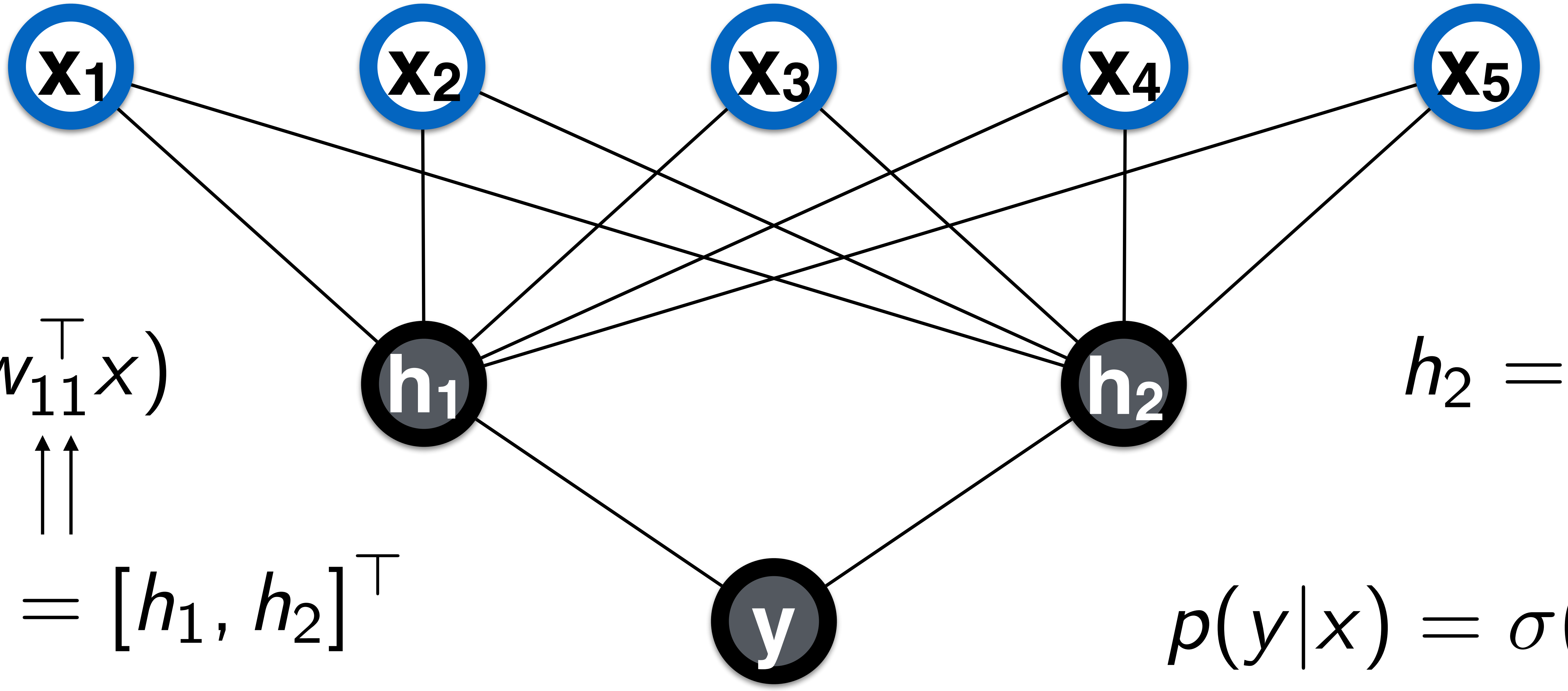
Deep Learning Overview
Part 1

Machine Learning
CS4824/ECE4424
Bert Huang
Virginia Tech

Outline

- **Review of neural networks**
- **New advances**
- **Popular neural network structures in modern applications**
- Sequence-to-sequence models
- Generative adversarial learning
- Open questions

Multi-Layered Perceptron



$$h_1 = \sigma(w_{11}^\top x)$$

$$\Uparrow\!\Uparrow$$

$$h = [h_1, h_2]^\top$$

$$h_2 = \sigma(w_{12}^\top x)$$

$$p(y|x) = \sigma(w_{21}^\top h)$$

$$p(y|x) = \sigma\left(w_{21}^\top \left[\sigma(w_{11}^\top x), \sigma(w_{12}^\top x)\right]^\top\right)$$

Matrix Gradient Recipe

$h_1 = s(W_1 x)$

$h_i = s(W_i h_{i-1})$

$f(x, W) = s(W_m h_{m-1})$

$$J(W) = \ell(f(x, W))$$

Feed Forward Propagation

$$\delta_i = (W_{i+1}^\top \delta_{i+1}) \odot s'(W_i h_{i-1})$$

$$\delta_m = \ell'(f(x, W))$$

$$\nabla_{W_1} J = \delta_1 x^\top$$

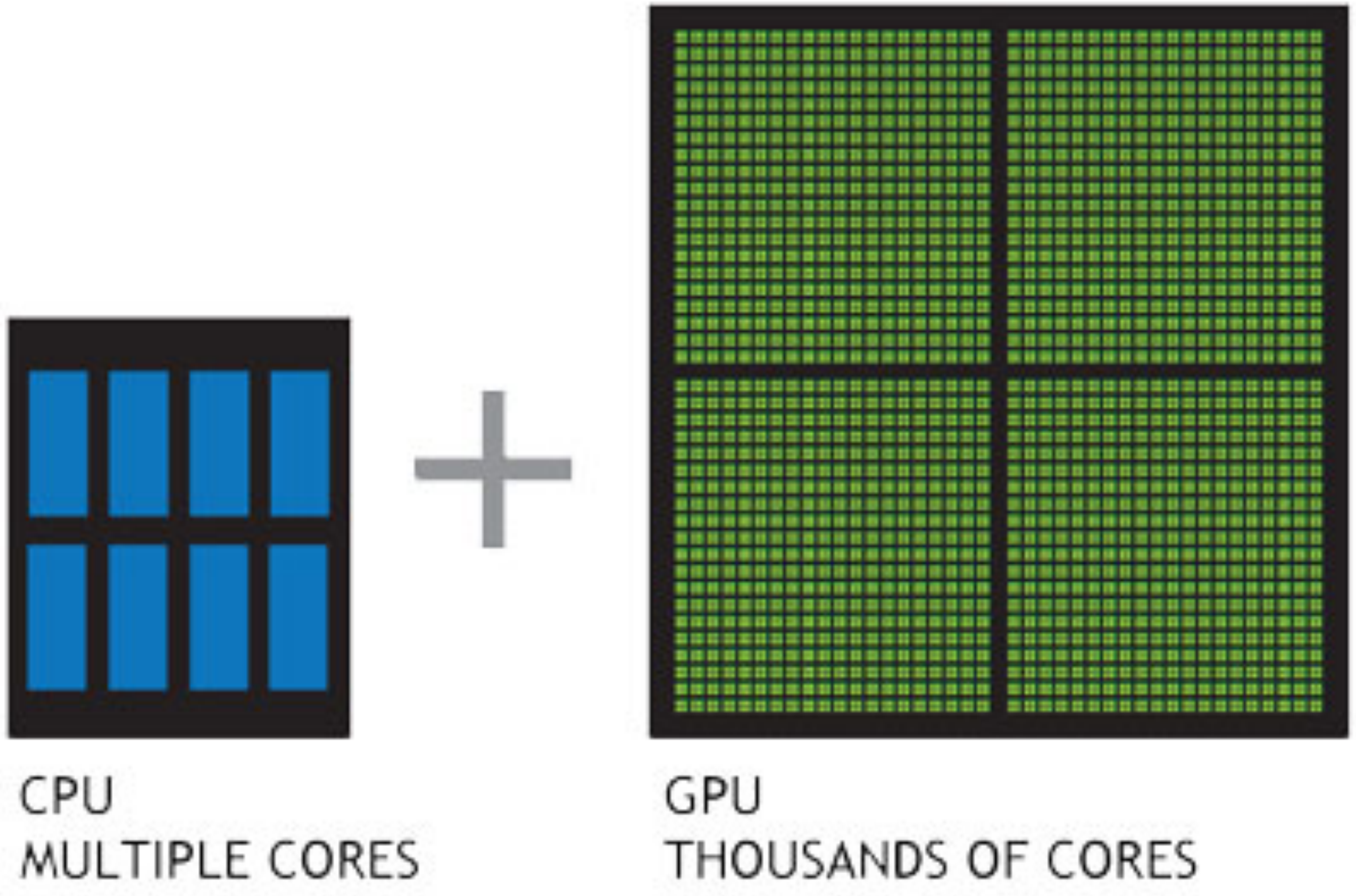
$$\nabla_{W_i} J = \delta_i h_{i-1}^\top$$

Back Propagation

Dataset Size Example

- 2001: Caltech ~4,000 images, four categories
- 2004: Caltech101 ~9,000 images, 101 categories
- 2006: Caltech256 ~31,000 images, 256 categories
- 2009: ImageNet ~ 14,000,000 images, ~1,000 main categories

GPU Computing

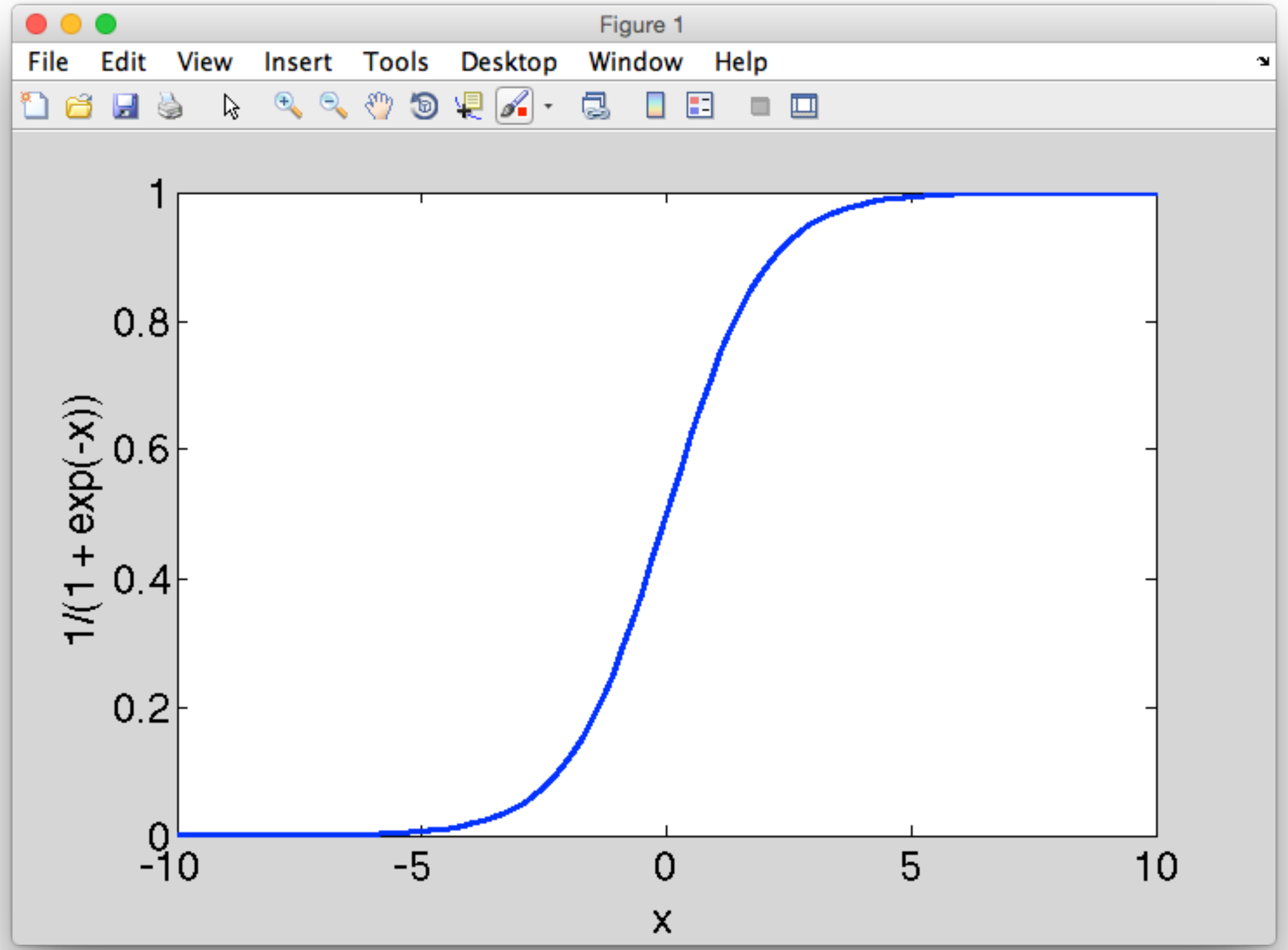


http://www.nvidia.com/object/what-is-gpu-computing.html

Vanishing Gradients

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

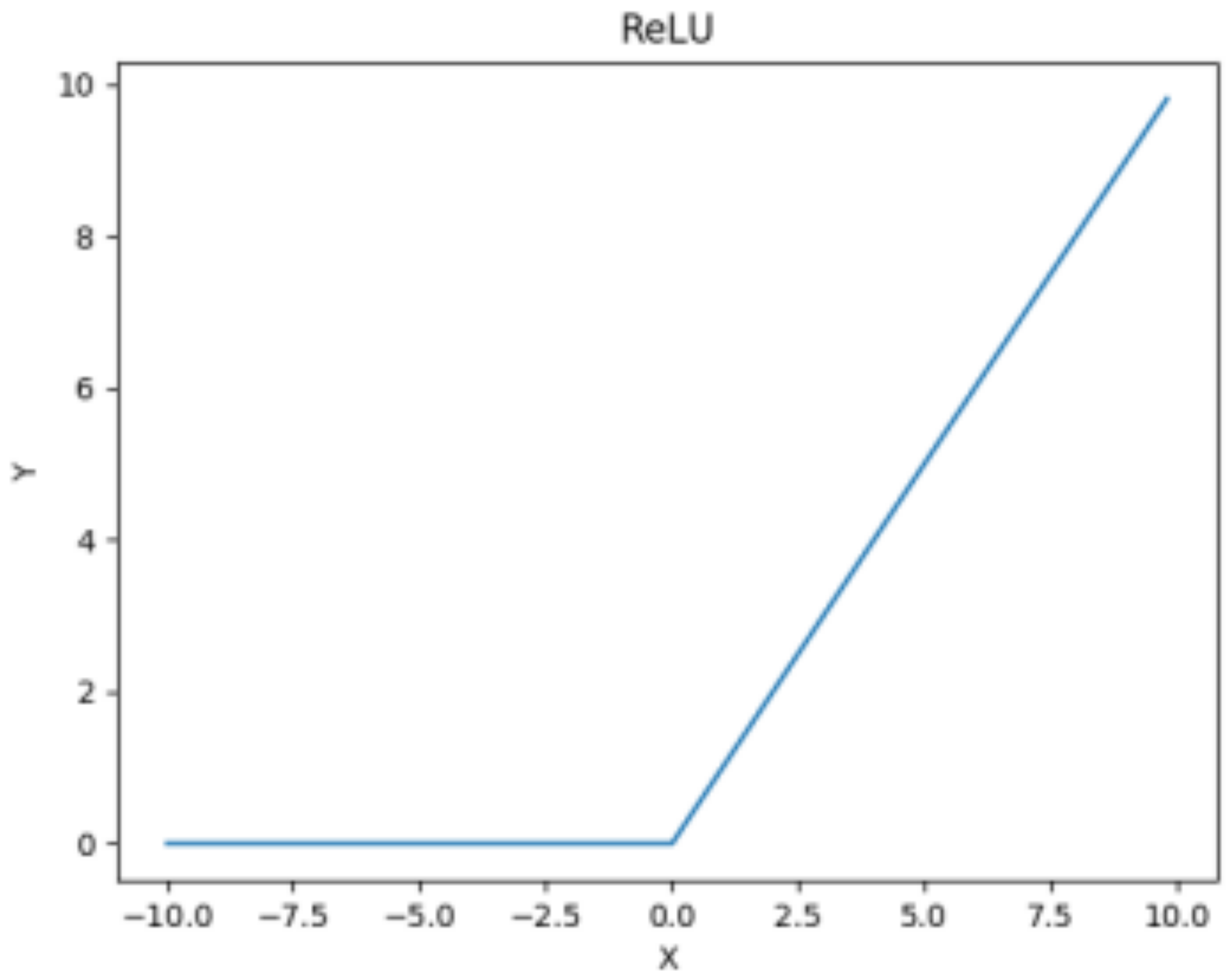
$$\frac{d\ \sigma(x)}{d\ x} = \sigma(x)(1 - \sigma(x))$$



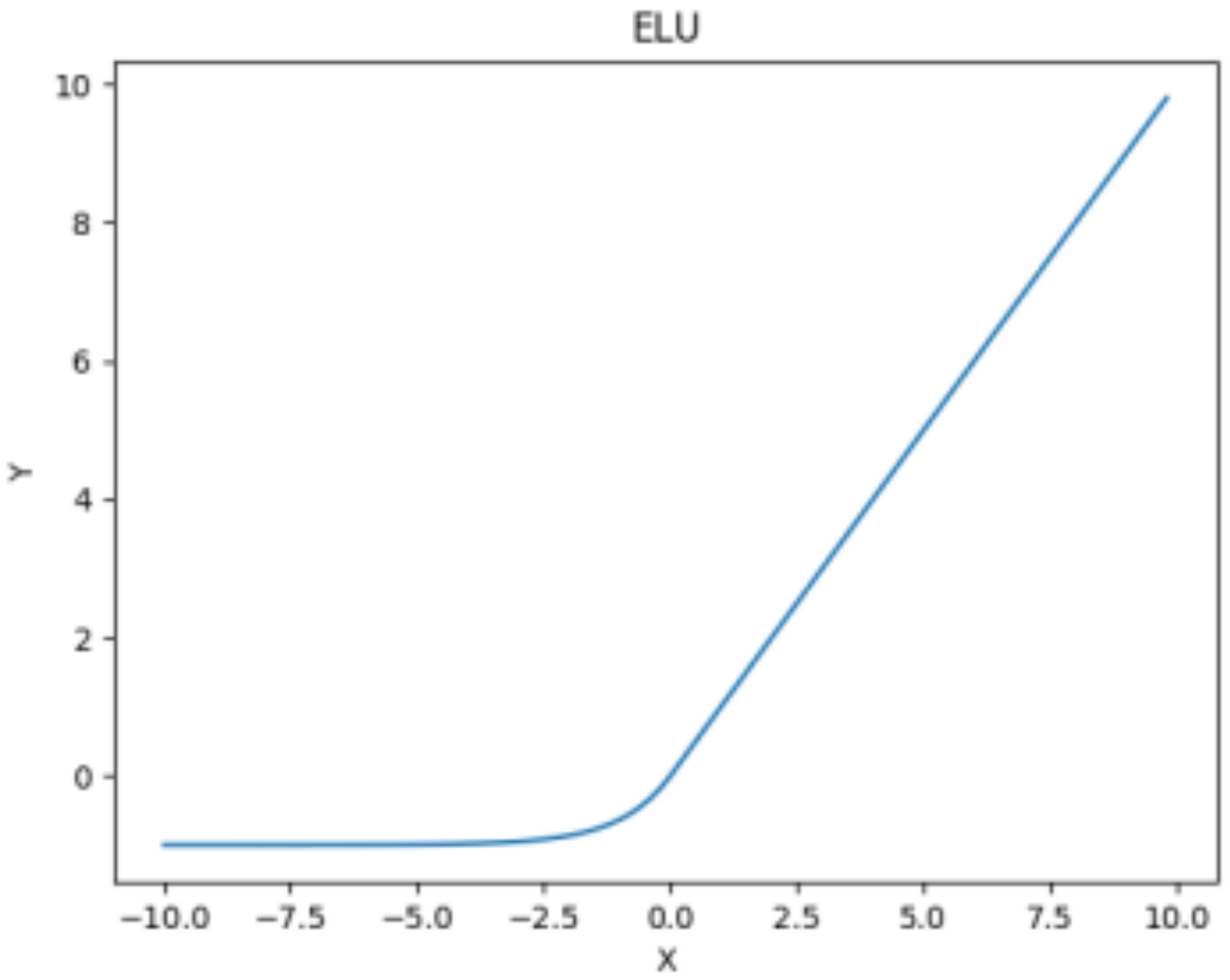
(Some) New Advances

- Activation/squashing functions
- Stochastic optimization algorithms
- Dropout regularization
- Batch normalization
- Automatic differentiation

Modern Squashing Functions



$$f(x) = \max(0, x)$$



$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ a(e^x - 1) & \text{otherwise} \end{cases}$$

plots from http://saikatbasak.in/sigmoid-vs-relu-vs-elu/

Stochastic Optimization

- Gradient descent on randomly chosen batch of examples

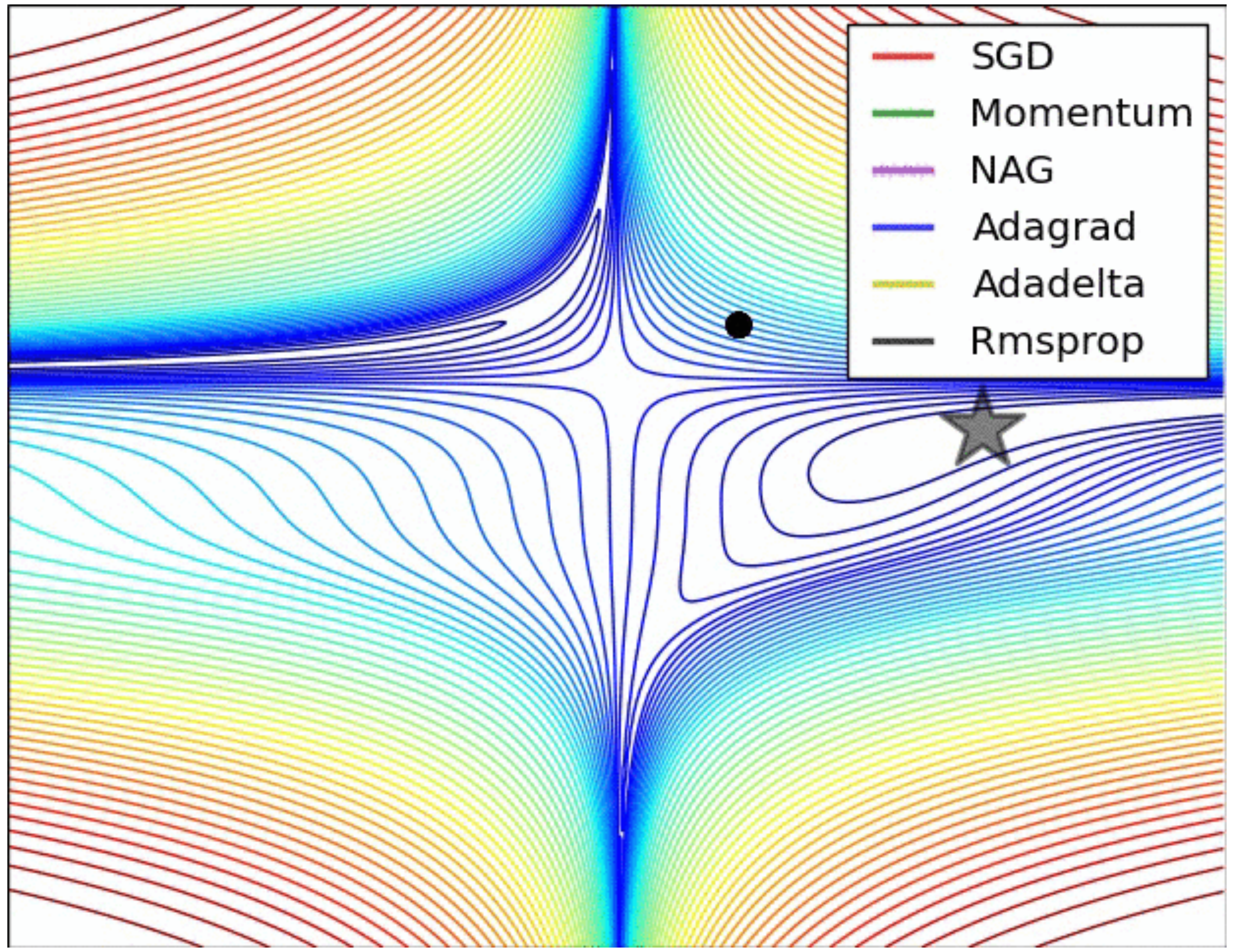
$$\theta \leftarrow \theta - \eta \nabla_\theta J(\theta; x_i, y_i)$$

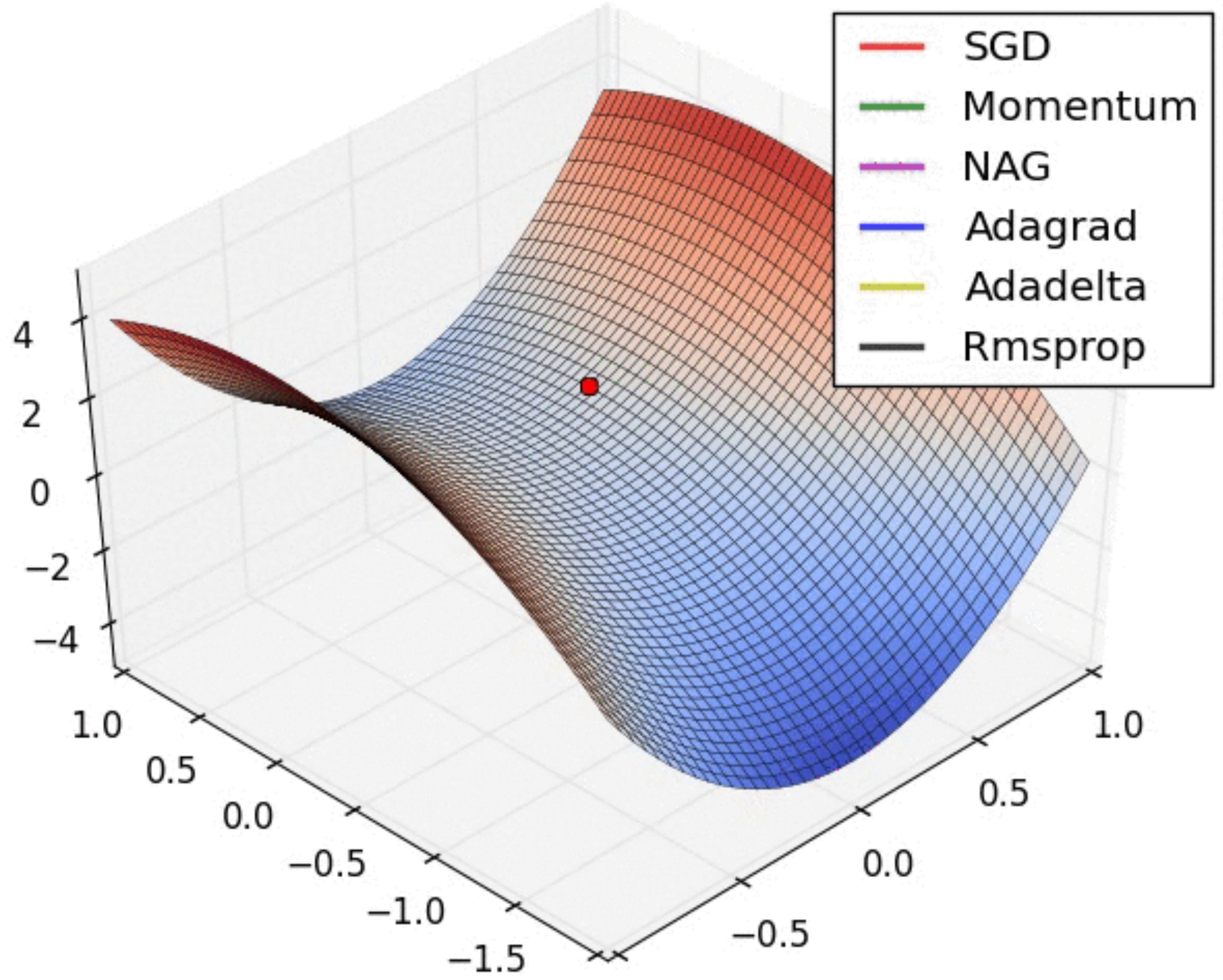
- Sensitive to learning rate. Adaptive learning rate methods, e.g., adagrad

$$\theta \leftarrow \theta - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla_\theta J(\theta; x_i, y_i)$$

- Adam and RMSProp add adaptive momentum, many other extensions

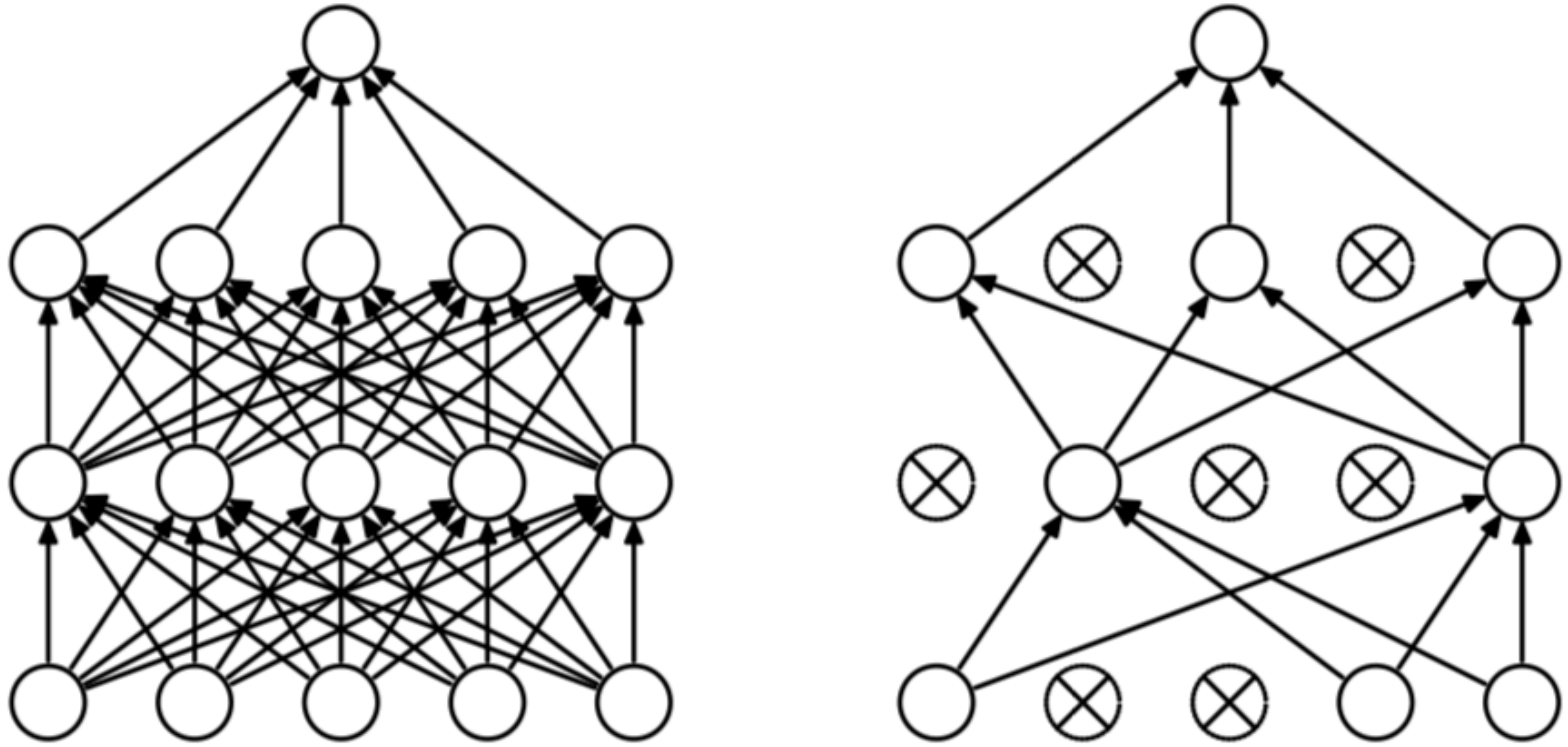
Stochastic Optimization





http://ruder.io/optimizing-gradient-descent/

Dropout Regularization



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

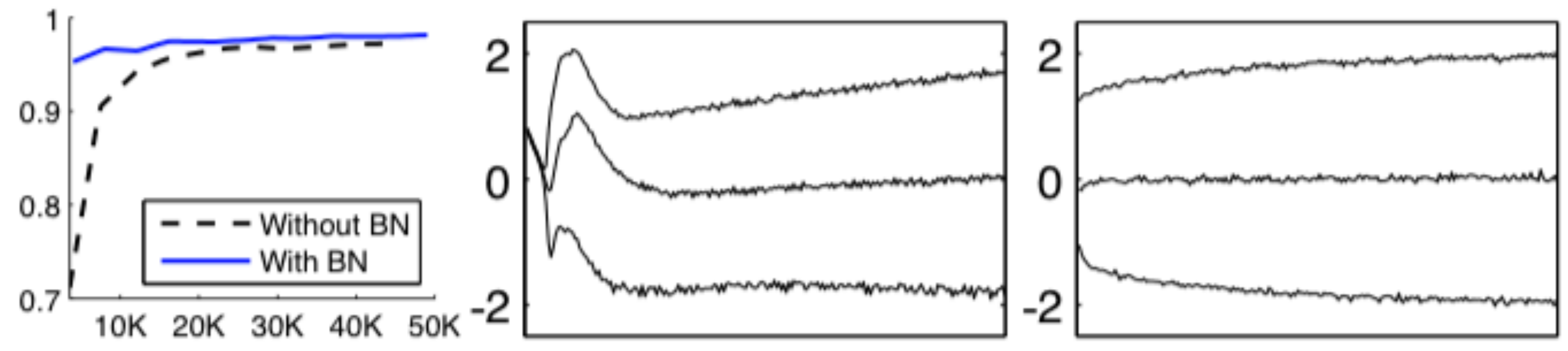
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

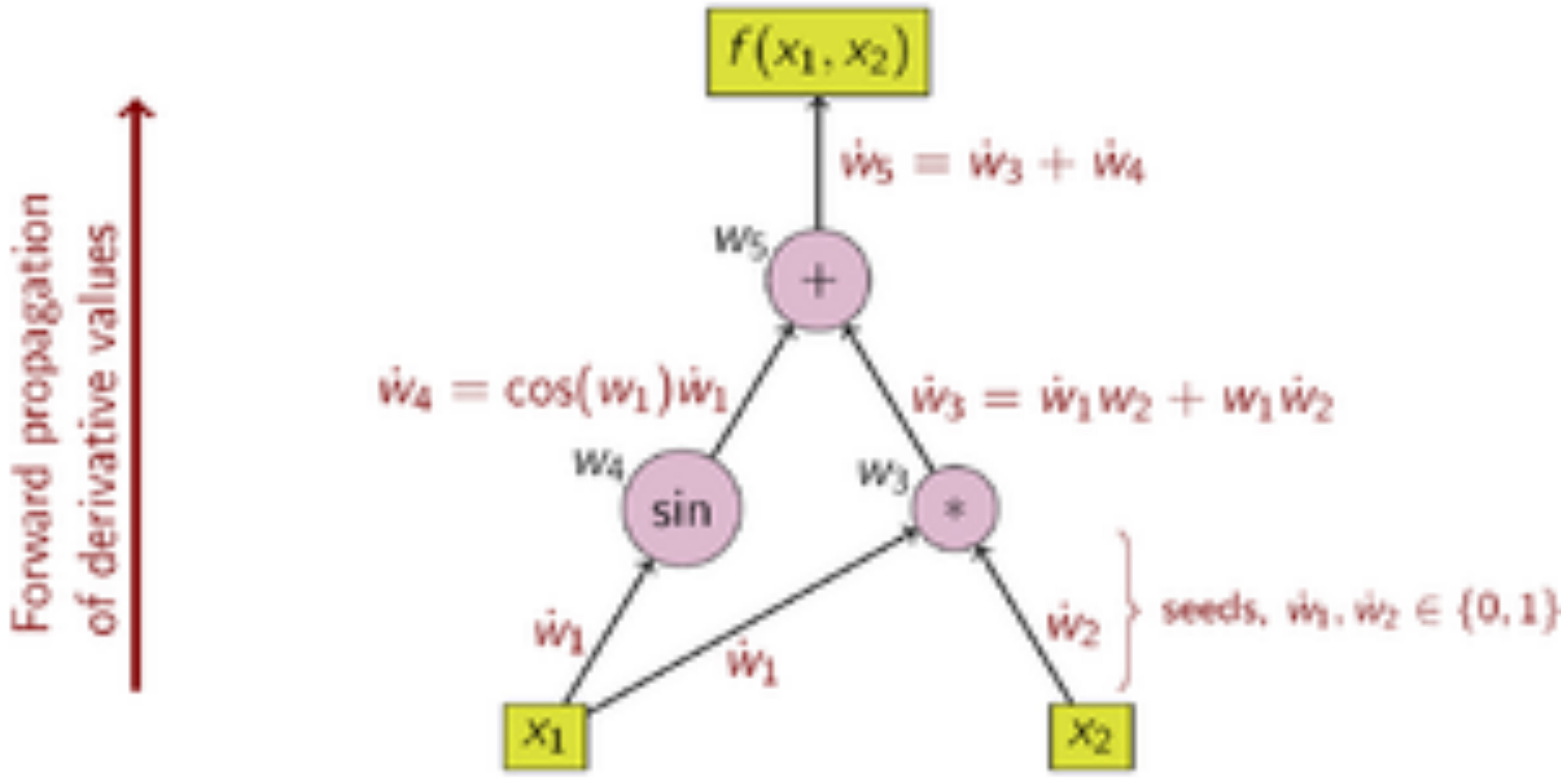


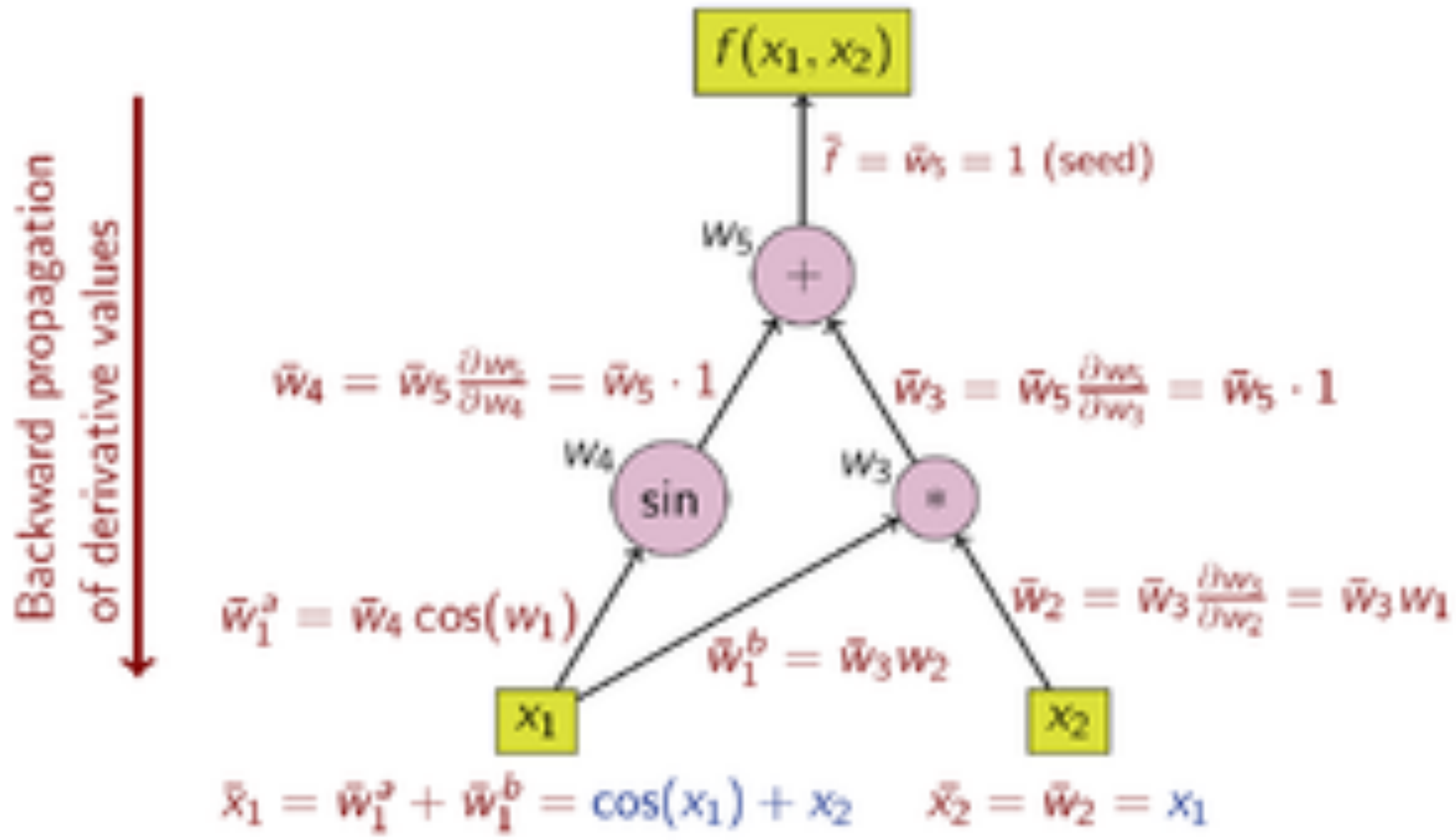
(a) (b) Without BN (c) With BN

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

https://arxiv.org/pdf/1502.03167v3.pdf

Automatic Differentiation



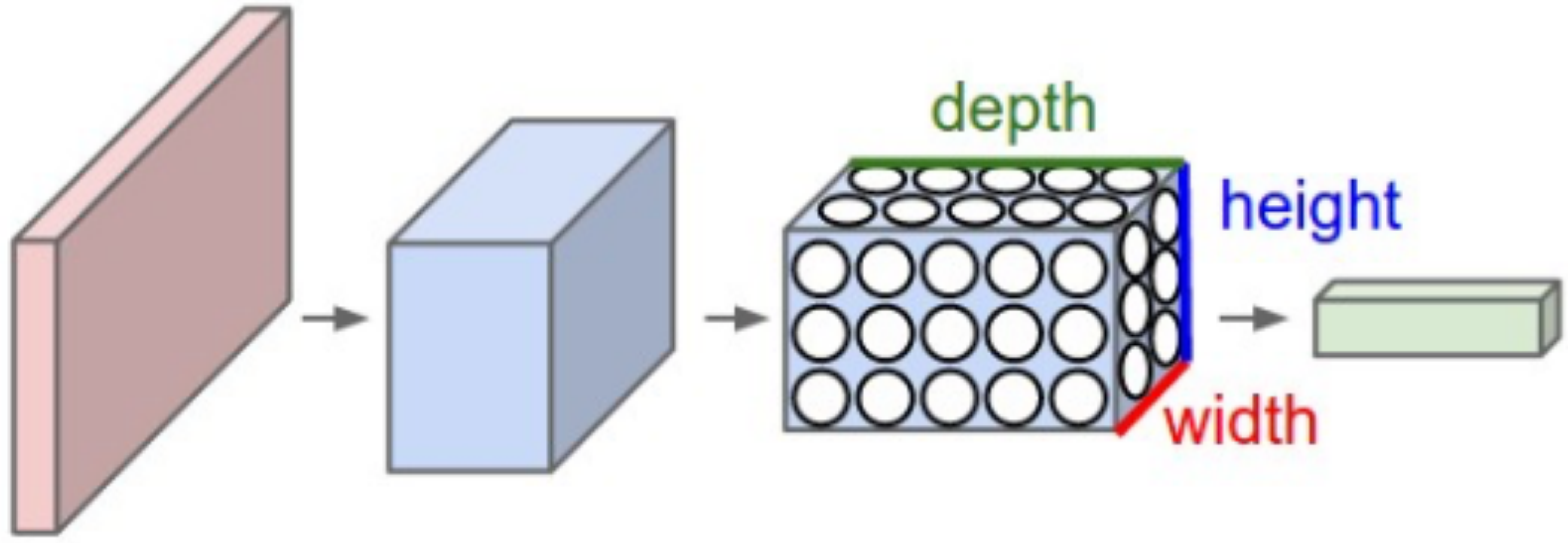


https://en.wikipedia.org/wiki/Automatic_differentiation

Popular Neural Network Architectures

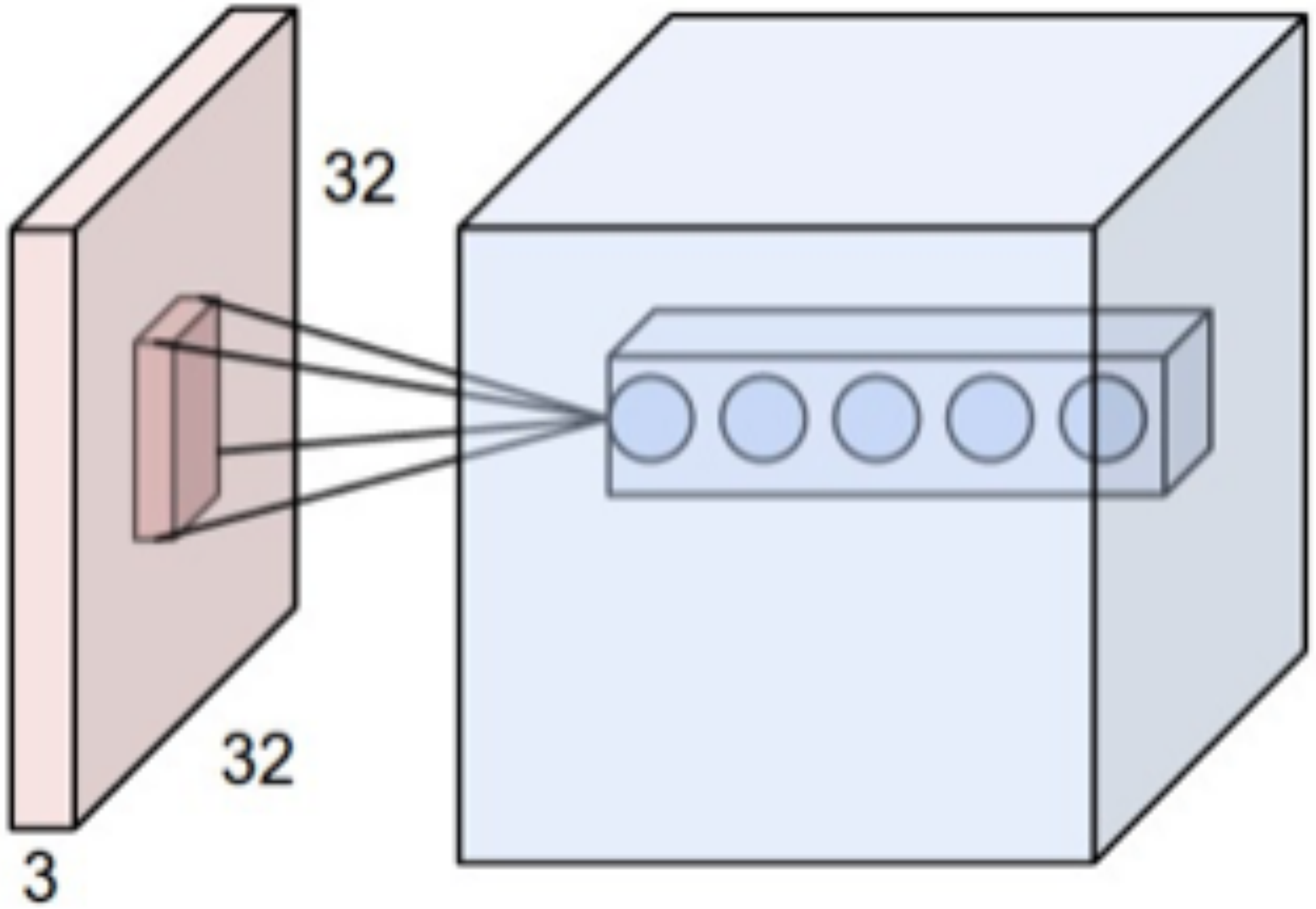
- Convolutional neural networks (ConvNets, CNNs)
- Recurrent neural networks (RNNs)
  - Long short term memory (LSTM) networks

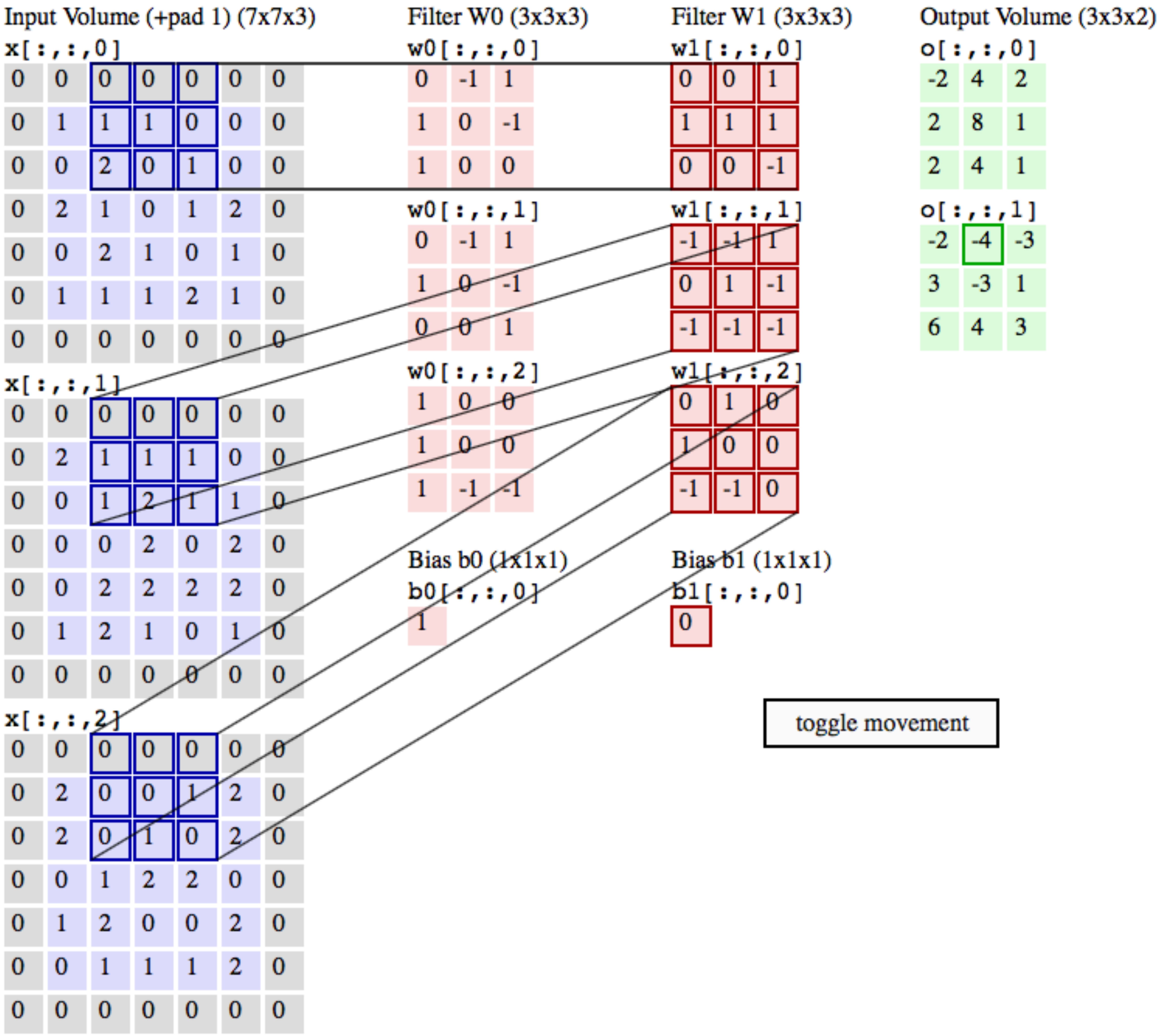
Convolutional Neural Networks



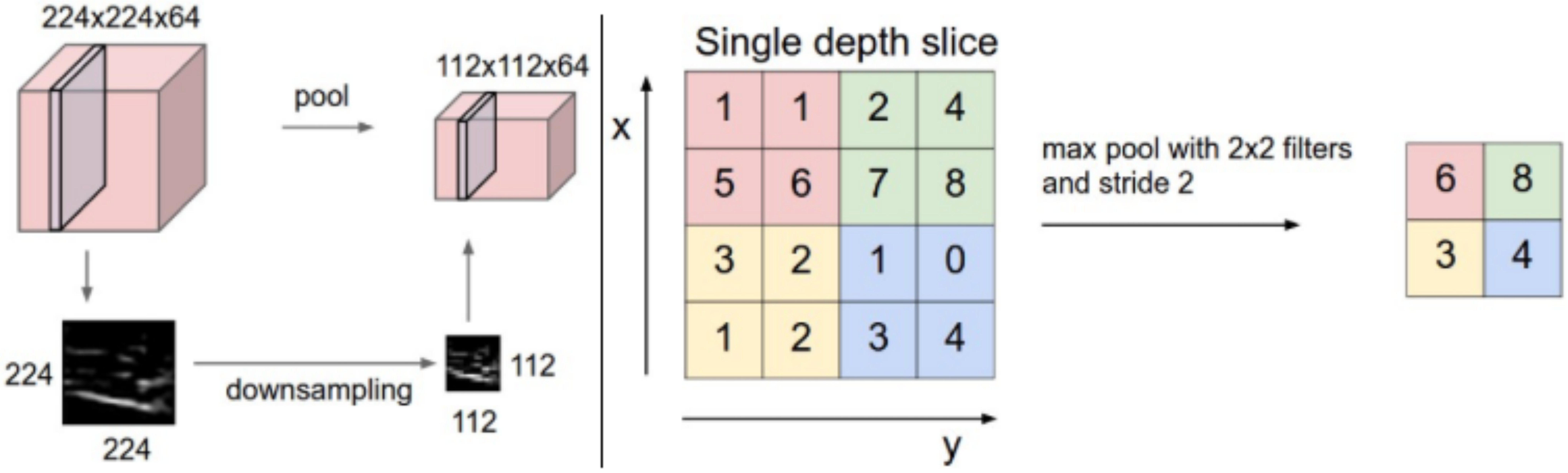
Figures from http://cs231n.github.io/convolutional-networks/

Convolutional Layers



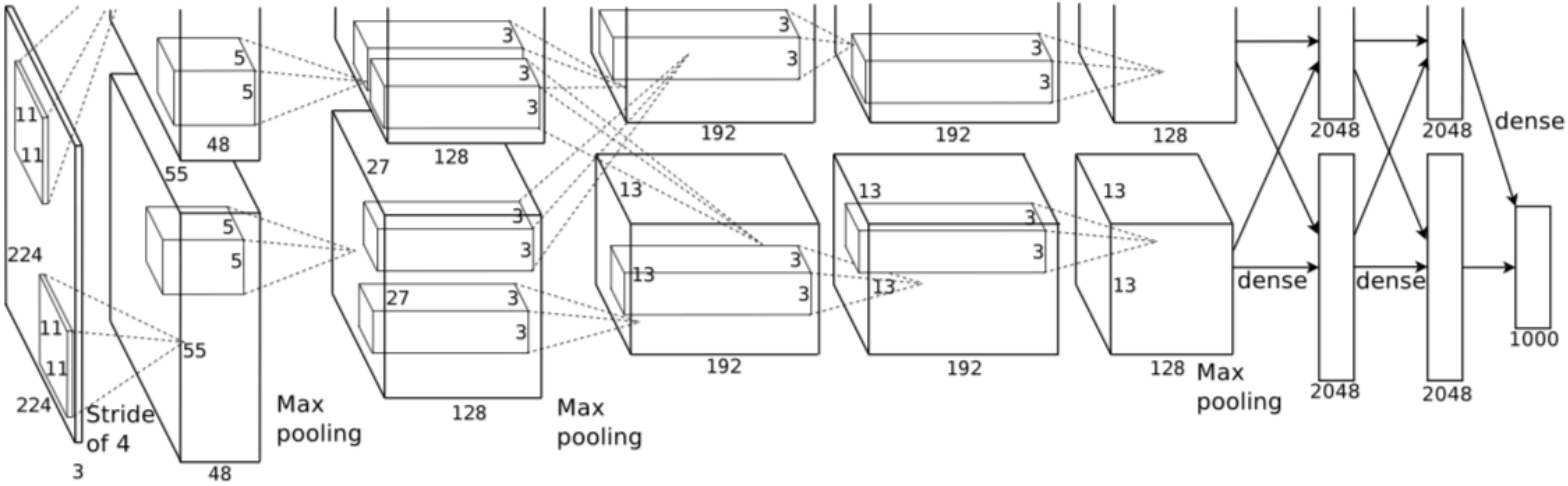

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 1 1 0 0 0
0 0 2 0 1 0 0
0 2 1 0 1 2 0
0 0 2 1 0 1 0
0 1 1 1 2 1 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 1 1 1 0 0
0 0 1 2 1 1 0
0 0 0 2 0 2 0
0 0 2 2 2 2 0
0 1 2 1 0 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 0 0 1 2 0
0 2 0 1 0 2 0
0 0 1 2 2 0 0
0 1 2 0 0 2 0
0 0 1 1 1 2 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 -1 1
1 0 -1
1 0 0
w0[:,:,1]
0 -1 1
1 0 -1
0 0 1
w0[:,:,2]
1 0 0
1 0 0
1 -1 -1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 0 1
1 1 1
0 0 -1
w1[:,:,1]
-1 -1 1
0 1 -1
-1 -1 -1
w1[:,:,2]
0 1 0
1 0 0
-1 -1 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-2 4 2
2 8 1
2 4 1
o[:,:,1]
-2 -4 -3
3 -3 1
6 4 3
toggle movement

Pooling Layers




POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

AlexNet

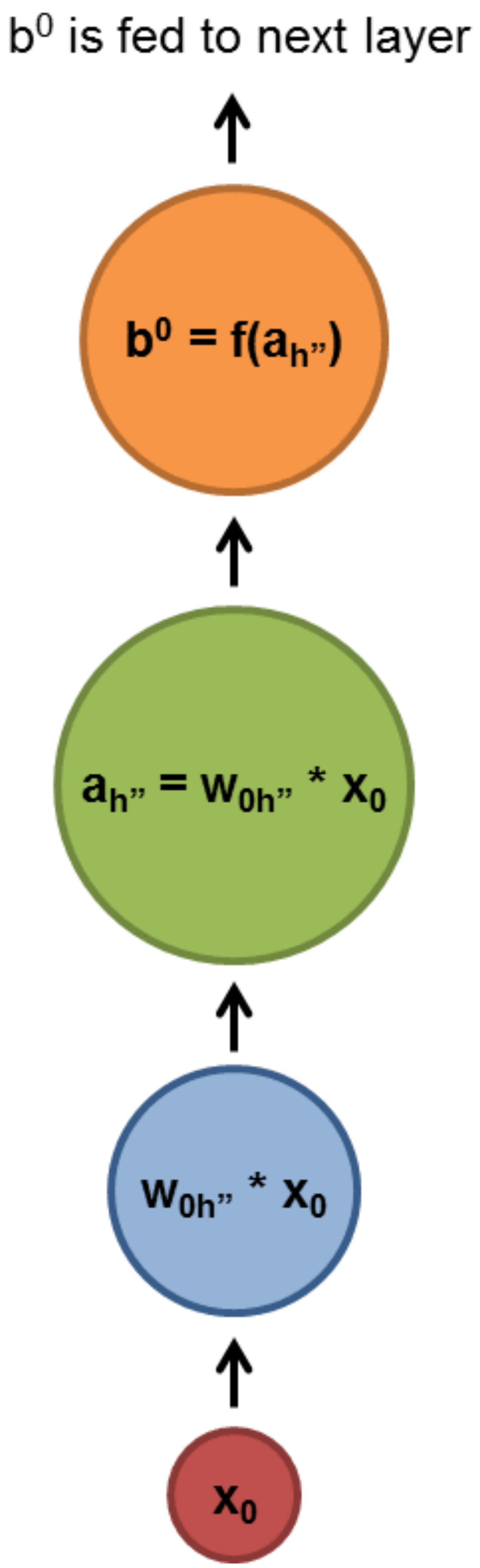


http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

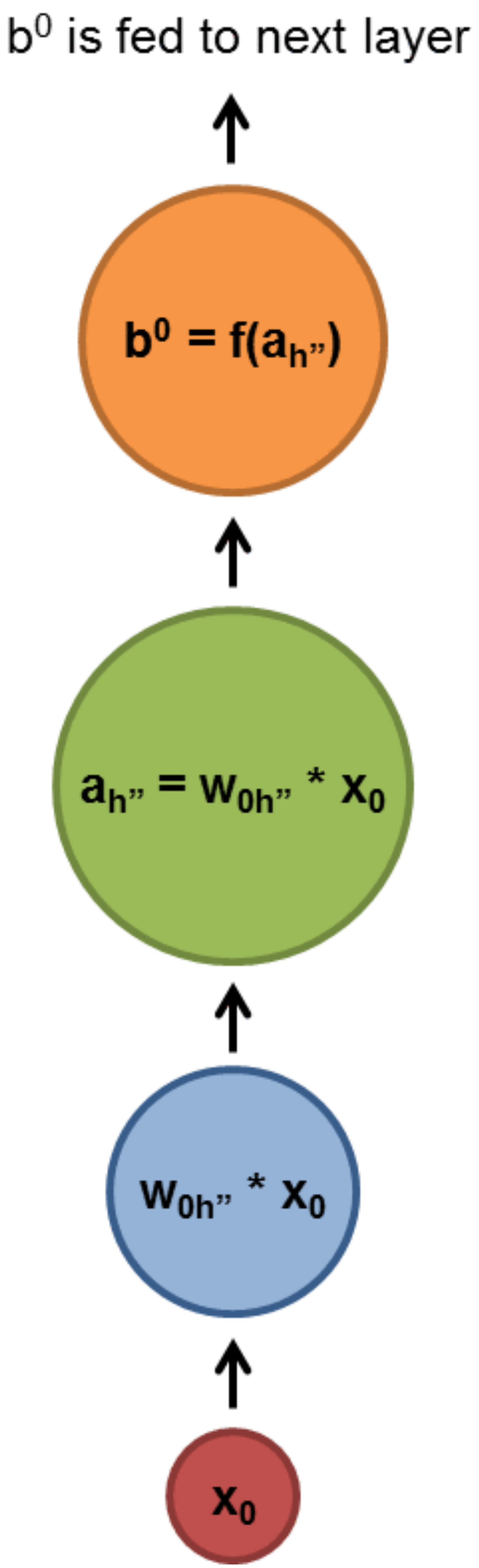
Browser Demos

- http://cs.stanford.edu/people/karpathy/convnetjs/

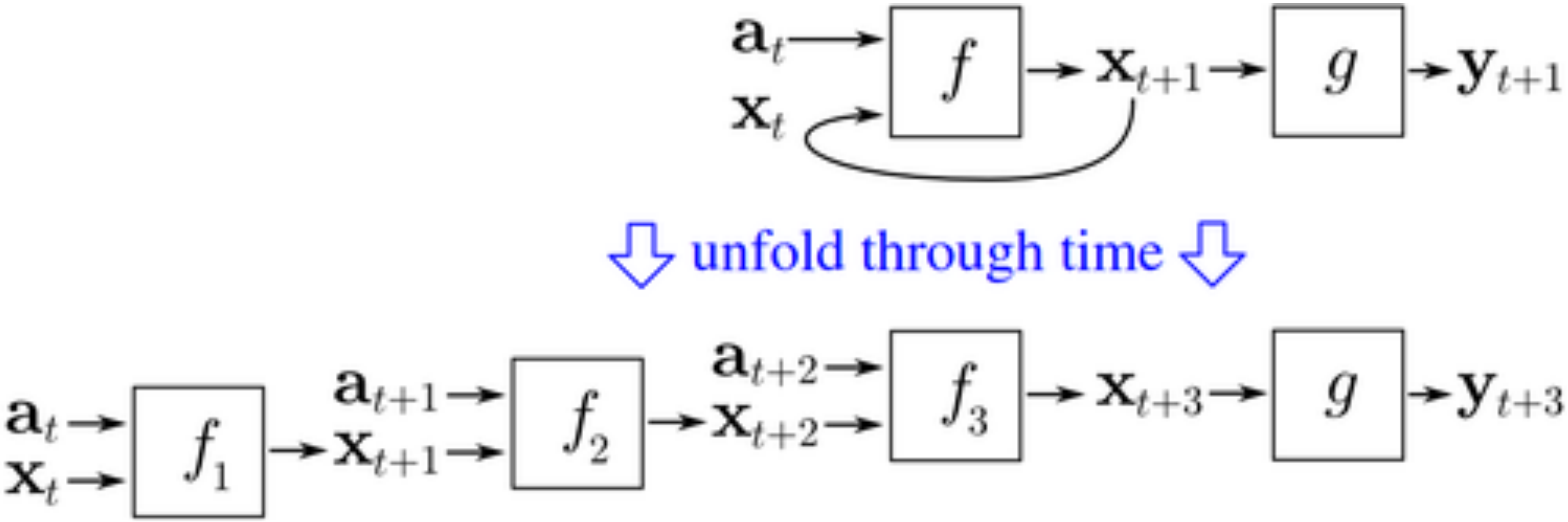
Recurrent Neural Networks



https://deeplearning4j.org/lstm.html

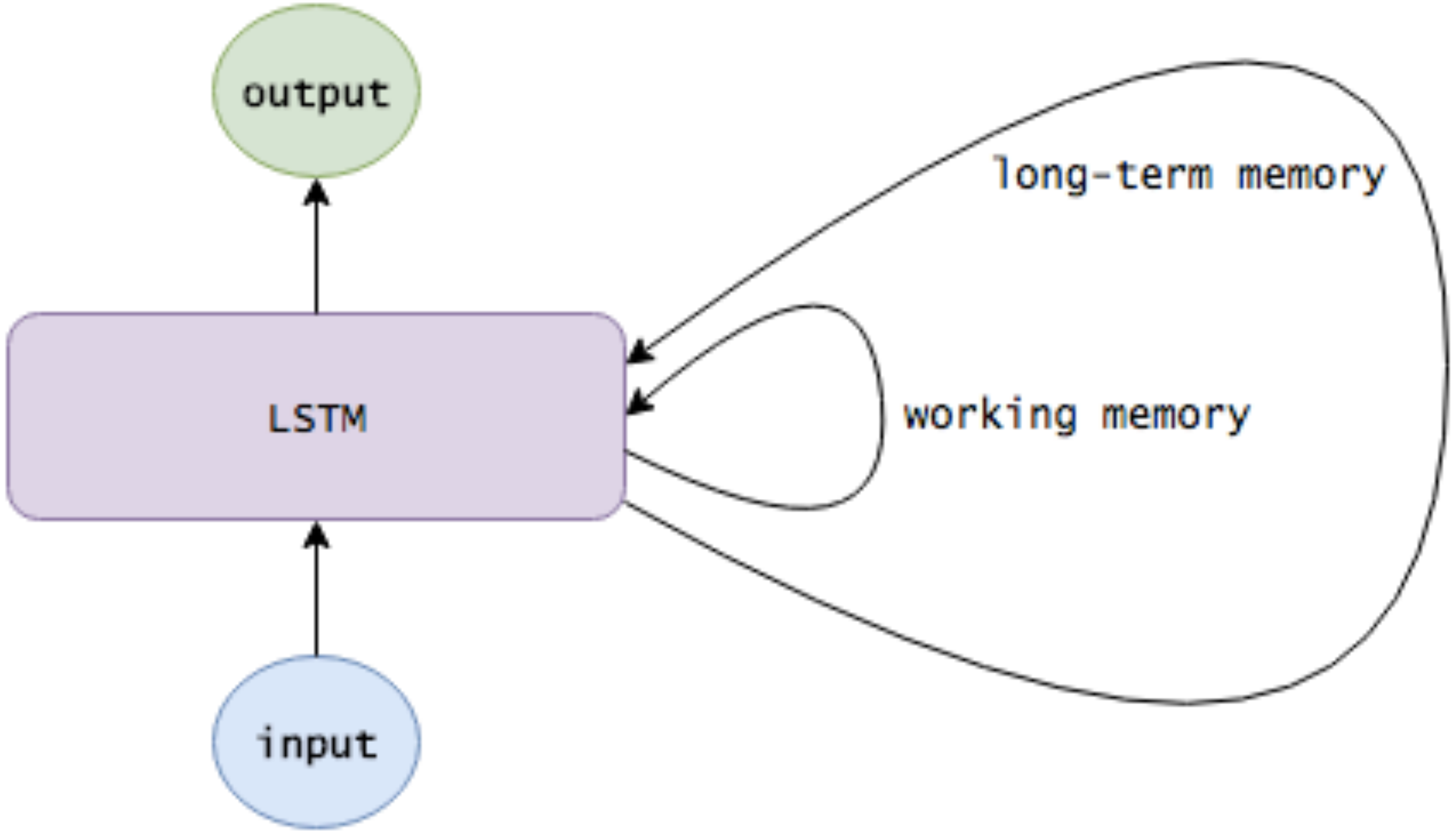

b^0 is fed to next layer
$b^0 = f(a_{h''})$
$a_{h''} = w_{0h''} * x_0$
$w_{0h''} * x_0$
x_0

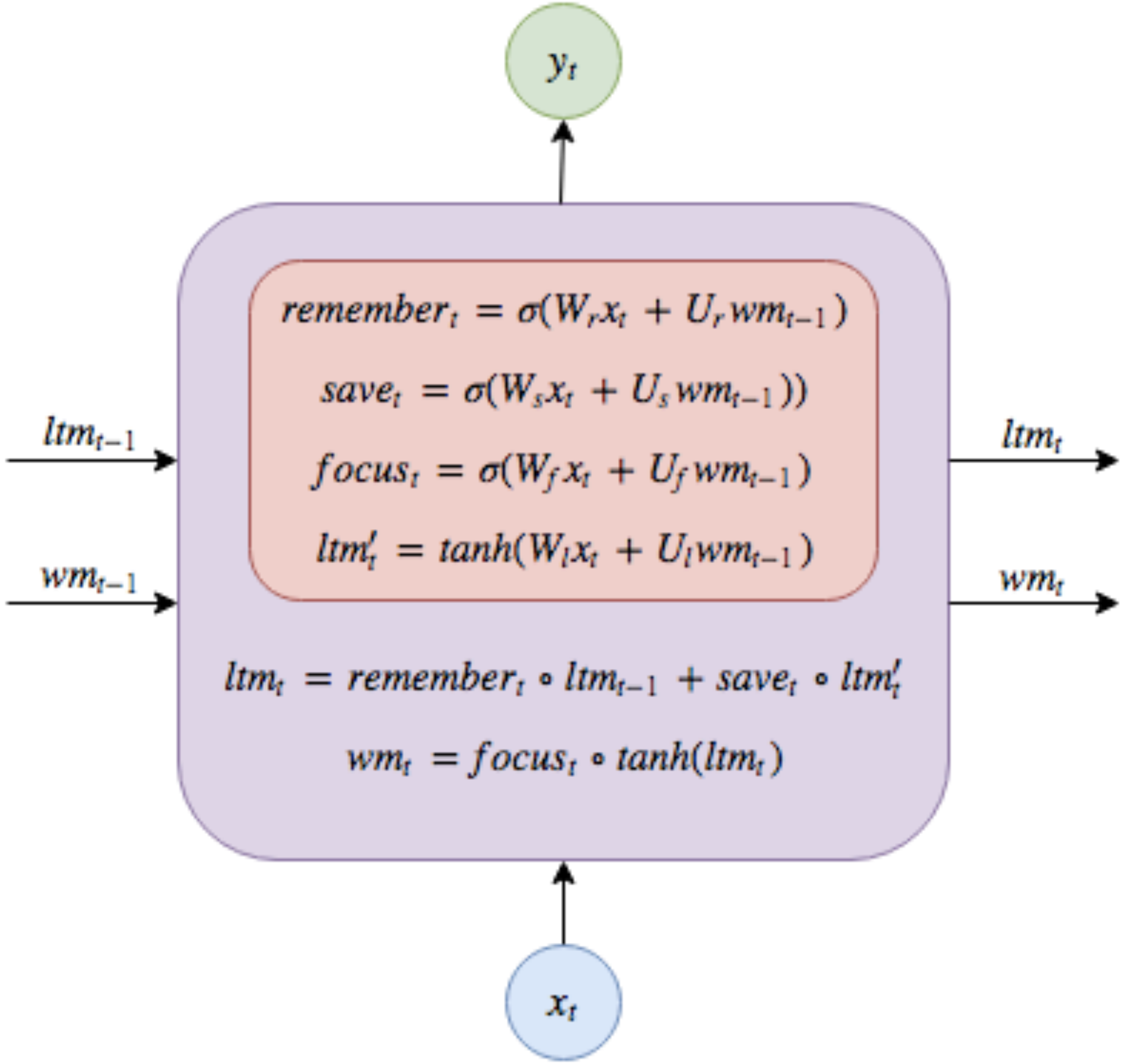
Back Propagation Through Time



https://en.wikipedia.org/wiki/Backpropagation_through_time

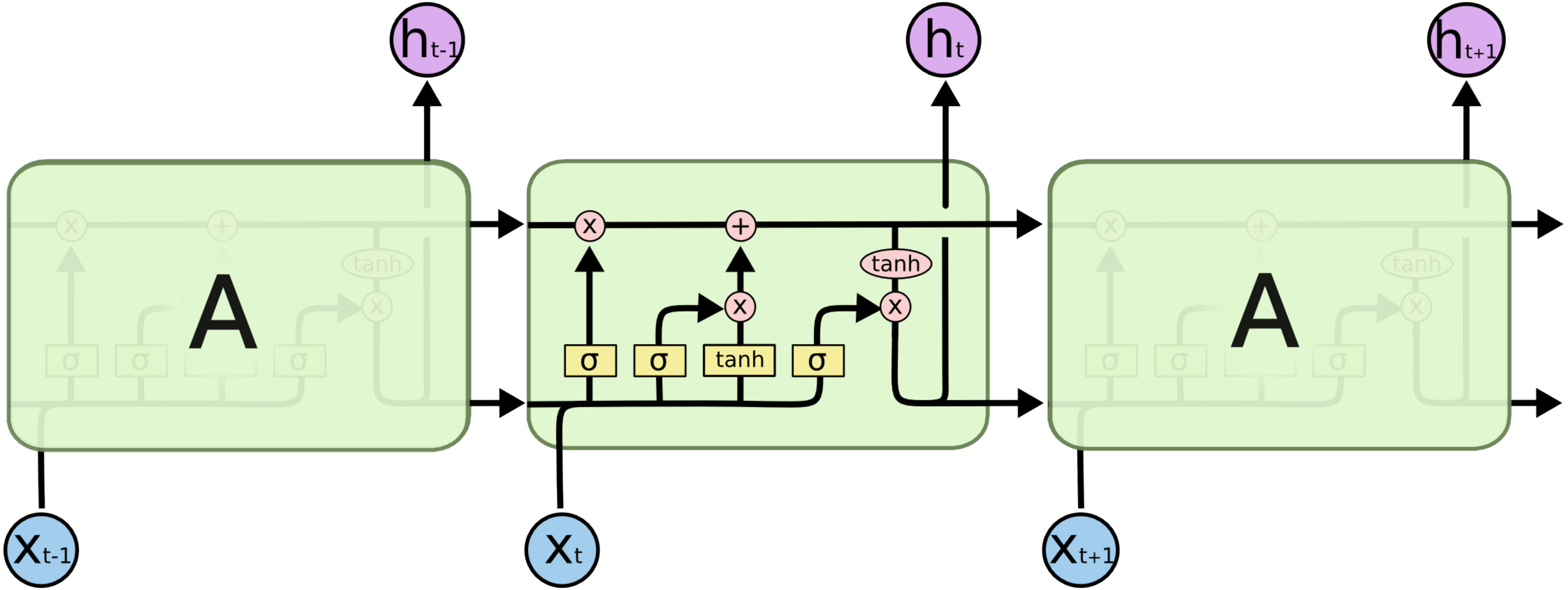
Long Short Term Memory





http://blog.echen.me/2017/05/30/exploring-lstms/

Another LSTM Diagram



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN Demo

http://cs.stanford.edu/people/karpathy/recurrentjs/