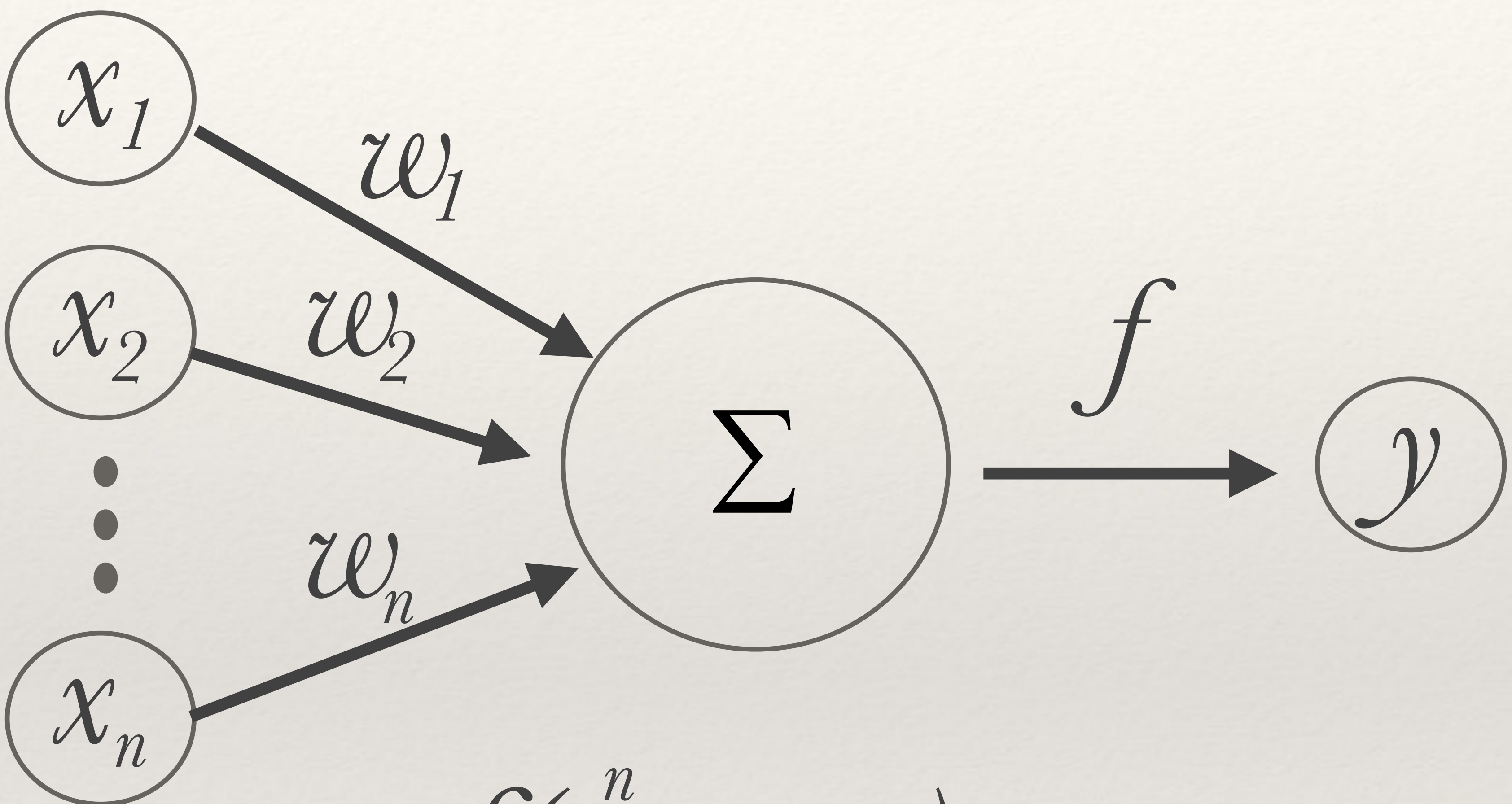
Michael Grossberg

Intro to Data Science CS59969

Neural Net and Deep Architecture

Artificial Neural Network

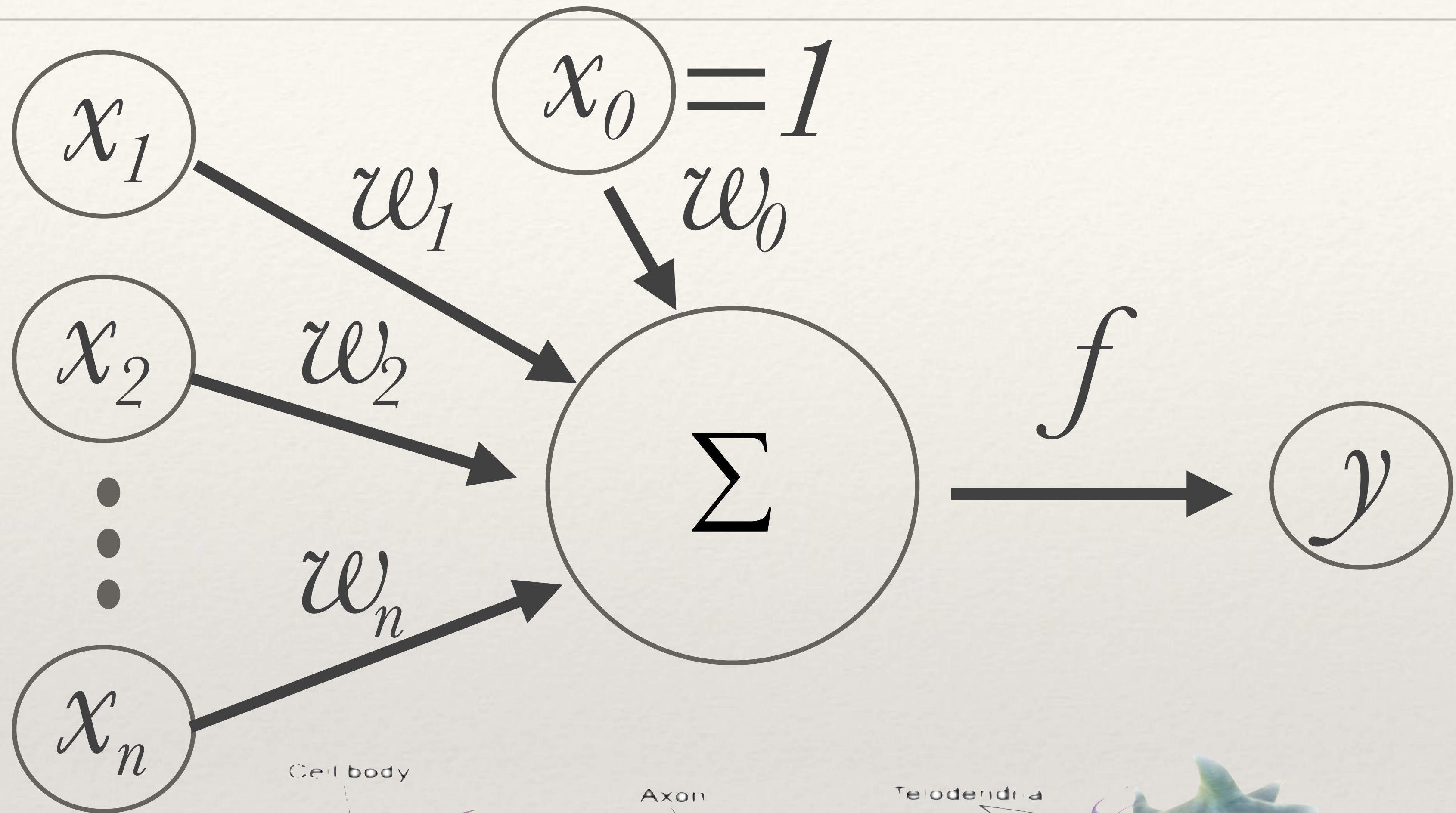
Perceptron

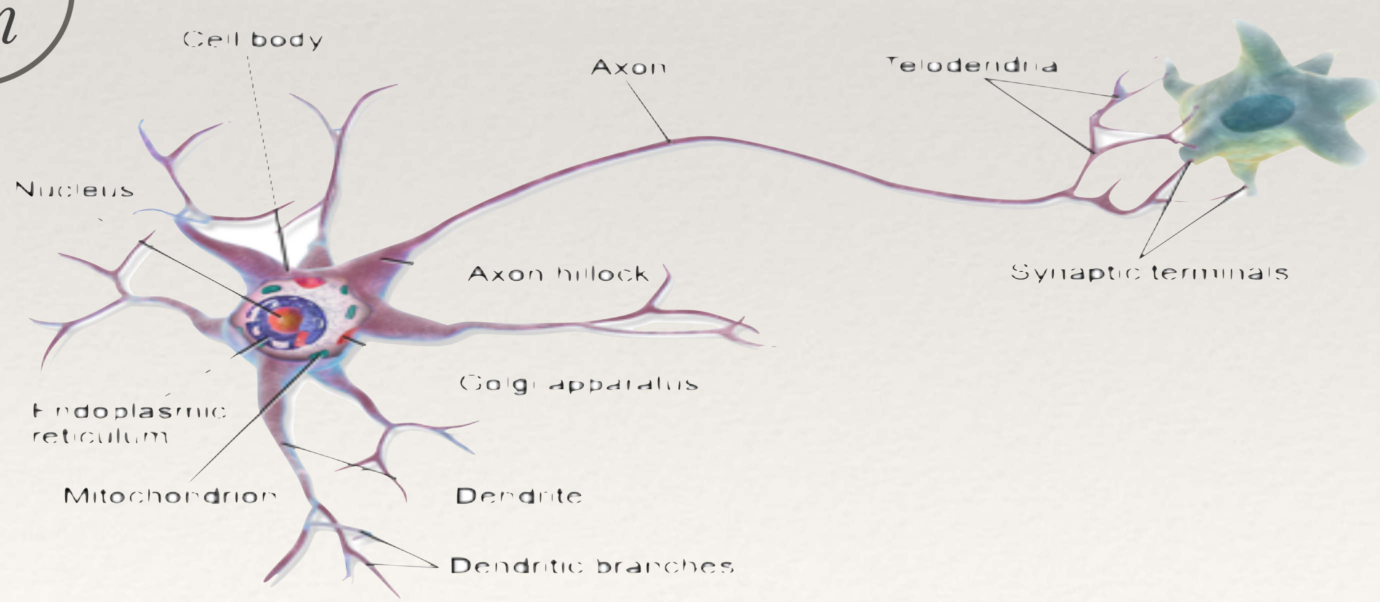


$$f\left(\sum_{i=1}^{n} w_i x_i\right) = y$$

Activation Function   Weights   Inputs   Output

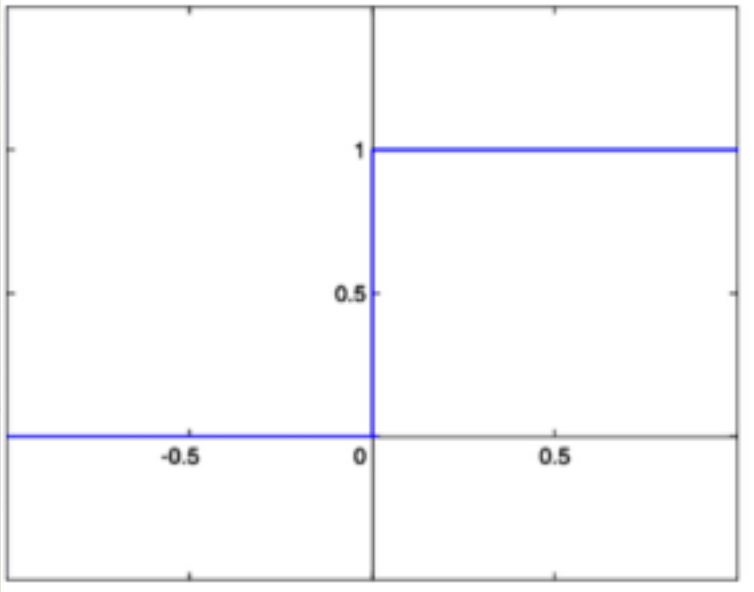
Neuron Analogy





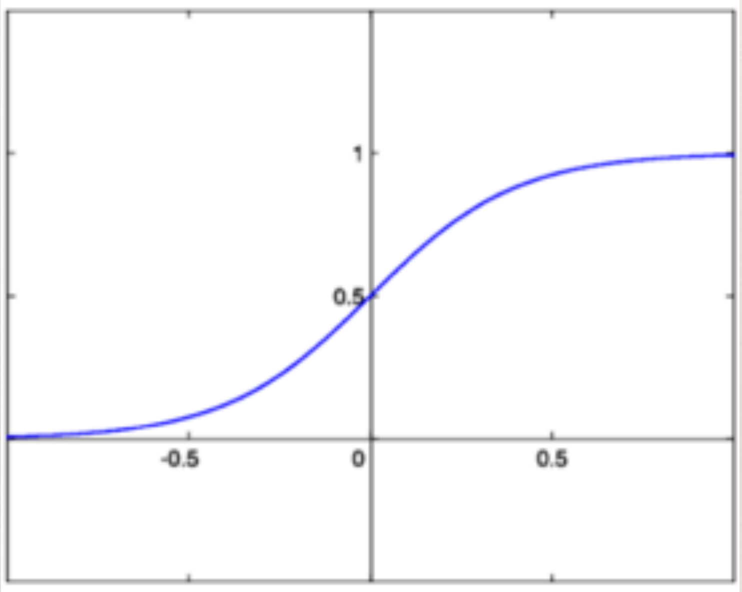
Activation Function

$$f = \begin{cases} 1 \text{ if } u \geq 0 \\ 0 \text{ if } u < 0 \end{cases} \quad \text{or} \quad = \frac{1}{1+e^{-u}}$$



Step

More intuitive

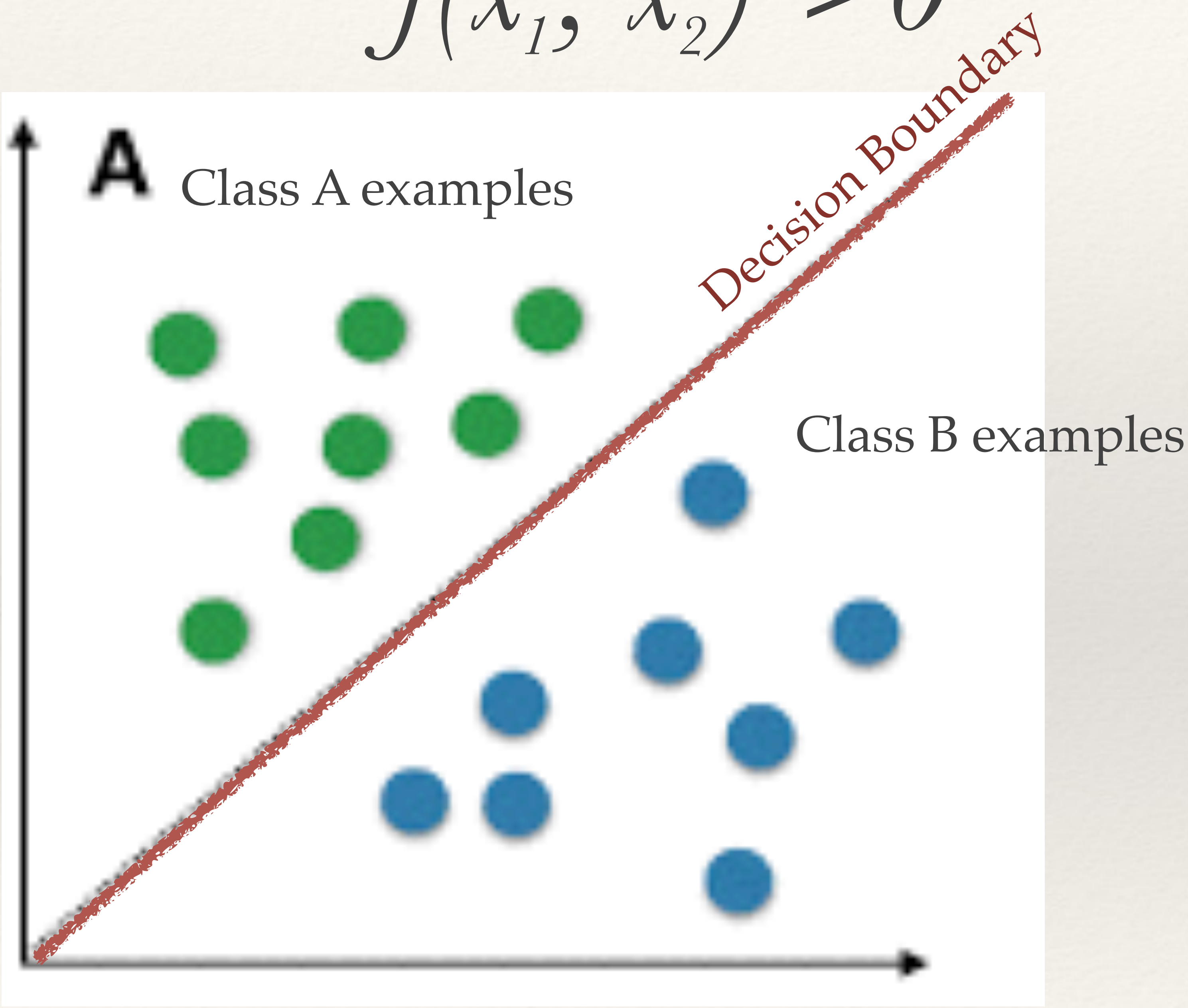


Sigmoid

Easier for optimization

Perceptron: Separating Hyperplane

$f(x_1, x_2) > 0$



$f(x_1, x_2) < 0$

Update Algorithm

f_w depends on $w_0, w_1, \ldots \; w_n$

$w_i(t)$ update for next t

$w_i(0)$ random

j th Data Point

$$f_{w(t)}(\boldsymbol{x}^j) = y^j$$

j th predicted output

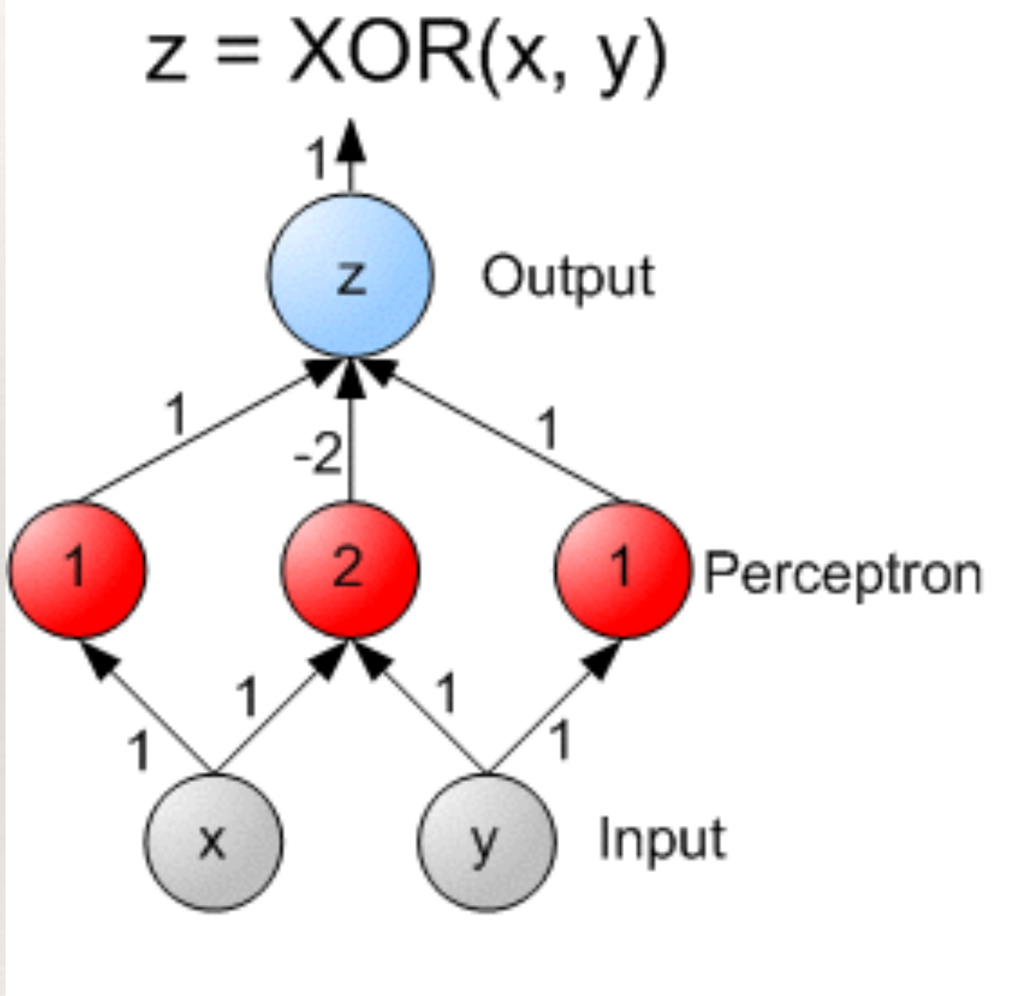
nth True Response

$$\boxed{w_i(t+1) = w_i(t) + (d^j - y_i^j) \cdot x_i^j}$$

Update Rule

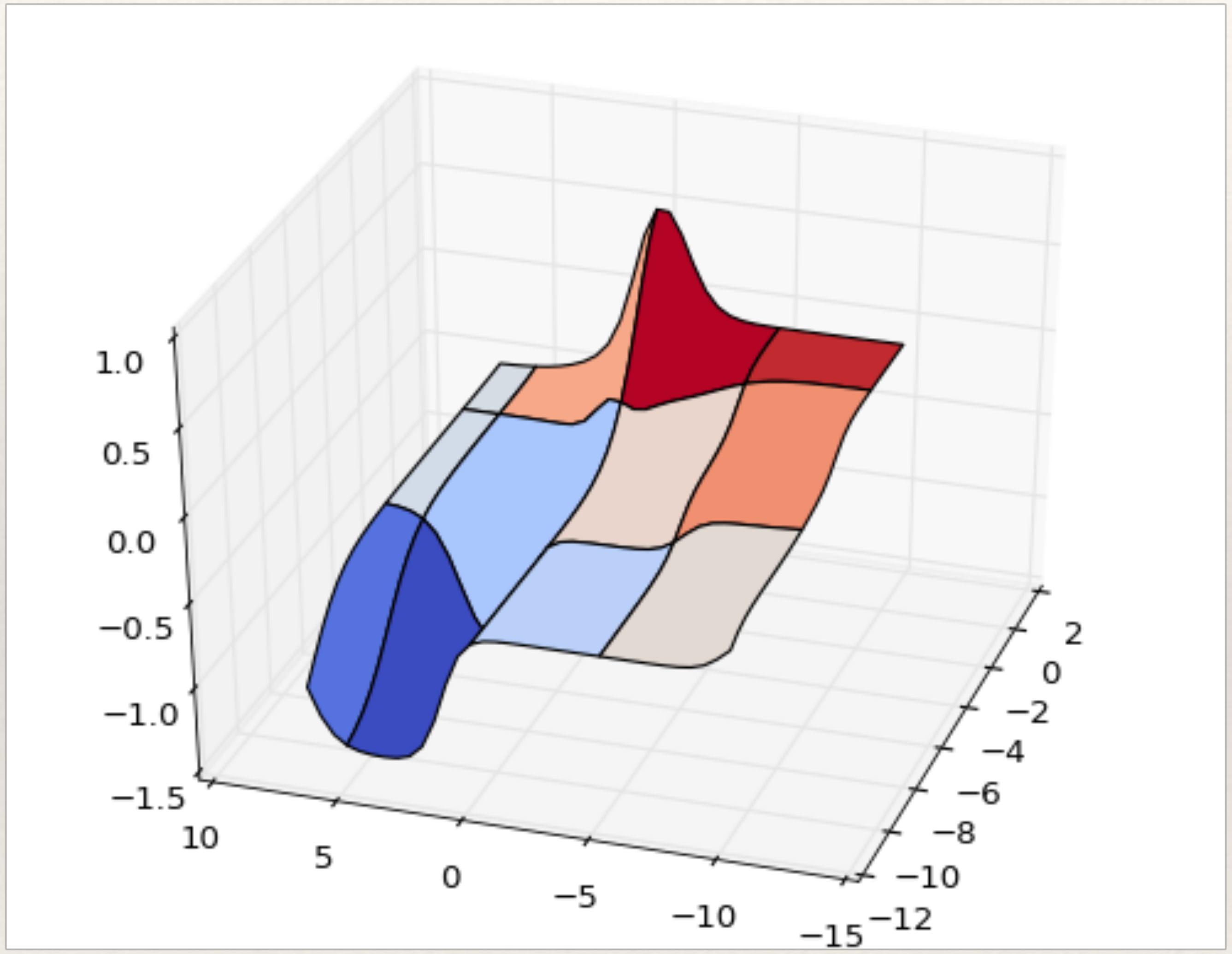
Converges …..

Multi-layer perceptron



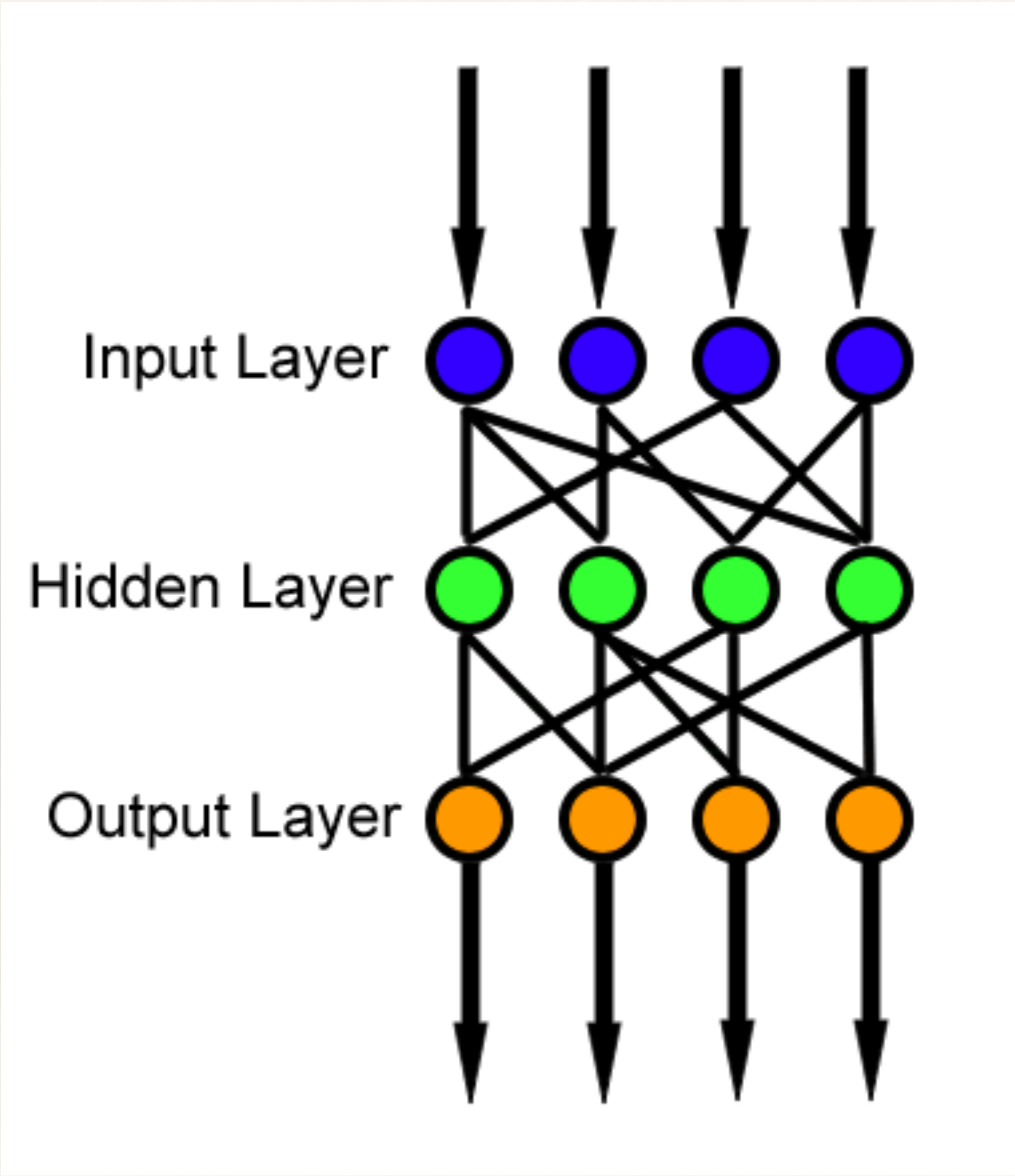
Outputs in one layer
Inputs of next layer

Non-linear Decision Boundaries

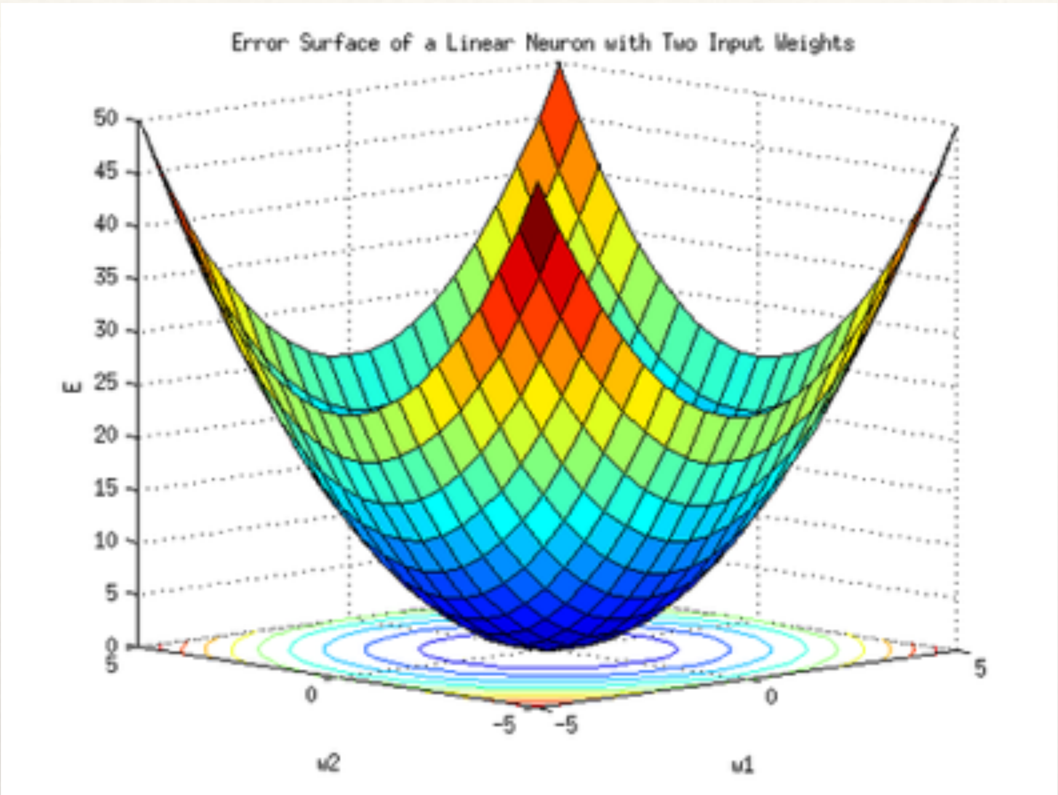


Complex NN

Feed Forward NN



More Complex Optimizations



Gradient Decent

Leads to

Back Propagation

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net_j}}\frac{\partial \mathrm{net_j}}{\partial w_{ij}}$$

Use partial derivatives to move error back
from output to input
to update weights

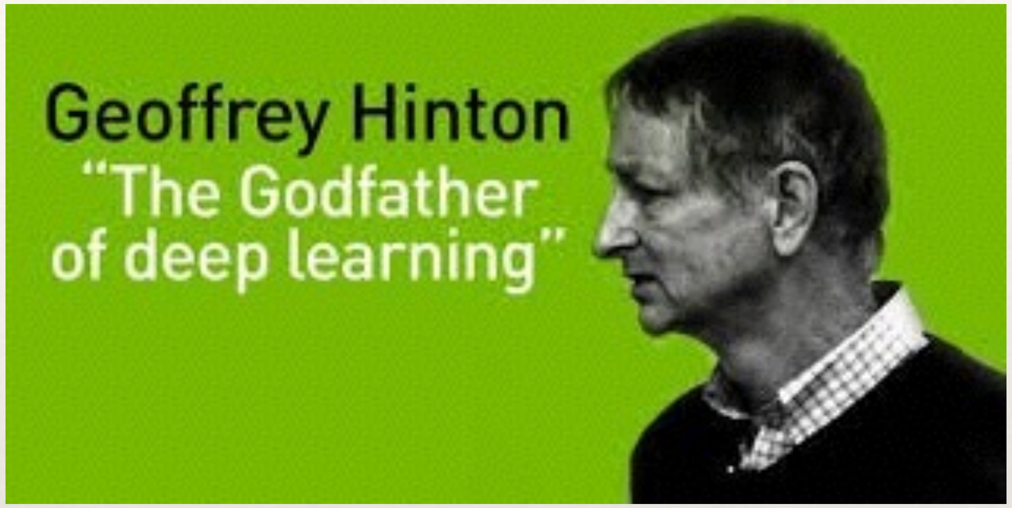
Traditional Problems with Neural Networks

- Black Box (what do the weights mean?)
- Picking the network
- Getting the weights to converge
- Overfitting
- Computational Cost
- Not enough Data

Traditional Problems with Neural Networks

- Black Box (what do the weights mean?)
- Picking the network
- Getting the weights to converge
- Overfitting
- ~~Computational Cost~~ Cheaper Cloud Computing
- ~~Not enough Data~~ Abundant Data

Neural Network Triumvirate



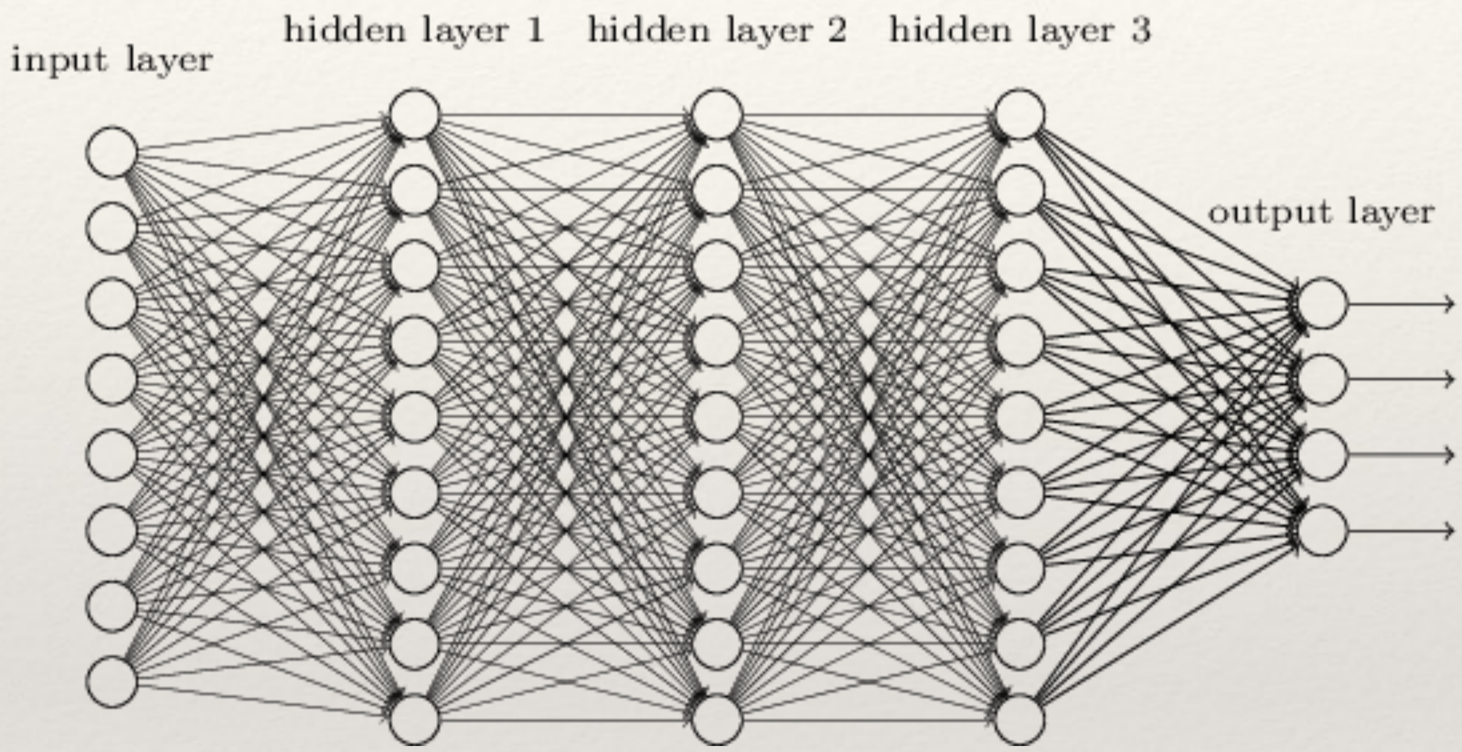
Traditional Problems with Neural Networks

- Black Box (what do the weights mean?)
- Picking the network
- Getting the weights to converge
- Overfitting
- Computational Cost
- Not enough Data

Task Specific Networks:
CNN, RNN

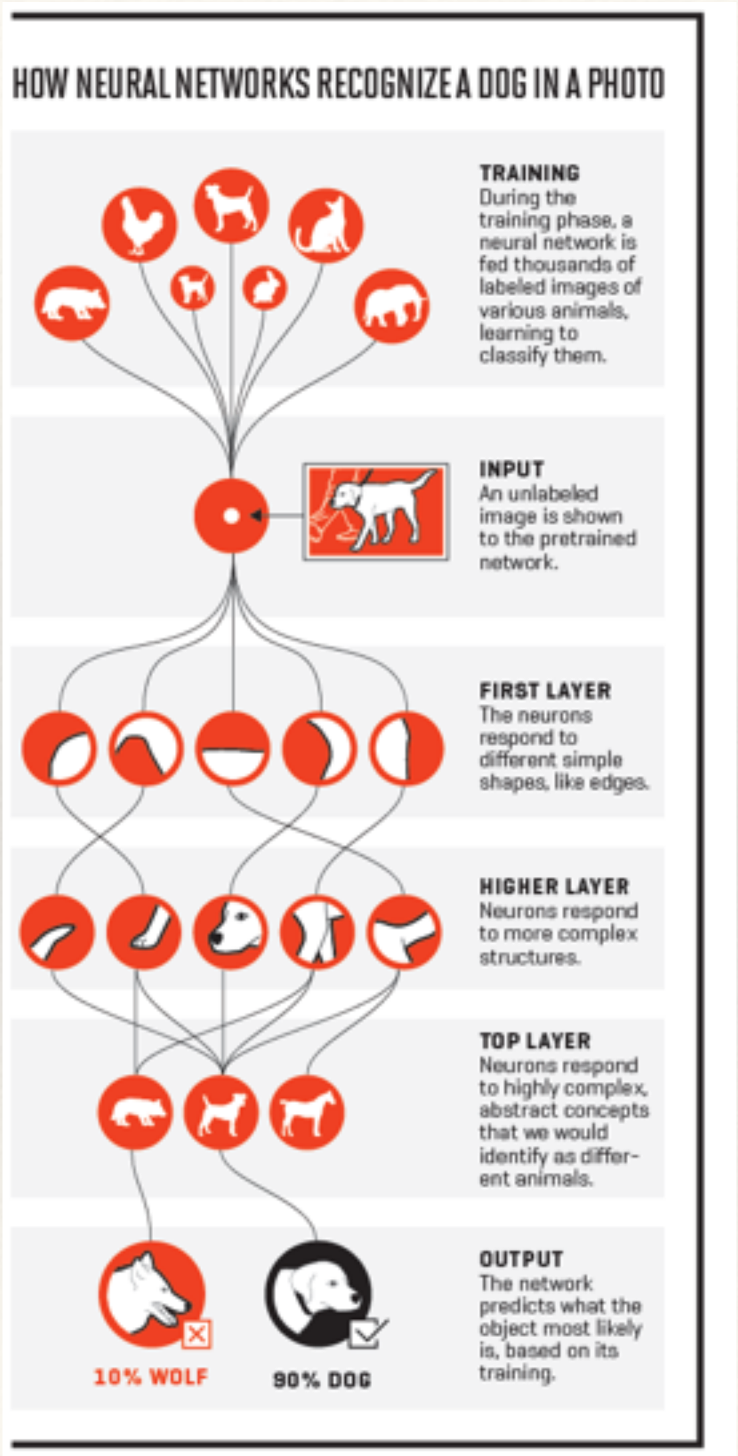
Pooling
Knockout

Deep Learning



ANN with MANY hidden layers

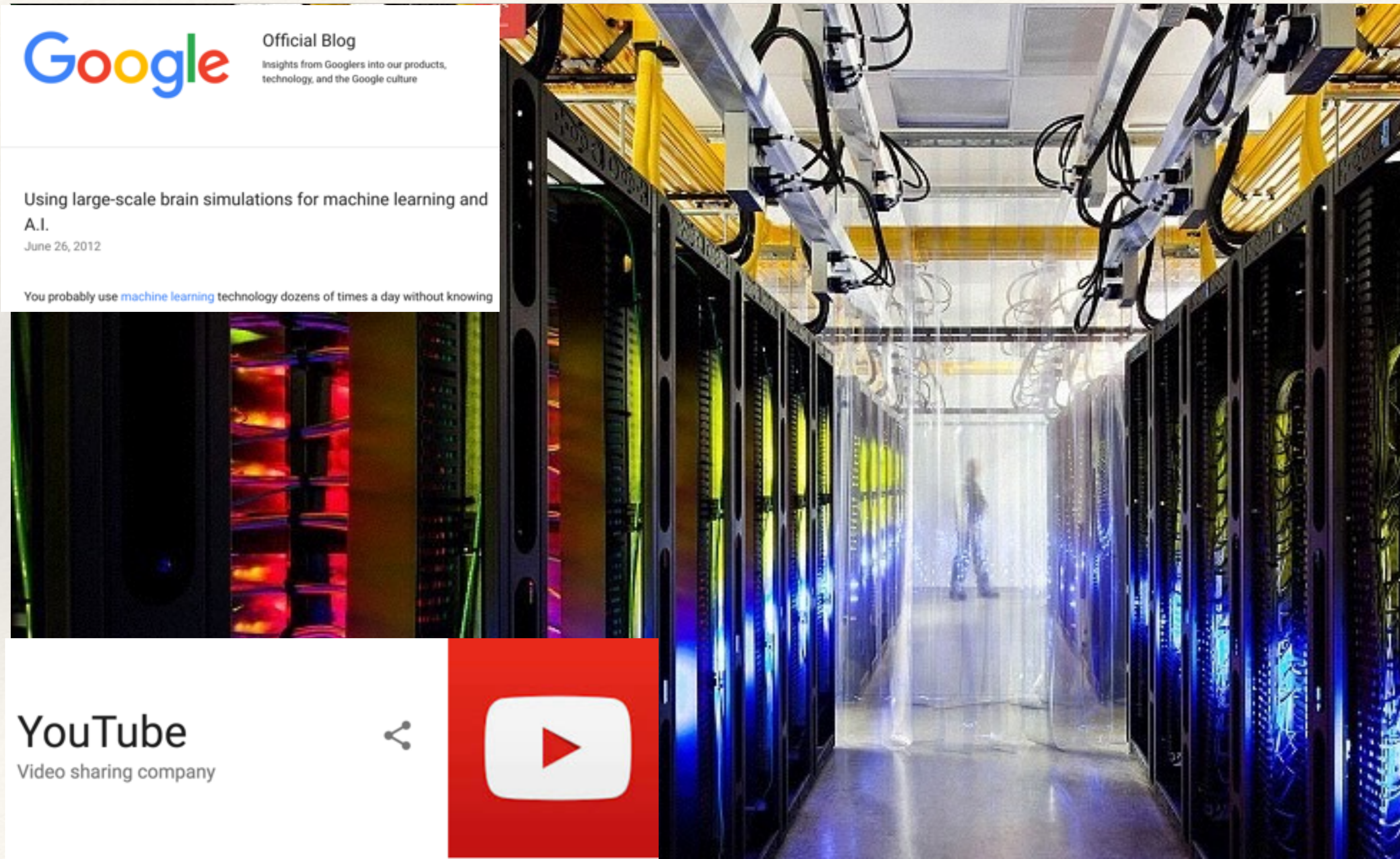
Why Should it work?



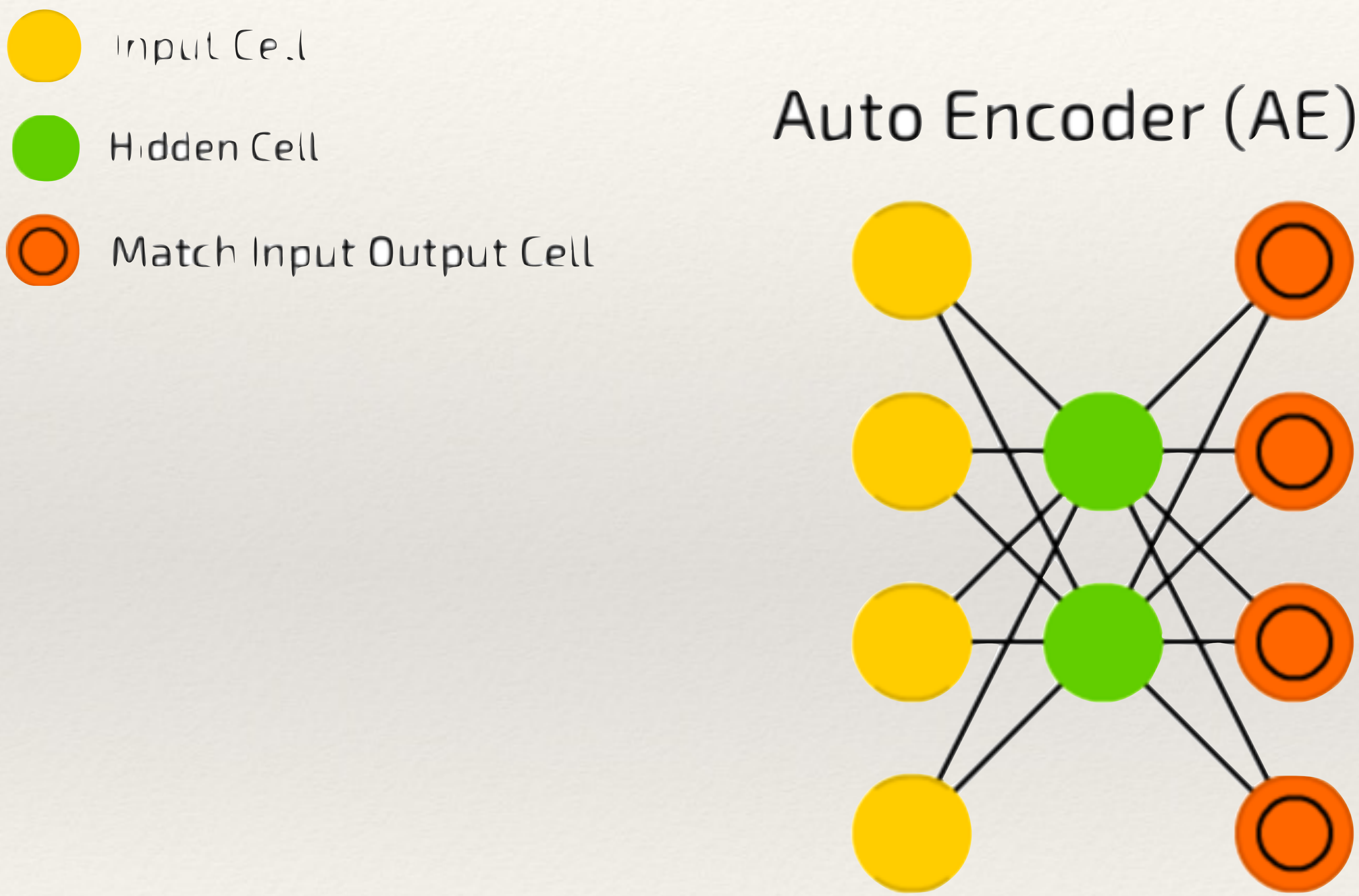
ImageNet

2007: 14 Million Labeled Images
Competition to Classify

2012: Google Brain

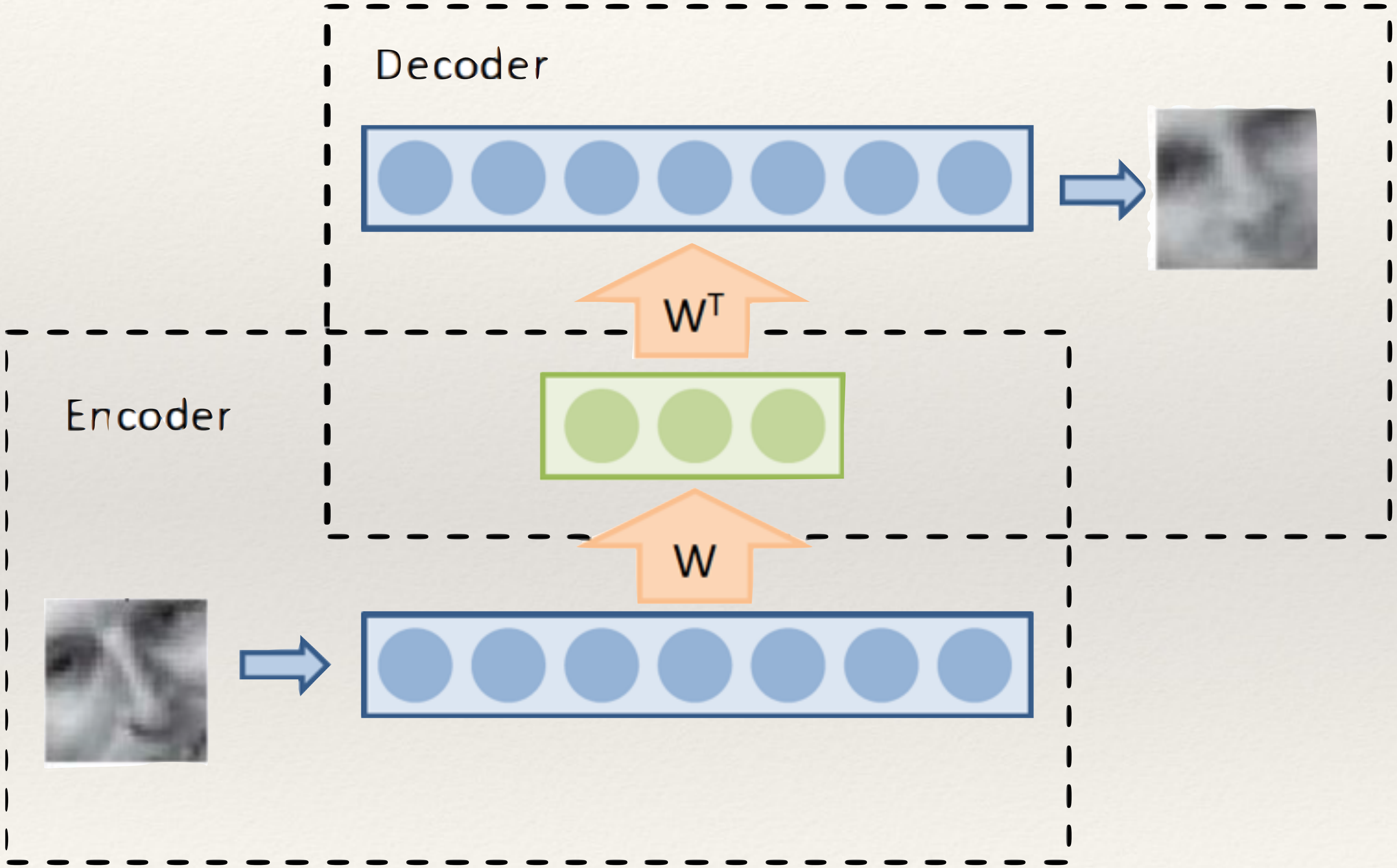


Auto-Encoder: Unsupervised Learning

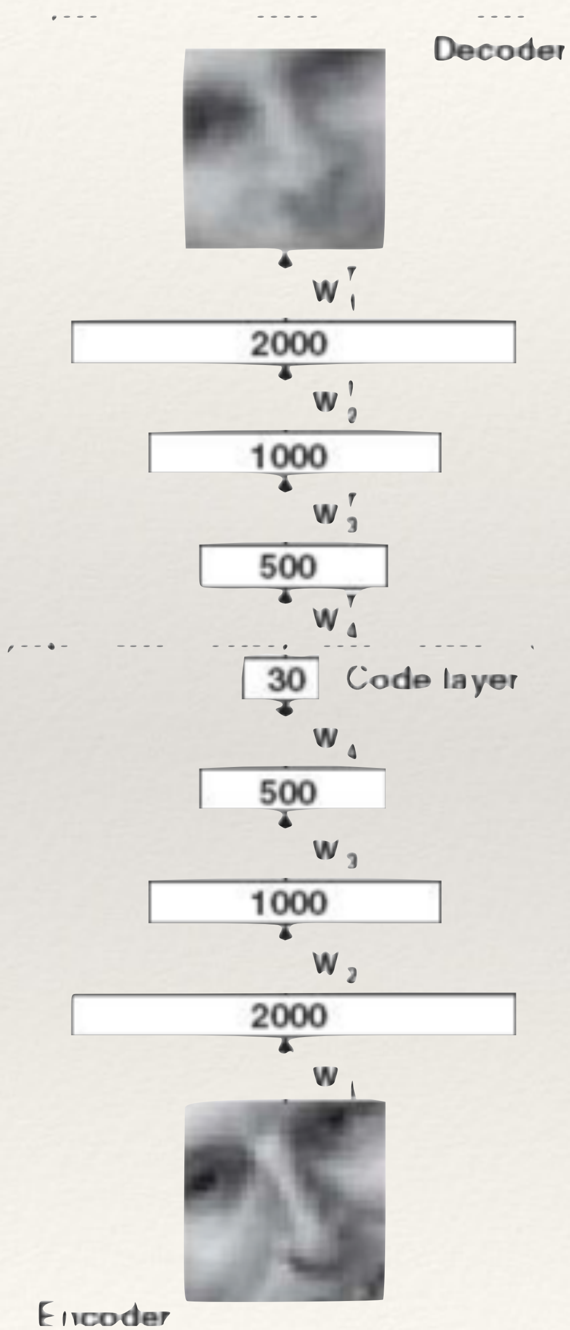


Learn Important Features

AutoEncoder

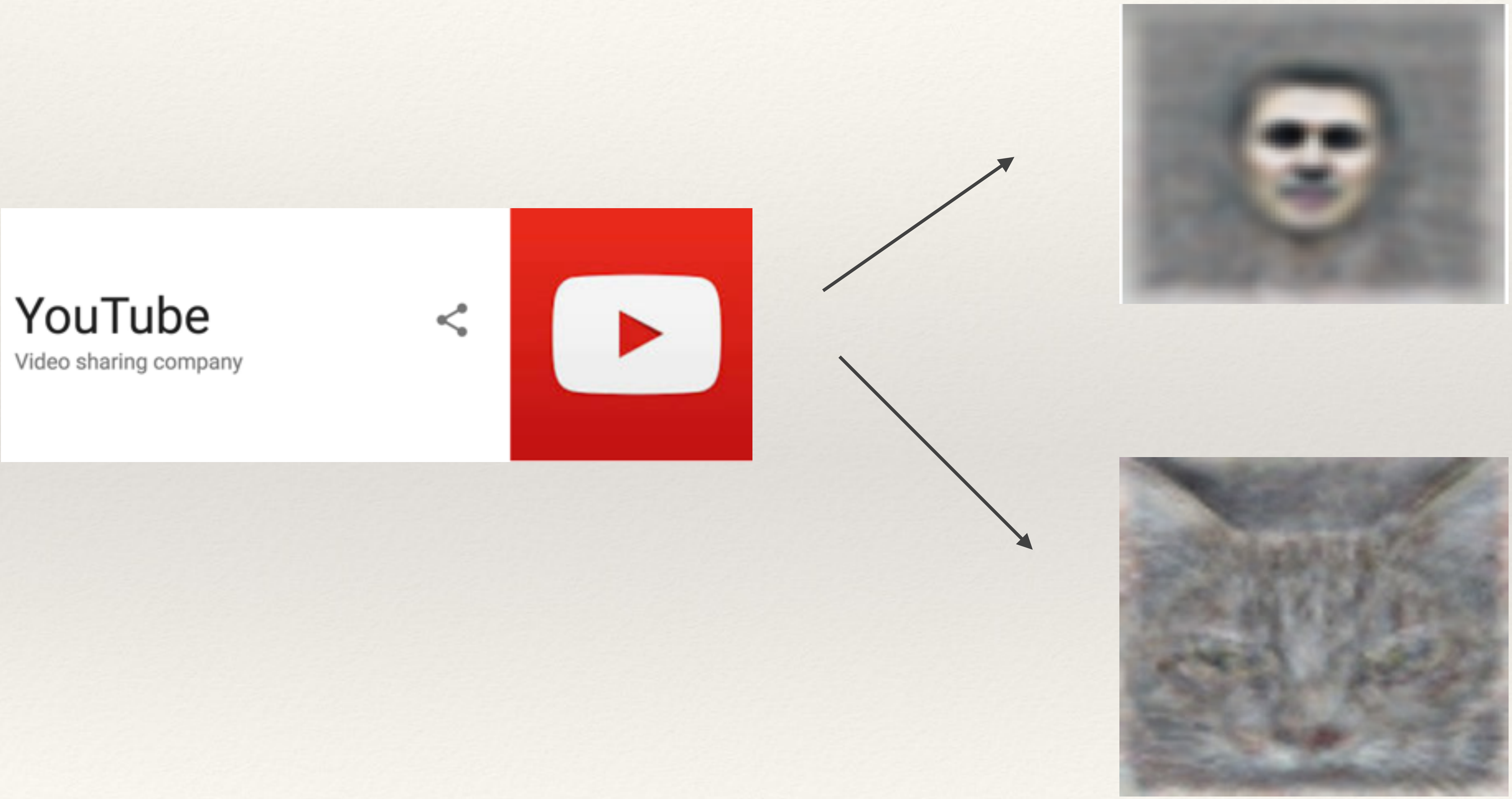


Deep Auto-Encoder

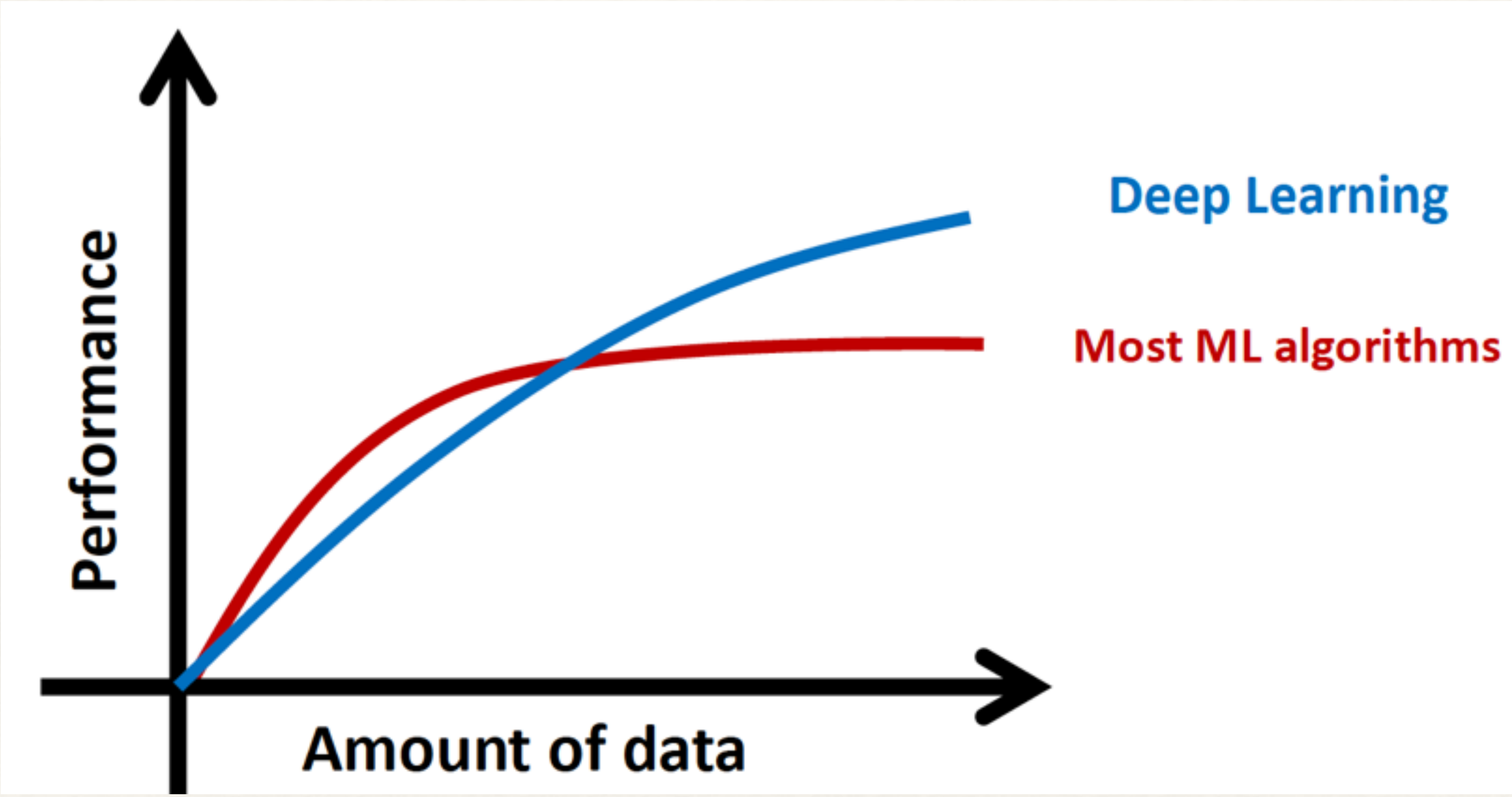


Linear -> Gives PCA
Non-linear Generalizes PCA
Use Noise to deal with overfitting
Features can be reused

What features?



Deep Learning Scales with Data



[Andrew Eng]

DeepMind 2013

Playing Atari with Deep Reinforcement Learning

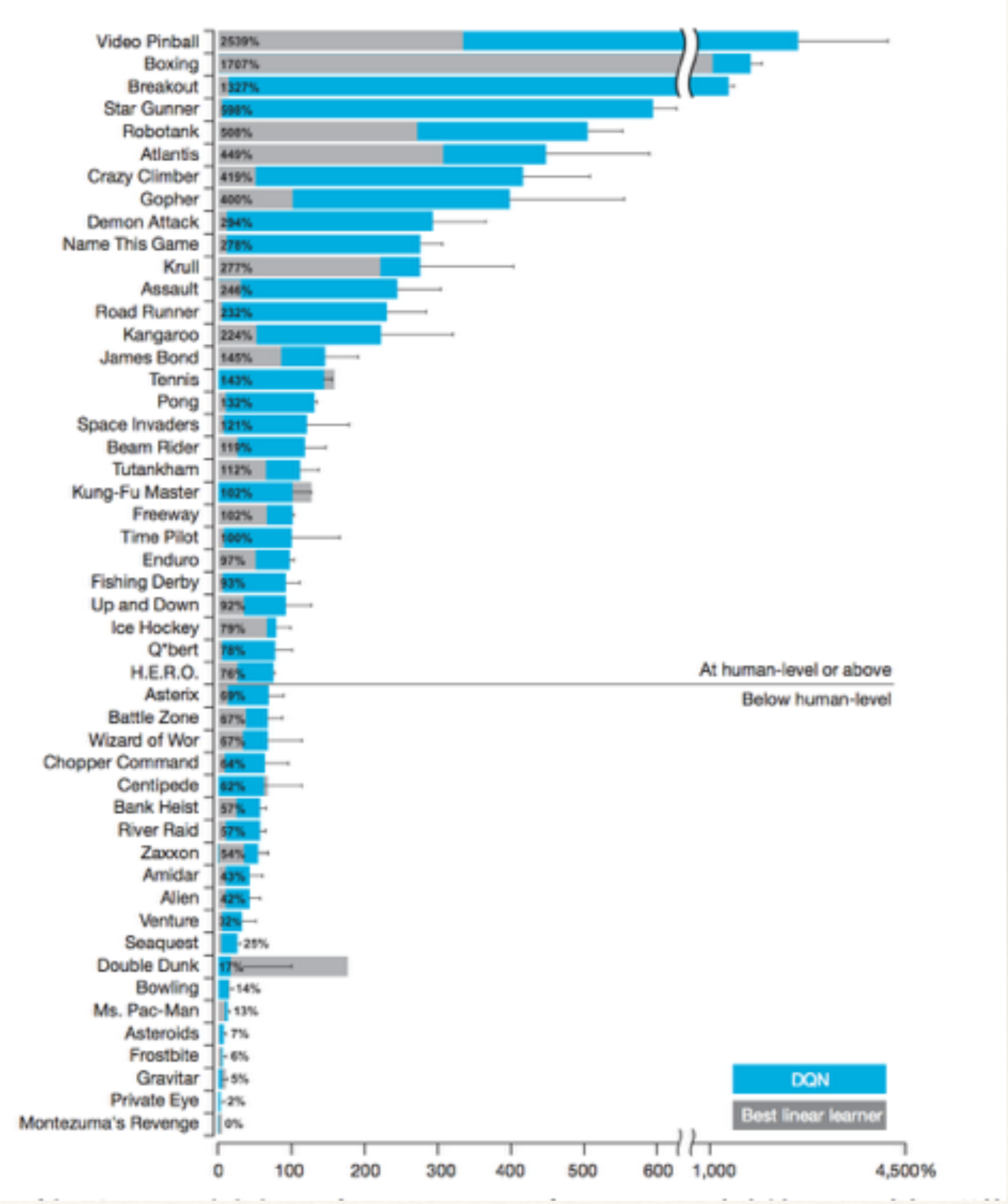
Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

Daan Wierstra **Martin Riedmiller**

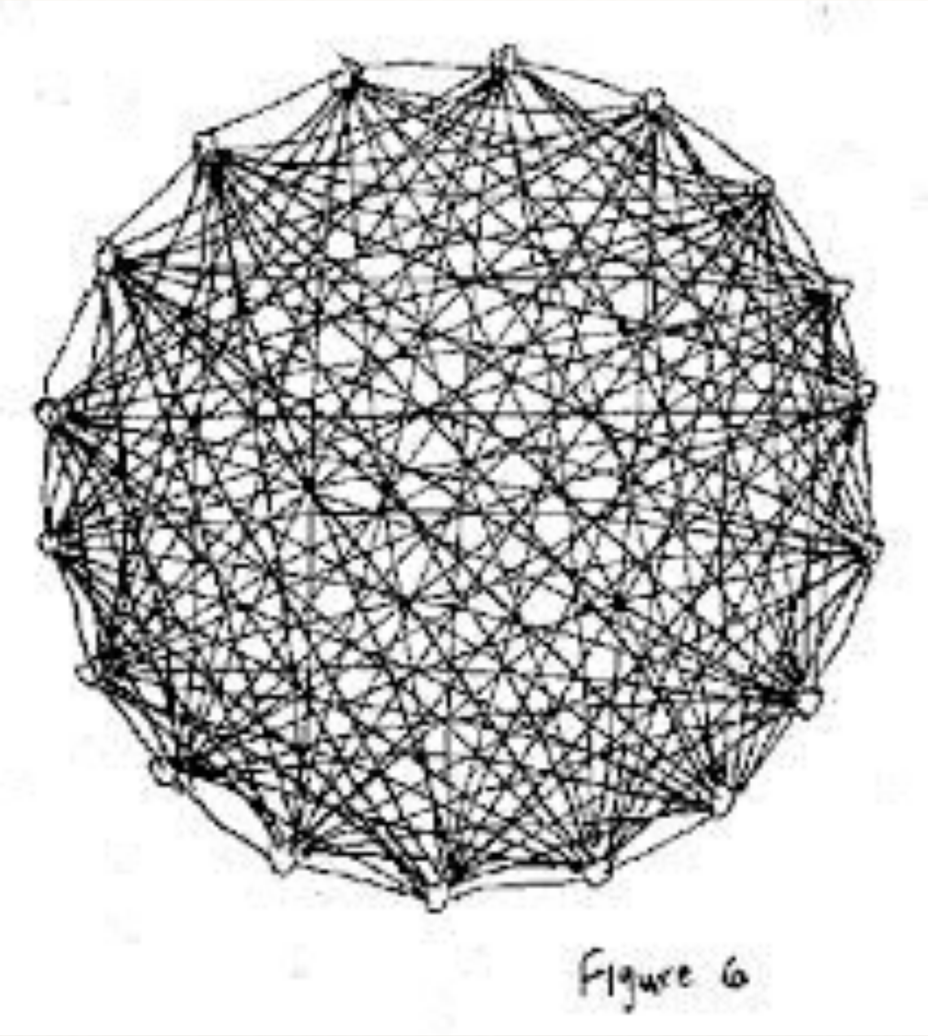
DeepMind Technologies

Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

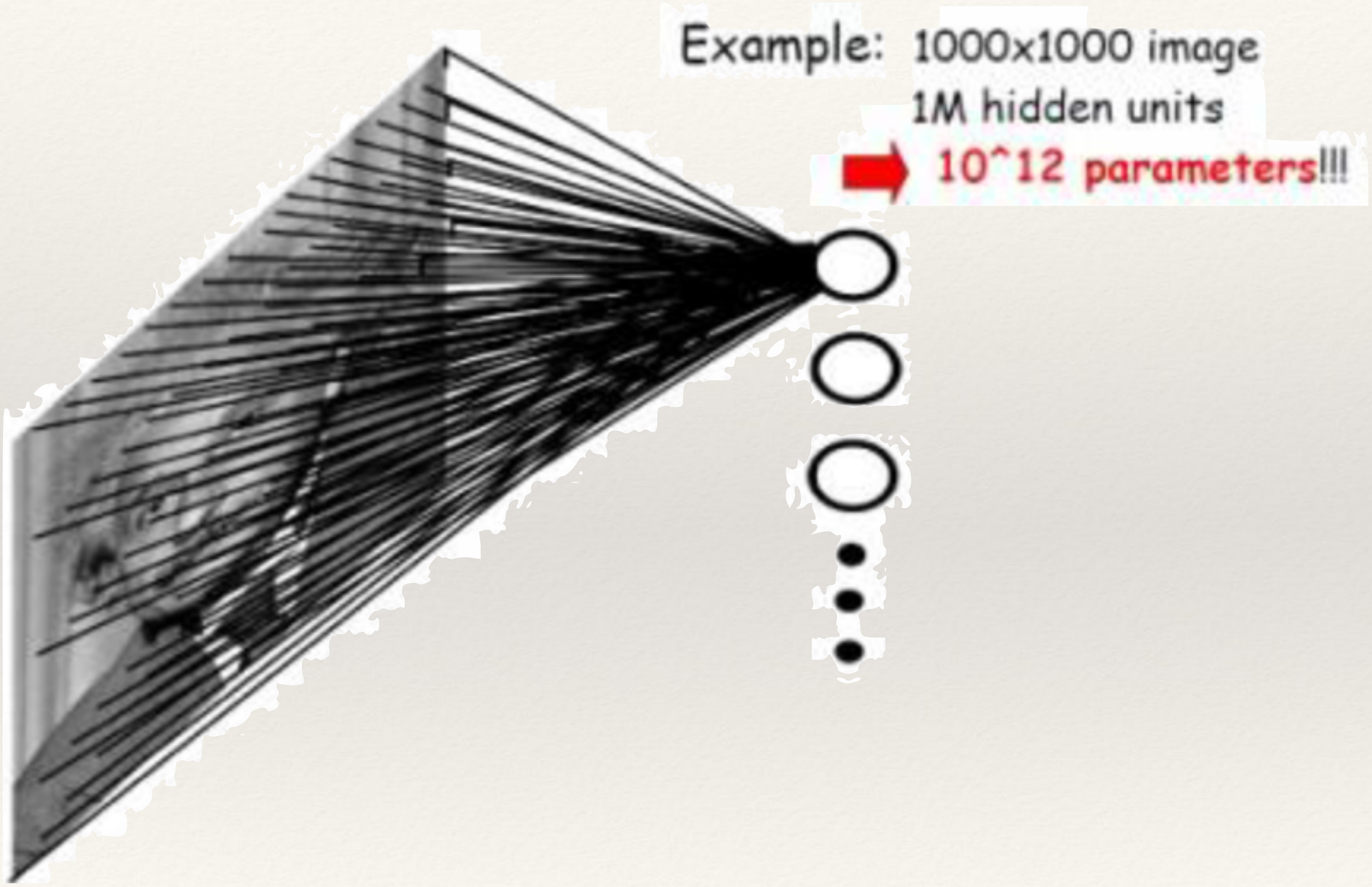
DeepMind Network: DQN per Game



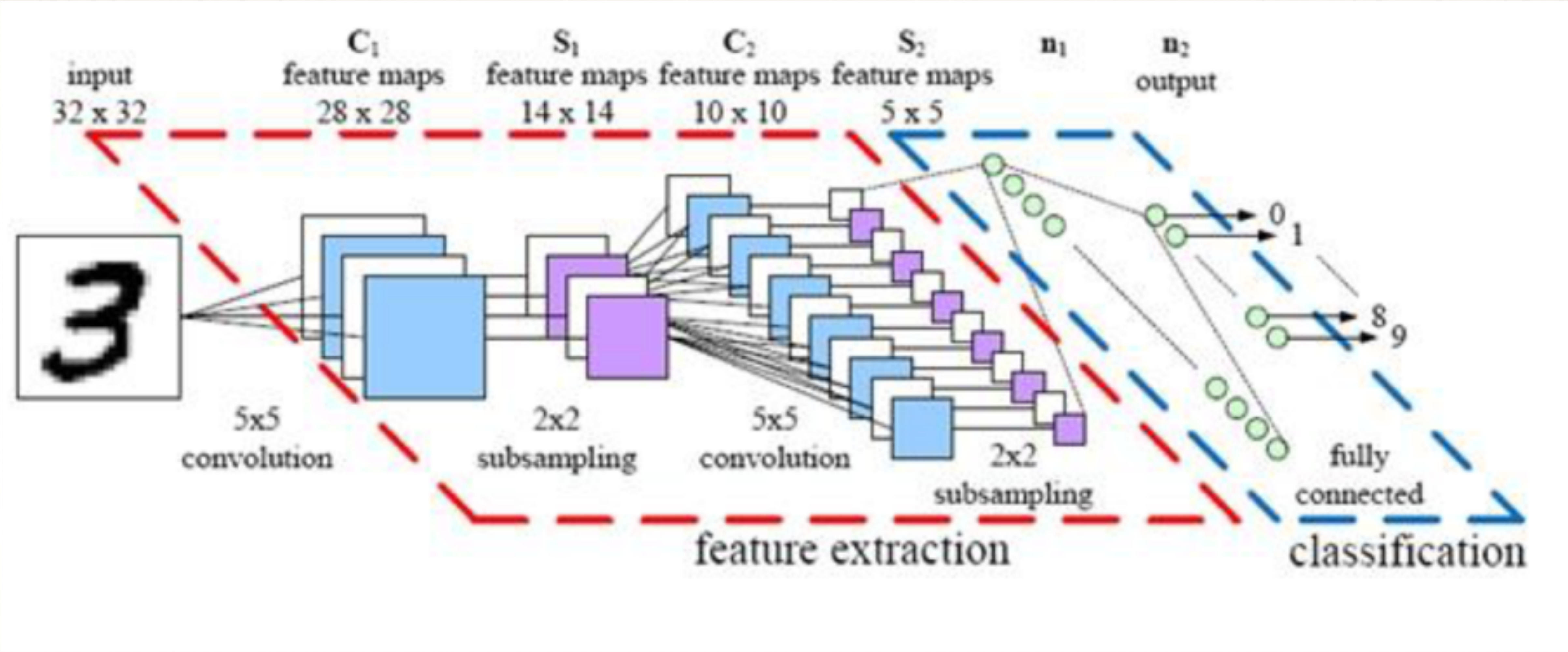
Complete Graph



Too Many Parameters



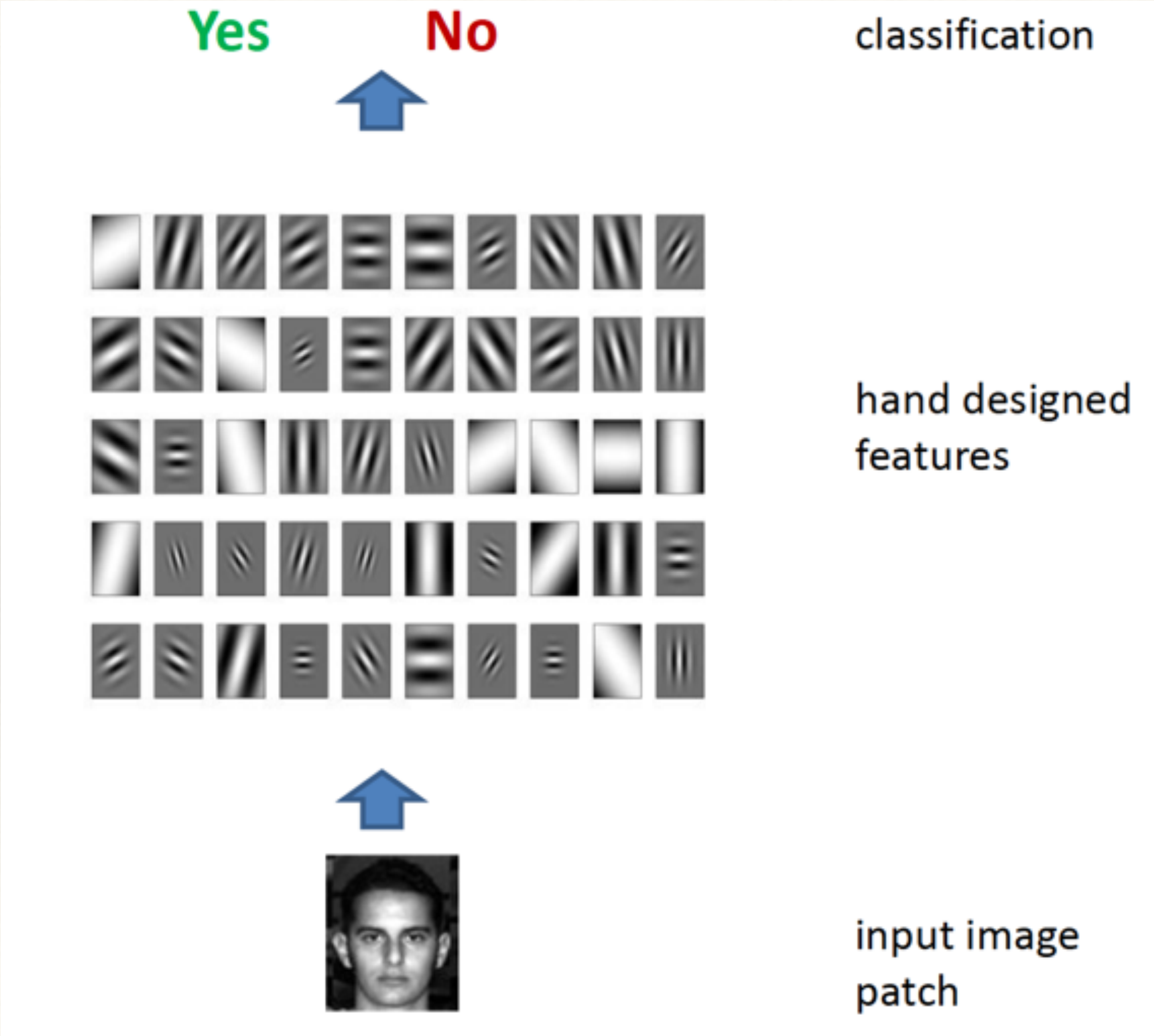
Convolution Neural Networks



ANN/CNN

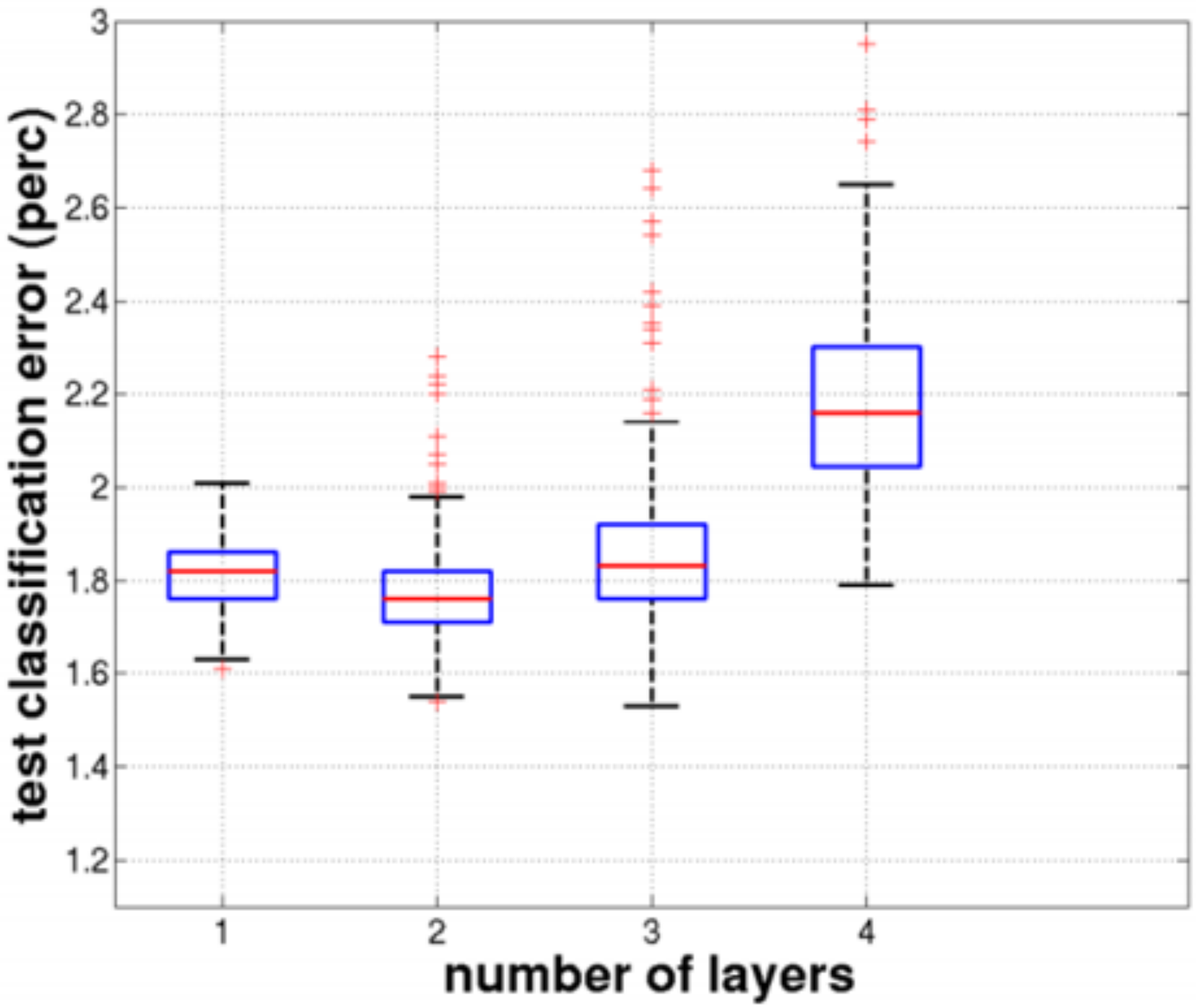
- Hard to train
- Big Breakthrough: Pre-Train with unsupervised networks auto encoder

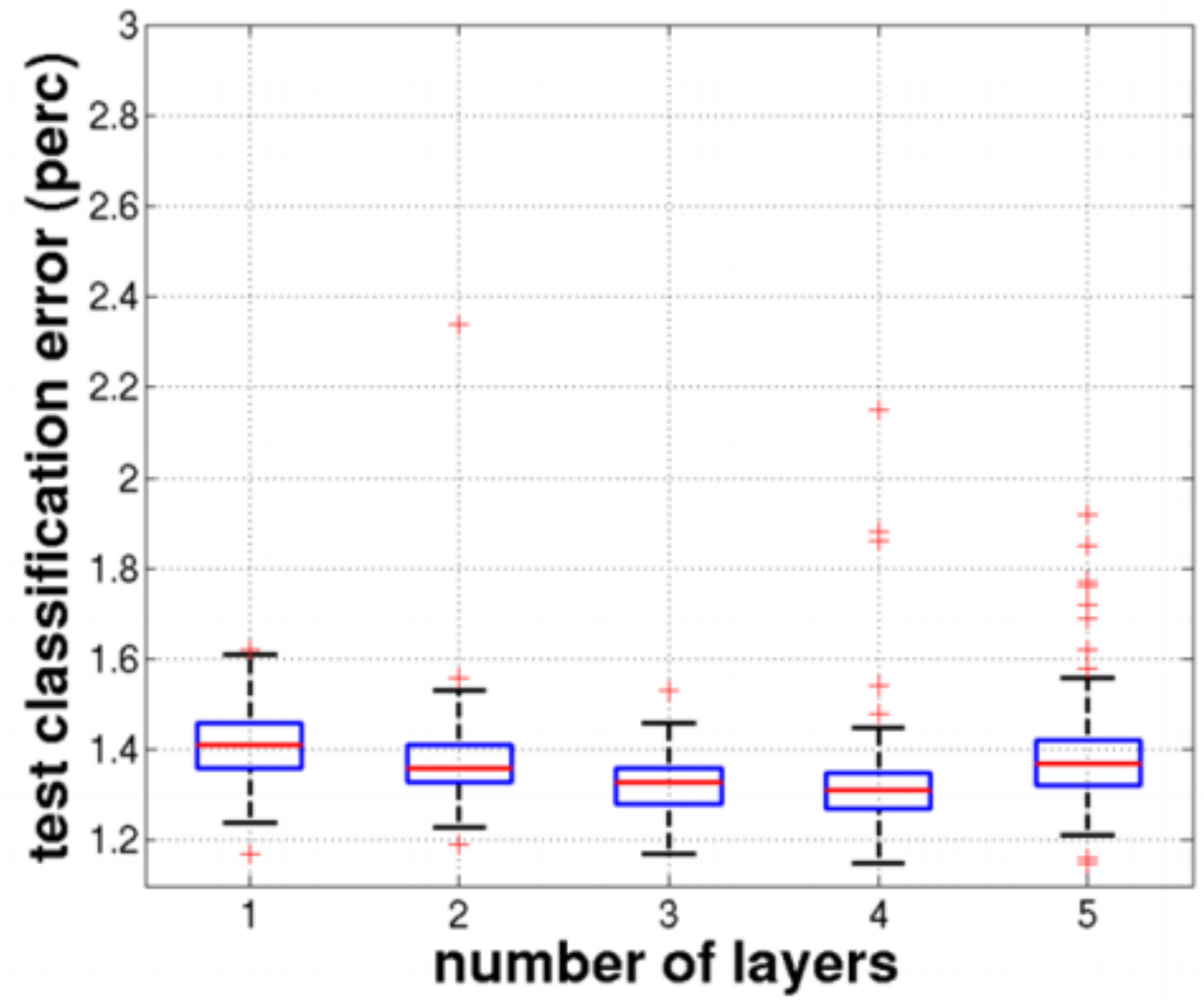
(Traditional) Feature Engineering



[Honglak Lee]

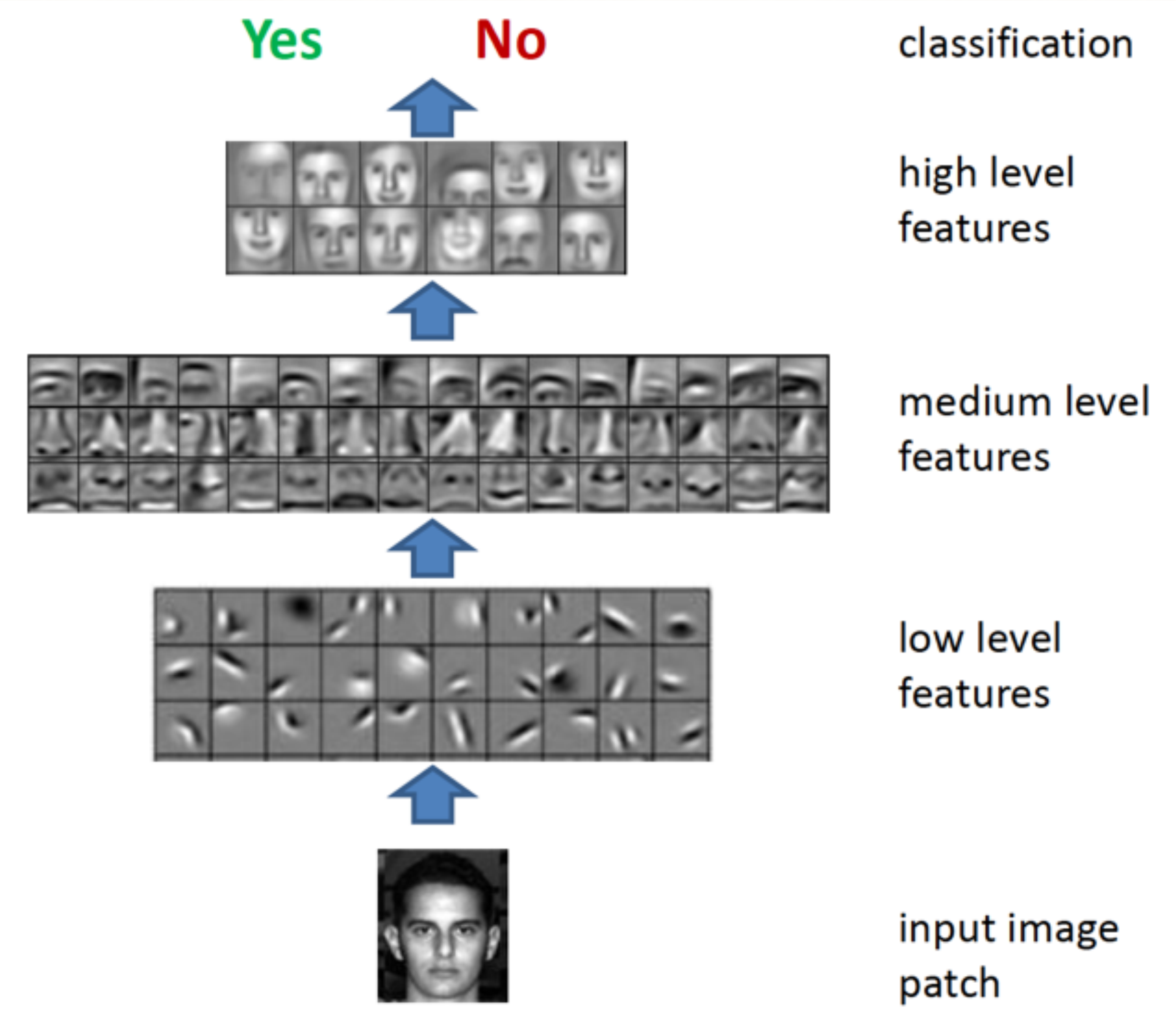
Pre-Training





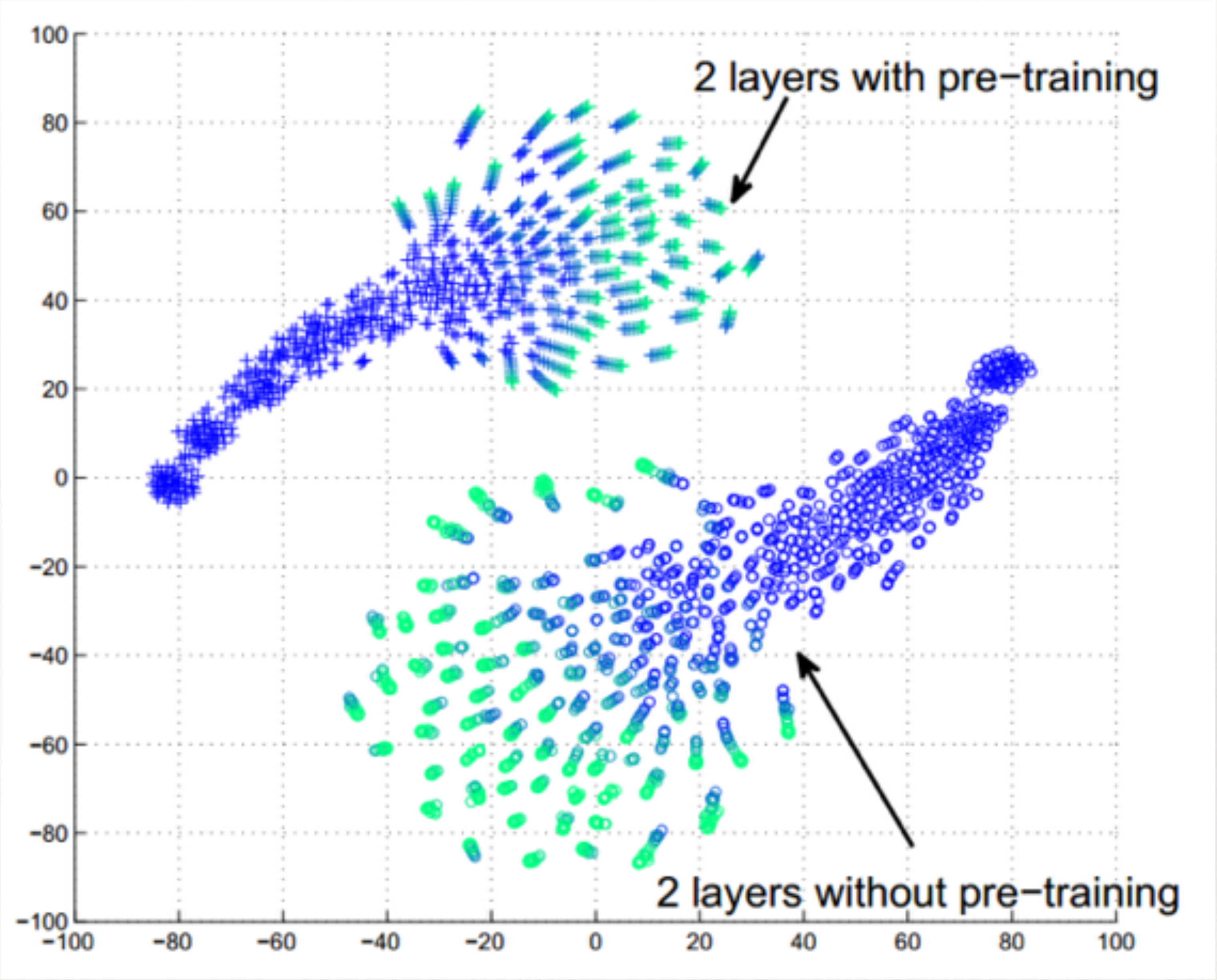
http://jmlr.org/papers/volume11/erhan10a/erhan10a.pdf

Deep Approach: Learned Features



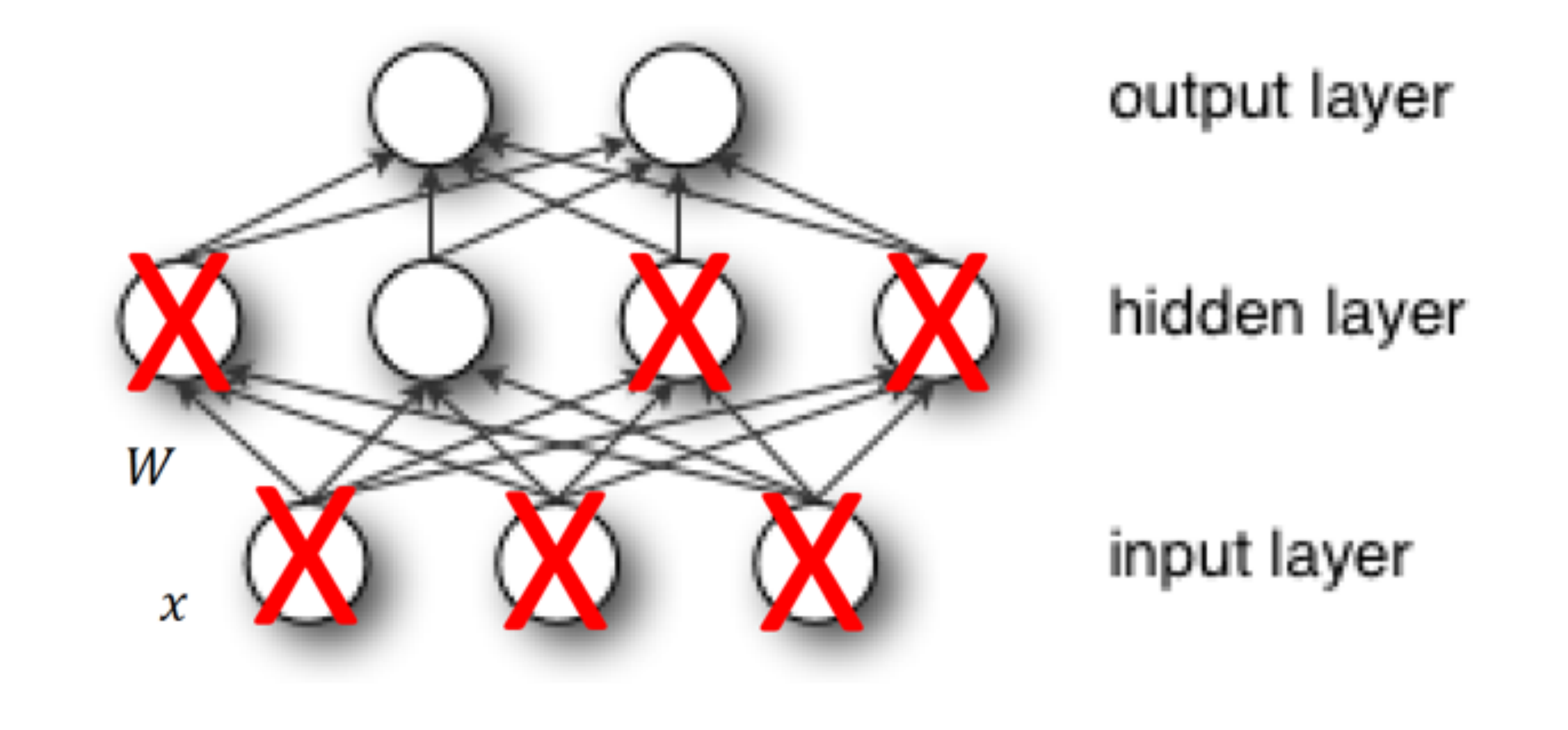
[Honglak Lee]

Very Different Convergence



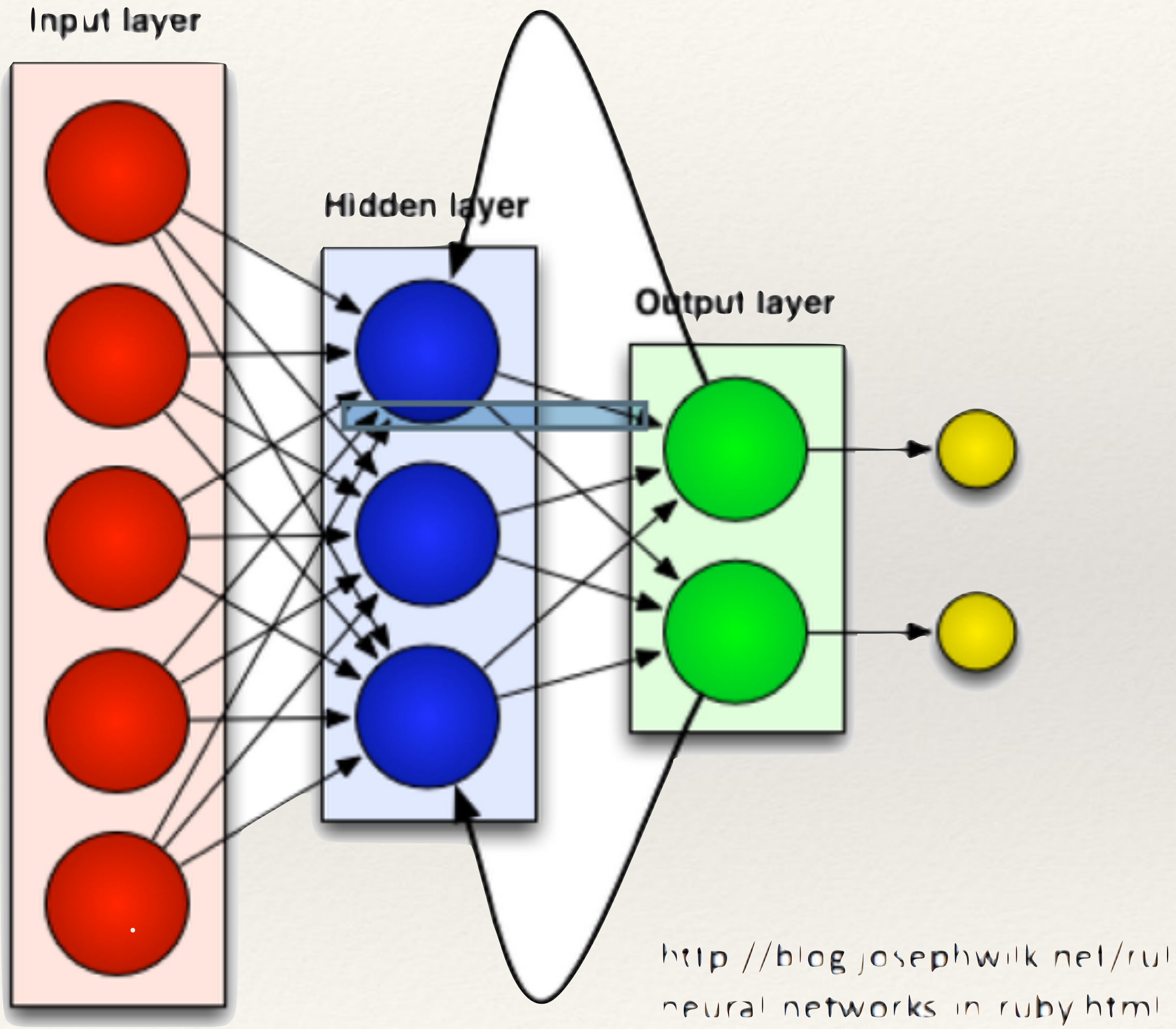
http://jmlr.org/papers/volume11/erhan10a/erhan10a.pdf

Dropout



Helps Address Overfitting

Recurrent Neural Net



http://blog.josephwilk.net/ruby/recurrent-neural-networks-in-ruby.html

Libraries

- Keras (python)
- Theano (python)
- TensorFlow (many wrappers)
- Cafe (C++)
- Torch (Lua)
- Deeplearning4j (java/spark)
- MXnet
- Comparison: https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software