Feature Learning with Deep Networks for Image Classification

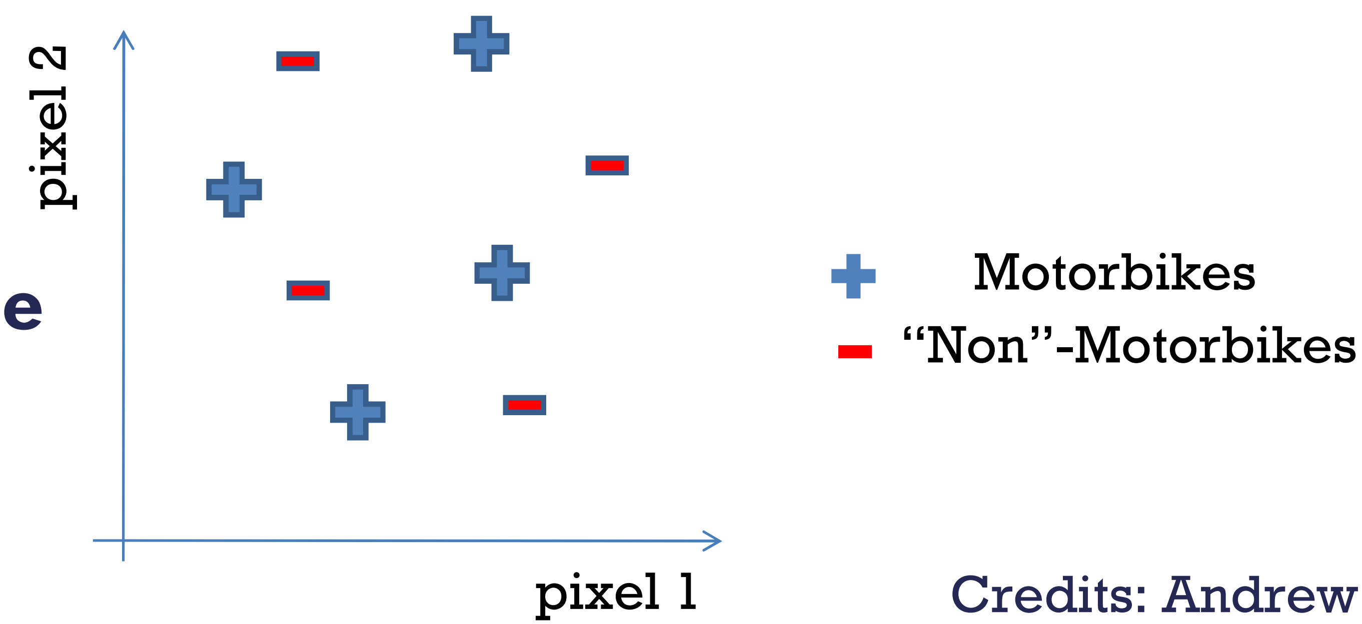
Pardis Noorzad
Department of Computer Engineering and IT
Amirkabir University of Technology

Computer Vision Seminar
Sharif University of Technology
Ordibehesht 1390

Feature representation: pixels

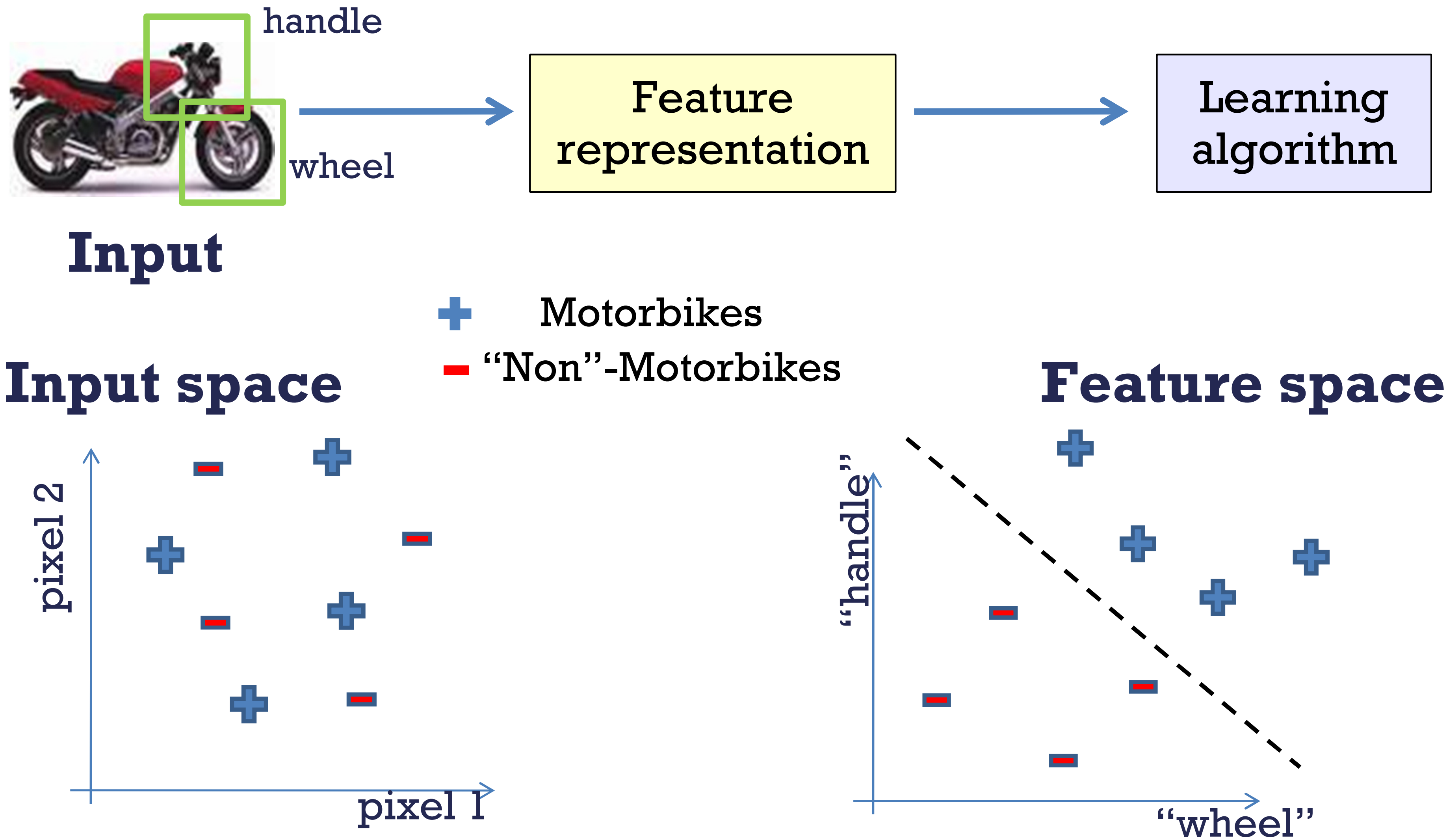
Input

Input space

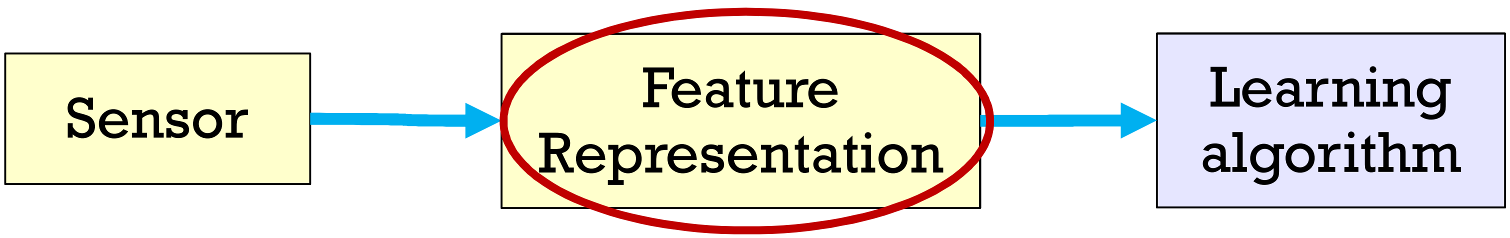


Credits: Andrew Ng

Feature representation: high level



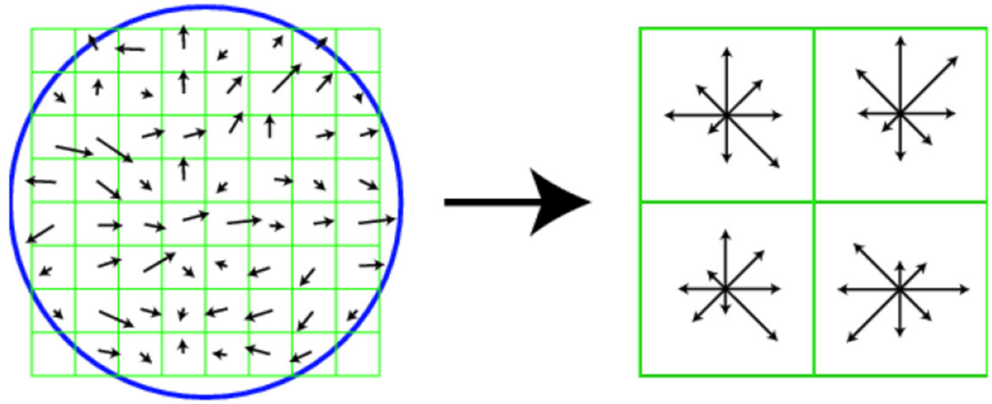
Feature representation



Credits: Andrew Ng

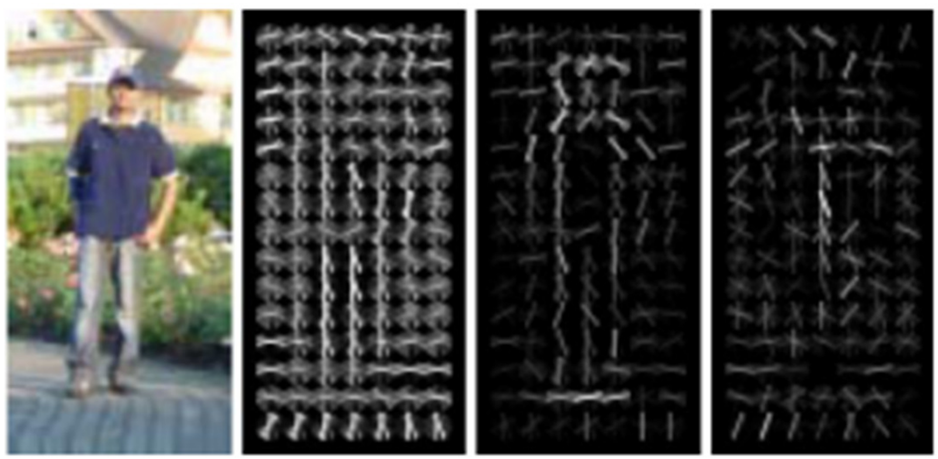
Computer vision features

SIFT



(a) image gradients (b) keypoint descriptor

HoG



PCA-SIFT

SURF

GLOH

LESH

GIST

etc.

Feature representation

- Features are designed to capture **invariance**
  - Scale-invariance

Problems of hand-tuned features

- Needs expert knowledge
- Time-consuming and expensive
- Does not generalize to other domains

- But we can't possibly be able to hard-code and foresee **all of them**
  - Out-of-plane rotations
  - deformable parts, etc.

Can we learn features?

Unlabeled images

↓

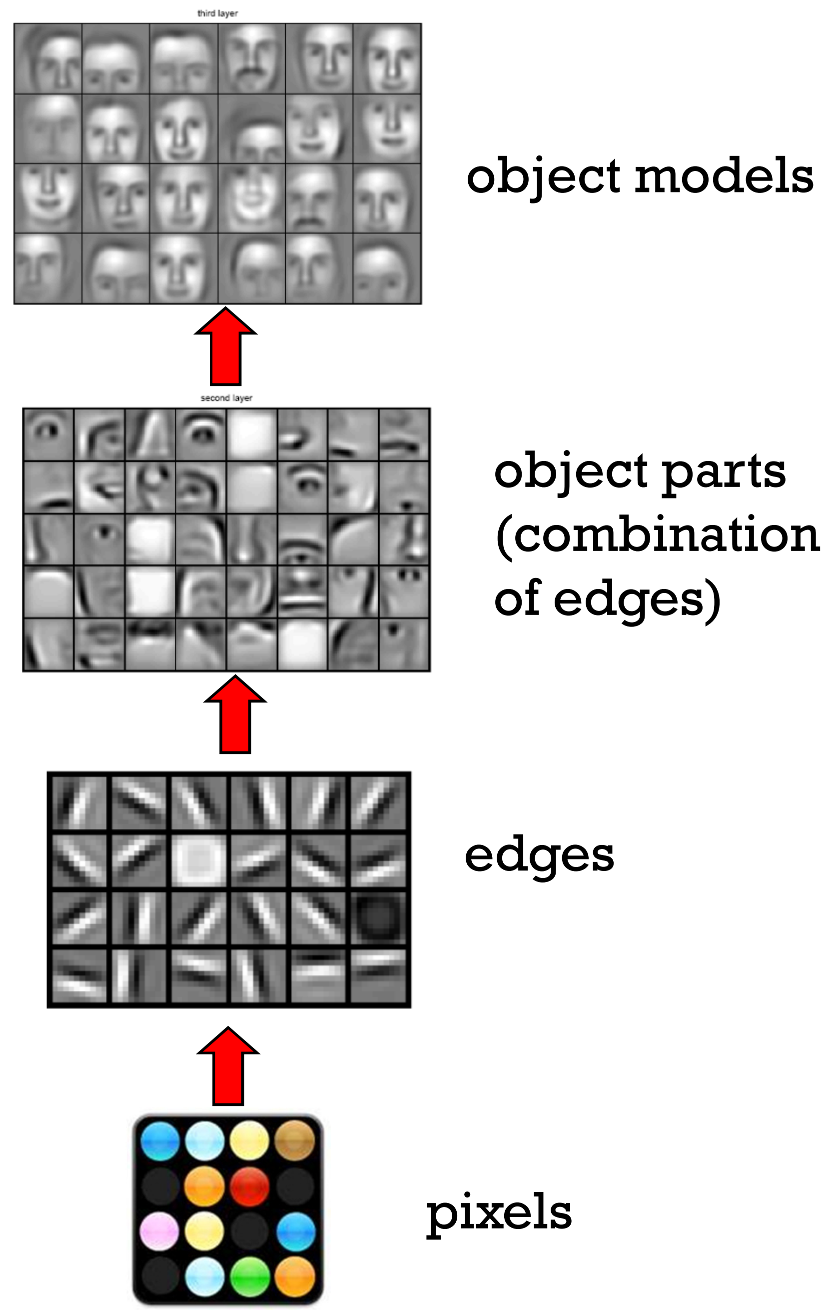
Learning algorithm

↓

Feature representation

Credits: Andrew Ng

Apparently, yes ☺



Credits: Andrew Ng

Self-taught learning

Unlabeled images (random internet images)

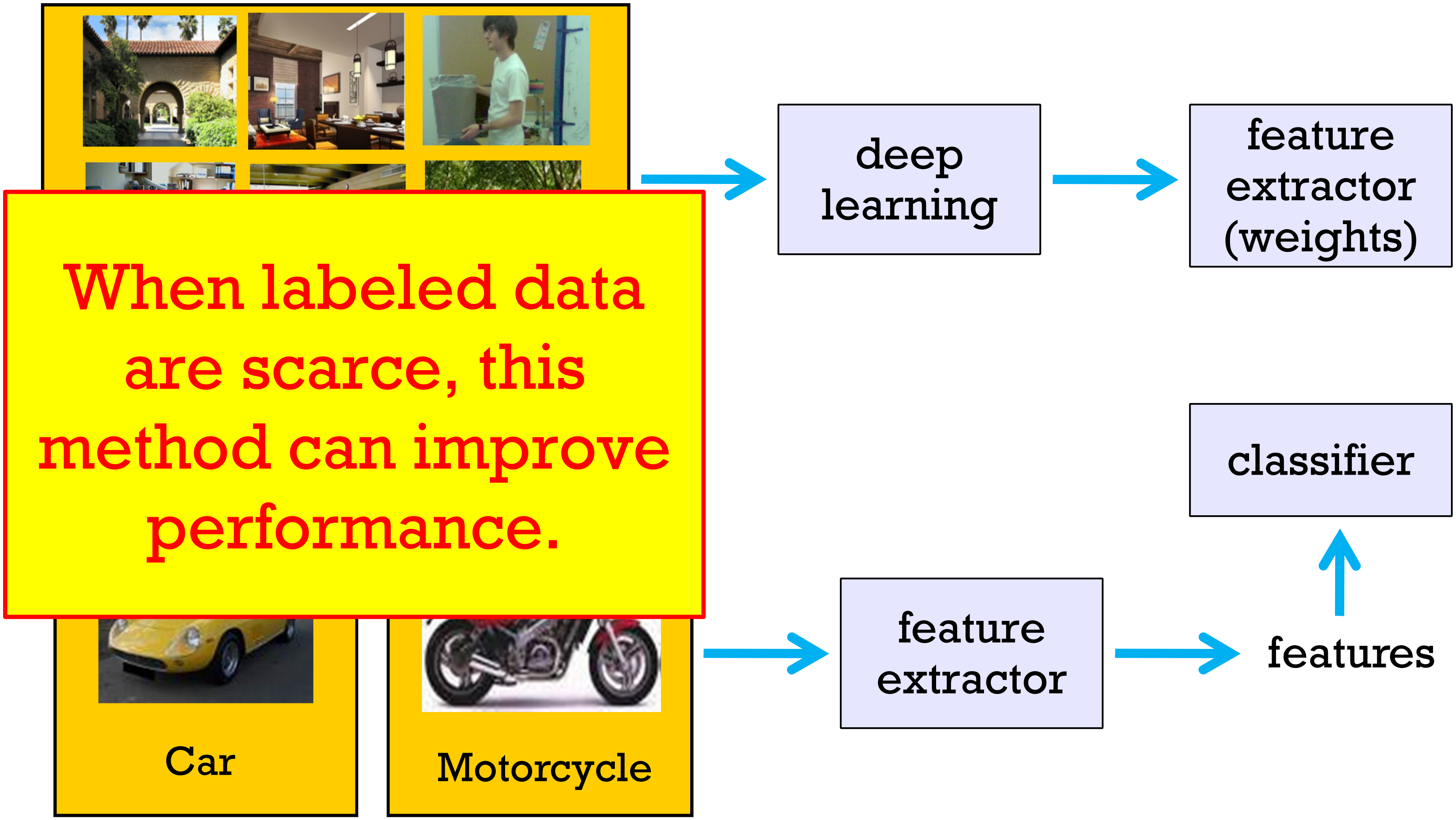
Car

Motorcycle

Testing:
What is this?

Credits: Andrew Ng

Self-taught learning: continued

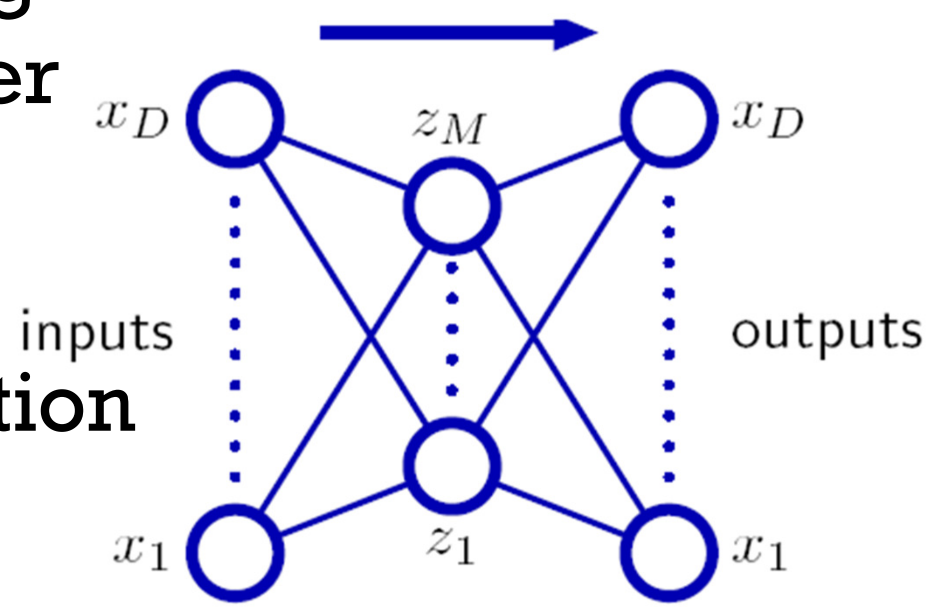


Neural nets for dimension reduction

- Nonlinear capabilities of Isomap and LLE were not brought by inherent **nonlinear models of data**
- Also, both methods use **'local'** generalization
- Apart from **supervised** learning for **classification**, neural nets can be used in the context of **unsupervised** learning for **dimensionality reduction**

Autoassociative NN
(M. A. Kramer, 1991)

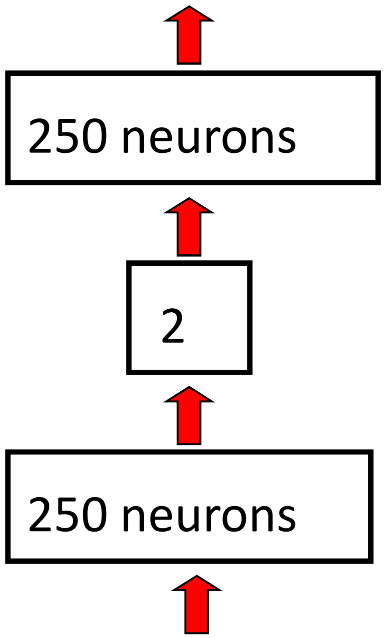
- DR achieved by using net with same number of input and outputs
- Optimize weights to minimize reconstruction error
- Net tries to map each input vector onto itself



Credits: C.M. Bishop

Autoassociative NN: the intuition

- Net is trained to reproduce its input at the output
- So it packs as much information as possible into the central bottleneck



Autoassociative NN: optimization

- Number of hidden units is smaller than number of inputs
  - there exists a **reconstruction error**
- Determine network weights by minimizing the reconstruction sum-of-squares error:

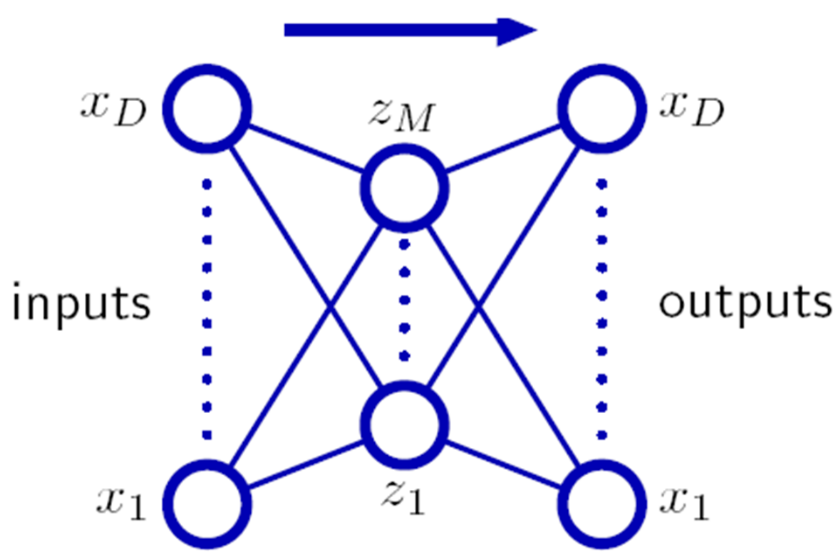
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\|^2$$

Autoassociative NN and PCA

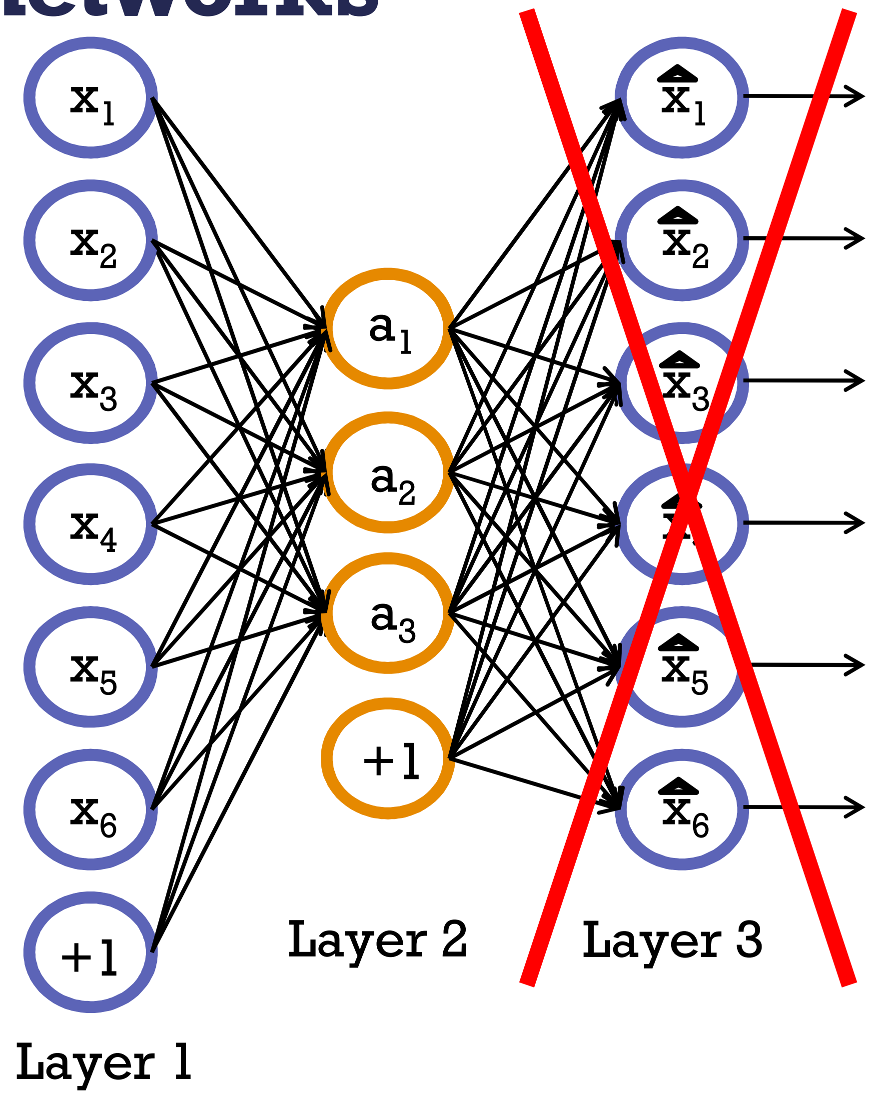
- Here’s an interesting fact:
- If hidden units have linear activation functions,
- Error function has a **unique global minimum**
- At this minimum, the network performs a projection onto an **M**-dimensional subspace
  - spanned by the **first M PCs** of the data!

Autoassociative NN and PCA: continued

- Vector of weights leading into z_i's from a basis set which spans the principal subspace
- These vectors need not be orthonormal



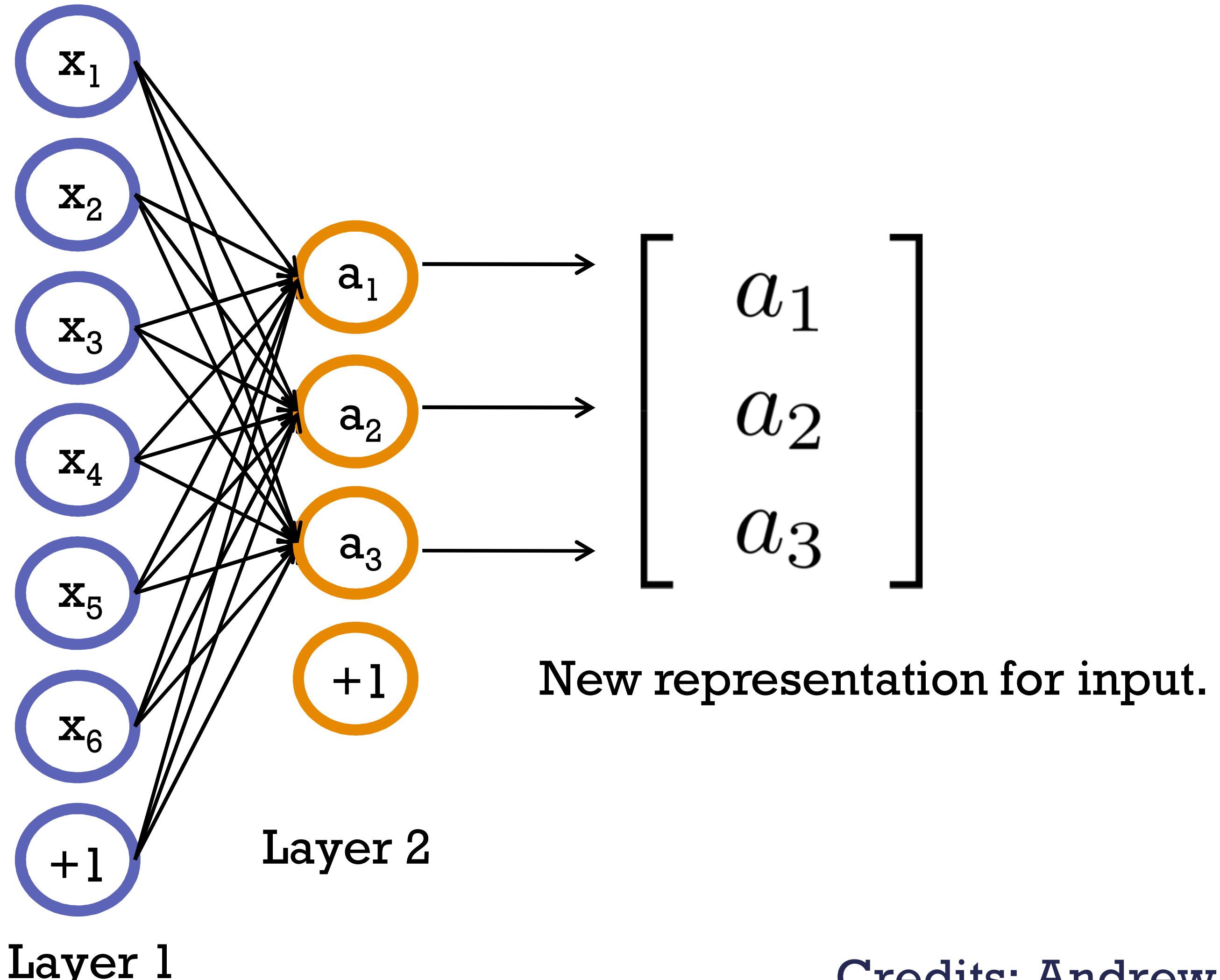
Unsupervised feature learning with neural networks



$$h_\theta(x) \approx x$$

Credits: Andrew Ng

Unsupervised feature learning with neural networks



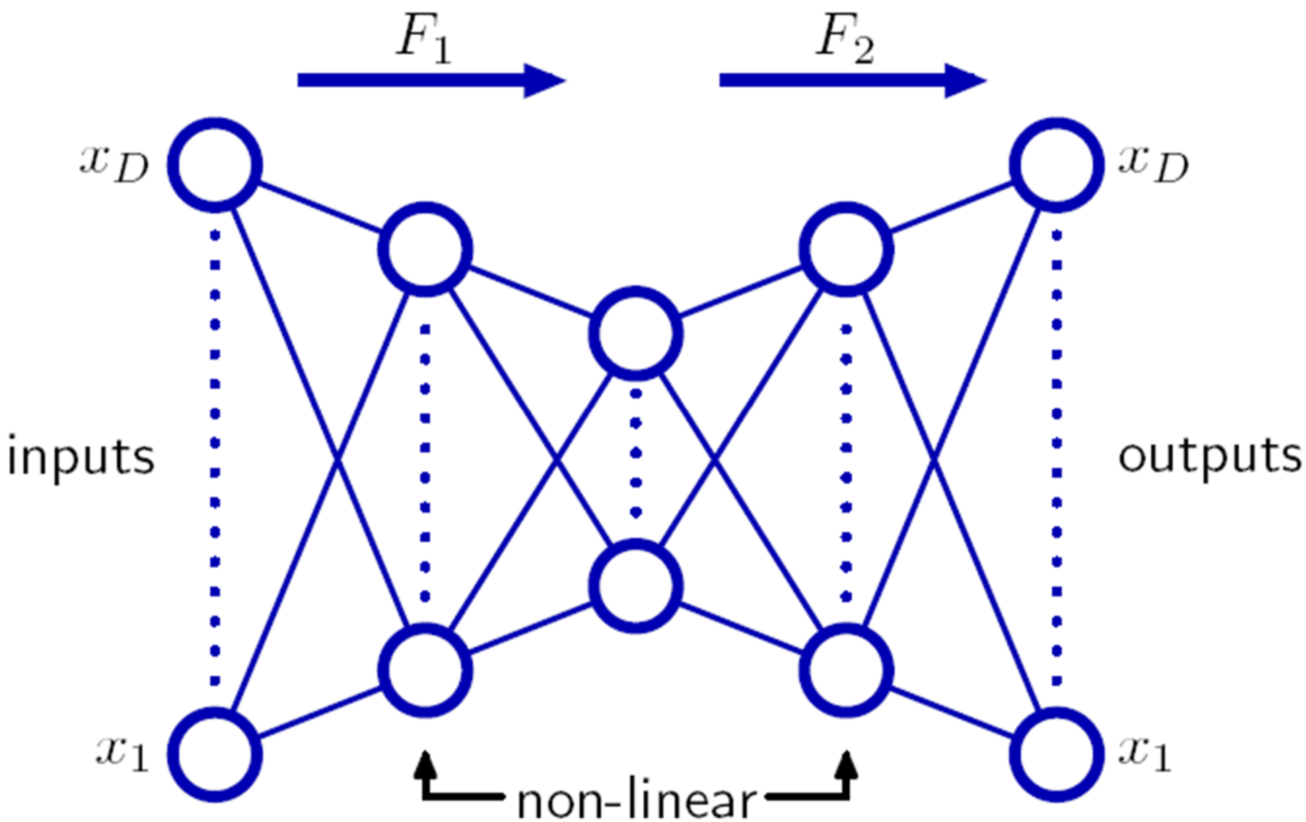
Credits: Andrew Ng

Autoassociative NN and PCA: continued

- **BUT**, even with nonlinear activation functions for the hidden units,
  - the min error solution is again the projection onto the PC subspace
  - so there is no advantage in using 2-layer NNs to perform DR
  - standard PCA techniques based on SVD are better

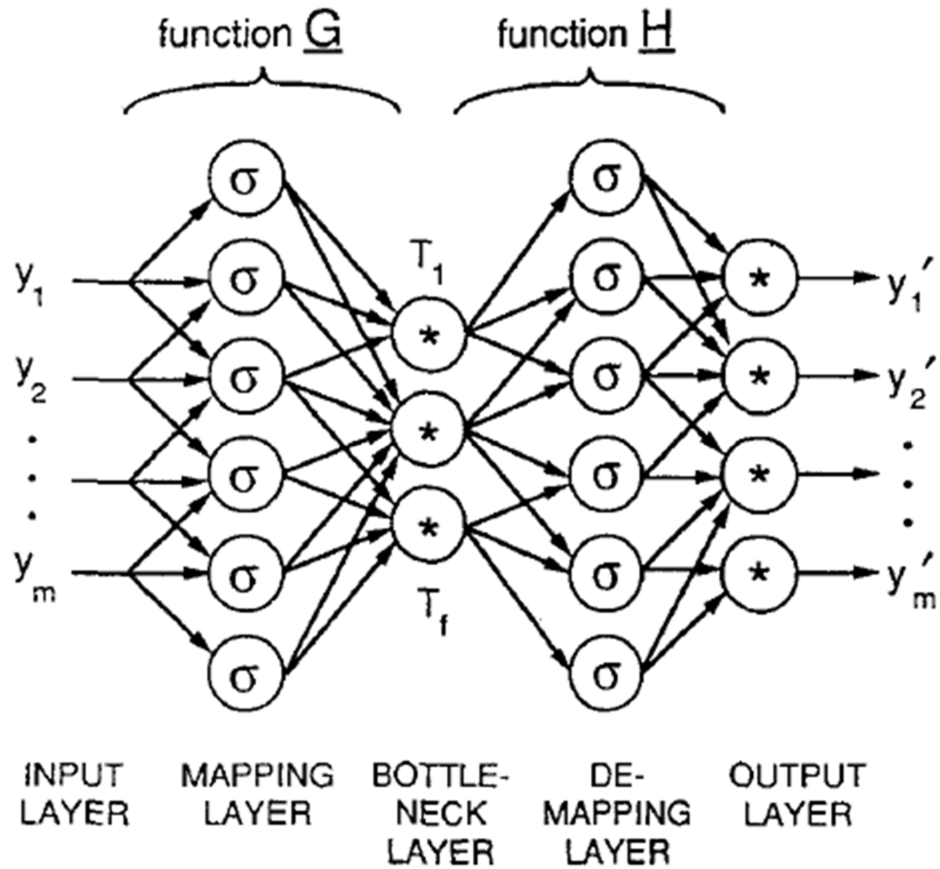
Autoassociative NN: nonlinear PCA

- What we need is additional hidden layers – e.g. the 4-layer net below



Autoassociative NN: NLPCA

- Training to learn the identity mapping is called
  - **self-supervised backpropagation** or
  - **autoassociation**
- After training, the combined net has no utility
  - and is divided into two single-hidden layer nets **G** and **H**



NLPCA: discussion

- Start with random weights,
- The two nets (**G** and **H**) can be trained together by minimizing the discrepancy between the original data and its reconstruction
- Error function as before (sum-of-squares)
  - no longer a quadratic function of net params. ☹
- Dimension of subspace must be specified before training ☹

Autoencoder

(G. E. Hinton and R. R. Salakhutdinov, 2006)

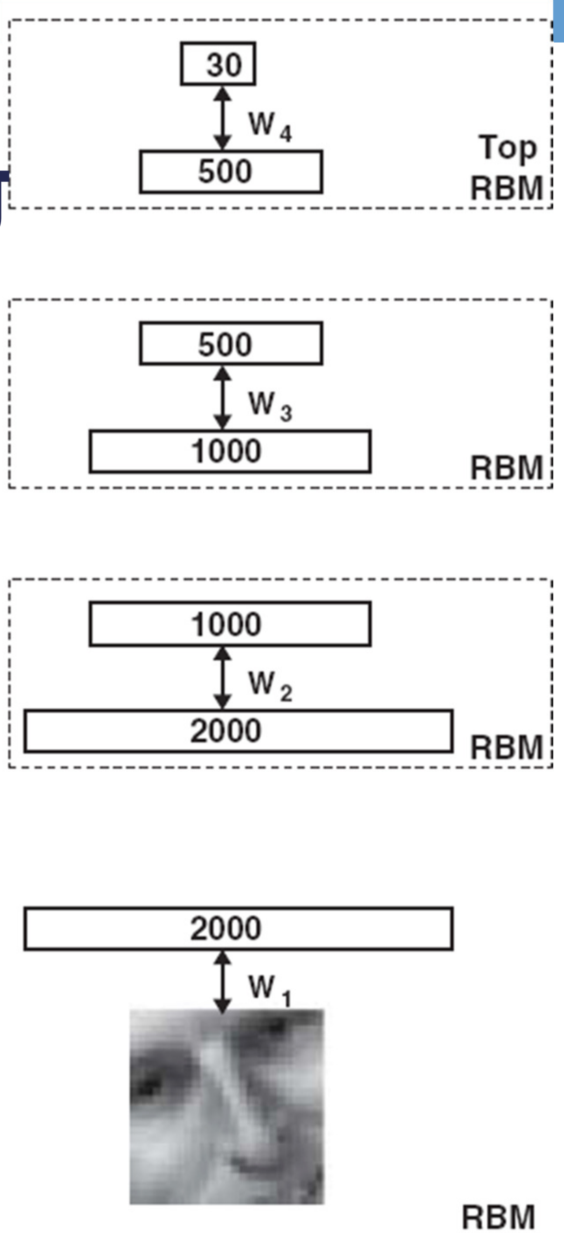
- It was known since the 1980s that **backpropagation** through **deep neural nets** would be very effective for **nonlinear dimensionality reduction** -- subject to:
  - fast computers … OK
  - big data sets … OK
  - **good initial weights** …

Autoencoder: continued

- BP = backpropagation (CG methods, steepest descent, …)
- Fundamental problems in training **nets with many hidden layers** (**“deep” nets**) with BP
  - learning is slow, results are poor
- But, results can be improved significantly if **initial weights** are close to solution

Autoencoder: pretraining

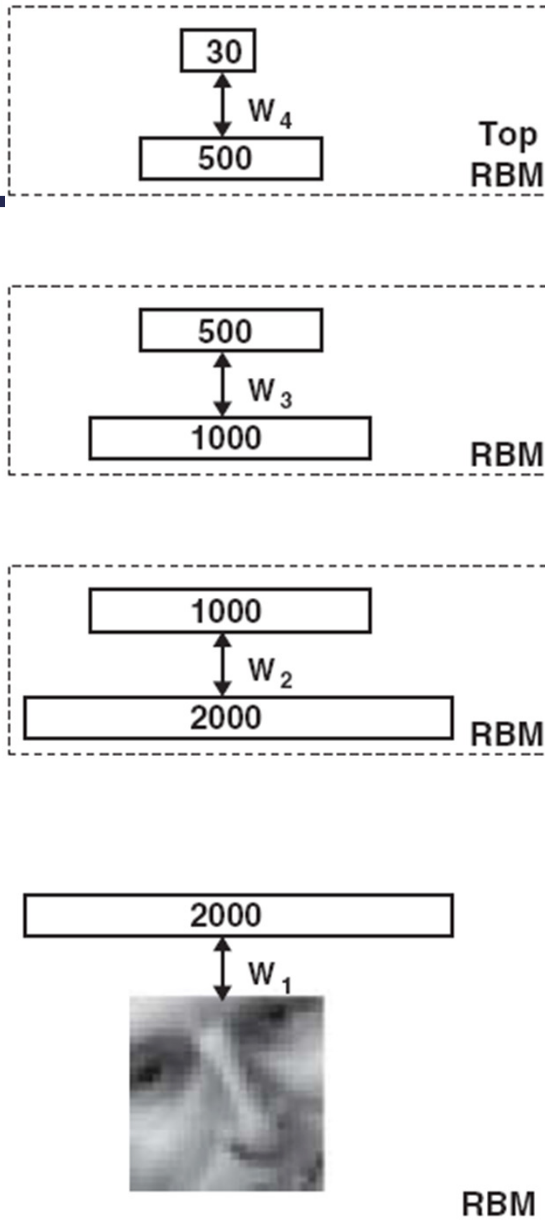
- Treating each neighboring set of two layers like an RBM
  - to approximate a good initial solution
- RBM = Restricted Boltzmann Machine
  - we'll explain later



Pretraining

Autoencoder: continued

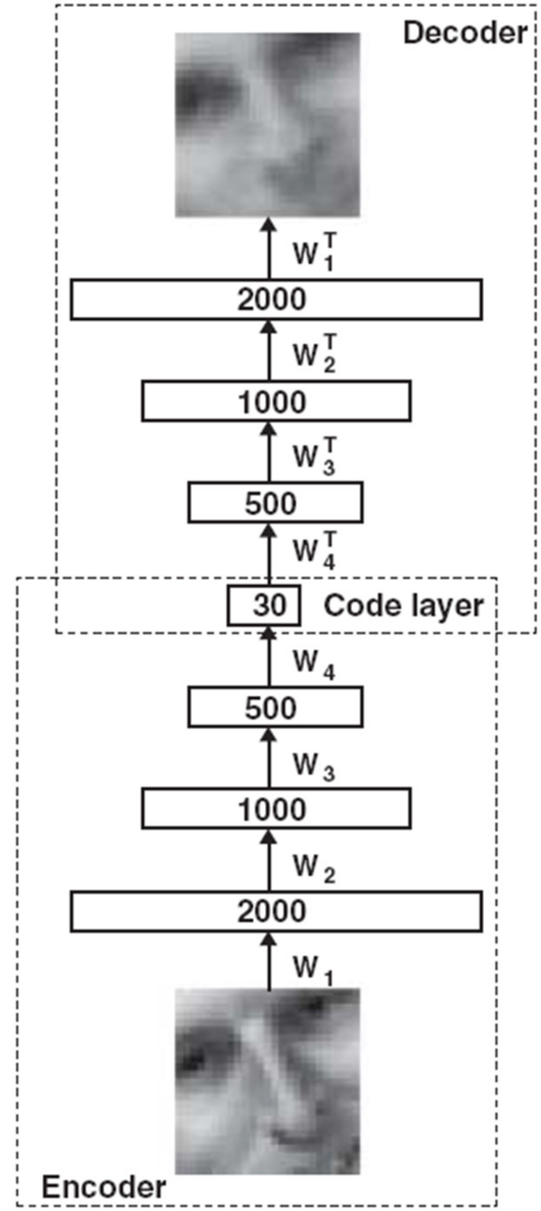
- The learned features of one RBM are used as data for training the next RBM in the stack
- The learning is unsupervised.



Pretraining

Autoencoder: unrolling

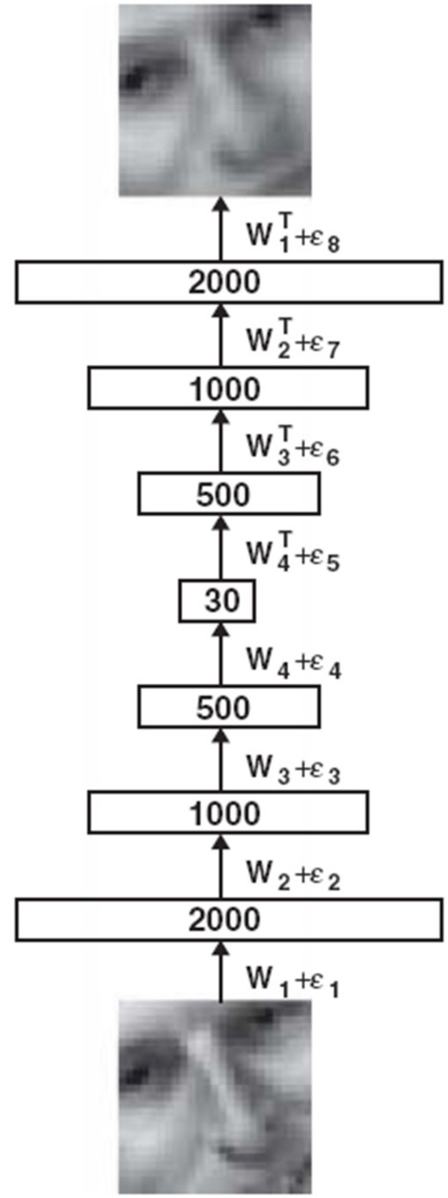
- After pretraining, the model is unfolded
- Produces encoder and decoder networks that use the same weights



Unrolling

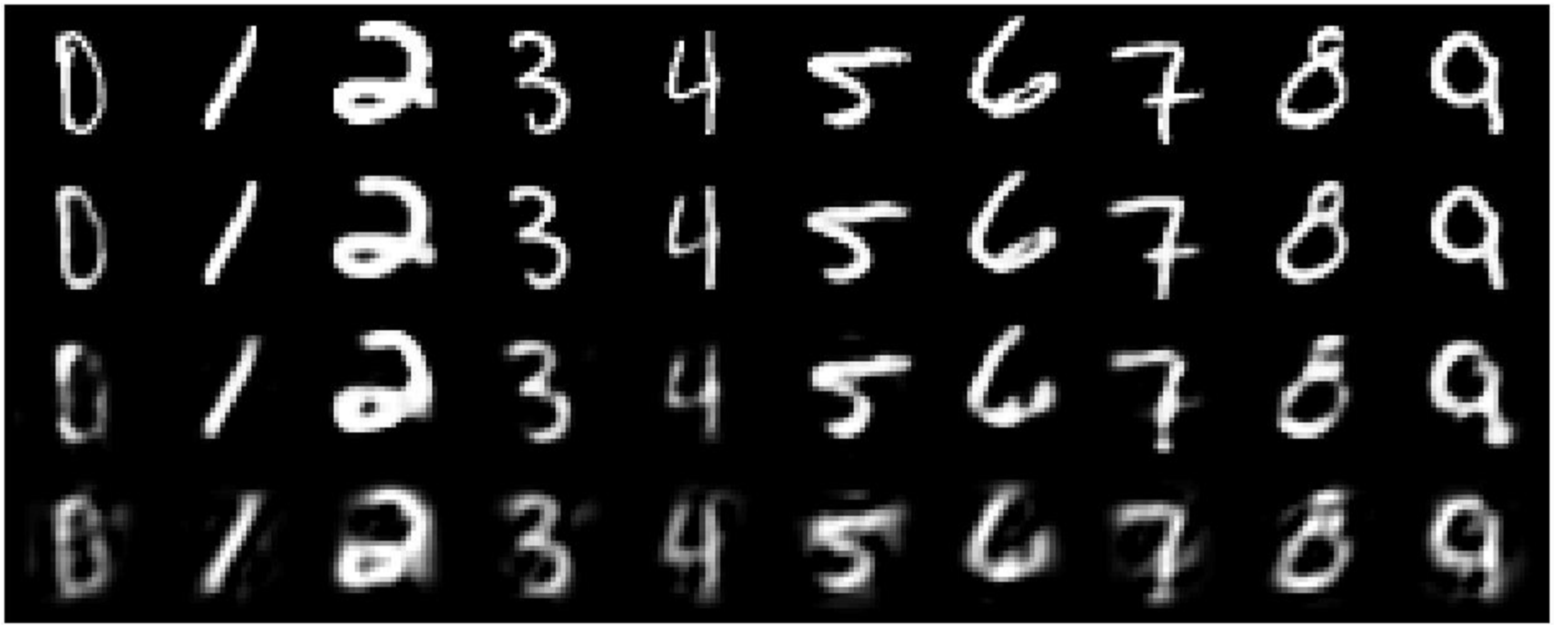
Autoencoder: fine-tuning

- Now use BP of error derivatives to fine-tune ☺
- So we don't run BP until we have good initial weights



Fine-tuning

Autoencoder: results



real data

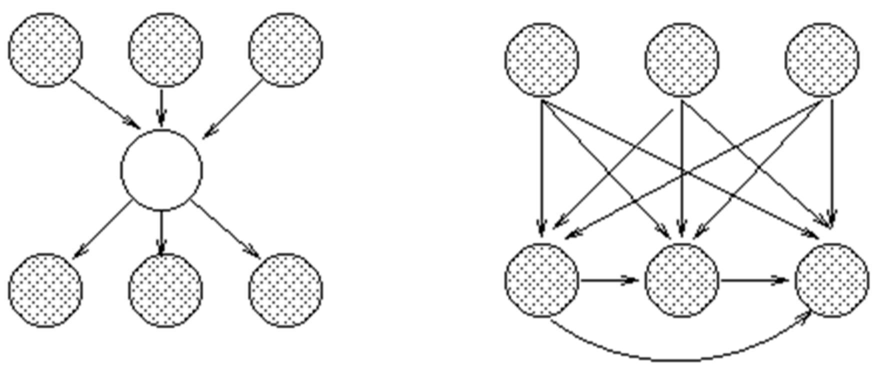
30-D deep auto

30-D logistic PCA

30-D PCA

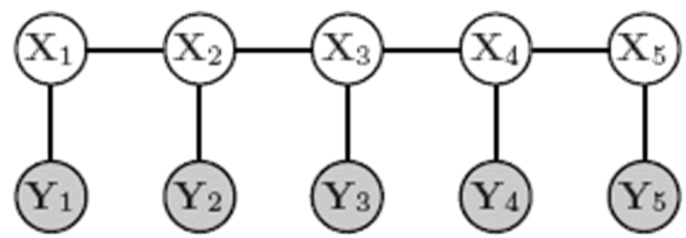
Graphical model

- "A **graphical model** is a probabilistic model for which a graph denotes the conditional independence structure between random variables." --Wikipedia



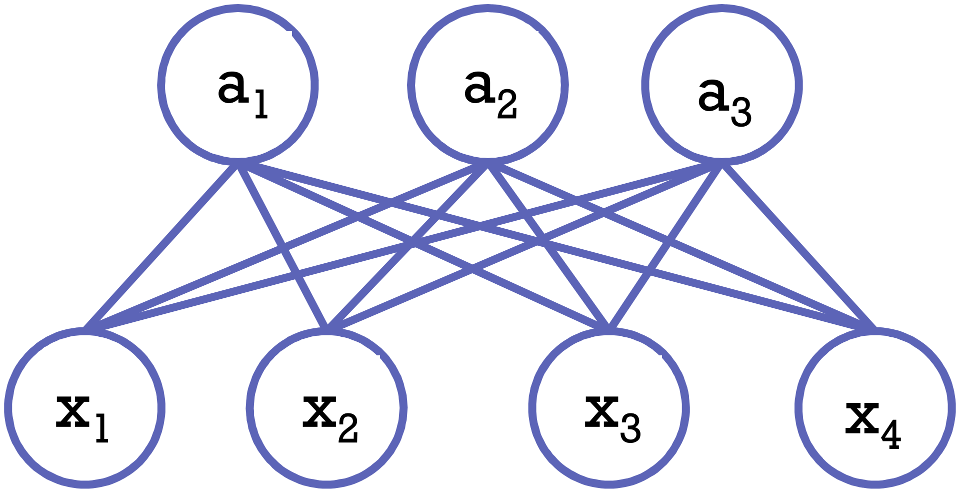
(a) (b)

Credits: Kevin Murphy



Credits: Leonid Sigal

Restricted Boltzmann machine (RBM)



Layer 2: $[a_1, a_2, a_3]$
(binary-valued)

Input $[x_1, x_2, x_3, x_4]$

MRF with joint distribution:

Simplest graphical model with hidden variables

Gi

likelihood estimation:

$$\max_W P(x) = \max_W \sum_a P(x, a)$$

Credits: Andrew Ng

Deep belief network (DBN)

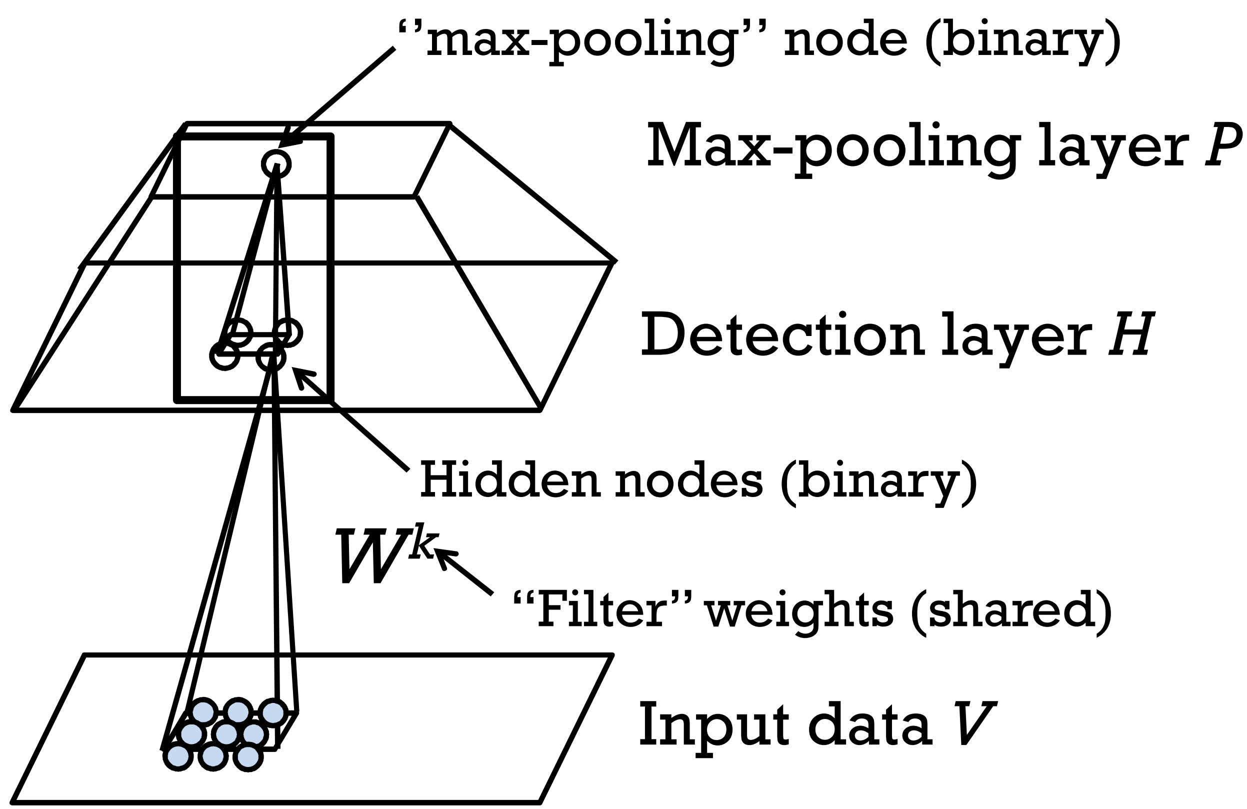
(G. E. Hinton et al., 2006)

- First train a layer of features that receive input directly from the pixels **(an RBM)**
- Then treat the activations of the trained features as if they were pixels and learn features of features in a second hidden layer.

It can be proved that each time we add another layer of features we improve a **variational lower bound on the log probability of the training data**. – G. Hinton

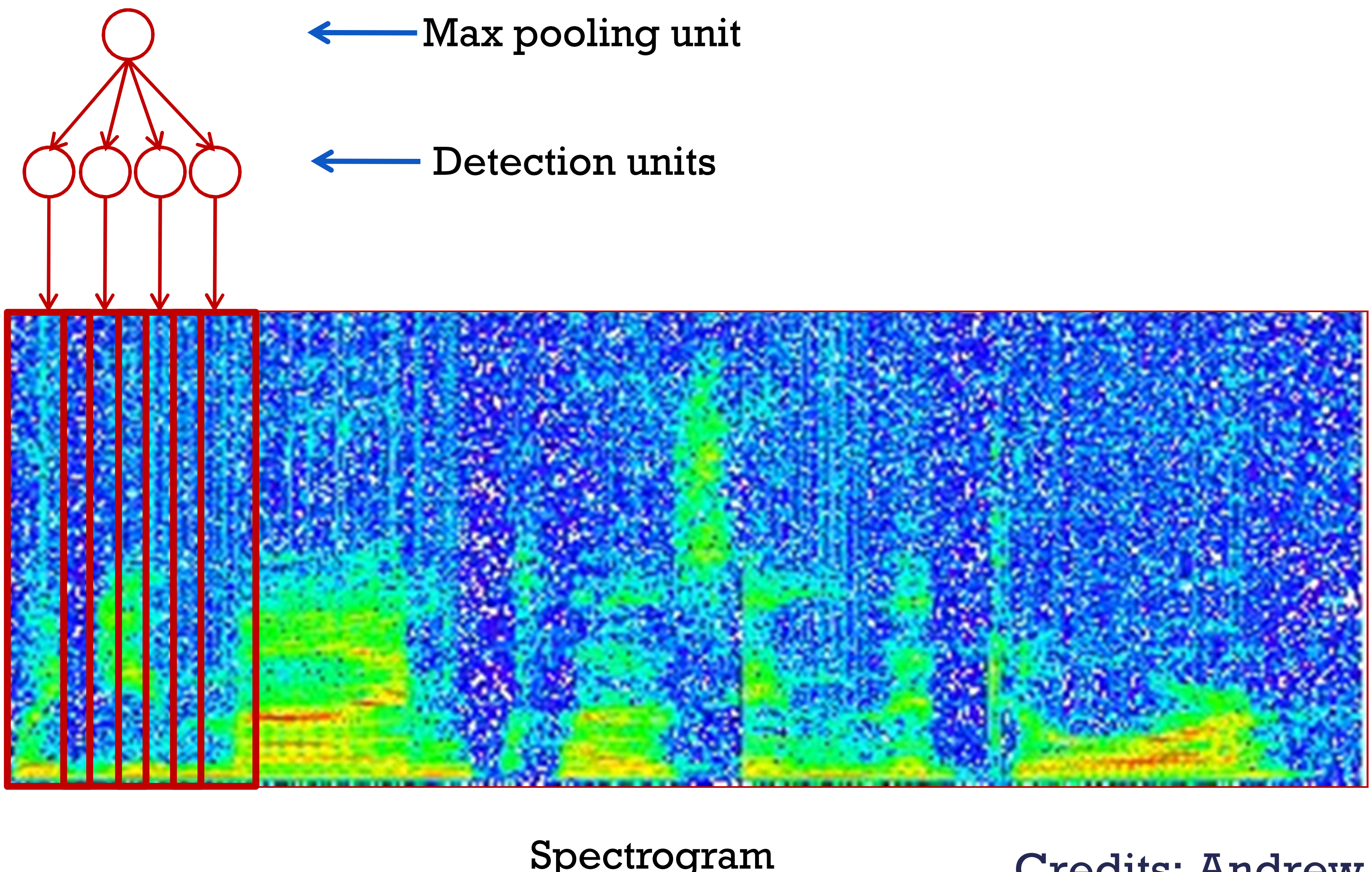
Convolutional DBN

(Lee et al. , ICML'09)



Credits: Andrew Ng

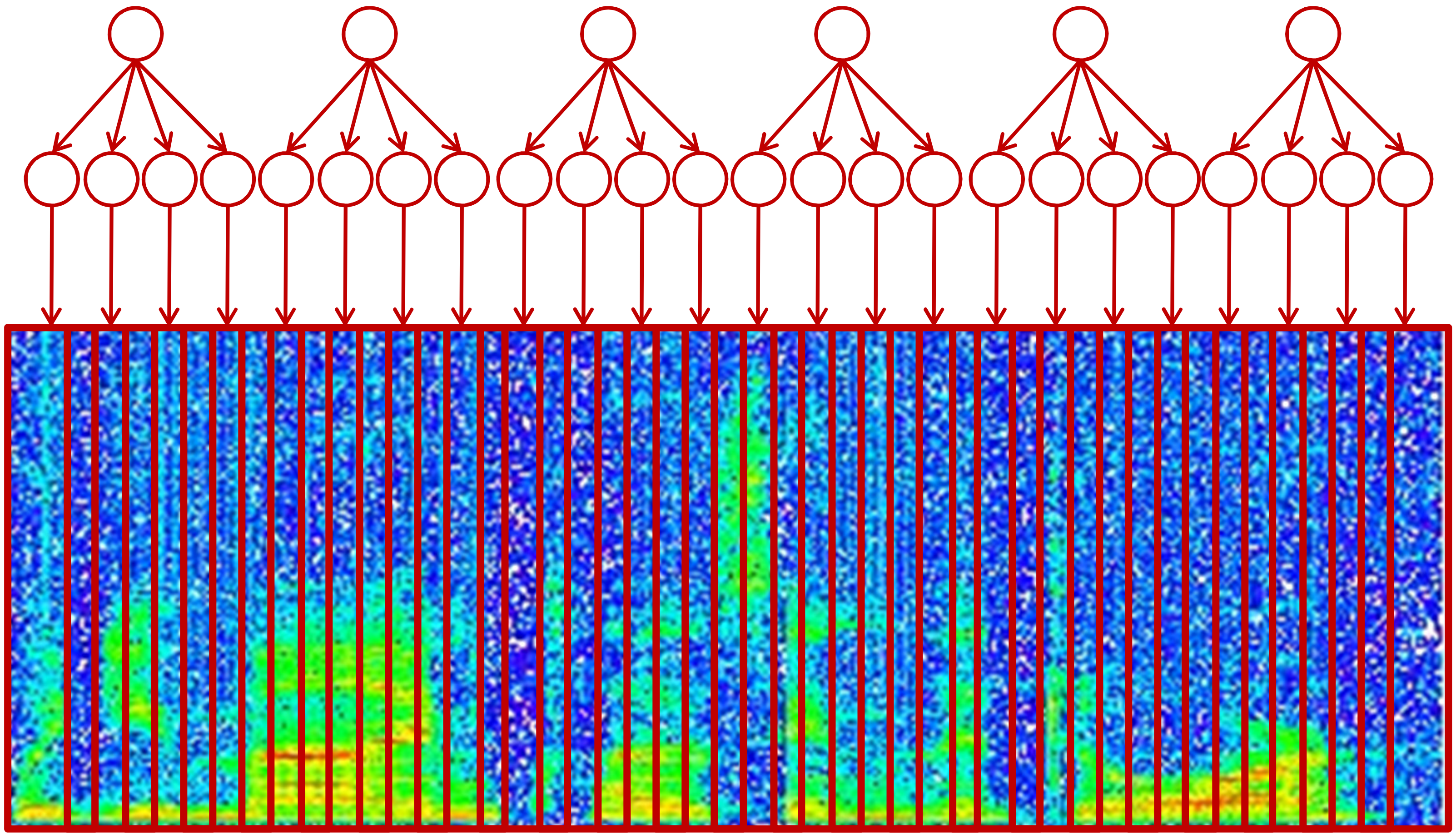
Convolutional DBN for audio



Credits: Andrew Ng

Convolutional DBN for audio

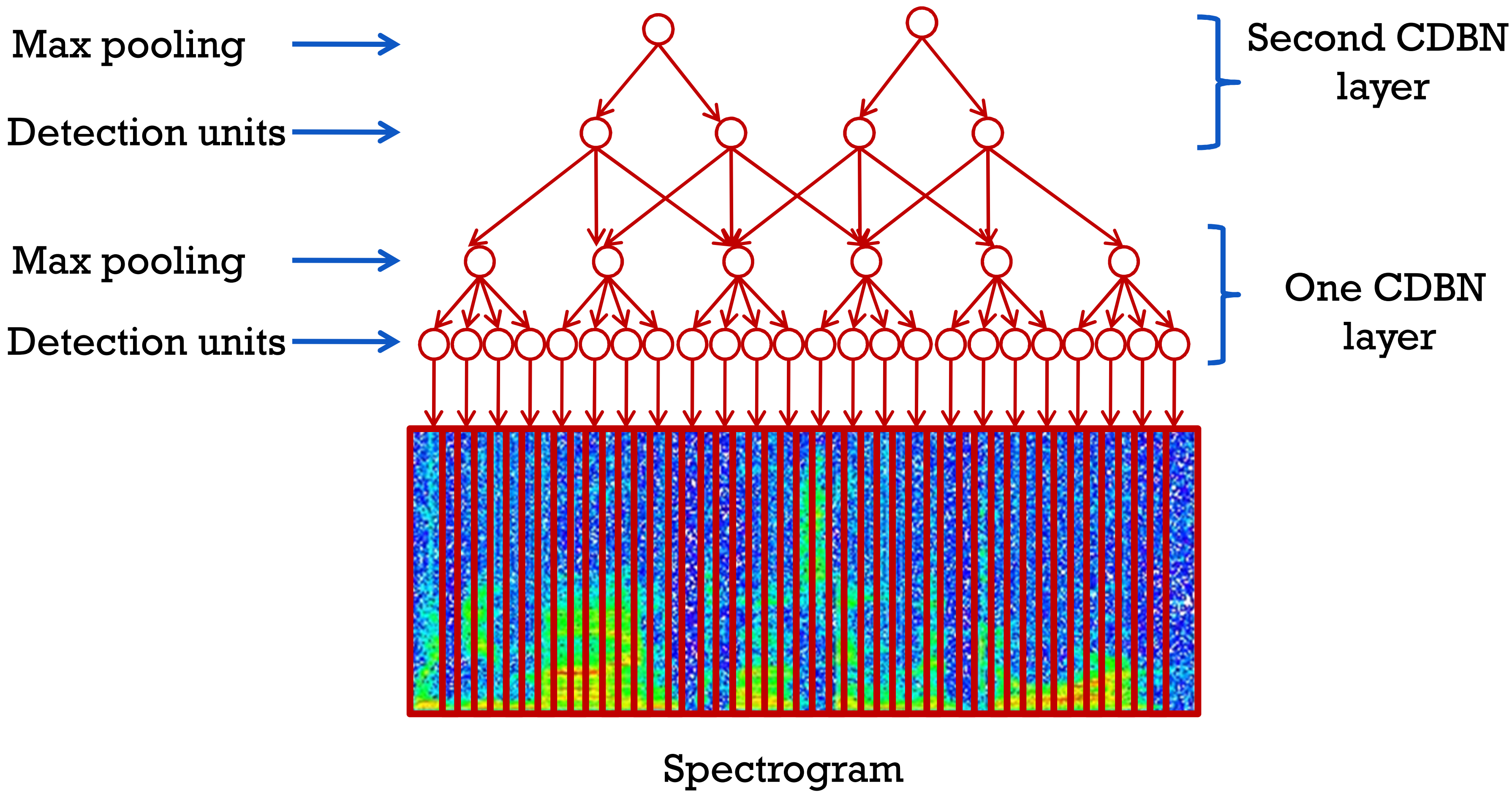
(Lee et al. NIPS'09)



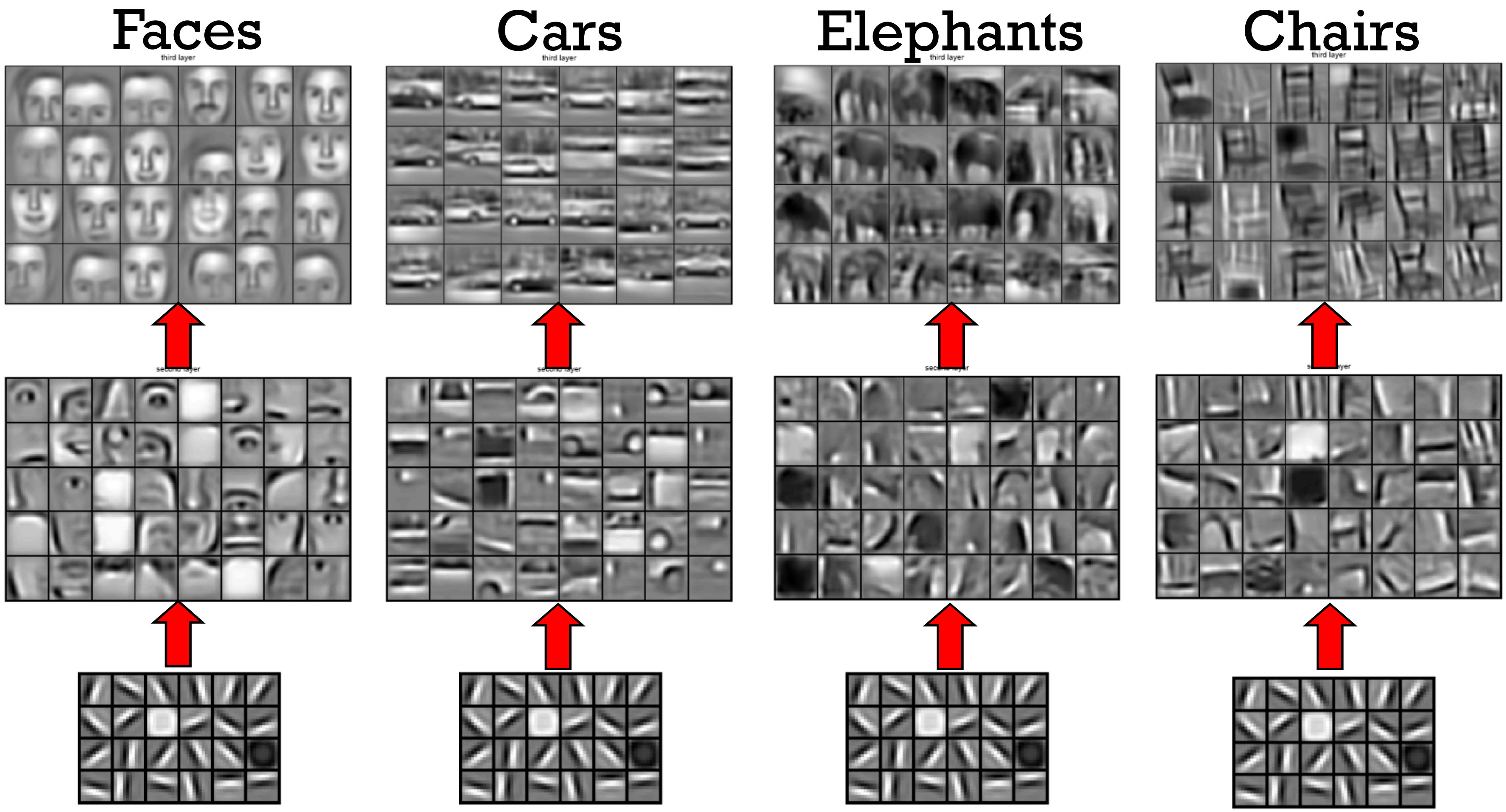
Spectrogram

Credits: Andrew Ng

Convolutional DBN for audio



Some results (Lee et al., ICML'09)



Credits: Andrew Ng

Some results on Caltech 101

(Lee et al., ICML'09)

Training Size	15	30
CDBN (first layer)	53.2±1.2%	60.5±1.1%
CDBN (first+second layers)	57.7±1.5%	65.4±0.5%
Raina et al. (2007)	46.6%	-
Ranzato et al. (2007)	-	54.0%
Mutch and Lowe (2006)	51.0%	56.0%
Lazebnik et al. (2006)	54.0%	64.6%
Zhang et al. (2006)	59.0±0.56%	66.2±0.5%

What to take away…

- **Feature learning with deep networks** can work **better than single hand-tuned features** on some classification tasks.
- Unsupervised feature learning can boost classification performance **when labeled data is scarce.**
- "when a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficiently deep one" – Y. Bengio

References

1. Bay Area Vision Meeting -- "Unsupervised Feature Learning and Deep Learning" by Andrew Ng (http://www.youtube.com/watch?v=ZmNOAtZIgIk)
2. "Pattern Recognition and Machine Learning" by Christopher M. Bishop
3. ECCV 2010 Tutorial on Feature Learning (http://ufldl.stanford.edu/eccv10-tutorial/)
4. "Computer Vision: Algorithms and Applications" by Richard Szeliski (http://szeliski.org/Book/)
5. UCL tutorial on "Deep Belief Nets" by Geoff Hinton

Thank you!

Have a good evening ☺