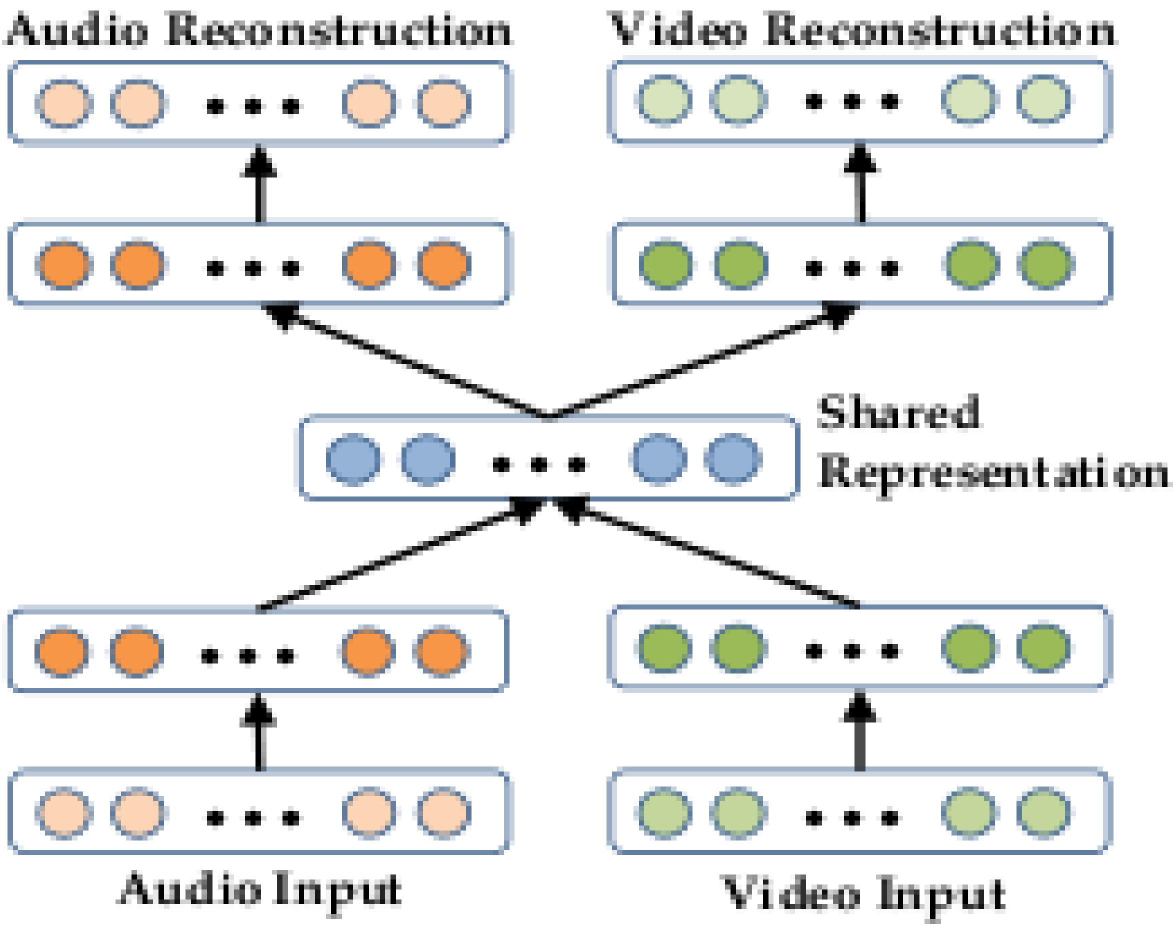
Multi-Modal Learning



Frome et al. “DeVISE: A deep visual semantic embedding model” NIPS 2013

Socher et al. Zero-shot learning though cross modal transfer” NIPS 2013

Multi-Modal Learning



(b) Bimodal Deep Autoencoder

Ngiam et al. "Multimodal deep learningl" ICML 2011

Srivastava et al. "Multi-modal learning with DBM" ICML 2012

Outline

- Theory: Energy-Based Models
  - Energy function
  - Loss function
- Examples:
    - Supervised learning: neural nets
    - Supervised learning: convnets
    - Unsupervised learning: sparse coding
    - Unsupervised learning: gated MRF
- Other examples
- Practical tricks for CNNs

CHOOSING THE ARCHITECTURE

- Task dependent

- Cross-validation

- [Convolution $\rightarrow$ LCN $\rightarrow$ pooling]* + fully connected layer

- The more data: the more layers and the more kernels
  - Look at the number of parameters at each layer
  - Look at the number of flops at each layer

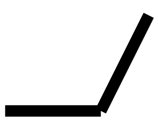
- Computational cost

- Be creative :)

HOW TO OPTIMIZE

- SGD (with momentum) usually works very well

- Pick learning rate by running on a subset of the data
  Bottou "Stochastic Gradient Tricks" Neural Networks 2012
  - Start with large learning rate and divide by 2 until loss does not diverge
  - Decay learning rate by a factor of ~1000 or more by the end of training

- Use  non-linearity

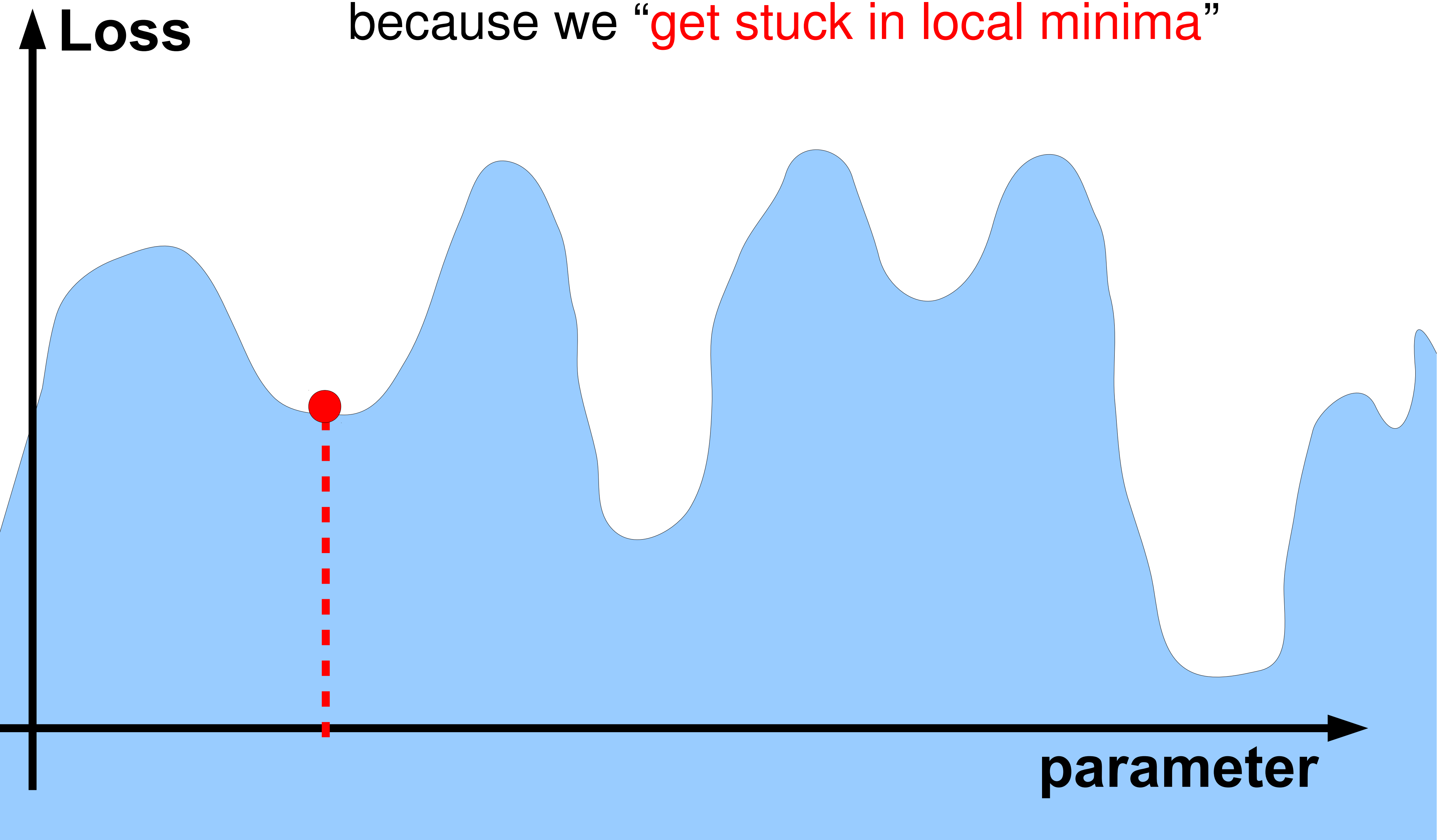
- Initialize parameters so that each feature across layers has similar variance. Avoid units in saturation.

HOW TO IMPROVE GENERALIZATION

- Weight sharing (greatly reduce the number of parameters)
- Data augmentation (e.g., jittering, noise injection, etc.)
- Dropout
  Hinton et al. “Improving Nns by preventing co-adaptation of feature detectors” arxiv 2012
- Weight decay (L2, L1)
- Sparsity in the hidden units
- Multi-task (unsupervised learning)

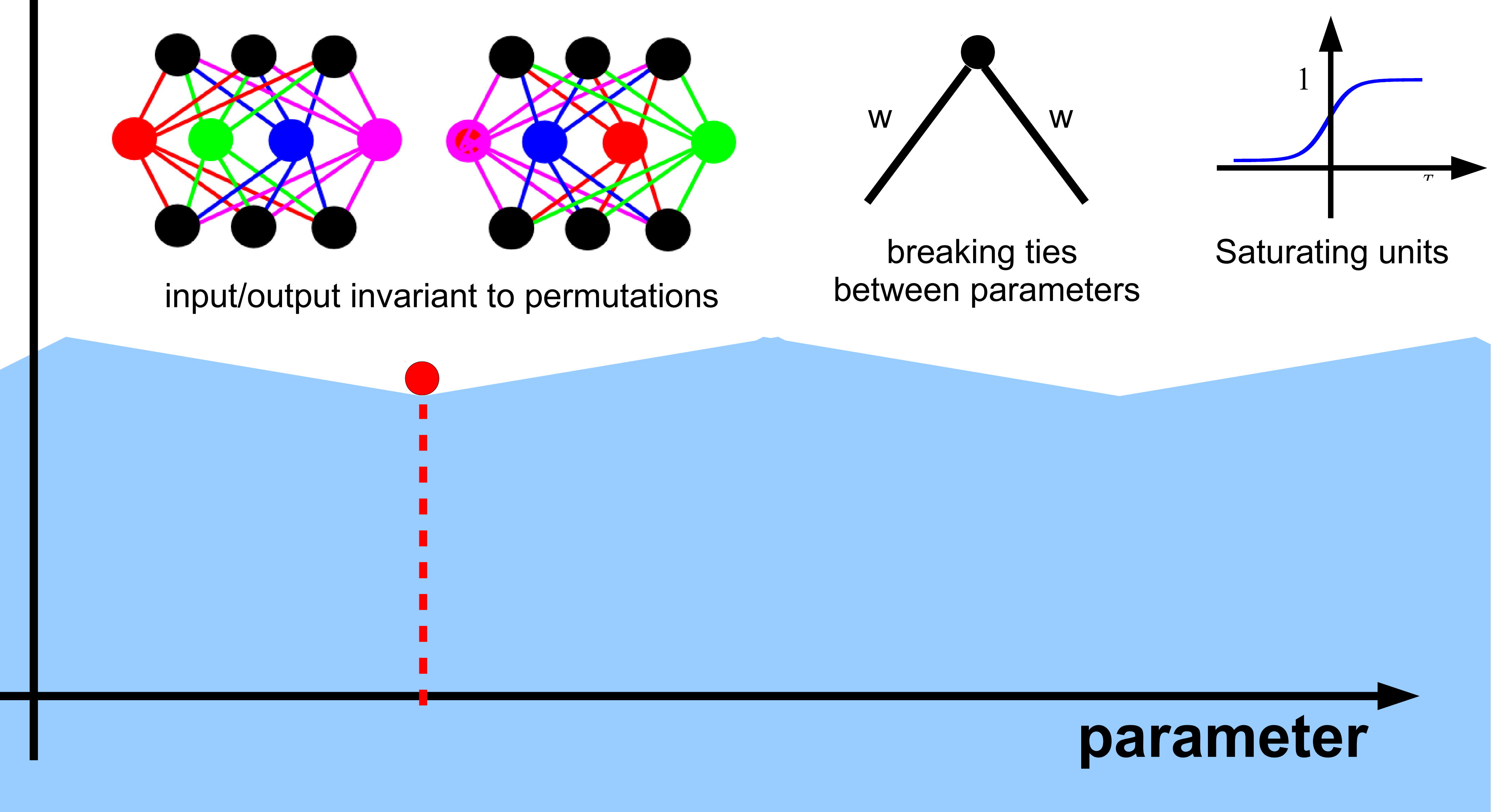
ConvNets: till 2012

Common wisdom: training does not work because we "get stuck in local minima"



ConvNets: today

Local minima are all similar, there are long plateaus, it can take long time to break symmetries.



Neural Net Optimization is...

ConvNets: today

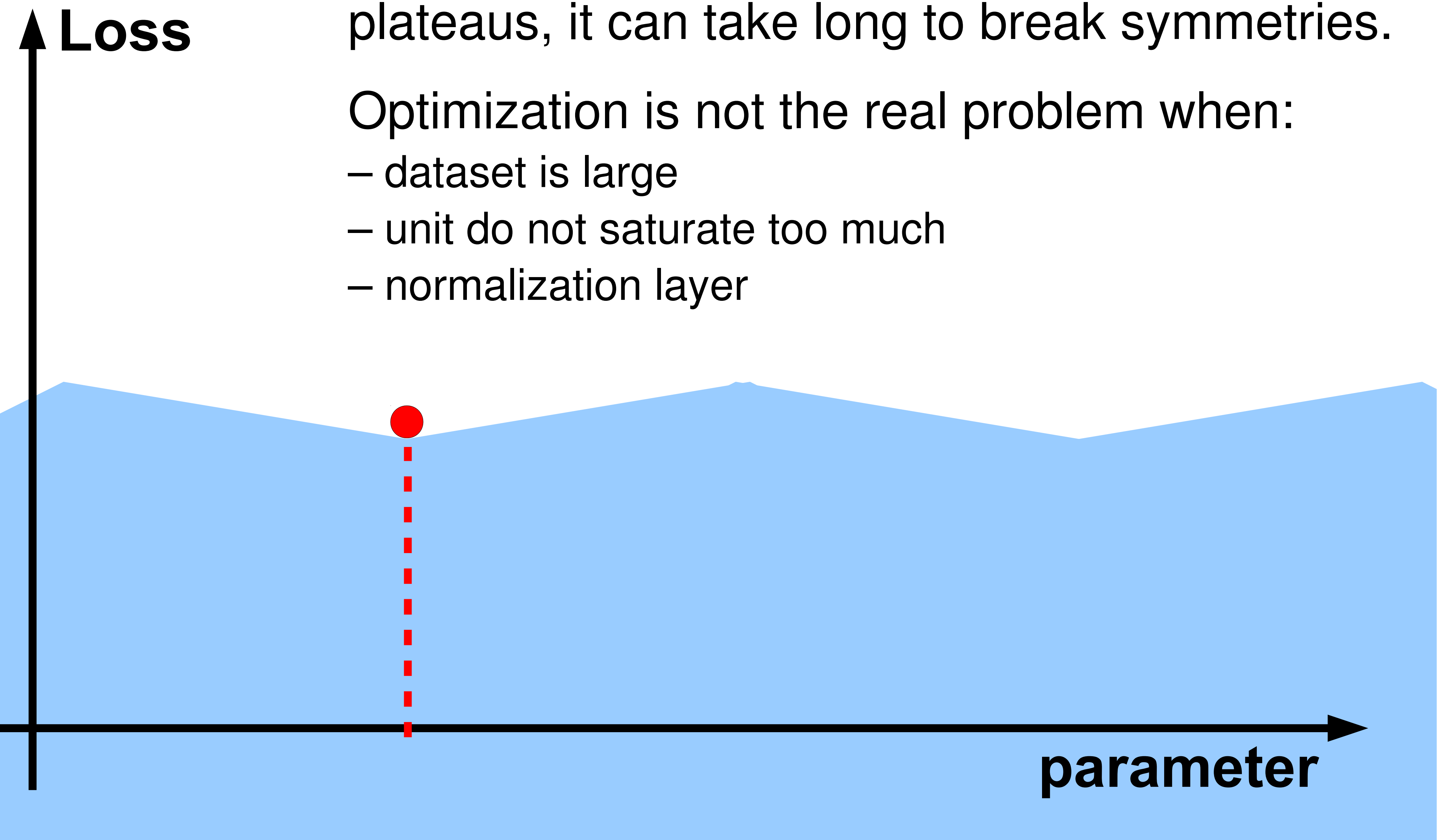
Local minima are all similar, there are long plateaus, it can take long to break symmetries.

Optimization is not the real problem when:

– dataset is large

– unit do not saturate too much

– normalization layer



ConvNets: today

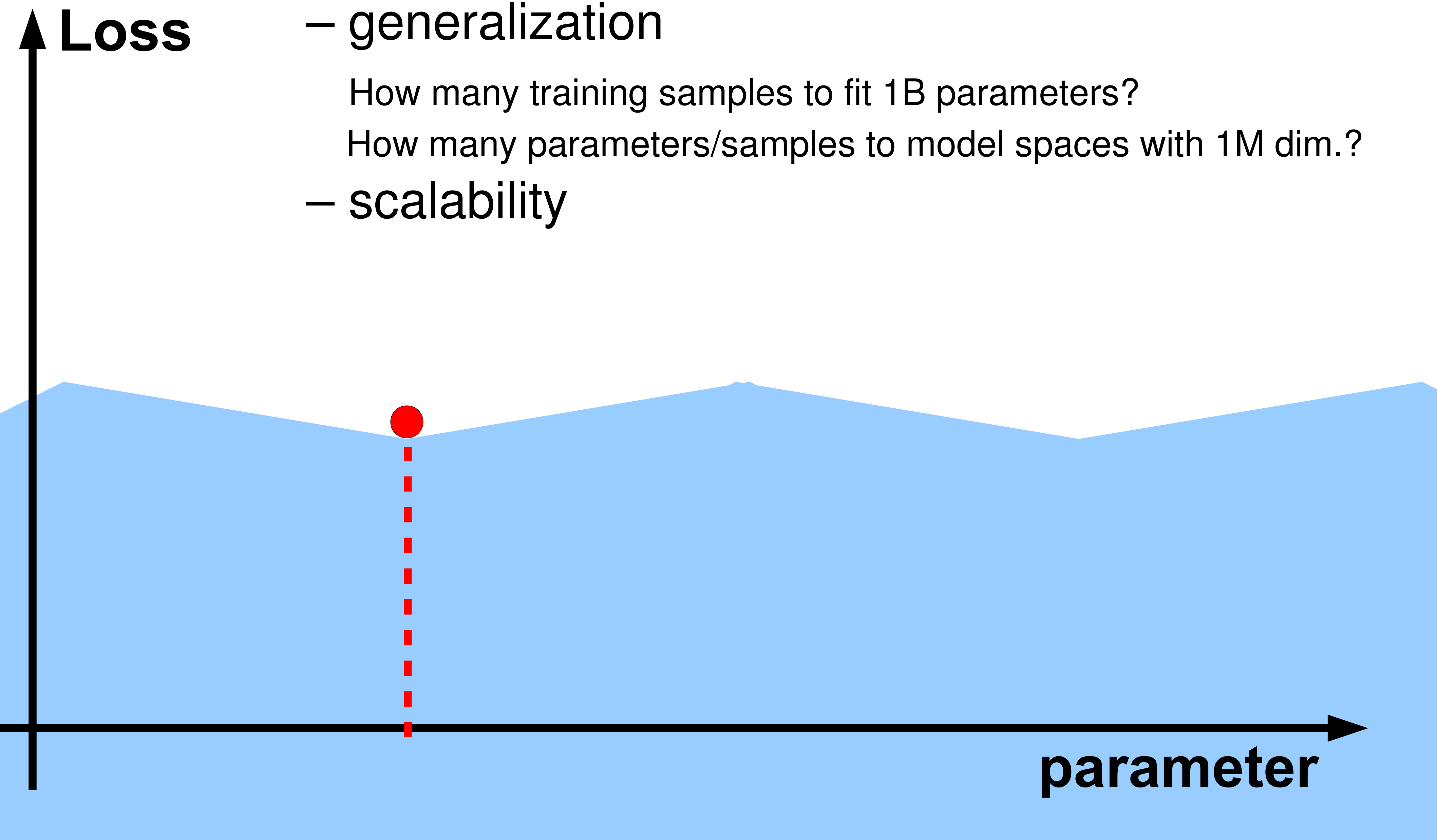
Today's belief is that the challenge is about:

– generalization

How many training samples to fit 1B parameters?

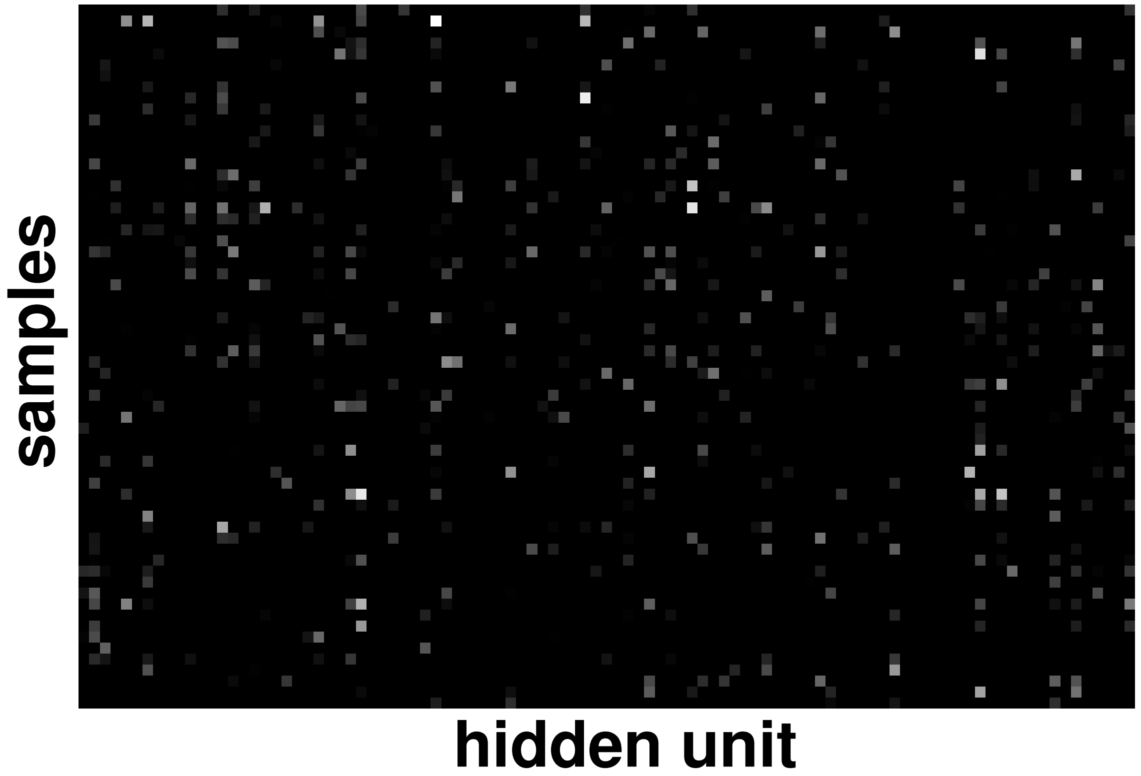
How many parameters/samples to model spaces with 1M dim.?

– scalability



OTHER THINGS GOOD TO KNOW

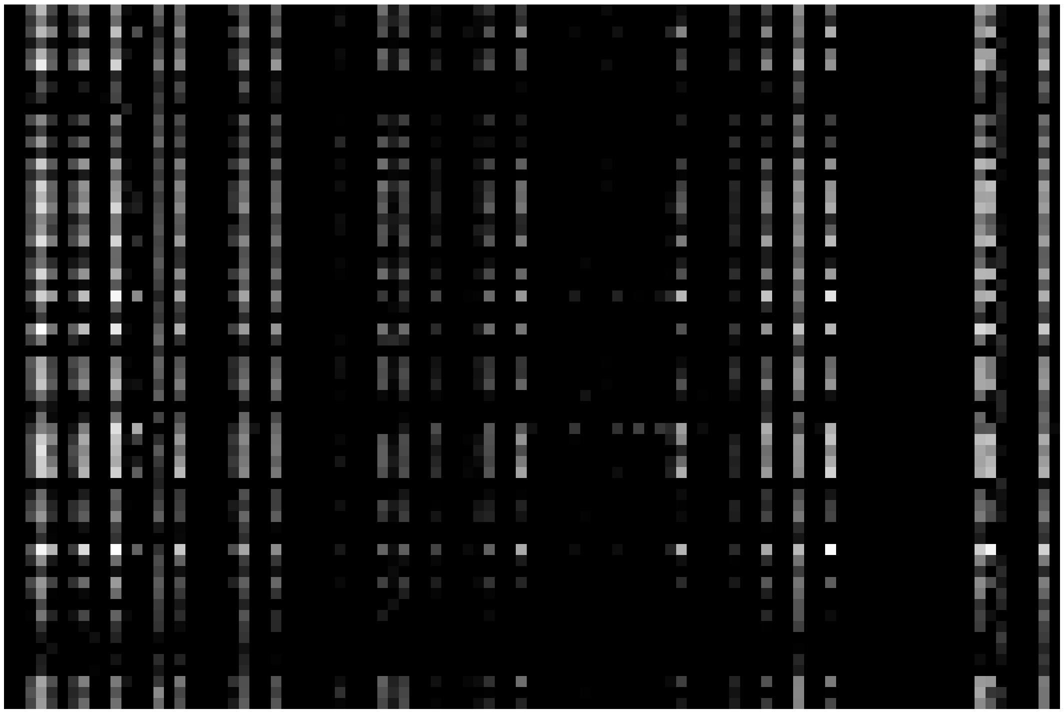
- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.



Good training: hidden units are sparse across samples and across features.

OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.

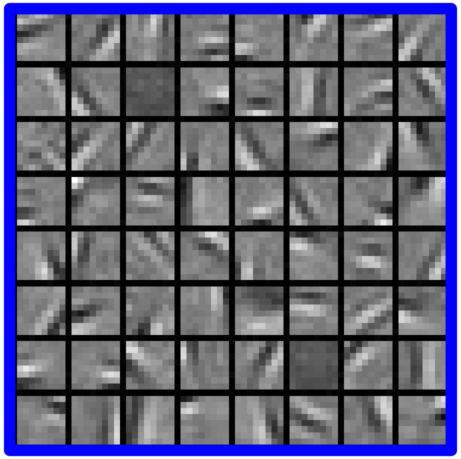


Bad training: many hidden units ignore the input and/or exhibit strong correlations.

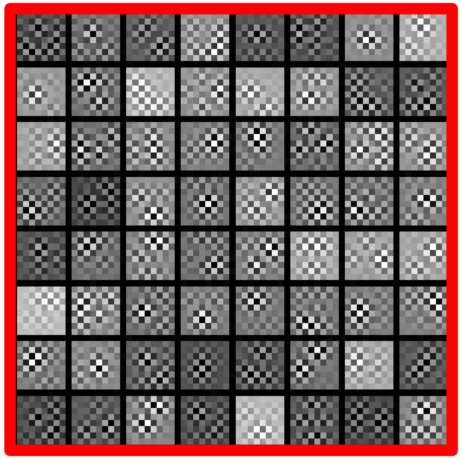
OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters

GOOD

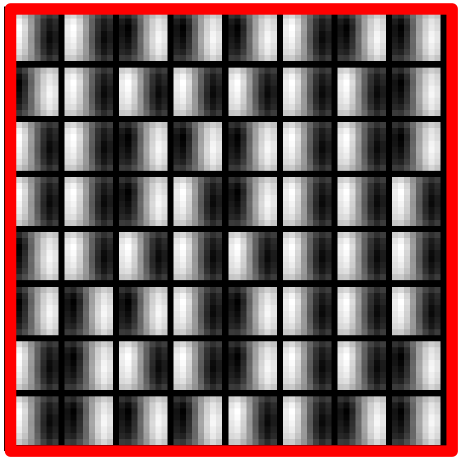


BAD



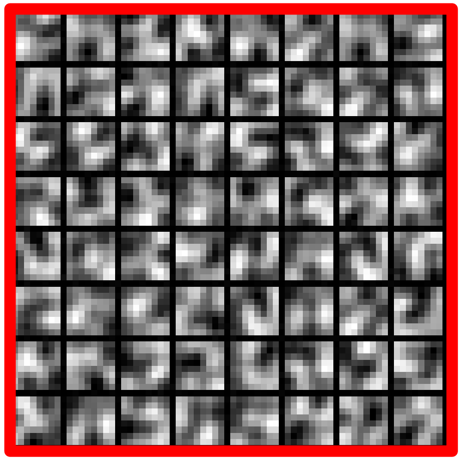
too noisy

BAD



too correlated

BAD



lack structure

Good training: learned filters exhibit structure and are uncorrelated.

OTHER THINGS GOOD TO KNOW

- Check gradients numerically by finite differences
- Visualize features (feature maps need to be uncorrelated) and have high variance.
- Visualize parameters
- Measure error on both training and validation set.
- Test on a small subset of the data and check the error $\rightarrow 0$.

WHAT IF IT DOES NOT WORK?

- Training diverges:
  - Learning rate may be too large $\rightarrow$ decrease learning rate
  - BPROP is buggy $\rightarrow$ numerical gradient checking
- Parameters collapse / loss is minimized but accuracy is low
  - Check loss function:
    - Is it appropriate for the task you want to solve?
    - Does it have degenerate solutions? Check "pull-up" term.
- Network is underperforming
  - Compute flops and nr. params. $\rightarrow$ if too small, make net larger
  - Visualize hidden units/params $\rightarrow$ fix optmization
- Network is too slow
  - Compute flops and nr. params. $\rightarrow$ GPU,distrib. framework, make net smaller

SUMMARY

- Deep Learning = Learning Hierarchical representations. Leverage compositionality to gain efficiency.
- Unsupervised learning: active research topic.
- Supervised learning: most successful set up today.
- Optimization
  - Don't we get stuck in local minima? No, they are all the same!
  - In large scale applications, local minima are even less of an issue.
- Scaling
  - GPUs
  - Distributed framework (Google)
  - Better optimization techniques
- Generalization on small datasets (curse of dimensionality):
  - Input distortions
  - weight decay
  - dropout

THANK YOU!

NOTE: IJCV Special Issue on Deep Learning.
Deadline: 9 Feb. 2014.

SOFTWARE

Torch7: learning library that supports neural net training

http://www.torch.ch

http://code.cogbits.com/wiki/doku.php (tutorial with demos by C. Farabet)

Python-based learning library (U. Montreal)

- http://deeplearning.net/software/theano/ (does automatic differentiation)

Efficient CUDA kernels for ConvNets (Krizhevsky)

– code.google.com/p/cuda-convnet

Caffe (Yangqing Jia)

– http://caffe.berkeleyvision.org

REFERENCES

Convolutional Nets

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

- Krizhevsky, Sutskever, Hinton "ImageNet Classification with deep convolutional neural networks" NIPS 2012

– Jarrett, Kavukcuoglu, Ranzato, LeCun: What is the Best Multi-Stage Architecture for Object Recognition?, Proc. International Conference on Computer Vision (ICCV'09), IEEE, 2009

- Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, LeCun: Learning Convolutional Feature Hierachies for Visual Recognition, Advances in Neural Information Processing Systems (NIPS 2010), 23, 2010

– see yann.lecun.com/exdb/publis for references on many different kinds of convnets.

– see http://www.cmap.polytechnique.fr/scattering/ for scattering networks (similar to convnets but with less learning and stronger mathematical foundations)

– see http://www.idsia.ch/~juergen/ for other references to ConvNets and LSTMs.

REFERENCES

Applications of Convolutional Nets

– Farabet, Couprie, Najman, LeCun. Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers", ICML 2012

– Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala and Yann LeCun: Pedestrian Detection with Unsupervised Multi-Stage Feature Learning, CVPR 2013

- D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. NIPS 2012

- Raia Hadsell, Pierre Sermanet, Marco Scoffier, Ayse Erkan, Koray Kavackuoglu, Urs Muller and Yann LeCun. Learning Long-Range Vision for Autonomous Off-Road Driving, Journal of Field Robotics, 26(2):120-144, 2009

– Burger, Schuler, Harmeling. Image Denoisng: Can Plain Neural Networks Compete with BM3D?, CVPR 2012

– Hadsell, Chopra, LeCun. Dimensionality reduction by learning an invariant mapping, CVPR 2006

– Bergstra et al. Making a science of model search: hyperparameter optimization in hundred of dimensions for vision architectures, ICML 2013

REFERENCES

Deep Learning in general

– deep learning tutorial slides at ICML 2013

– Yoshua Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), pp.1-127, 2009.

– LeCun, Chopra, Hadsell, Ranzato, Huang: A Tutorial on Energy-Based Learning, in Bakir, G. and Hofman, T. and Schölkopf, B. and Smola, A. and Taskar, B. (Eds), Predicting Structured Data, MIT Press, 2006

THANK YOU

Ackknowledgements to Yann LeCun. Many slides from ICML 2013 and CVPR 2013 tutorial on deep learning.

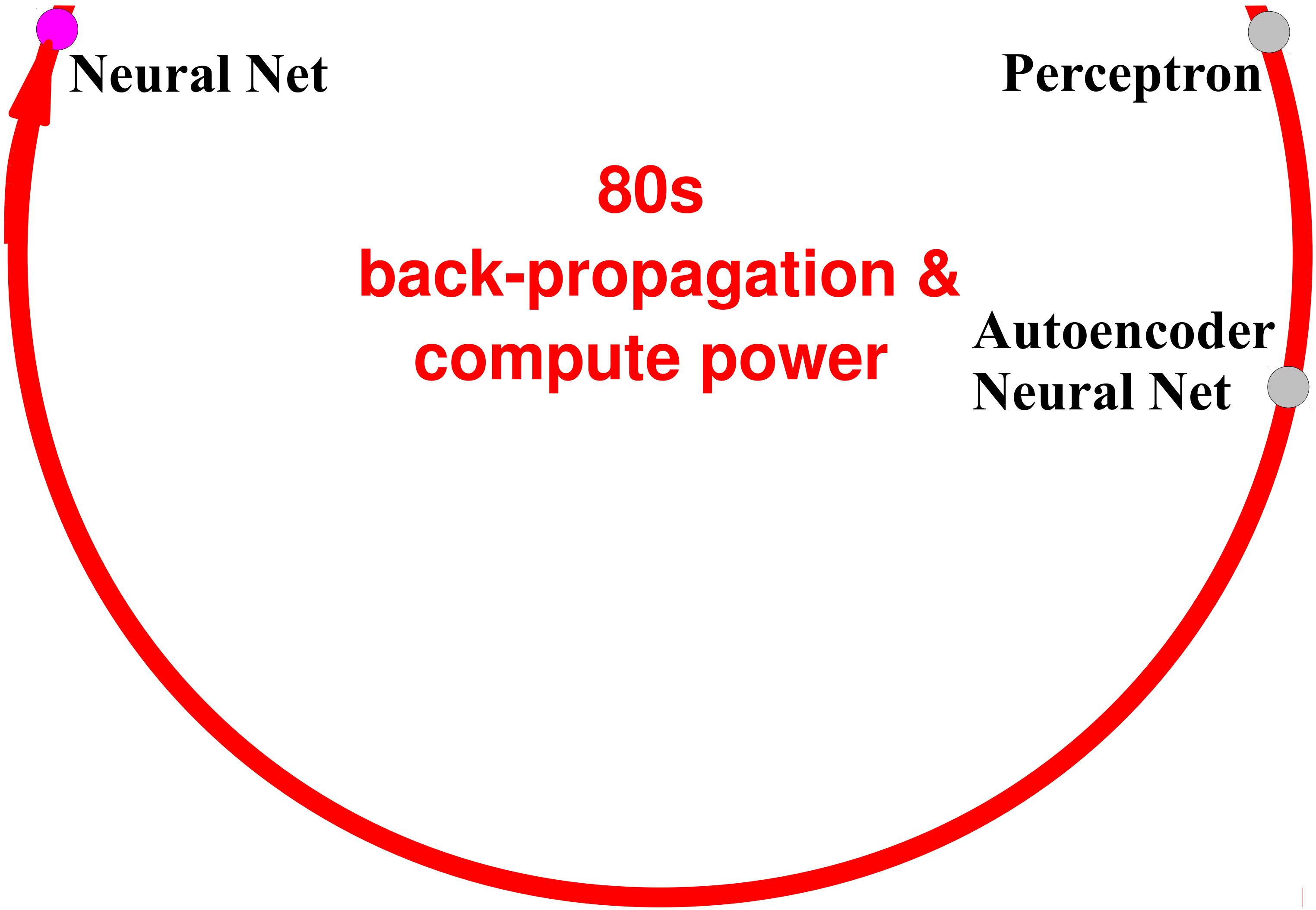
IJCV SPECIAL ISSUE ON DEEP LEARNING: 9 Febrruary 2014.

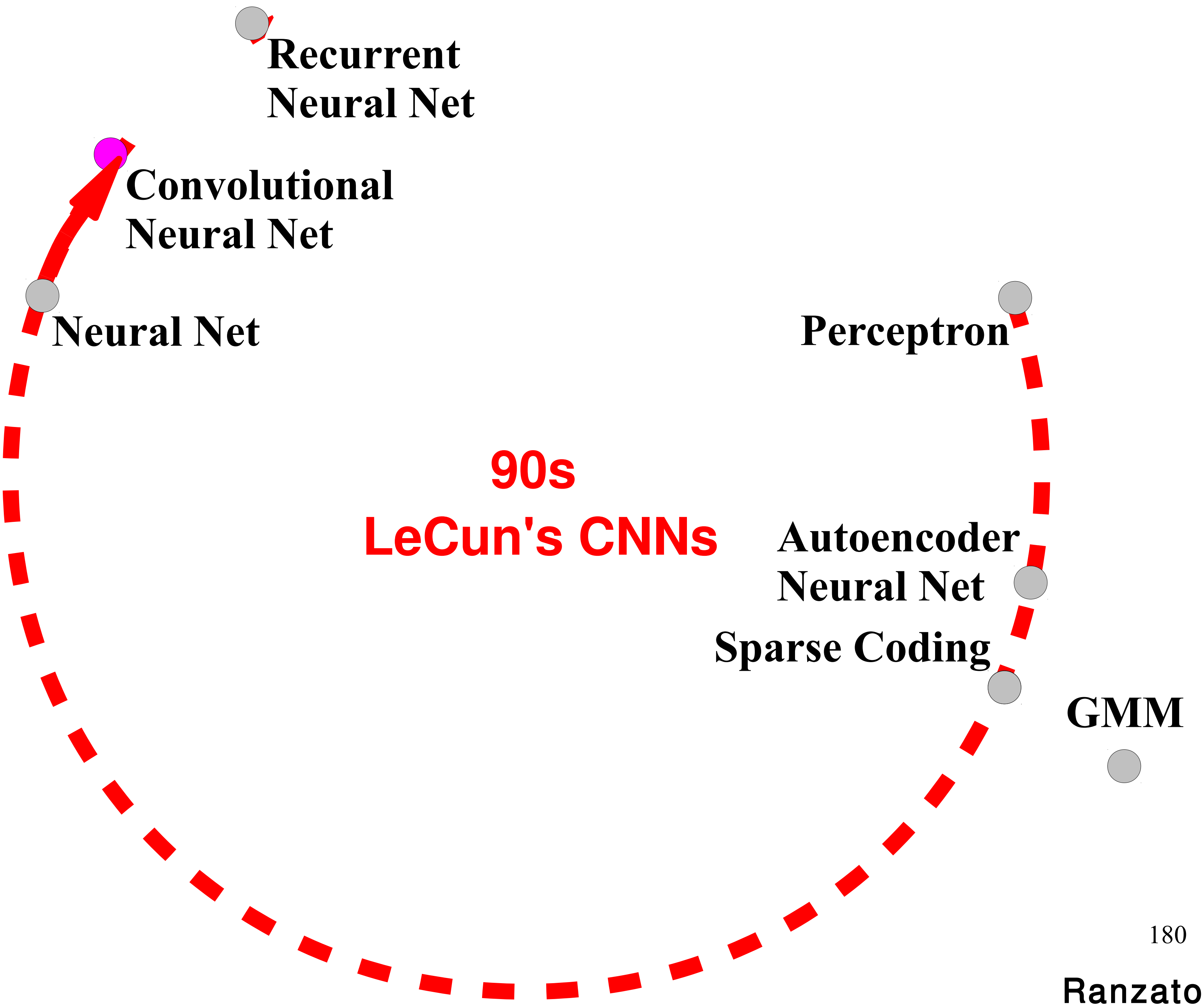
Perceptron

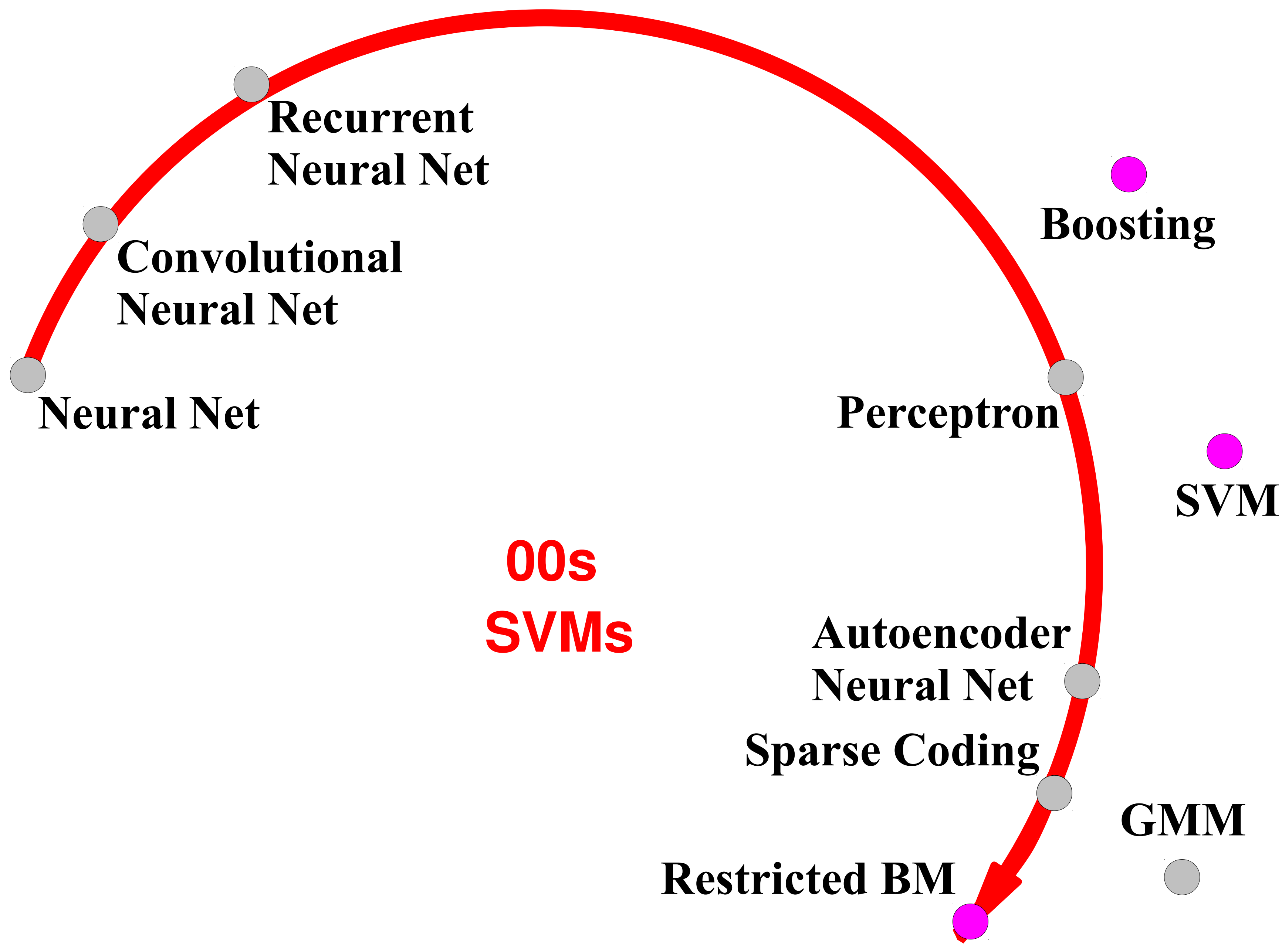
1957

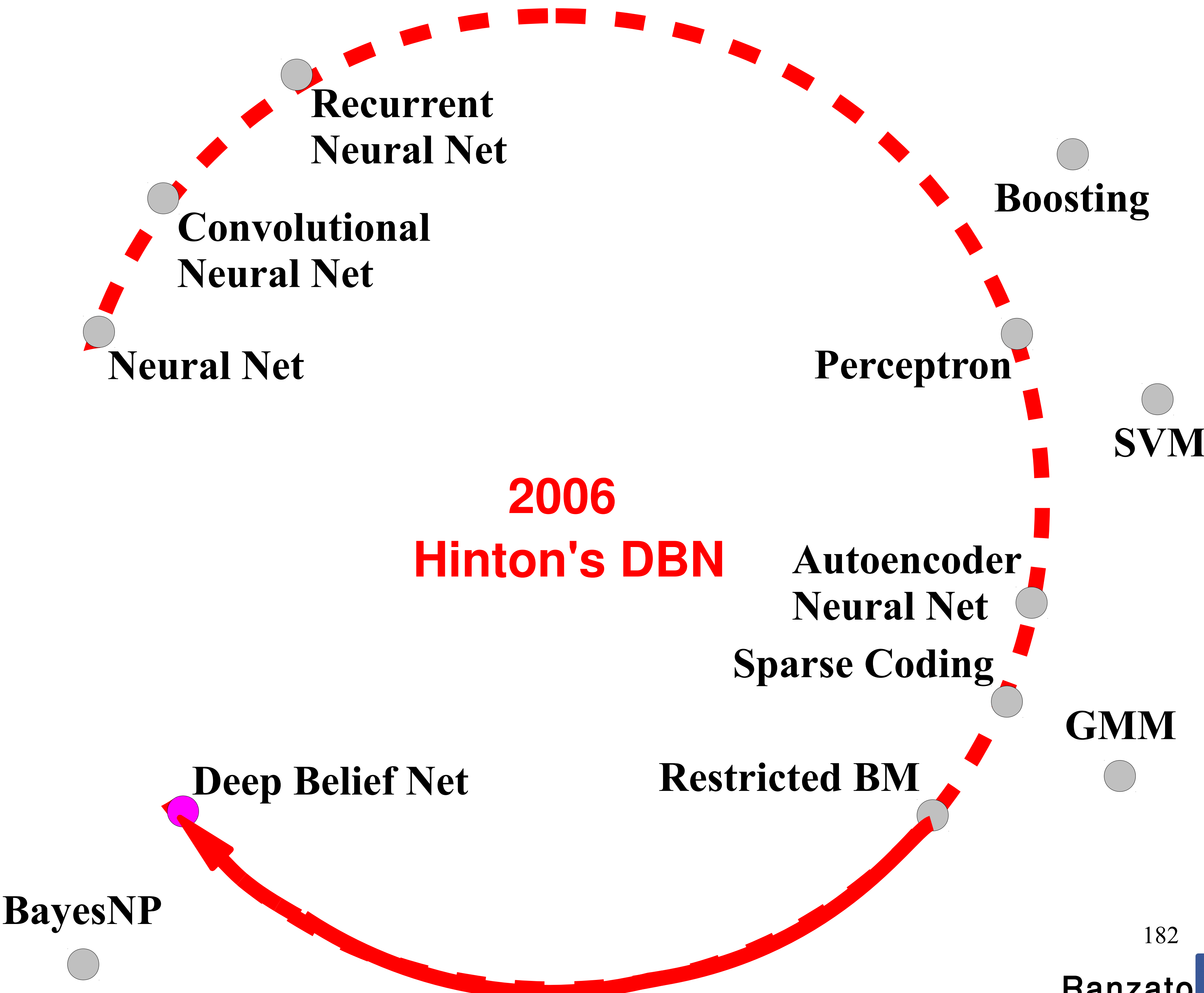
Rosenblatt

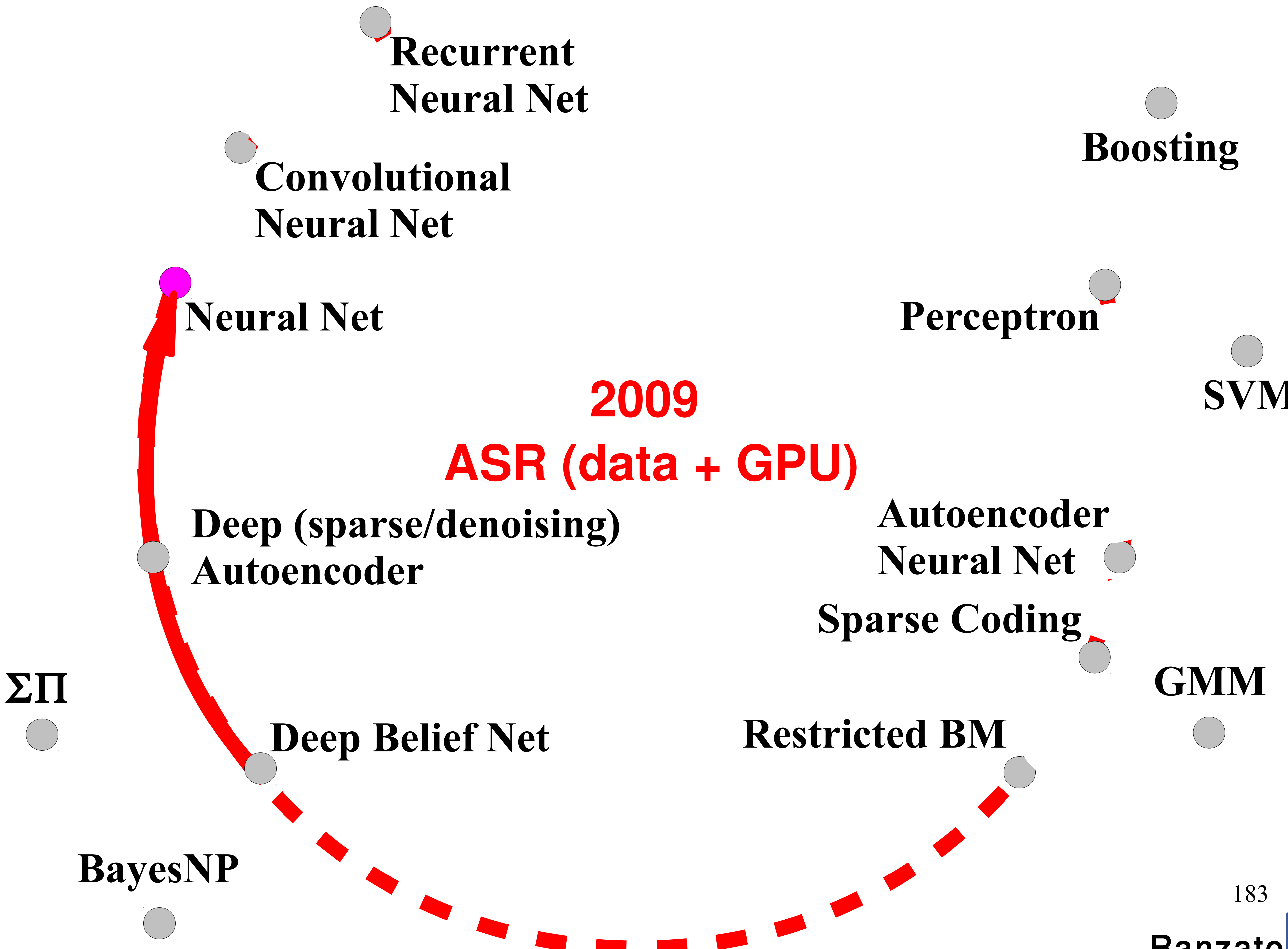
THE SPACE OF
MACHINE LEARNING METHODS


Neural Net
Perceptron
80s
back-propagation &
compute power
Autoencoder
Neural Net

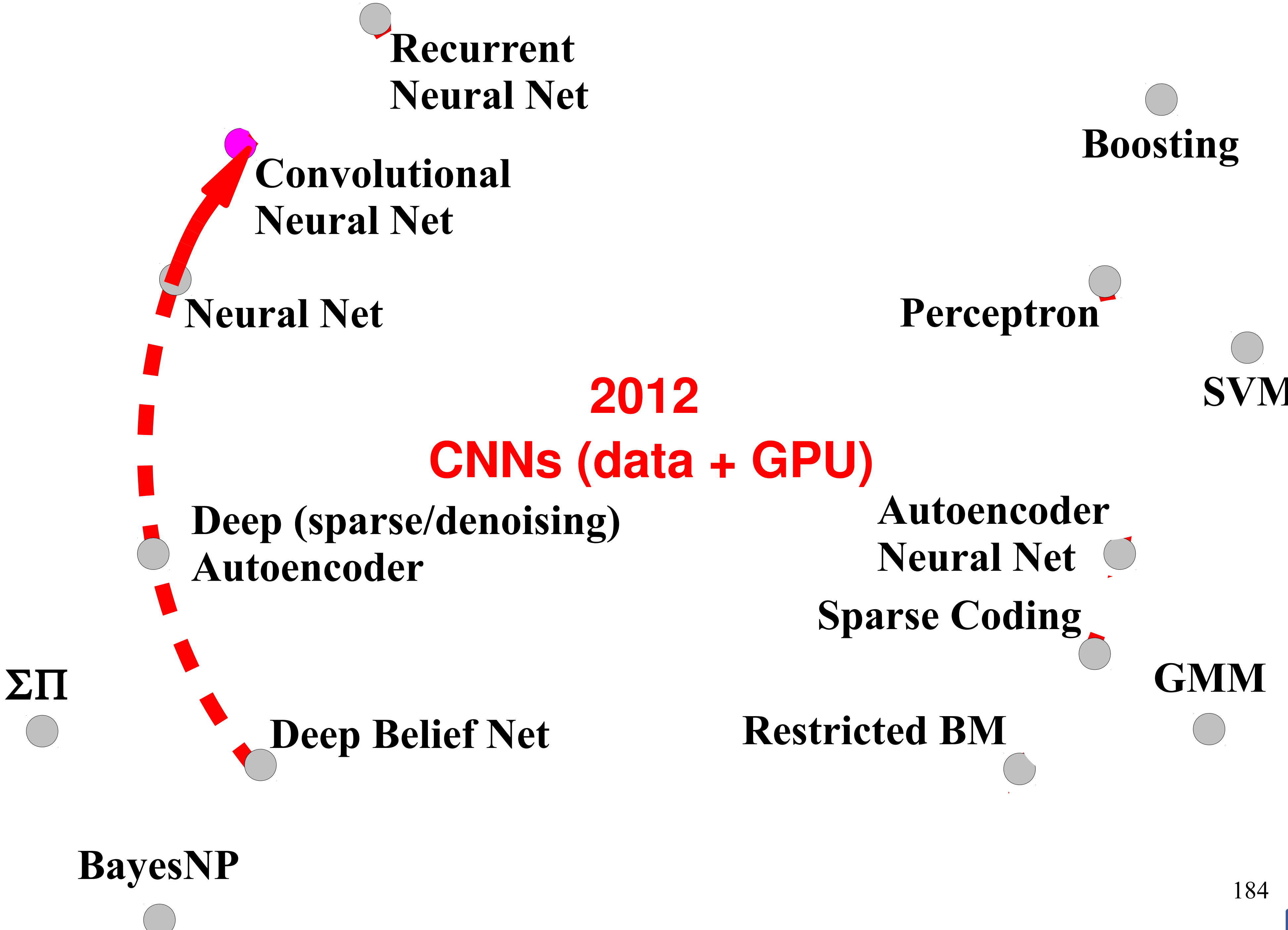

Recurrent
Neural Net
Convolutional
Neural Net
Neural Net
Perceptron
90s
LeCun's CNNs
Autoencoder
Neural Net
Sparse Coding
GMM


Recurrent
Neural Net
Convolutional
Neural Net
Neural Net
Boosting
Perceptron
SVM
00s
SVMs
Autoencoder
Neural Net
Sparse Coding
GMM
Restricted BM

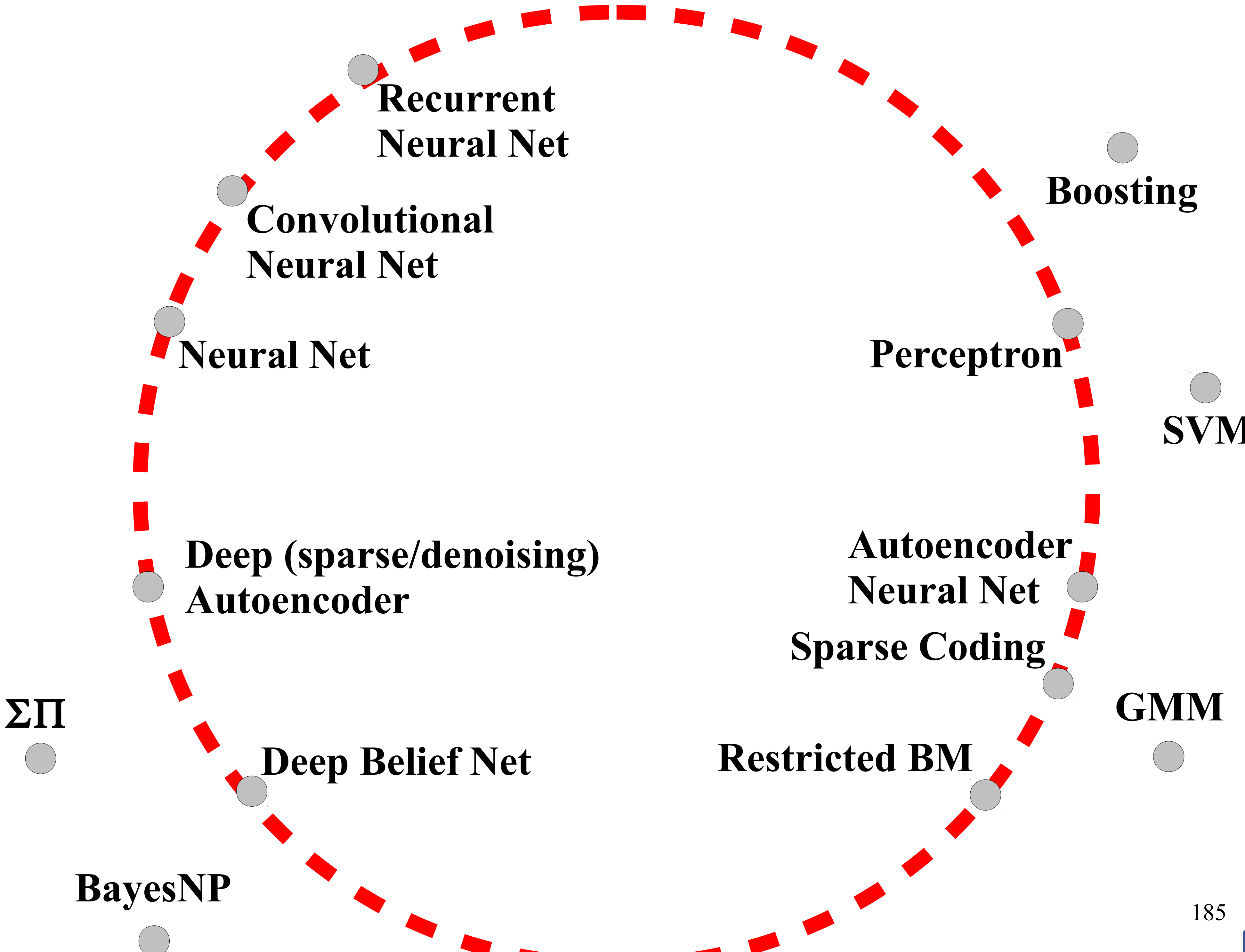

Recurrent
Neural Net
Convolutional
Neural Net
Neural Net
Boosting
Perceptron
SVM
2006
Hinton's DBN
Autoencoder
Neural Net
Sparse Coding
GMM
Deep Belief Net
Restricted BM
BayesNP

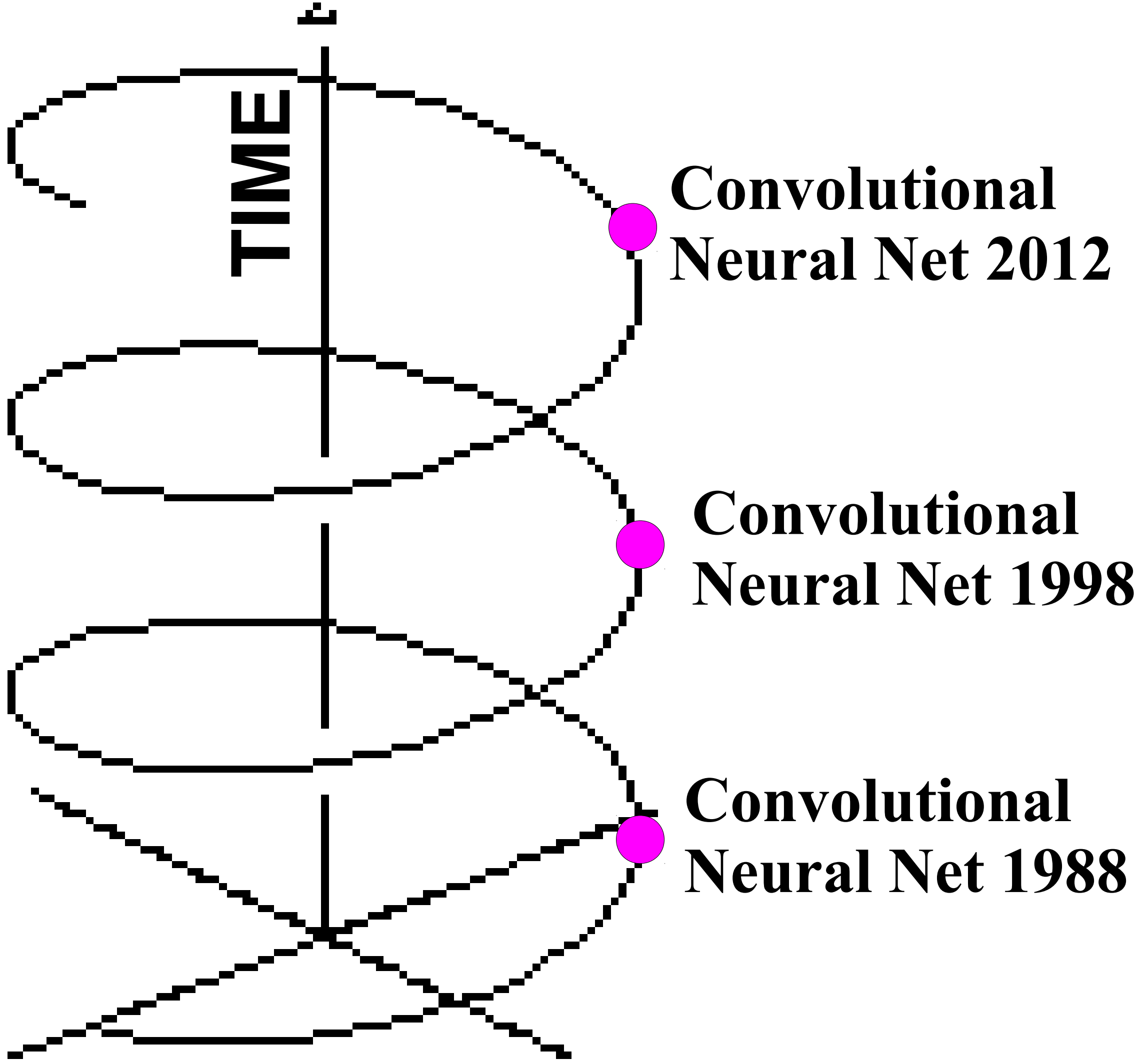


Recurrent Neural Net
Boosting
Convolutional Neural Net
Neural Net
Perceptron
SVM
2009
ASR (data + GPU)
Deep (sparse/denoising) Autoencoder
Autoencoder Neural Net
Sparse Coding
GMM
ΣΠ
Deep Belief Net
Restricted BM
BayesNP


Recurrent
Neural Net
Boosting
Convolutional
Neural Net
Neural Net
Perceptron
SVM
2012
CNNs (data + GPU)
Deep (sparse/denoising)
Autoencoder
Autoencoder
Neural Net
Sparse Coding
ΣΠ
GMM
Deep Belief Net
Restricted BM
BayesNP

Ranzato


Recurrent
Neural Net
Boosting
Convolutional
Neural Net
Neural Net
Perceptron
SVM
Deep (sparse/denoising)
Autoencoder
Autoencoder
Neural Net
Sparse Coding
ΣΠ
GMM
Deep Belief Net
Restricted BM
BayesNP



Q.: Did we make any prgress since then?

A.: The main reason for the breakthrough is: **data** and **GPU,** but we have also made networks **deeper** and more **non-linear**.

ConvNets: History

- 1980 **Fukushima**: designed network with same basic structure but did not train by backpropagation.

- late 80s **LeCun**: figured out backpropagation for CNN, popularized and deployed CNN for OCR applications and others.

- 1999 **Poggio**: same basic structure but learning is restricted to top layer (k-means at second stage)

- 2006 **LeCun:** unsupervised feature learning

- 2008 **DiCarlo:** large scale experiments, normalization layer

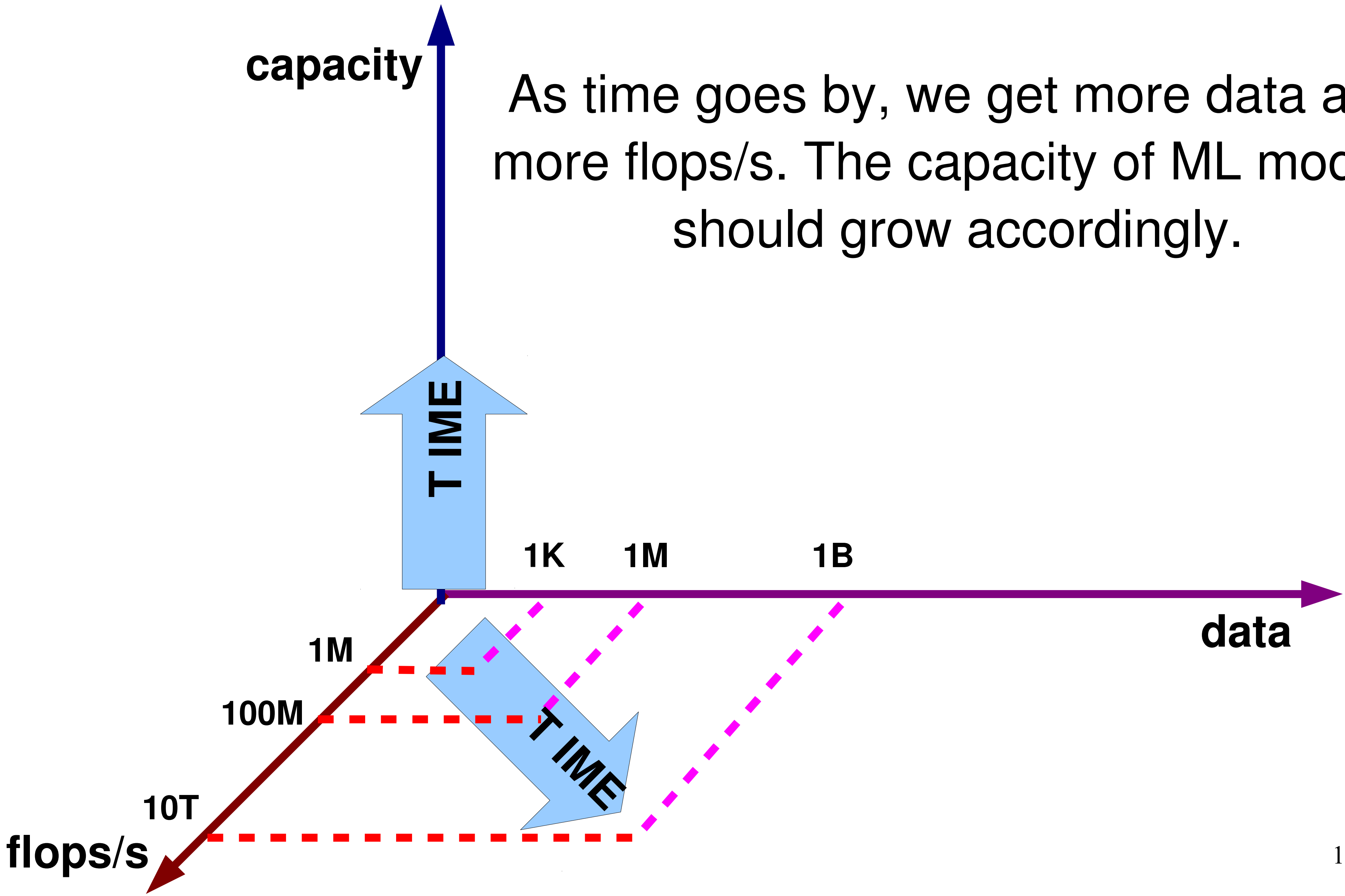
- 2009 **LeCun:** harsher non-linearities, normalization layer, learning unsupervised and supervised.

- 2011 **Mallat:** provides a theory behind the architecture

- 2012 **Hinton:** use bigger & deeper nets, GPUs, more data

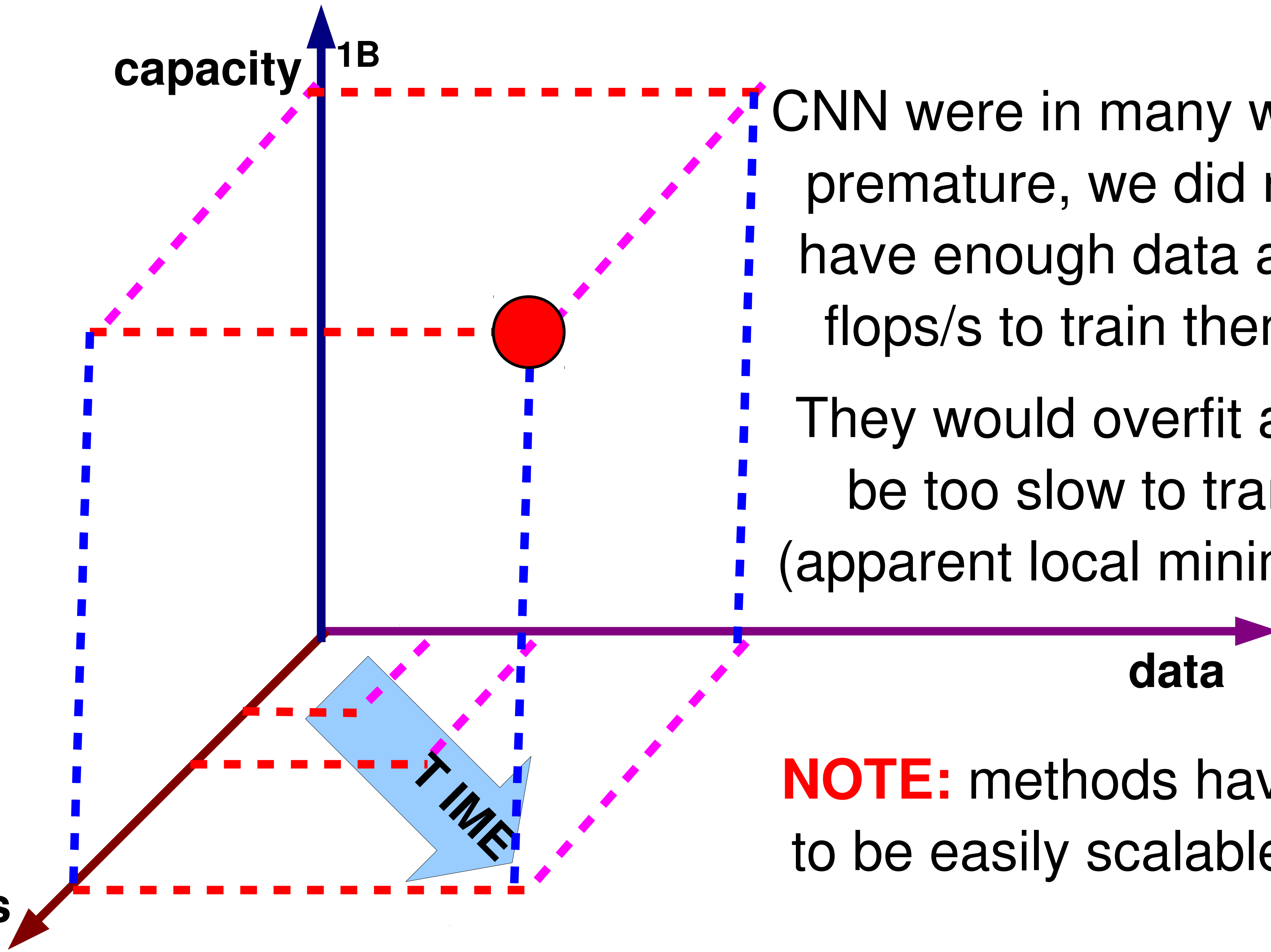
LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998

ConvNets: Why so successful now?



As time goes by, we get more data and more flops/s. The capacity of ML models should grow accordingly.

ConvNets: Why so successful now?



CNN were in many ways premature, we did not have enough data and flops/s to train them.

They would overfit and be too slow to tran (apparent local minima).

NOTE: methods have to be easily scalable!