Deep Learning

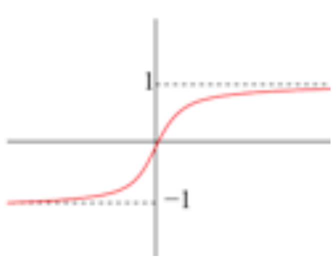
Hamid Mohammadi

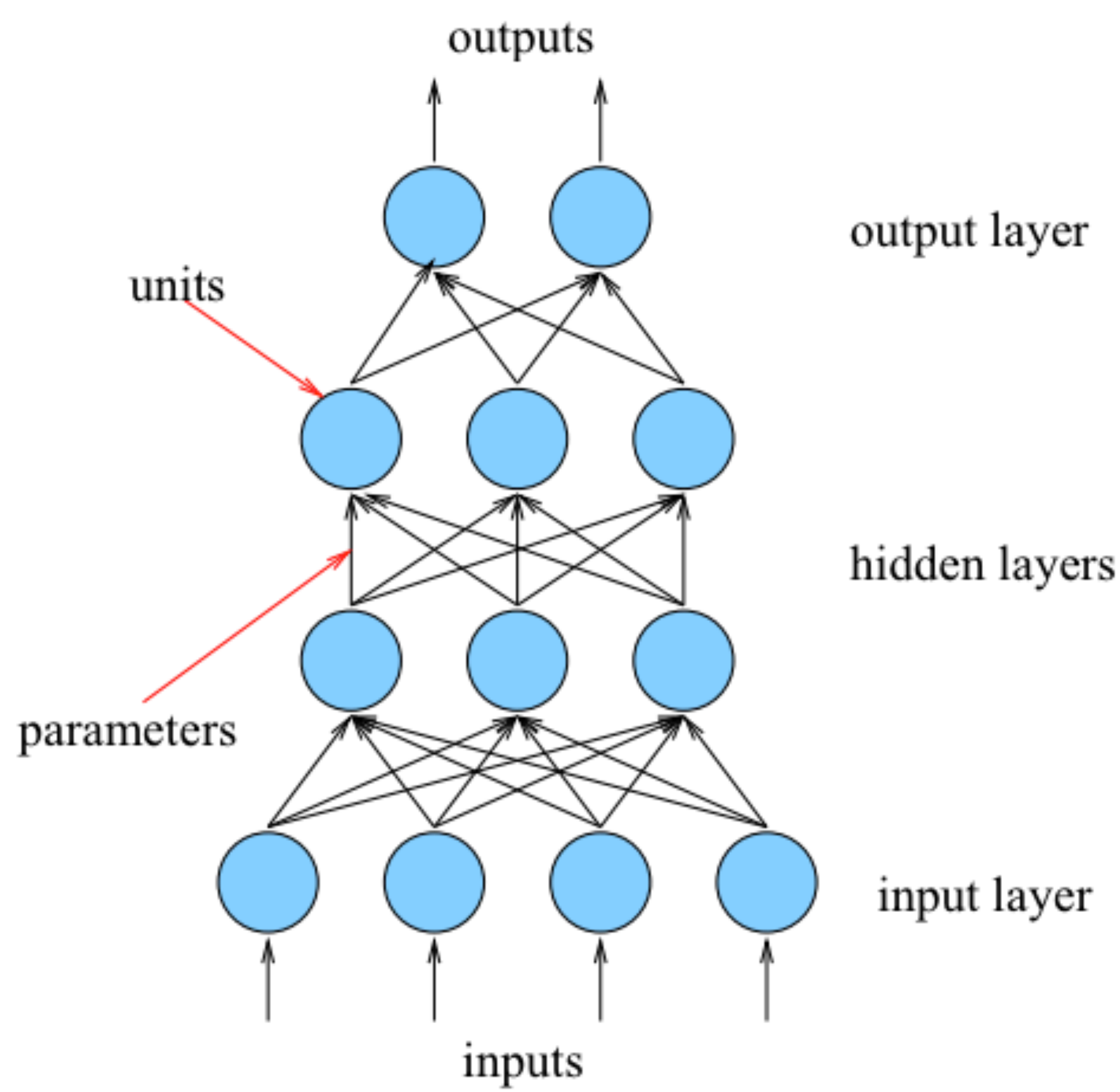
Machine Learning Course @ OHSU

2015-06-01

Recap: ANNs

- ANN is composed of multiple layers
- Layers perform non-linear transformations
    - `y=g(Wx+b)`

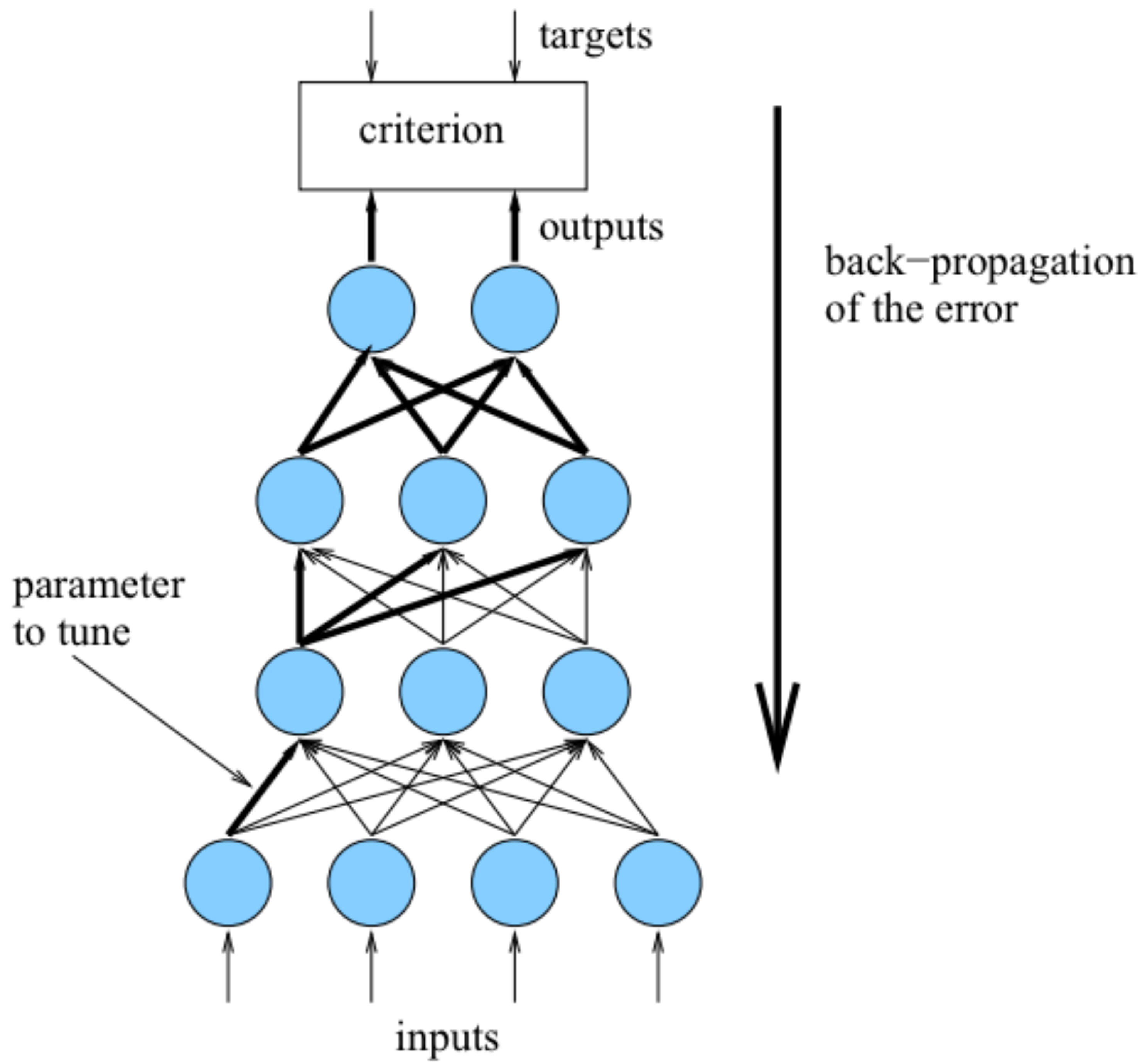




http://bengio.abracadoudou.com/lectures/old/tex_ann.pdf

Backpropagation

- Estimating model Parameters `Ws` and `bs`



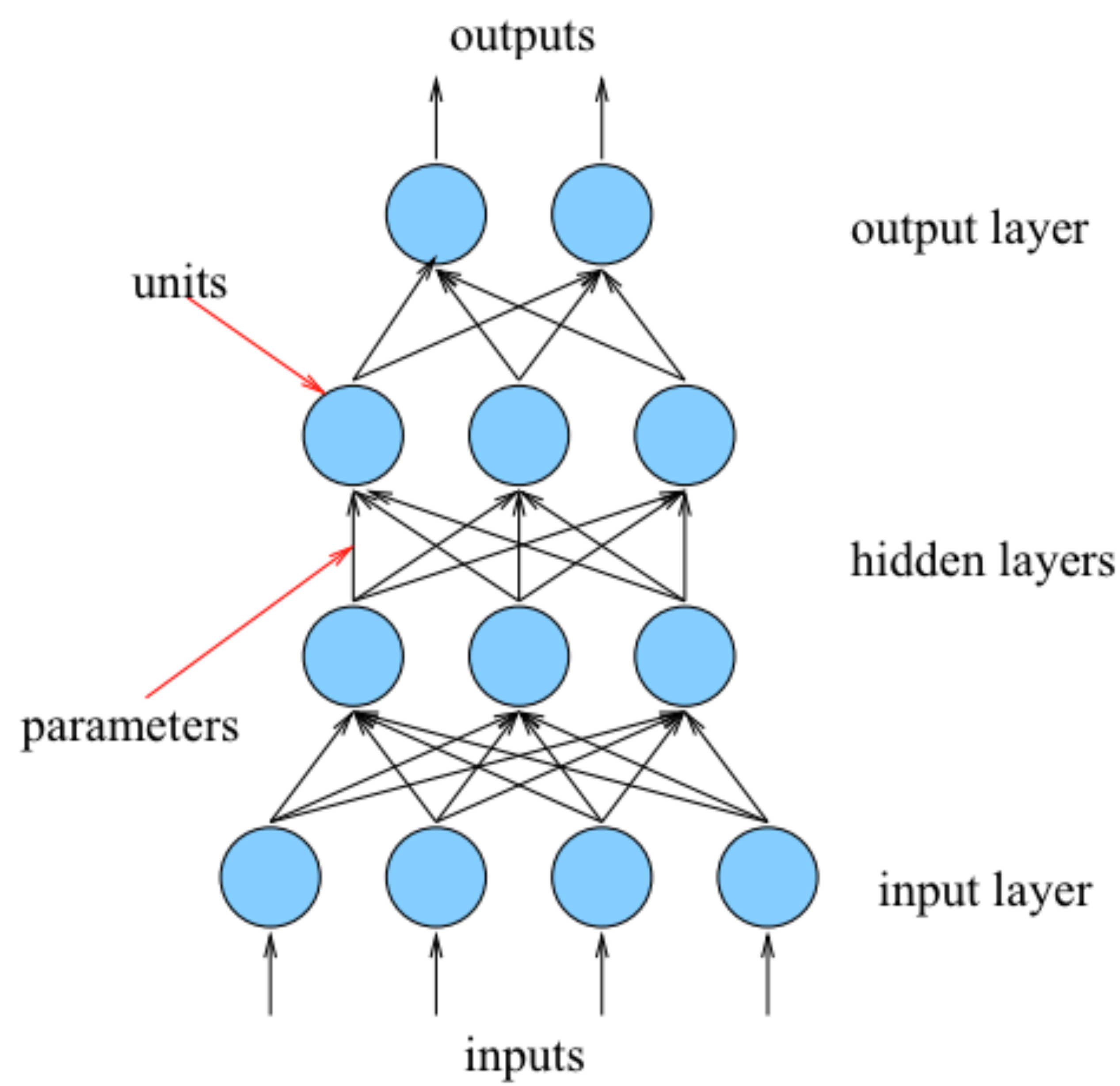
http://bengio.abracadoudou.com/lectures/old/tex_ann.pdf

Backpropagation

- Criterion for ANN
  - Mean Squared Error:
    - `Error=(ŷ-y)^2`
  - Cross-entropy
    - `Error=-sum(ŷlog(y)+ (1-ŷ)log(1-y))`

Deep ANNs

- ANN is called
  - Shallow if only # layers=2
  - Deep if #layers>2



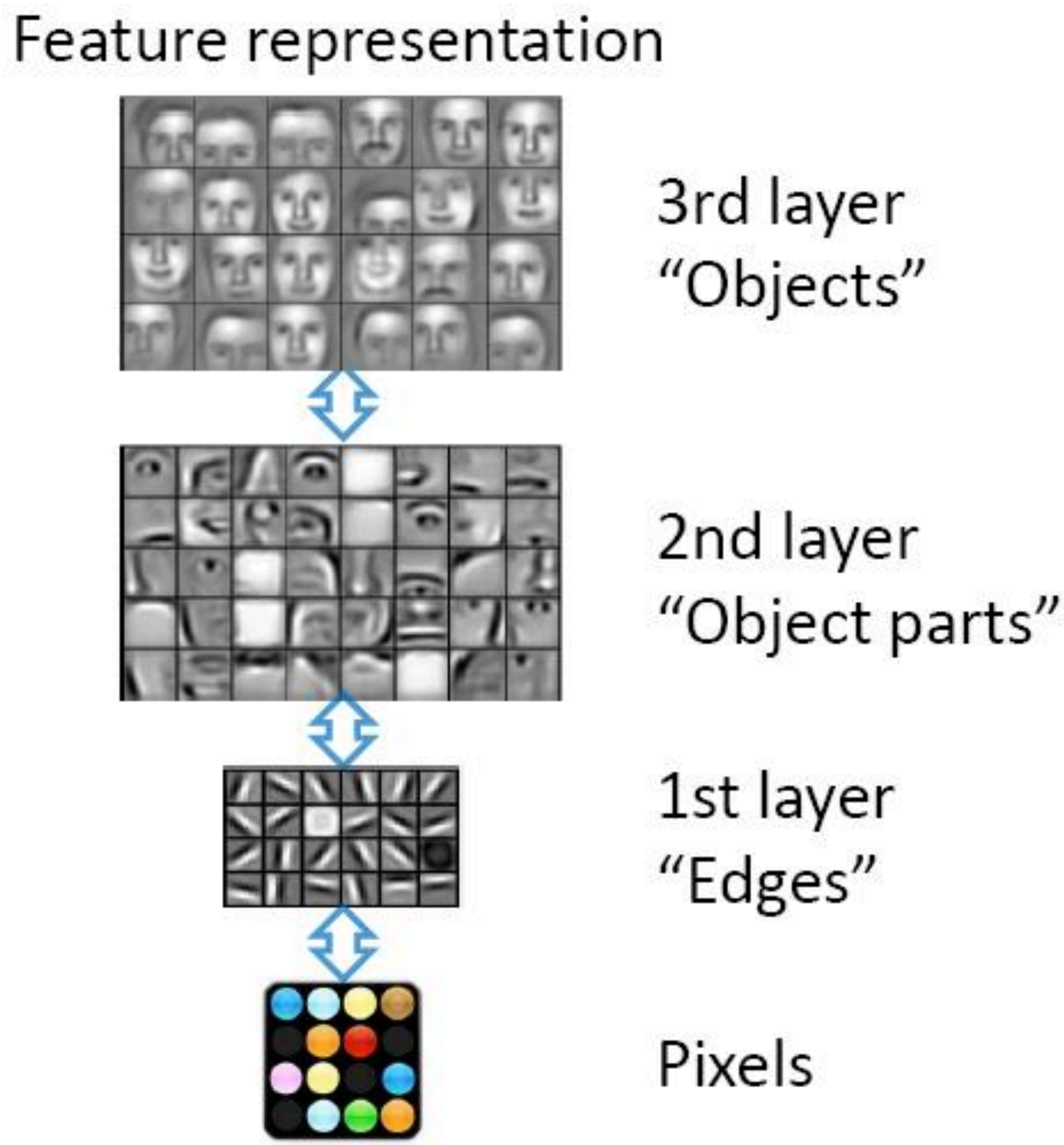
Why deep architecture?

- Isn't a two-layer ANN a universal approximator?
- Why deep architectures are needed?
  - The brain has a deep architecture
  - Cognitive processes seem deep
  - Insufficient depth can hurt

Why deep architecture?

- The brain has a deep architecture:
  - visual cortex has a sequence of levels
  - Each level represents the input at a different level of abstraction,
  - more abstract features further up in the hierarchy, defined in terms of the lower-level ones.
- Cognitive processes seem deep
- Insufficient depth can hurt

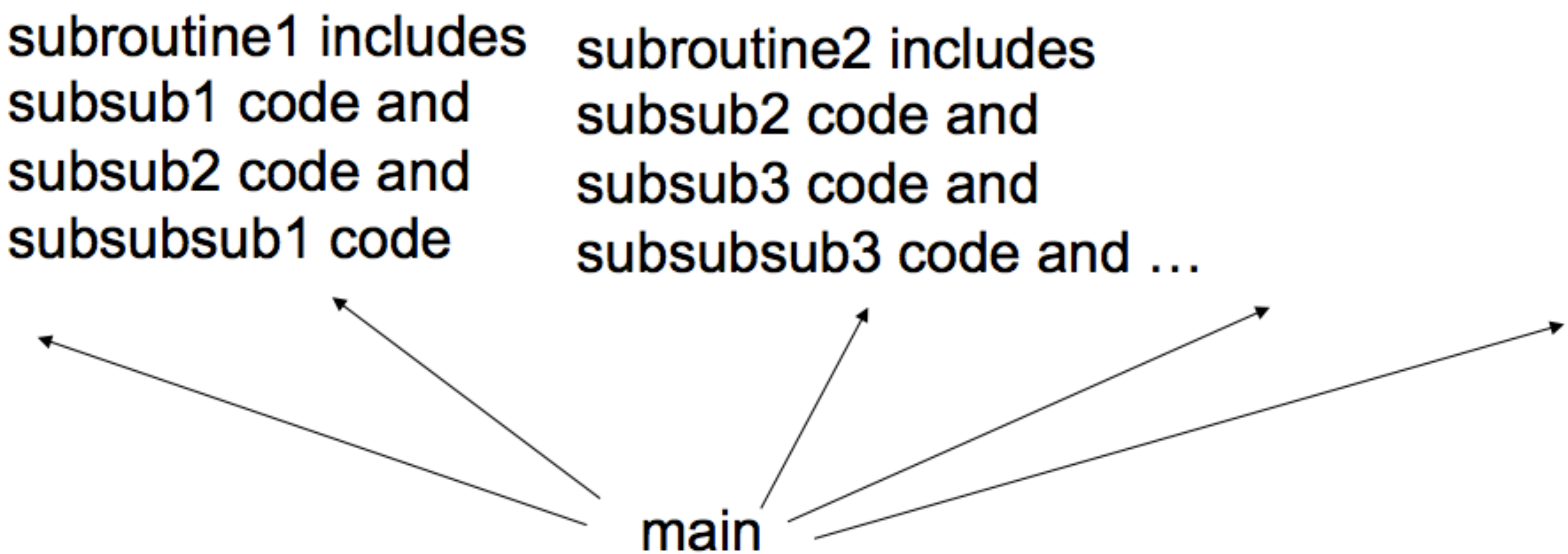
Why deep architecture?



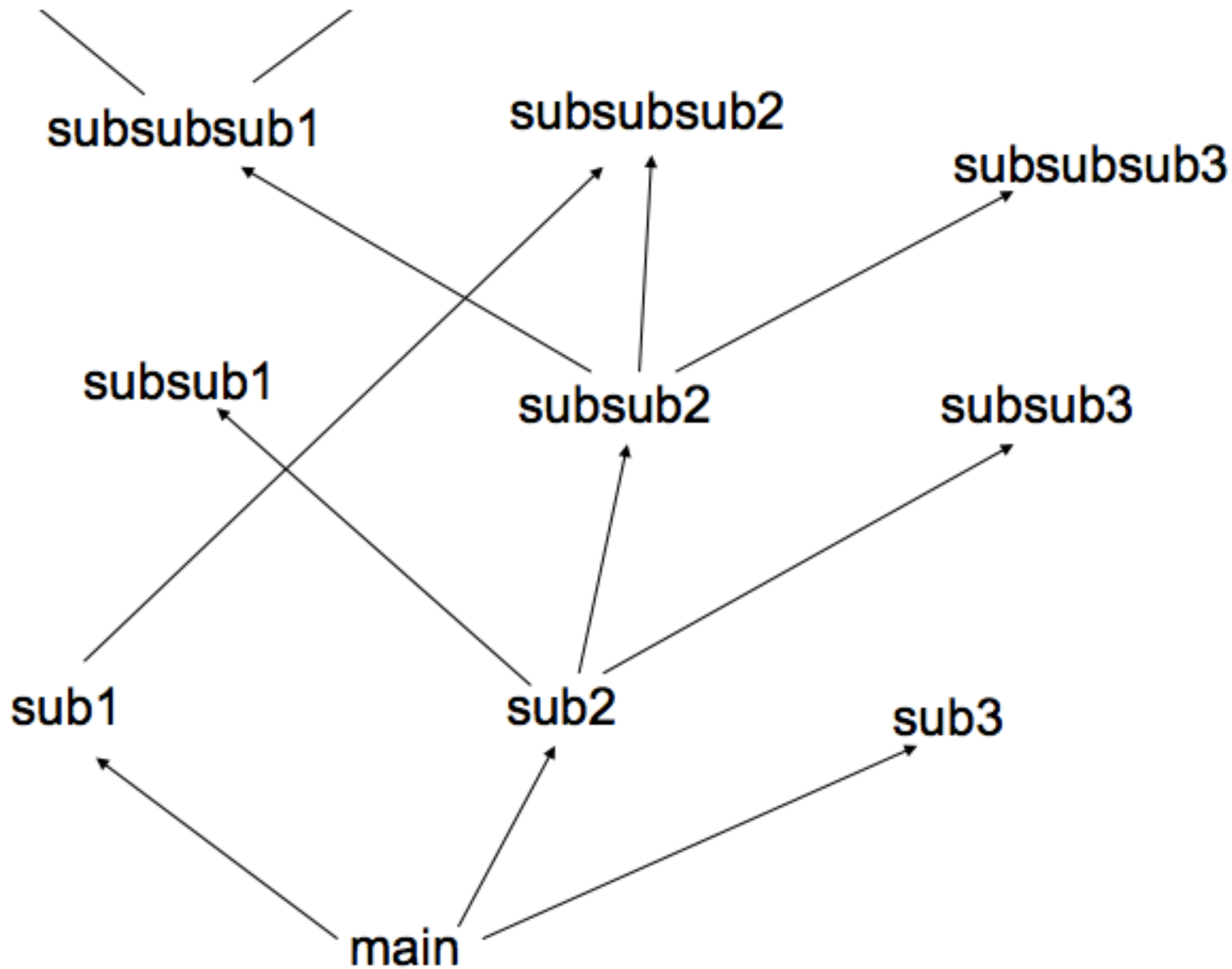
Why deep architecture?

- The brain has a deep architecture:
- Cognitive processes seem deep
  - Humans organize their ideas and concepts hierarchically.
  - Humans first learn simpler concepts and then compose them to represent more abstract ones.
  - Engineers break-up solutions into multiple levels of abstraction and processing
- Cognitive processes seem deep

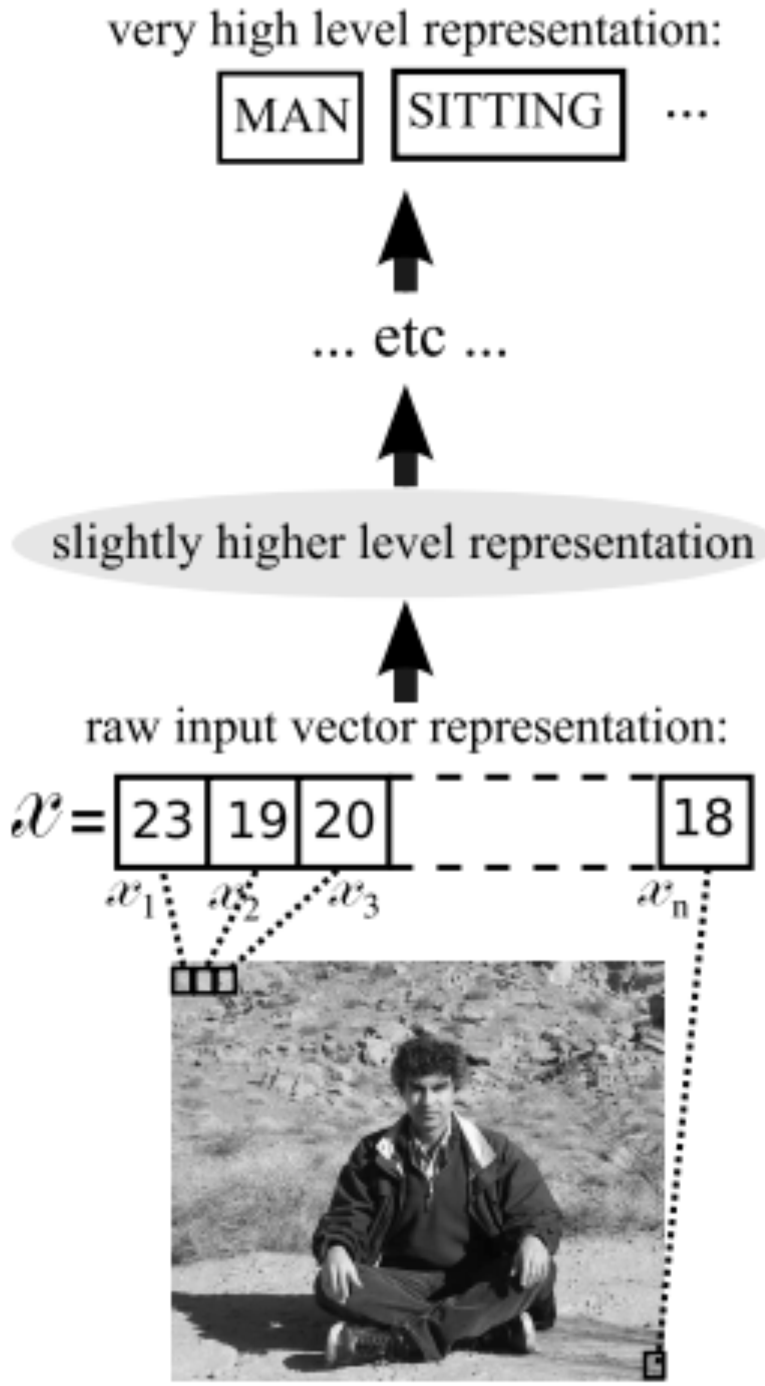
Why deep architecture?



Why deep architecture?



Why deep architecture?



Why deep architecture?

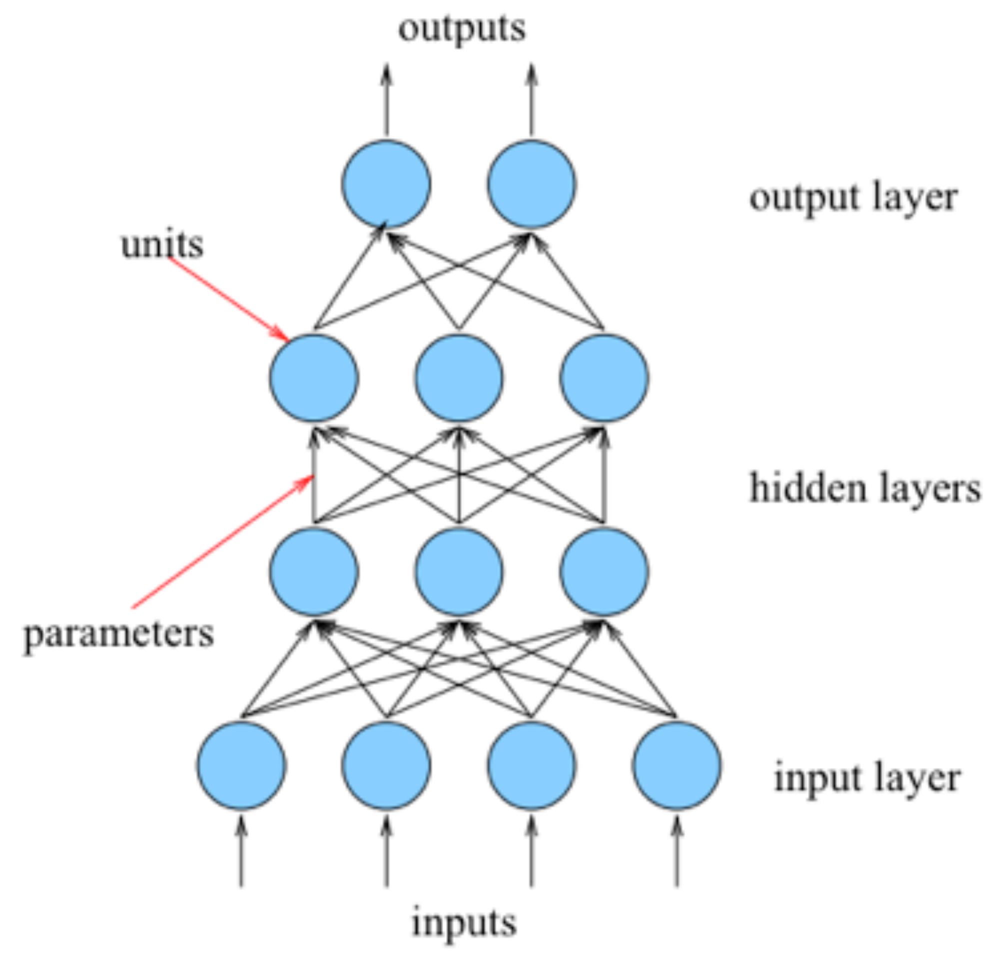
- The brain has a deep architecture:
- Cognitive processes seem deep
- Insufficient depth can hurt
  - there exist function families which the required number of nodes may grow exponentially with the input size [Hastad 1986]

Why deep architecture?

- Some families of functions which can be efficiently (compactly) represented with O(n) nodes (for n inputs) for **depth d**
- but for which an exponential number (O(2^n)) of nodes is needed if **depth is restricted to d-1**

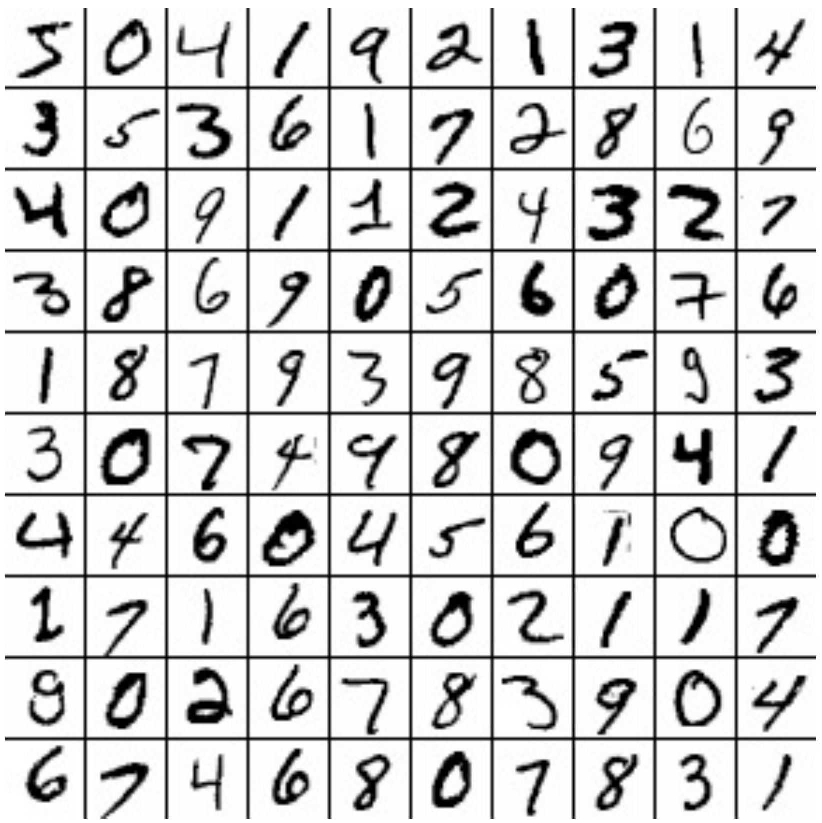
DNN

- It is hard to effectively train a deep ANN



MNIST Corpus

- 28x28 pixels, pixel values range from 0 to 1
- Contains 70,000 images
  - 50,000 training set
  - 10,000 validation set
  - 10,000 test set
- Task: Classify 10 digit classes

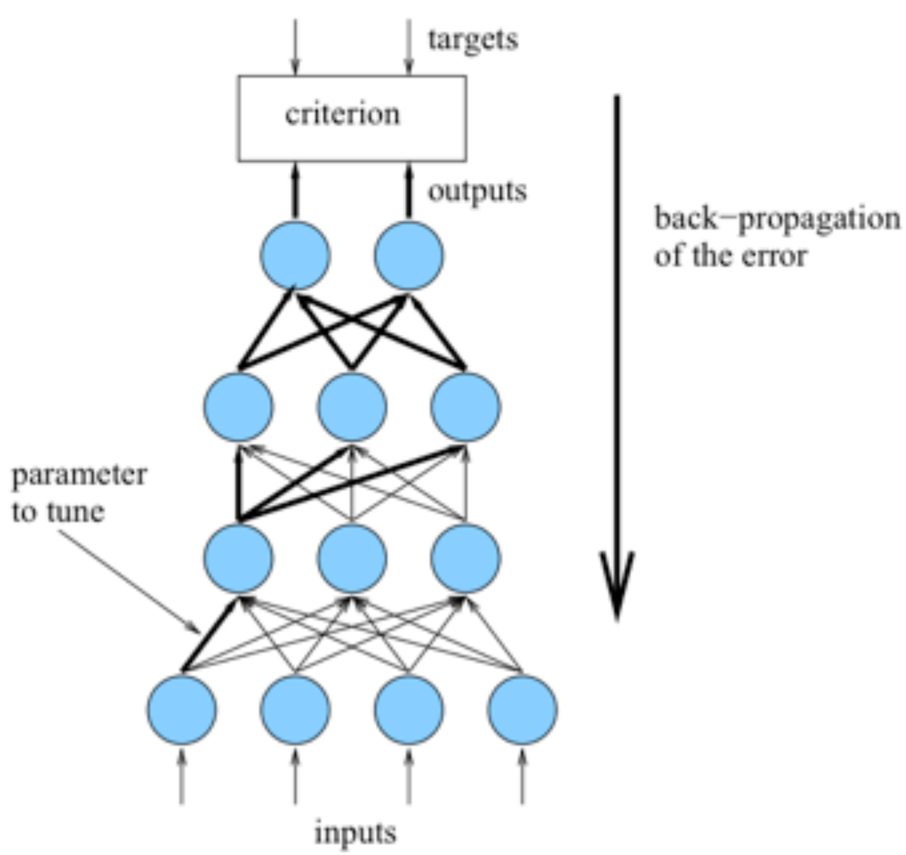


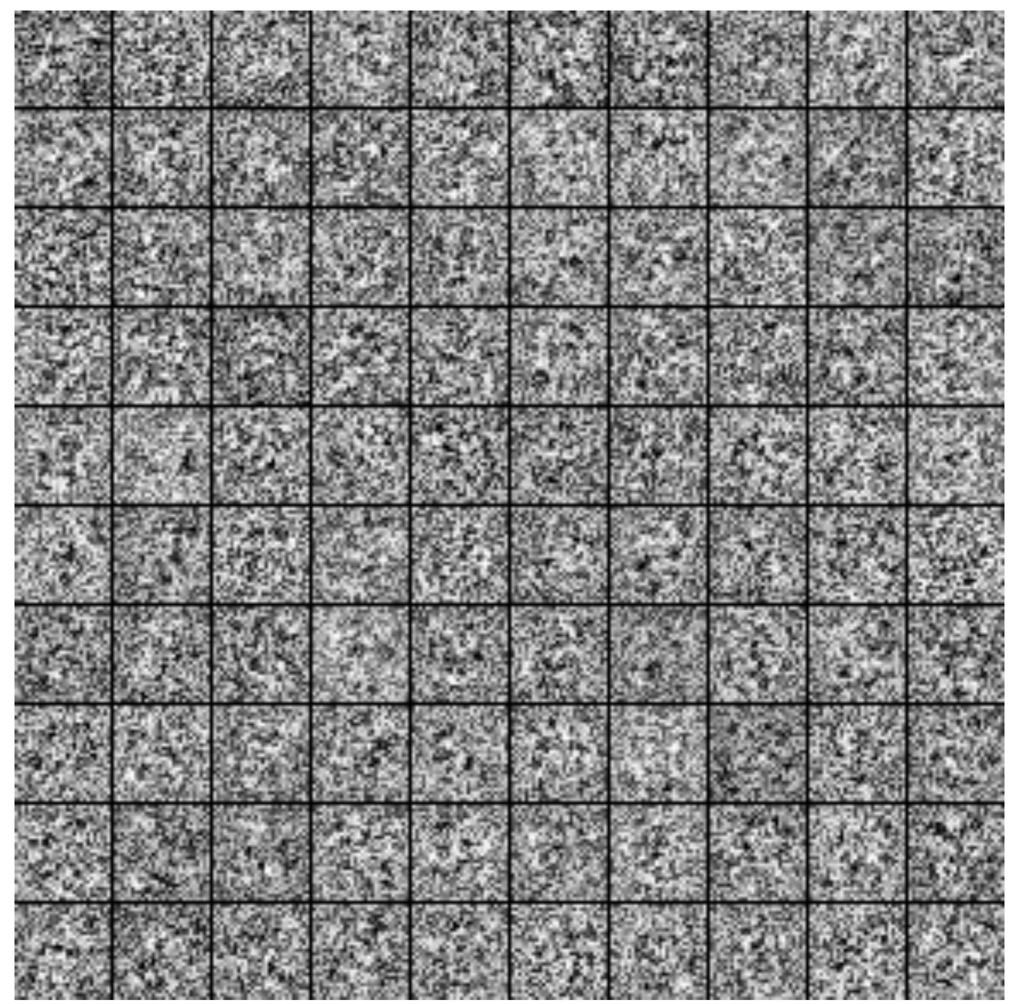
MNIST Corpus

Convolutional net Boosted LeNet-4, [distortions]	none	0.7
Trainable feature extractor + SVMs [no distortions]	none	0.83
Trainable feature extractor + SVMs [elastic distortions]	none	0.56
Trainable feature extractor + SVMs [affine distortions]	none	0.54
unsupervised sparse features + SVM, [no distortions]	none	0.59
Convolutional net, cross-entropy [affine distortions]	none	0.6
Convolutional net, cross-entropy [elastic distortions]	none	0.4
large conv. net, random features [no distortions]	none	0.89
large conv. net, unsup features [no distortions]	none	0.62
large conv. net, unsup pretraining [no distortions]	none	0.60
large conv. net, unsup pretraining [elastic distortions]	none	0.39
large conv. net, unsup pretraining [no distortions]	none	0.53
large/deep conv. net, 1-20-40-60-80-100-120-120-10 [elastic distortions]	none	0.35
committee of 7 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.27 +-0.02
committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.23

Weight decay

- What weights might look like this if DNN is trained using simple back-propagation





ANN

- The simple backprop would either
  - get stuck in local minima and give bad results or
  - it might give better results but the weights are hard to describe (how does it work?)

DNN training

- How to train a DNN effectively?

DNN training

- How to train a DNN effectively?
  - First breakthrough: Unsupervised pre-training
  - Huge amounts of data: requires high computation power. Lots of work on GPUs
  - New structures: activation functions like ReLU and maxout, other structures like CNNs and RNNS
  - Clever training: dropout

DNN training

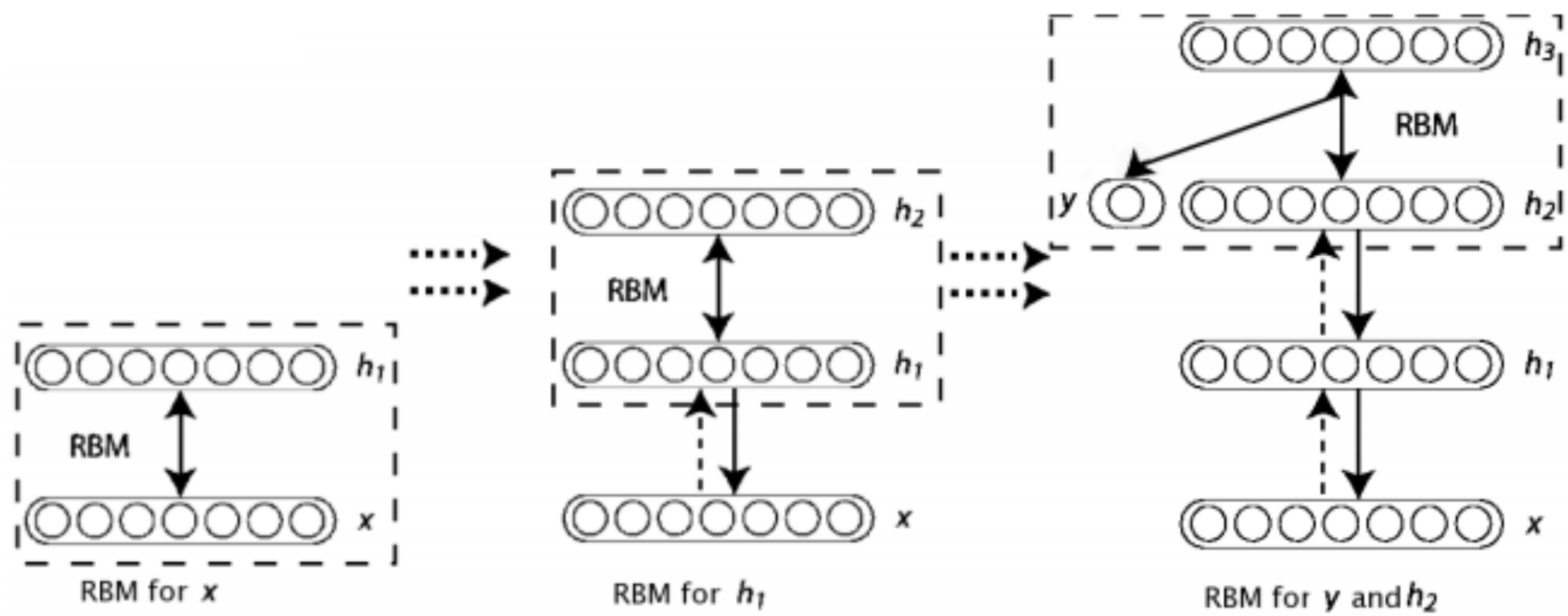
- How to train a DNN effectively?
  - First breakthrough: Unsupervised pre-training
  - Huge amounts of data: requires high computation power. Lots of work on GPUs
  - New structures: activation functions like ReLU and maxout, other structures like CNNs and RNNS
  - Clever training: dropout

Unsupervised Pre-training

- previous purely supervised attempt failed
- Unsupervised feature learners:
    - Restricted Boltzmann Machines
    - Auto-encoder variants
    - Sparse coding variants

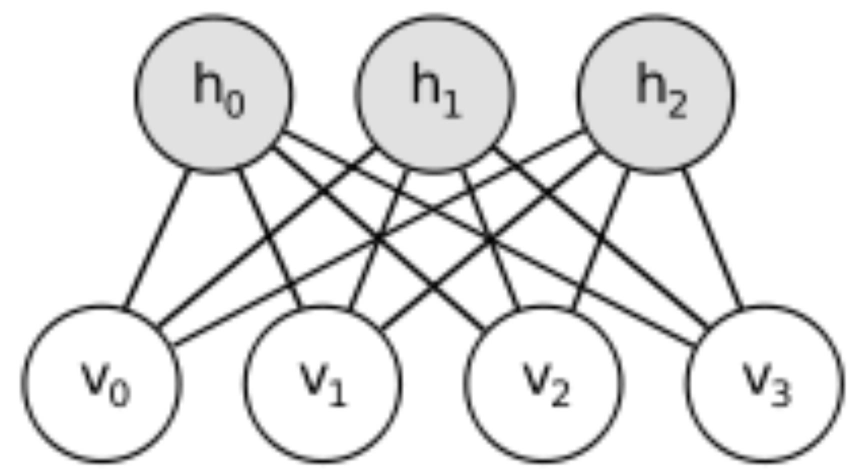
Unsupervised Pre-training

- One of the big ideas from 2006: **layer-wise** unsupervised ore-training



Unsupervised Pre-training

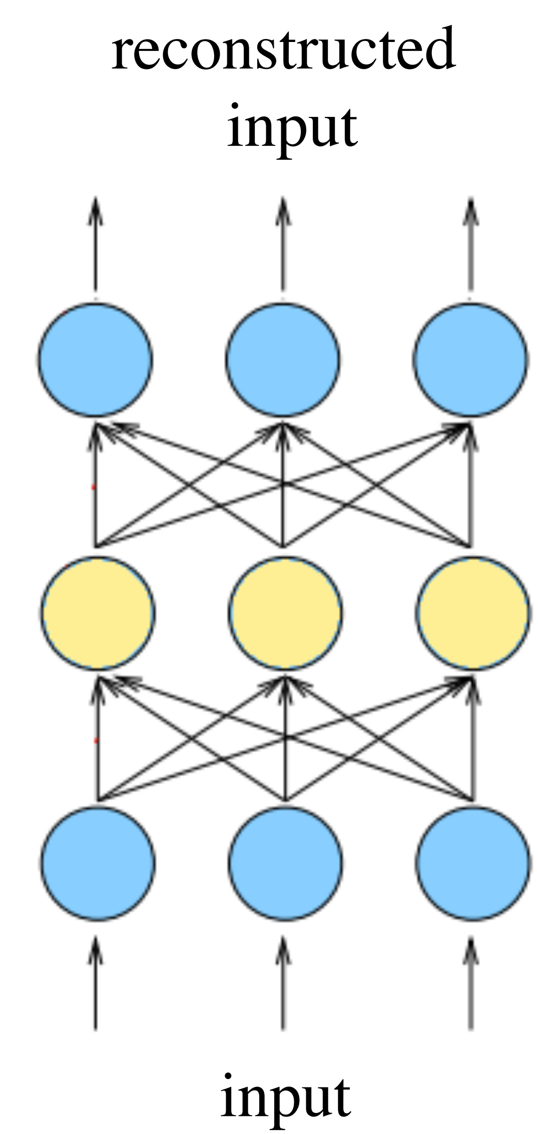
- RBMs
- `h=g(Wv+b)`
- `v=g(W'h+c)`



- RBMs are Energy-based models trained to maximize the energy

Autoencoders

- These networks try to reconstruct the input
- `h=g(Wx+b1)`
- `x_rec=g(W'h+b2)`
- where the first and second layer weights are tied `W'=transpose(W)`



Autoencoders vs RBMs

- RBMs and denoising autoencoders (DAE) have shown to converge to the same solution under certain conditions

Unsupervised Pre-training

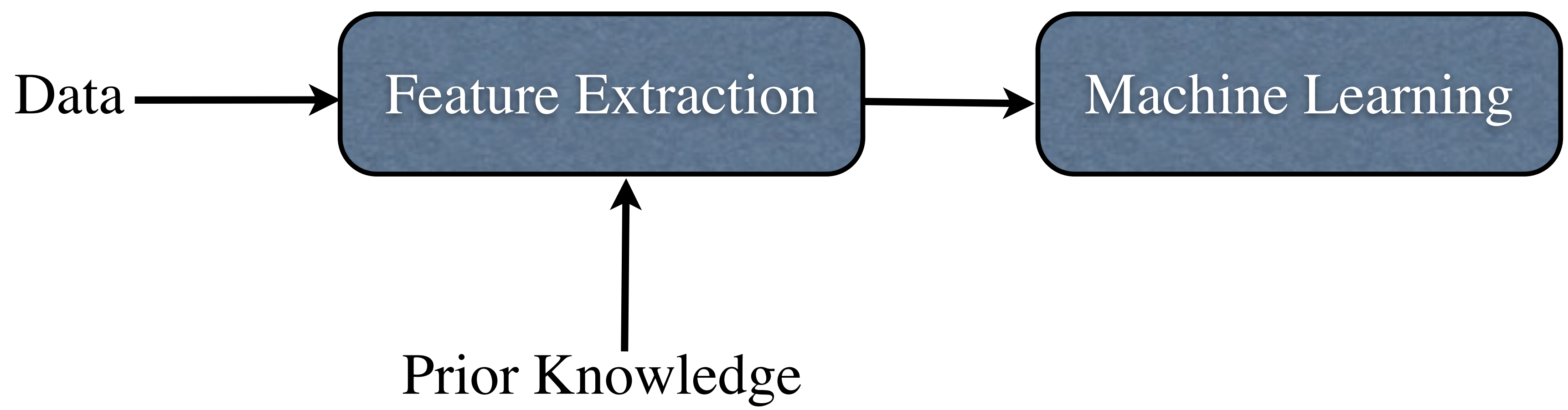
- Start training the network using backprop from the RBM or DAE weights and not initial randomization.
- Why does unsupervised pre-training work?
  - It is a form of feature learning.

Unsupervised Pre-training

- The goal of unsupervised pre-training is
  - to see a lot of unlabeled examples
  - learn features from it

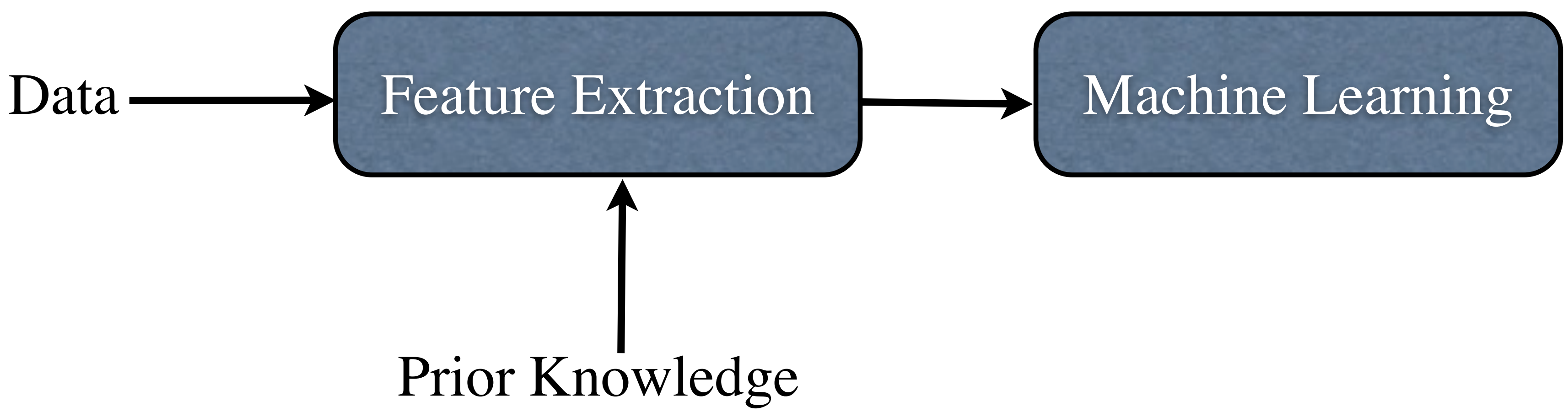
Feature Learning

- Usual Machine Learning applications have two steps



Feature Learning

- Usual Machine Learning applications have two steps

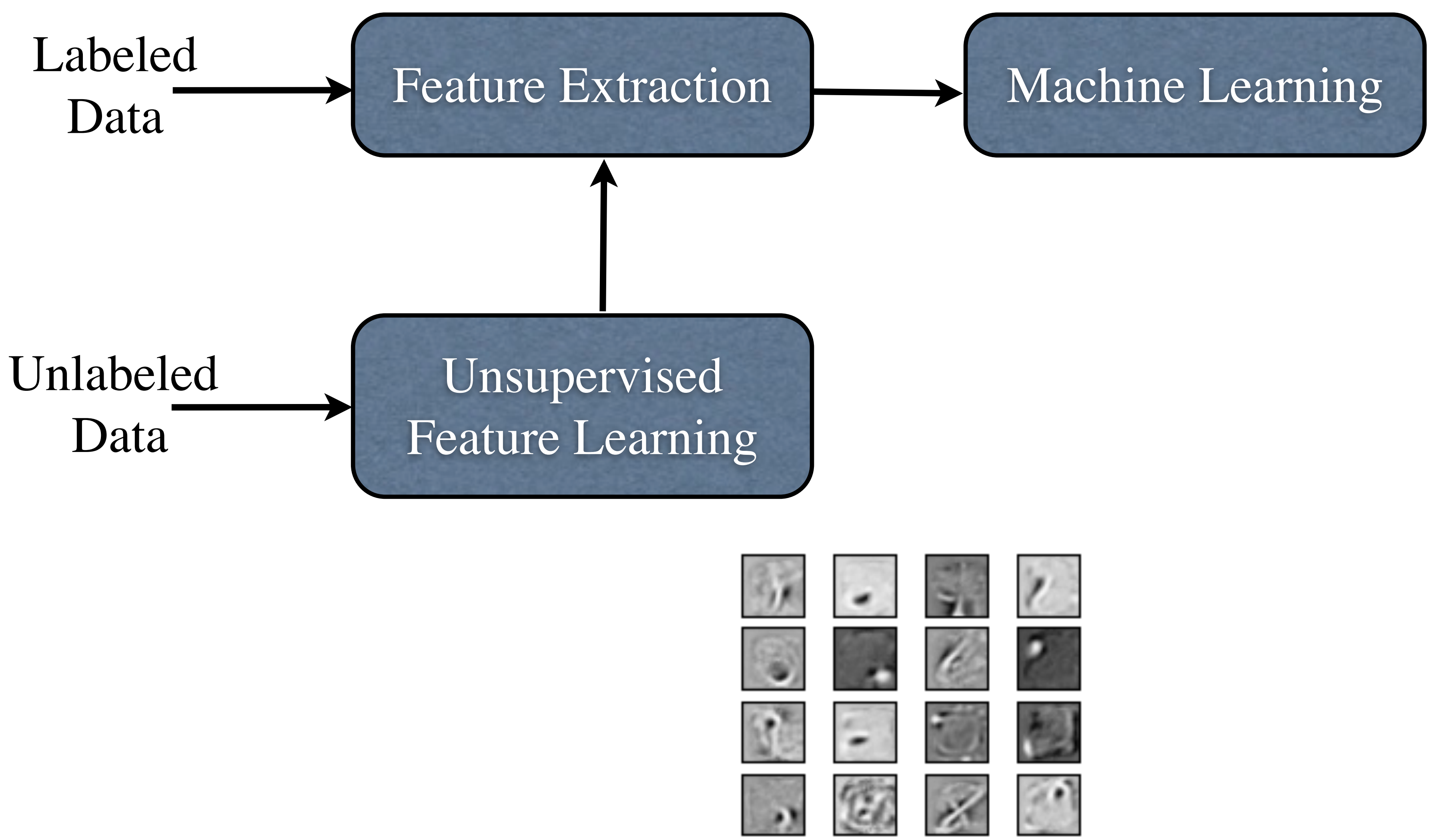


how many straight vertical lines? how long?

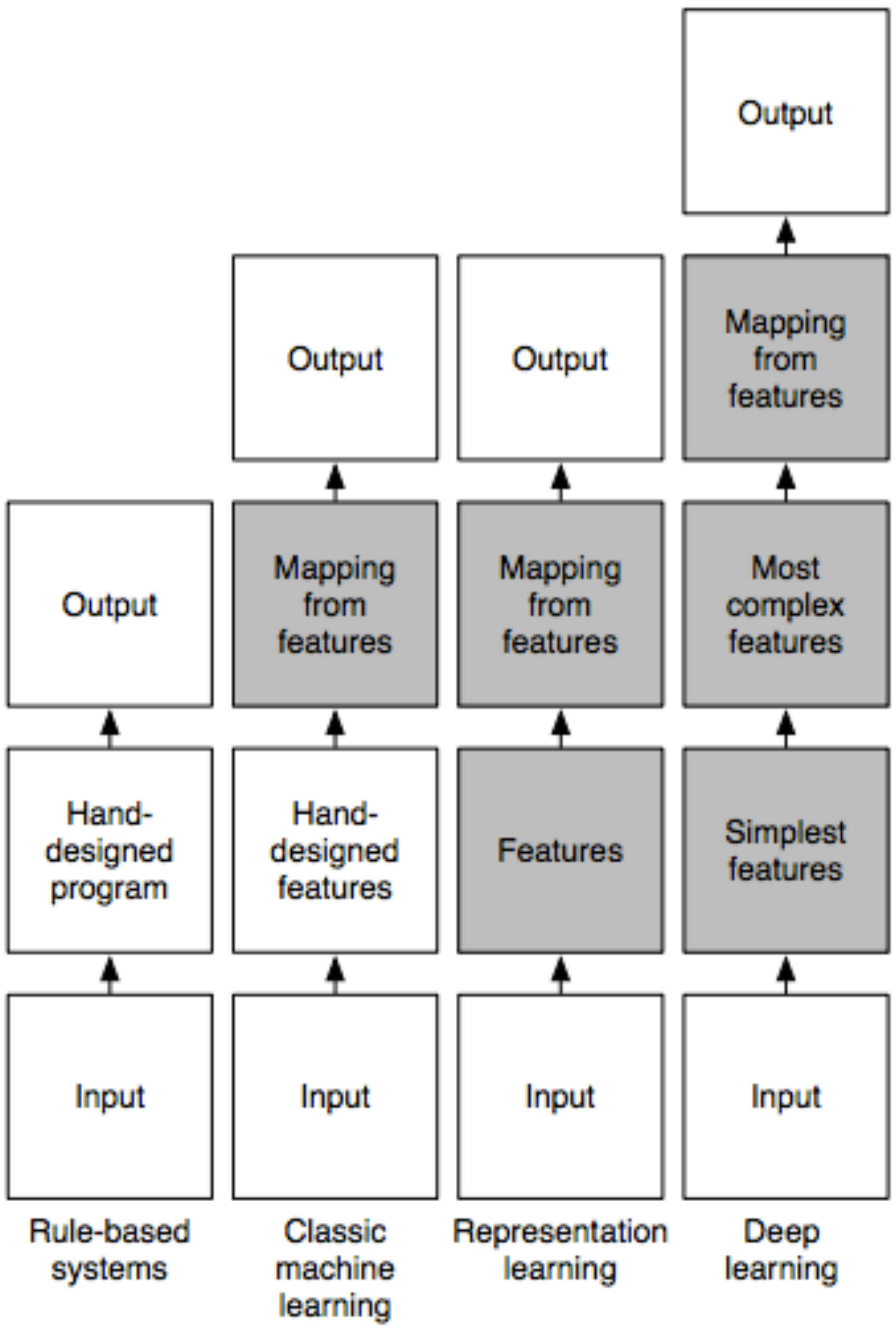
how many circles?

how many straight horizontal lines? how long?

Feature Learning

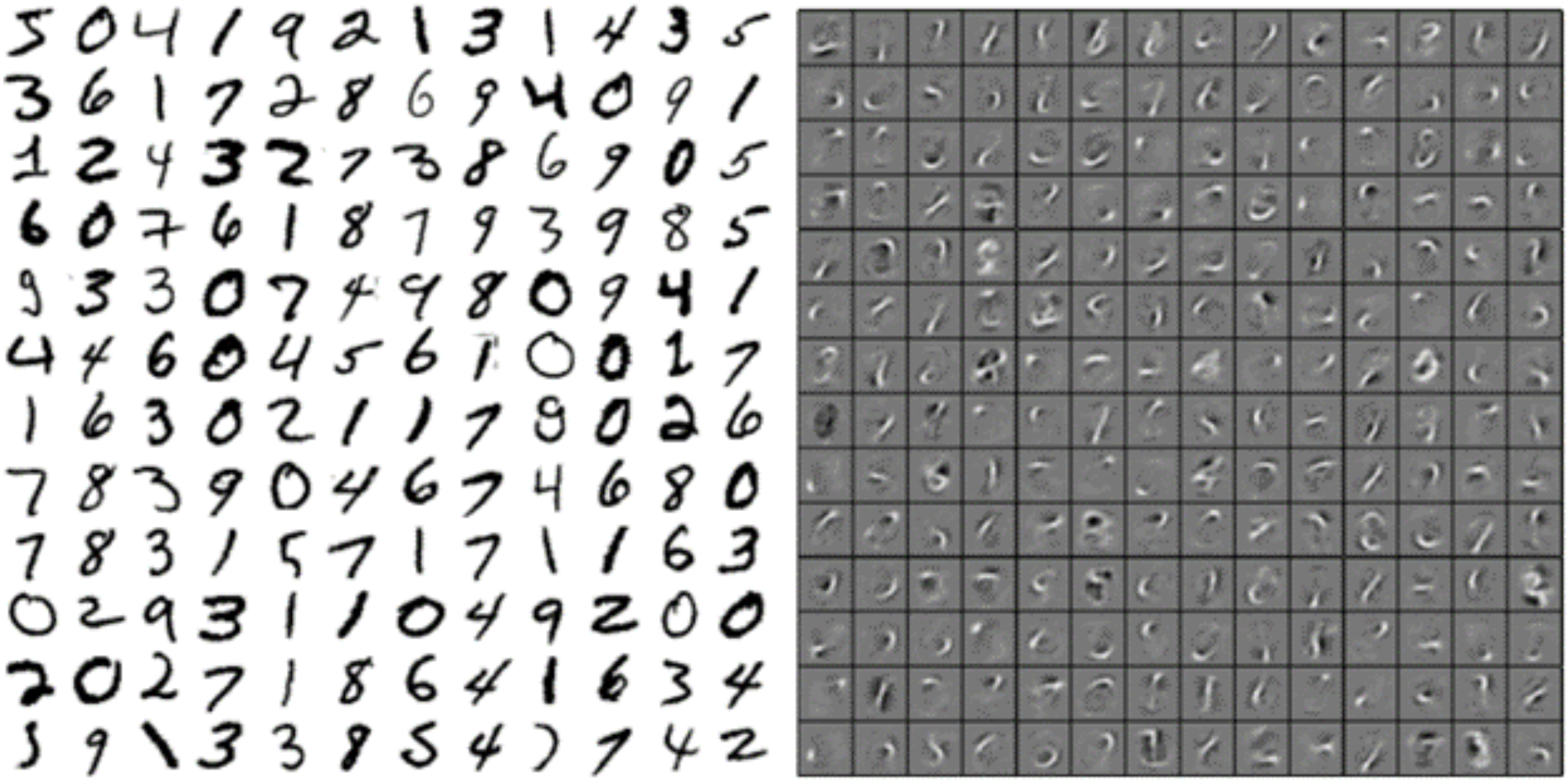


Feature learning

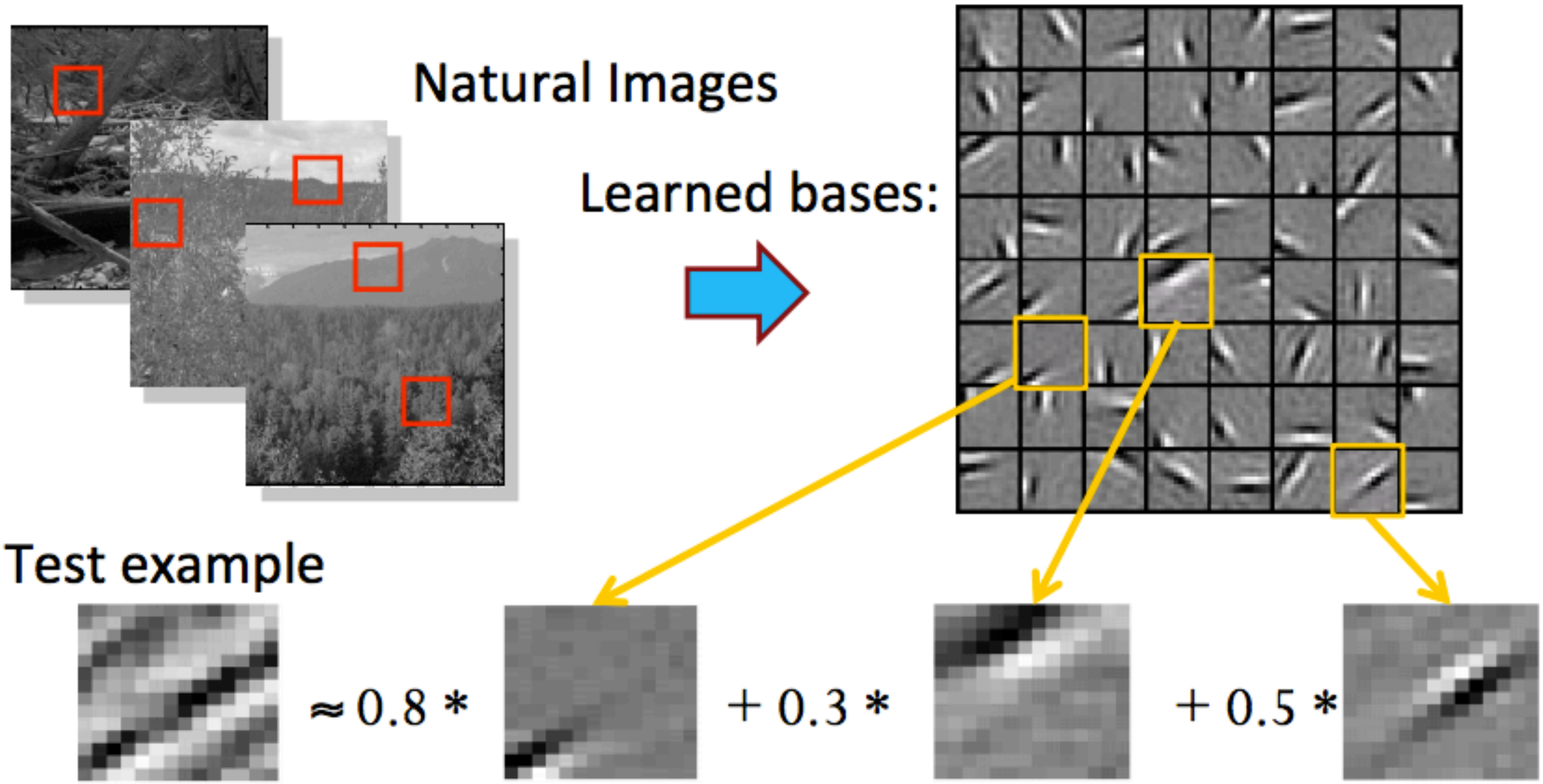


Edge detection

Unsupervised Feature Learning



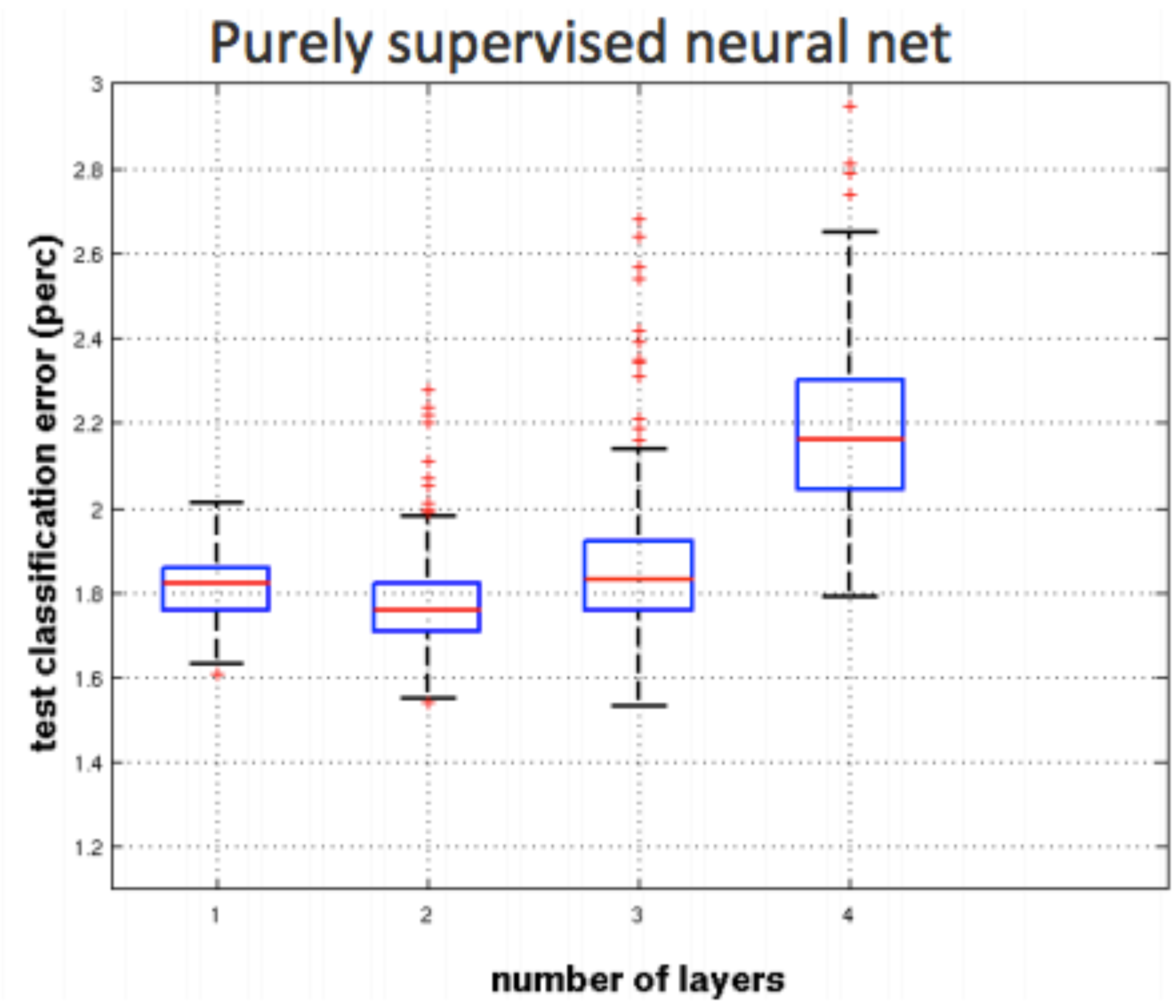
Sparse Coding

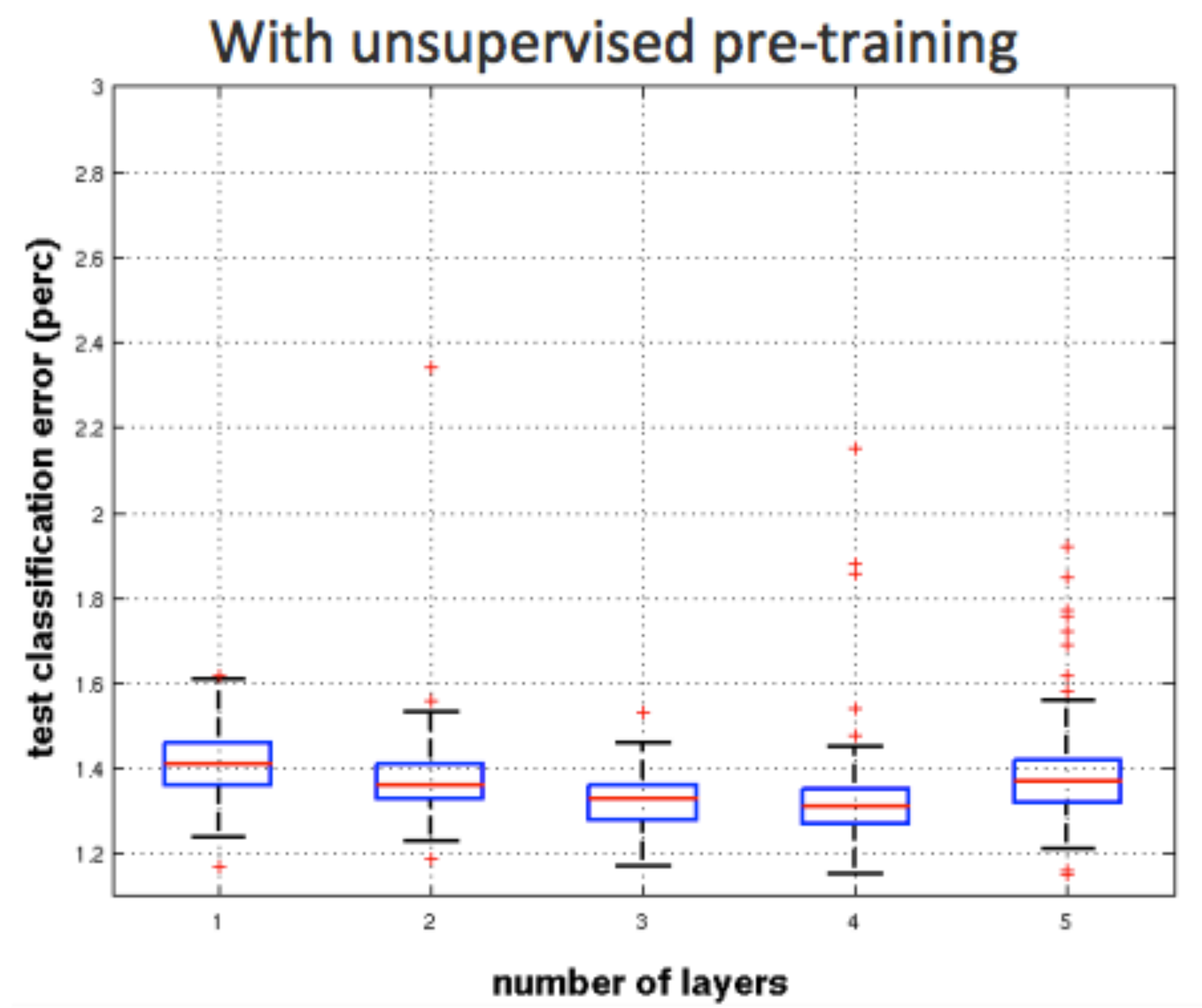


$[h_1, \ldots, h_{64}]$ = [0, 0, ..., 0, **0.8**, 0, ..., 0, **0.3**, 0, ..., 0, **0.5**, 0]
(feature representation)

Results

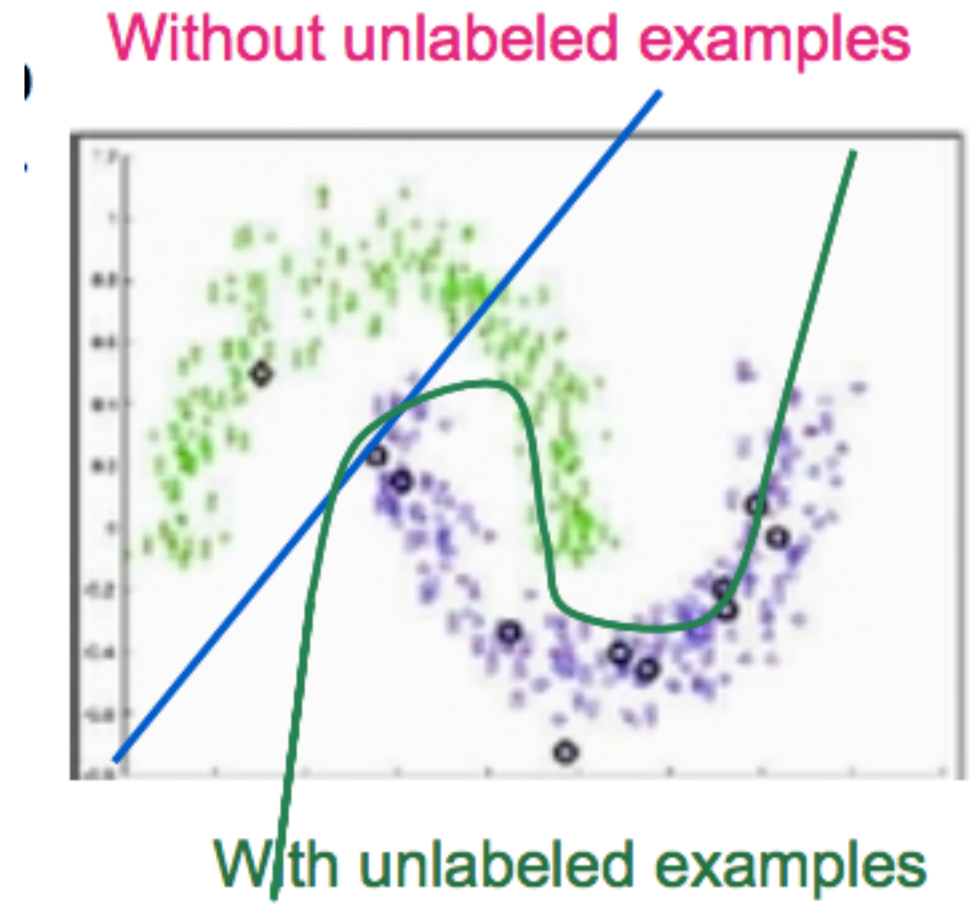
- MNIST results





Unsupervised

- Recently, with enough data, and advances in the field, it has been shown that unsupervised learning is not always necessary
- but helps if:
    - data size is small
    - low computation power



DNN training

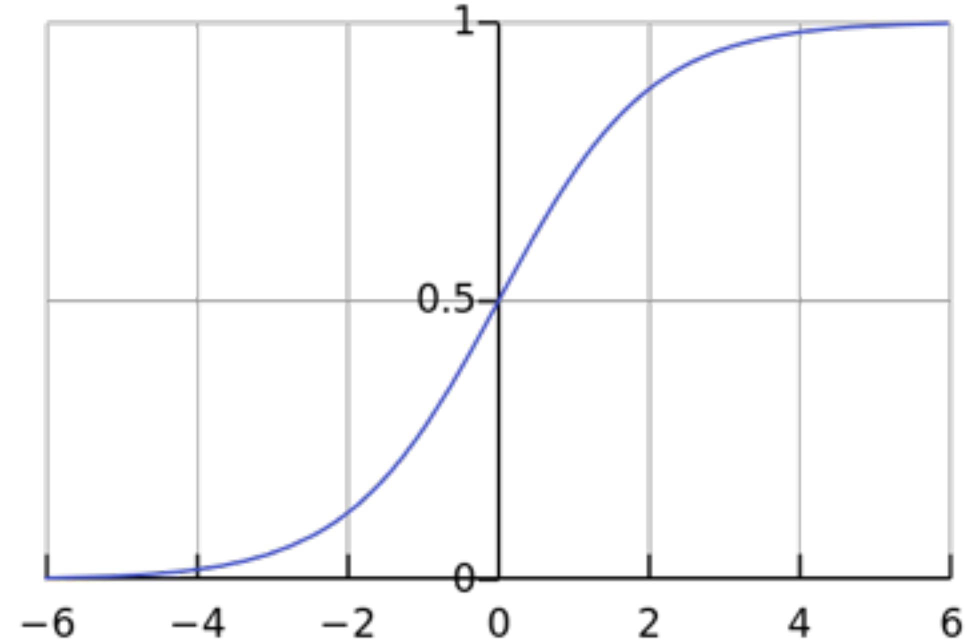
- How to train a DNN effectively?
  - First breakthrough: Unsupervised pre-training
  - Huge amounts of data: requires high computation power. Lots of work on GPUs
  - New structures: activation functions like ReLU and maxout, other structures like CNNs and RNNS
  - Clever training: dropout

DNN training

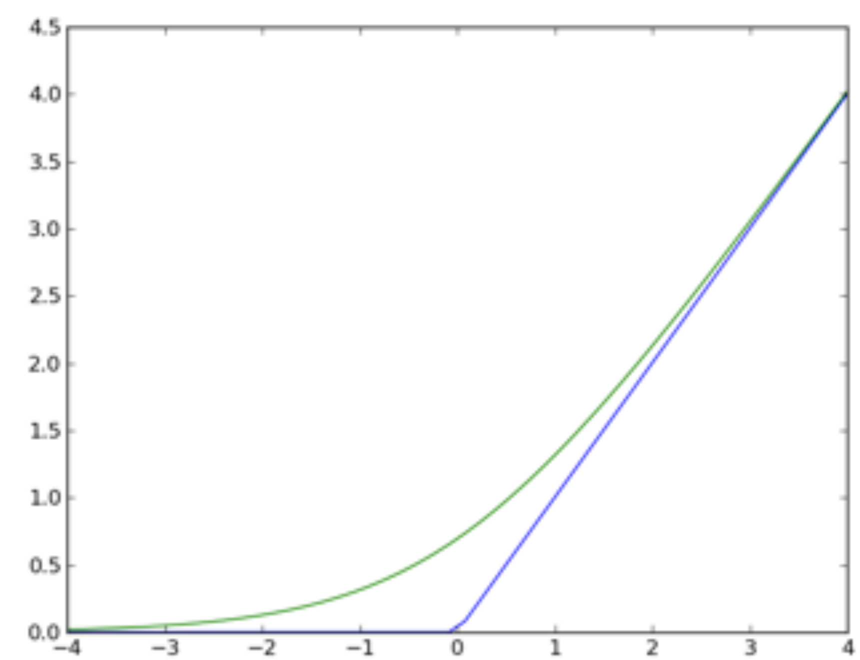
- How to train a DNN effectively?
  - First breakthrough: Unsupervised pre-training
  - Huge amounts of data: requires high computation power. Lots of work on GPUs
  - New structures: activation functions like ReLU and maxout, other structures like CNNs and RNNS
  - Clever training: dropout

Activation functions

- Old style ones:
  - Sigmoid
  - Tanh



- Rectifier f(x) = max(0, x)
- Softplus f(x) = ln(1 + e^x)

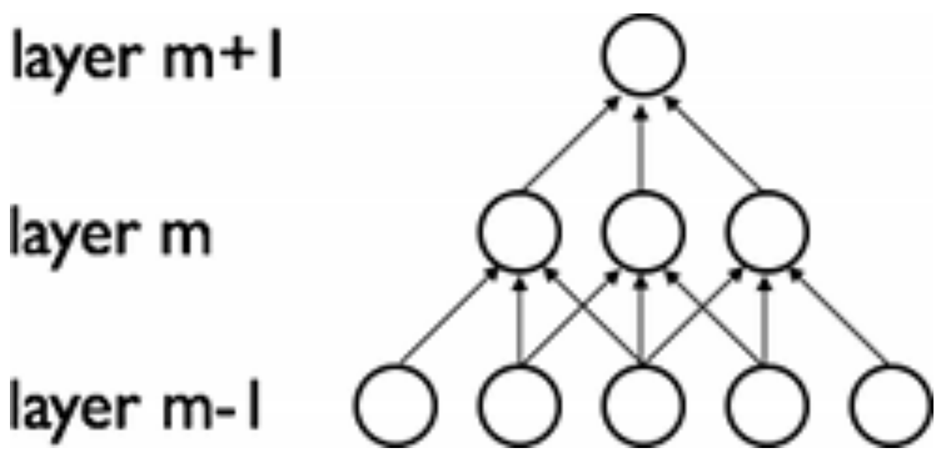


Convolutional NN

- Convolutional Neural Networks (CNN) are biologically-inspired variants of MLPs.
- Mimic visual cortex cell arrangemens
- exploit the strong spatially local correlation present in natural images

Convolutional NN

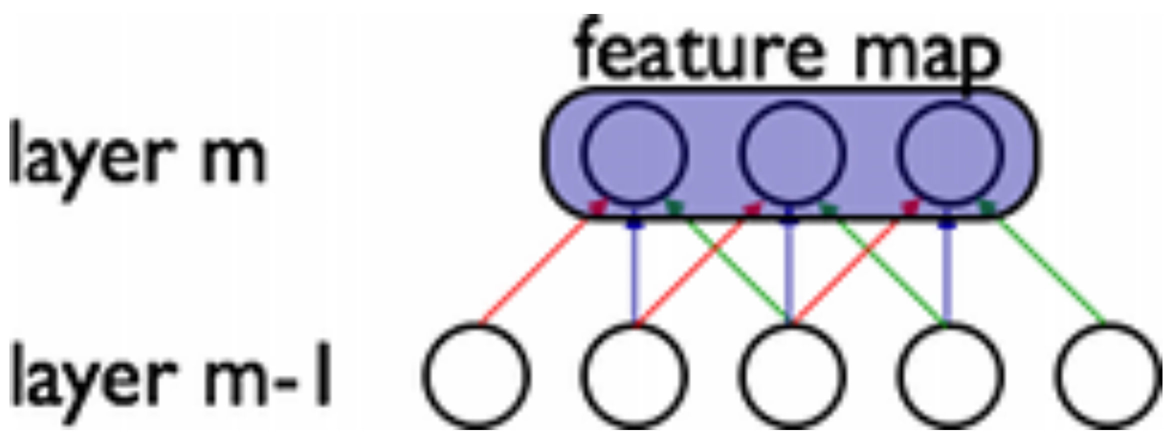
- Sparse Connectivity



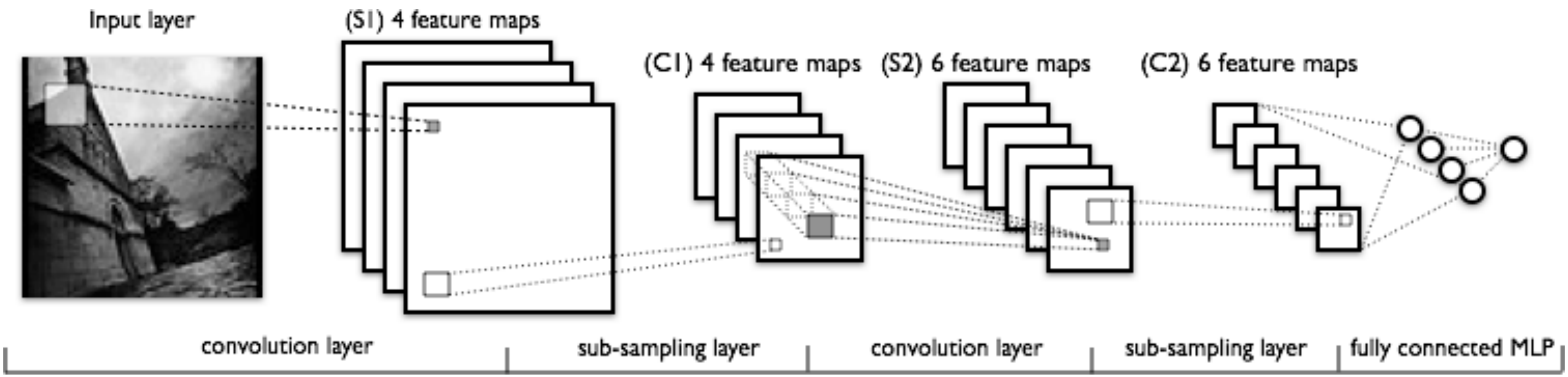
-

Convolutional NN

- Weight sharing

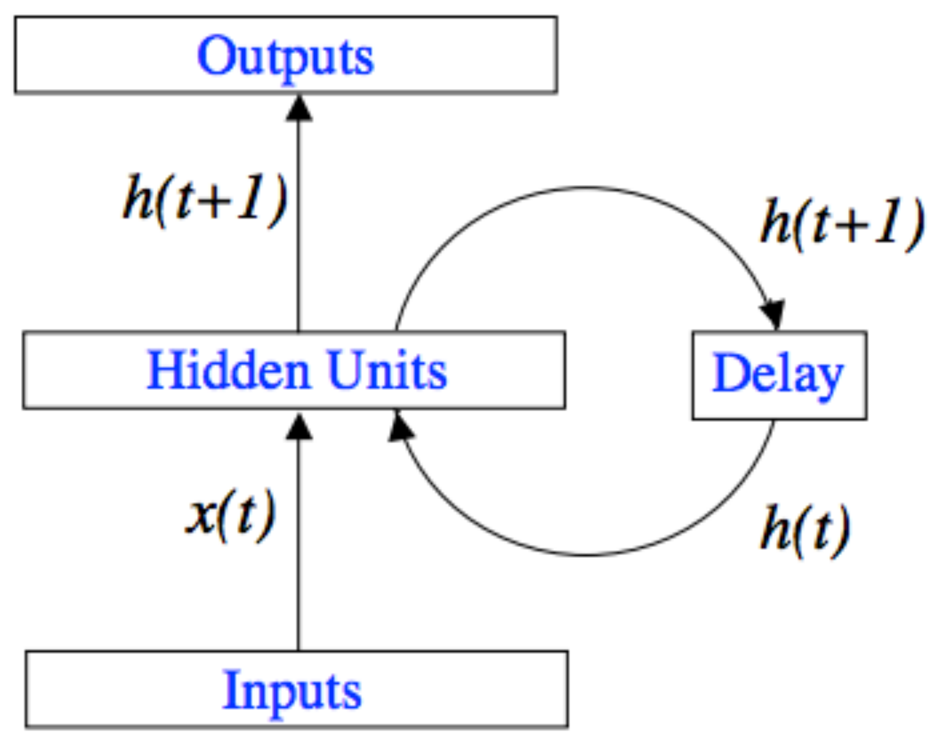


Deep CNNs



Input layer
(S1) 4 feature maps
(C1) 4 feature maps
(S2) 6 feature maps
(C2) 6 feature maps
convolution layer
sub-sampling layer
convolution layer
sub-sampling layer
fully connected MLP

Recurrent NNs



Application specific

- CNNs have shown significant improvements for vision field
- RNNs have shown significant improvements for speech field
- New machine learning: less signal processing (feature engineering) and more model engineering

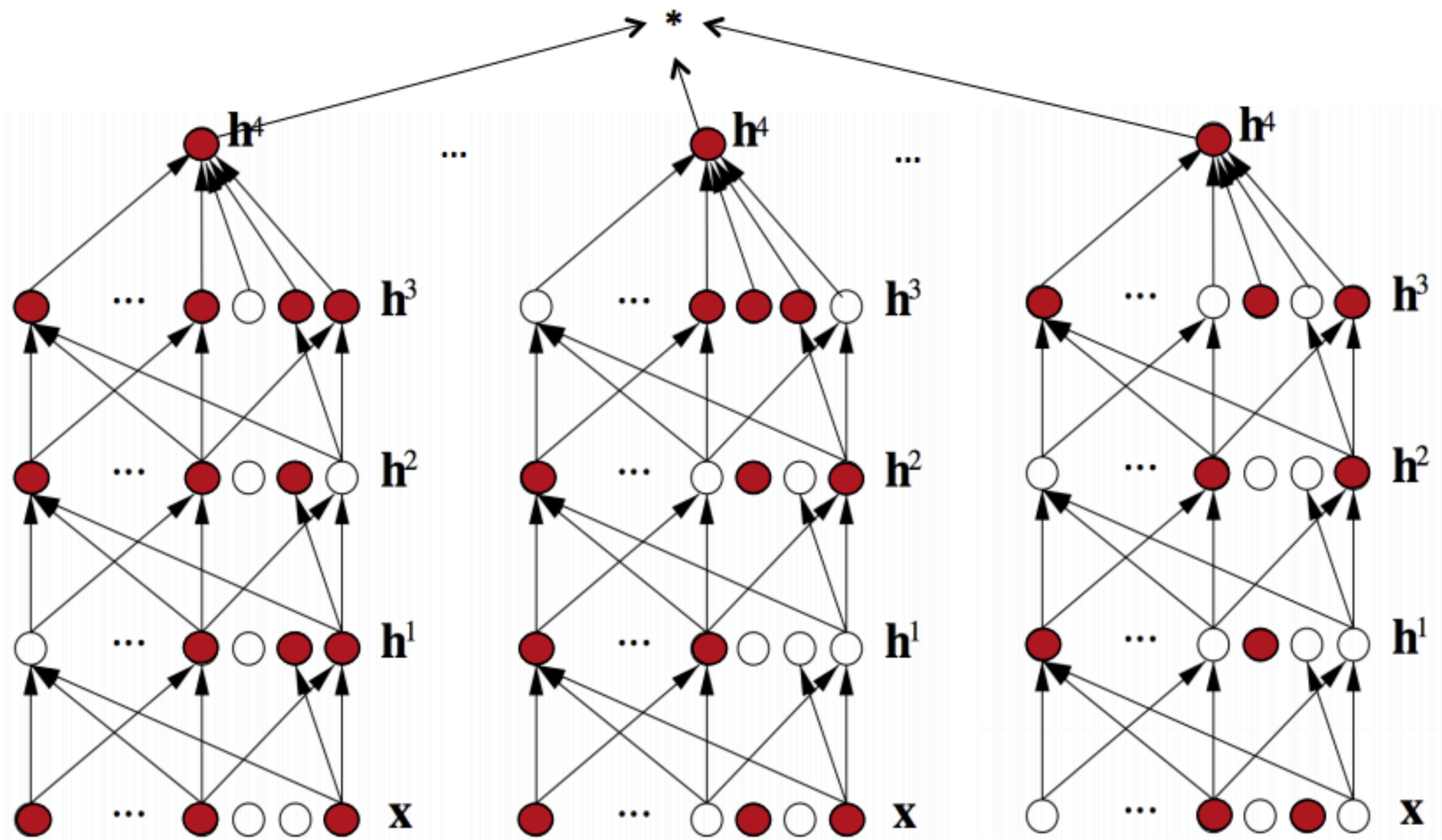
DNN training

- How to train a DNN effectively?
  - First breakthrough: Unsupervised pre-training
  - Huge amounts of data: requires high computation power. Lots of work on GPUs
  - New structures: activation functions like ReLU and maxout, other structures like CNNs and RNNS
  - Clever training: dropout

Dropout

- during training multiply neuron output by random 0/1 bit (p=0.5),
- during test weight by 0.5 to adjust
- works very good with ReLU and maxout

Dropout



Dropout

- New Machine Learning:
  - set the number of your parameters to more than it is actually needed
  - use clever regularizations such as dropout to avoid over-fitting

Conclusion

- How to train a DNN effectively?
  - First: Unsupervised pre-training
  - Huge amounts of data, high computation power
  - Activation functions like ReLU
  - Clever training: dropout
- The last three innovations have made unsupervised learning less necessary

Conclusion

- Designing models rather than feature engineering -> is signal processing going to be extinct?!

- Huge number of parameters (more than needed) but use regularization

References

- [1] Y. Bengio, Learning Deep Architectures for AI, 2009
- [2] Y. Bengio, Deep Learning, MLSS, 2015
- [3] http://www.iro.umontreal.ca/~pift6266/H10/notes/deepintro.html
- [4] deeplearning.net
- [5] http://www.cs.bham.ac.uk/~jxb/INC/