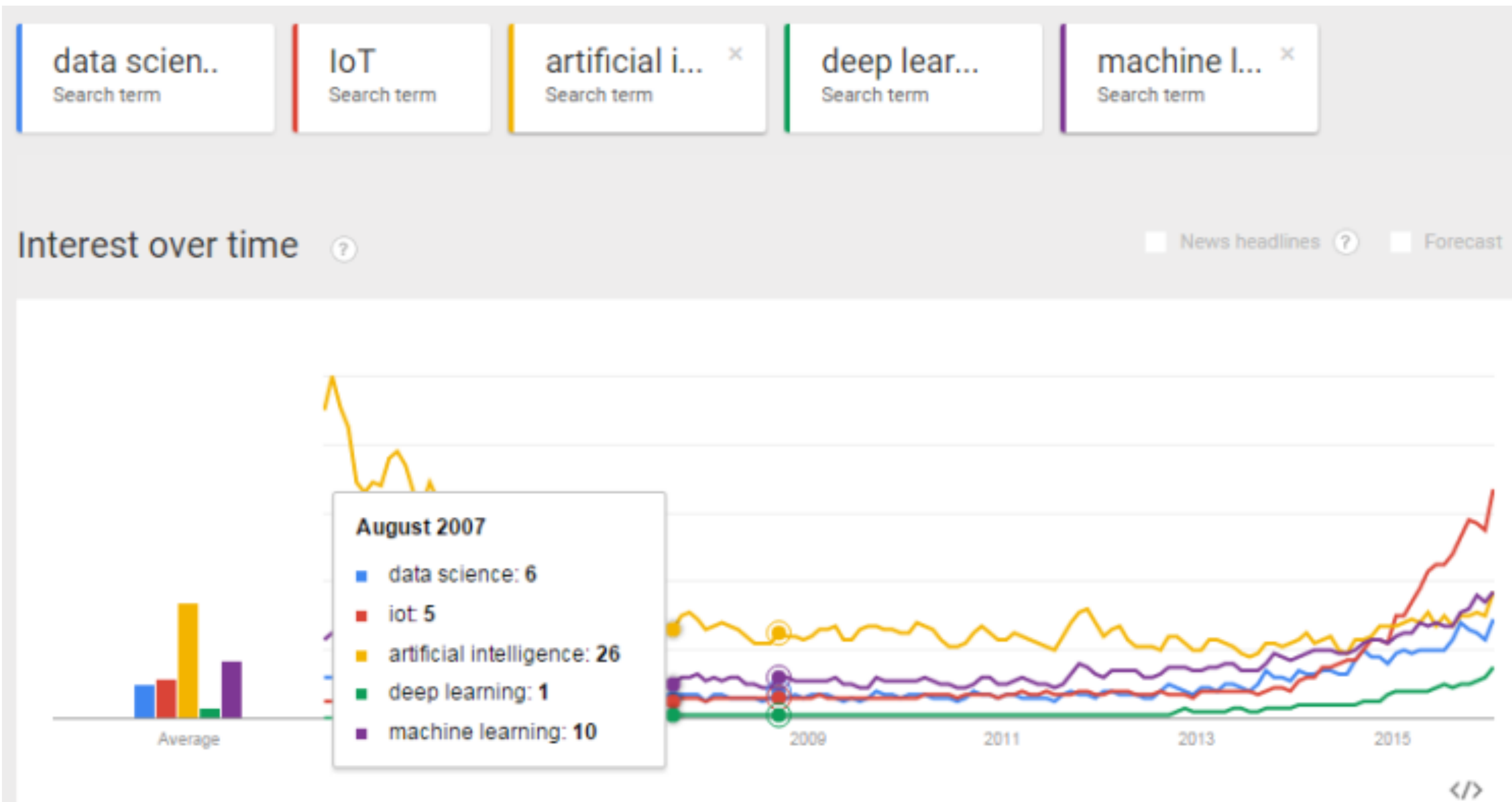
Introduction to Deep Learning

CS 584: Big Data Analytics

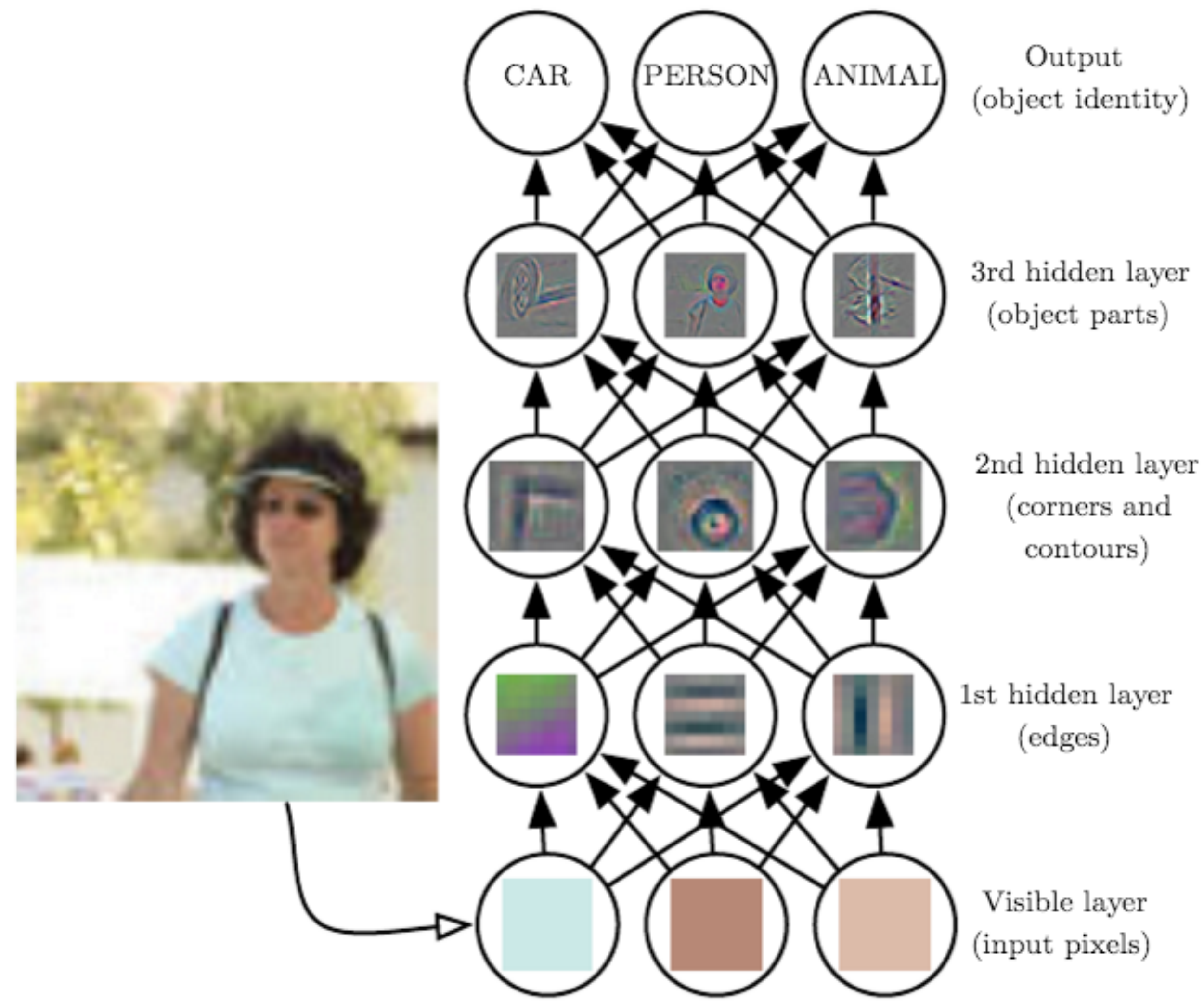
Deep Learning: “The New Cool”



http://www.datasciencecentral.com/profiles/blogs/data-science-ml-deep-learning-iot-ai-exploding

Deep Learning: Overview

- Form of representation learning
- Aimed at learning feature hierarchies
- Features from higher levels of the hierarchy are formed by lower level features
- Each hidden layer allows for more complex features of input

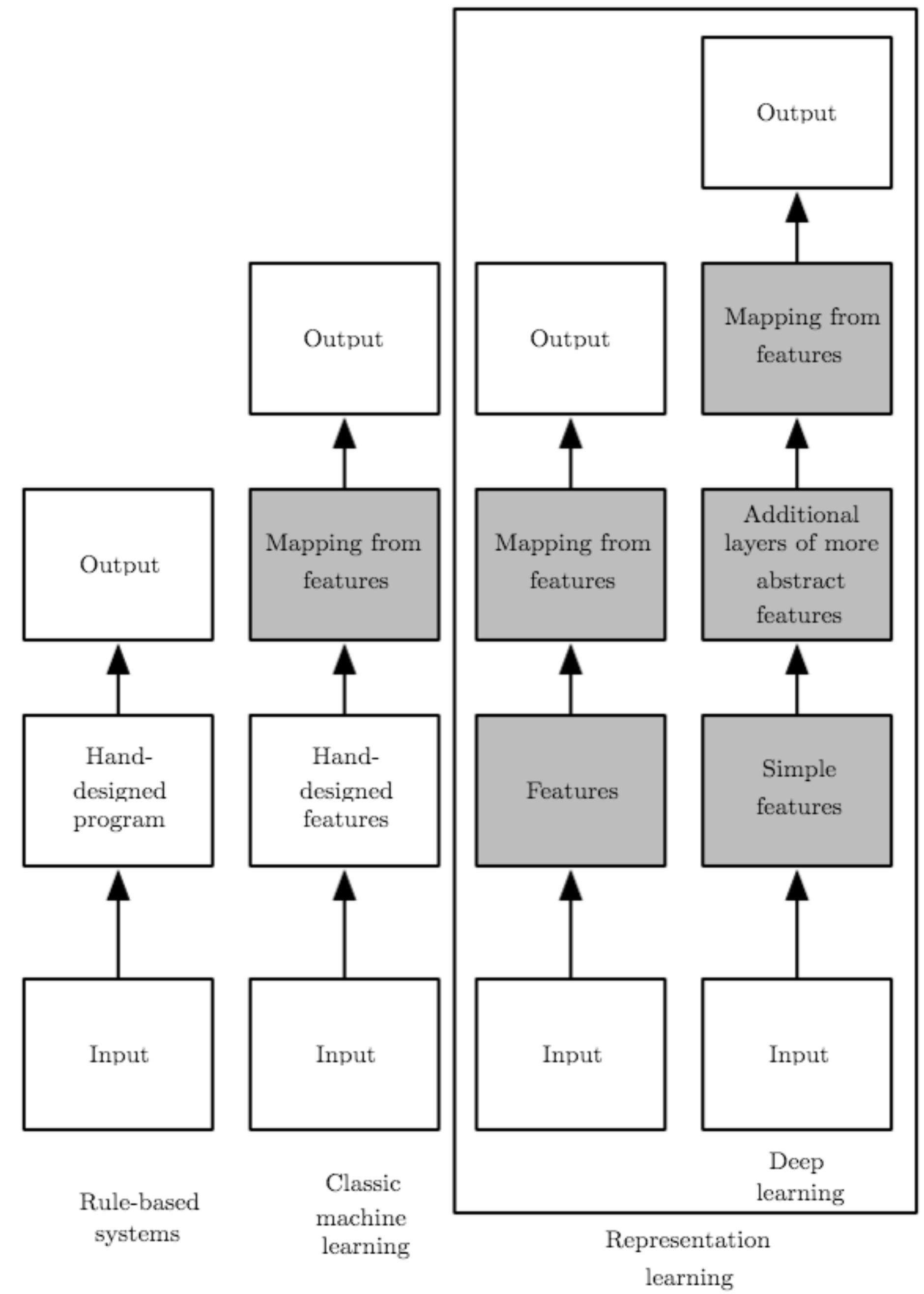


http://www.deeplearningbook.org/contents/intro.html

Deep Learning: The Promised Land

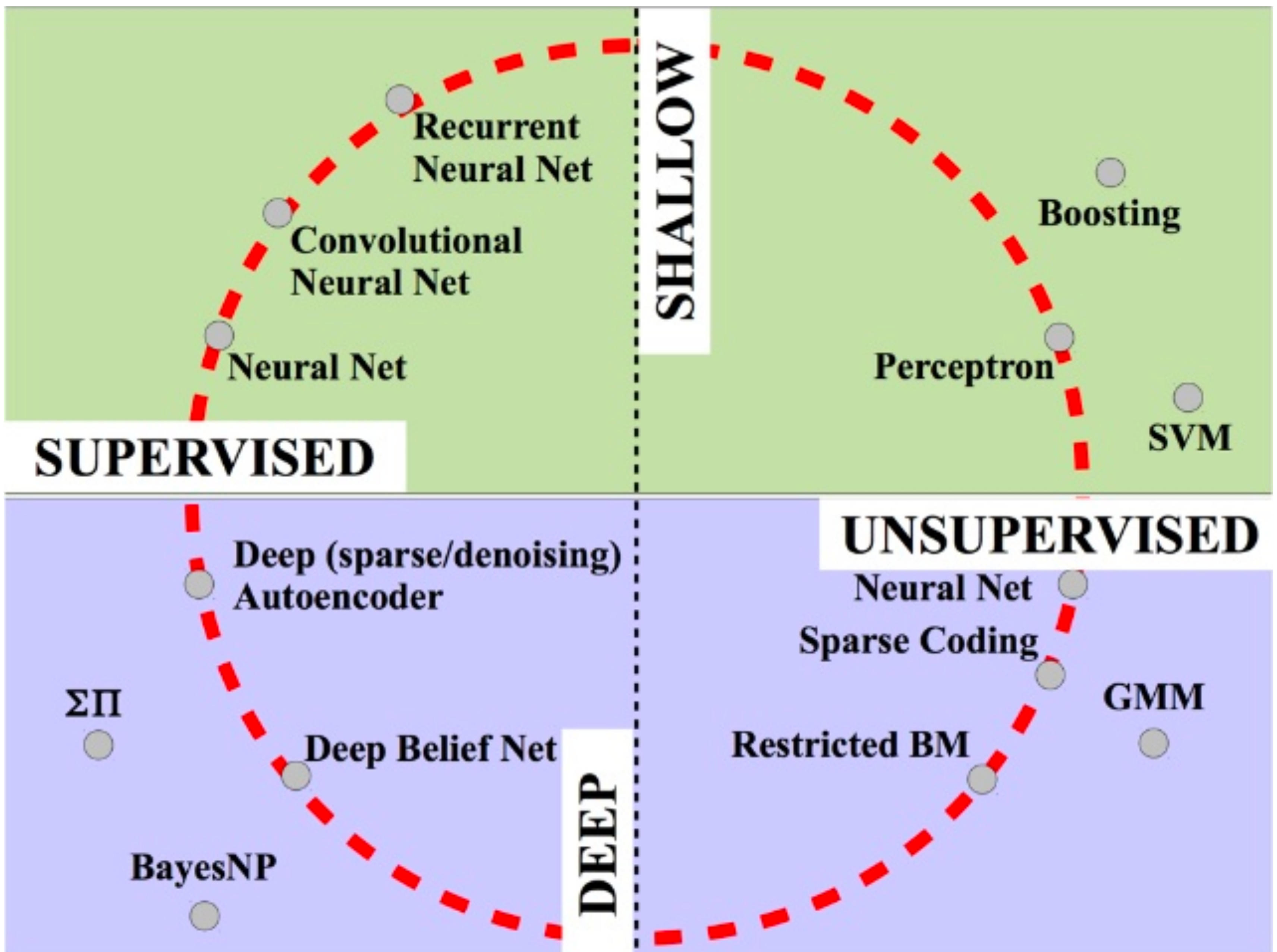
Automatic feature discovery

- Hidden layers discover semantically meaningful concepts
- Features learned without need for seeing exponentially large number of configuration of other features
- Expressiveness of deep networks



http://www.deeplearningbook.org/contents/intro.html

Deep vs Shallow Architectures



http://www.slideshare.net/roelofp/041114-dl-nlpwordembeddings

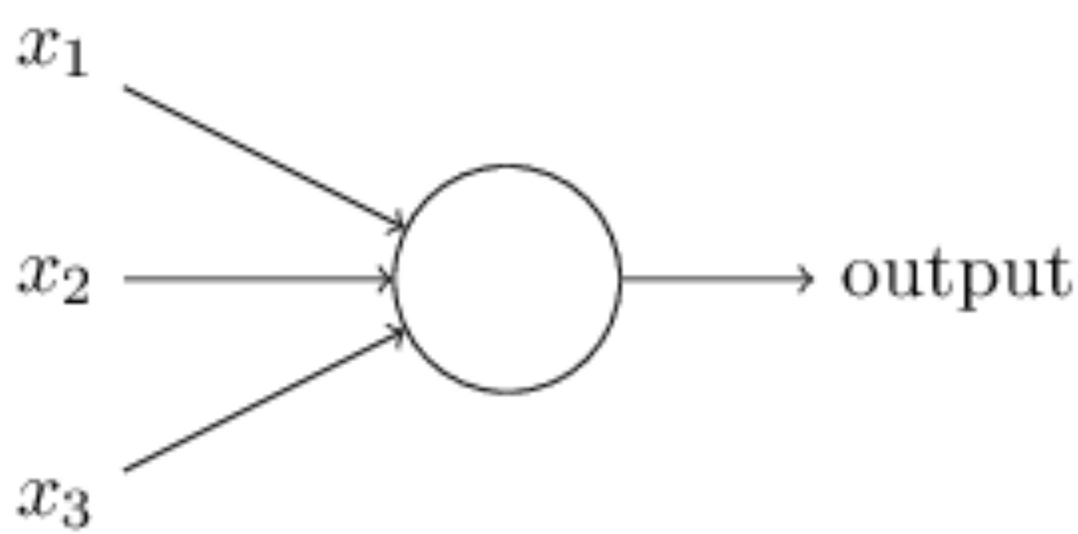
History of Deep Learning

- Inspired by architectural depth of the brain, researchers wanted to train deep multi-layer neural networks
- No successful attempts were reported before 2006 except convolutional neural networks [LeCun, 1998]
  - Positive experimental results with two or three levels (or or two hidden layers), but training deeper networks was computationally infeasible or yielded poor results
- Breakthrough in 2006: Deep Belief Networks [Hinton et al., 2006] & Autoencoders [Bengio et al., 2007]

http://www.cs.cmu.edu/~aarti/Class/10701_Spring14/slides/DeepLearning.pdf

Perceptron [Rosenblatt, 1957]

- Binary classifier that maps input to an output value
- Basic neural network building block
- Simplest feedforward neural network

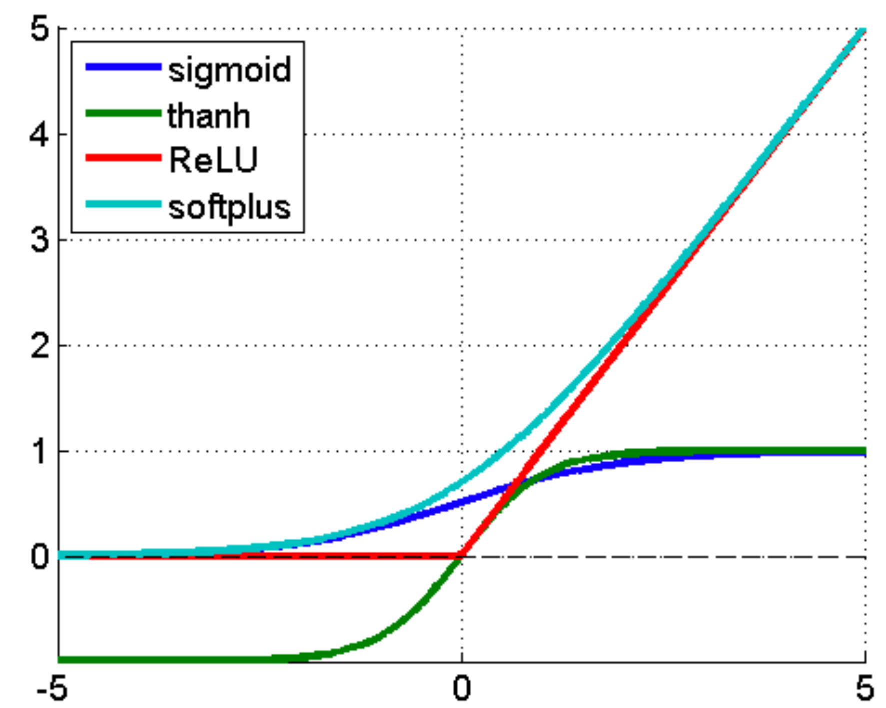


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

http://neuralnetworksanddeeplearning.com/chap1.html

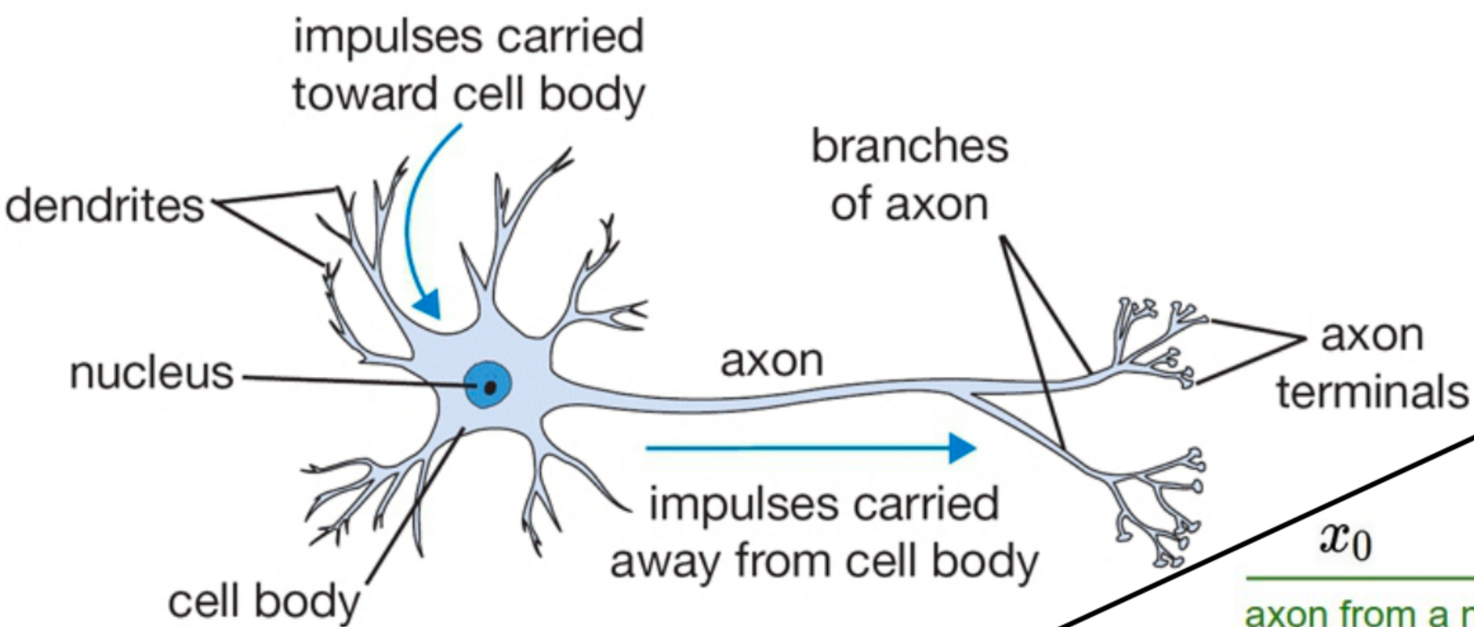
Perceptron: General Case

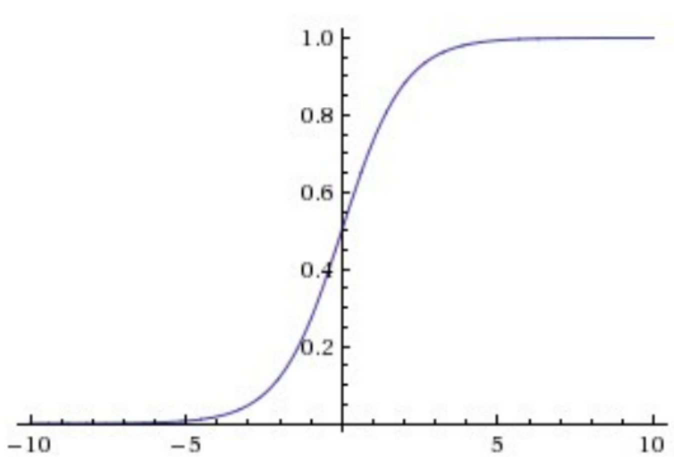
- Threshold is cumbersome and can be replaced by a bias (input of 1)
- Different activation functions can be utilized (for non-linear classification)
  - Sigmoid function
  - Hyperbolic tangent function
  - Rectified linear unit (ReLU)
  - Softplus



https://imiloainf.wordpress.com/2013/11/06/rectifier-nonlinearities/

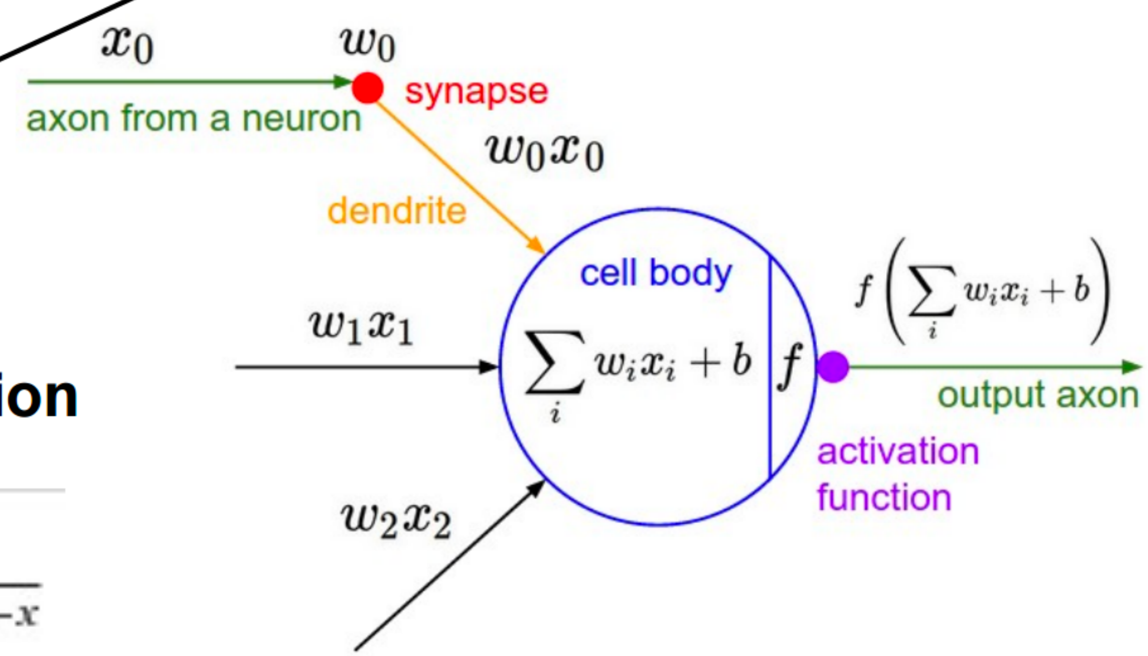
Neuron —> Perceptron





sigmoid activation function

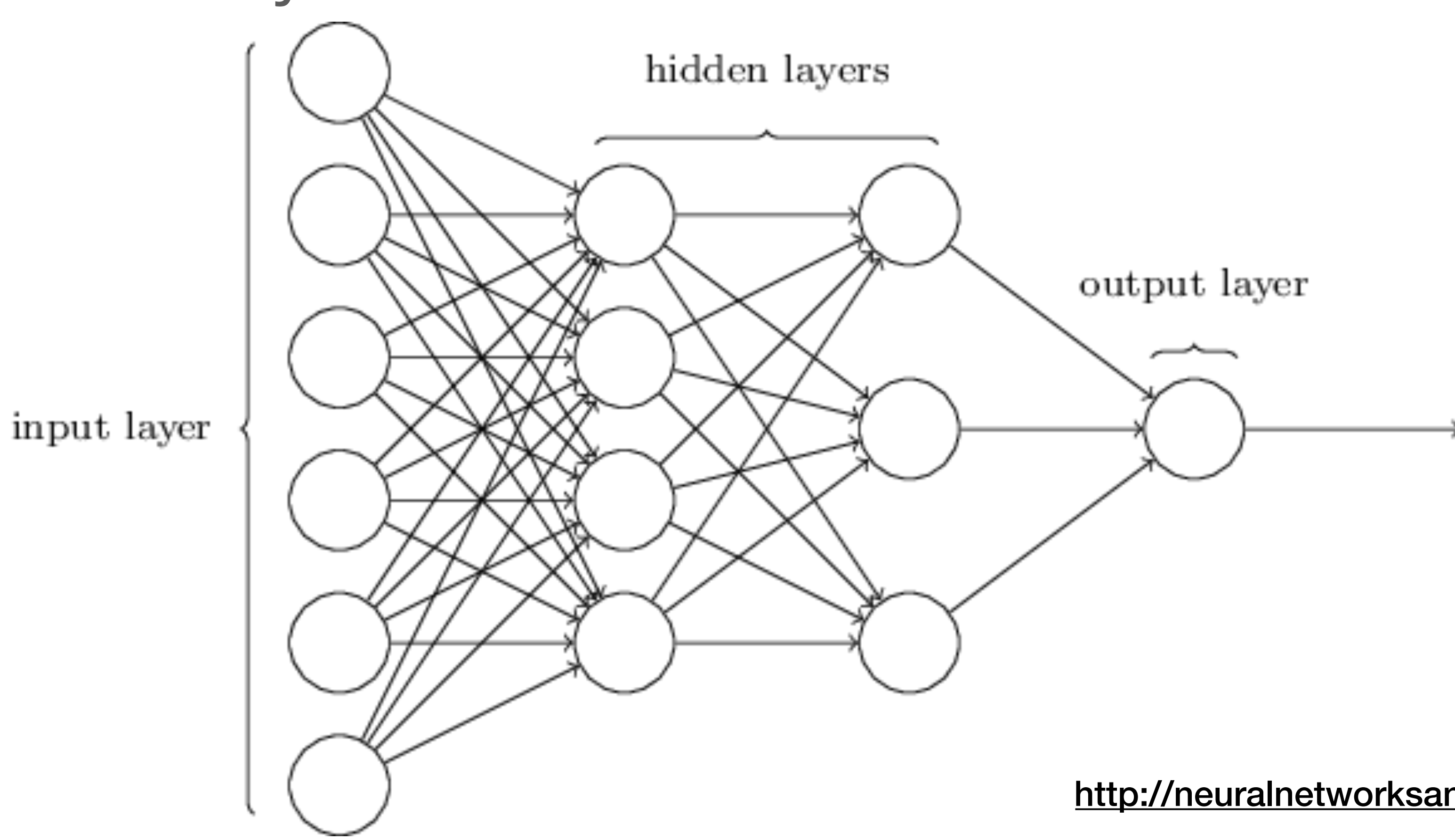
$$\frac{1}{1+e^{-x}}$$



http://vision.stanford.edu/teaching/cs231n/slides/lecture5.pdf

Multilayer Perceptrons (MLP): Feedforward Neural Network

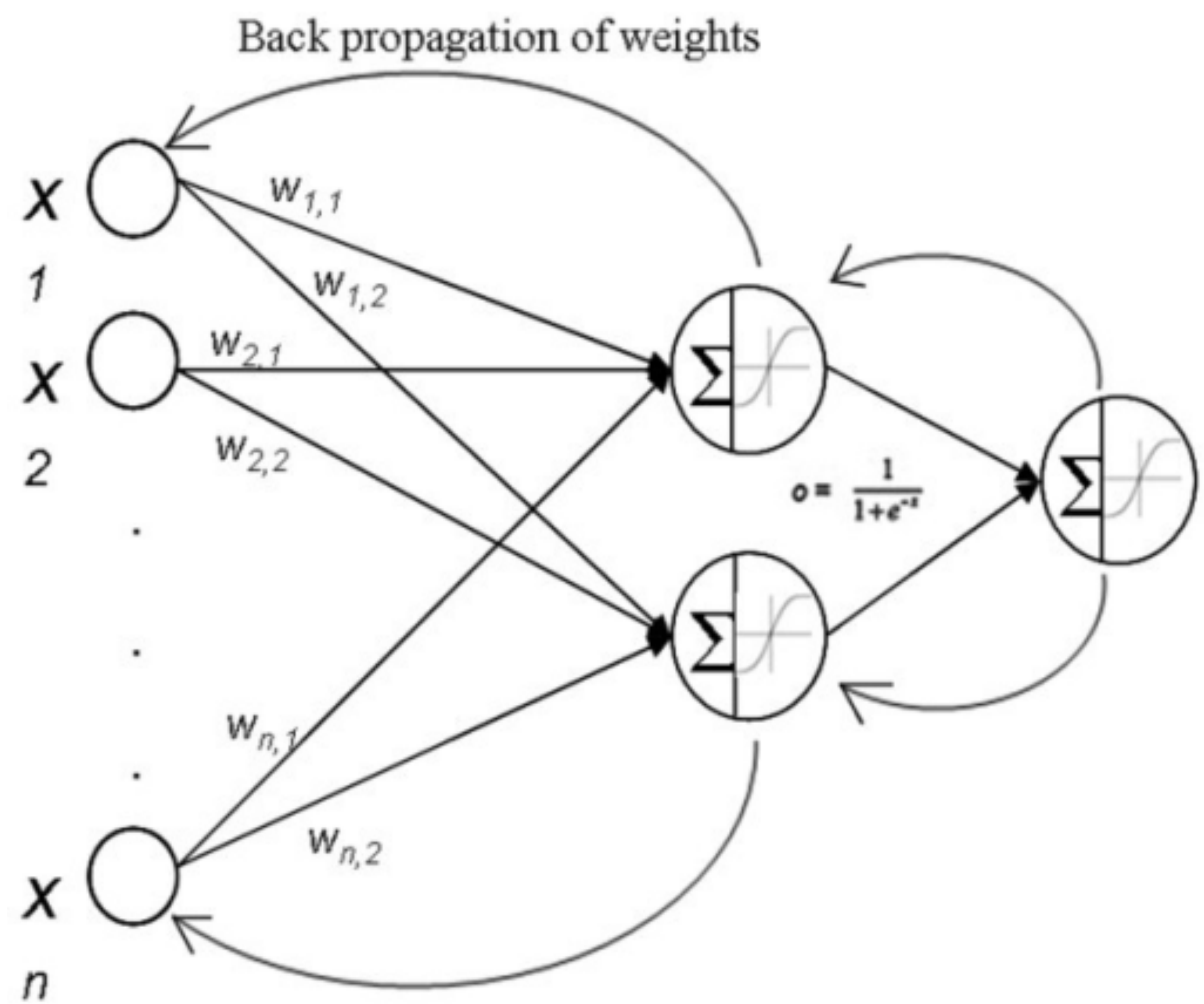
- Composition of perceptrons, connected in different ways and operation on different activation functions
- Each unit of layer t is typically connected to every unit of the previous layer t - 1



http://neuralnetworksanddeeplearning.com/chap1.html

Backpropogation Algorithm

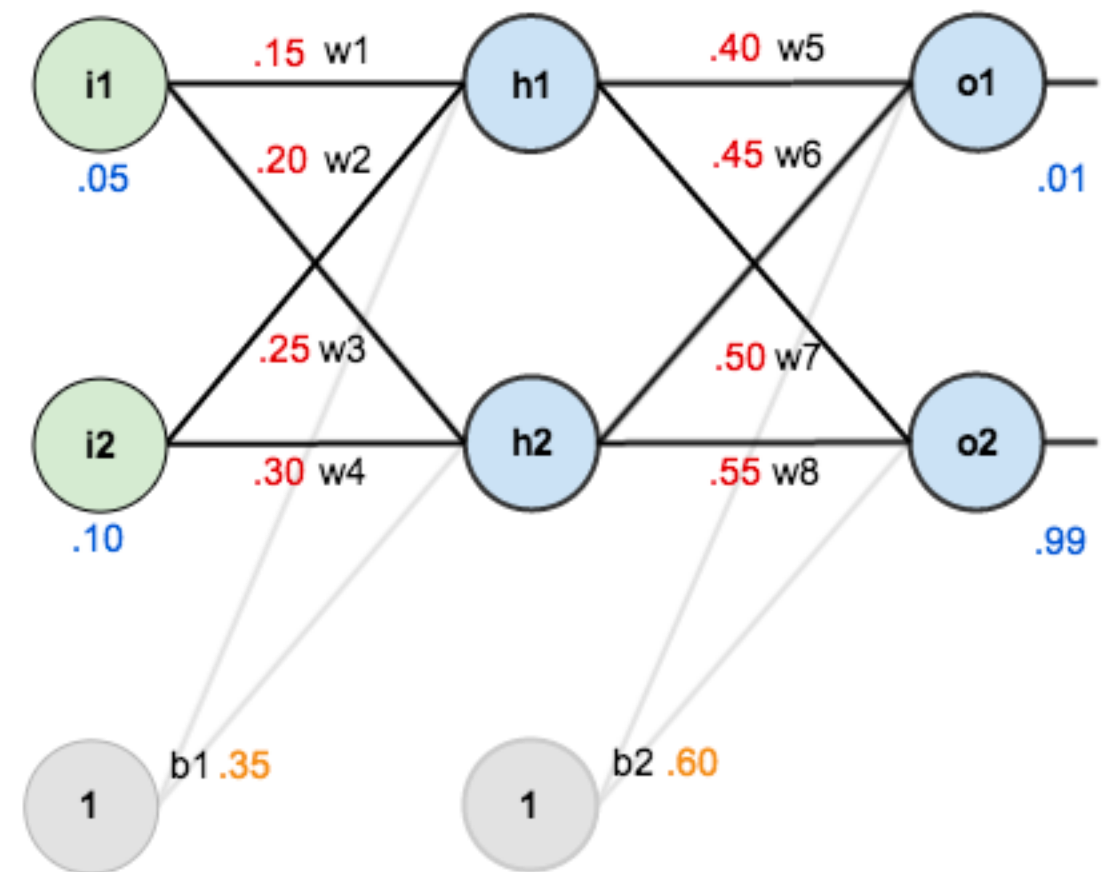
- Method of training neural network via gradient descent
- Calculate error at output layer for each training example
- Propagate errors backward through the network and update the weights accordingly



https://openi.nlm.nih.gov/imgs/512/121/2716495/PMC2716495_bcr2257-1.png

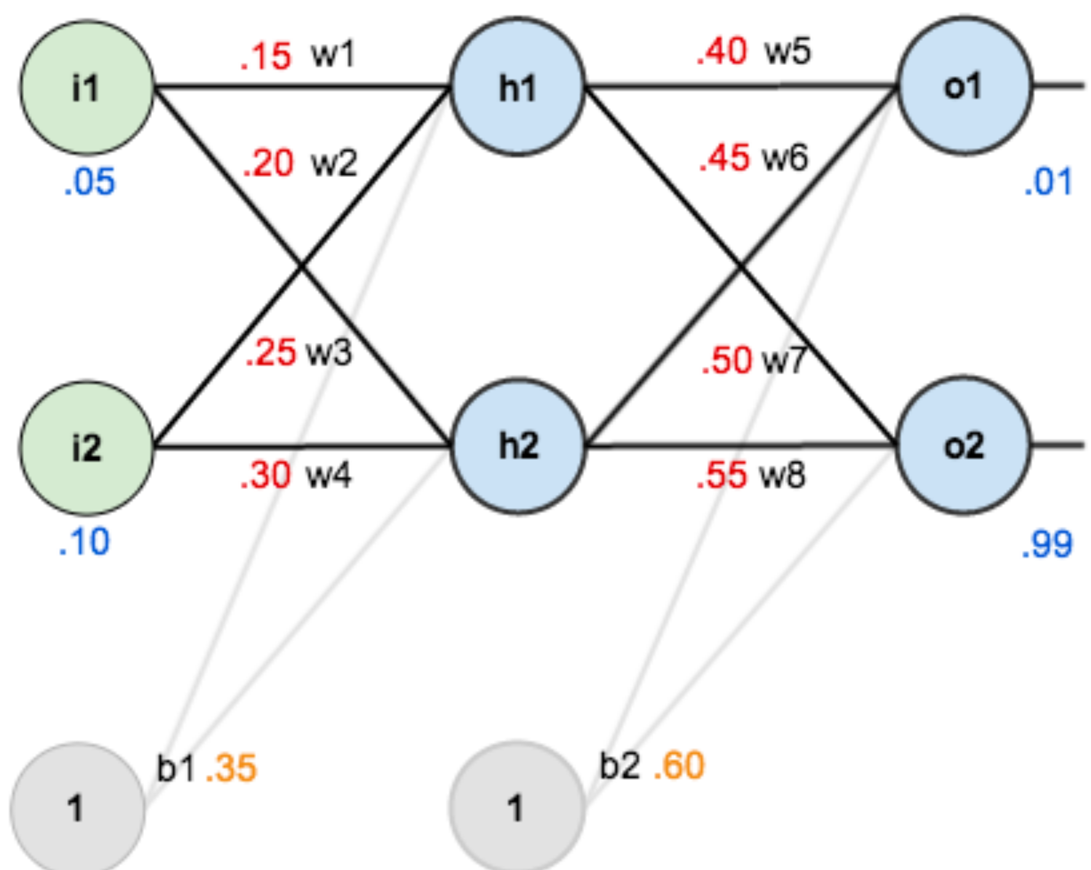
Example: Backpropogation

- Simple neural network with two inputs, two hidden neurons and two output neurons
- Activation function is logistic function
- Imagine single training set with inputs (0.05, 0.10) and want output to be 0.01 and 0.09 and want to minimize squared error



https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Example: Backpropogation (2)



$$h_1 = w_1 i_1 + w_2 i_2$$

$$h_2 = w_3 i_1 + w_4 i_2$$

$$o_1 = \frac{1}{1 + e^{-(w_5 h_1 + w_6 h_2)}}$$

$$o_2 = \frac{1}{1 + e^{-(w_7 h_1 + w_8 h_2)}}$$

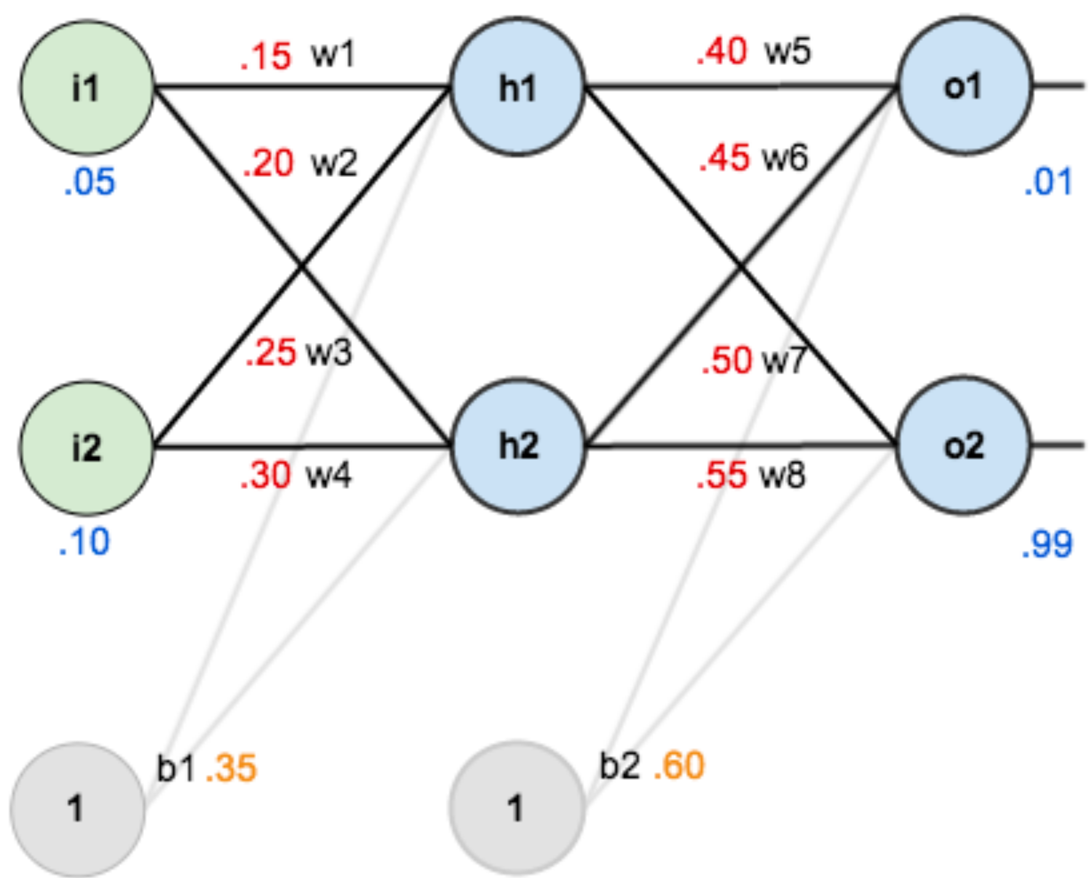
$$e_{\hat{o}_1} = \frac{1}{2}(o_1 - \hat{o}_1)^2 = 0.274811083$$

$$e_{\hat{o}_2} = 0.023560026$$

$$e_{total} = e_{\hat{o}_1} + e_{\hat{o}_2}$$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Example: Backpropogation (3)



$$\frac{\partial e_{total}}{\partial w_5} = \frac{\partial e_{total}}{\partial \hat{o}_1}\frac{\partial \hat{o}_1}{\partial \hat{h}_1}\frac{\partial \hat{h}_1}{\partial w_5}$$

$$= -(o_1 - \hat{o}_1)\hat{o}_1(1 - \hat{o}_1)\hat{h}_1$$

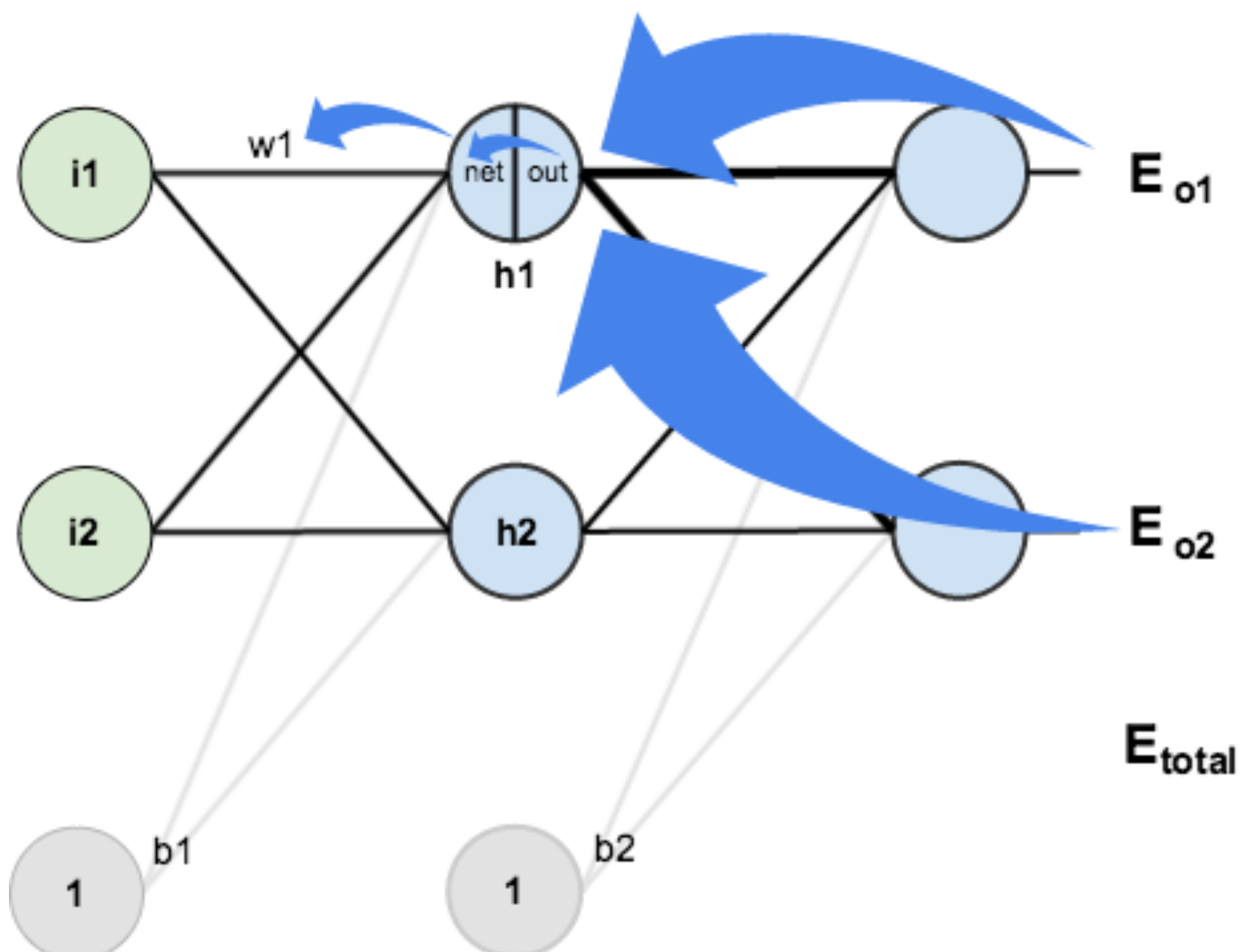
$$w_5^+ = w_5 - \eta\frac{\partial e_{total}}{\partial w_5}$$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Example: Backpropogation (4)

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$

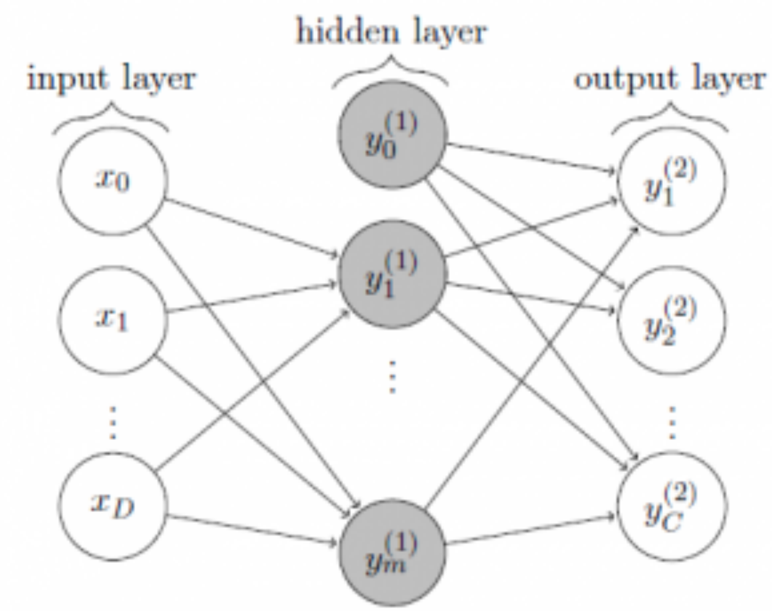
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

MNIST Dataset

- Scanned images of handwritten digits
- 28 x 28 greyscale images
- Training data
  - 60,000 images
  - 250 people
- Test Data
  - 10,000 images
  - Different 250 people

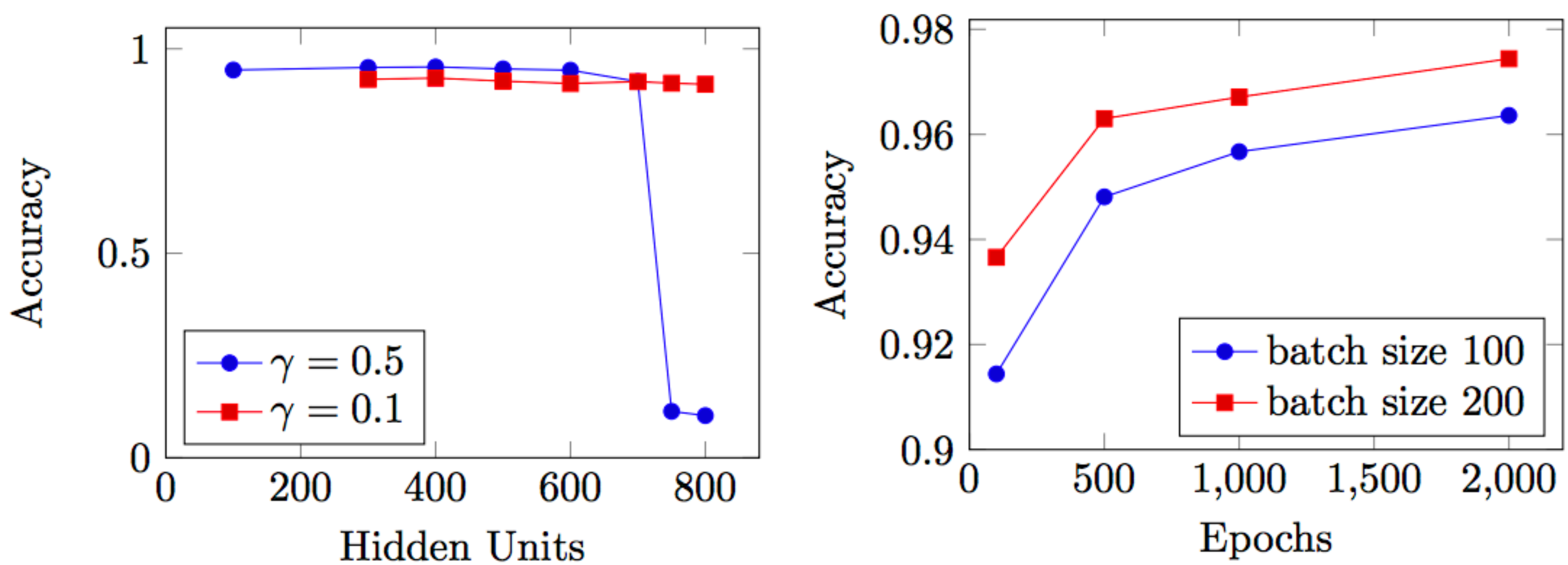
Experiment: 2 Layer Perceptron

- 784 input units, variable number of hidden units, and 10 output units
- Activation function = logistic sigmoid
- Sum of squared error function & backpropogation algorithm
  - Stochastic variant of mini-batch training



http://davidstutz.de/recognizing-handwritten-digits-mnist-dataset-twolayer-perceptron

Experiment: 2 Layer Perceptron



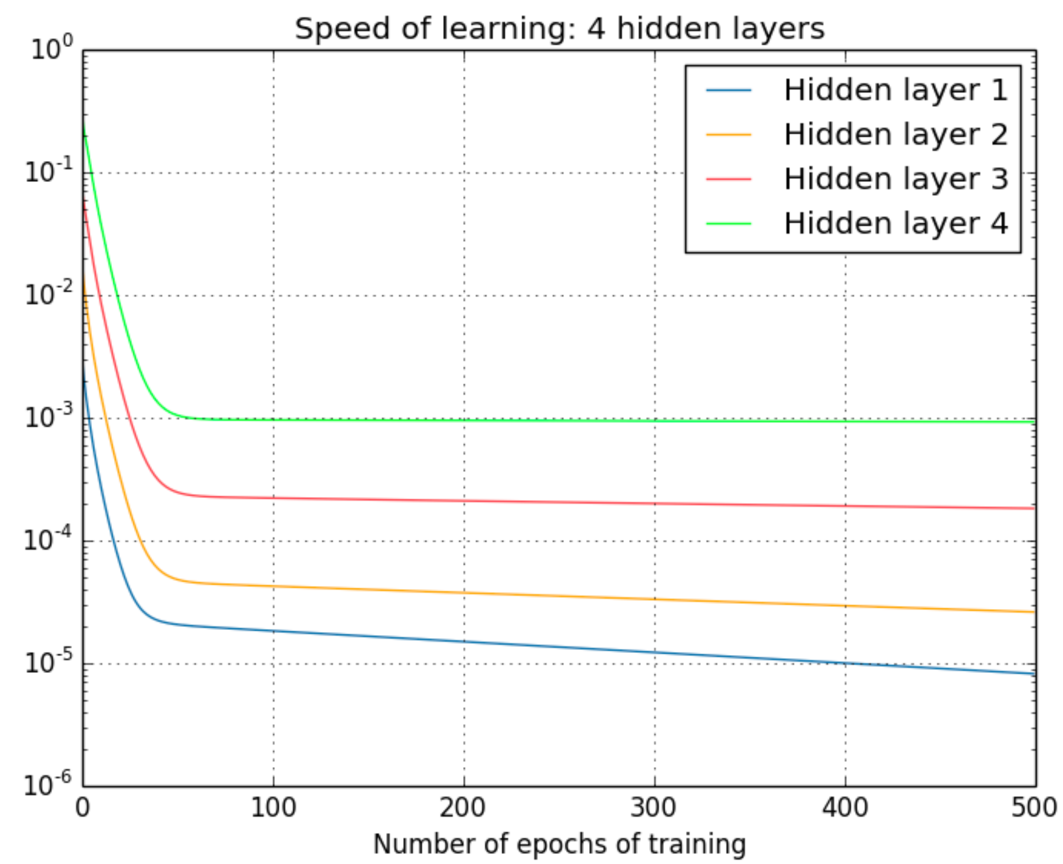
(a) 500 epochs with batch size 100.

(b) 500 epochs with learning rate $\gamma = 0.5$.

http://davidstutz.de/wordpress/wp-content/uploads/2014/03/seminar.pdf

Obstacles to Deep MLPs

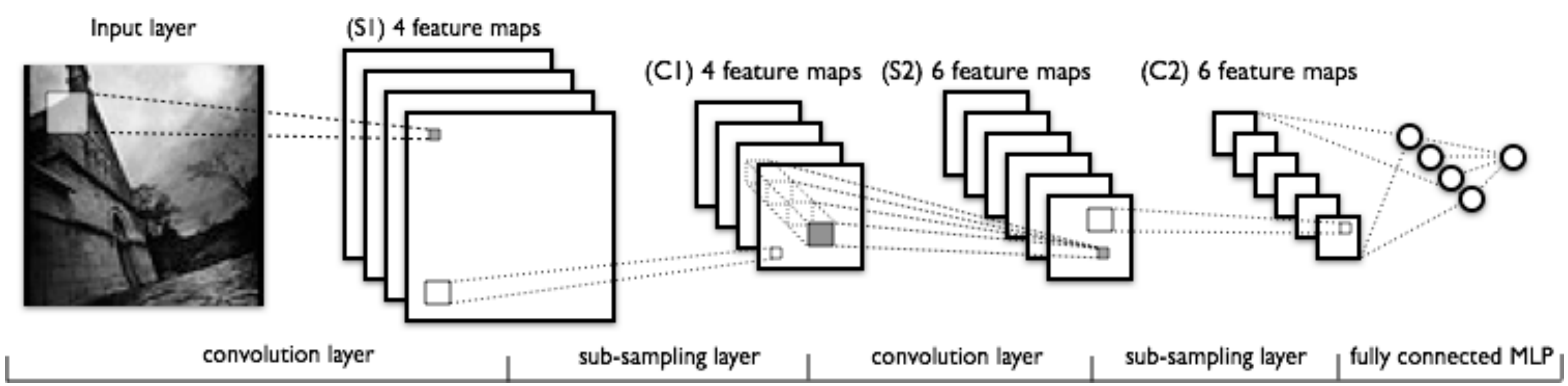
- Requires lots of labeled training data
- Computationally extremely expensive
  - Vanishing & unstable gradients
- Difficult to tune
  - Choice of architecture (layers + activation function)
  - Learning algorithm
  - Hyperparameters



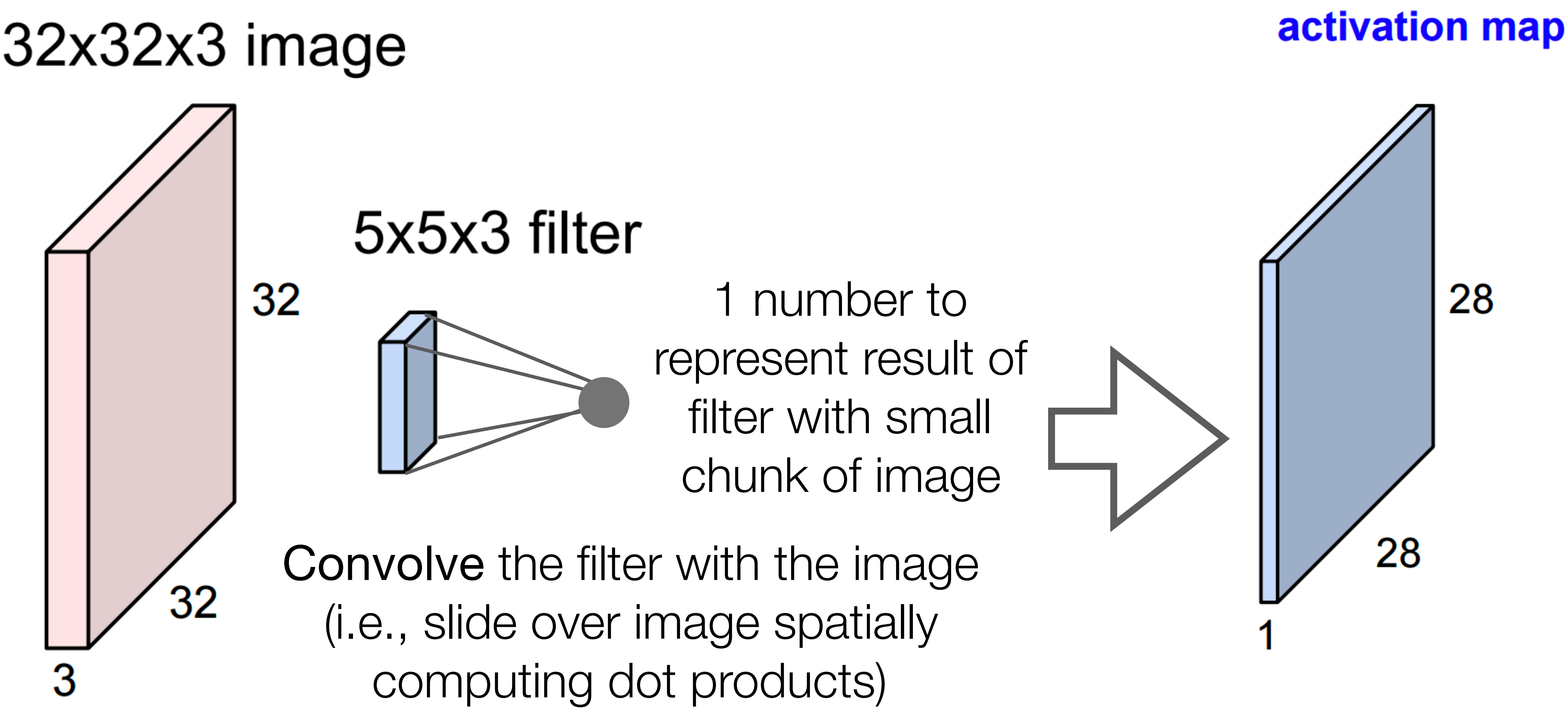
http://neuralnetworksanddeeplearning.com/chap5.html

Convolutional Neural Networks (CNN)

- Specialized neural network for processing known, grid-like topology
  - Powerful model for image, speech recognition
- Use convolution instead of general matrix multiplication in one of its layers



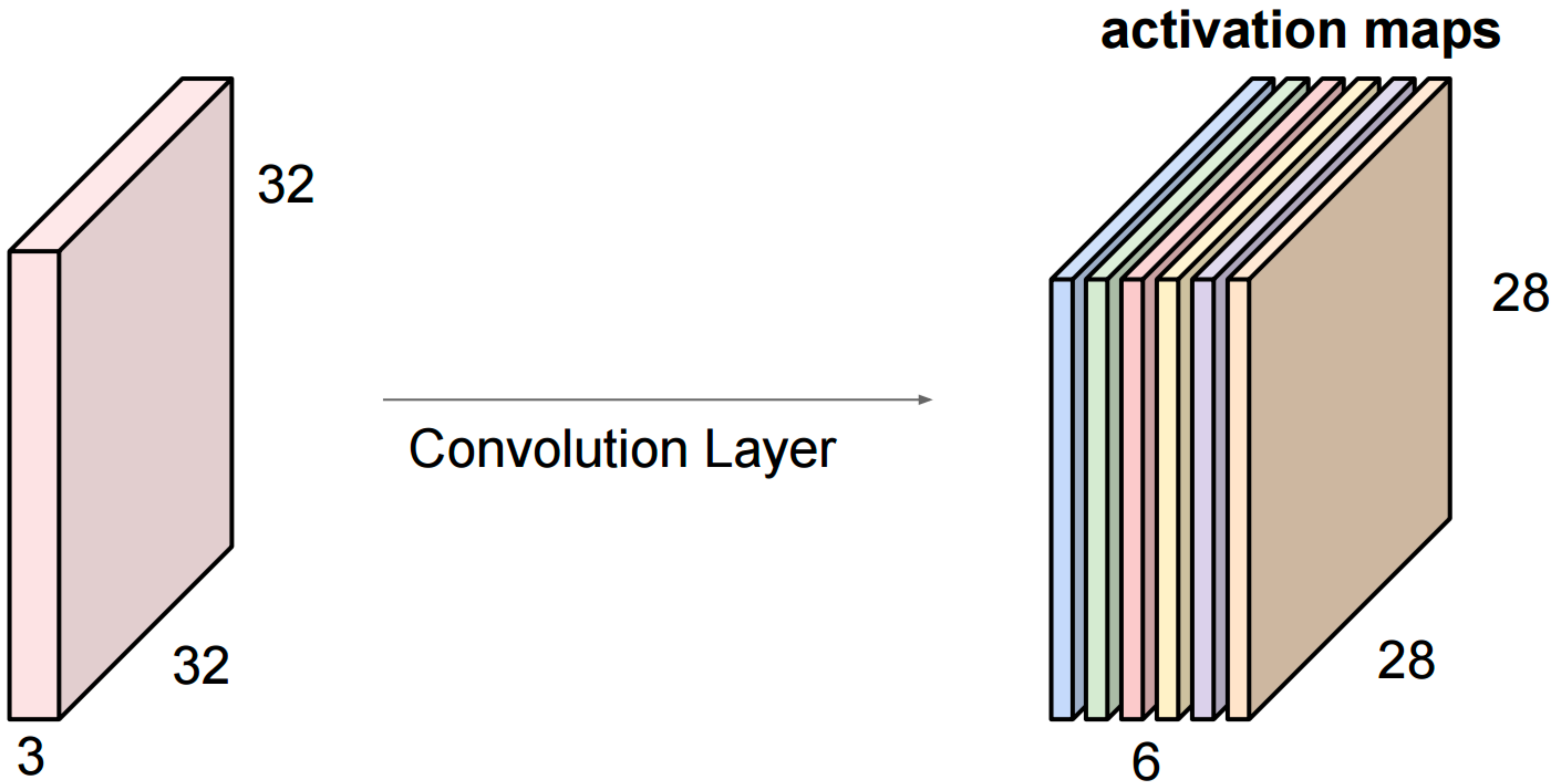
Convolution Layer



Convolve the filter with the image (i.e., slide over image spatially computing dot products)

http://vision.stanford.edu/teaching/cs231n/slides/winter1516_lecture7.pdf

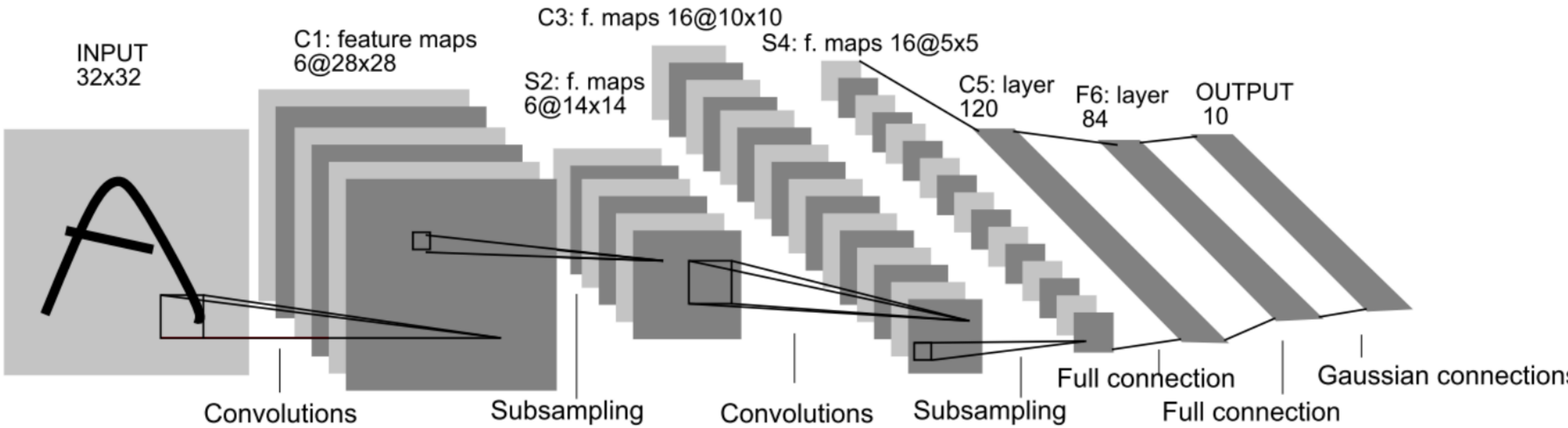
Convolution Layer (Multiple Filters)



Stacking up multiple filters yields “new image”

http://vision.stanford.edu/teaching/cs231n/slides/winter1516_lecture7.pdf

LeNet 5 [LeCun et al., 1998]



- 32 x 32 pixel with largest character 20 x 20
- Black and white pixel values are normalized to get mean of 0, standard deviation of 1
- Output layer uses 10 RBF (radial basis activation function), one for each digit

CNN: MNIST Dataset Results

- Original dataset (60,000 images)
  - Test error = 0.95%
- Distorted dataset (540,000 artificial distortions + 60,000 images)
  - Test error = 0.8%

Misclassified examples

Why is CNN Successful?

Compared to standard feedforward neural networks with similarly-sized (5-7) layers

- CNNS have much fewer connections and parameters => easier to train
  - Traditional fully-connected neural network is almost impossible to train when initialized randomly
- Theoretically-best performance is likely only slightly worse than vanilla neural networks

Recurrent Neural Networks (RNN)

- Family of neural networks for processing sequential data
- Shares the same weights across several time steps

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

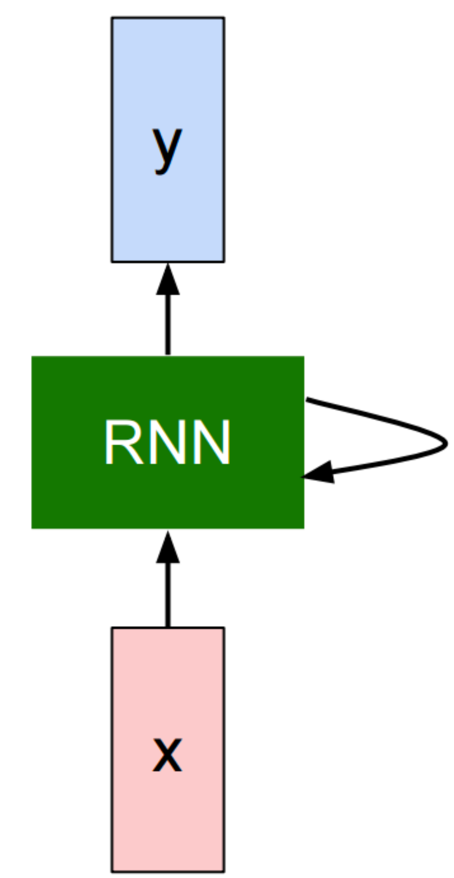
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

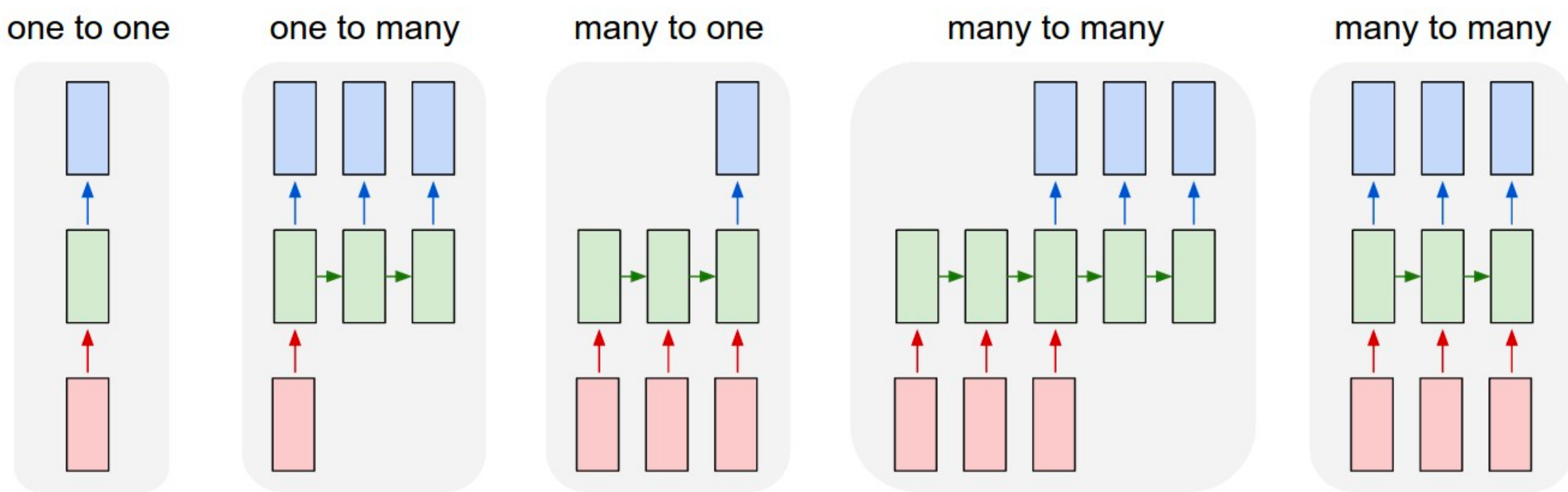
some function with parameters W

old state

input vector at some time step



The Need for Sequences



"Vanilla" NN: fixed-sized input to fixed-size output

Sequence output (e.g., image captioning task where image becomes sequence of words)

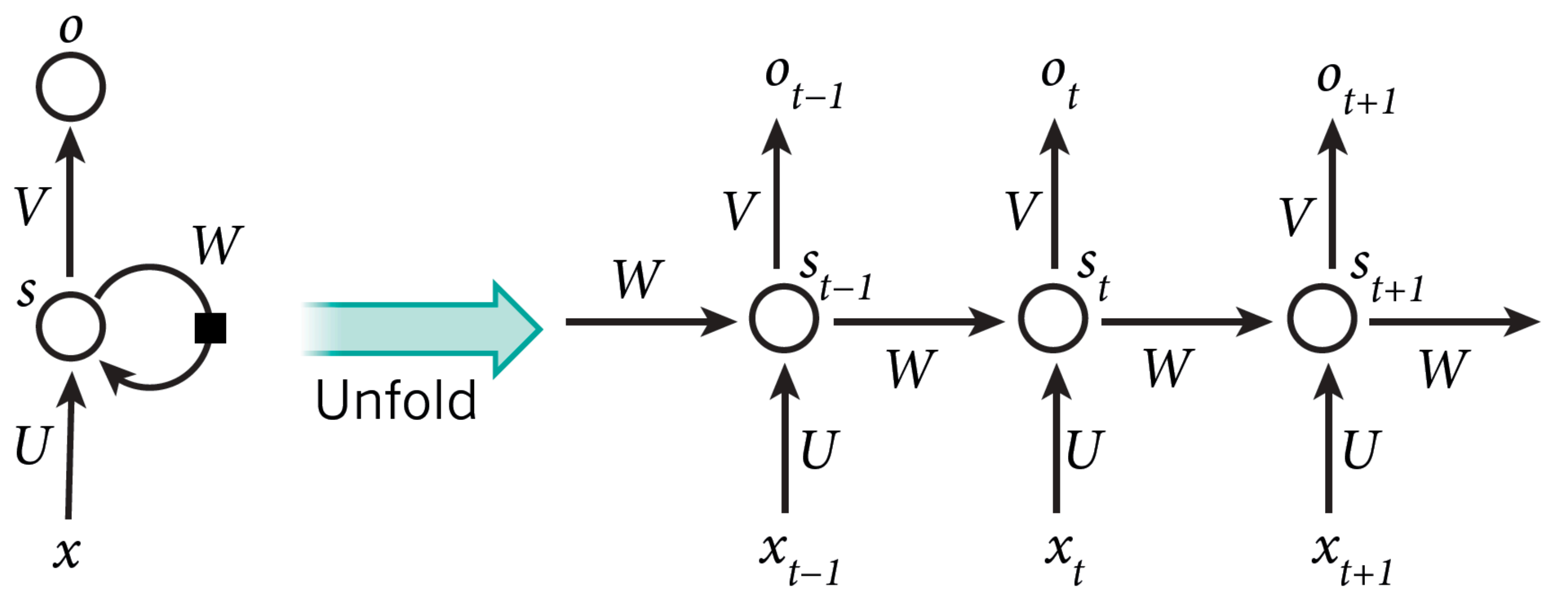
Sequence input (e.g., sentiment analysis)

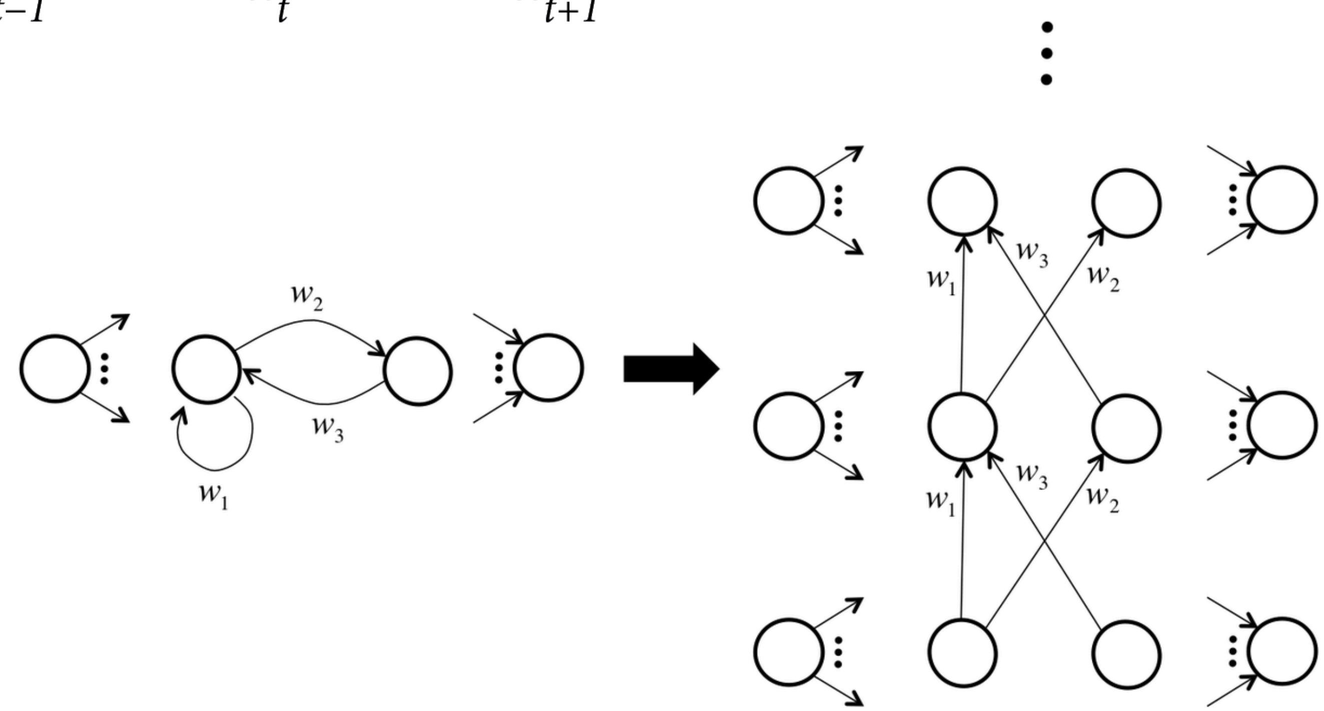
Sequence I/O (e.g., machine translation)

Synced sequence I/O (e.g., video classification on frame level)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Unfolding RNN for Backpropogation



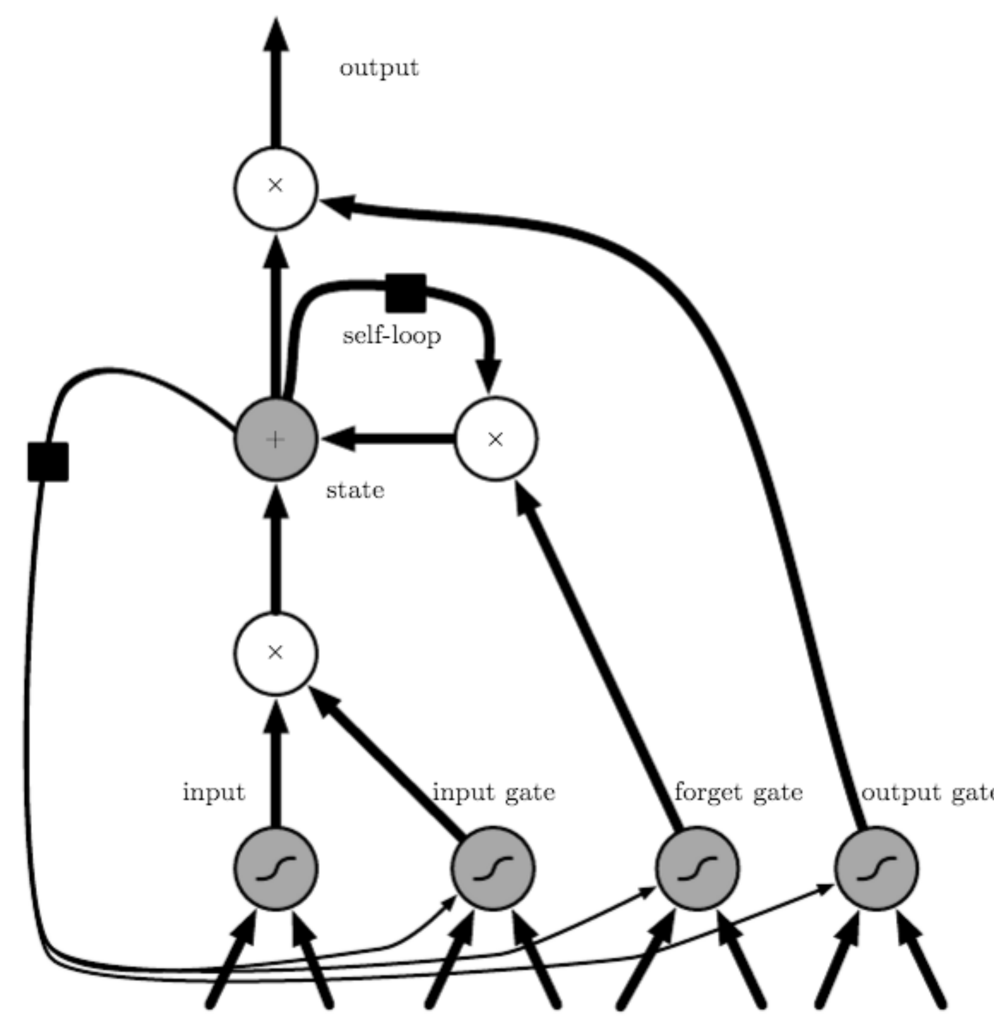


Long-Term Dependency Problems

- Appeal of RNN is to connect previous information to present task
- Gap between relevant information and point of needing it can be large (e.g., word prediction for a sentence like I grew up in France … I speak fluent ___)
- Long-range dependencies are difficult to learn because of vanishing gradient or exploding gradient problem (depending on the activation function)

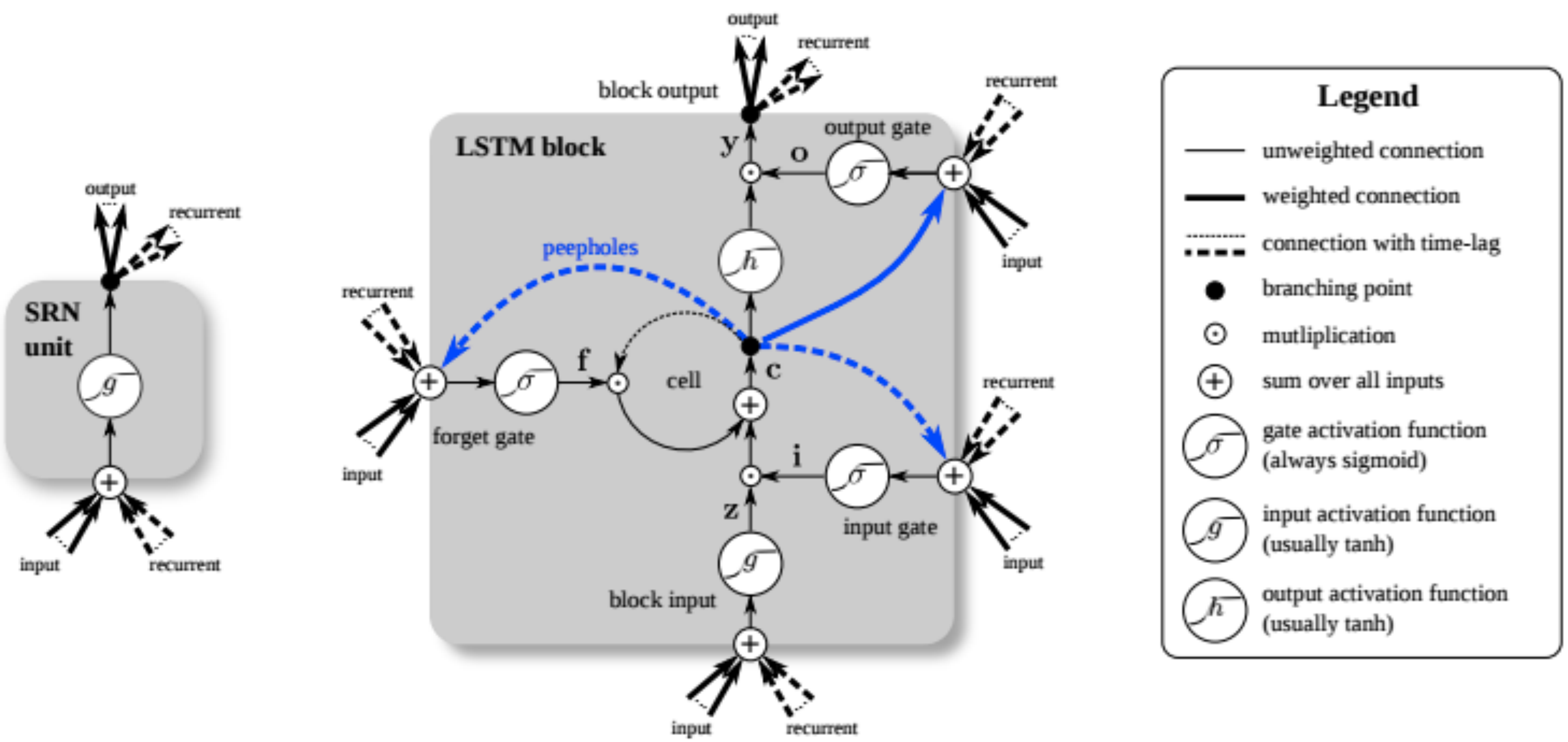
Long Short-Term Memory Units (LSTM)

- Introduction of a new structure called memory cell
- 4 components: input gate, a neuron with a self-recurrent connection, a forget gate, and an output gate
- Ability to remove or add information to the cell state through the gates



http://www.deeplearningbook.org/contents/rnn.html

Simple RNN vs LSTM



http://deeplearning4j.org/lstm.html

Experiment: Shakespearean Writing

- Download all works of Shakespeare into single file
- Train 3-layer RNN with 512 hidden nodes on each layer
- Create samples for both speaker's names and the contents

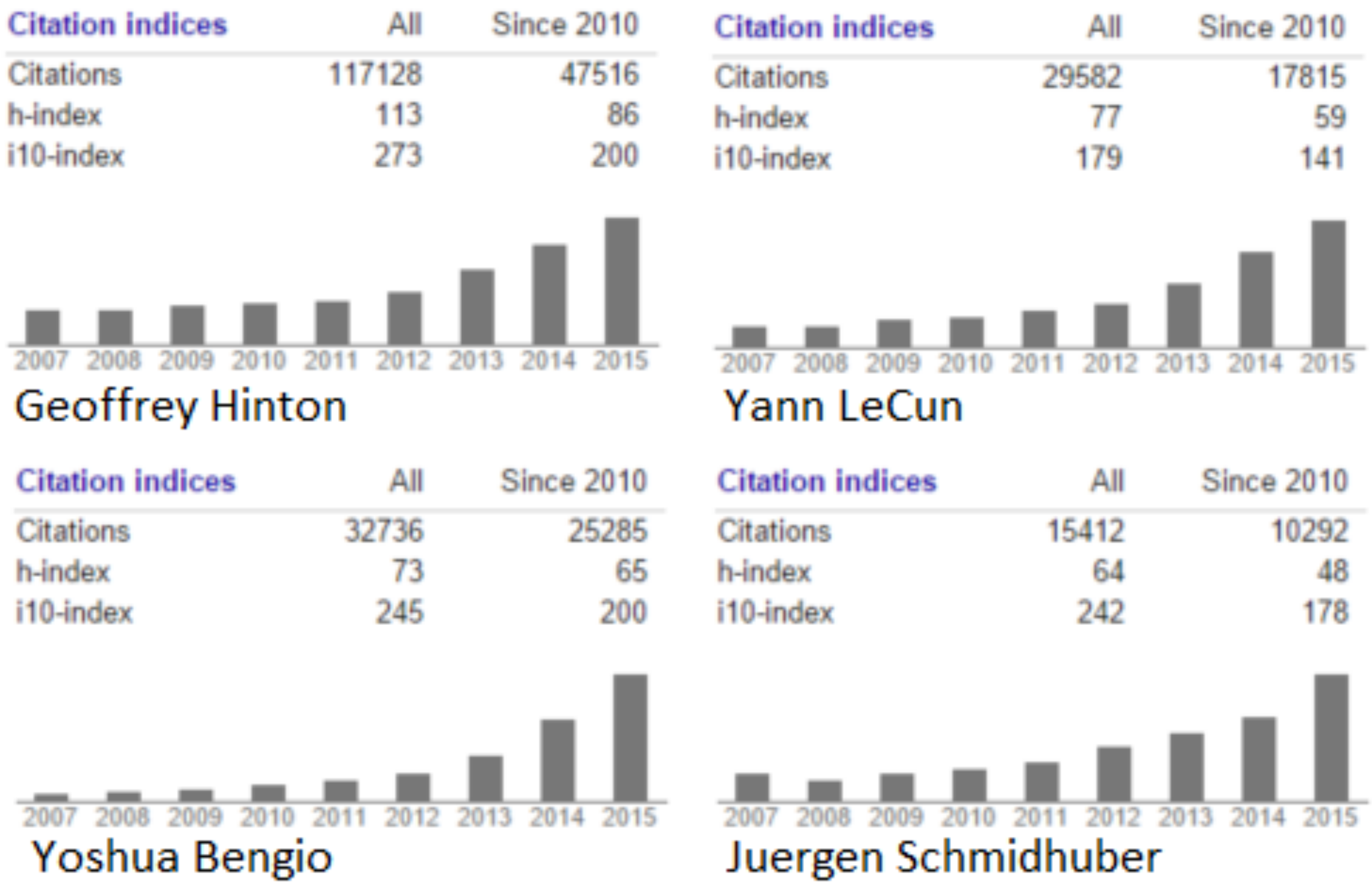
```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Deep Learning: The Dark Ages

- Early 2000s: failure of backpropogation + ascent of SVMs led to a slump
- Hinton & Bengio hatched plan to “rebrand” neural networks with deep learning
- Resurgence with “A fast learning algorithm for deep belief nets” [Hinton et al., 2006]
  - Clever way to initialize neural networks rather than randomly
- Followed by “Greedy layer-wise training of deep networks” [Bengio et al., 2007]

Deep Learning Rises Again



http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning-part-4/

Deep Learning Rises Again (2)

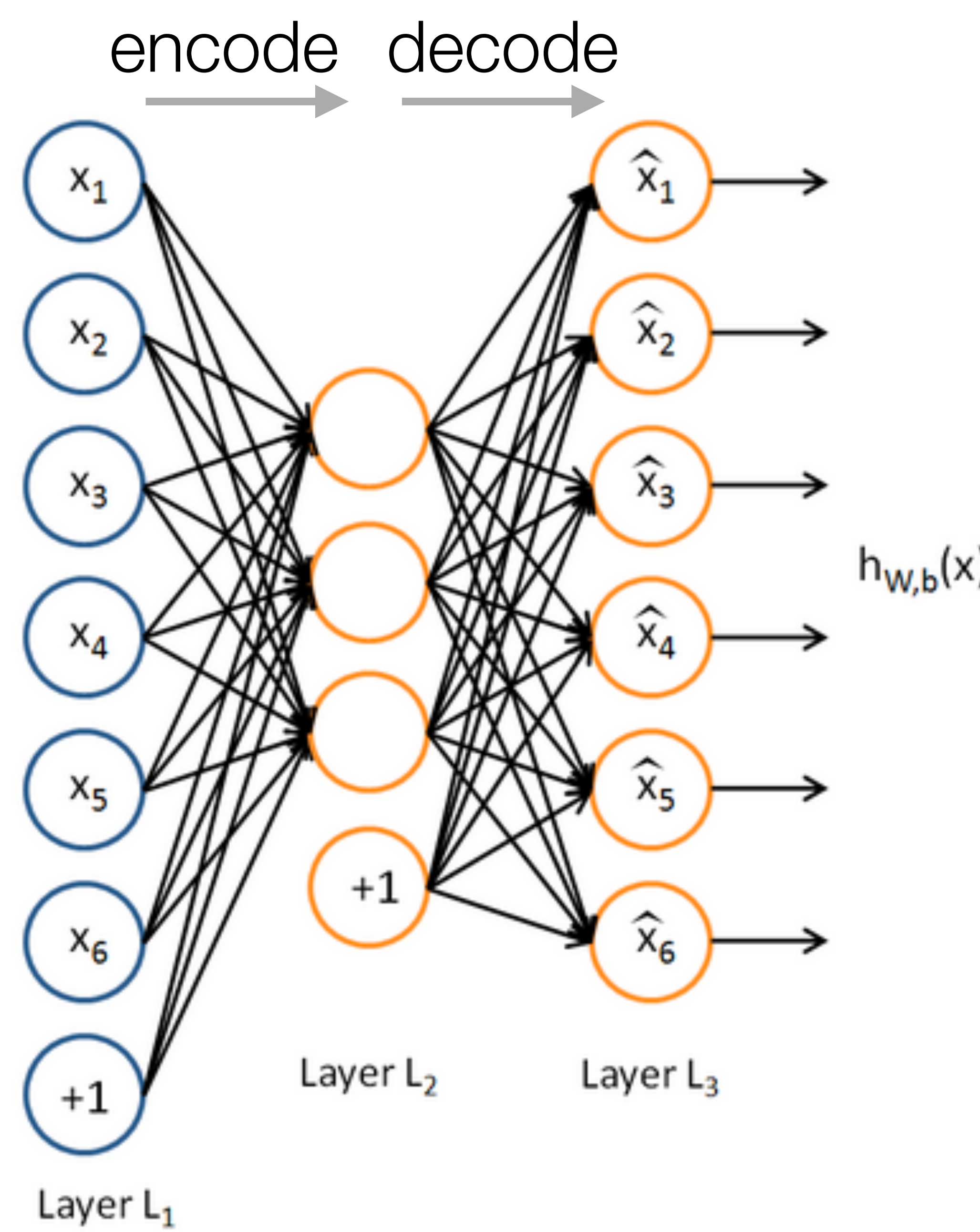
- Labeled datasets were thousands of times too small
    - Unsupervised pre-training could help mitigate bad initialization
- Computers were millions of times too slow
- Weights were initialized in a stupid way
- Used wrong type of non-linearity

Deep learning = lots of training data + parallel computation + scalable, smart algorithms

http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning-part-4/

Autoencoder

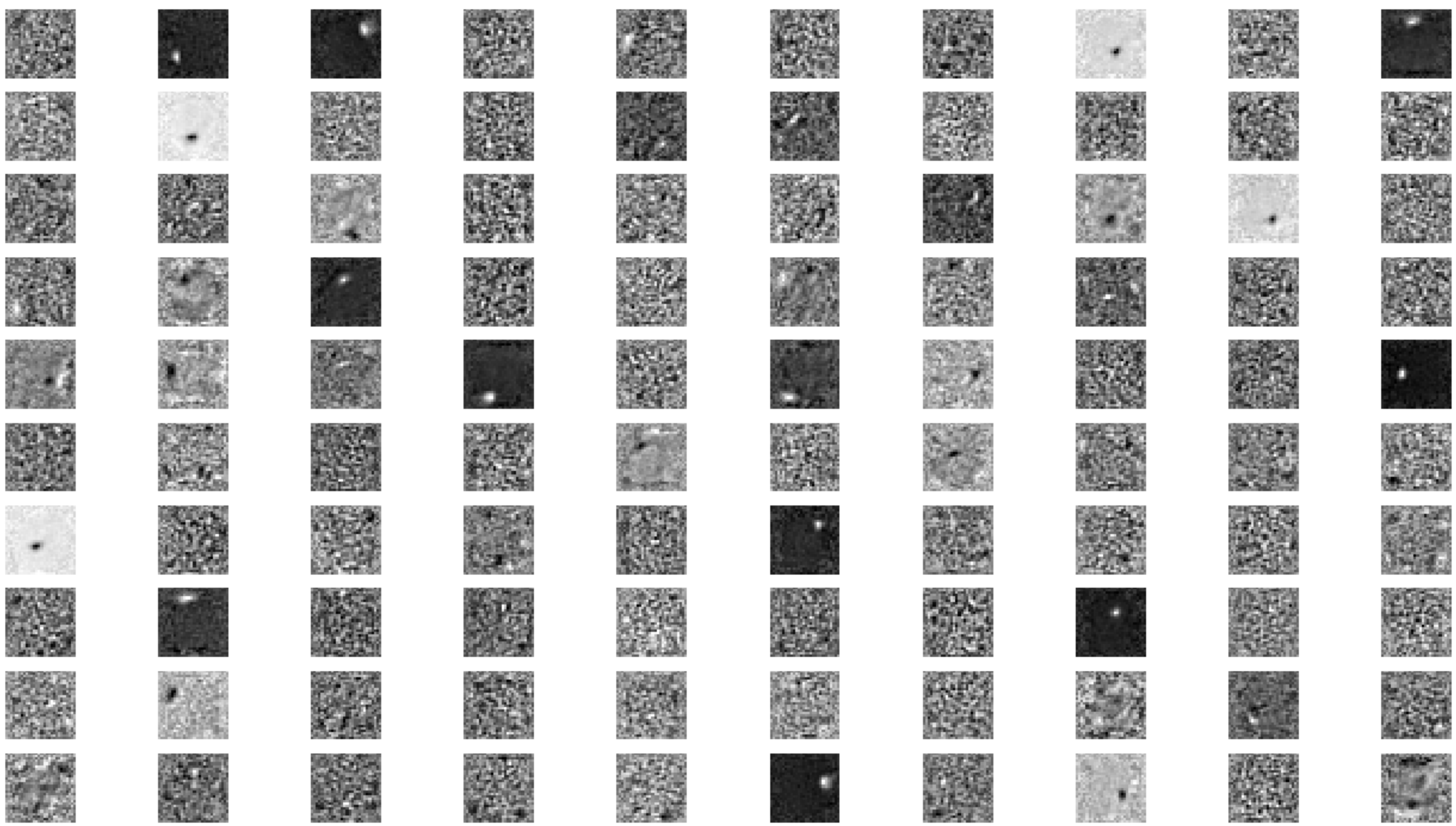
- MLP only works with labeled training examples
- Autoencoder learns compressed, distributed representation (encoding) of the dataset
- Aim to “recreate” the input



http://ufldl.stanford.edu/wiki/index.php/Autoencoders_and_Sparsity

Autoencoder: MNIST Results

500 hidden units with 20 epochs and mini batch size of 20

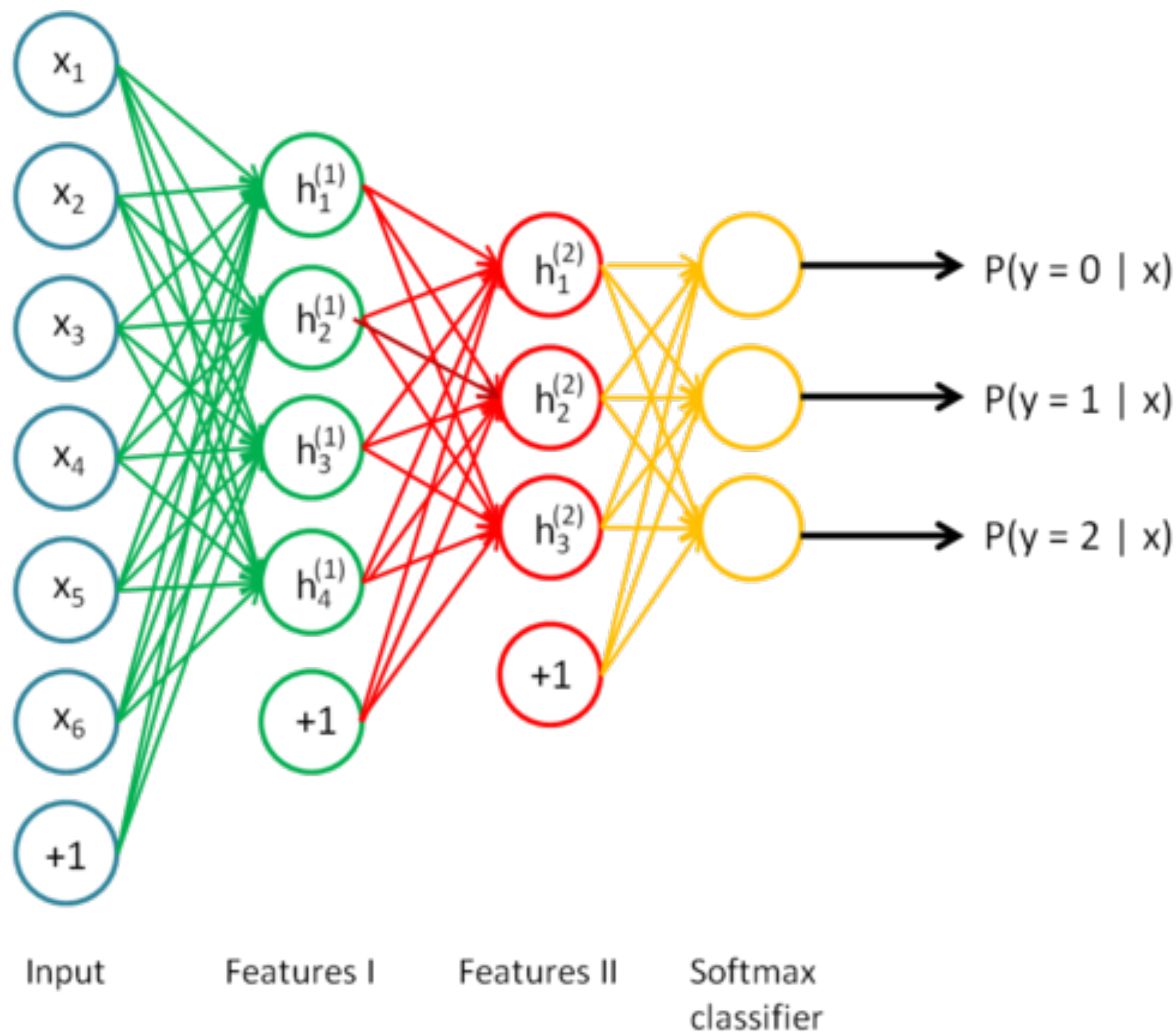


https://triangleinequality.wordpress.com/2014/08/12/theano-autoencoders-and-mnist/

Stacked Autoencoders

- Network of multiple stacked auto encoders
- Can capture “hierarchical grouping” or “part-whole decomposition” of input
- Greedy training algorithm
  - Train first autoencoder using backpropogation (to learn raw inputs)
  - Train second layer autoencoder using output of first layer to learn these secondary features

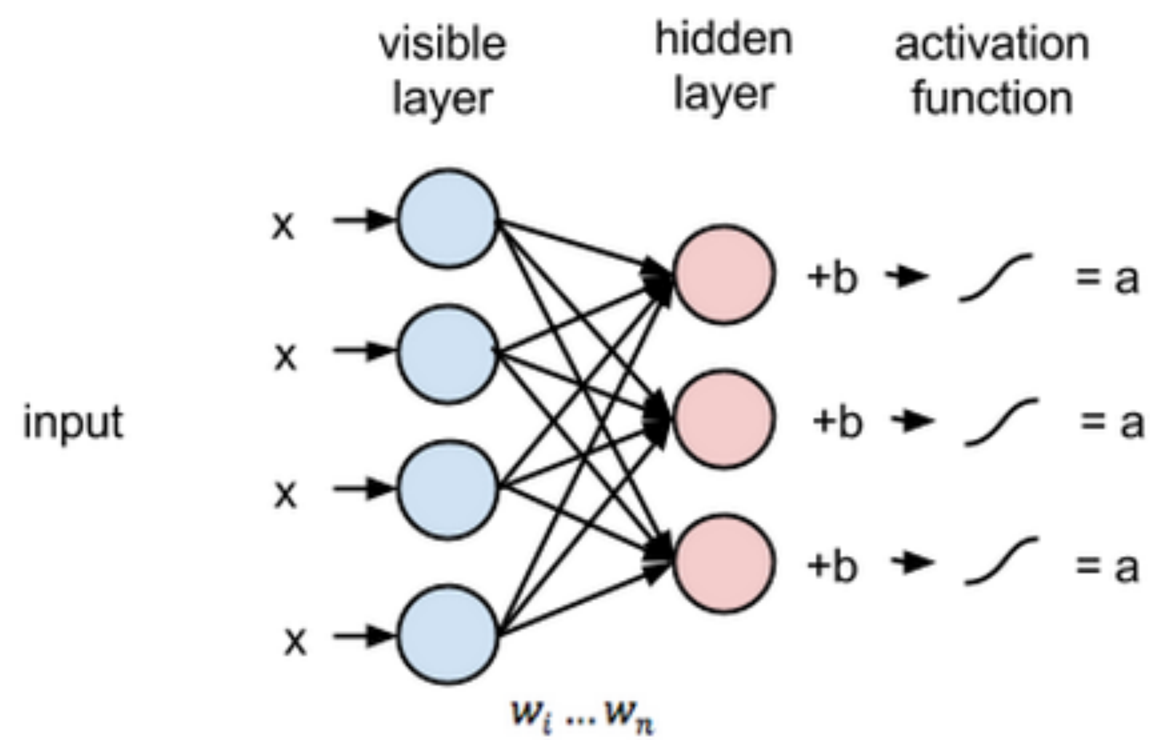
Stacked Autoencoders: Classification



http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

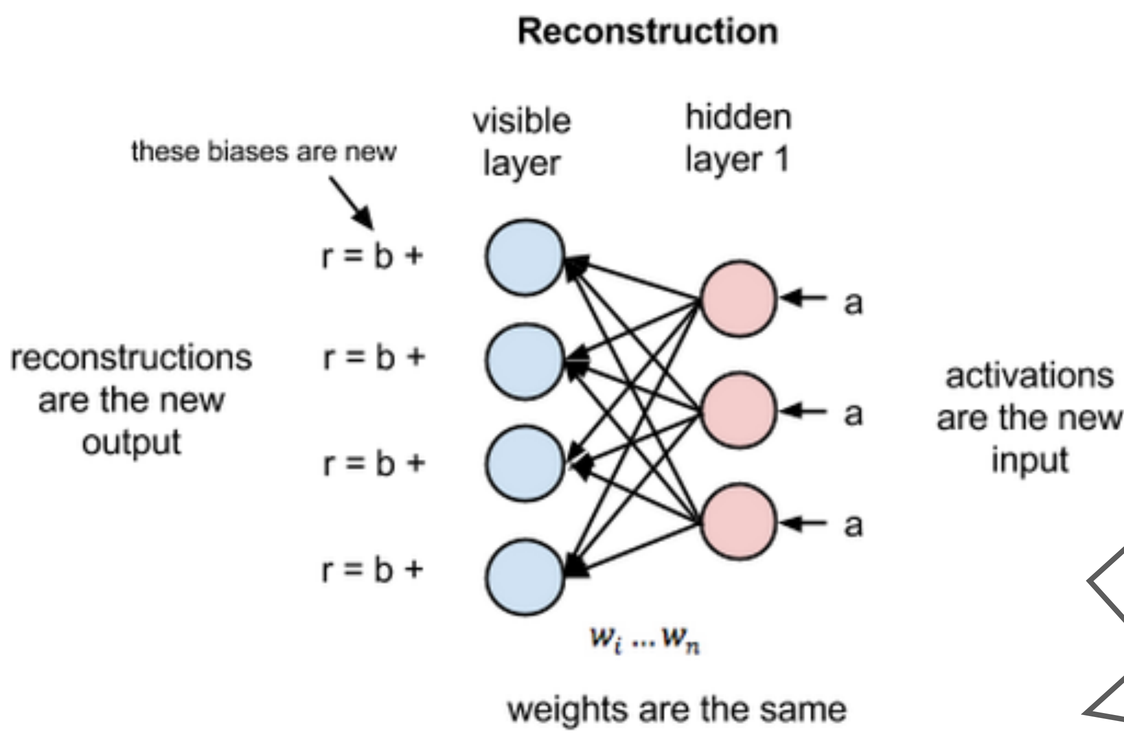
Restricted Boltzmann Machines (RBM)

- Generative stochastic neural network that can learn a probability distribution over its set of inputs
- Restrict connectivity to make learning easier
  - One layer of hidden units
  - No connections between hidden units

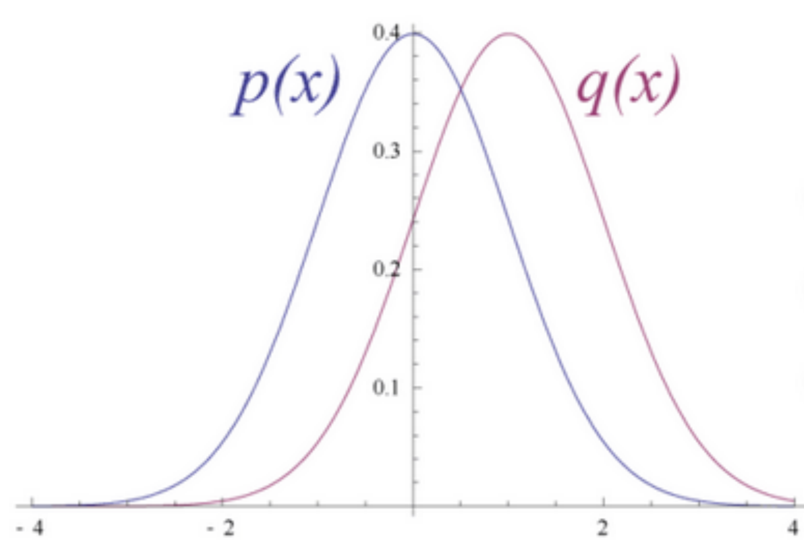


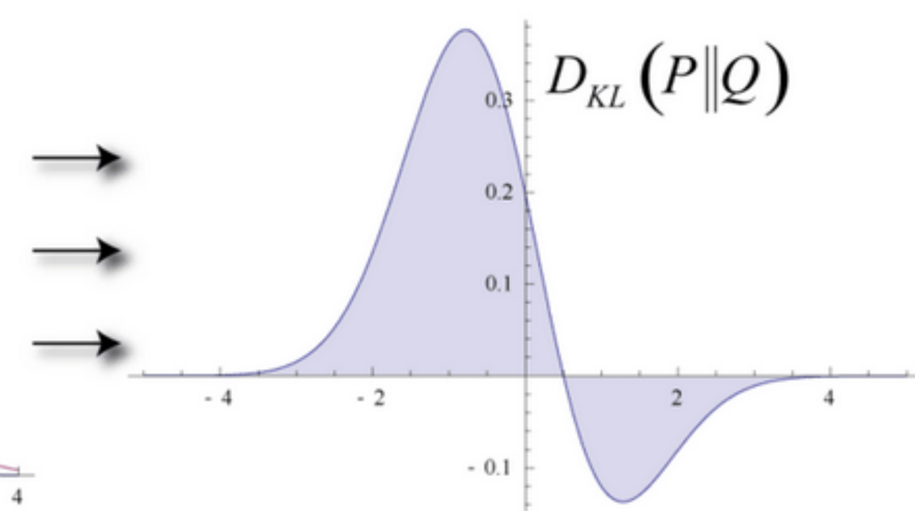
http://deeplearning4j.org/restrictedboltzmannmachine.html

RBM: Reconstruction via Backpropogation



minimize KL divergence between probability distribution and ground-truth distribution

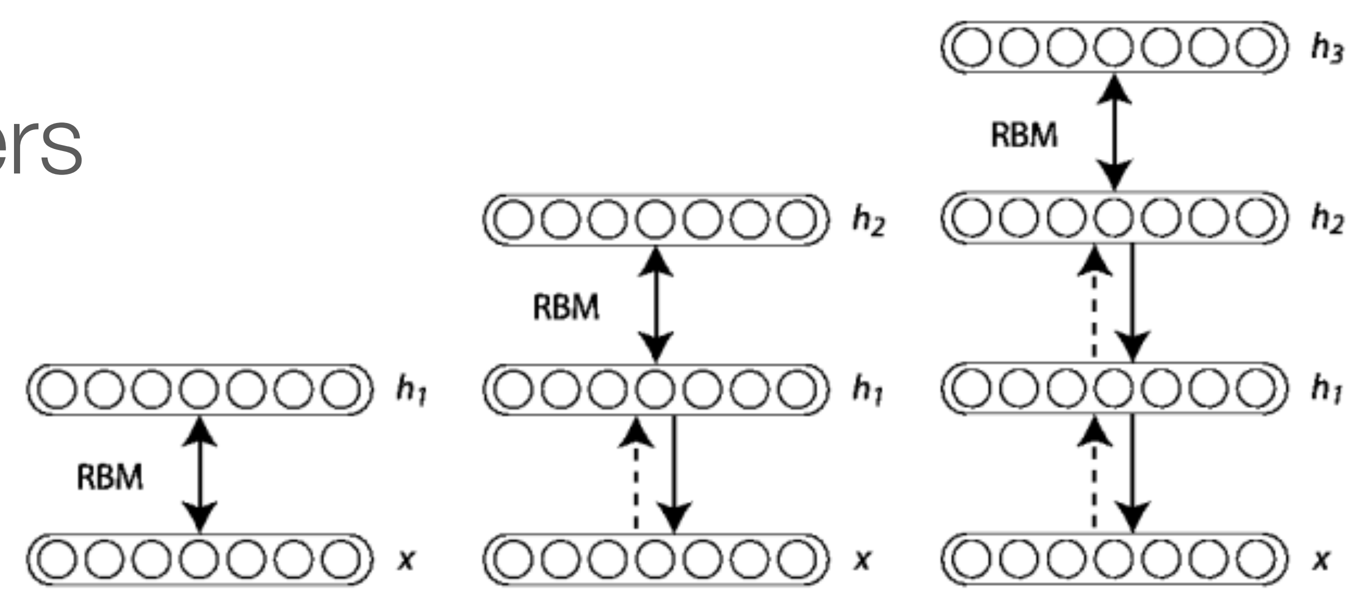




http://deeplearning4j.org/restrictedboltzmannmachine.html

Deep Belief Network (DBN)

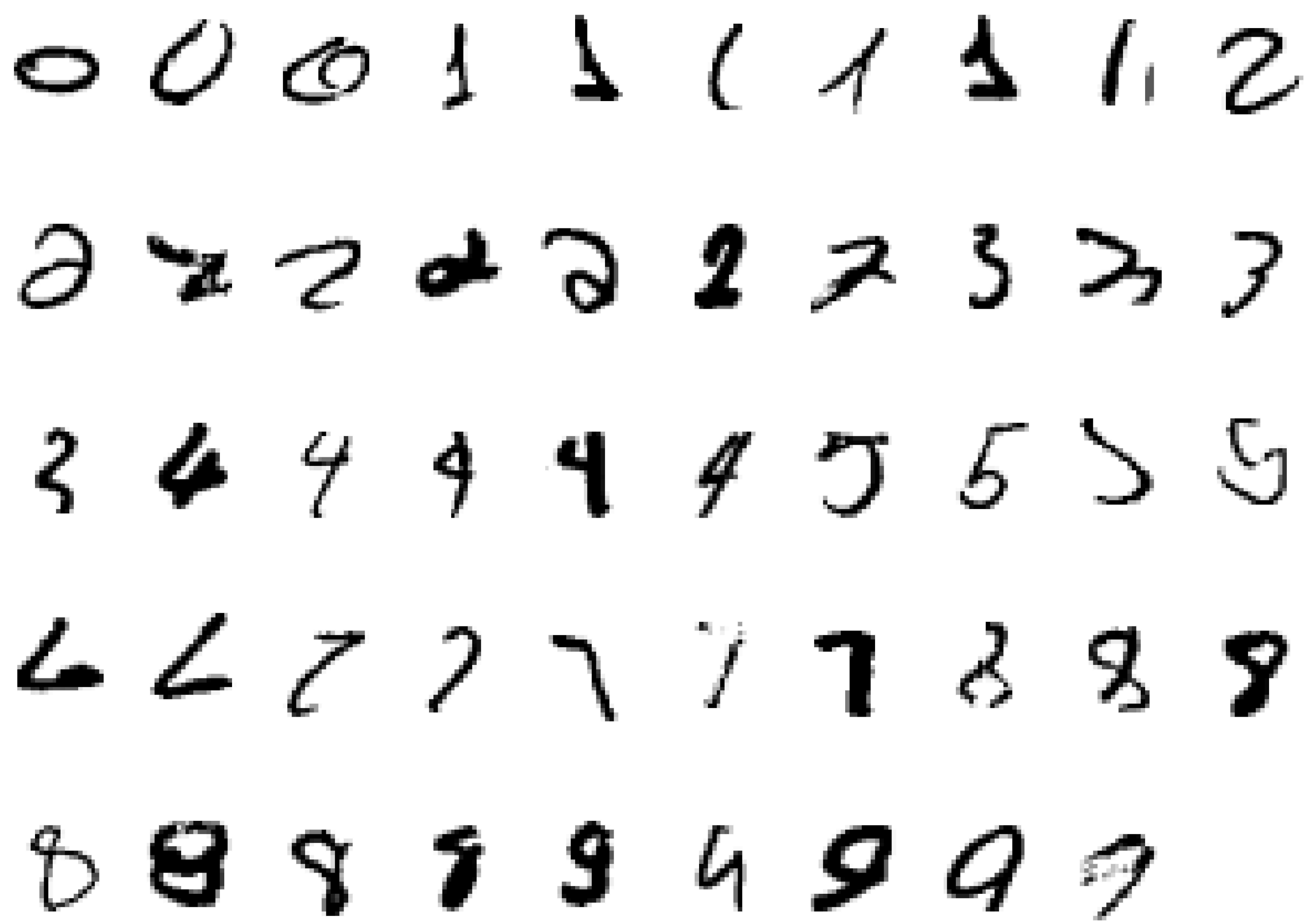
- Probabilistic generative models
- Deep architecture — multiple layers
  - Each layer contains high-order correlations between the activities of hidden features in the layer below
  - Stack RBM to get layers



http://www.pyimagesearch.com/wp-content/uploads/2014/09/deep_belief_network_example.png

DBN: MNIST Dataset Results

Examples of correctly recognized handwritten digits that the network hadn't seen before



DBN: MNIST Dataset Results (2)

Model	Test Error
Generative model via RBM	1.25%
SVM [Decoste et al.]	1.4%
Backpropogation with 1000 hidden units [Platt]	1.6%
Backpropogation with 500 —> 300 hidden units	1.6%
K-nearest neighbor	~3.3%

https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf

Deep Learning Resources

- Website with variety of resources and pointers at deeplearning.net
- Deep Learning Tutorial by Stanford (http://ufldl.stanford.edu/tutorial/)
- Neural Networks and Deep Learning online book (http://neuralnetworksanddeeplearning.com/)
- Deep Learning book by Goodfellow, Bengio, and Courville (http://www.deeplearningbook.org/)
- NIPS 2015 Tutorial by Hinton, Bengio & LeCun (http://www.iro.umontreal.ca/~bengioy/talks/DL-Tutorial-NIPS2015.pdf)

Deep Learning Resources (2)

- Deep Learning for Java (http://deeplearning4j.org/)
- ConvNetJS (http://cs.stanford.edu/people/karpathy/convnetjs/)
- Andrej Karpathy's Blog on Neural Networks (http://karpathy.github.io/)
- Colah's Blog on Neural Networks (https://colah.github.io/)

Deep Learning Toolkits

- TensorFlow (by Google)
- Theano (developed by academics)
- Torch (written by Lua)
- Caffe

For a reasonable comparison of the frameworks, see https://github.com/zer0n/deepframeworks/blob/master/README.md