Artificial Neural Network

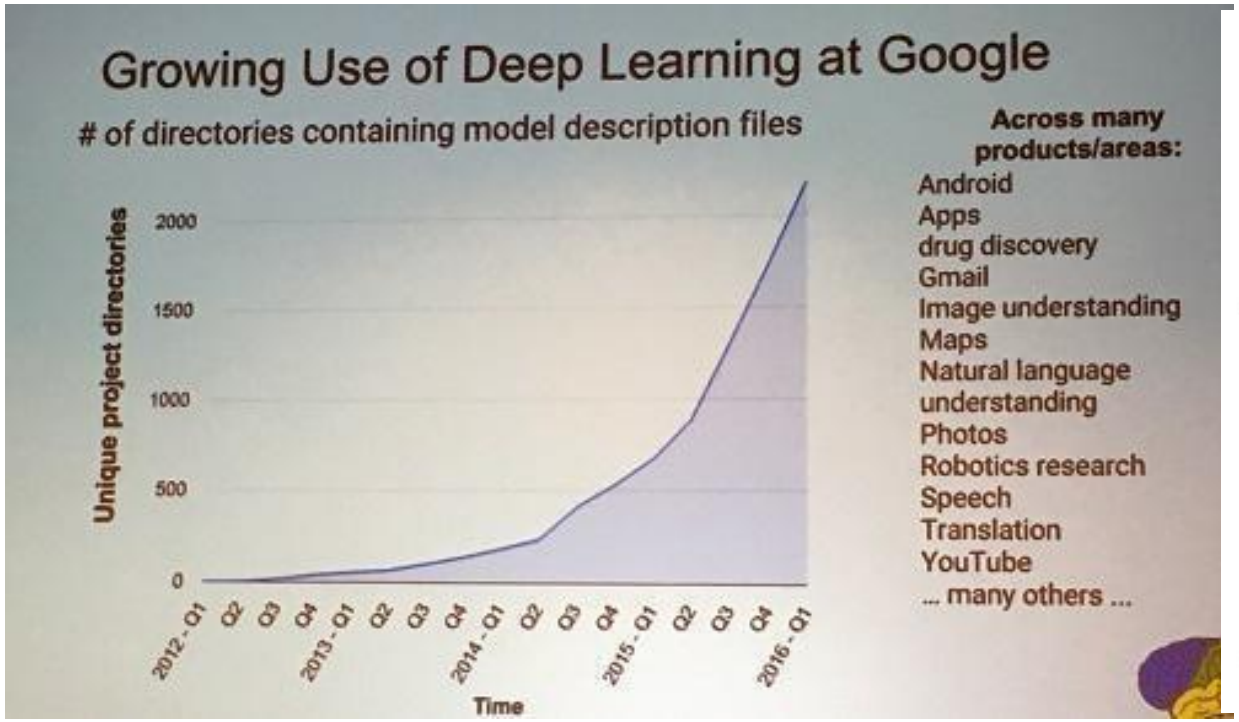
Simon Shim, Ph.D.
Professor
San Jose State University

Neural Networks

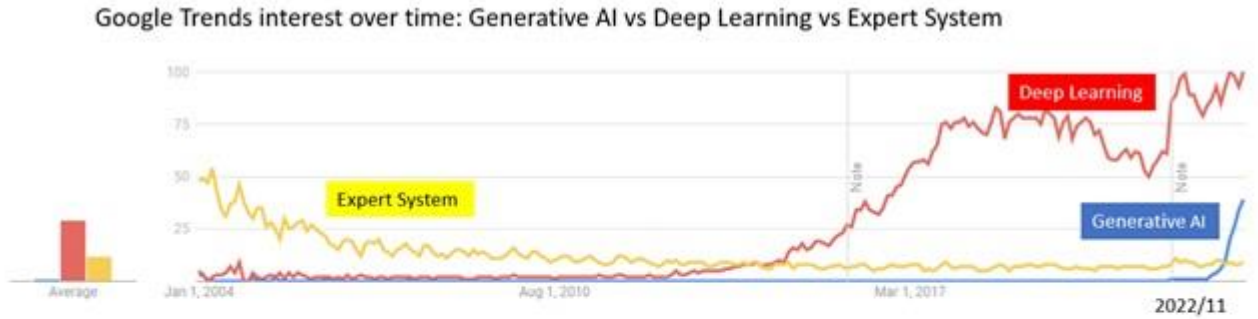
Origins: Algorithms that try to mimic the brain.
Was very widely used in 80s and early 90s; popularity diminished in late 90s.
Recent resurgence: State-of-the-art technique for many applications

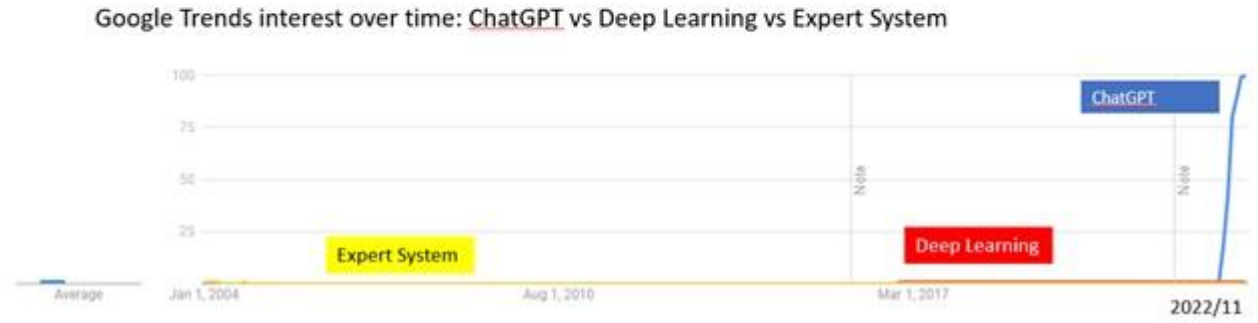
Deep learning attracts lots of attention.

- I believe you have seen lots of exciting results before.



Deep learning trends at Google. Source: SIGMOD/Jeff Dean

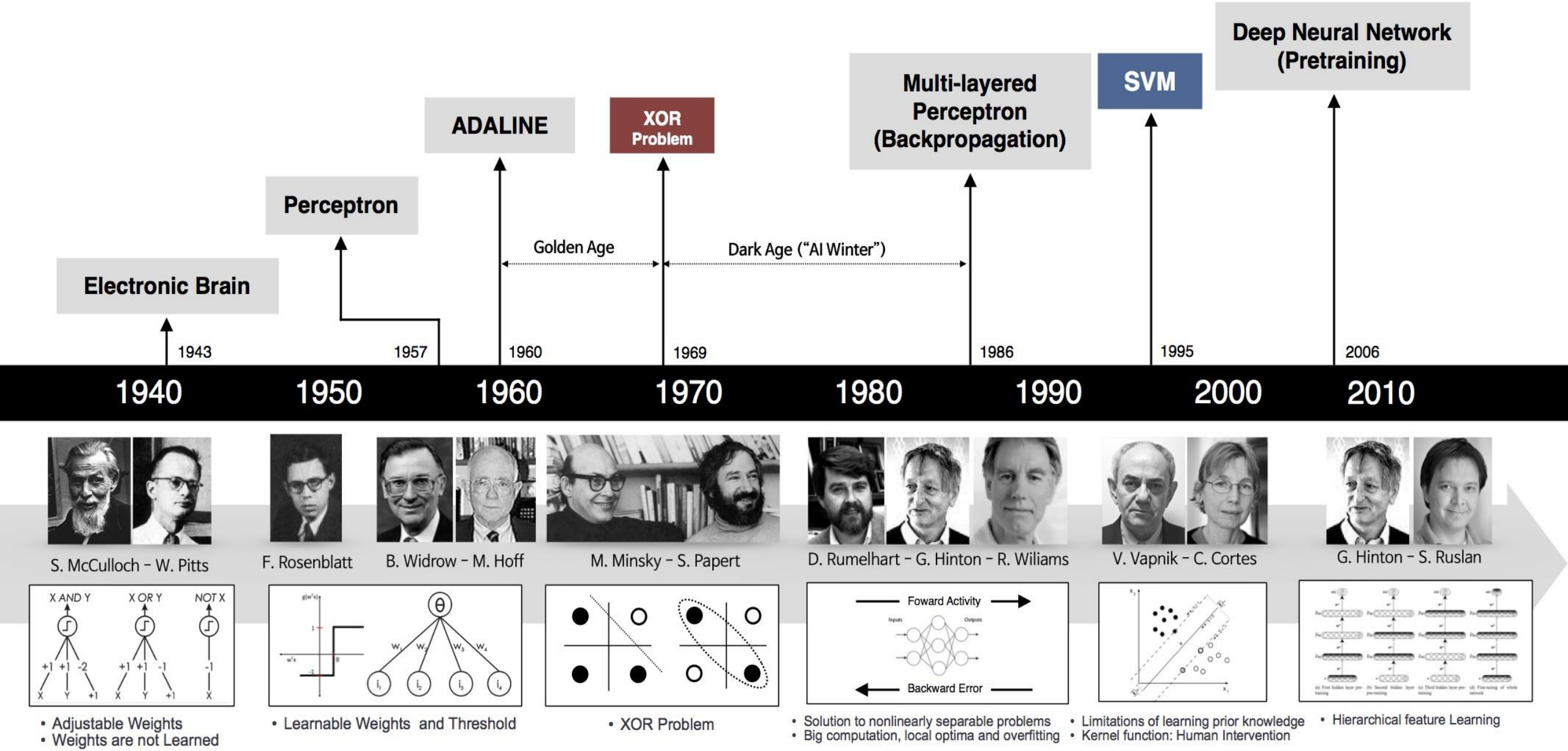




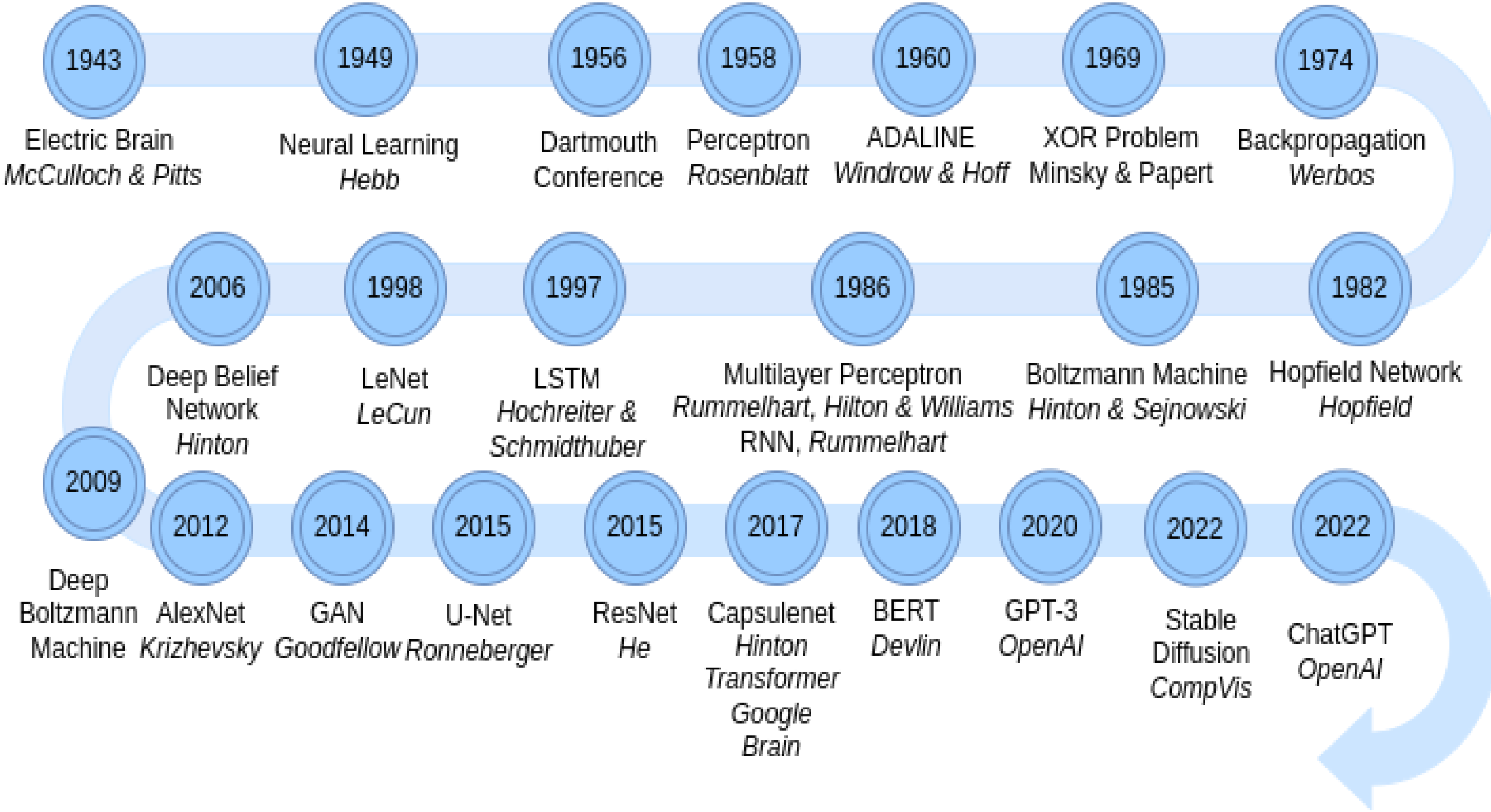
Source: Google Trends, 2023/05/11, compiled by DIGITIMES Asia

History of Deep Learning

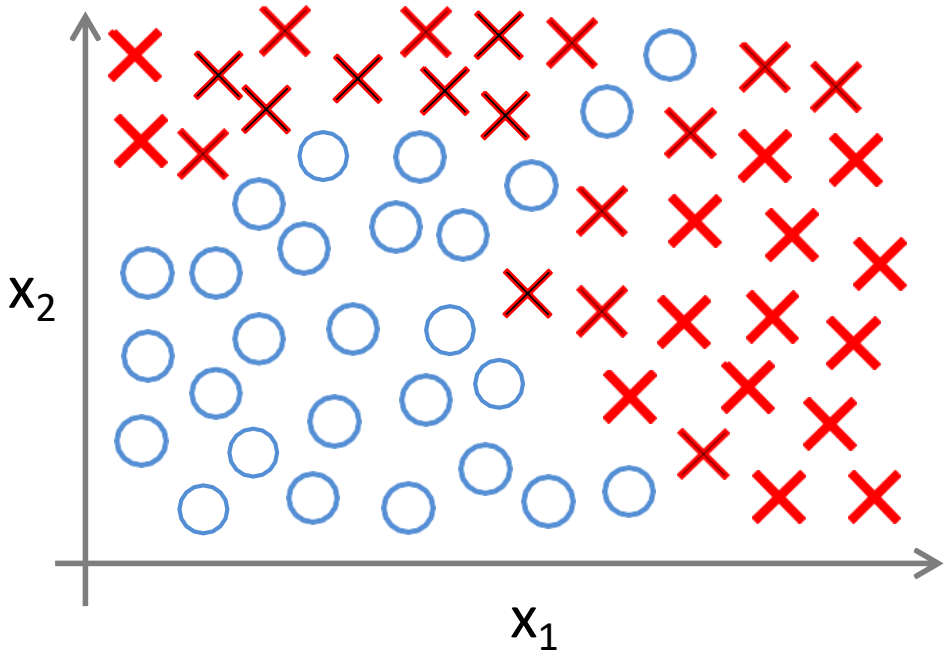
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
  - Do not have significant difference from DNN today
- 1986: Backpropagation
  - Usually more than 3 hidden layers is not helpful
- 1989: 1 hidden layer is “good enough”, why deep?
- 2006: RBM initialization (breakthrough)
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC image competition



http://beamlab.org/deeplearning/2017/02/23/deep_learning_101_part1.html


1943
Electric Brain
McCulloch & Pitts
1949
Neural Learning
Hebb
1956
Dartmouth
Conference
1958
Perceptron
Rosenblatt
1960
ADALINE
Windrow & Hoff
1969
XOR Problem
Minsky & Papert
1974
Backpropagation
Werbos
1982
Hopfield Network
Hopfield
1985
Boltzmann Machine
Hinton & Sejnowski
1986
Multilayer Perceptron
Rummelhart, Hilton & Williams
RNN, Rummelhart
1997
LSTM
Hochreiter &
Schmidthuber
1998
LeNet
LeCun
2006
Deep Belief
Network
Hinton
2009
Deep
Boltzmann
Machine
2012
AlexNet
Krizhevsky
2014
GAN
Goodfellow
2015
U-Net
Ronneberger
2015
ResNet
He
2017
Capsulenet
Hinton
Transformer
Google
Brain
2018
BERT
Devlin
2020
GPT-3
OpenAI
2022
Stable
Diffusion
CompVis
2022
ChatGPT
OpenAI

Non-linear Classification

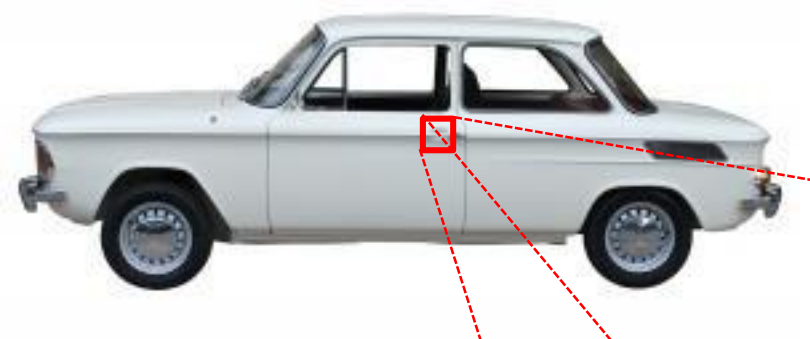


$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

$x_1 =$ size
$x_2 =$ # bedrooms
$x_3 =$ # floors
$x_4 =$ age
$\dots$
x_{100}

What is this?

You see this:



But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Computer Vision: Car detection





Cars



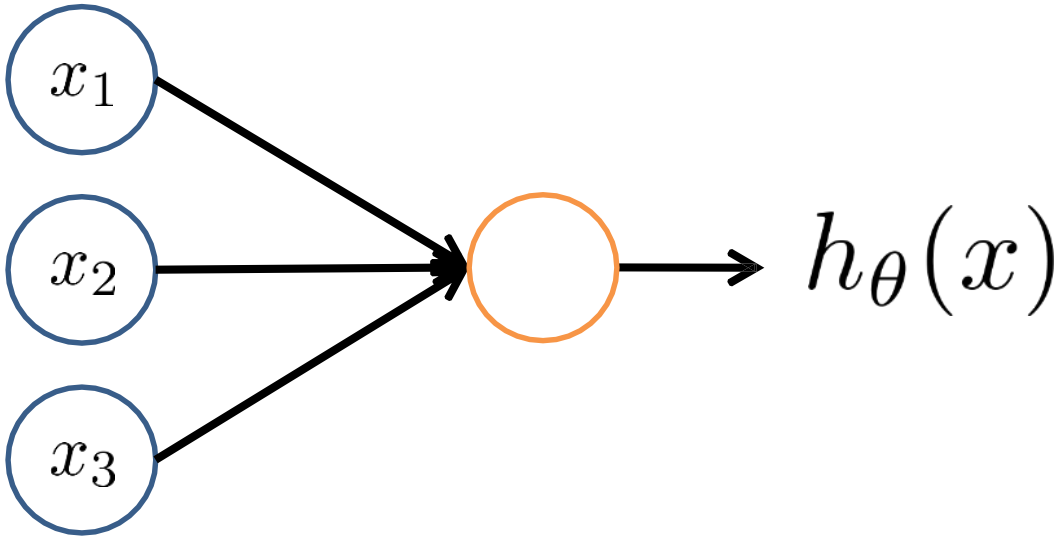



Not a car

Testing:

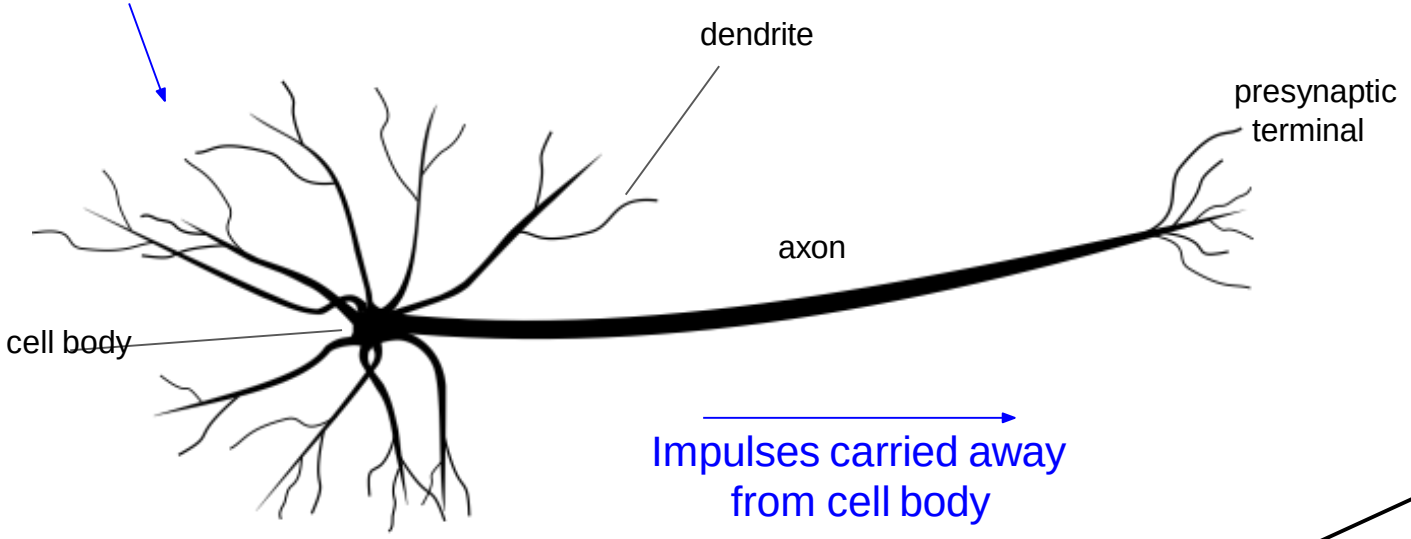
Neuron model: Logistic unit

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

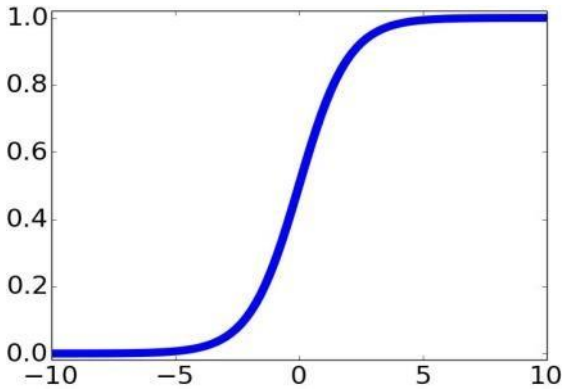


Sigmoid (logistic) activation function.

Impulses carried toward cell body

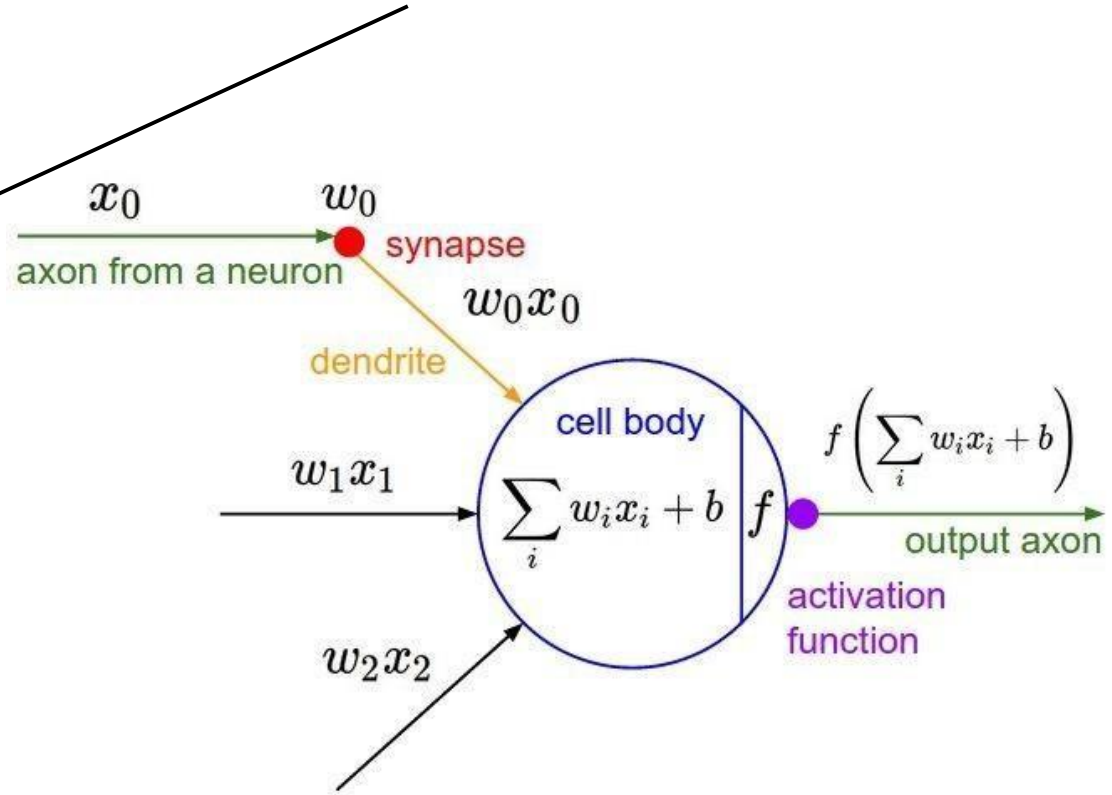


Impulses carried away from cell body



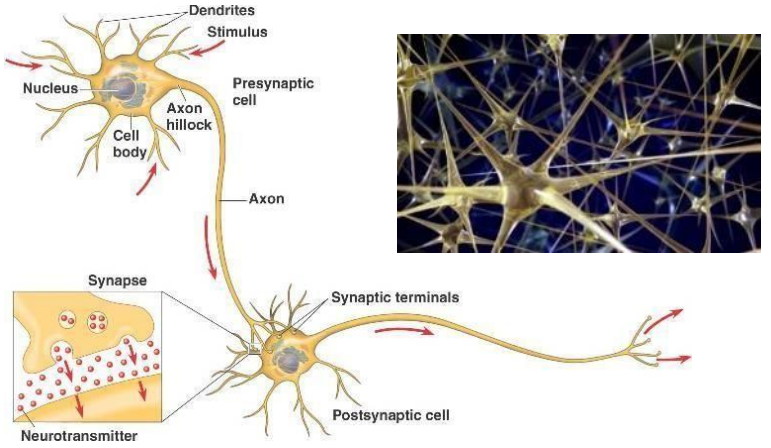
sigmoid activation function

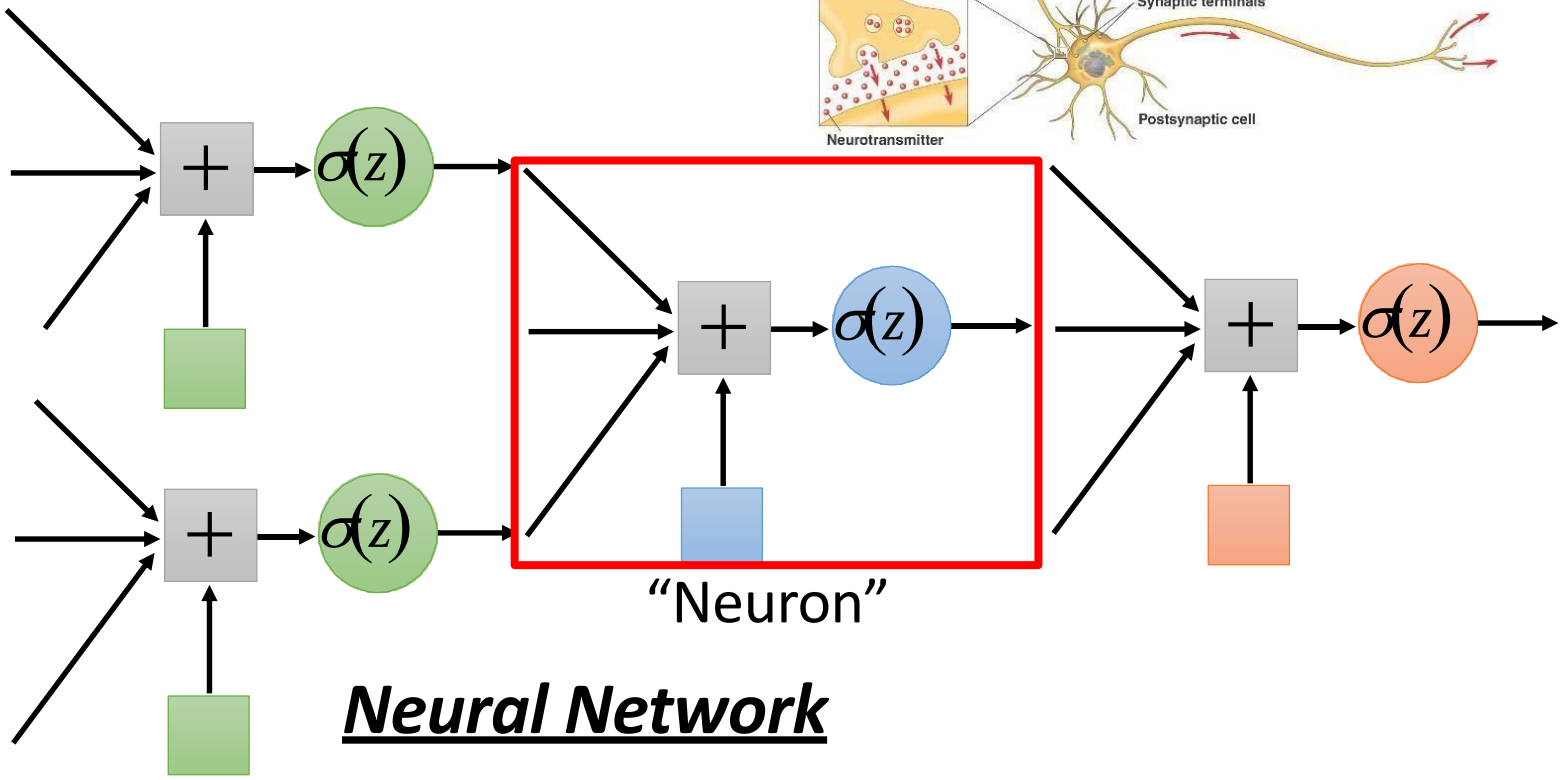
$$\frac{1}{1+e^{-x}}$$



Fei-Fei Li & Justin Johnson & Serena Yeung

Neural Network





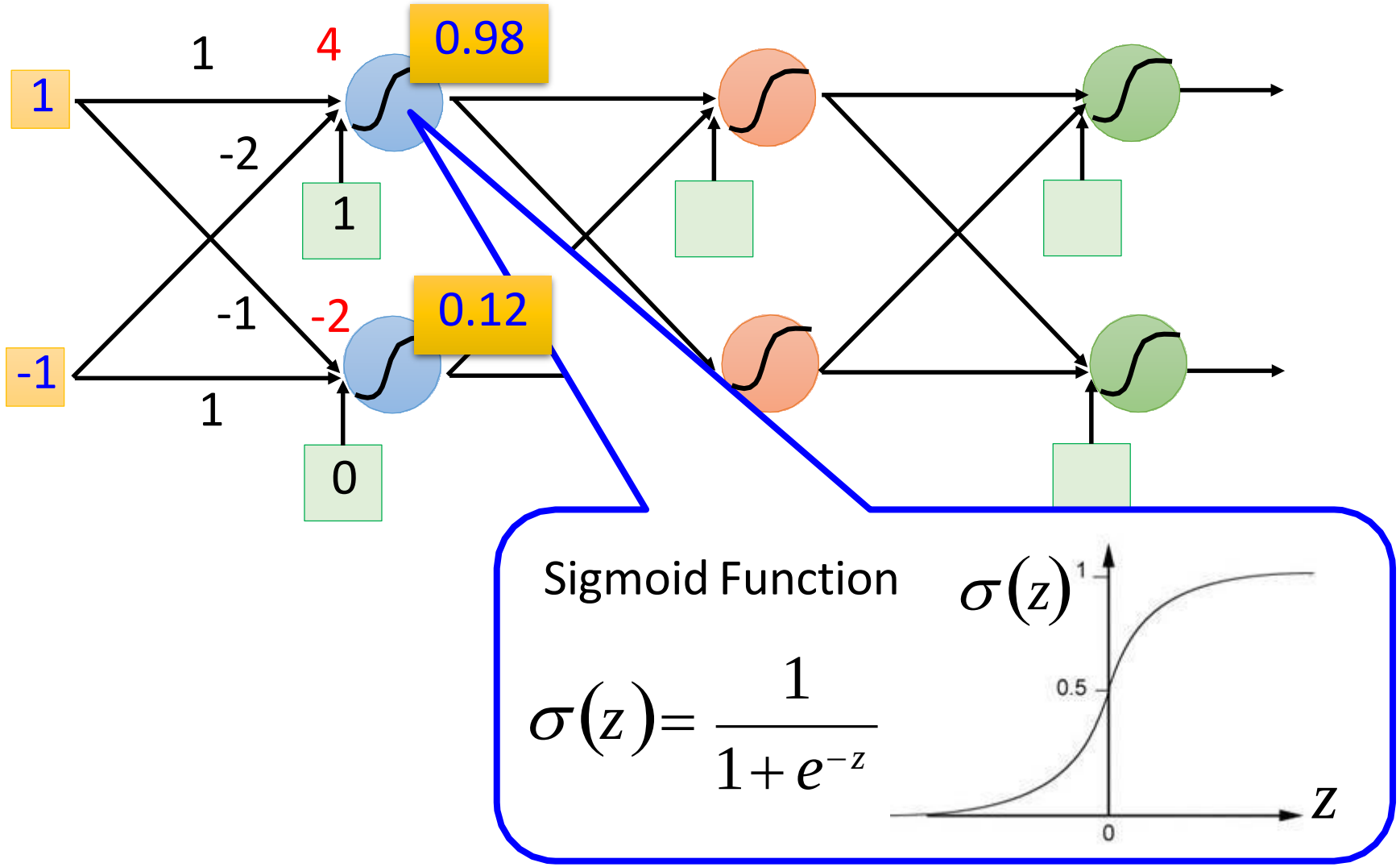
"Neuron"

Neural Network

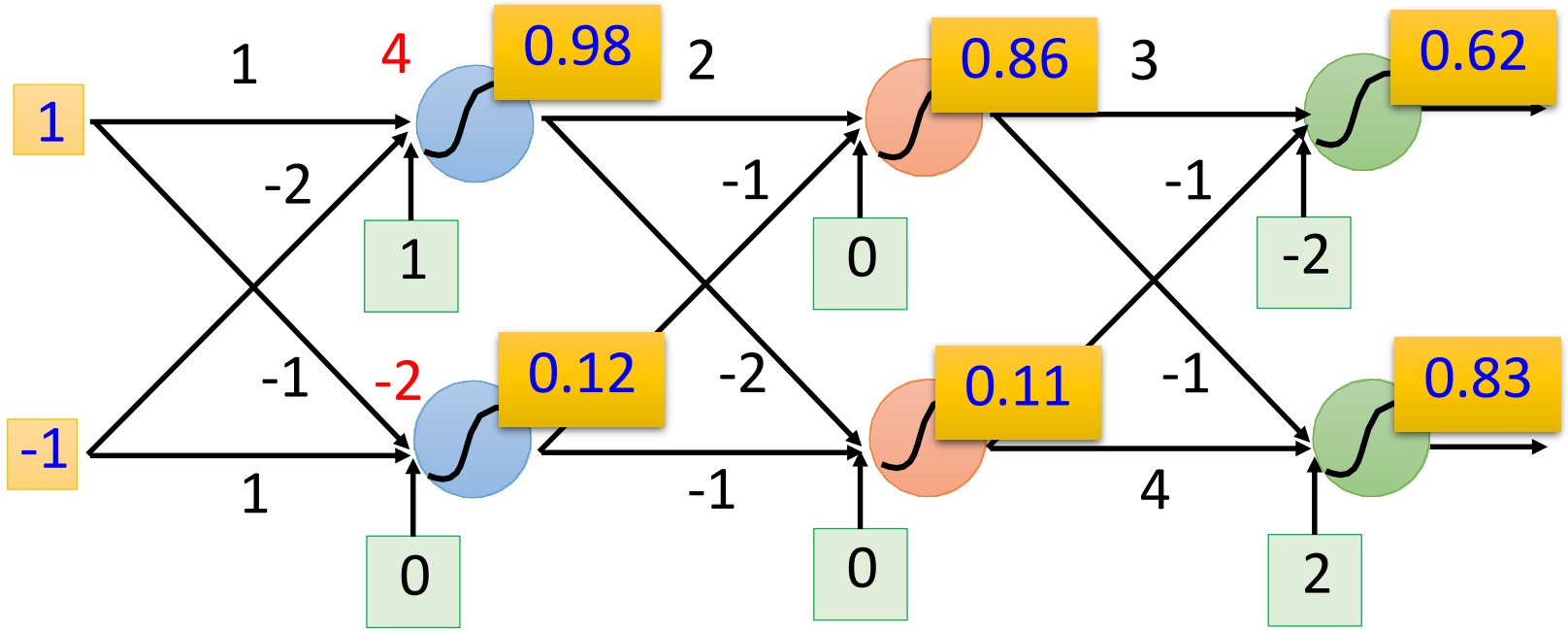
Different connection leads to different network structures

Network parameter θ: all the weights and biases in the "neurons"

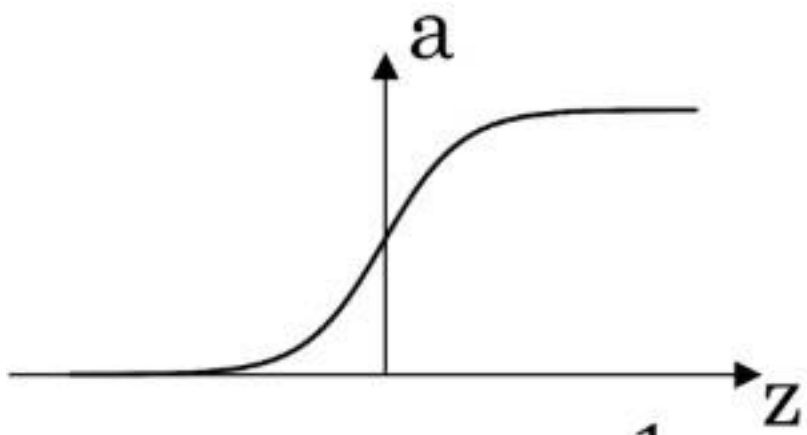
Fully Connect Feedforward Network



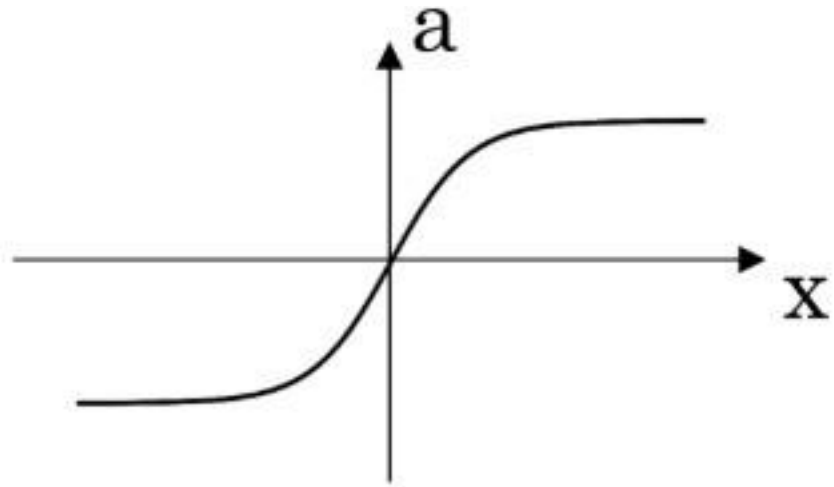
Fully Connect Feedforward Network



Activation Functions

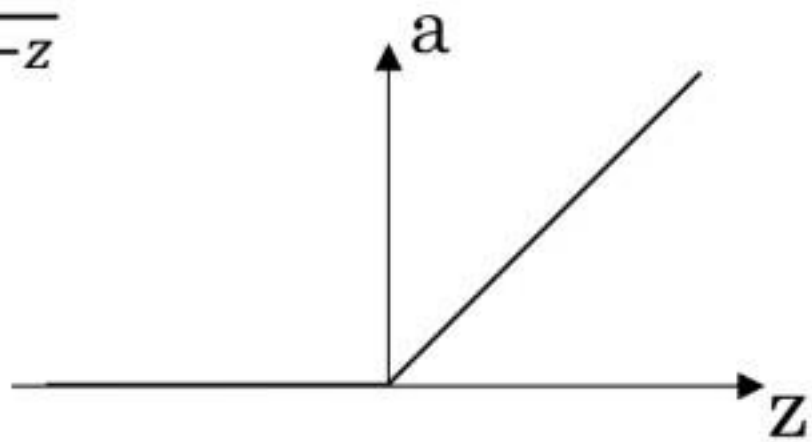


sigmoid: $a = \dfrac{1}{1 + e^{-z}}$



$g(z) = \tanh(z)$

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

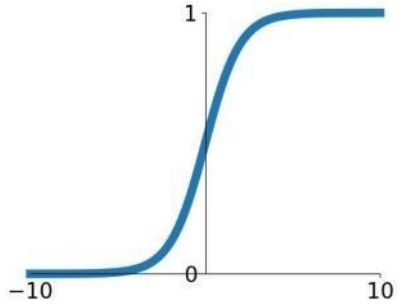


ReLU

Activation functions

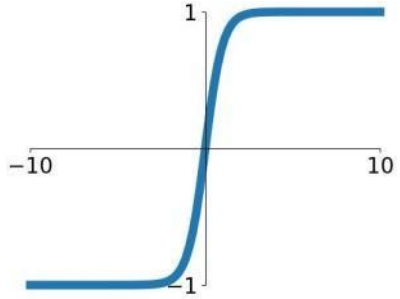
Sigmoid

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



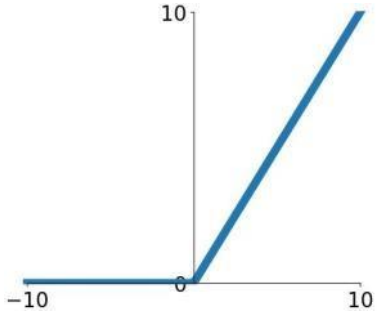
tanh

$$\tanh(x)$$



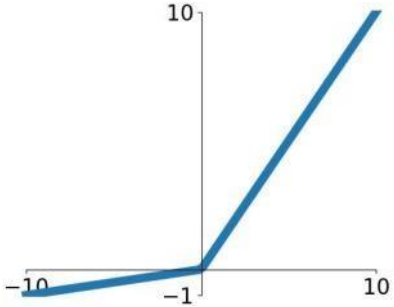
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

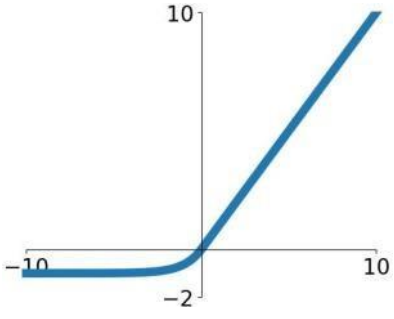


Maxout

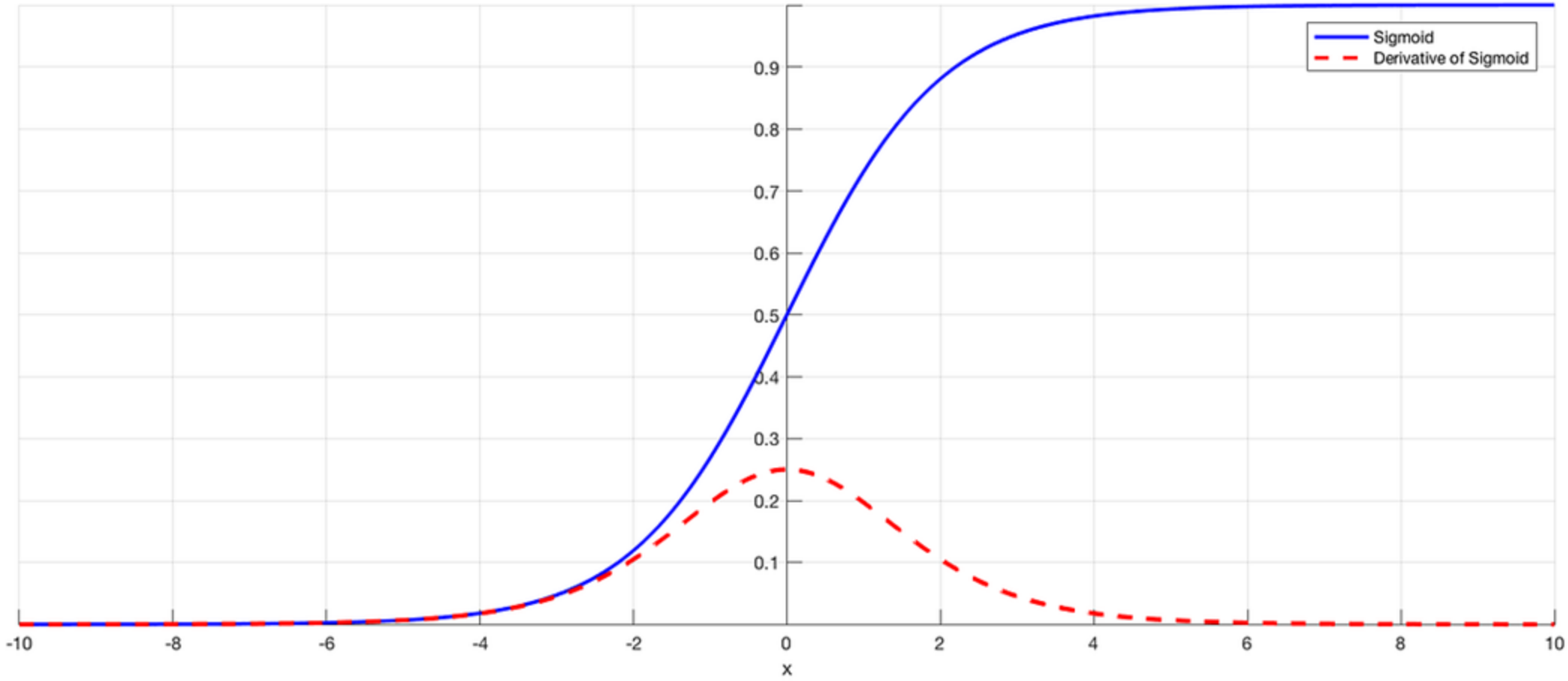
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

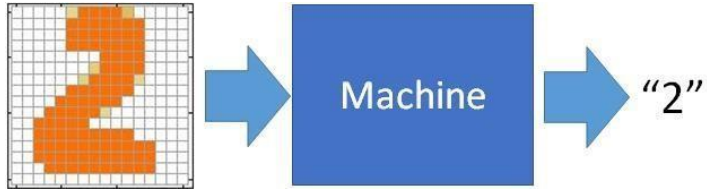


Sigmoid!

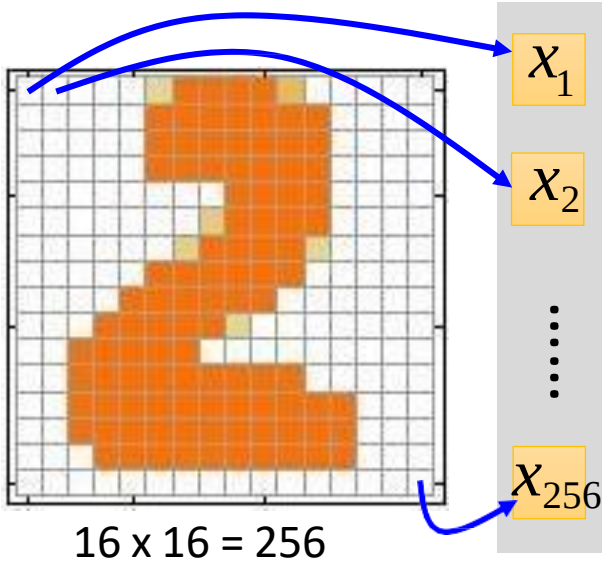


https://isaacchanghau.github.io/img/deeplearning/activationfunction/sigmoid.png

Example Application



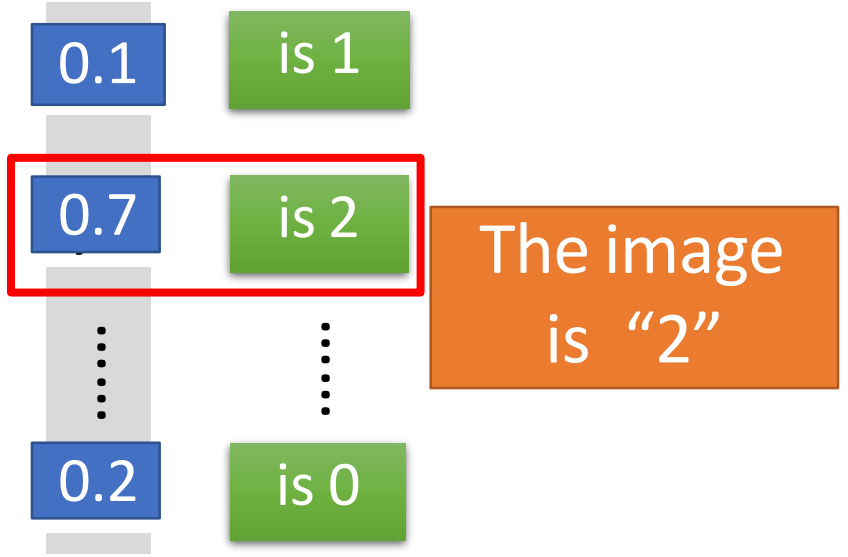
Input



Ink → 1

No ink → 0

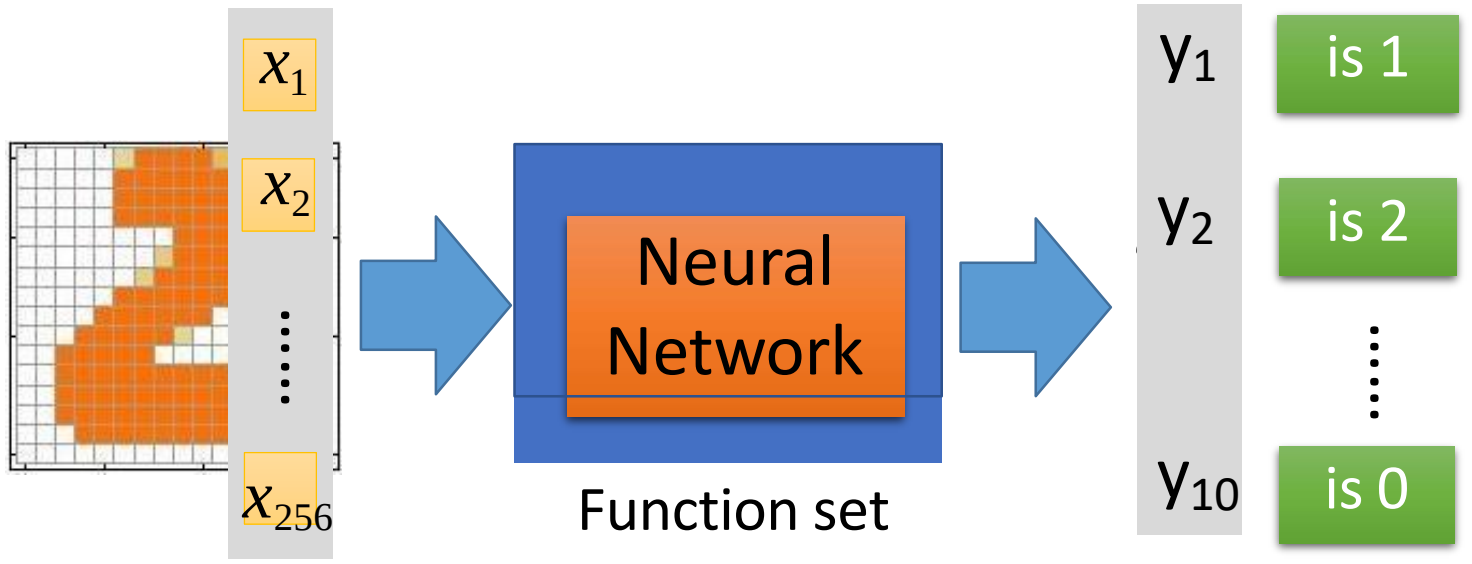
Output



Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

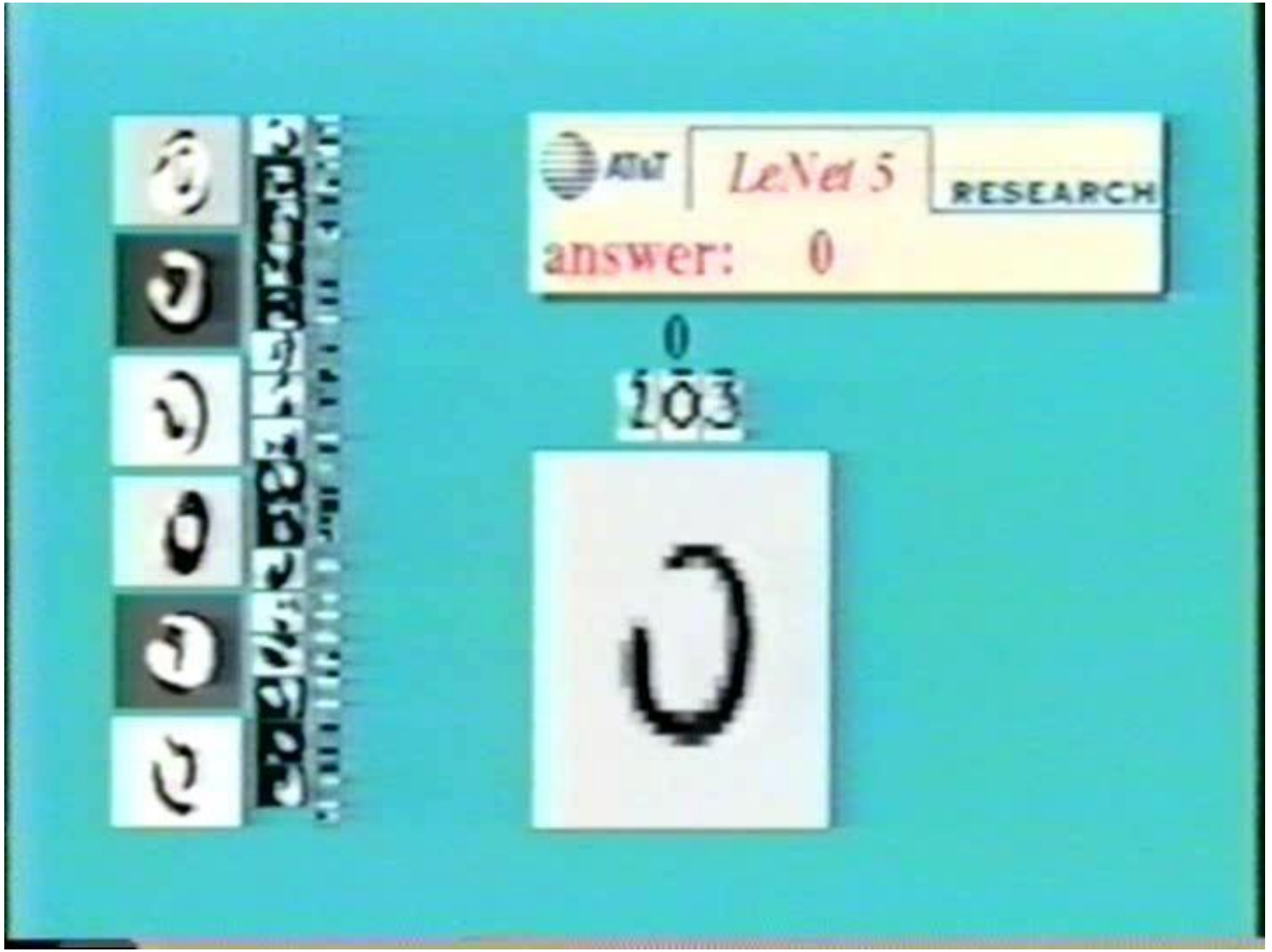


Input:
256-dim vector

output:
10-dim vector

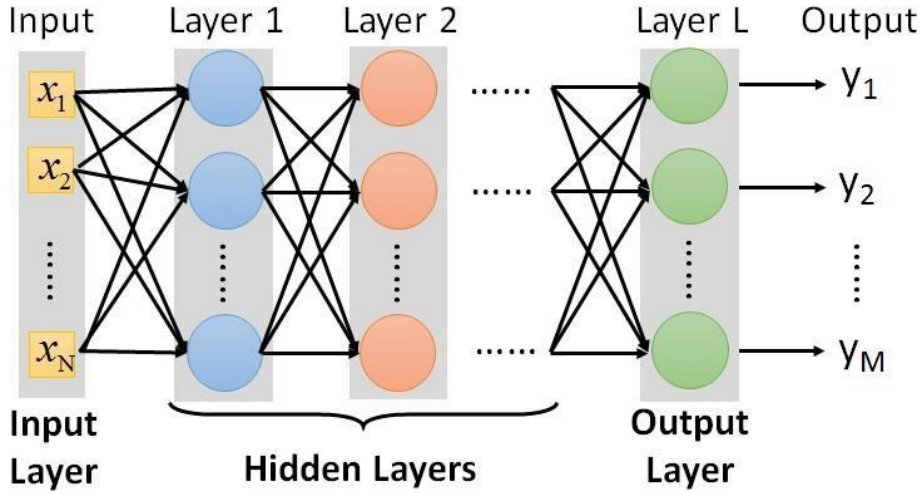
Handwritten digit classification

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



[Yann LeCun]

FAQ

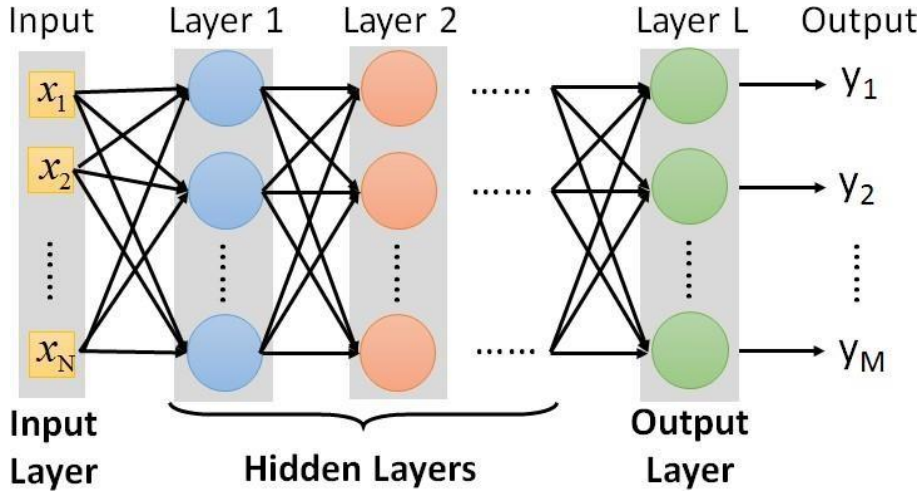


- Q: How many layers? How many neurons for each layer?

- Q: Can the structure be automatically determined?

- Q: Can we design the network structure?

FAQ



- Q: How many layers? How many neurons for each layer?

Trial and Error + Intuition

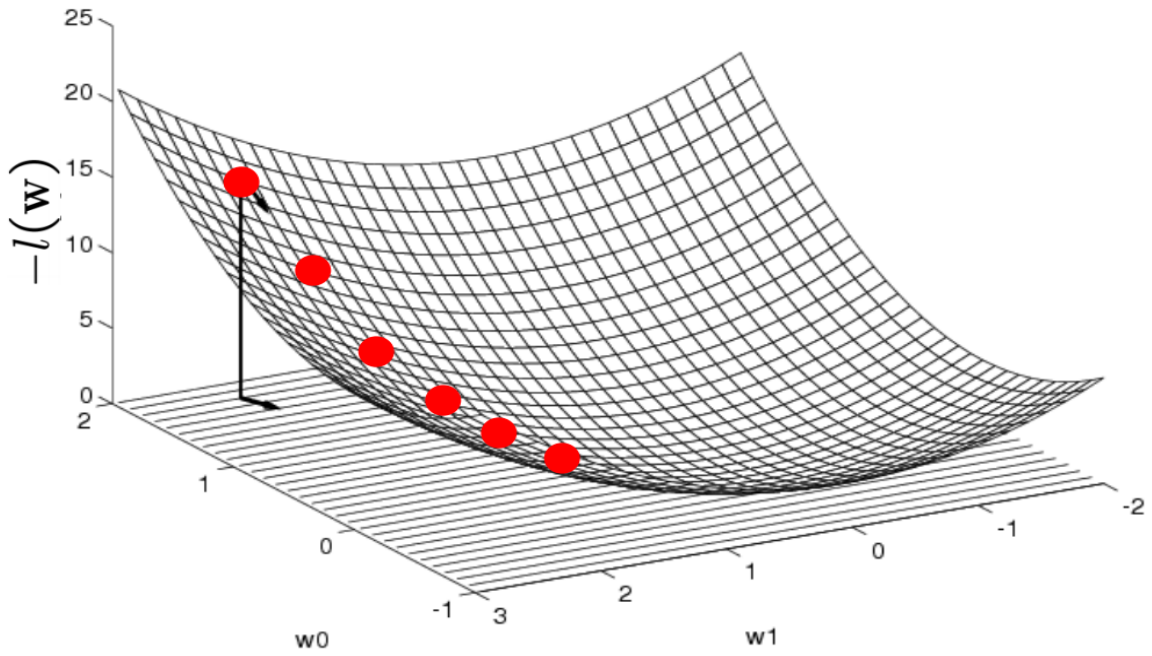
- Q: Can the structure be automatically determined?
  - E.g. Evolutionary Artificial Neural Networks
- Q: Can we design the network structure?

Convolutional Neural Network (CNN)

Optimizing concave/convex function

- Conditional likelihood for Logistic Regression is concave
- Maximum of a concave function = minimum of a convex function

Gradient Ascent (concave)/ Gradient Descent (convex)



Gradient:

$$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_d}]'$$

Update rule: **Learning rate, η>0**

$$\triangle \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

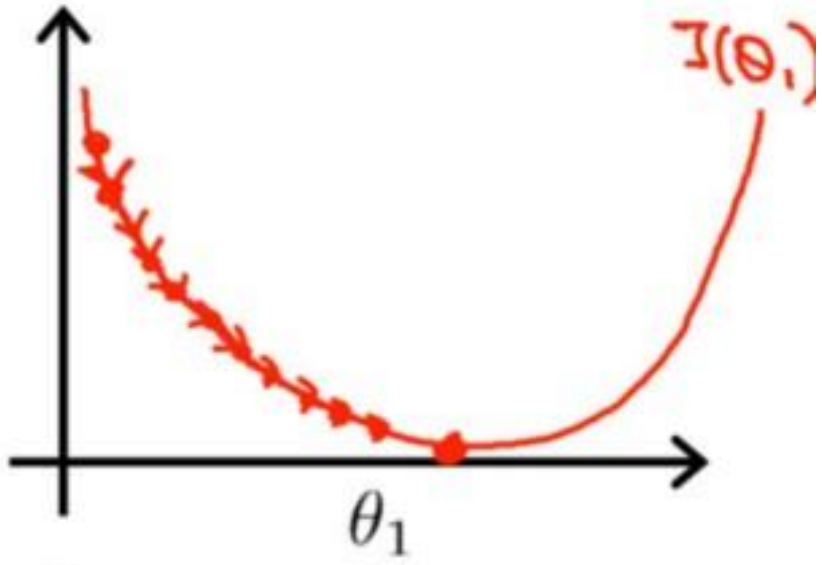
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left. \frac{\partial l(\mathbf{w})}{\partial w_i} \right|_t$$

Source: Aarti Singh

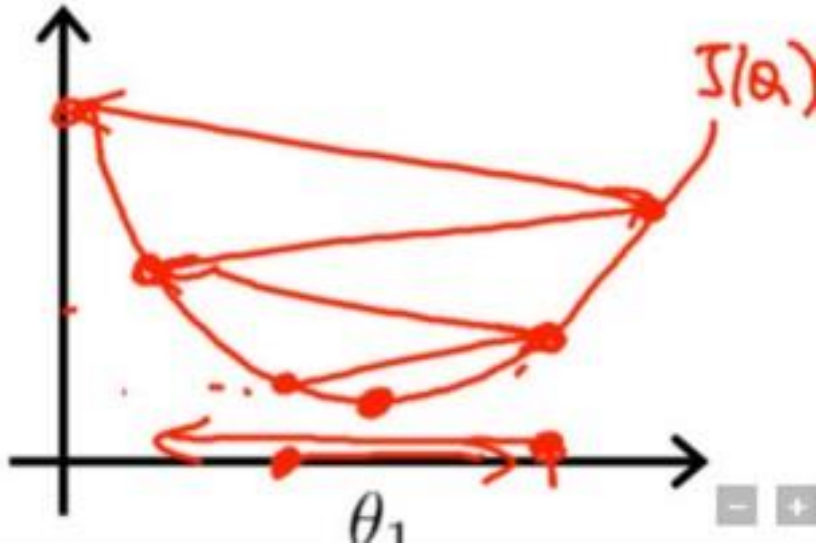
Learning Rate

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.

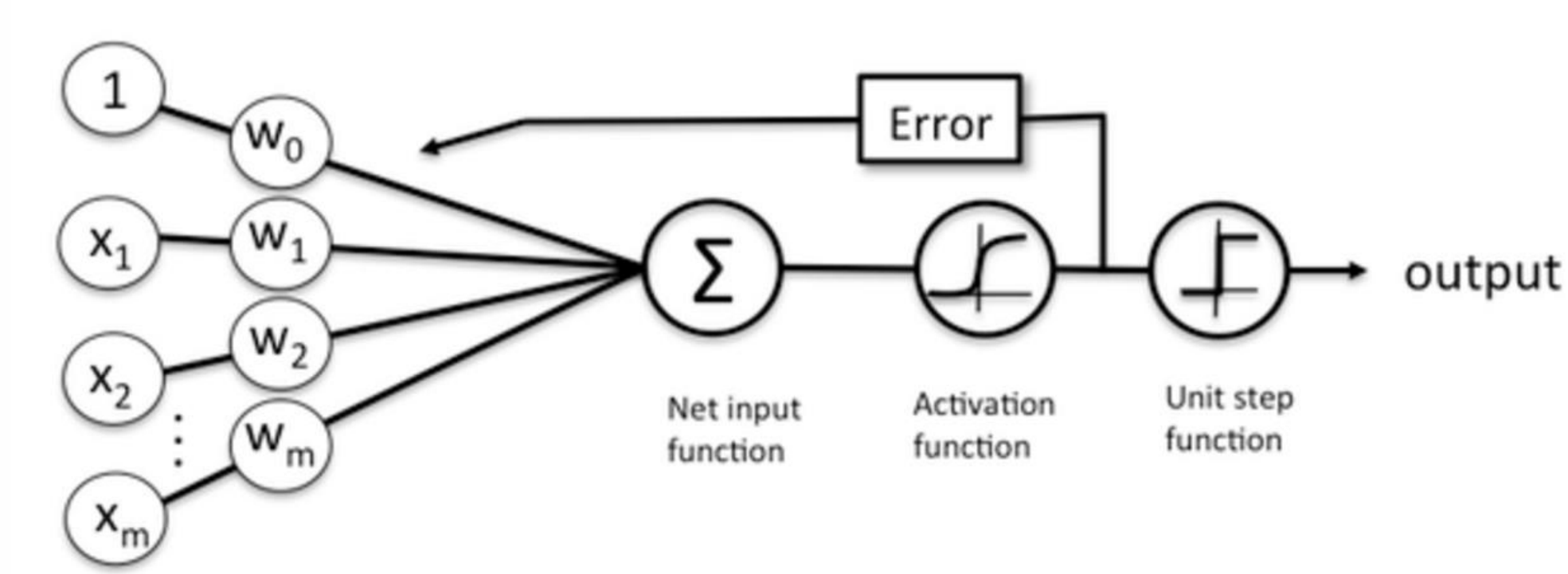


If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

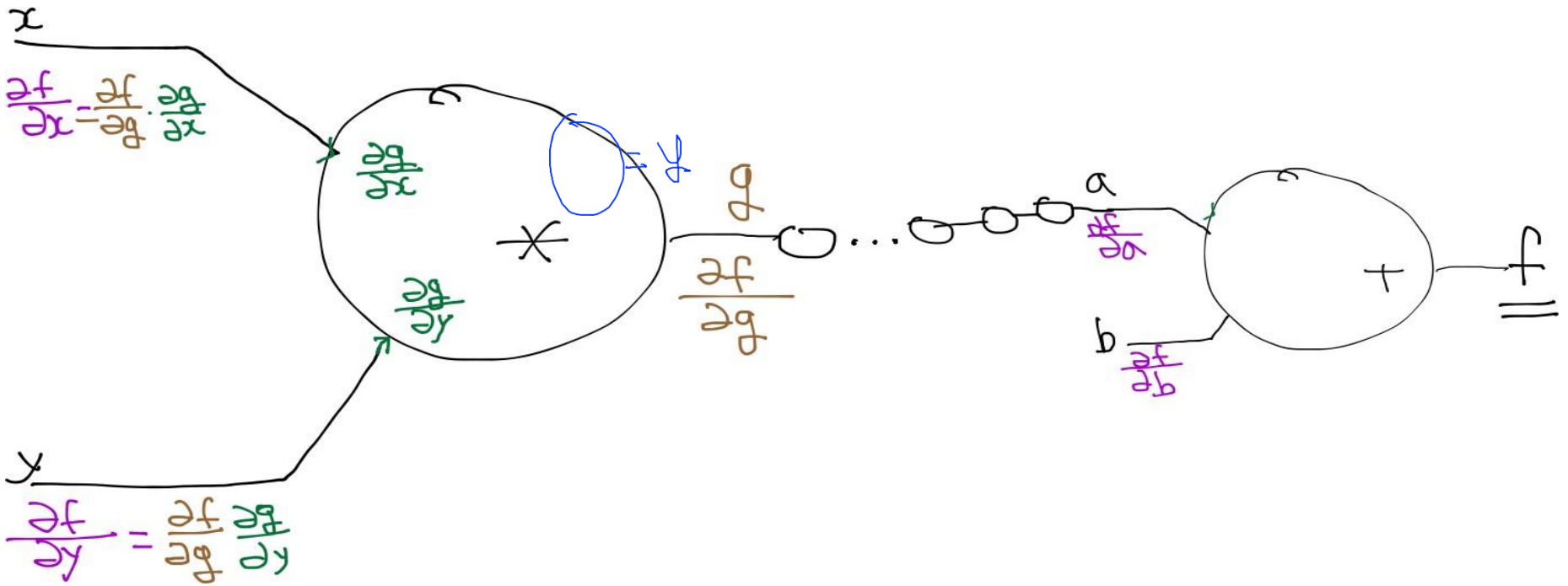


Source: Andrew Ng

Training Steps



Backpropagation (chain rule)



http://cs231n.stanford.edu/

chain rule

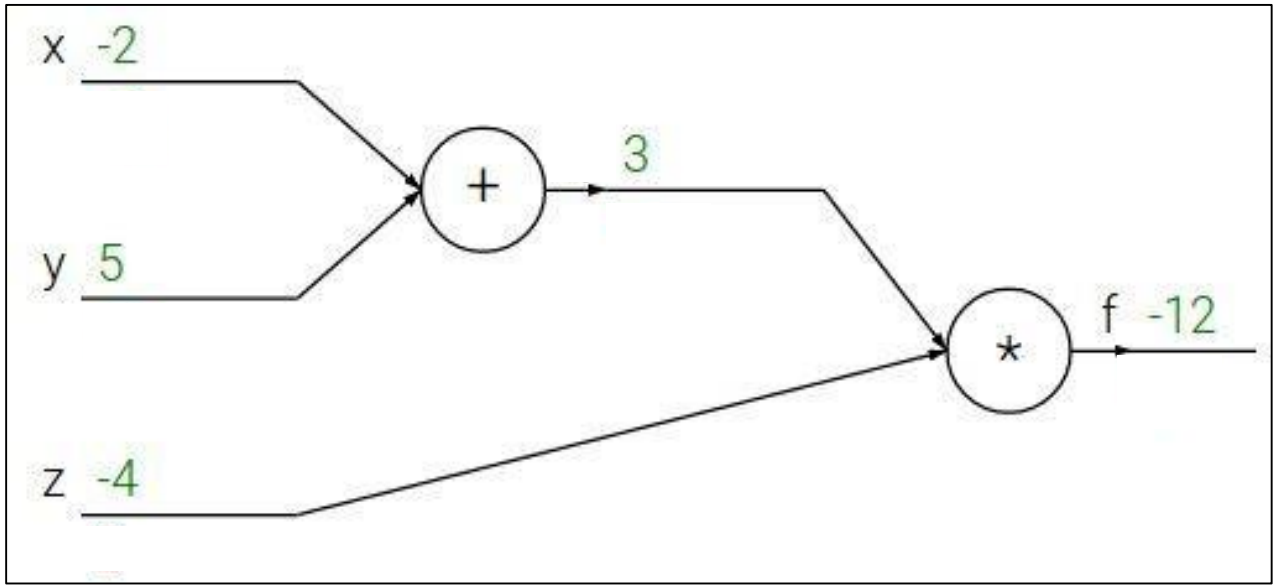
$$f_1\left(f_2\left(f_3\left(f_4(x)\right)\right)\right)$$

$$\frac{\partial f_1}{\partial x} = \frac{\partial f_1}{\partial f_2} \cdot \frac{\partial f_2}{\partial f_3} \cdot \frac{\partial f_3}{\partial f_4} \cdot \frac{\partial f_4}{\partial x}$$

Sam Abrahams

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



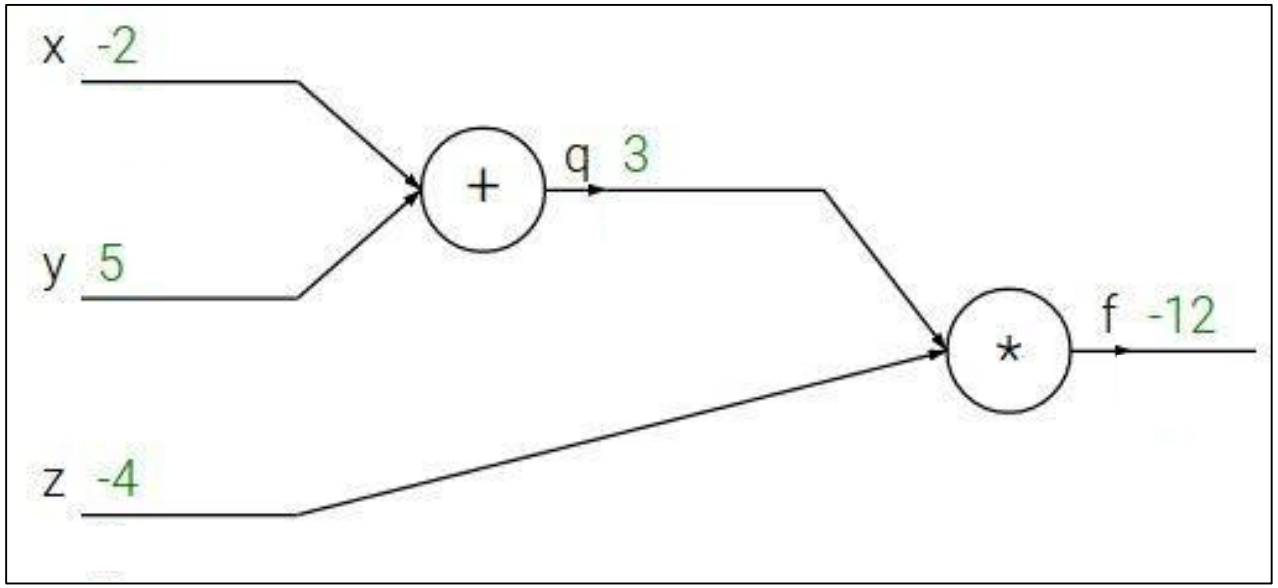
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



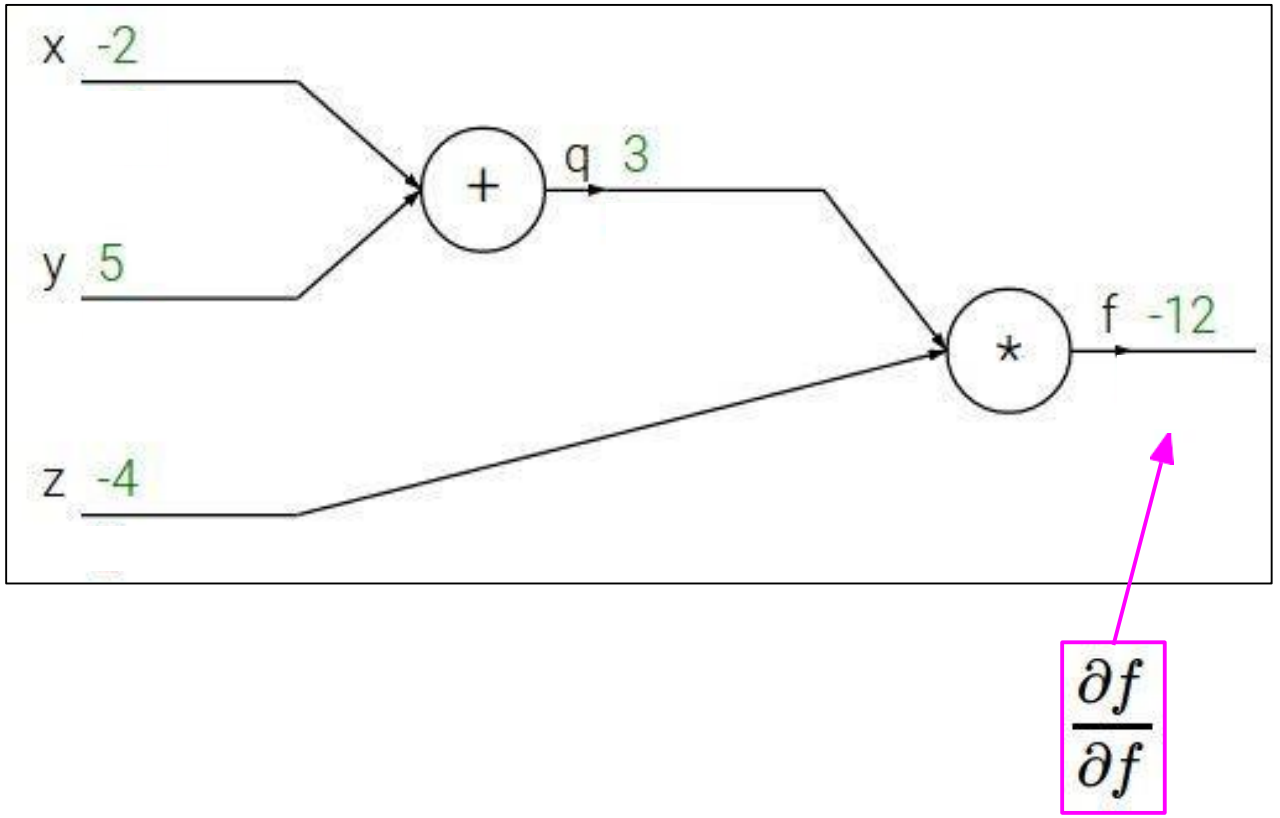
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



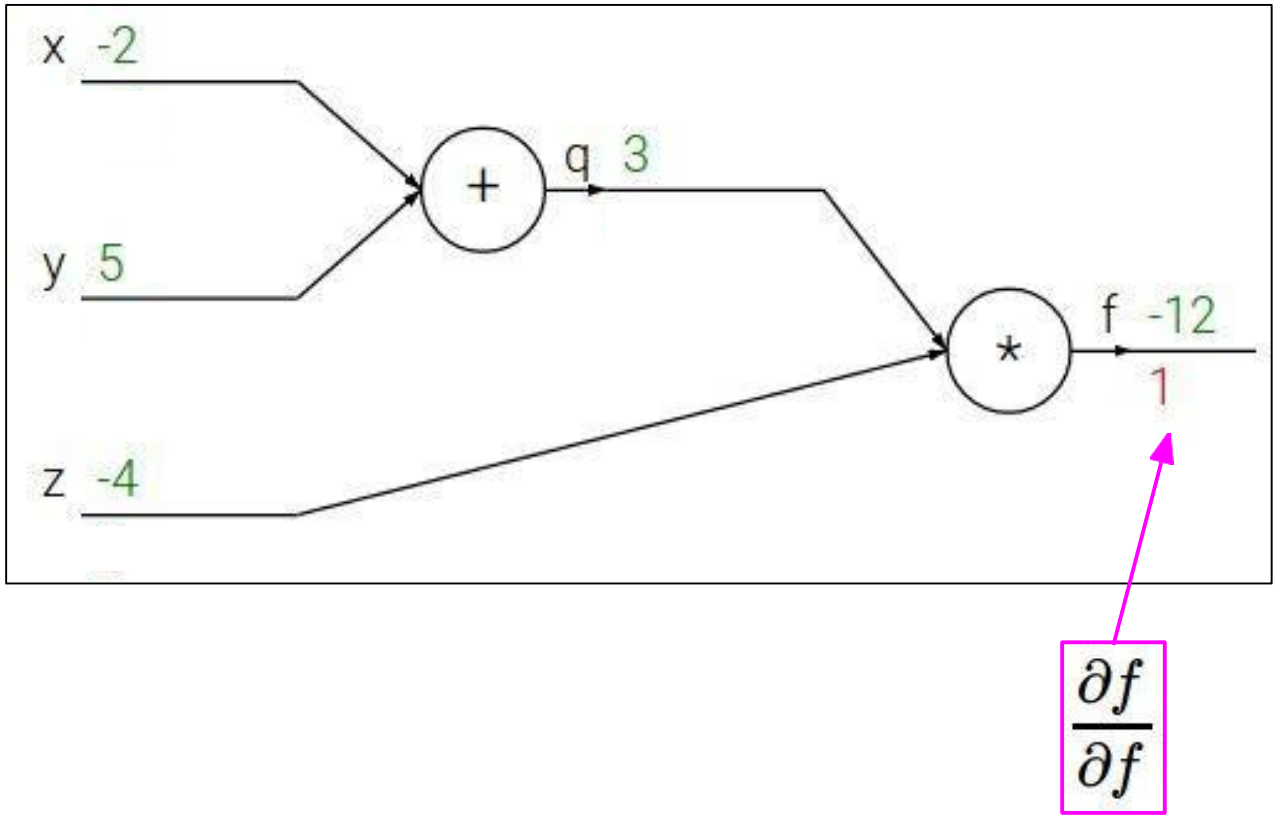
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

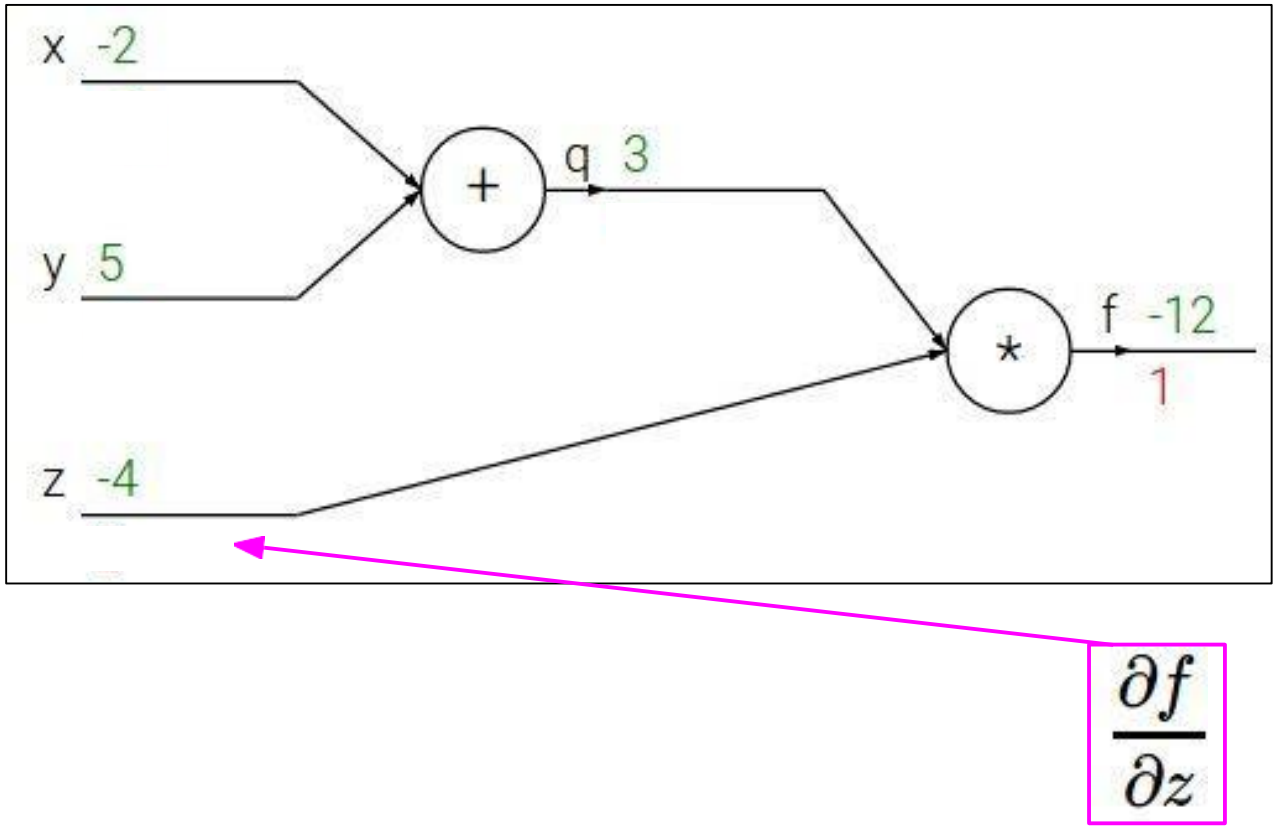


$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



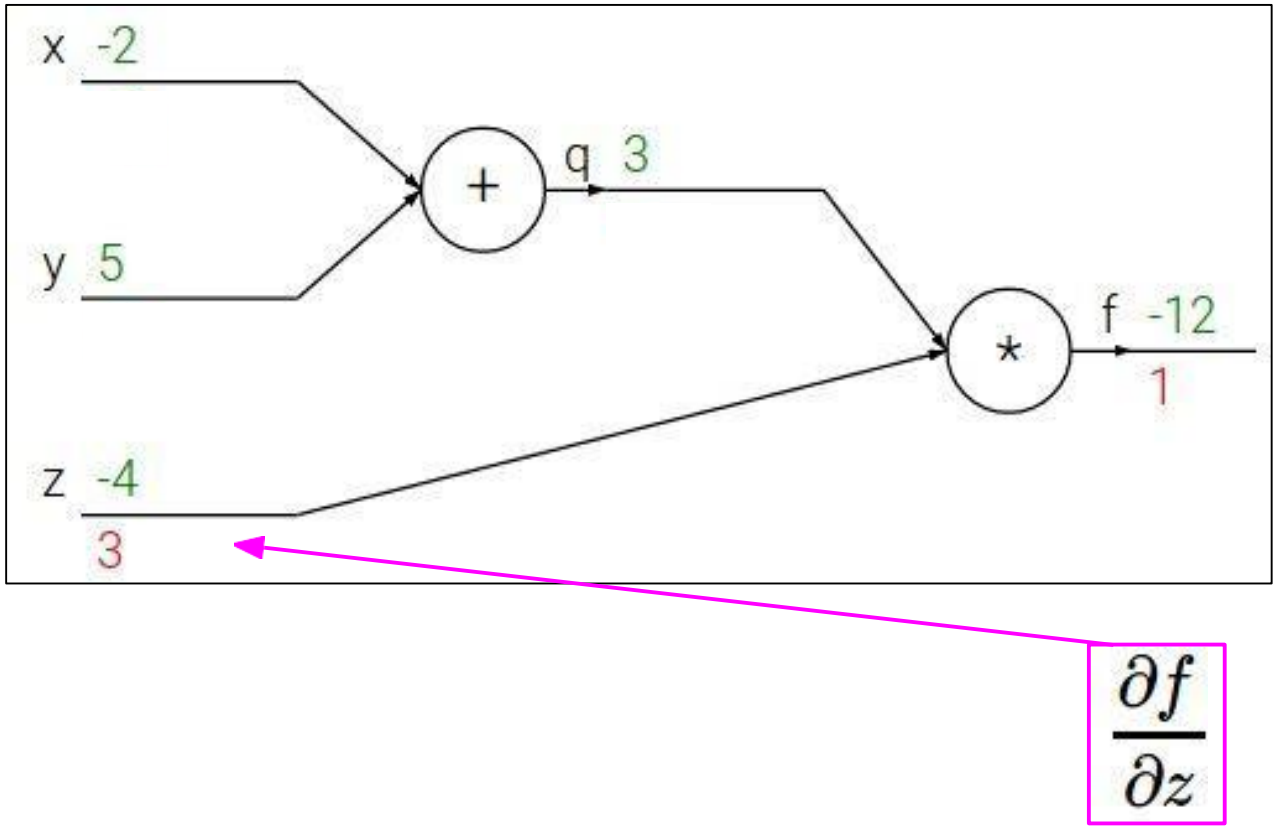
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

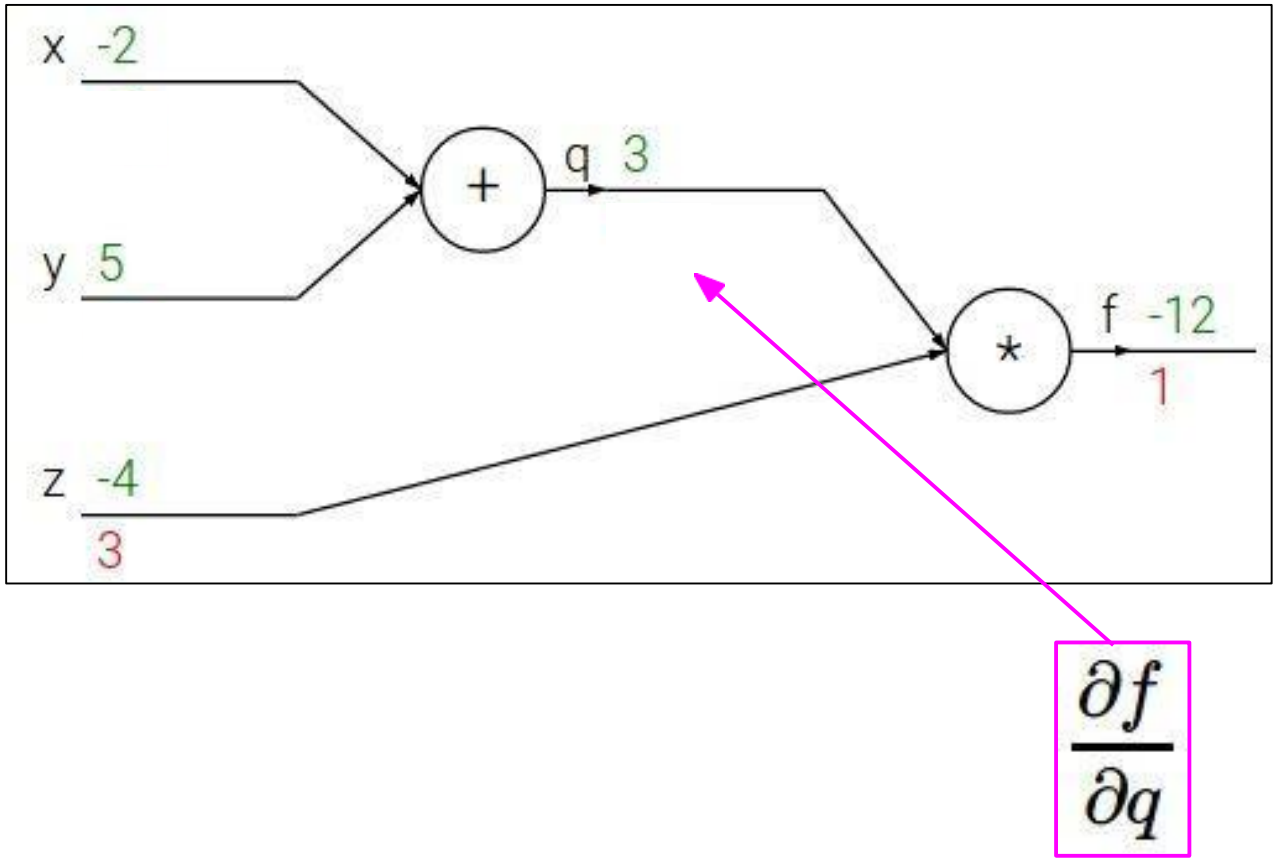
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



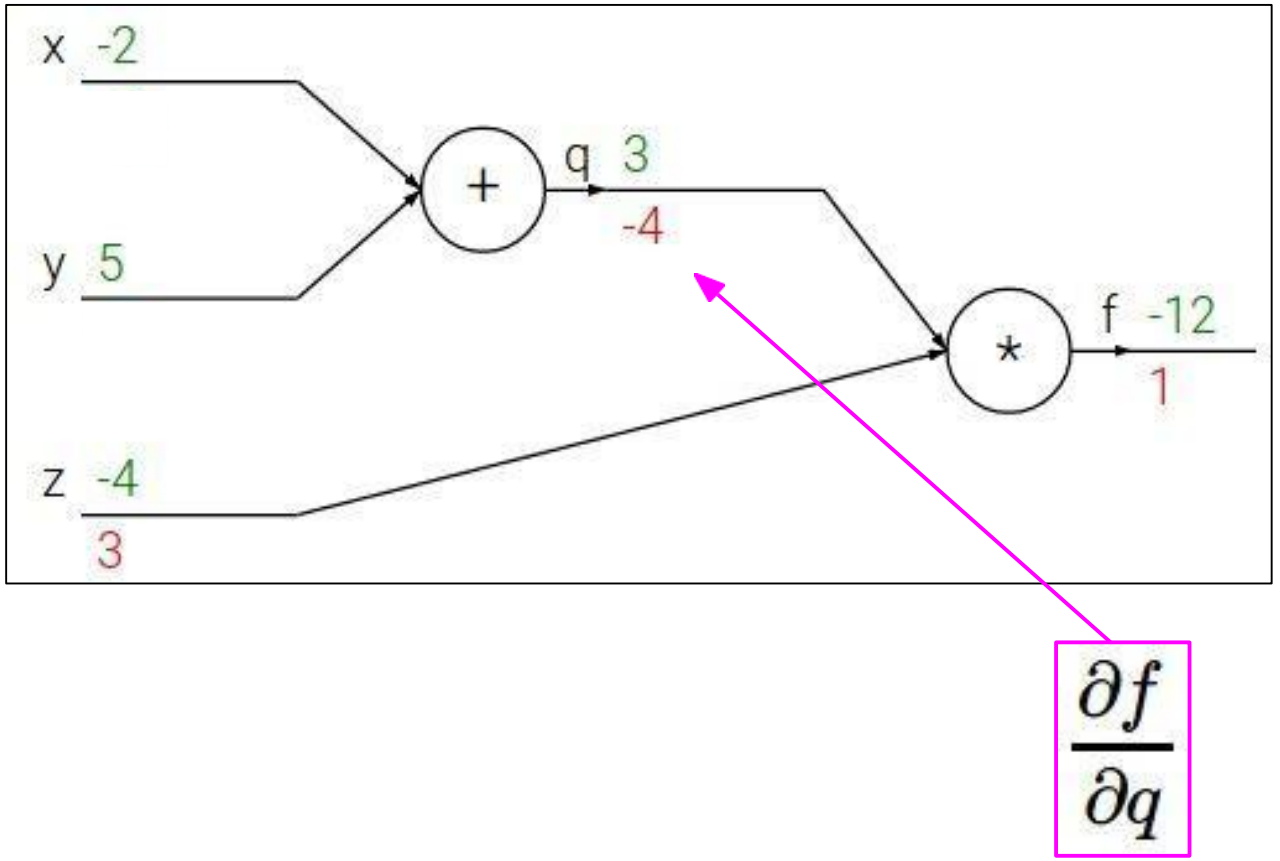
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



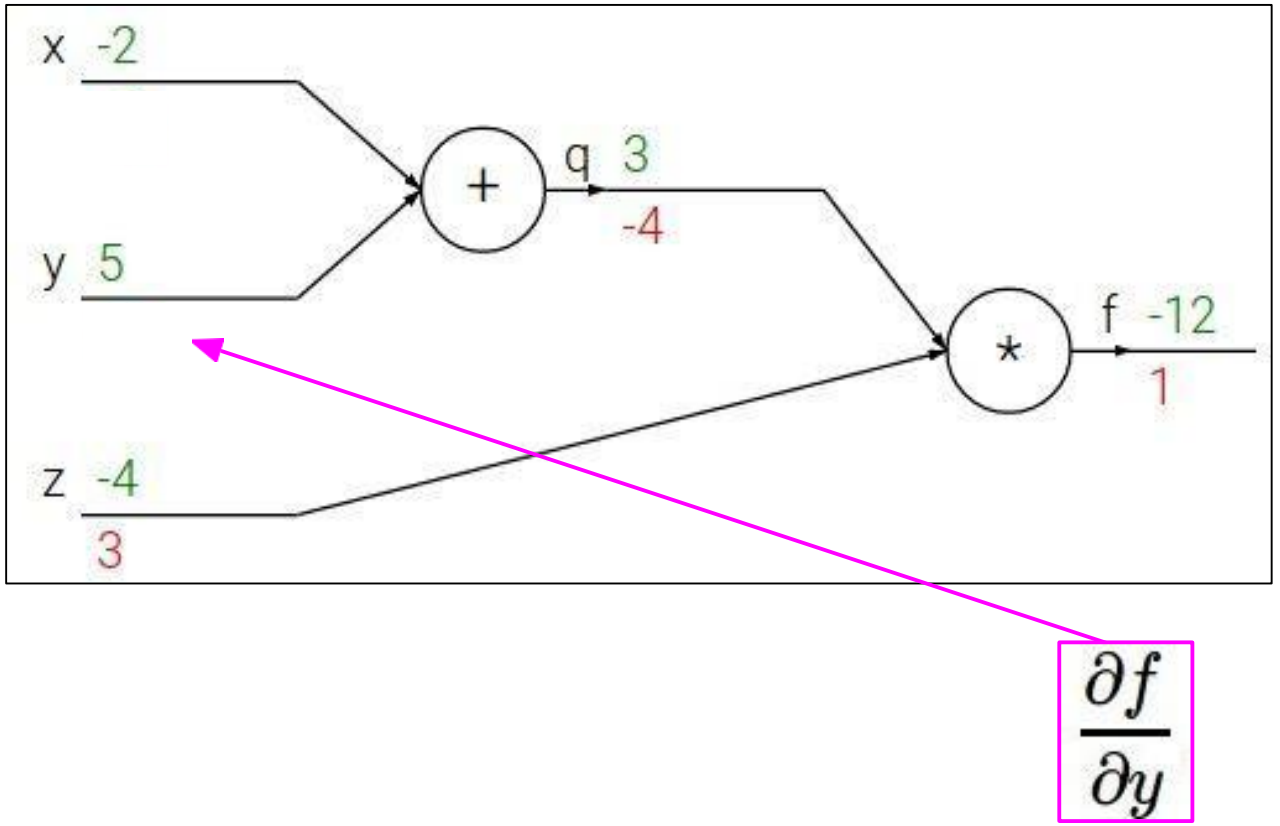
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



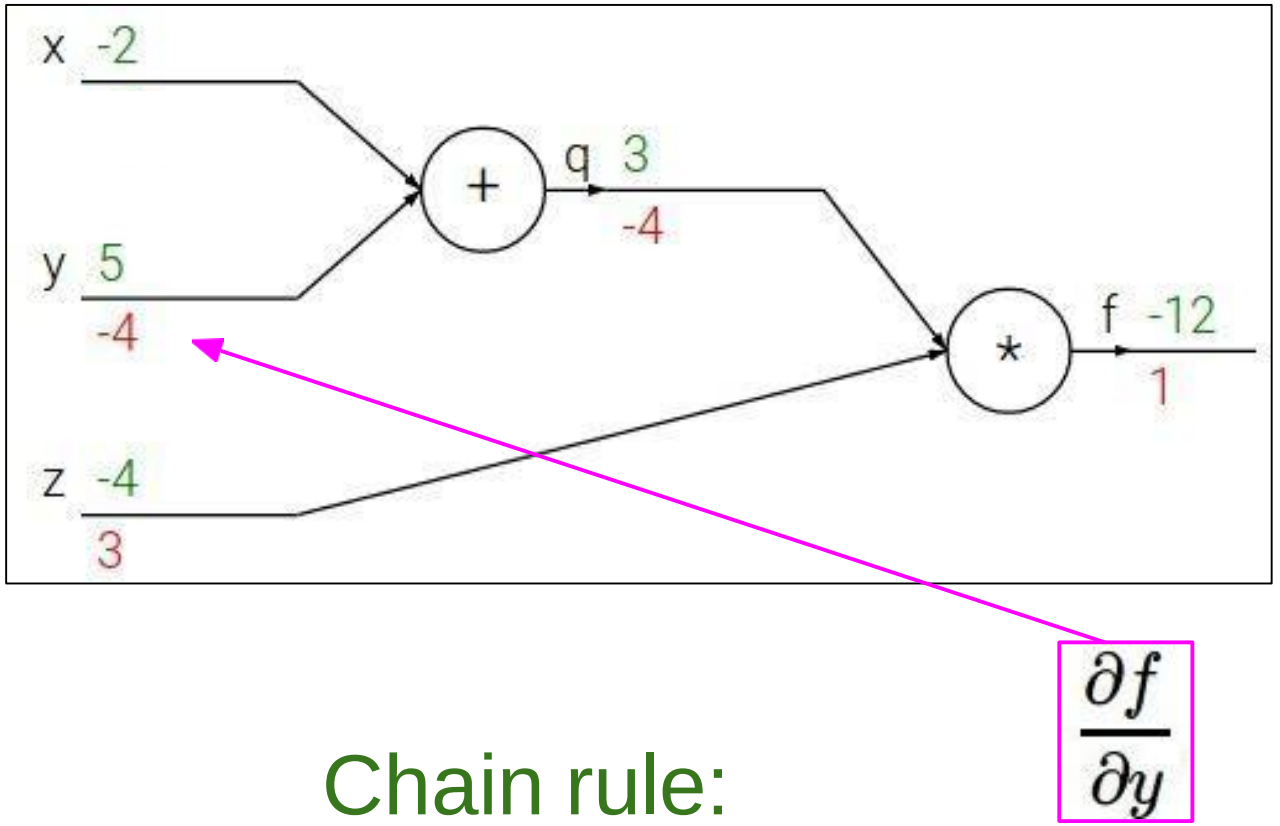
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

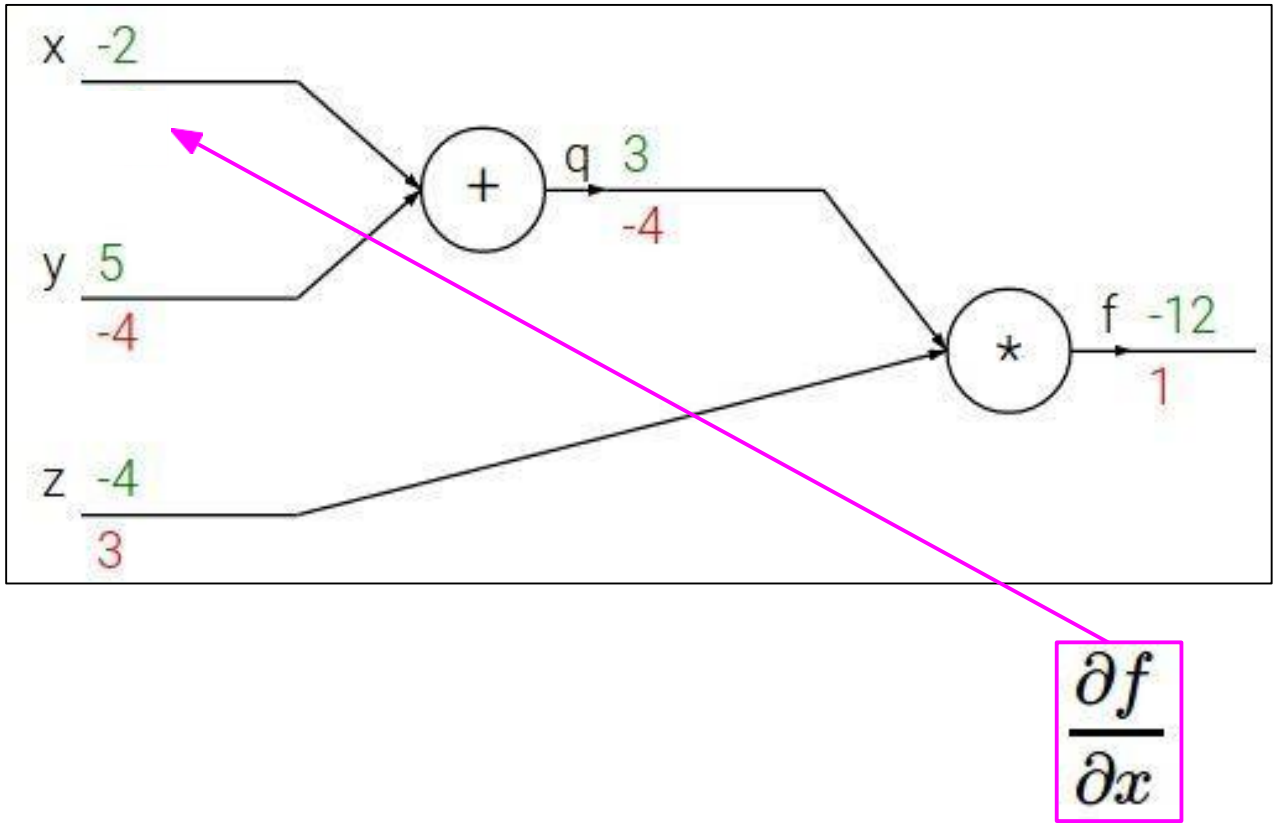
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



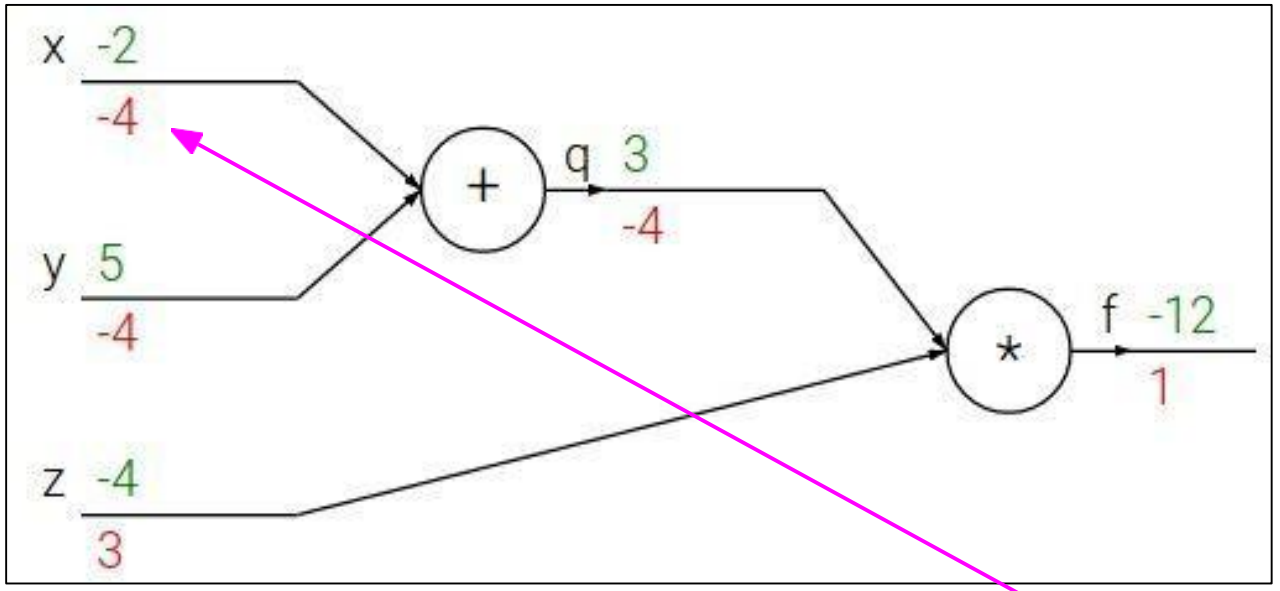
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

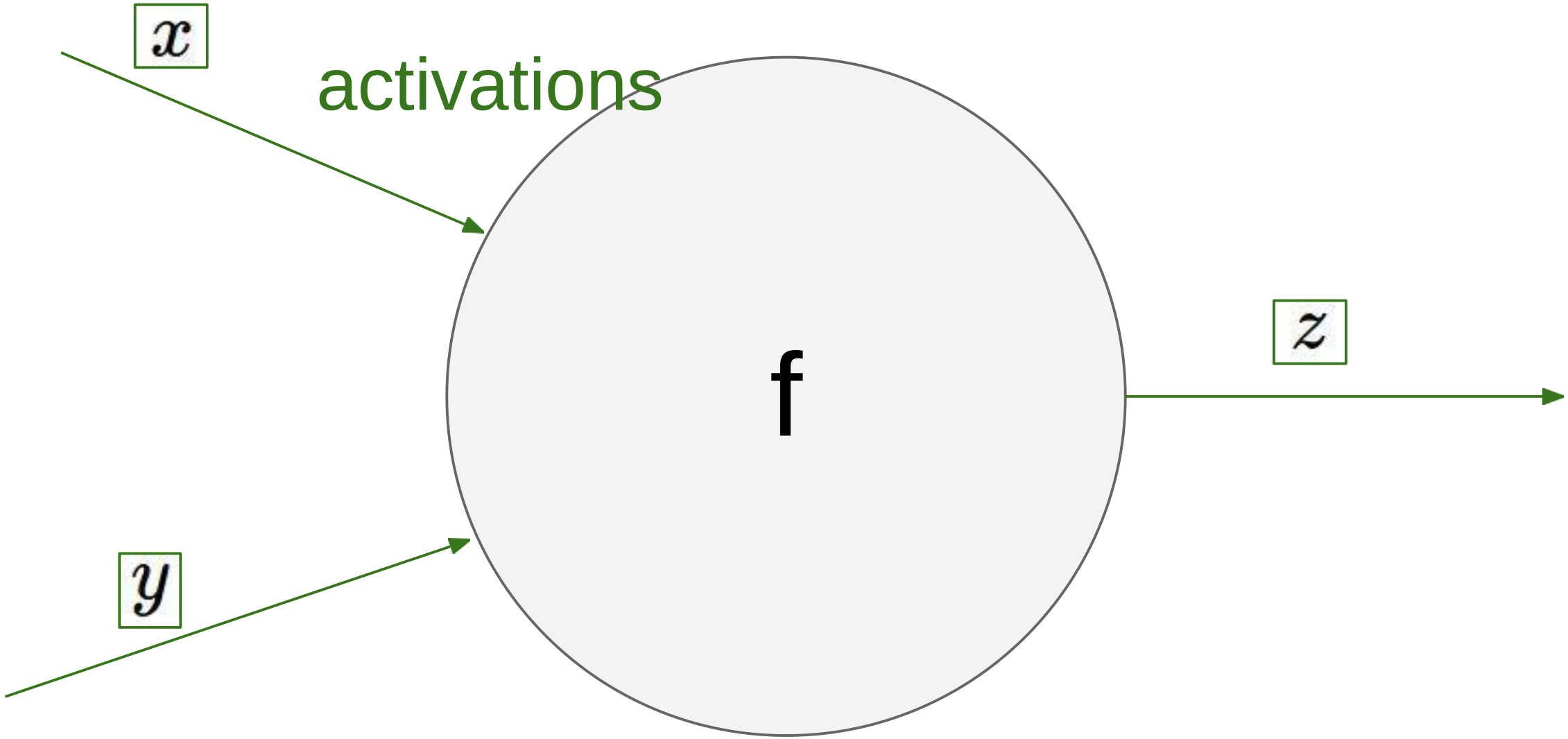
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



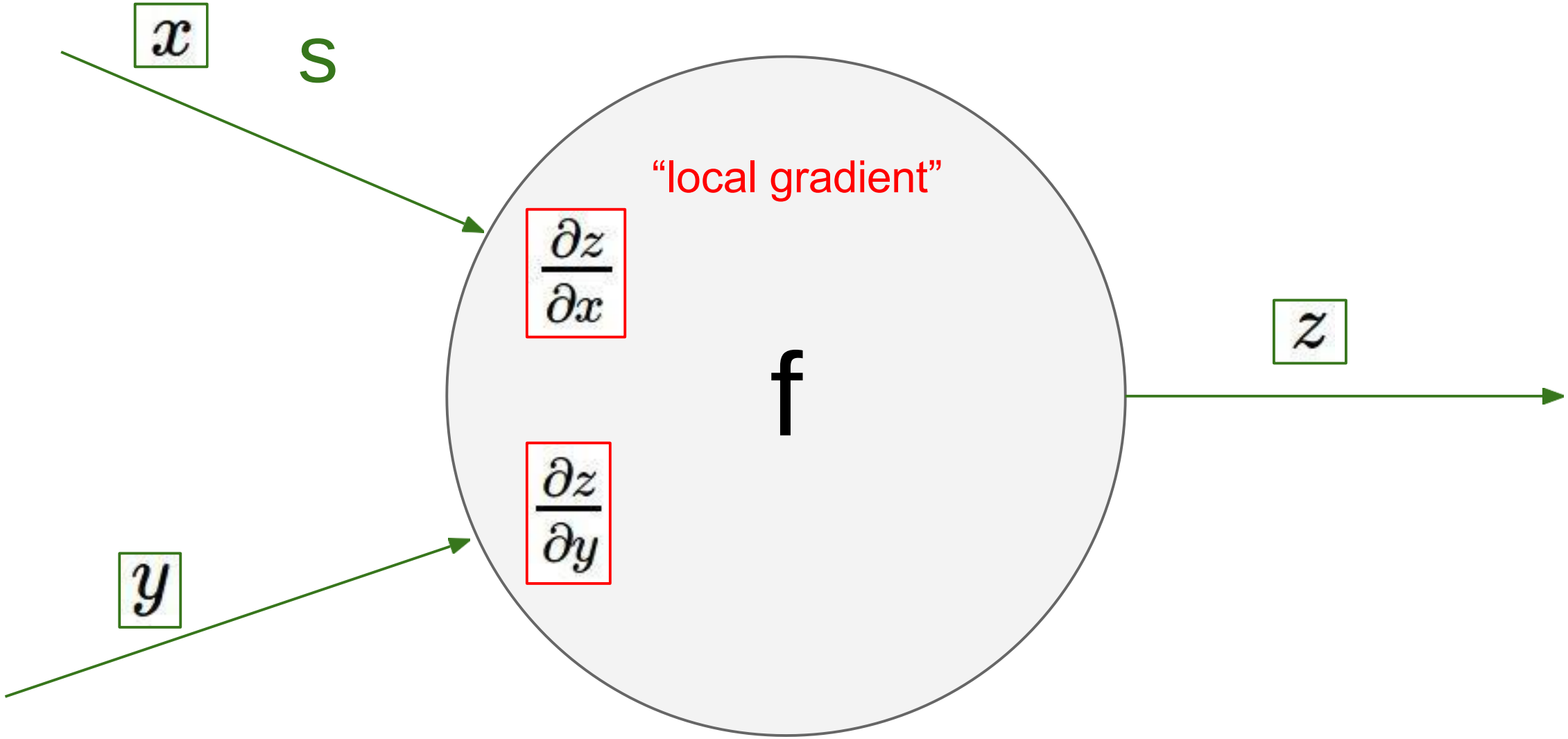
$\frac{\partial f}{\partial x}$

Chain rule:

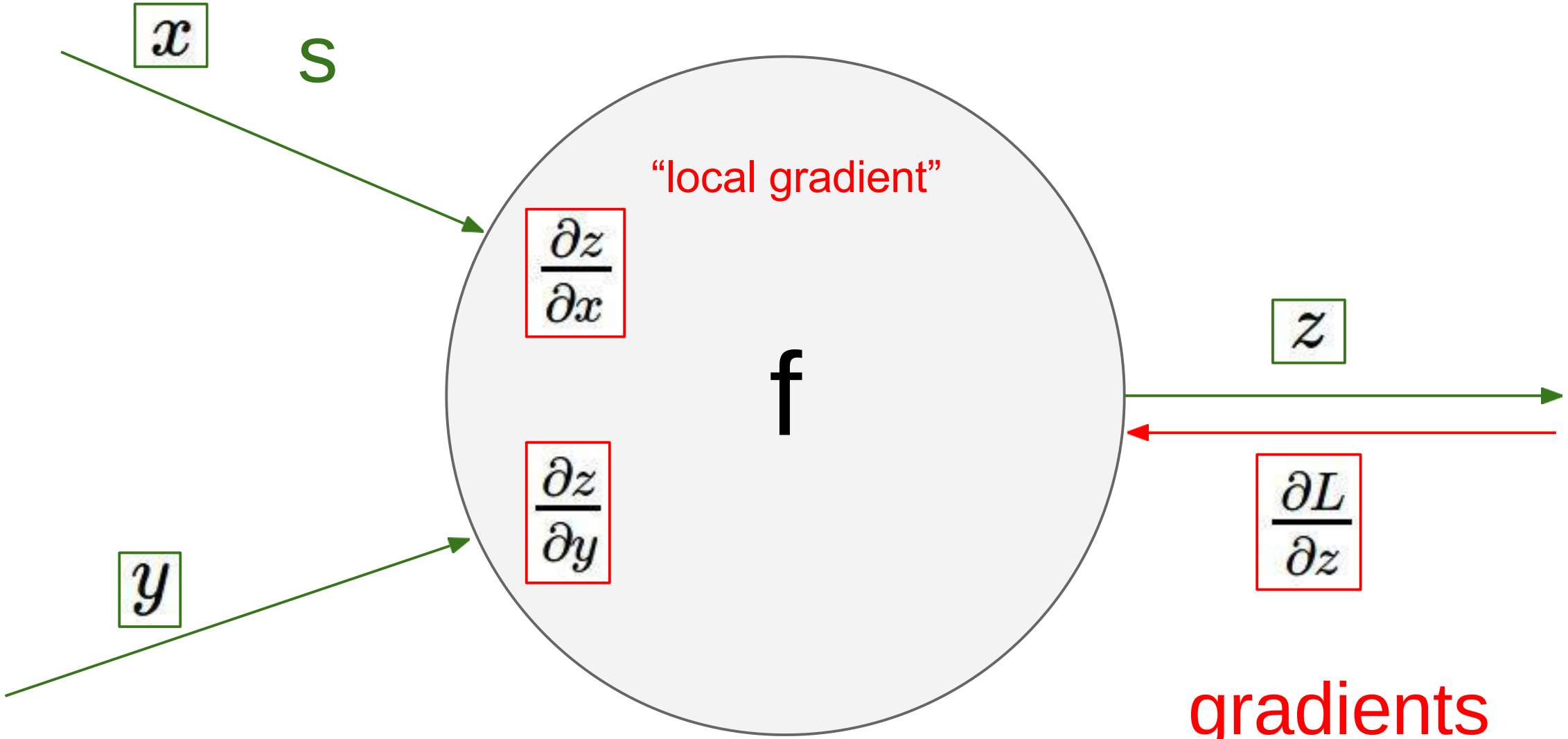
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$


x
activations
f
z
y

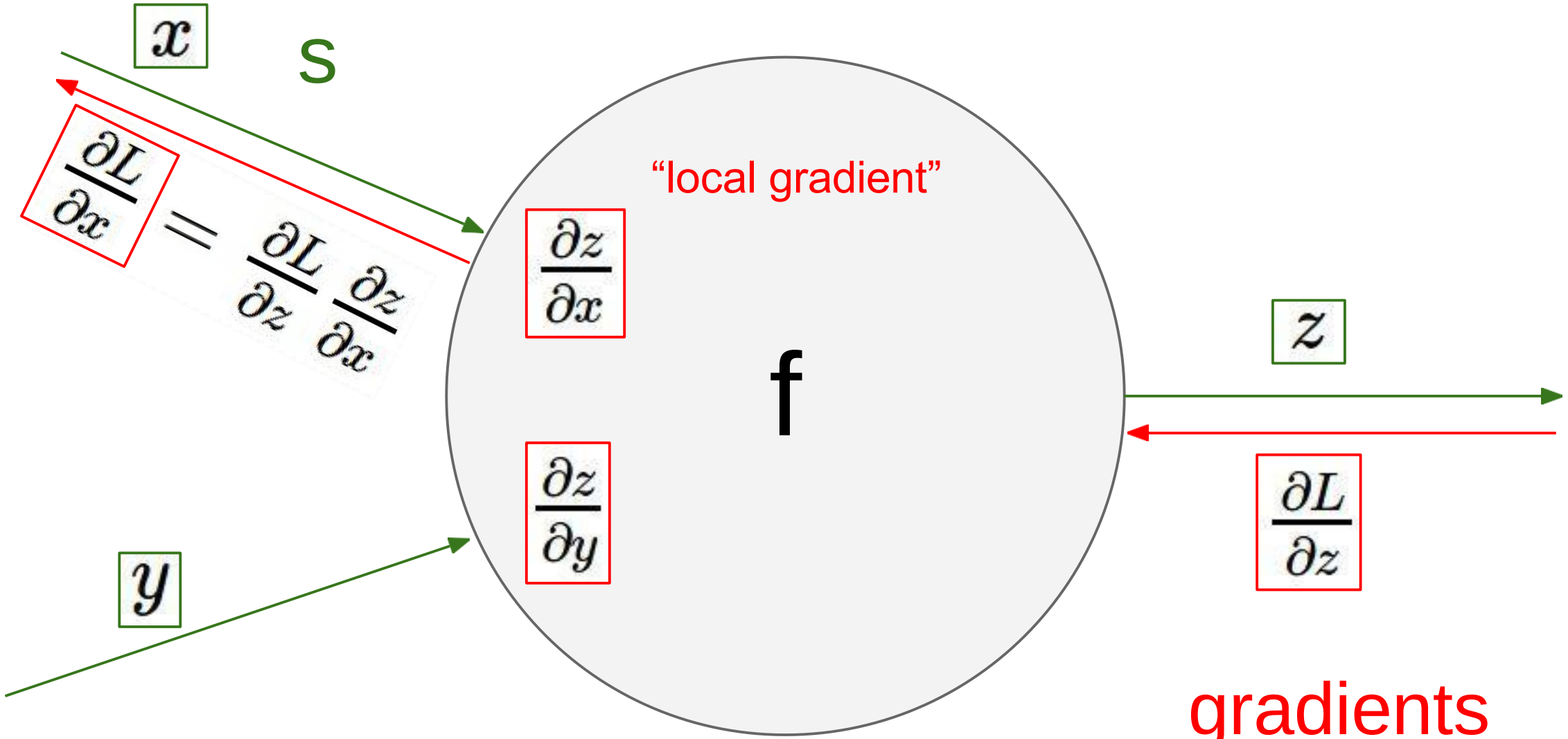
activation
s



activation
s

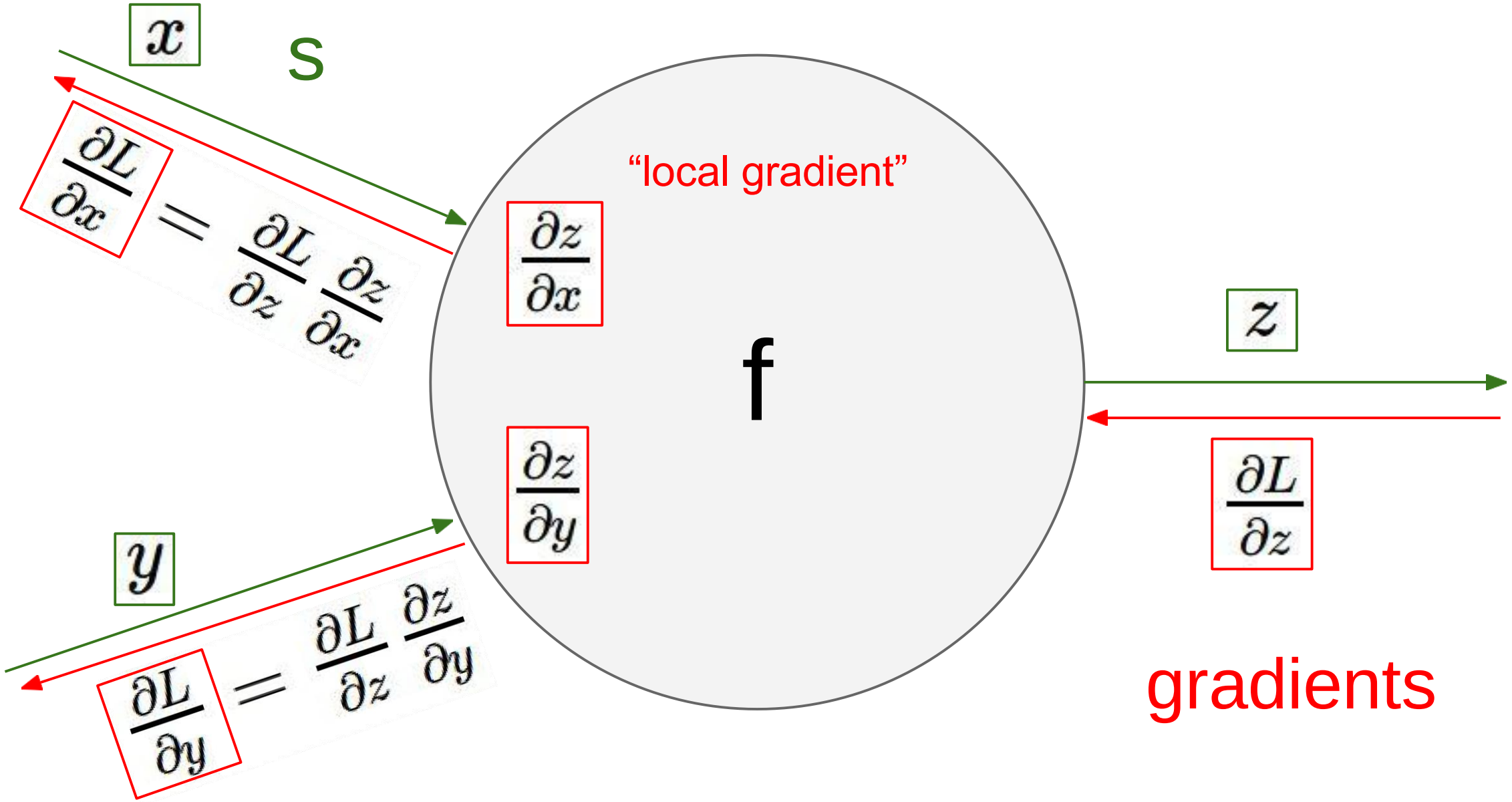


activation
s



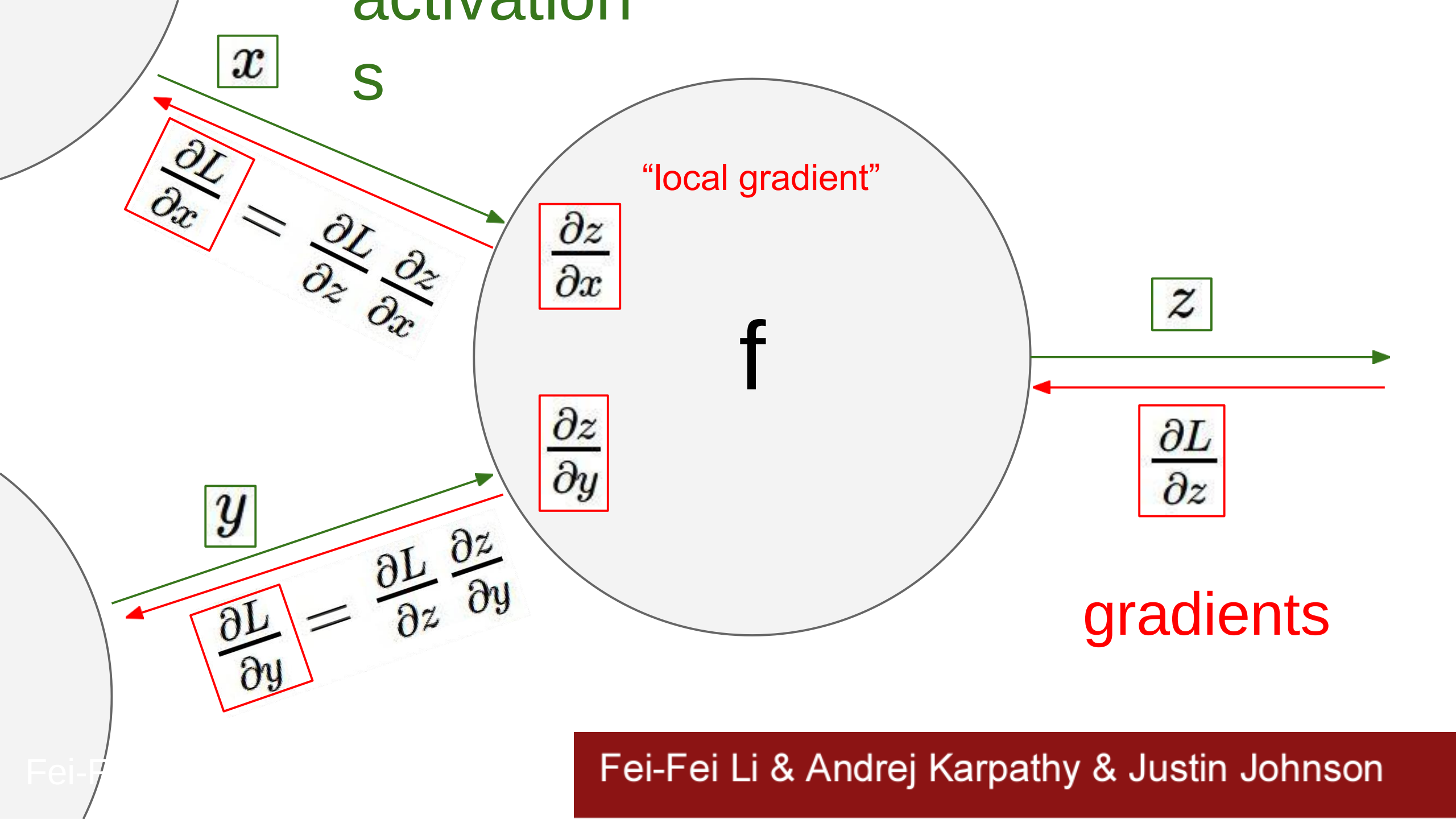
gradients

activation
s



gradients

activation
s



gradients

NN Training

1. Do **forwards propagation**, calculate the input sum and activation of each neuron by iteratively do matrix-vector multiplication and taking component-wise transfer function of all neurons in every layer. Save the results for later.
2. Calculate the **error signal of the final layer *L***, by obtaining the gradient of the cost function with respect to the outputs of the network.
3. Do **backwards propagation**, calculate the error signals of the neurons in each layer. The input sum of each neuron is required to do this, and it is also done by iteratively computing matrix-vector multiplications

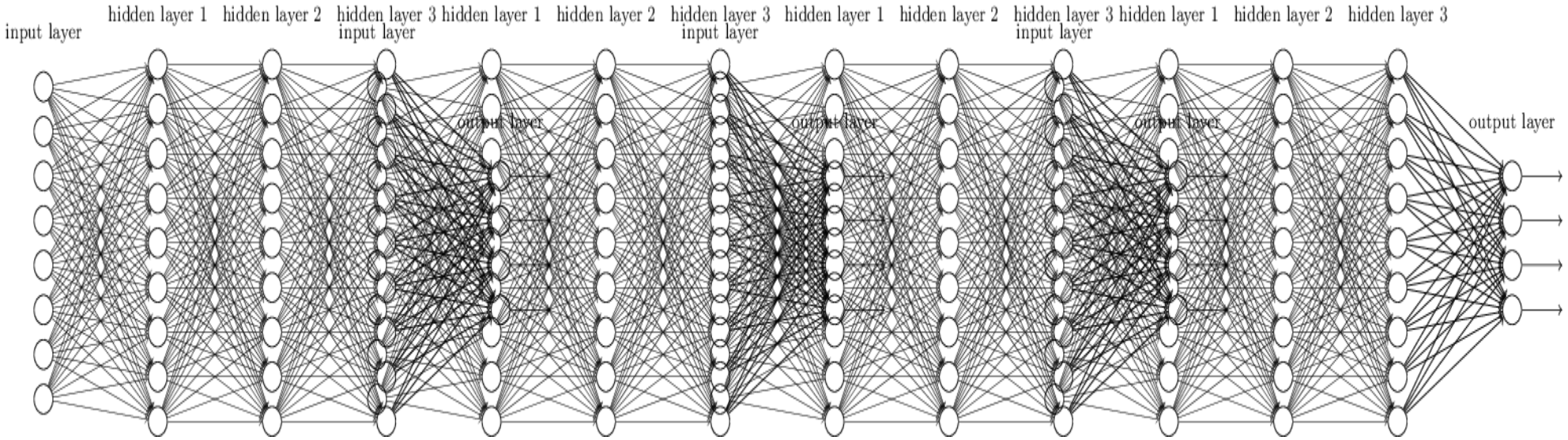
NN training cont’d

4.Calculate the **derivative of the cost function with respect to the weights**, the activation of each neuron is required to do this. This will be a matrix with the same shape as the weight matrix.

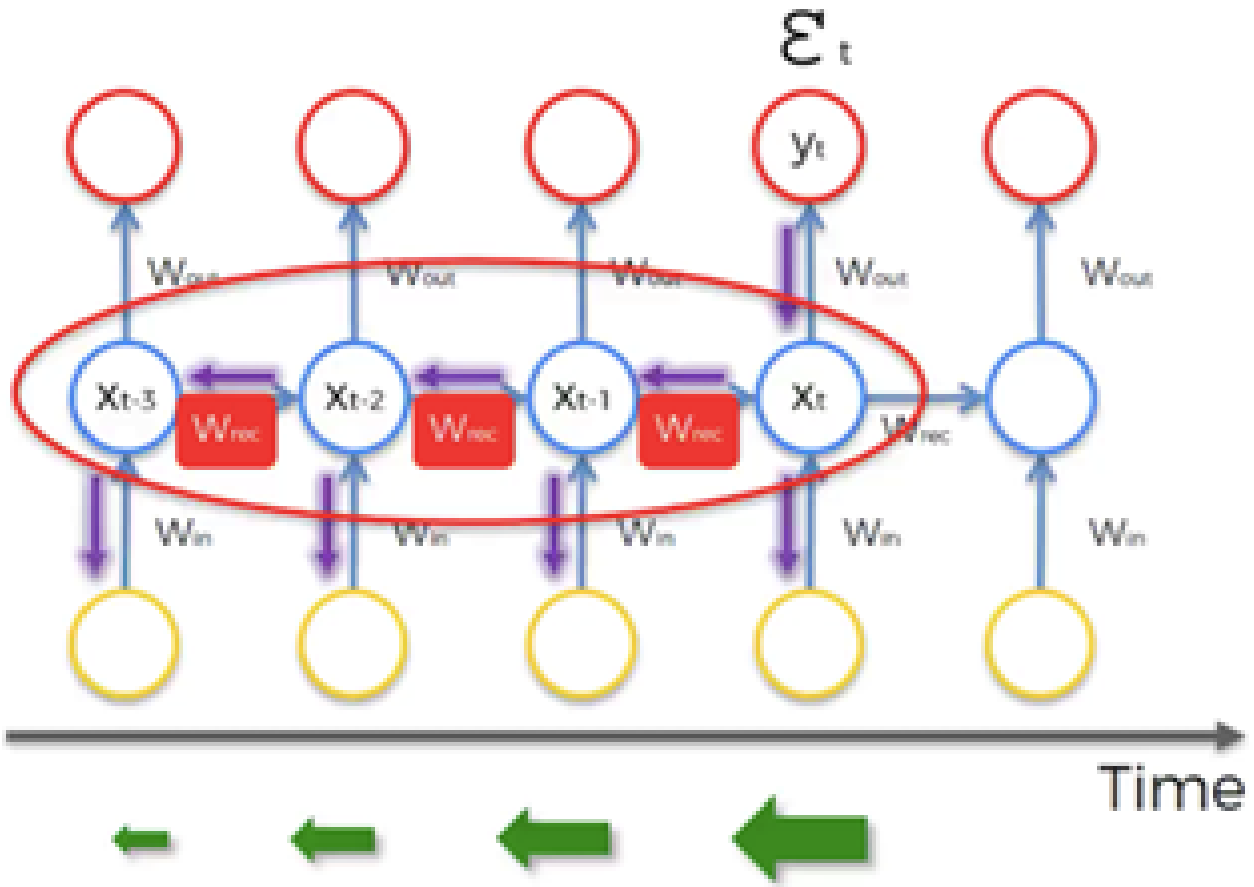
5.Calculate the **derivative of the cost function with respect to the biases** (this can be skipped). This will only be a column vector.

6.Update the weights according to the a gradient descent learning rule.

Back propagation



The Vanishing Gradient Problem



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

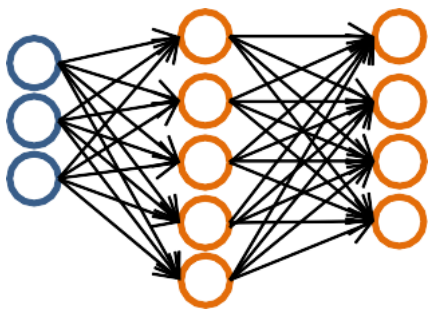
W_{rec} ~ small ⇨ Vanishing

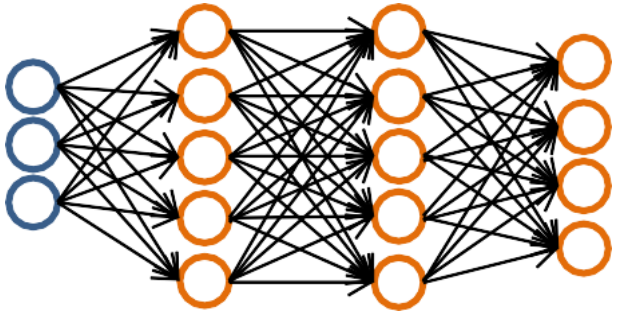
W_{rec} ~ large ⇨ Exploding

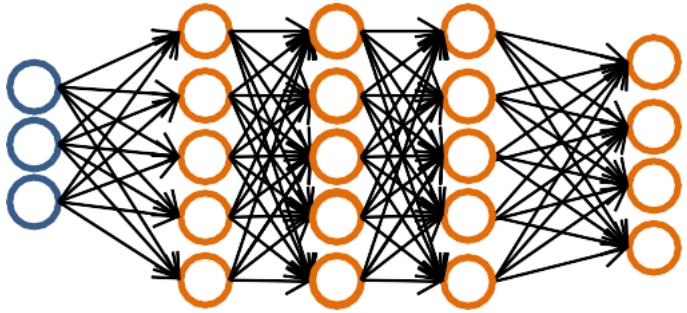
Formula Source: Razvan Pascanu et al. (2013)

Training a neural network

Pick a network architecture (connectivity pattern between neurons)







$x^{(i)}$

No. of input units: Dimension of features

No. output units: Number of classes

Reasonable default: 1 hidden layer, or if >1 hidden layer, have same no. of hidden units in every layer (usually the more the better)

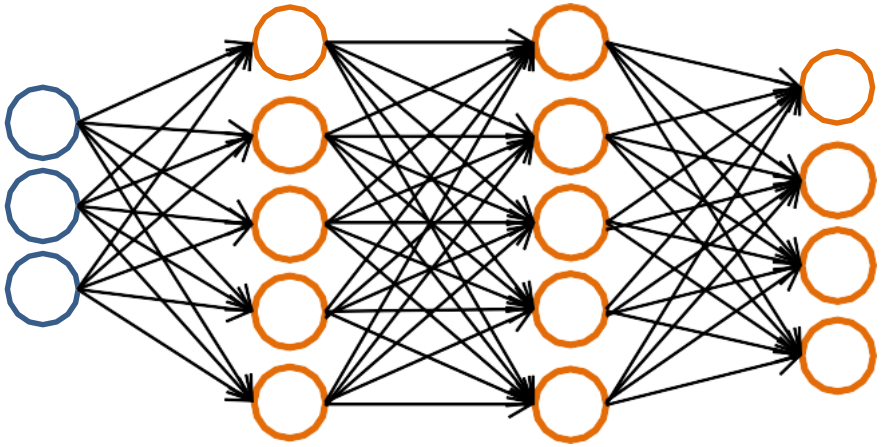
Multiple output units: One-vs-all.

Pedestrian

Car

Motorcycle

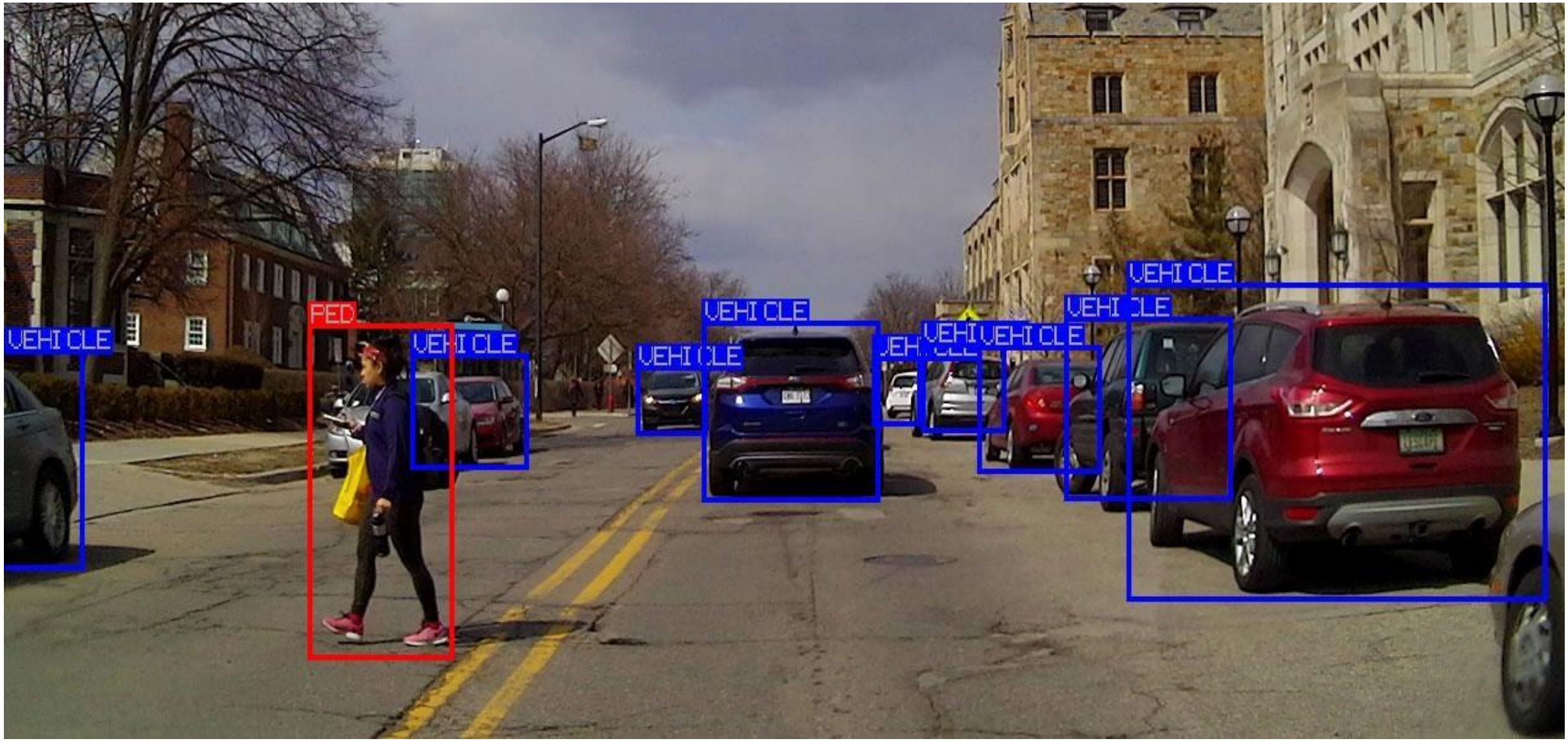
Truck



$$h_\Theta(x) \in \mathbb{R}^4$$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle


PED
VEHICLE
VEHICLE
VEHICLE
VEHICLE
VEHICLE
VEHICLE
VEHICLE

Issues in Training Neural Networks

- Initial values of parameters
  - Back-propagation finds local minimum
- Overfitting
  - Neural networks have too many parameters
  - Early stop and regularization
- Scaling of the inputs
  - Inputs are typically scaled to have zero mean and unit standard deviation
- Number of hidden units and layers
  - Better to have too many than too few
  - With ‘traditional’ back-propagation a long NN gets stuck in local minima and does not learn well

Neural network objective

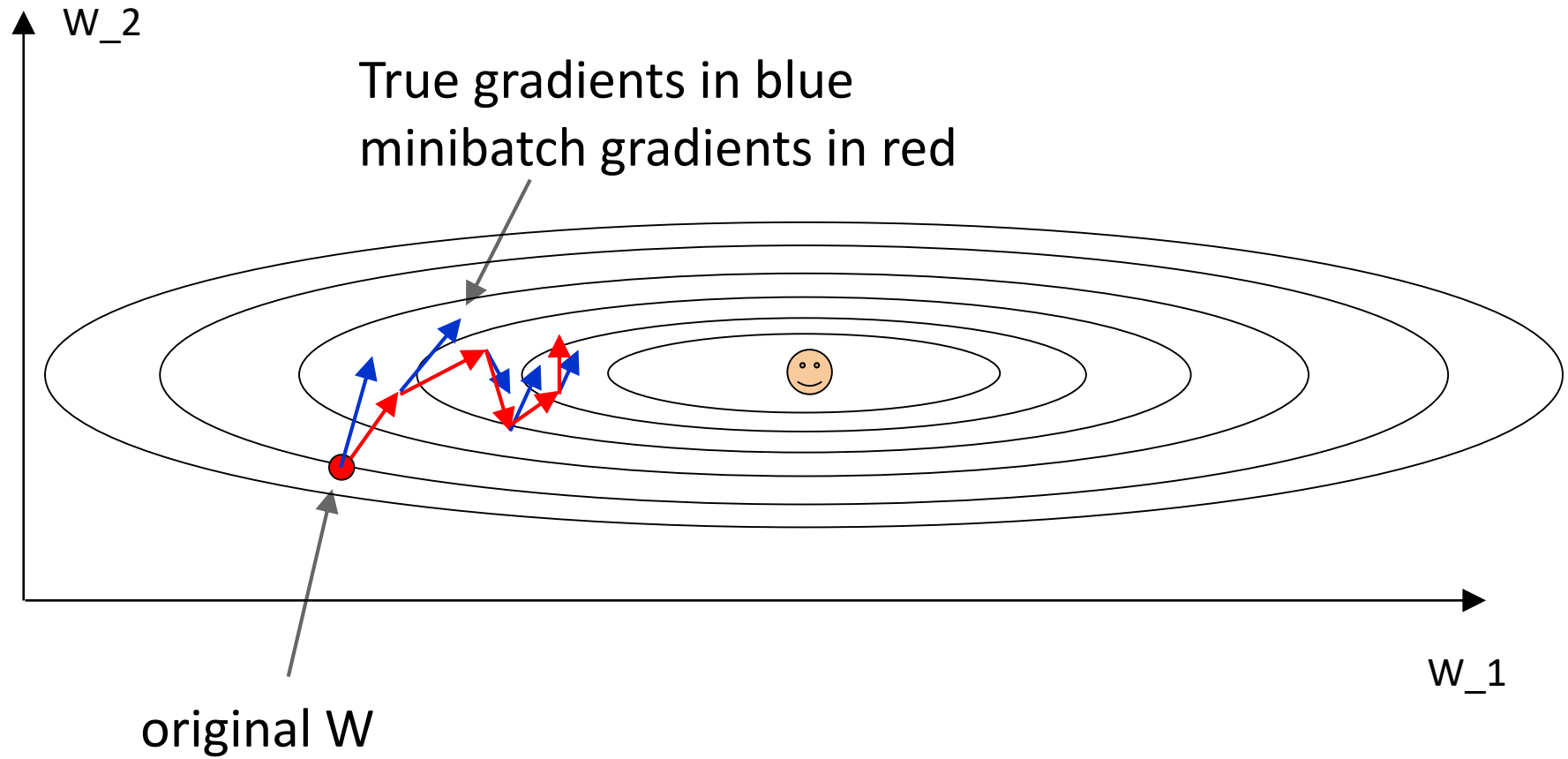
- Non-linear objective, multiple local minima
- As a result optimization is much harder than that of a convex objective
- Standard approach: gradient descent:
  - Calculate first derivatives of each hidden variable
  - For inner layers we use the chain rule

SGD vs. GD

GD spans over entire dataset once (which is same as one epoch) before each update, whereas SGD randomly takes just one data point for each update

In mini-batch GD we update the parameters based on small samples of size ‘b’ of the dataset. So instead of taking 1 data point as in the case of SGD or complete dataset as in the case of GD, we randomly take ‘b’ data points from the dataset, calculate the gradient and update the parameter, where 1 <b <N (where N is the size of the dataset).

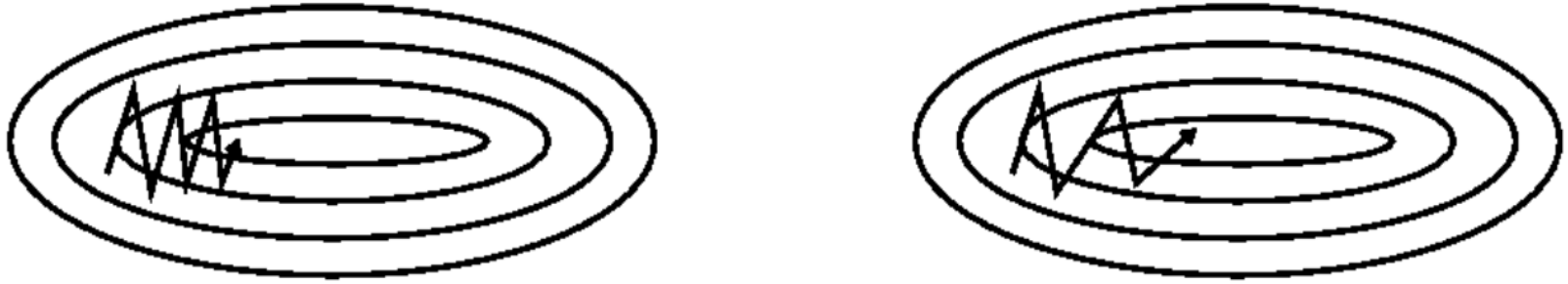
we can see that SGD is just a special case of mini-batch GD where b = 1. And when b = N, it is GD.



Gradients are noisy but still make good progress on average

Momentum

An overview of gradient descent optimization algorithms
https://arxiv.org/pdf/1609.04747.pdf



(a) SGD without momentum

(b) SGD with momentum

Figure 2: Source: Genevieve B. Orr

Momentum [17] is a method that helps accelerate SGD in the relevant direction and dampens oscillations as can be seen in Figure 2b. It does this by adding a fraction γ of the update vector of the past time step to the current update vector[8]

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{4}$$

- Momentum in neural networks is a **variant of the *stochastic gradient descent***. It replaces the gradient with a *momentum* which is an aggregate of gradients

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment — learning rate — weight gradient

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor — weight increment, previous iteration

Adam (Adaptive Moment Estimation) (Momentum + RMS Prop)

Adaptive Moment Estimation (Adam) [10] is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{aligned} \tag{19}$$

m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method. As m_t and v_t are initialized as vectors of 0's, the authors of Adam observe that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e. β_1 and β_2 are close to 1).

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Optimizers

- If you are working with novel machine learning methods, odds are there exist one or more reliable papers which cover a similar problem or handle similar data. Oftentimes the authors of the paper have done extensive cross-validation and report only the most successful configurations. Try to understand the reasoning for their choice of optimizer.

Optimizer	State Memory [bytes]	# of Tunable Parameters	Strengths	Weaknesses
SGD	0	1	Often best generalization (after extensive training)	Prone to saddle points or local minima Sensitive to initialization and choice of the learning rate α
SGD with Momentum	$4n$	2	Accelerates in directions of steady descent Overcomes weaknesses of simple SGD	Sensitive to initialization of the learning rate α and momentum β
AdaGrad	$\sim 4n$	1	Works well on data with sparse features Automatically decays learning rate	Generalizes worse, converges to sharp minima Gradients may vanish due to aggressive scaling
RMSprop	$\sim 4n$	3	Works well on data with sparse features Built in Momentum	Generalizes worse, converges to sharp minima
Adam	$\sim 8n$	3	Works well on data with sparse features Good default settings Automatically decays learning rate α	Generalizes worse, converges to sharp minima Requires a lot of memory for the state
AdamW	$\sim 8n$	3	Improves on Adam in terms of generalization Broader basin of optimal hyperparameters	Requires a lot of memory for the state
LARS	$\sim 4n$	3	Works well on large batches (up to 32k) Counteracts vanishing and exploding gradients Built in Momentum	Computing norm of gradient for each layer can be inefficient