From perceptrons to Alpha Go and beyond

An overview of Deep Learning

Dennis Wilson

19/01/2018

IRIT

Background

Movie recommendation system

Movie name	Mary's rating	John's rating	I liked it
Lord of the Rings II	1	5	No
Spider-Man	3.5	4	Yes
Nightcrawler	4.5	5	Yes
Batman vs Superman	1	4	No
...	...	...	..
La La Land	2.5	5	Yes
Star Wars I	4.5	4	Yes
Gravity	3	3	?

Movie recommendation system

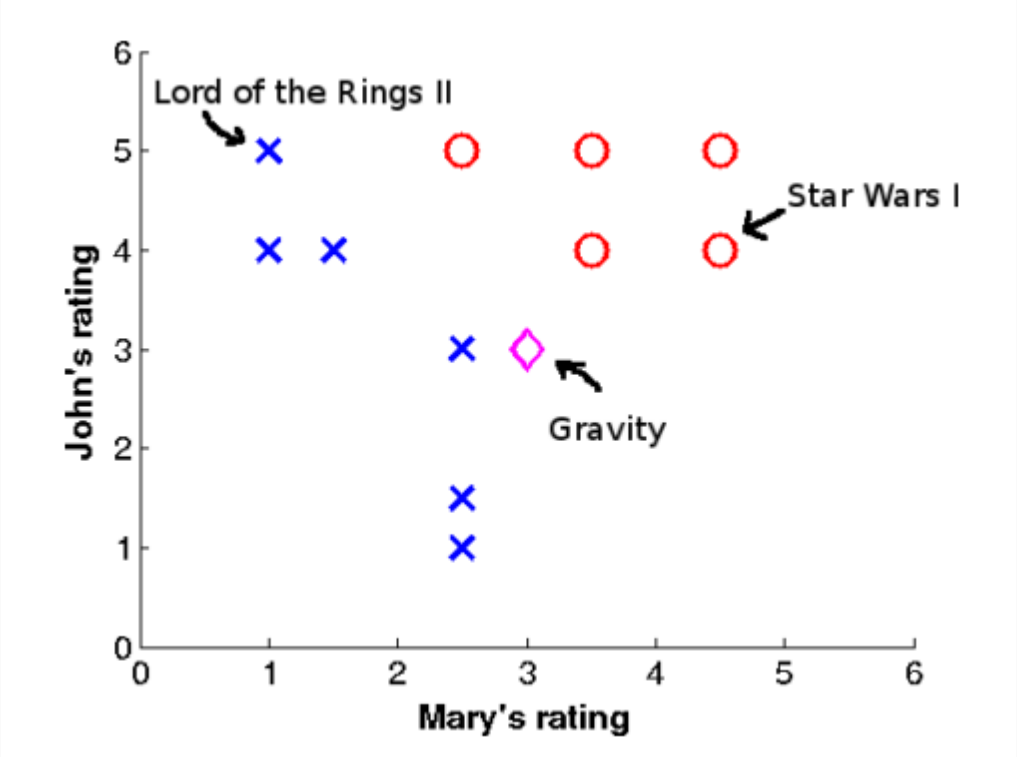
Movie name	x_1 Mary's rating	x_2 John's rating	y I liked it
Lord of the Rings II	1	5	No
Spider-Man	3.5	4	Yes
Nightcrawler	4.5	5	Yes
Batman vs Superman	1	4	No
...	...	...	..
La La Land	2.5	5	Yes
Star Wars I	4.5	4	Yes
Gravity	3	3	?

$$x_1 = [1, 3.5, 4.5, 1, ..., 2.5, 4.5, 3]$$

$$x_2 = [5, 4, 5, 4, ..., 5, 4, 3]$$

$$y = [0, 1, 1, 0, ..., 1, 1, ?]$$

Movie recommendation system



Le 2014

$$x_1 = [1, 3.5, 4.5, 1, ..., 2.5, 4.5, 3]; \quad x_2 = [5, 4, 5, 4, ..., 5, 4, 3]$$

$$y = [0, 1, 1, 0, ..., 1, 1, ?]$$

$$h(x) \approx y$$

$$x_1 = [1, 3.5, 4.5, 1, ..., 2.5, 4.5, 3]; \quad x_2 = [5, 4, 5, 4, ..., 5, 4, 3]$$

$$y = [0, 1, 1, 0, ..., 1, 1, ?]$$

$$h(x) \approx y$$

$$h(x; \theta, b) = \theta_1 x_1 + \theta_2 x_2 + b$$

$$x_1 = [1, 3.5, 4.5, 1, \ldots, 2.5, 4.5, 3]; \quad x_2 = [5, 4, 5, 4, \ldots, 5, 4, 3]$$

$$y = [0, 1, 1, 0, \ldots, 1, 1, ?]$$

$$h(x) \approx y$$

$$h(x; \theta, b) = \theta_1 x_1 + \theta_2 x_2 + b$$

$$h(x; \theta, b) = \theta^T x + b$$

$$x_1 = [1, 3.5, 4.5, 1, ..., 2.5, 4.5, 3]; \quad x_2 = [5, 4, 5, 4, ..., 5, 4, 3]$$

$$y = [0, 1, 1, 0, ..., 1, 1, ?]$$

$$h(x) \approx y$$

$$h(x; \theta, b) = \theta_1 x_1 + \theta_2 x_2 + b$$

$$h(x; \theta, b) = \theta^T x + b$$

$$h(x; \theta, b) = g(\theta^T x + b), \text{where } g(z) = \begin{cases} 0 & \text{if } z <= 0 \\ 1 & \text{if } z > 0 \end{cases}$$

$$x_1 = [1, 3.5, 4.5, 1, ..., 2.5, 4.5, 3]; \quad x_2 = [5, 4, 5, 4, ..., 5, 4, 3]$$

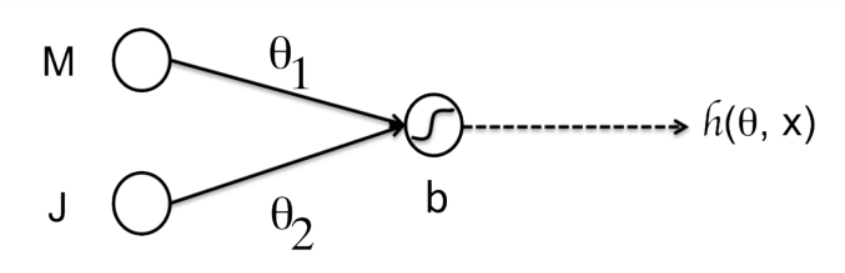
$$y = [0, 1, 1, 0, ..., 1, 1, ?]$$

$$h(x) \approx y$$

$$h(x; \theta, b) = \theta_1 x_1 + \theta_2 x_2 + b$$

$$h(x; \theta, b) = \theta^T x + b$$

$$h(x; \theta, b) = g(\theta^T x + b), \text{where } g(z) = \begin{cases} 0 & \text{if } z <= 0 \\ 1 & \text{if } z > 0 \end{cases}$$



Perceptron training

- Initialize θ and b randomly at $t = 0$
- For each timestep t calculate h(x)

$$h(x) = \theta_1(t)x_1 + \theta_2(t)x_2 + b(t)$$

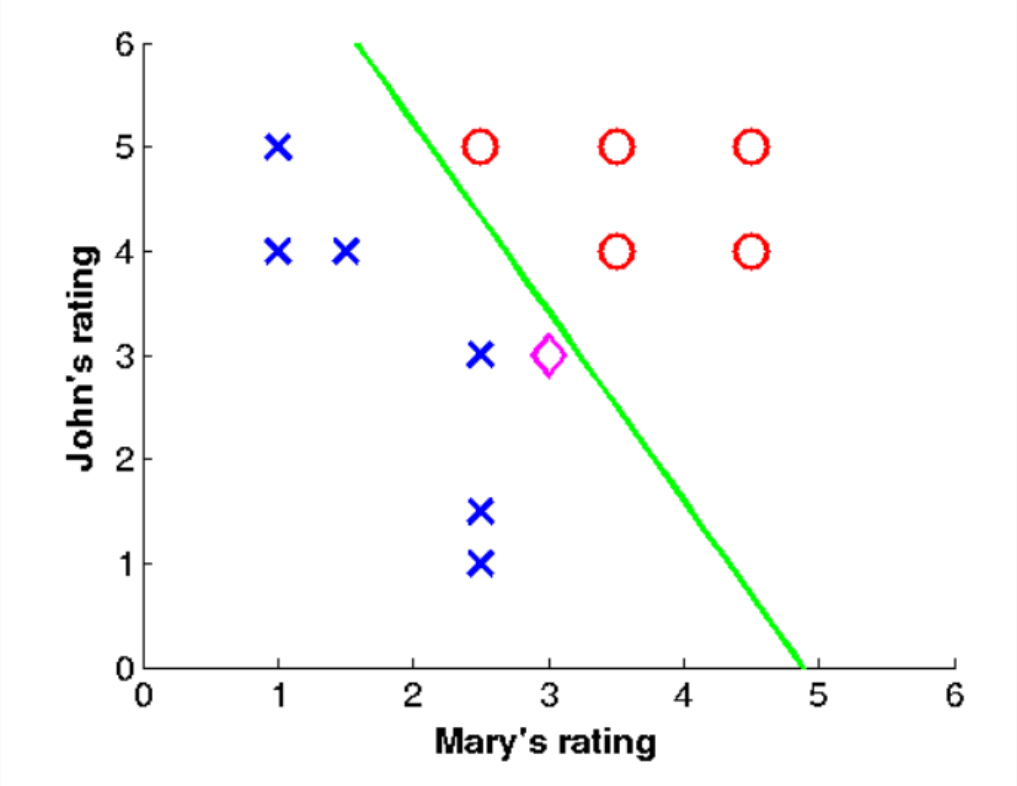
- Update each weight and the bias

$$\theta_i(t+1) = \theta_i(t) + (y - h(x))x_i$$

$$b(t+1) = b(t) + (y - h(x))$$

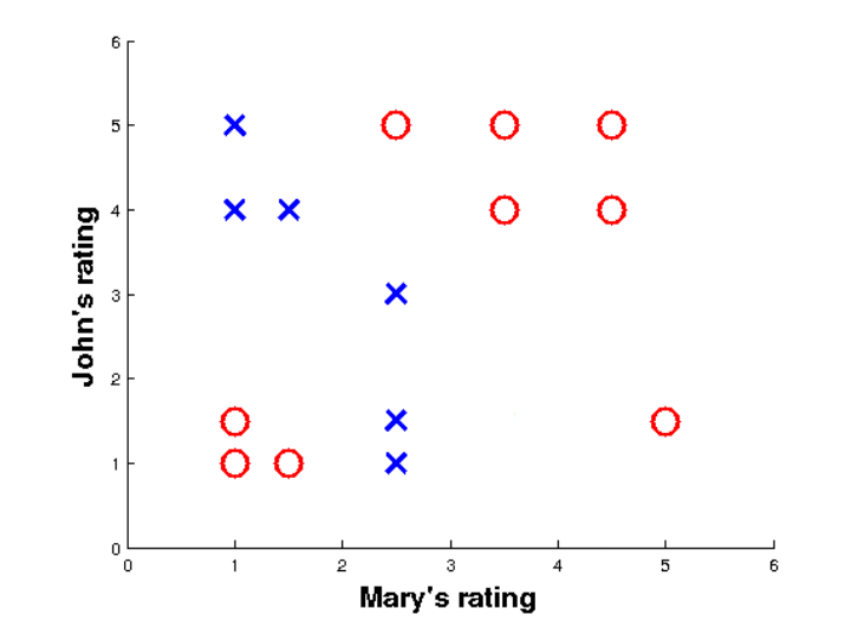
- Repeat until error $\sum_x |y - h(x)|$ is below threshold

Trained perceptron

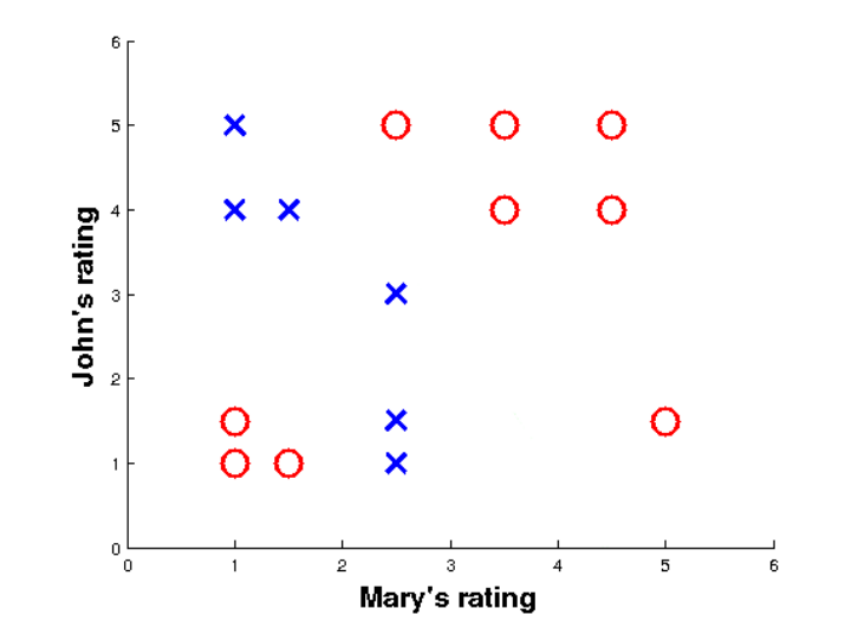


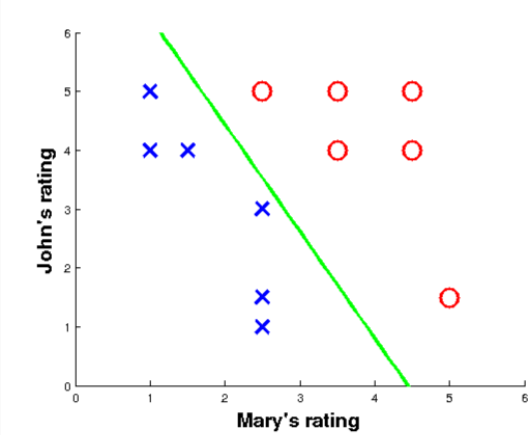
Le 2014

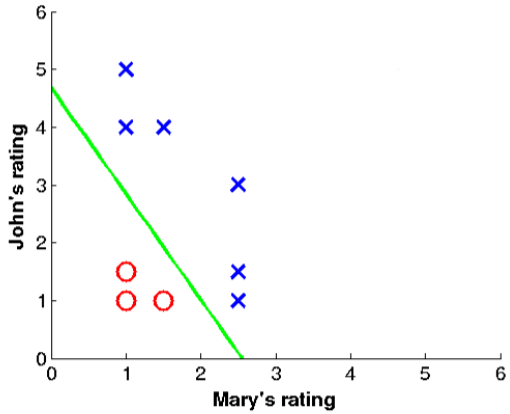
Limits of single perceptron



Limits of single perceptron







Multi-layer perceptron

$$h(x) \approx y$$

Multi-layer perceptron

$$h(x) \approx y$$

$$h_1(x; \theta, b) = \theta_1 x_1 + \theta_2 x_2 + b_1$$
$$h_2(x; \theta, b) = \theta_3 x_1 + \theta_4 x_2 + b_2$$

Multi-layer perceptron

$$h(x) \approx y$$

$$h_1(x; \theta, b) = \theta_1 x_1 + \theta_2 x_2 + b_1$$
$$h_2(x; \theta, b) = \theta_3 x_1 + \theta_4 x_2 + b_2$$

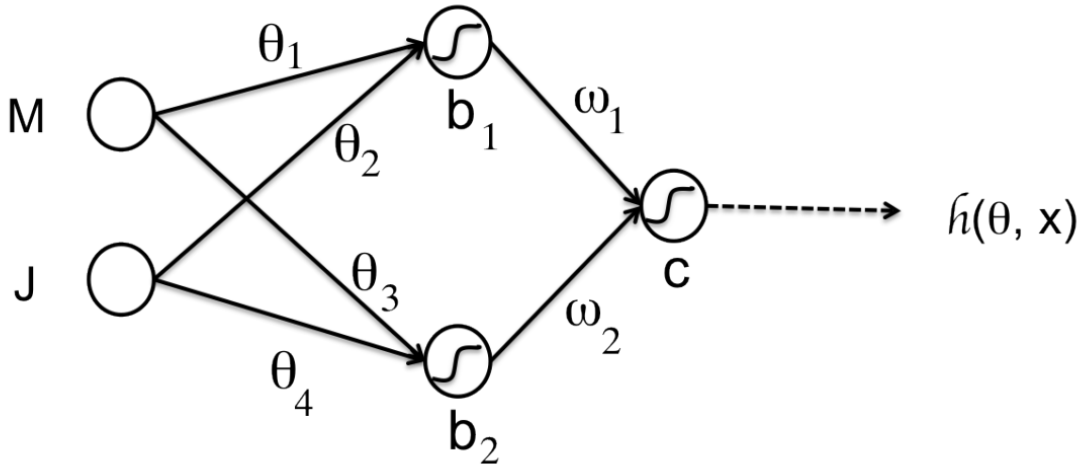
$$h((h_1(x), h_2(x)); \omega, c) = \omega_1 h_1(x) + \omega_2 h_2(x) + c$$

Multi-layer perceptron

$$h(x) \approx y$$

$$h_1(x; \theta, b) = \theta_1 x_1 + \theta_2 x_2 + b_1$$
$$h_2(x; \theta, b) = \theta_3 x_1 + \theta_4 x_2 + b_2$$

$$h((h_1(x), h_2(x)); \omega, c) = \omega_1 h_1(x) + \omega_2 h_2(x) + c$$



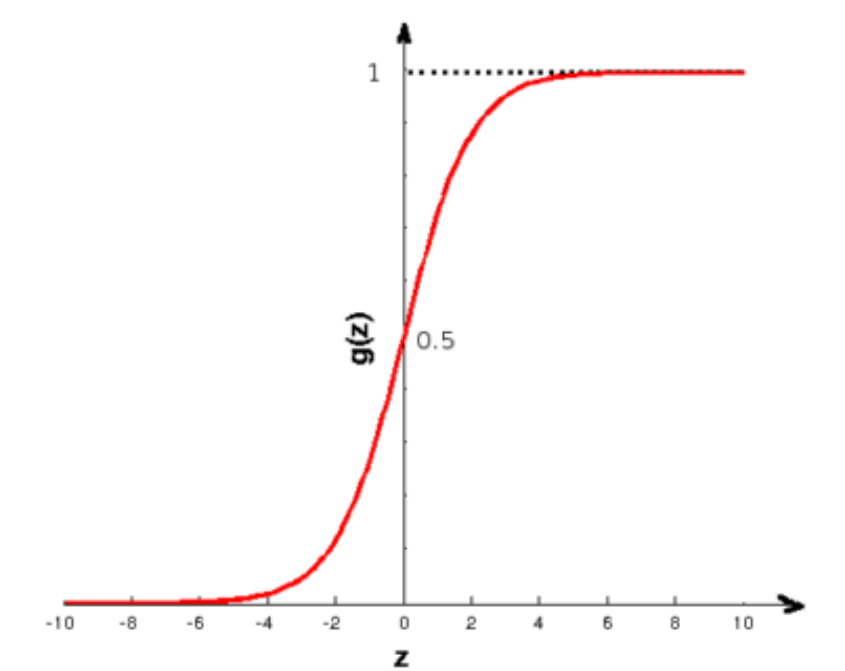
Activation function

$$h(x; \theta, b) = g(\theta^T x + b), \text{where } g(z) = \begin{cases} 0 & \text{if } z <= 0 \\ 1 & \text{if } z > 0 \end{cases}$$

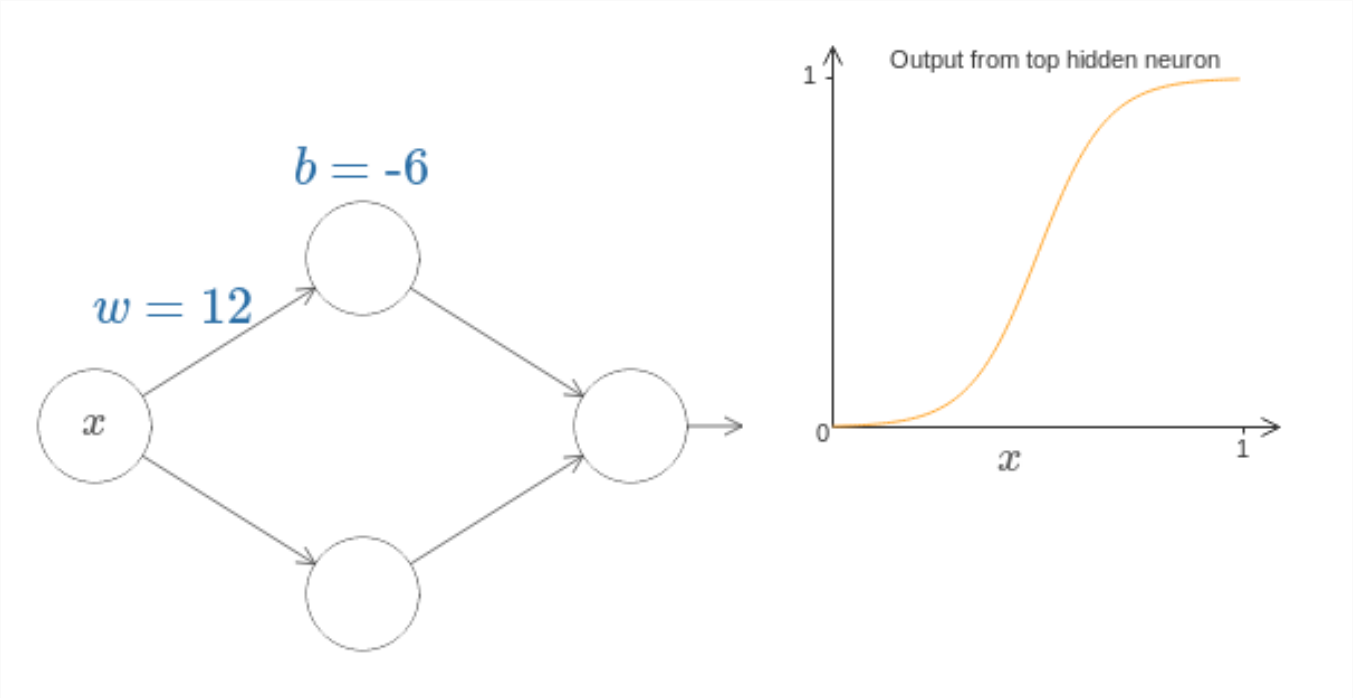
Activation function

$$h(x;\theta,b) = g(\theta^T x + b), \text{where } g(z) = \begin{cases} 0 & \text{if } z <= 0 \\ 1 & \text{if } z > 0 \end{cases}$$

$$h(x;\theta,b) = g(\theta^T x + b), \text{where } g(z) = \frac{1}{1 + exp(z)}$$

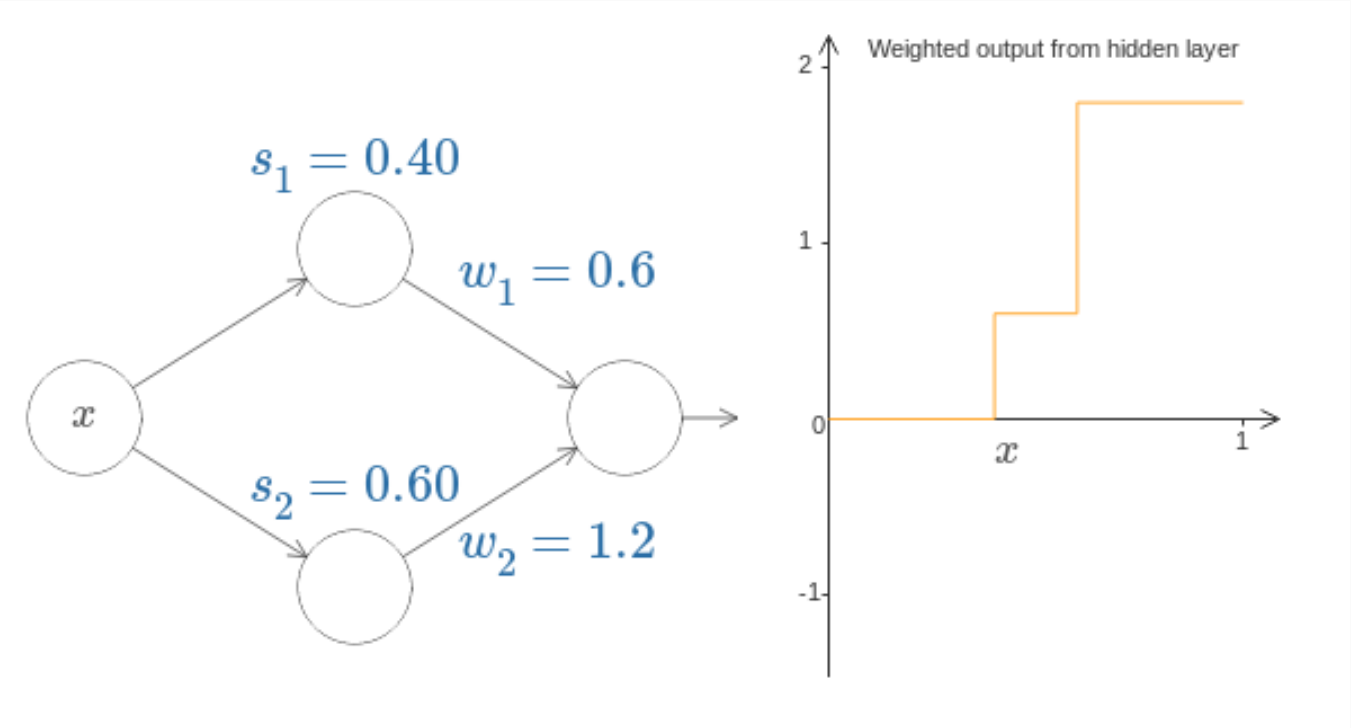


Sigmoid neurons



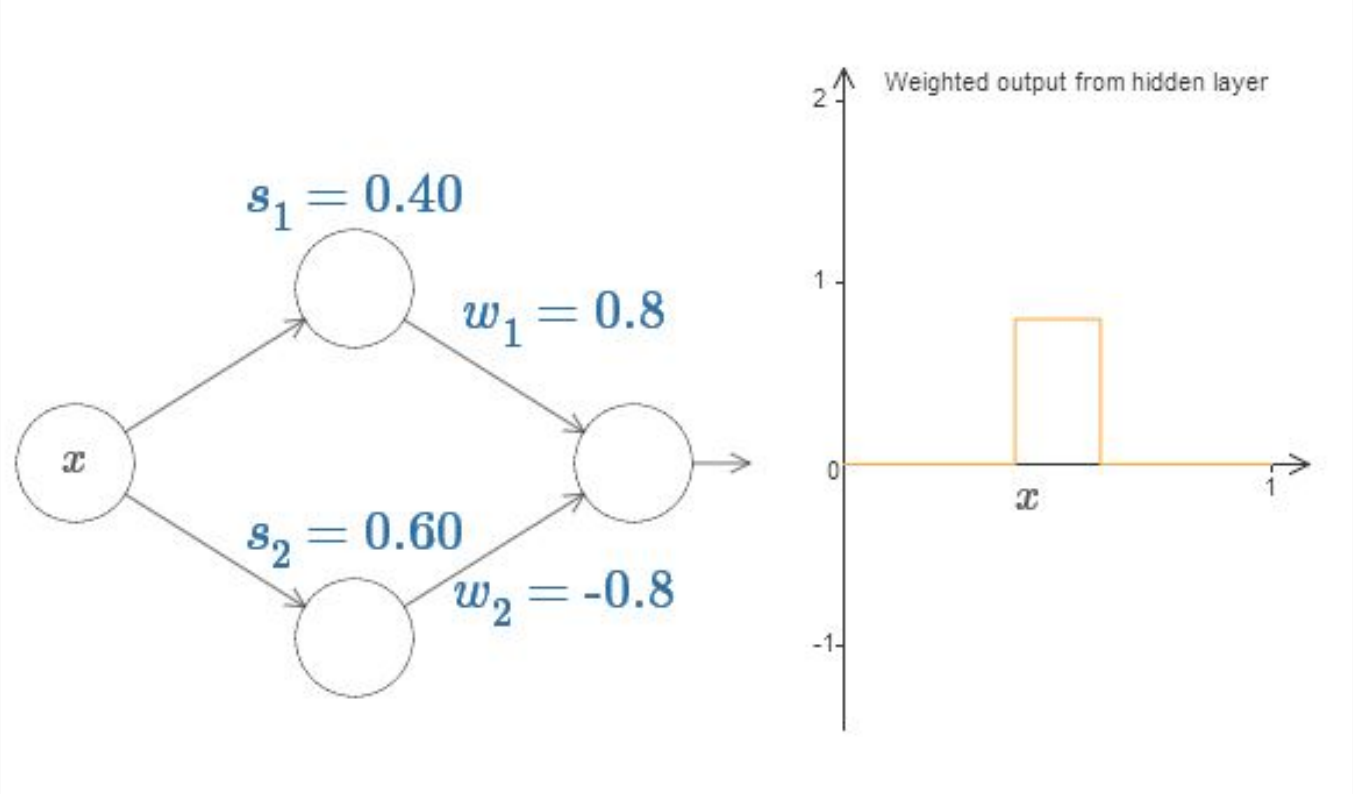
Nielsen 2015

Sigmoid neurons



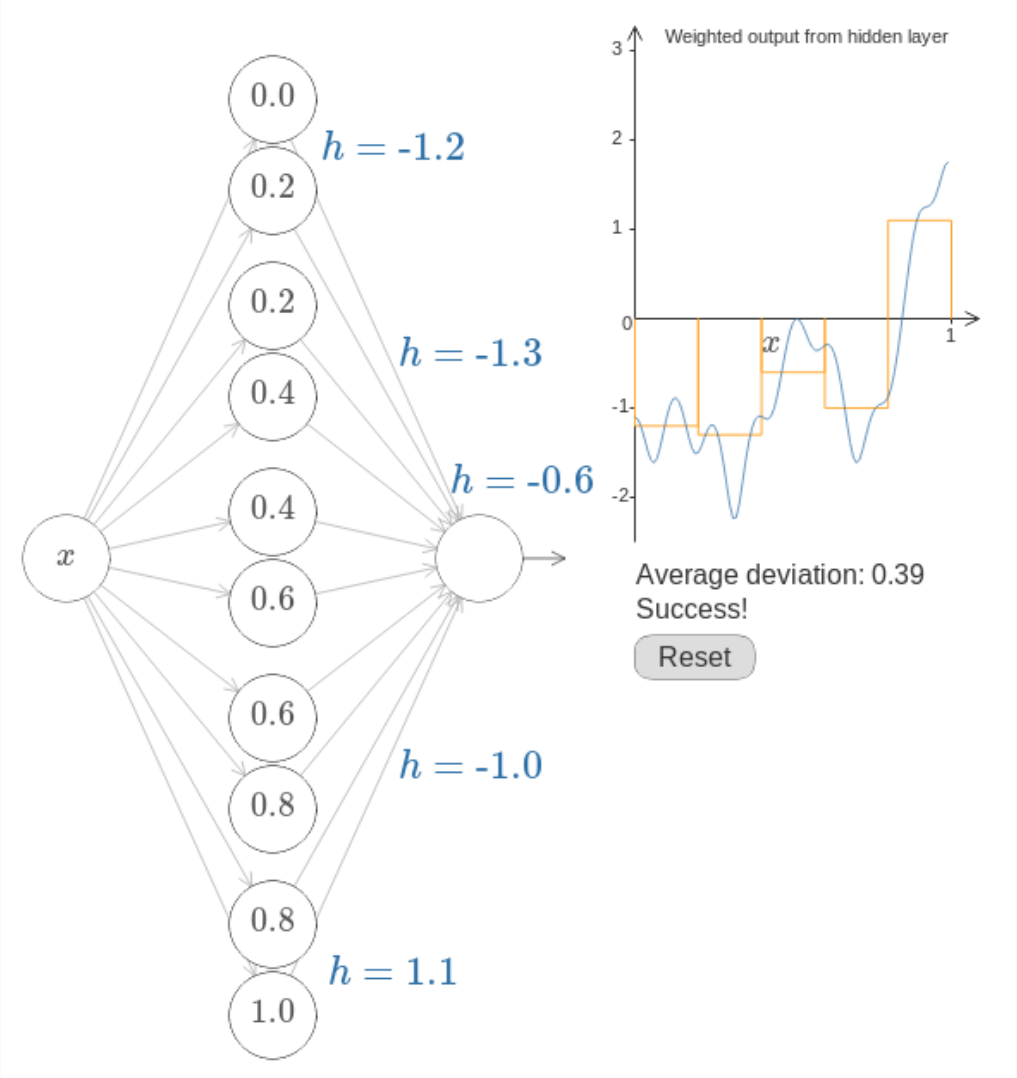
Nielsen 2015

Sigmoid neurons



Nielsen 2015

Sigmoid neurons



http://neuralnetworksanddeeplearning.com Nielsen 2015

Perceptron training

- Initialize θ and b randomly at $t = 0$
- For each timestep t calculate h(x)

$$h(x) = \theta_1(t)x_1 + \theta_2(t)x_2 + b(t)$$

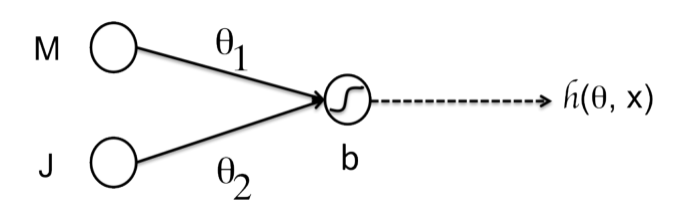
- Update each weight and the bias

$$\theta_i(t+1) = \theta_i(t) + (y - h(x))x_i$$

$$b(t+1) = b(t) + (y - h(x))$$

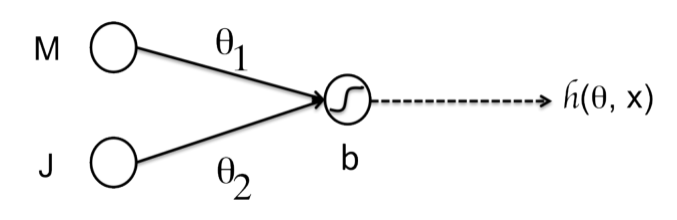
- Repeat until error $\sum_x |y - h(x)|$ is below threshold

Sigmoid neuron weight update



$$\theta_1(t+1) = \theta_1(t) - \alpha\Delta\theta_1(t)$$
$$\theta_2(t+1) = \theta_2(t) - \alpha\Delta\theta_2(t)$$
$$b(t+1) = b(t) - \alpha\Delta b$$

Sigmoid neuron weight update



$$\theta_1(t+1) = \theta_1(t) - \alpha\Delta\theta_1(t)$$

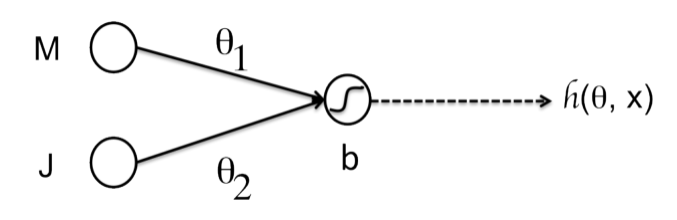
$$\theta_2(t+1) = \theta_2(t) - \alpha\Delta\theta_2(t)$$

$$b(t+1) = b(t) - \alpha\Delta b$$

$$\mathcal{L} = (h(x;\theta,b) - y)^2$$

$$\Delta\theta_1 = \frac{\delta}{\delta\theta_1}\mathcal{L}$$

Sigmoid neuron weight update



$$\theta_1(t+1) = \theta_1(t) - \alpha\Delta\theta_1(t)$$
$$\theta_2(t+1) = \theta_2(t) - \alpha\Delta\theta_2(t)$$
$$b(t+1) = b(t) - \alpha\Delta b$$

$$\mathcal{L} = (h(x;\theta,b) - y)^2$$
$$\Delta\theta_1 = \frac{\delta}{\delta\theta_1}\mathcal{L}$$

$$\Delta\theta_1 = 2[g(\theta^T x + b) - y][1 - g(\theta^T x + b)]g(\theta^T x + b)x_1$$

Stochastic Gradient Descent

- Initialize w and b randomly at $t = 0$
- For each timestep t compute the partial derivatives

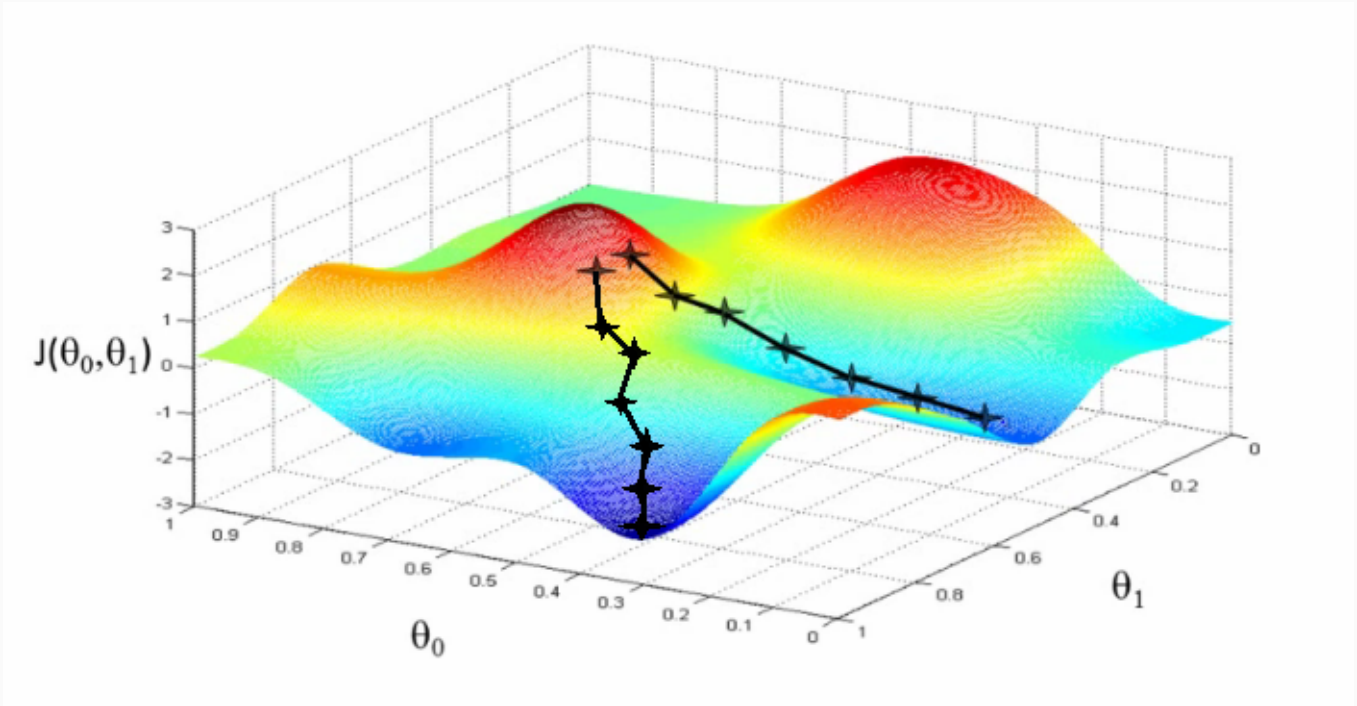
$$\Delta\theta_i = 2[g(\theta^T x + b) - y][1 - g(\theta^T x + b)]g(\theta^T x + b)x_i$$
$$\Delta b = 2[g(\theta^T x + b) - y][1 - g(\theta^T x + b)]g(\theta^T x + b)$$

- Update each weight and the bias

$$\theta_i(t+1) = \theta_i(t) - \alpha\Delta\theta_i(t)$$
$$b(t+1) = b(t) - \alpha\Delta b$$

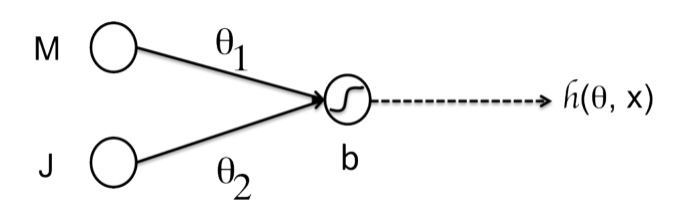
- Repeat until error $\sum_x |y - h(x)|$ is below threshold

Stochastic Gradient Descent



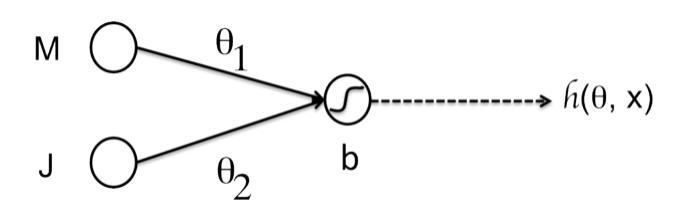
http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/

Stochastic Gradient Descent

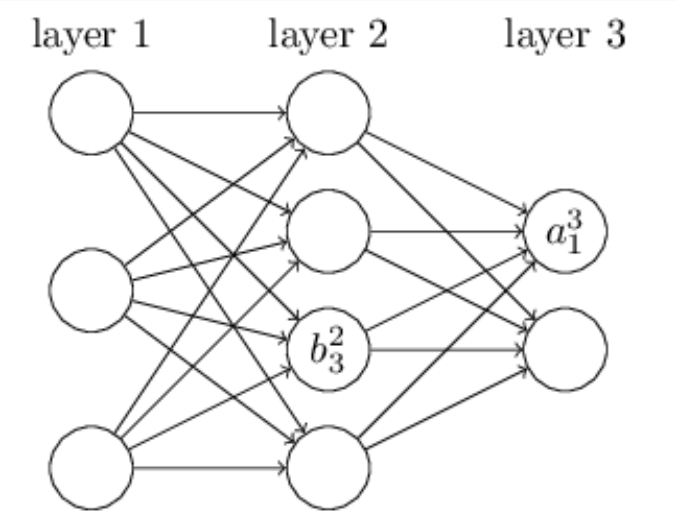


Le 2014

Stochastic Gradient Descent



Le 2014



Nielsen 2015

How to compute $\Delta\theta^{(1)}, \Delta\theta^{(2)}, \ldots$?

Backpropagation

- Pass inputs through network in a "feedforward pass", computing $h^{(1)}, h^{(2)}, \ldots$
- At the output layer, L, compute

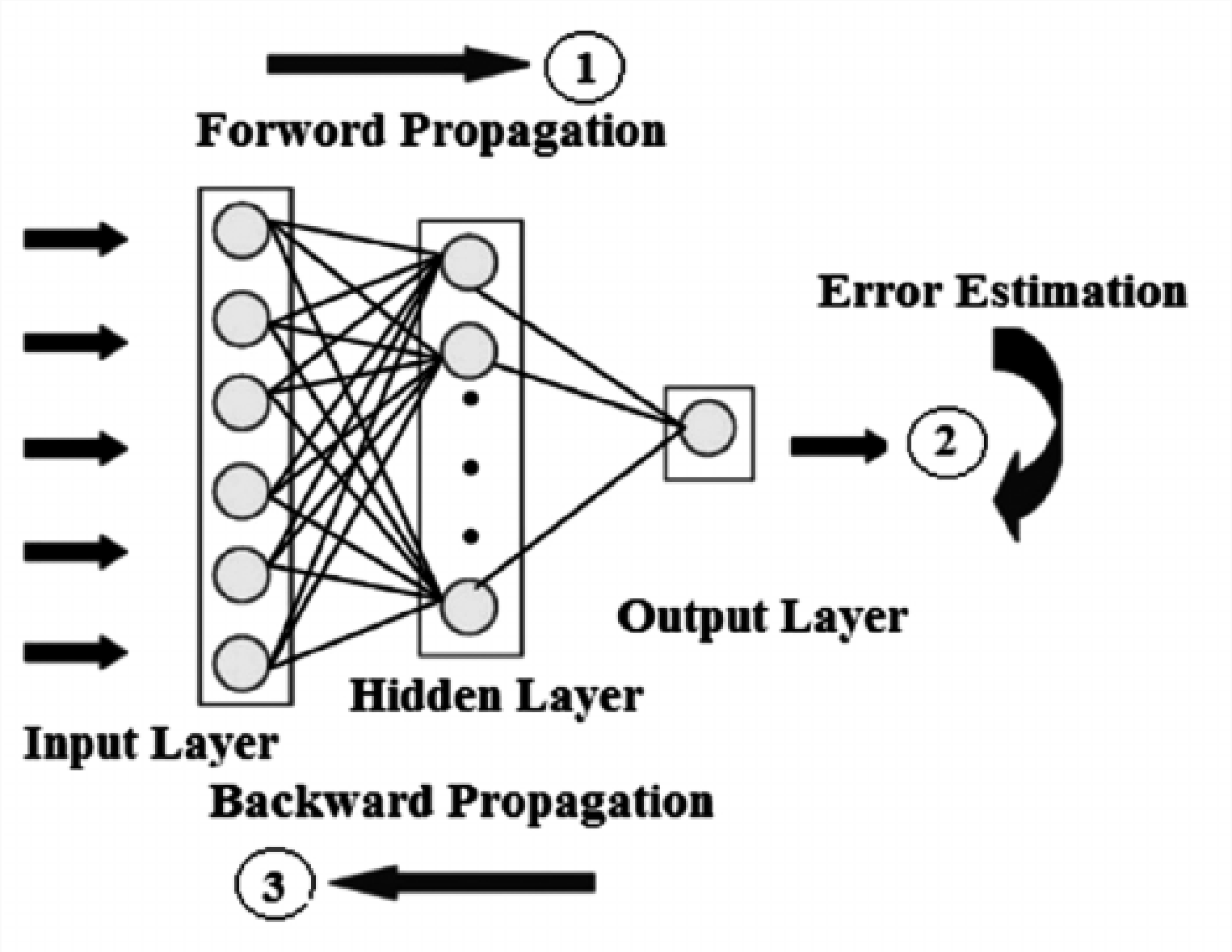
$$\delta^{(L)} = 2(h^{(L)} - y) \odot (g'((\theta^{(L-1)})^T h^{(L-1)} + b^{(L-1)}))$$

- Work backwards, one layer at a time, for $l = L-1, L-2, \ldots$

$$\delta^{(l)} = ((\theta^{(l)})^T \delta^{(l+1)}) \odot (g'((\theta^{(l-1)})^T h^{(l-1)} + b^{(l-1)}))$$

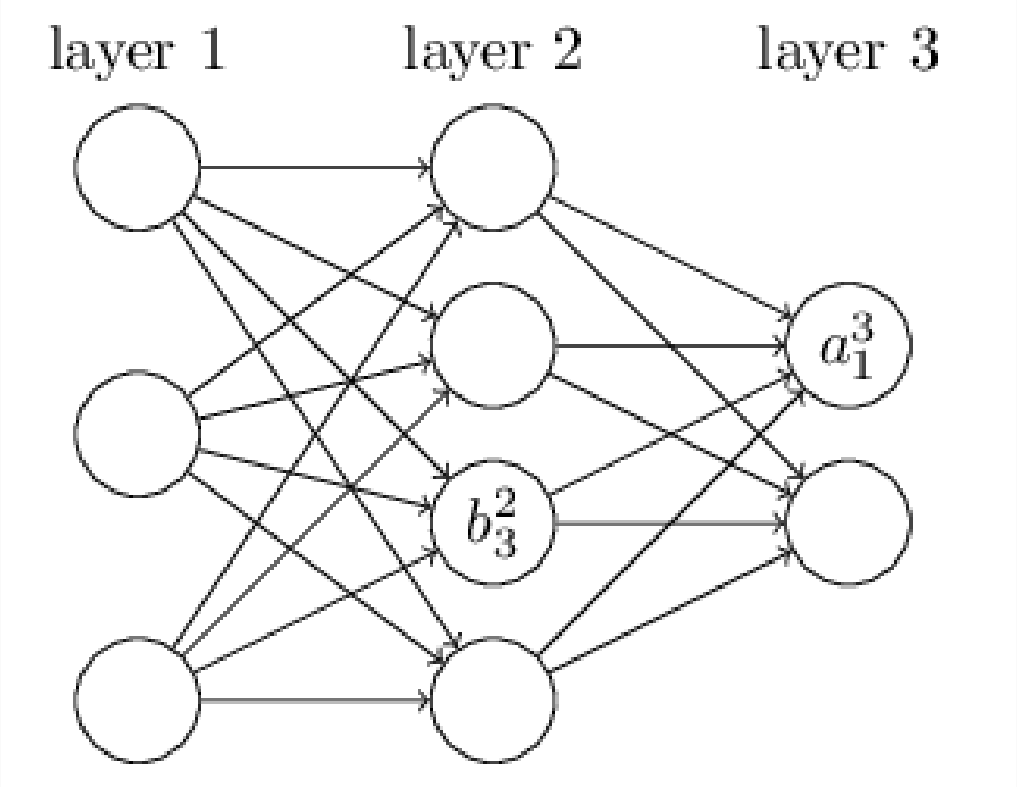
- Calculate final weight updates for each layer

$$\Delta\theta^{(l)} = \delta^{(l+1)}(h^{(l)})^T$$
$$\Delta b^{(l)} = \delta^{(l+1)}$$



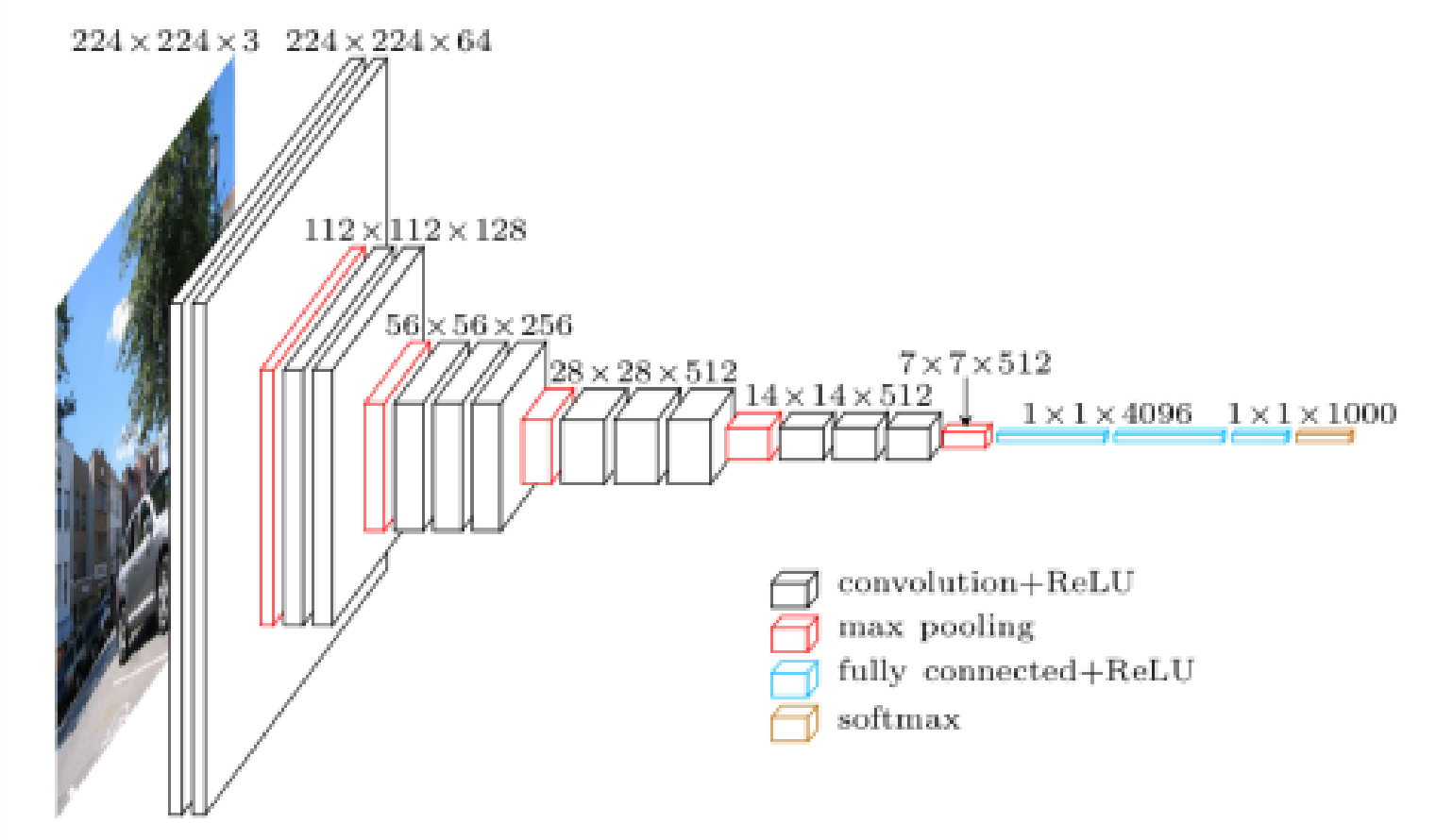
Abedini et al. 2012

Feed-forward network



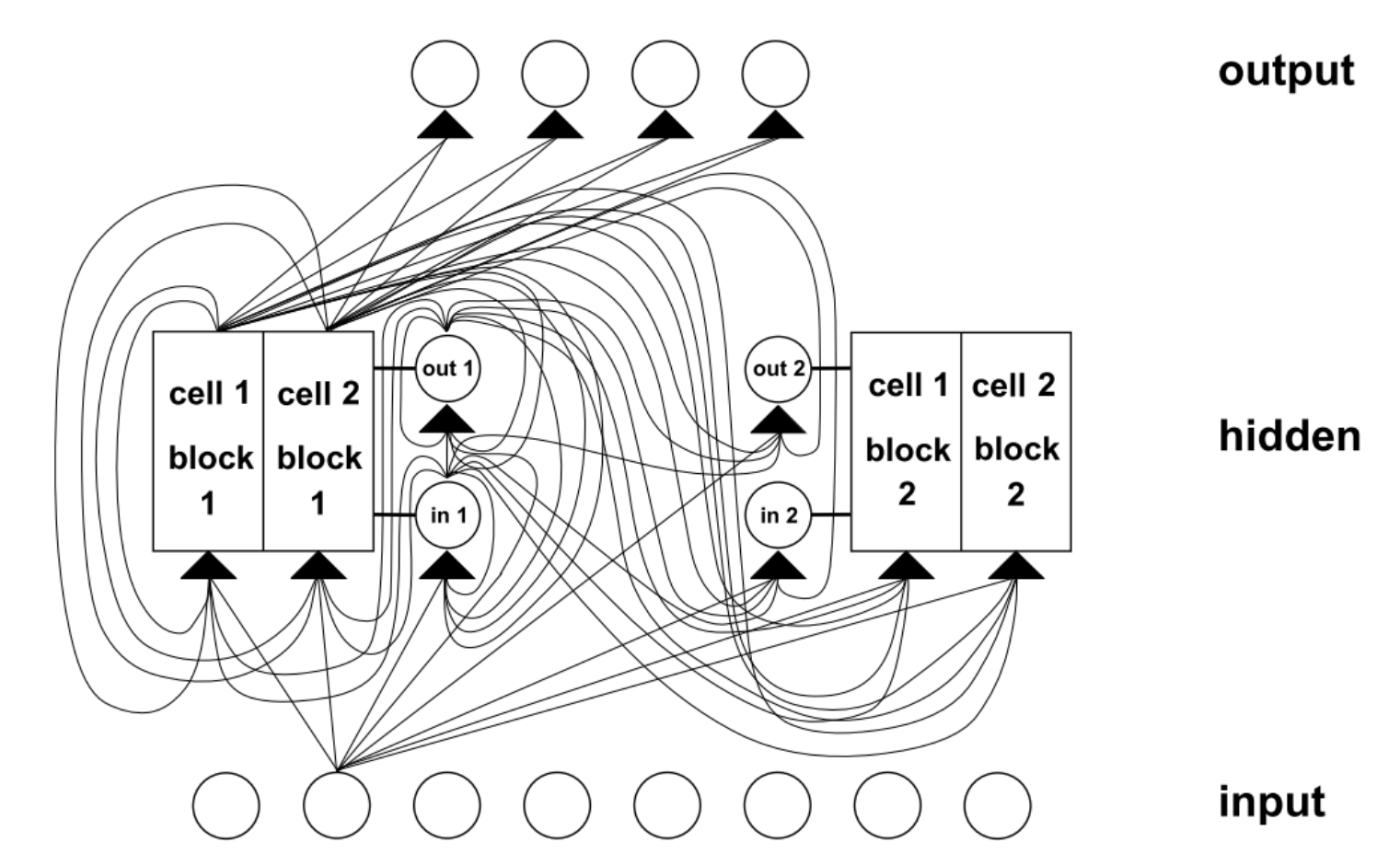
Nielsen 2015

Convolutional networks: VGG16



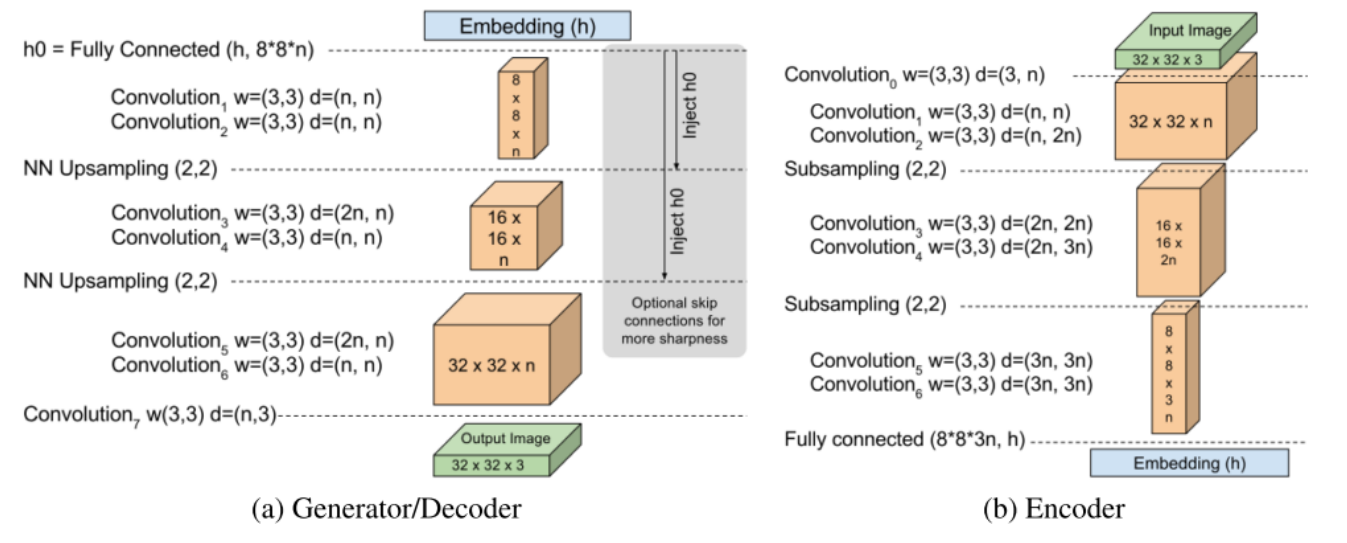
Russakovsky et al. 2015

Long Short-Term Memory network



Hochreiter and Schmidhuber 1997

BEGAN: Boundary Equilibrium Generative Adversarial Networks



Berthelot, Schumm, and Metz 2017

Deep learning

Activation functions

Sigmoid

$$h(x; \theta, b) = g(\theta^T x + b), \text{where } g(z) = \frac{1}{1 + exp(z)}$$

Activation functions

Sigmoid

$$h(x; \theta, b) = g(\theta^T x + b), \text{where } g(z) = \frac{1}{1 + exp(z)}$$

ReLU (Rectified Linear Unit)

$$h(x; \theta, b) = g(\theta^T x + b), \text{where } g(z) = max(z, 0)$$

Activation functions

Sigmoid

$$h(x;\theta,b) = g(\theta^T x + b), \text{where } g(z) = \frac{1}{1+exp(z)}$$

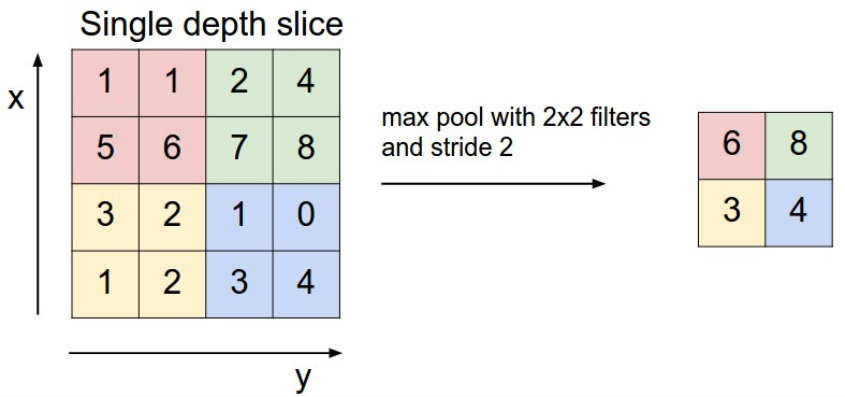
ReLU (Rectified Linear Unit)

$$h(x;\theta,b) = g(\theta^T x + b), \text{where } g(z) = max(z,0)$$

tanh

$$h(x;\theta,b) = g(\theta^T x + b), \text{where } g(z) = tanh(z)$$

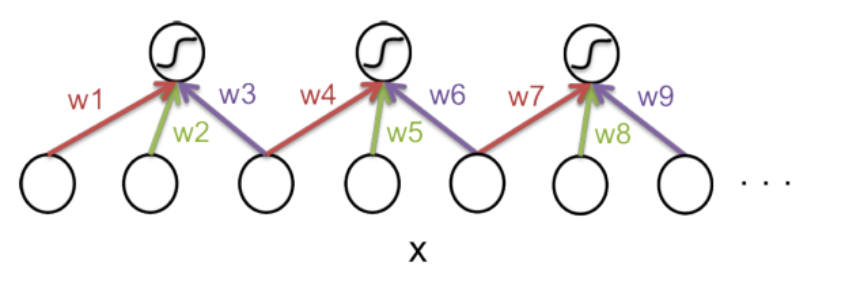
Pooling layers



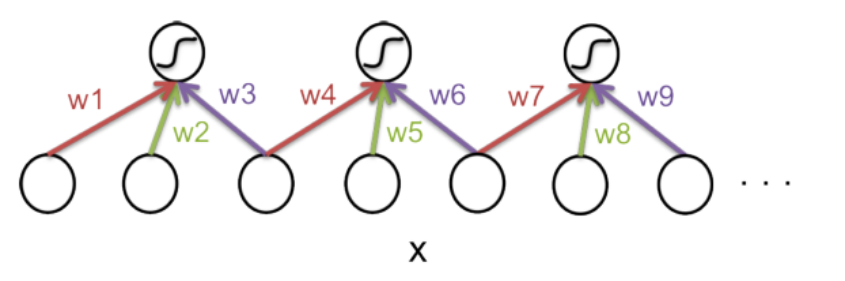
Maximum, mean of previous layer. Not trained

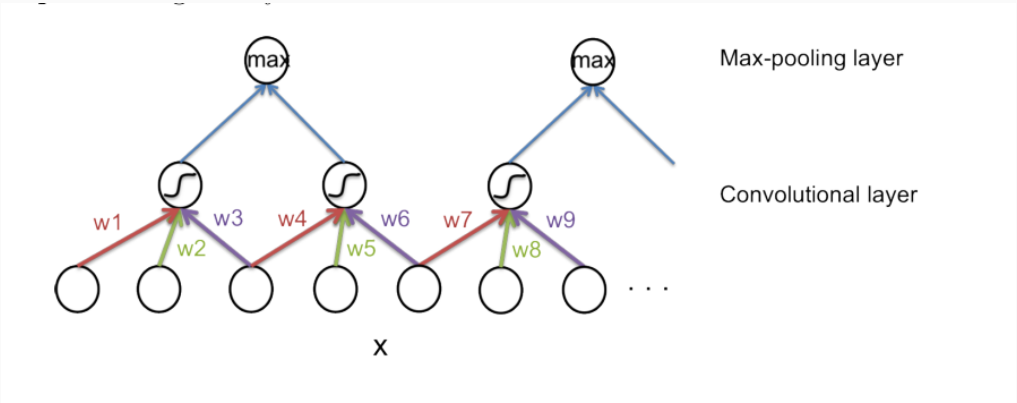
http://cs231n.github.io

Convolutional layer

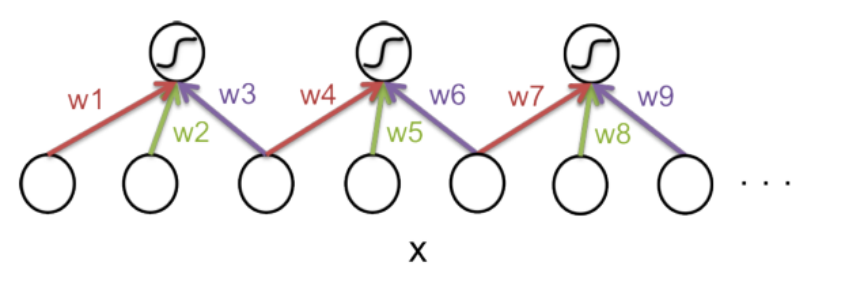


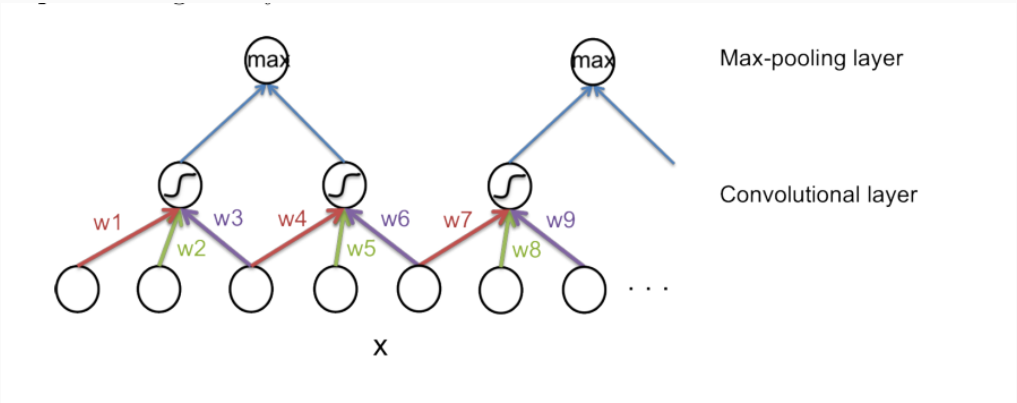
Convolutional layer

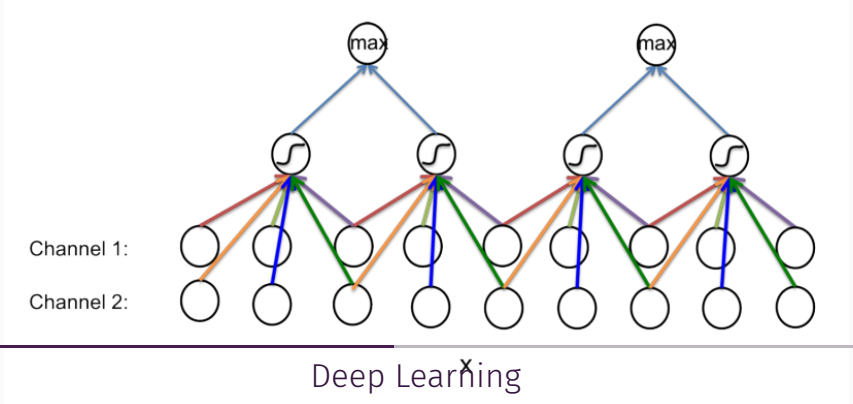




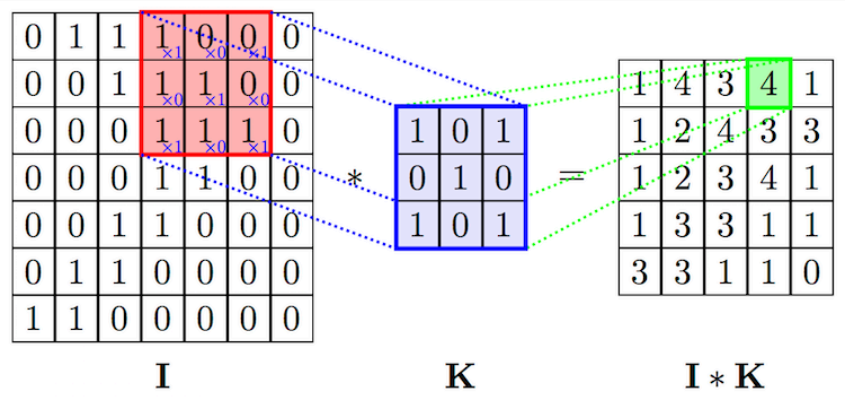
Convolutional layer





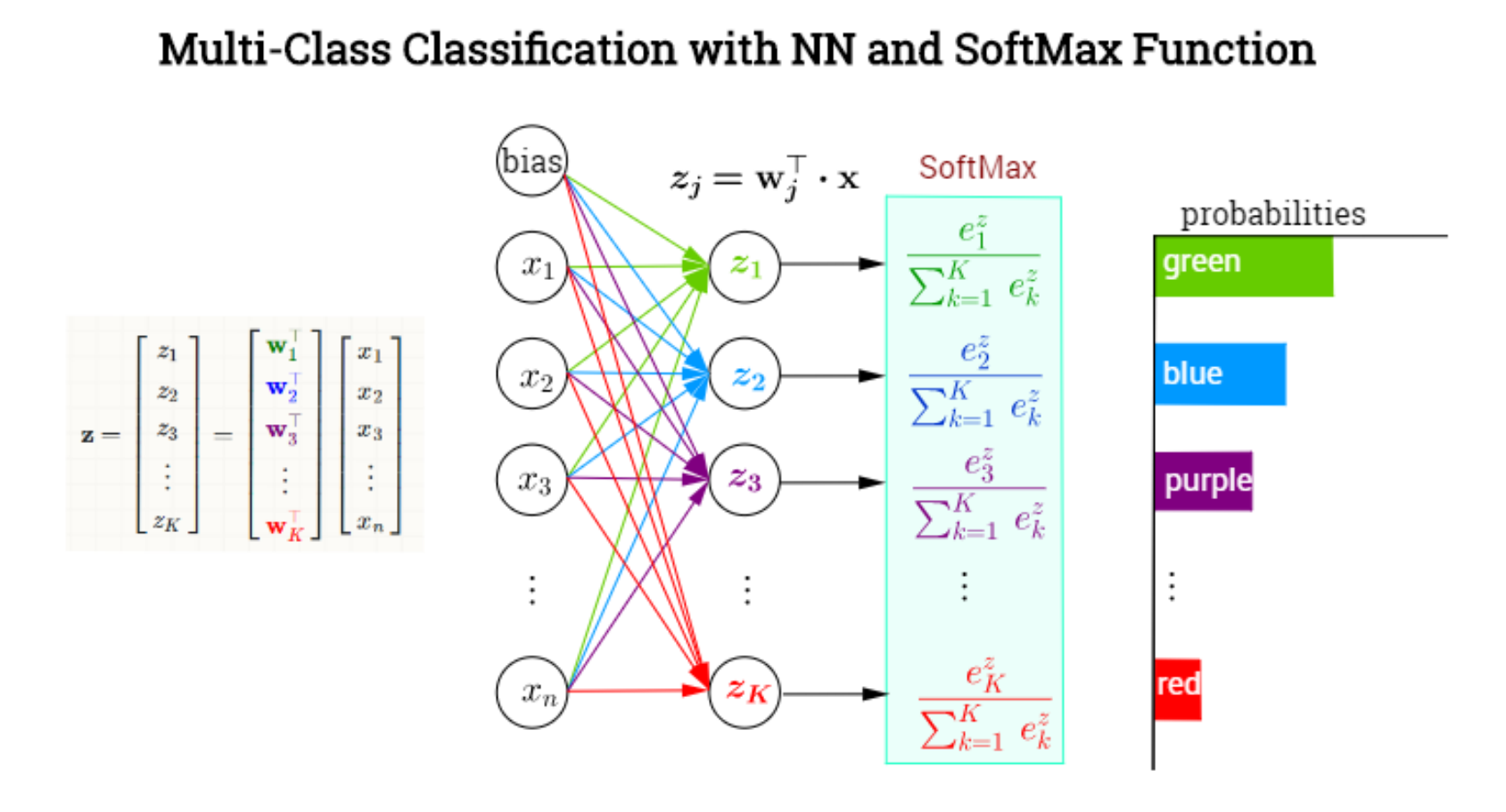


Convolutional layer

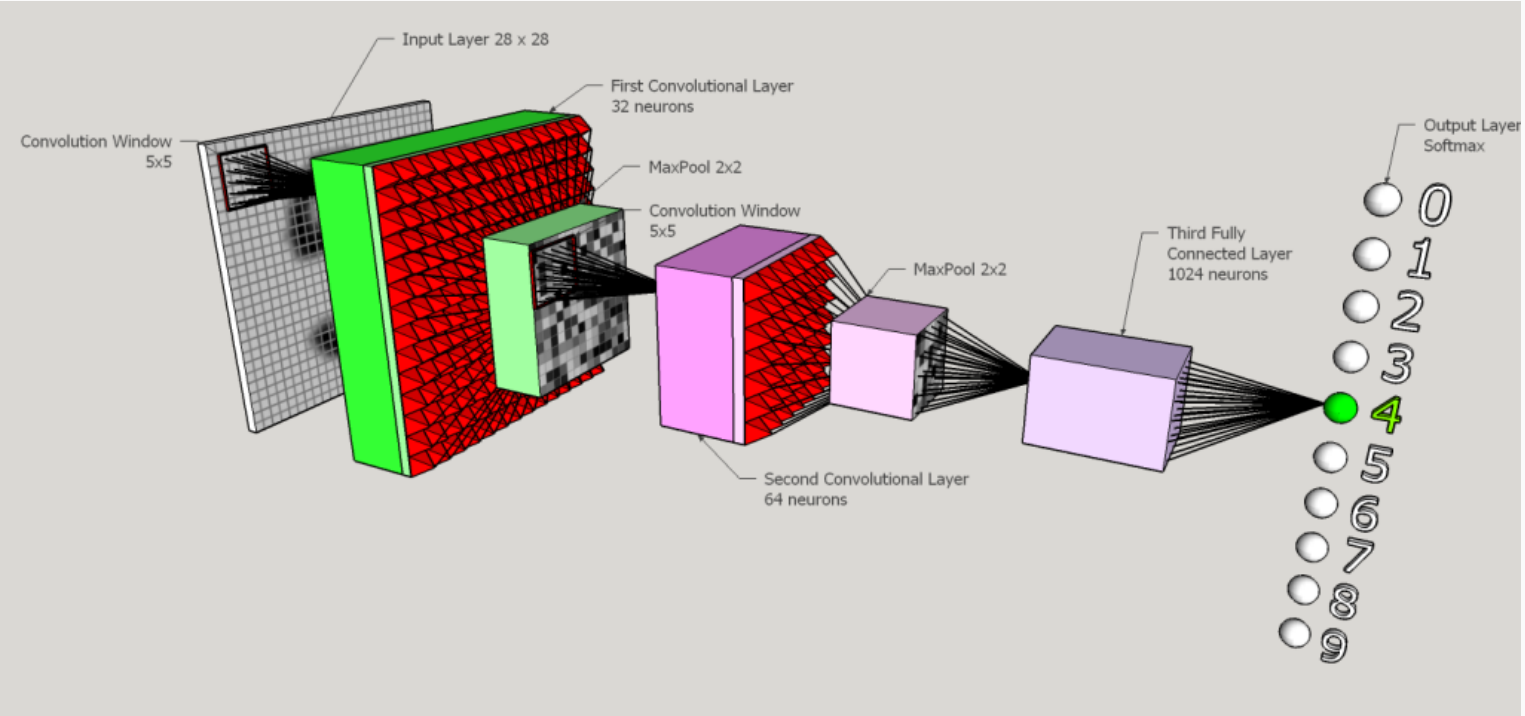


https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

Softmax layer

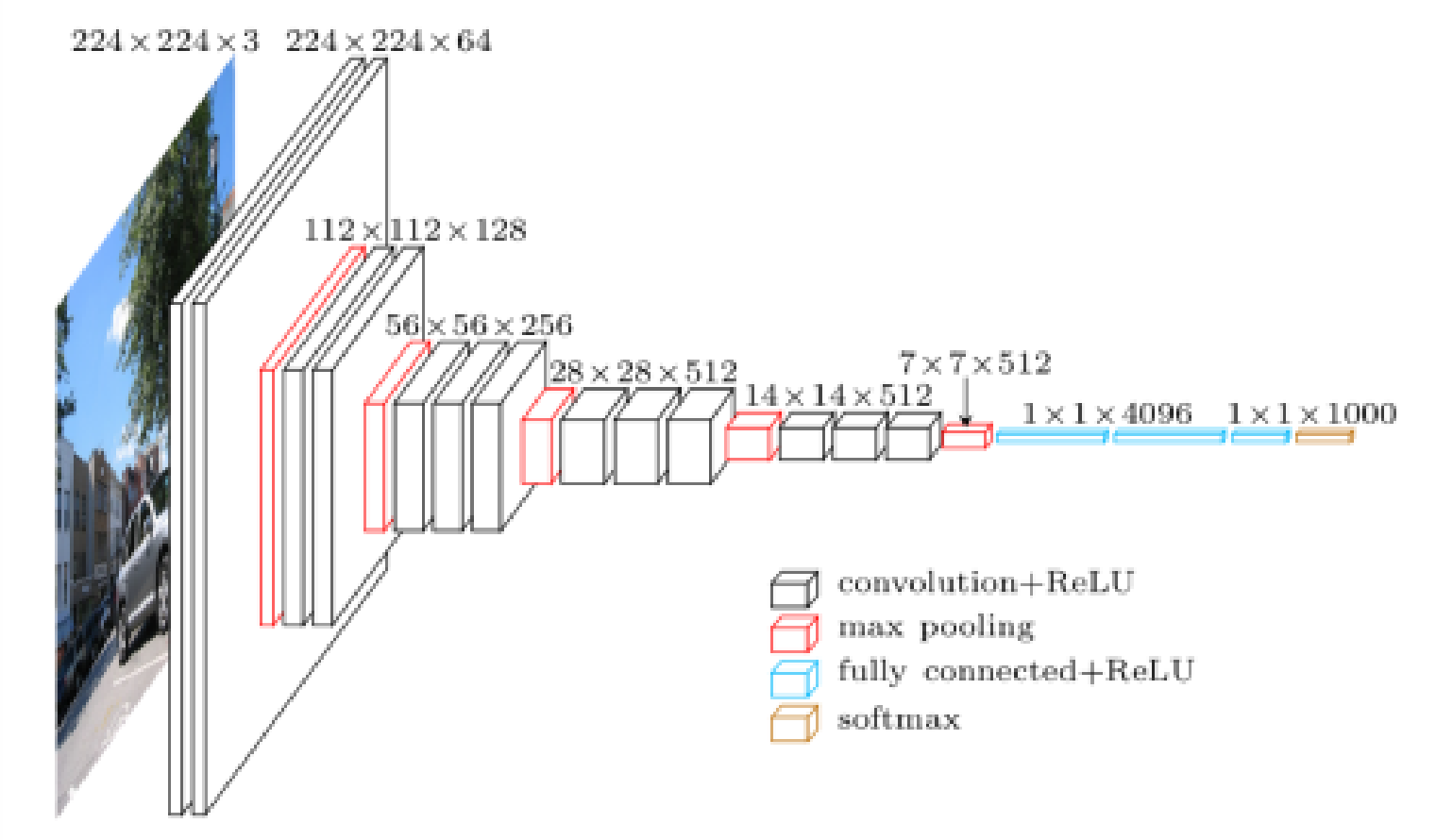


Convolutional network



http://zderadicka.eu/revival-of-neural-networks/

Convolutional networks: VGG16



Russakovsky et al. 2015

LeNet

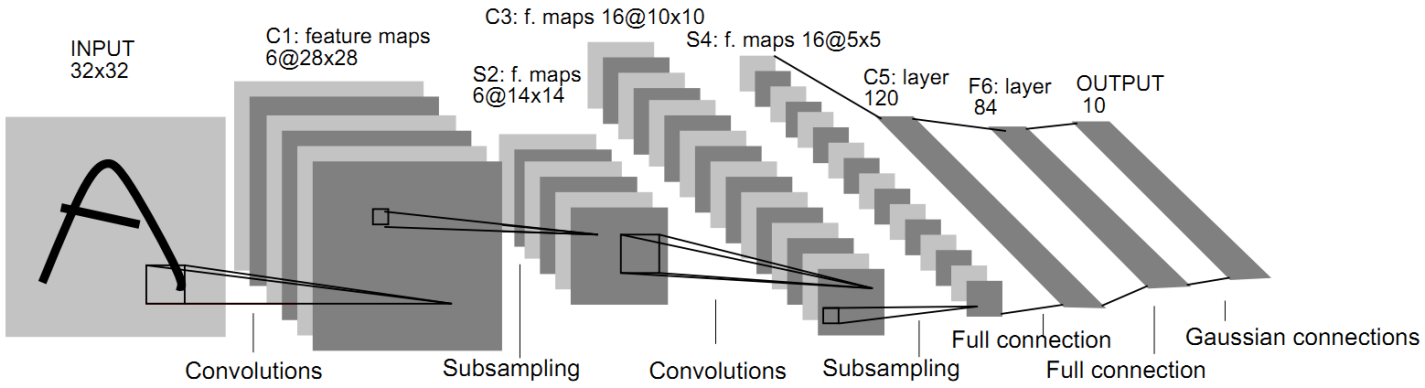
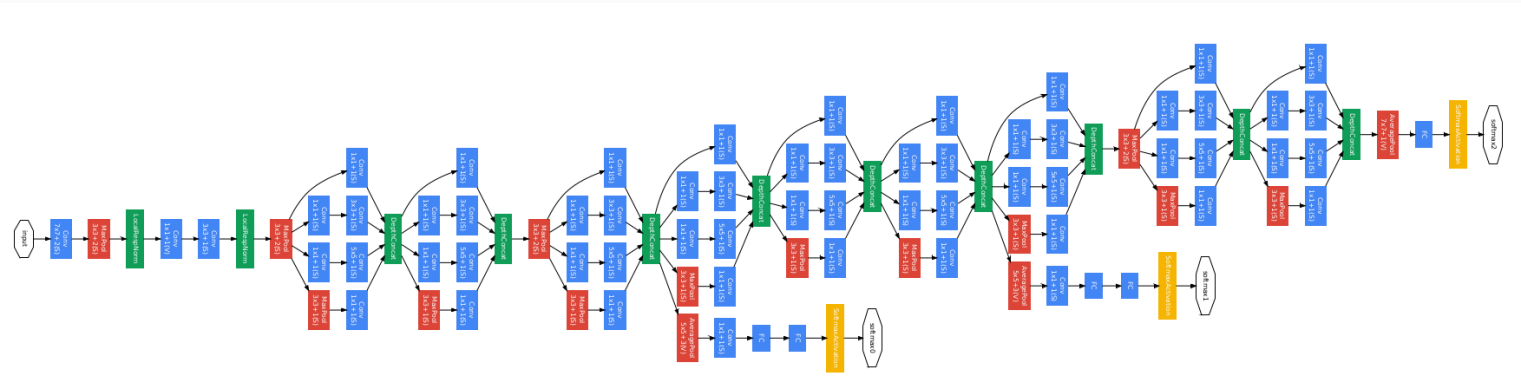


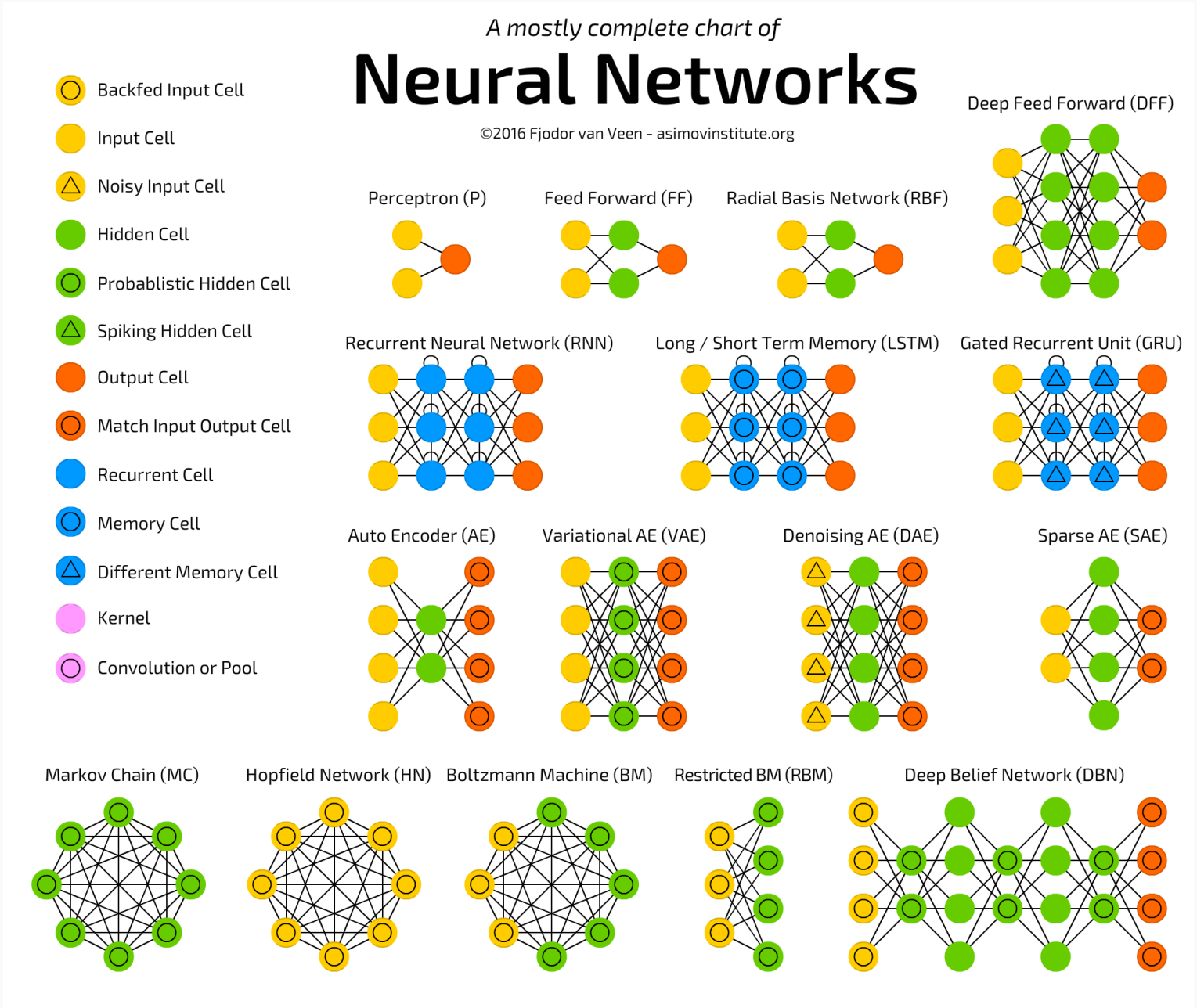
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

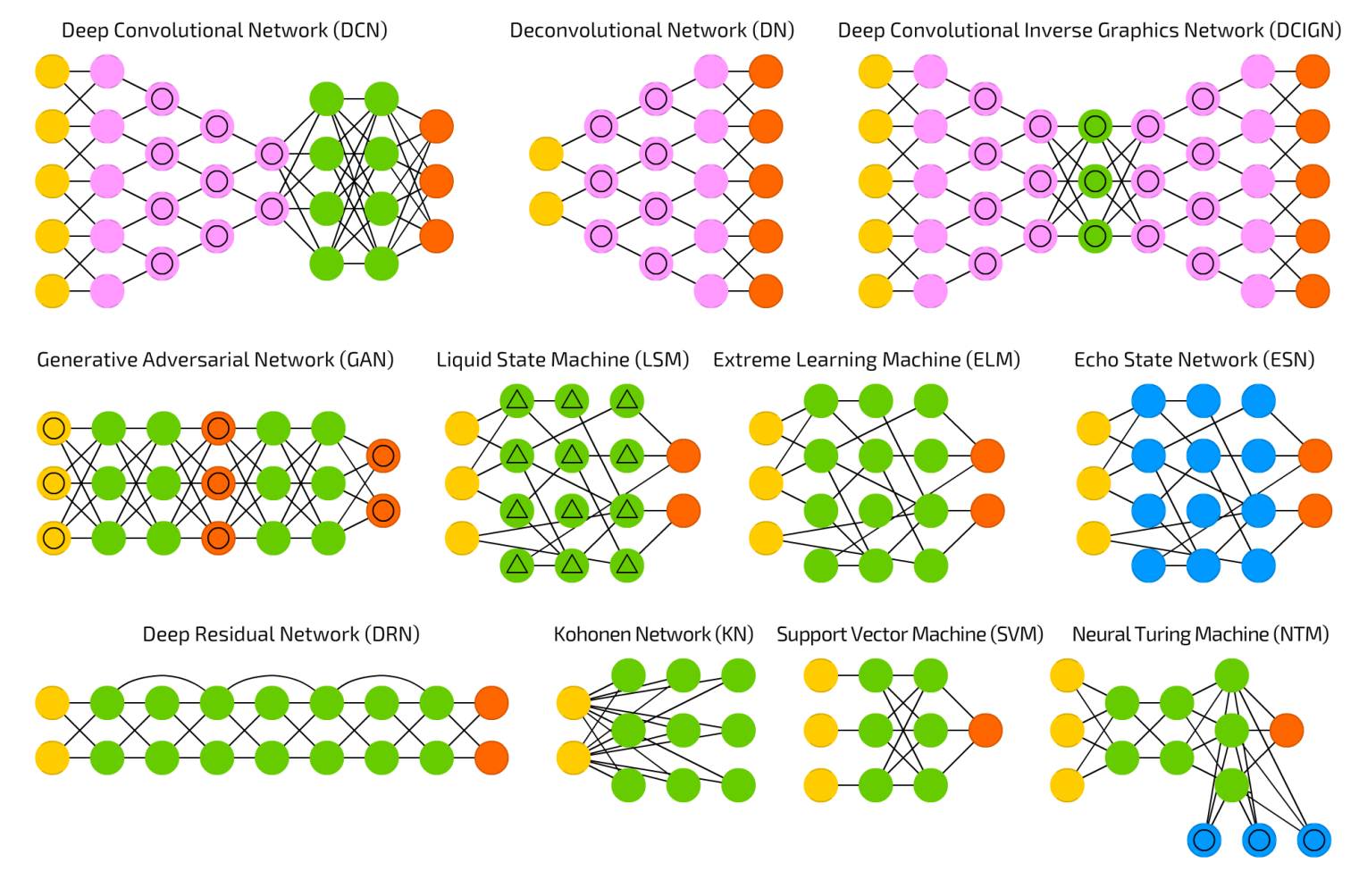
LeCun et al. 1998

GoogLeNet



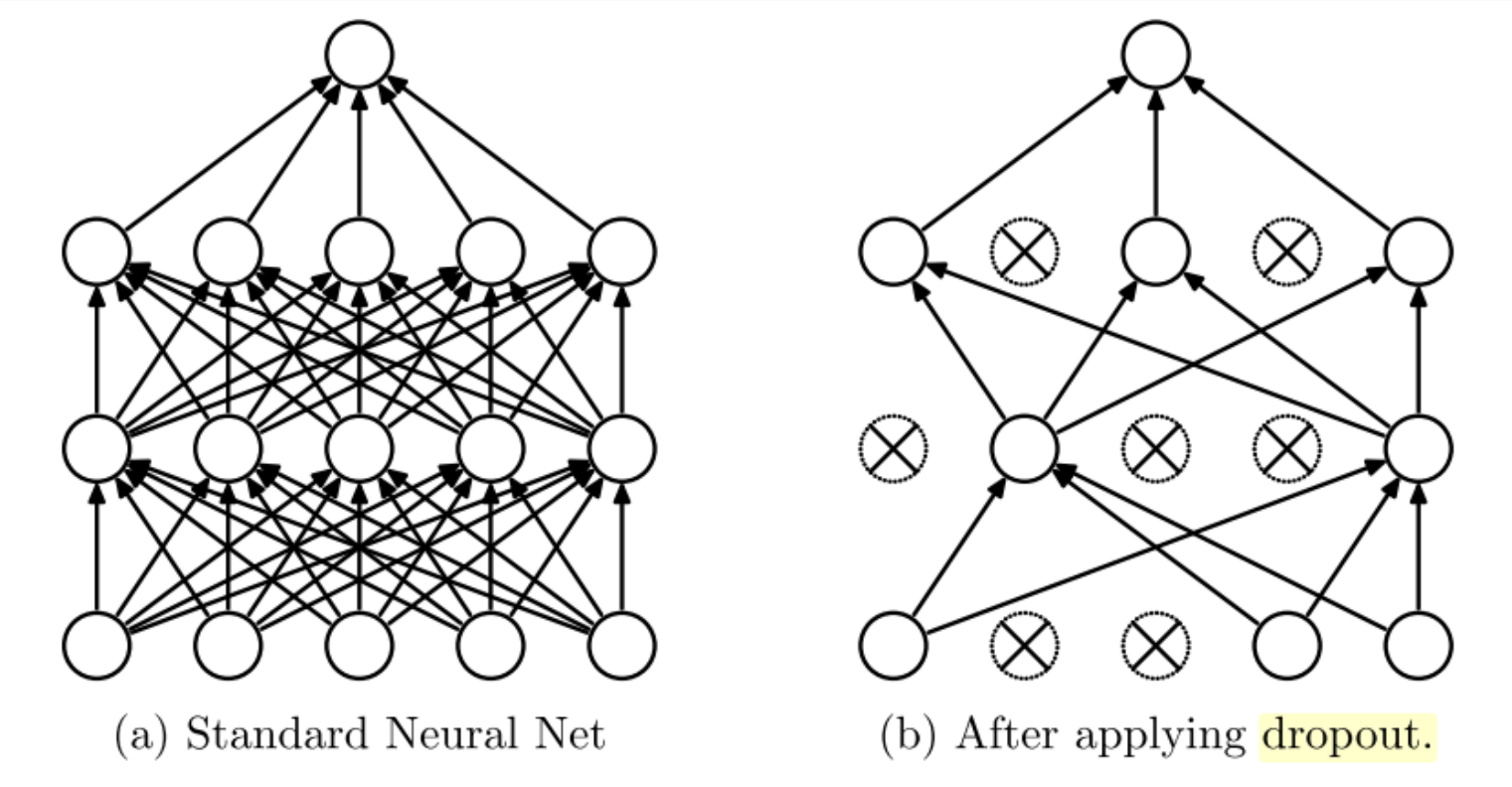
Szegedy et al. 2015


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)



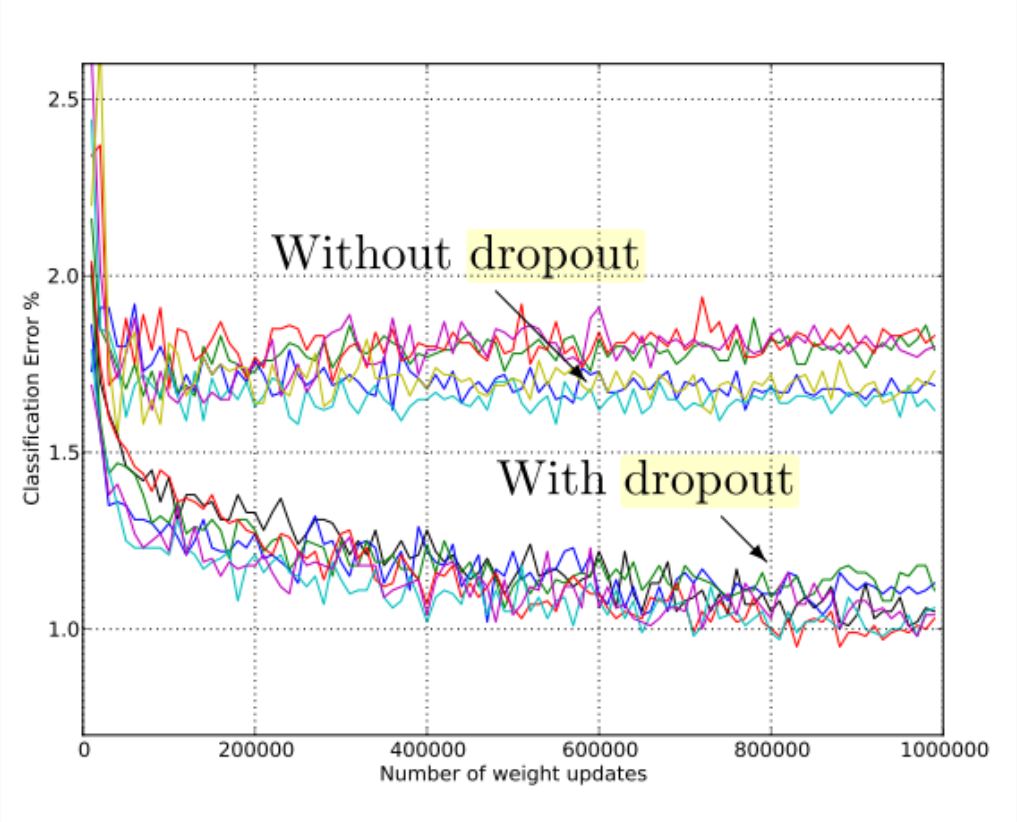
https://becominghuman.ai/

Dropout



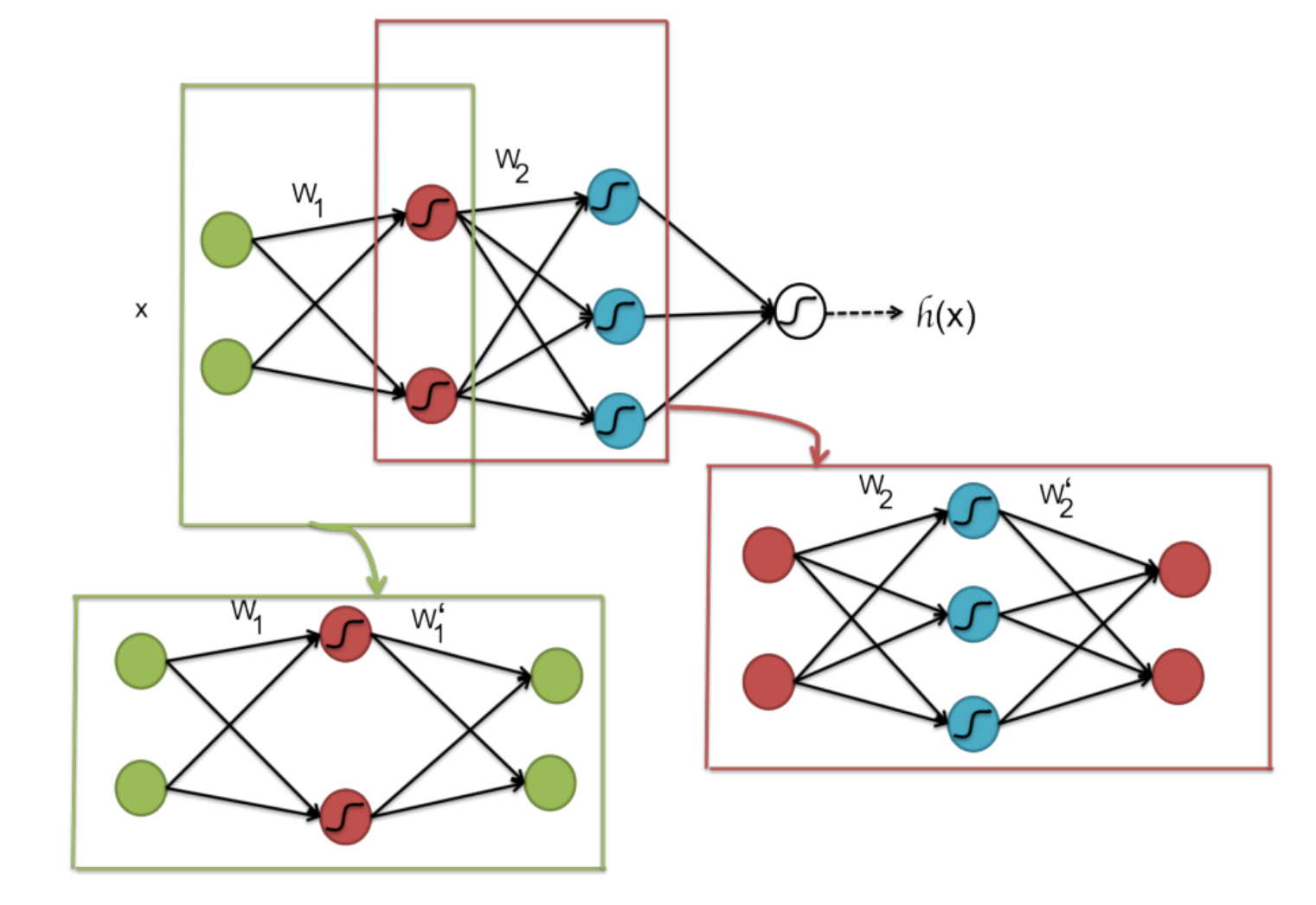
Srivastava et al. 2014

Dropout



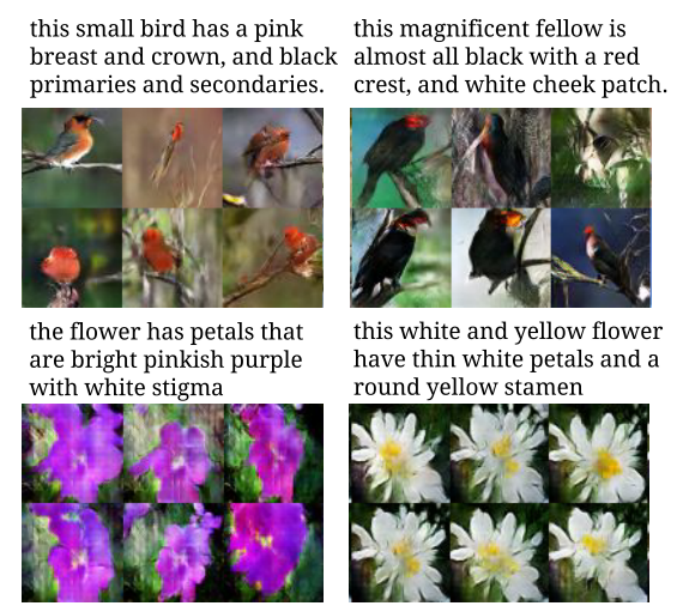
Srivastava et al. 2014

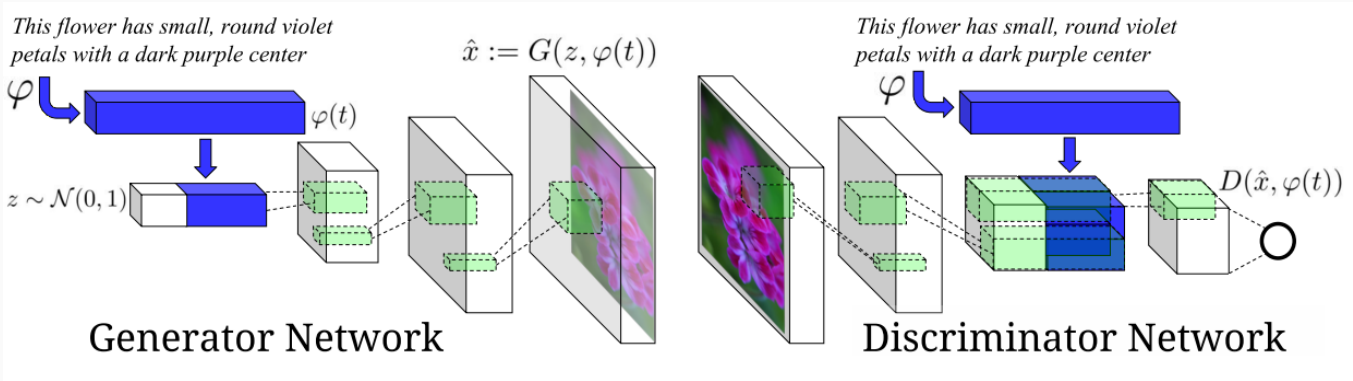
Autoencoding



Le, Brain, and View 2015

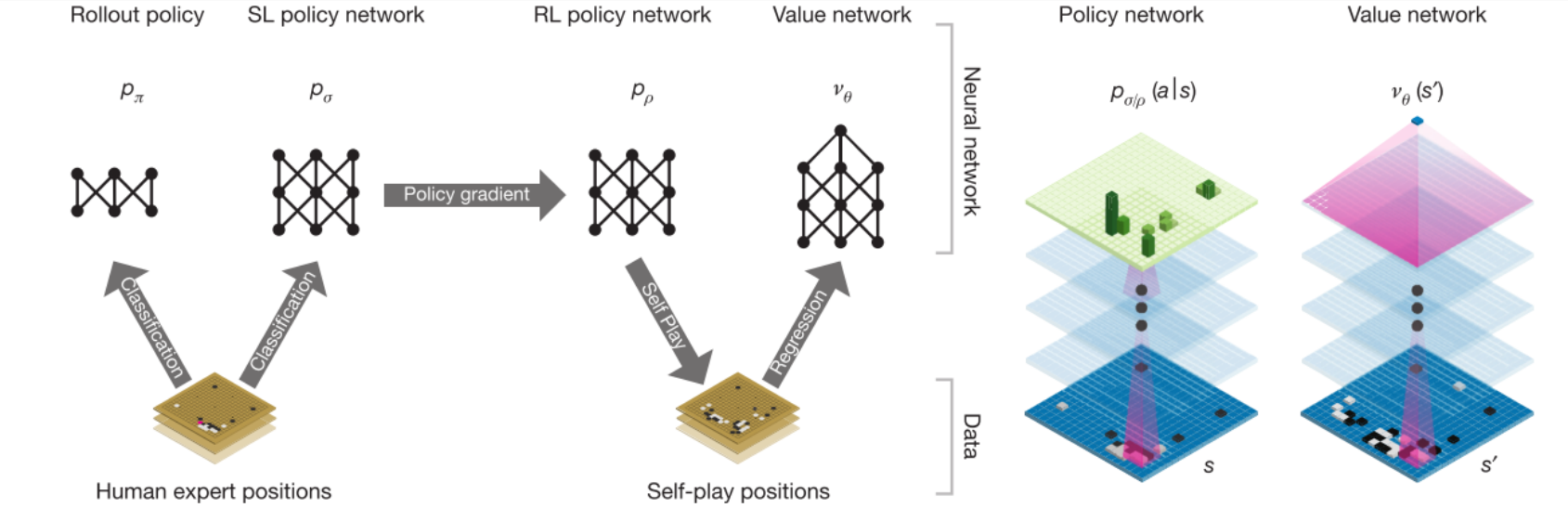
Adversarial training





Reed et al. 2016

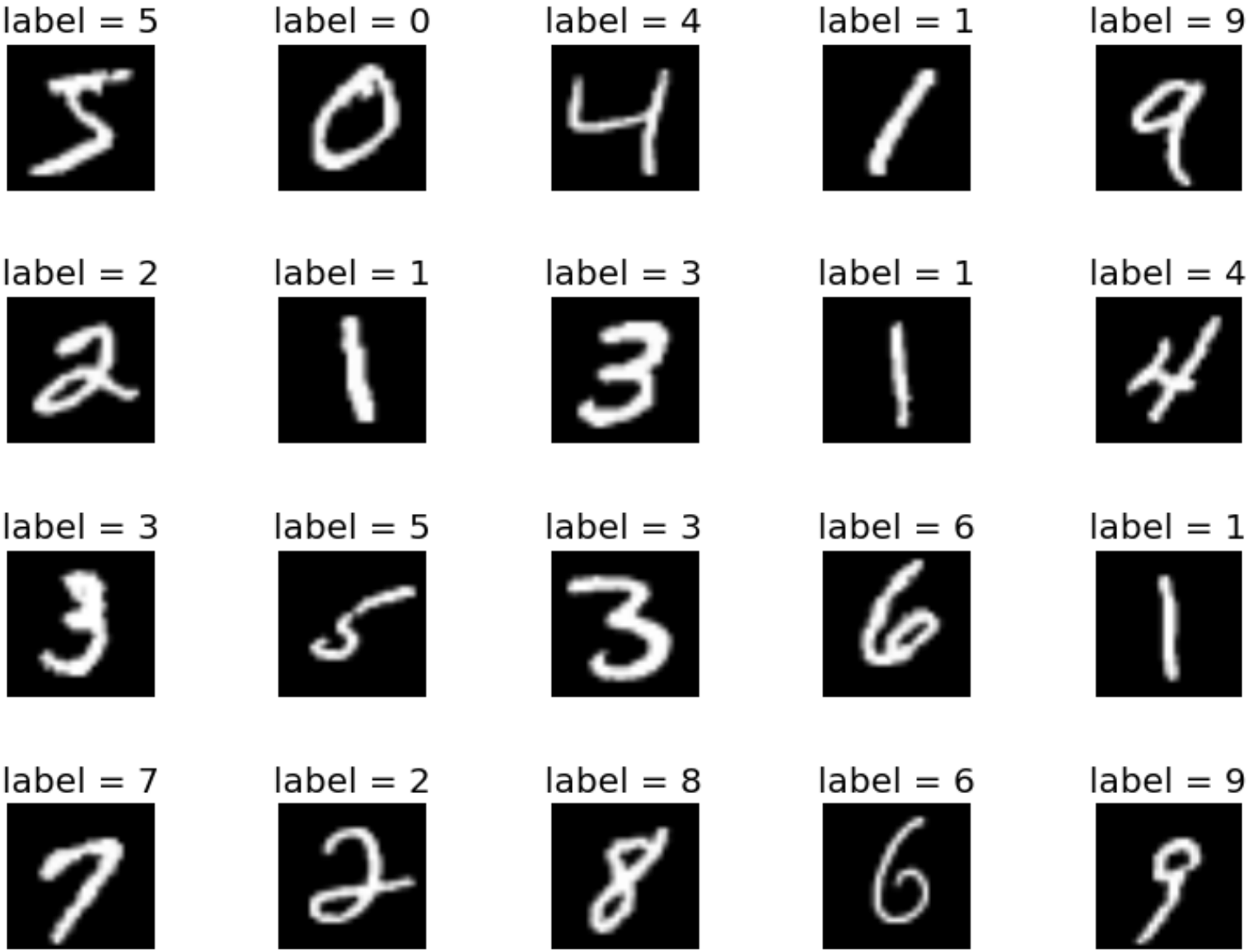
Actor-critic



Silver et al. 2016

Examples

MNIST


label = 5
label = 0
label = 4
label = 1
label = 9
label = 2
label = 1
label = 3
label = 1
label = 4
label = 3
label = 5
label = 3
label = 6
label = 1
label = 7
label = 2
label = 8
label = 6
label = 9

MNIST

```
mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

# Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.matmul(x, W) + b

# Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 10])

# The raw formulation of cross-entropy,
#
#   tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(tf.nn.softmax(y)),
#                                 reduction_indices=[1]))
#
# can be numerically unstable.
#
# So here we use tf.nn.softmax_cross_entropy_with_logits on the raw
# outputs of 'y', and then average across the batch.
cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

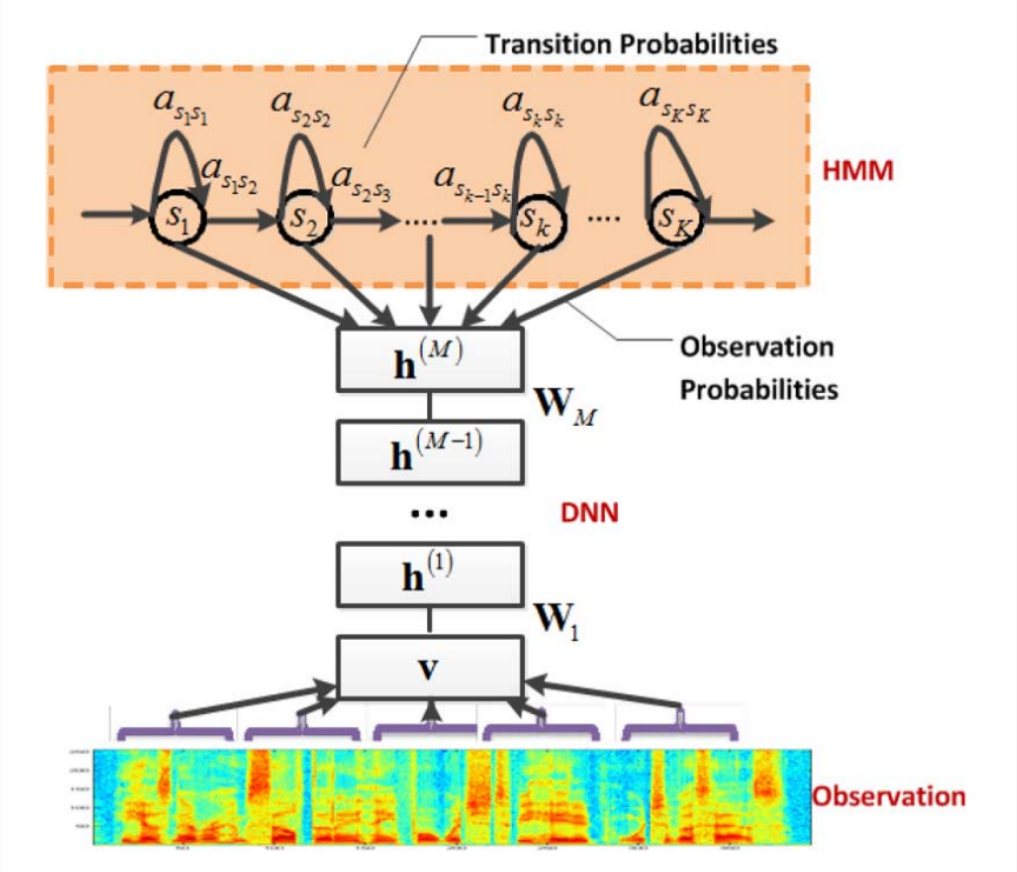
MNIST

```
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()
# Train
for _ in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(sess.run(accuracy, feed_dict={x: mnist.test.images,
                                    y_: mnist.test.labels}))
```

https://www.tensorflow.org/get_started/mnist/pros

DNN-HMM



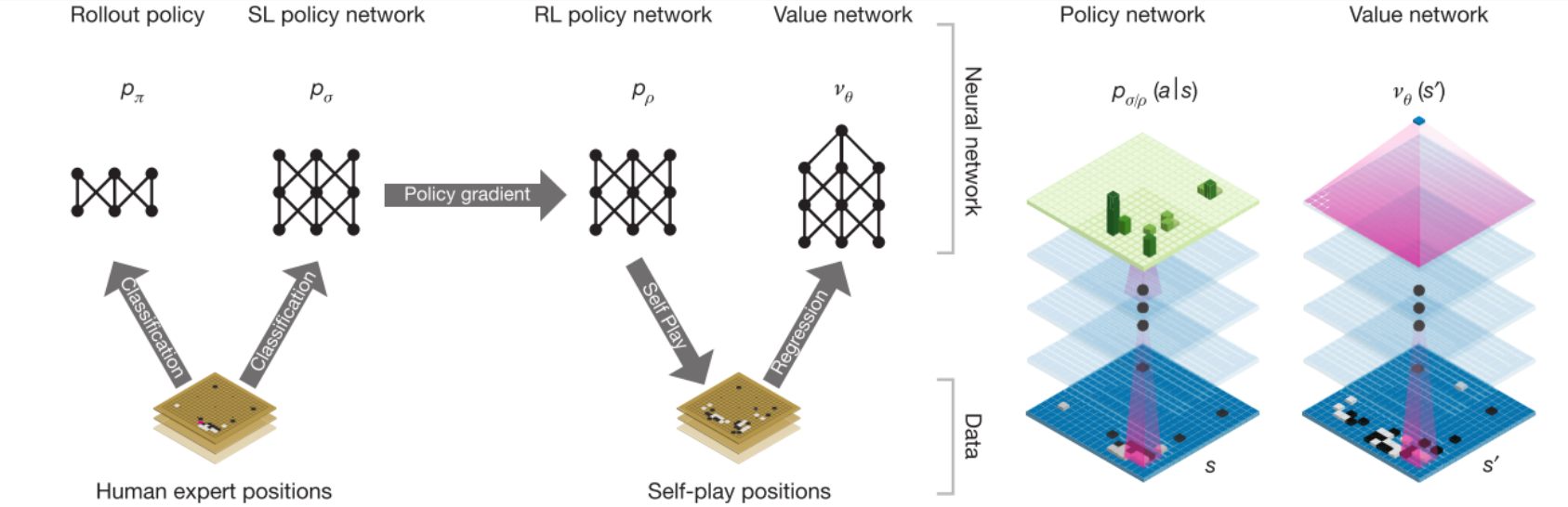
Dahl et al. 2012

DNN vs GMM

Features	Setup	Error Rates
A: TIMIT Phone recognition (3 hours of training)		
GMM	w. Hidden dynamics	24.8%
DNN	5 layers × 2048	23.0%
B: Voice Search SER (24–48 hours of training)		
GMM	MPE (760 24-mix)	36.2%
DNN	5 layers × 2048	30.1%
C: Switch Board WER (309 hours of training)		
GMM	BMMI (9K 40-mix)	23.6%
DNN	7 layers × 2048	15.8%
D: Switch Board WER (2000 hours of training)		
GMM	BMMI (18K 72-mix)	21.7%
DNN	7 layers × 2048	14.6%

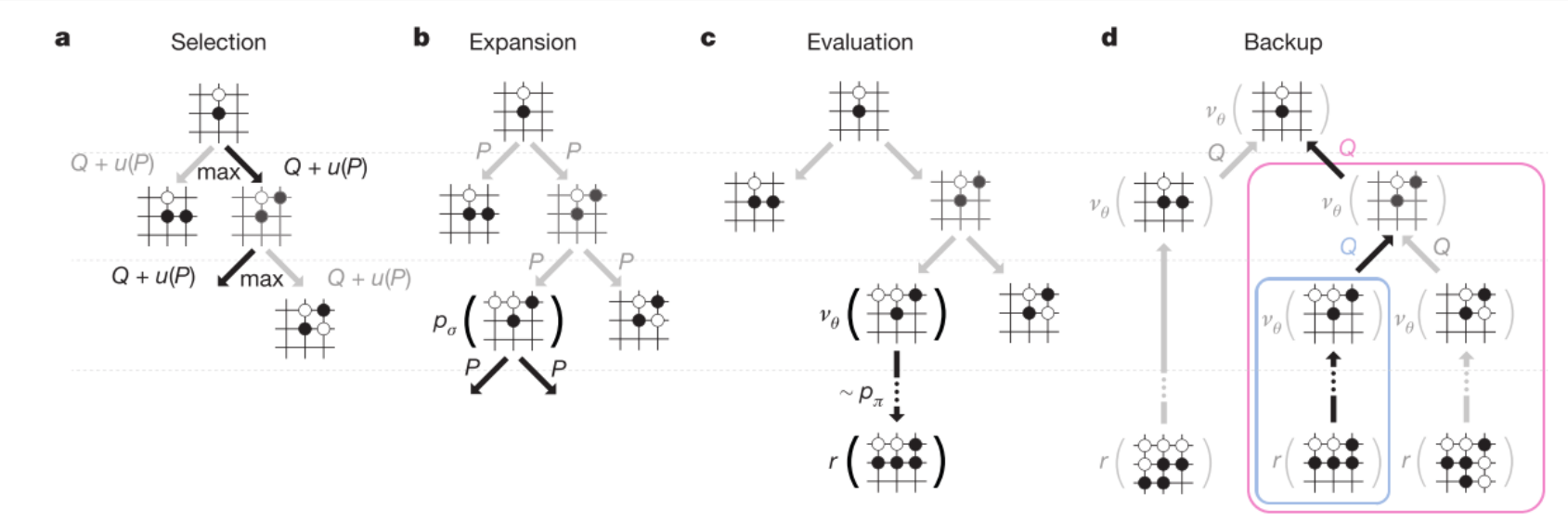
L. Deng, Yu, et al. 2014

Alpha Go



Silver et al. 2016

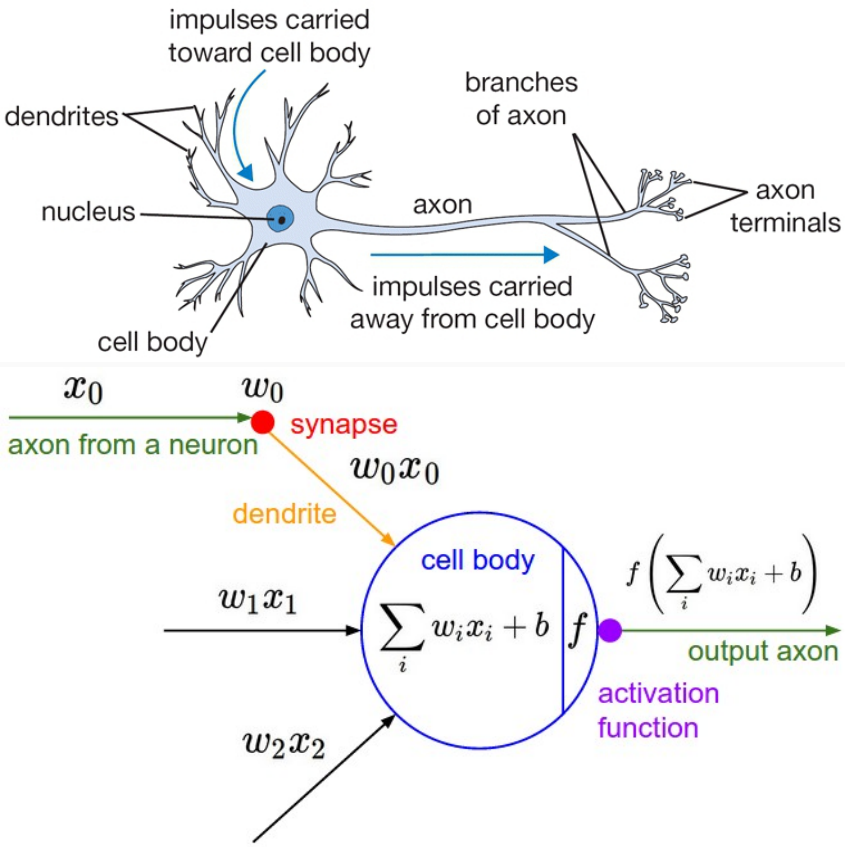
Alpha Go



Silver et al. 2016

Context

Neural model



http://cs231n.github.io

Neural networks

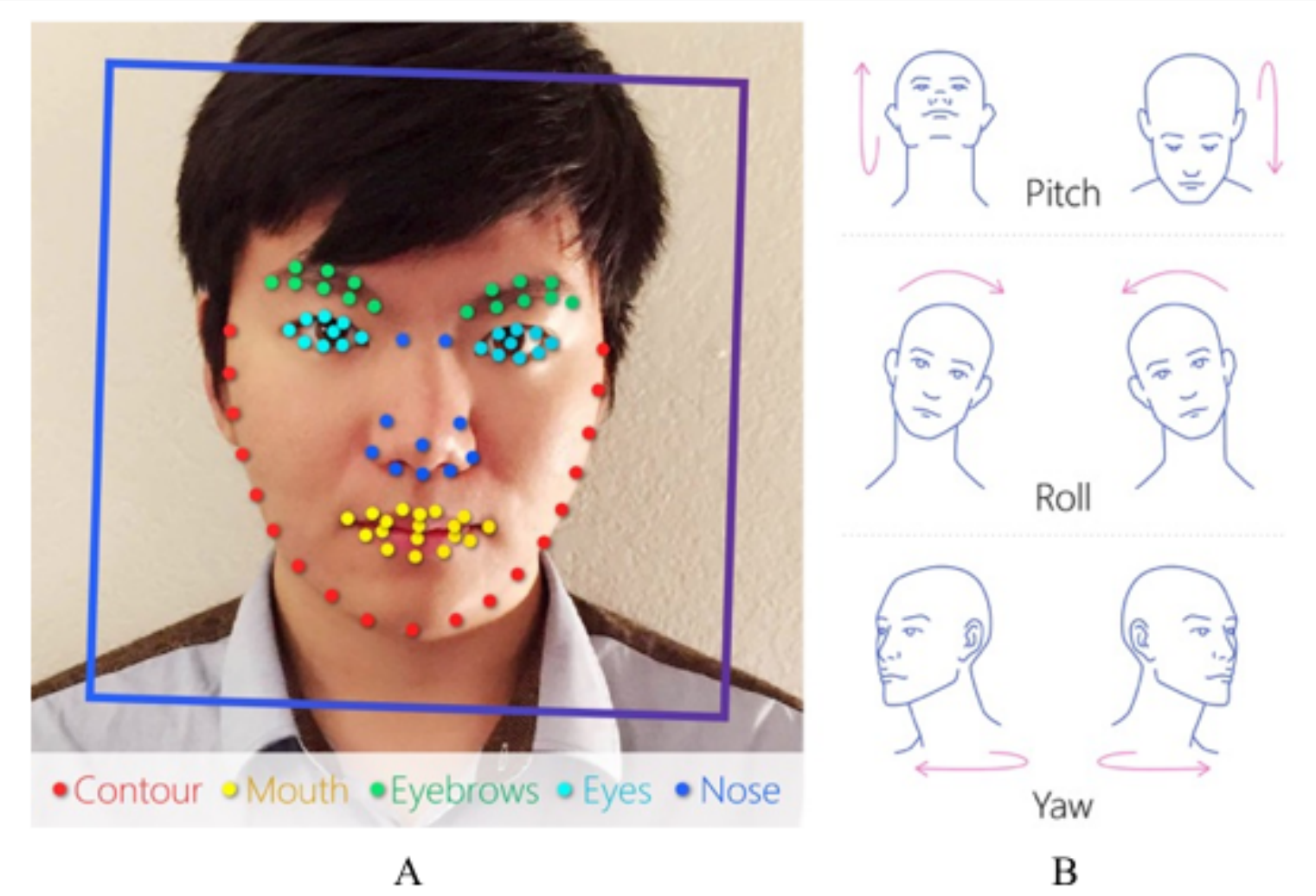
Deep Learning Bio-inspiration

- Activation function: sigmoid
- Network layout: cortical columns
- Training: Hebbian learning
- Adversarial training: Neural darwinism

Many other forms of neural networks

- Spiking neural networks
- Spike time dependent plasticity
- HyperNEAT
- Developmental neural networks

Matz et al. 2017

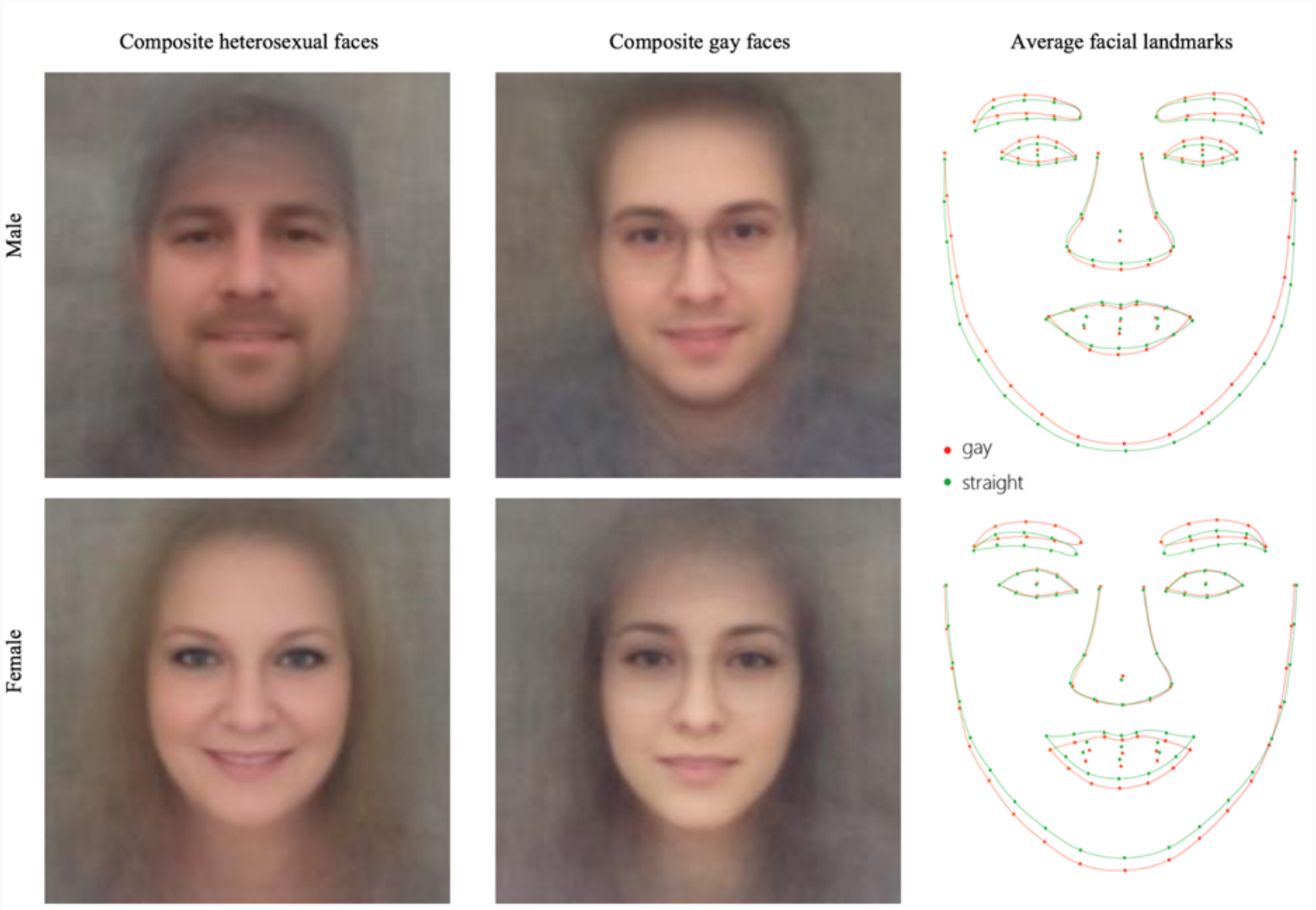


Wang and Kosinski 2017

	Men		Women	
	Gay	Heterosexual	Lesbian	Heterosexual
Unique users	3,947	3,947	3,441	3,441
Median age (IQR)	33 (30–36)	33 (30–36)	29 (25–34)	29 (25–34)
Total images	8,996	8,645	7,457	10,228
Users with at least:				
1 image	3,947	3,947	3,441	3,441
2 images	2,438	2,439	2,037	2,878
3 images	1,363	1,367	1,058	1,951
4 images	562	731	494	1,114
5 images	219	327	223	491

Note. IQR stands for interquartile range.

Wang and Kosinski 2017 “Given a single facial image, a classifier could correctly distinguish 34 between gay and heterosexual men in 81% of cases, and in 71% of cases for women. Human 35 judges achieved much lower accuracy: 61% for men and 54% for women. The accuracy of the 36 algorithm increased to 91% and 83%, respectively, given five facial images per person.”



Wang and Kosinski 2017

Sensationalism and the long history of AI

https://www.youtube.com/watch?v=aygSMgK3BEM

References

References

Abedini, R et al. (2012). “The prediction of undersaturated crude oil viscosity: An artificial neural network and fuzzy model approach”. In: *Petroleum Science and Technology* 30.19, pp. 2008–2021.

Berthelot, David, Tom Schumm, and Luke Metz (2017). “Began: Boundary equilibrium generative adversarial networks”. In: *arXiv preprint arXiv:1703.10717*.

Dahl, George E et al. (2012). “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition”. In: *IEEE Transactions on audio, speech, and language processing* 20.1, pp. 30–42.

Deng, Li, Dong Yu, et al. (2014). “Deep learning: methods and applications”. In: *Foundations and Trends® in Signal Processing* 7.3–4, pp. 197–387.

- Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.
- Le, Quoc V (2014). “A Tutorial on Deep Learning Part 1: Nonlinear Classifiers and The Backpropagation Algorithm”. In: pp. 1–18.
- Le, Quoc V, Google Brain, and Mountain View (2015). “A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks”. In: pp. 1–20.
- LeCun, Yann et al. (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.
- Matz, SC et al. (2017). “Psychological targeting as an effective approach to digital mass persuasion”. In: *Proceedings of the National Academy of Sciences*, p. 201710966.
- Nielsen, Michael A (2015). *Neural networks and deep learning*.


Reed, Scott et al. (2016). “Generative Adversarial Text to Image Synthesis”. In: *Icml*, pp. 1060–1069. eprint: **1605.05396**.


Russakovsky, Olga et al. (2015). “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3, pp. 211–252.


Silver, David et al. (2016). “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587, pp. 484–489. ISSN: 0028-0836.


Srivastava, Nitish et al. (2014). “Dropout : A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research (JMLR)* 15, pp. 1929–1958. ISSN: 15337928. eprint: **1102.4807**.


Szegedy, Christian et al. (2015). “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.


Wang, Yilun and Michal Kosinski (2017). “Deep neural networks are more accurate than humans at detecting sexual orientation from facial images.”. In: