

BUILDING BLOCKS

OF

DEEP

LEARNING

A **ADI** TYA PRAKASH

September 12, 2018

[Online version link \(with animations\)](#)



SCREAM



SHIPWRECK
OF MINOTAUR



UDNIE



WAVE



RAIN
PRINCESS

ART LIVES FOREVER



MUNCH

TURNER

PICABIA

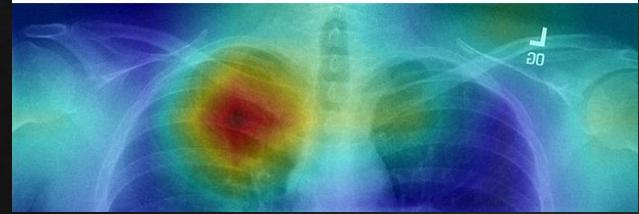
HOKUSAI

AFREMOV

~~ART LIVES FOREVER
ARTIST~~

New AI Can Diagnose Pneumonia Better Than Doctors

The software can greatly help in avoiding the misdiagnosis of pneumonia.





WHAT WE WILL LEARN

Neural
Networks

Learning
Process

Convolution
Networks

TEXTBOOK

[HTTP://WWW.DEEPLEARNINGBOOK.ORG](http://www.deeplearningbook.org)

COURSE

[HTTP://CS231N.STANFORD.EDU](http://cs231n.stanford.edu)

QUICK GUIDE

[HTTP://NEURALNETWORKSANDDEEPLEARNING.COM](http://neurallnetworksanddeeplearning.com)

[YOUTUBE: 3BLUE1BROWN](#)

Machine Learning recap

Apply a prediction function to a feature representation of given input (x) and get the desired output.

$$f(\text{apple icon}) = \text{"apple"}$$

$$f(\text{bird icon}) = \text{"bird"}$$

$$f(\text{bike icon}) = \text{"bike"}$$

Machine Learning recap

Training Data → model → output

x

f

\tilde{y}

$f(x) \rightarrow \tilde{y}$

$y - \tilde{y} \rightarrow \text{error}$

$L(y, \tilde{y}) \rightarrow \text{Loss/Objective}$

Loss function L quantifies how unhappy you would be if you used ' f ' to make predictions on x .
It is the objective we want to minimize

Machine Learning recap

Training Data → model → output

x f \tilde{y}

$$\text{TrainLoss} = \sum_i \text{Loss}(f(x_i), y_i)$$

$$\text{Model} = \underset{f}{\operatorname{argmin}} \text{ TrainLoss}$$

Find ' f ' such that it minimizes the 'training loss', and hope that this is also true for 'test' loss.

Machine Learning recap



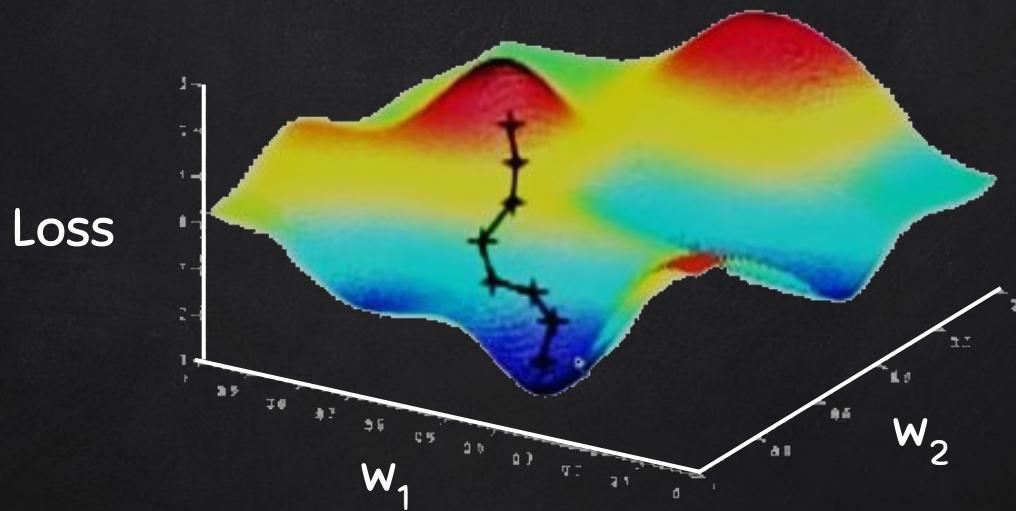
$$\text{Model} = \underset{f}{\operatorname{argmin}} \text{ TrainLoss}$$

$$W \leftarrow W - \eta \nabla_w \text{TrainLoss}(f)$$

$$w = w - \eta \frac{\partial L}{\partial w}$$

Process of updating 'weights' like this is called gradient descent

Gradient Descent



1.

ARTIFICIAL NEURAL NETWORK



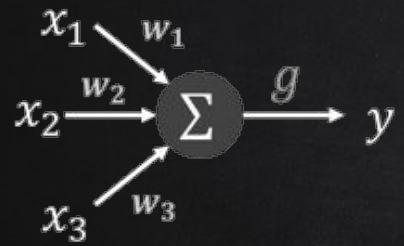
ARTIFICIAL NEURAL NETWORK

Organic is overrated

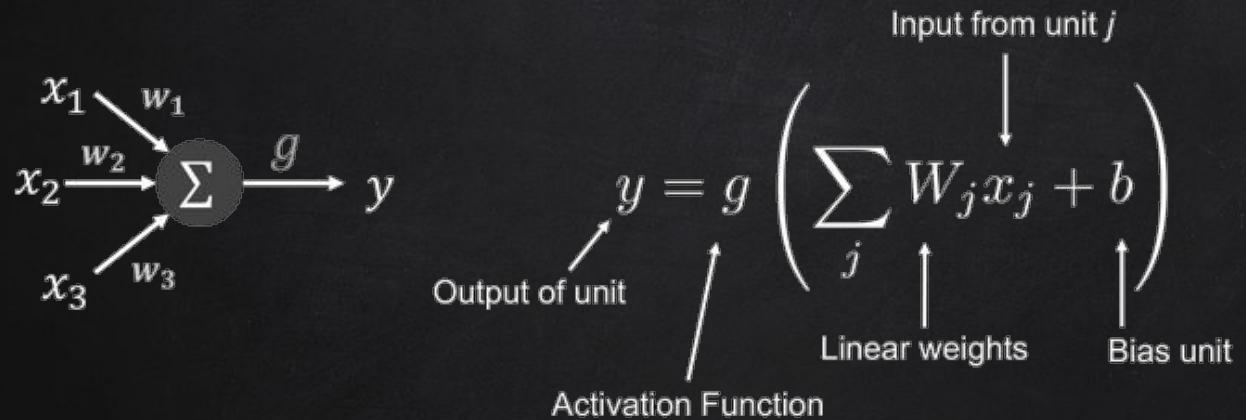
NEURON



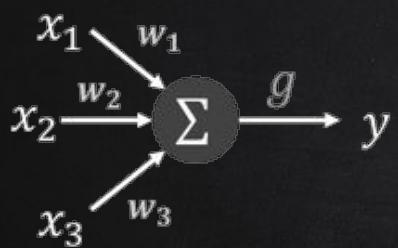
ARTIFICIAL NEURON



ARTIFICIAL NEURON



ARTIFICIAL NEURON

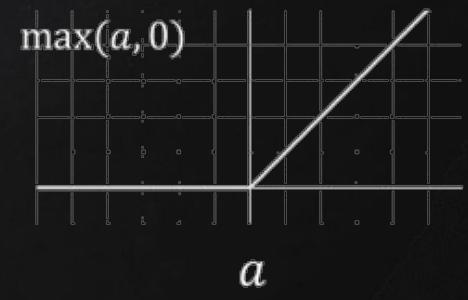
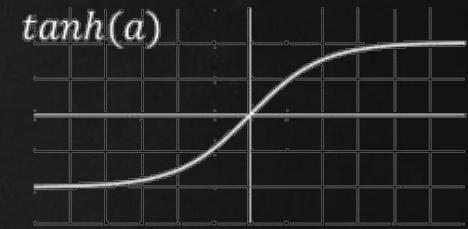


The mathematical formula for an artificial neuron is:

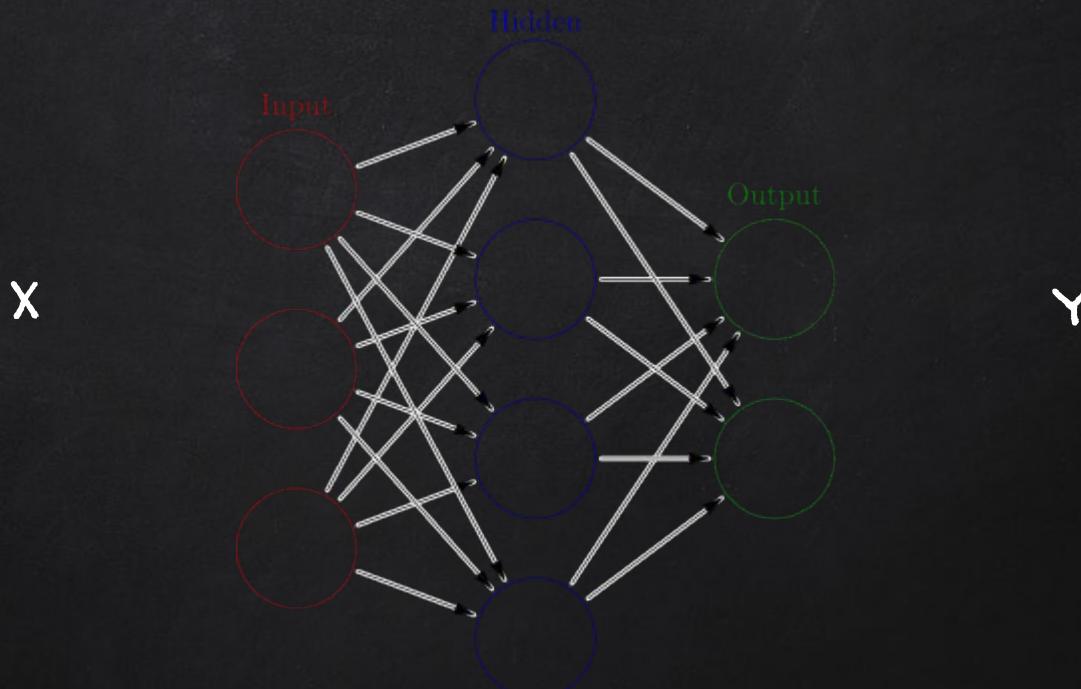
$$y = g \left(\sum_j W_j x_j + b \right)$$

Annotations explain the components:

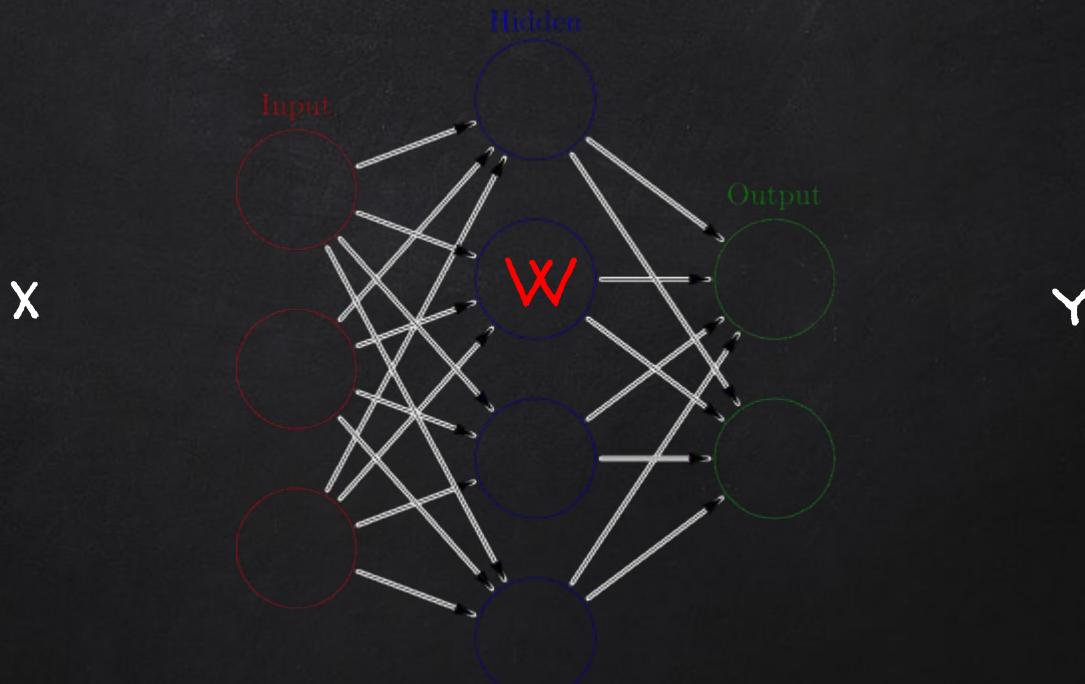
- "Input from unit j " points to the term $W_j x_j$.
- "Output of unit" points to the variable y .
- "Linear weights" points to the term $\sum_j W_j x_j$.
- "Bias unit" points to the term b .
- "Activation Function" points to the function g .



ARTIFICIAL NEURAL NETWORKS



ARTIFICIAL NEURAL NETWORKS



ARTIFICIAL NEURAL NETWORKS

$F(x, w) \rightarrow y$

ARTIFICIAL NEURAL NETWORKS

$F(x, w)$ → y

+ B

IMAGE CLASSIFICATION



Choose among the following ---

Cairn terrier

(b) Norwich terrier

(c) Australian terrier

IMAGE CLASSIFICATION



Choose among the following ---

Cairn terrier

(b) Norwich terrier

(c) Australian terrier

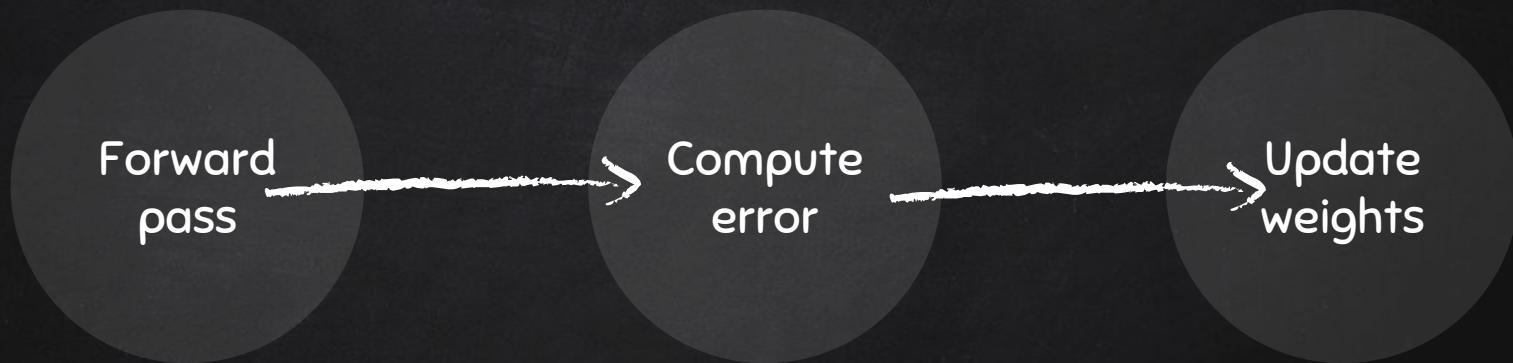


USING MAGIC

Neural networks and **back**propagation



BACK PROPAGATION

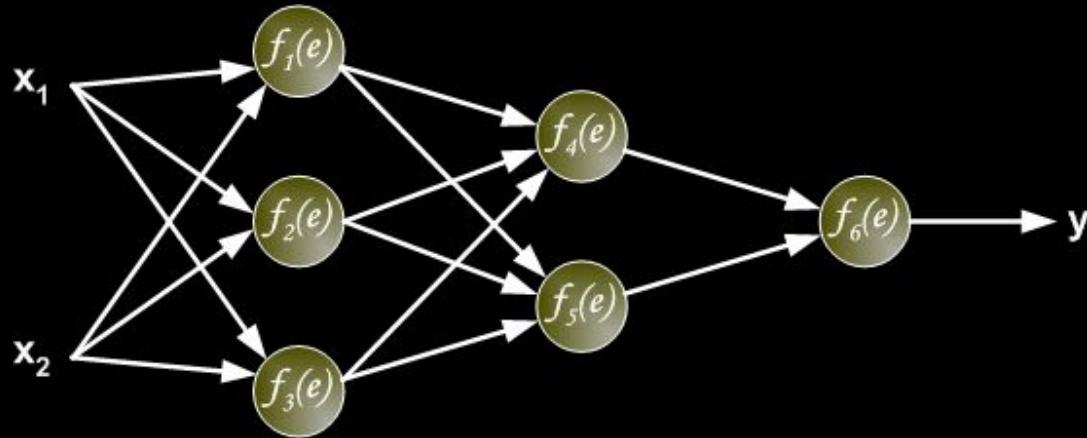




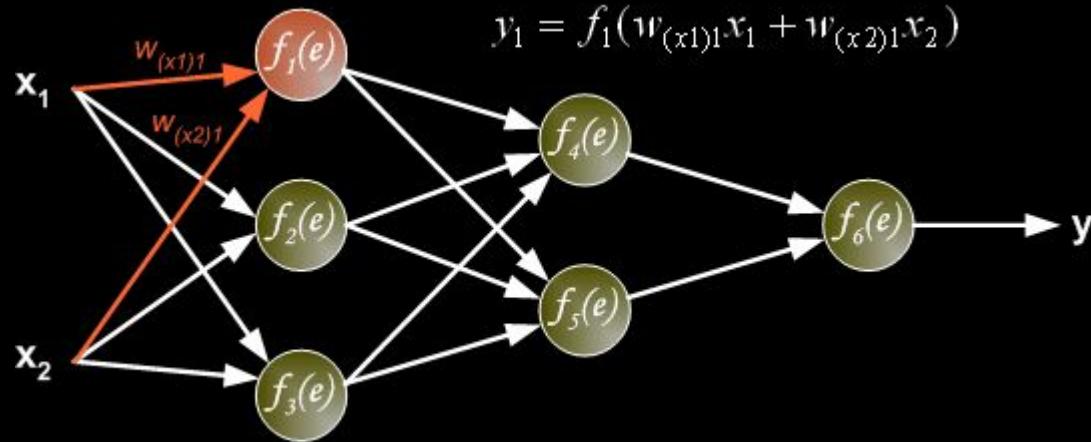
BACK PROPAGATION

Forward
Pass

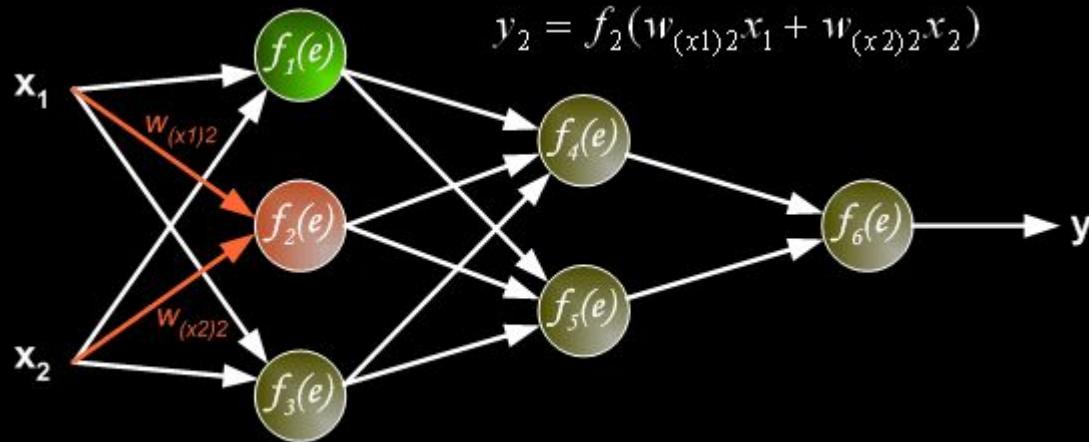
FORWARD PASS



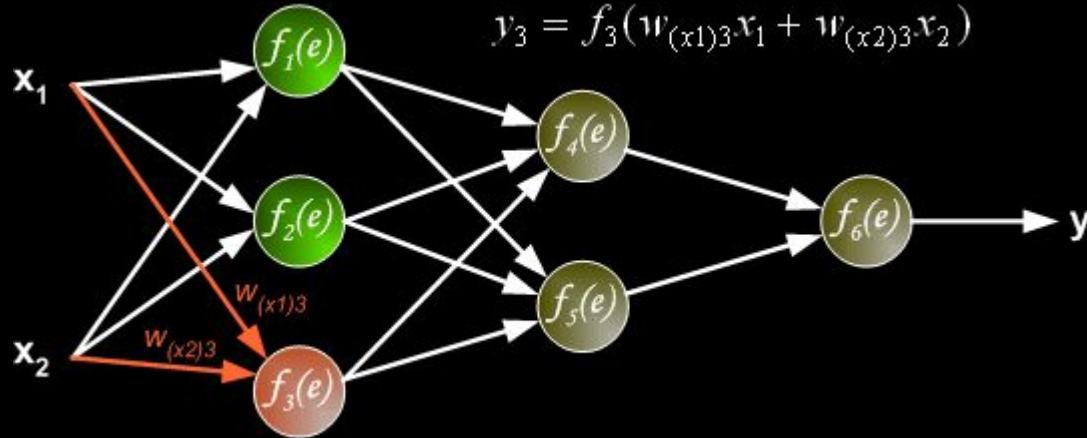
FORWARD PASS



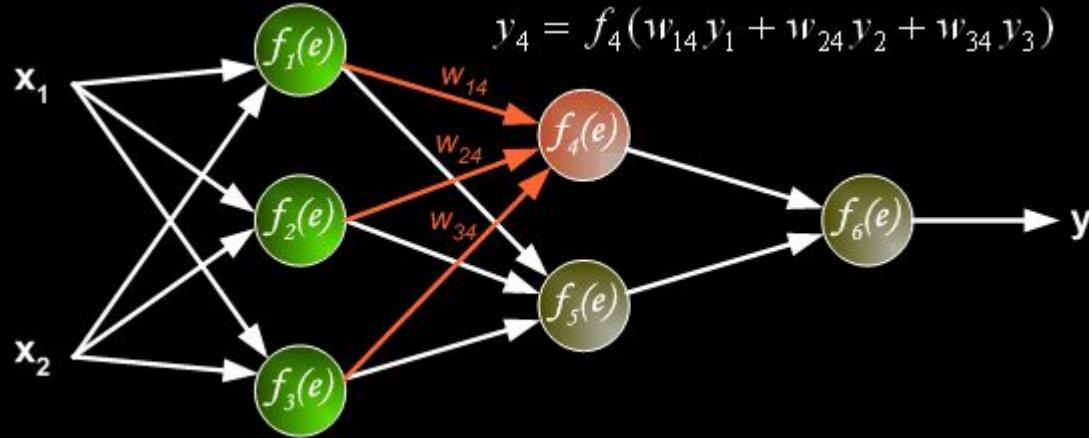
FORWARD PASS



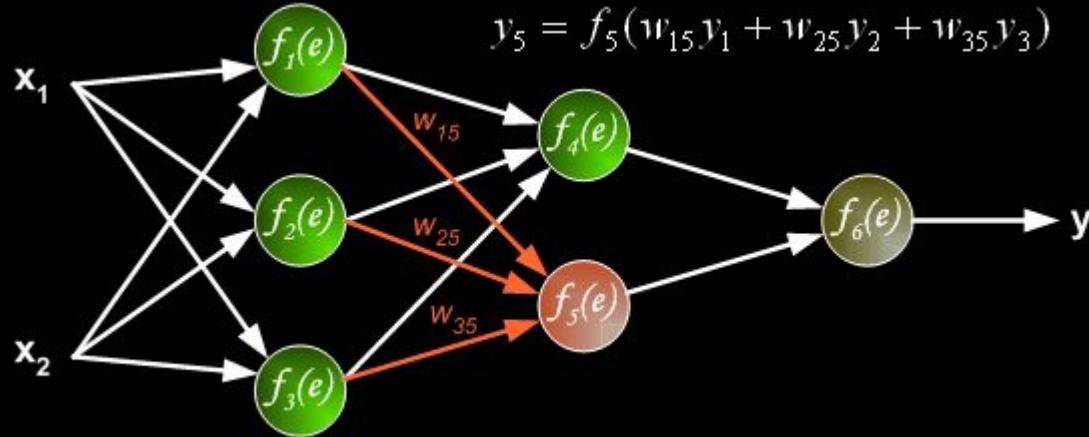
FORWARD PASS



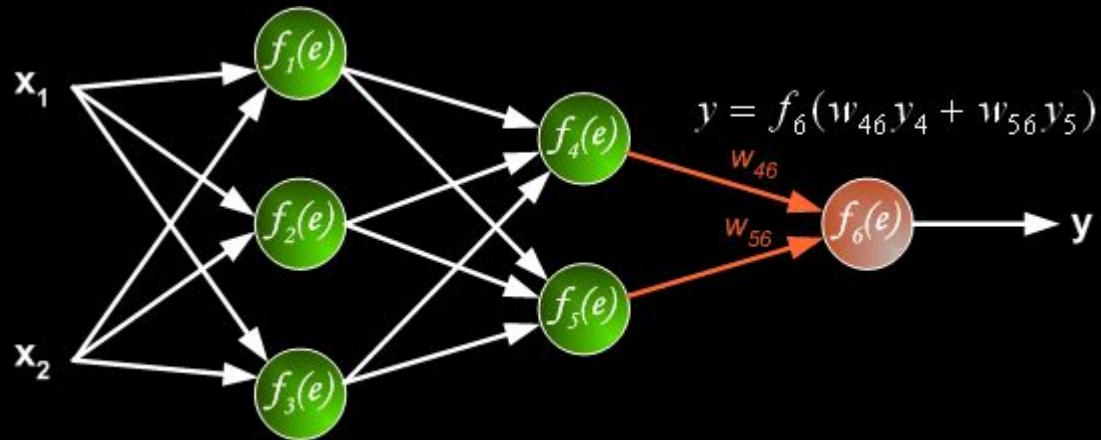
FORWARD PASS



FORWARD PASS



FORWARD PASS





BACK PROPAGATION

Compute
error



BACK PROPAGATION

DERIVATIVE CHAIN RULE

$$(f \circ g(x))' = g'(x)f'(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$$



BACK PROPAGATION

DERIVATIVE CHAIN RULE

$$(f \circ g(x))' = g'(x)f'(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du} \frac{du}{dx}$$

$$f(x) = \sin(5x) \quad f'(x) = 5[\cos(5x)]$$



BACK PROPAGATION

DERIVATIVE CHAIN RULE

$$x \xrightarrow{w} f(xw) \longrightarrow y \quad L = z - y$$



BACK PROPAGATION

DERIVATIVE CHAIN RULE

$$x \xrightarrow{w} f(xw) \longrightarrow y \quad L = z - y$$

$$\frac{\partial L}{\partial w}$$



BACK PROPAGATION

DERIVATIVE CHAIN RULE

$$x \xrightarrow{w} f(xw) \longrightarrow y \quad L = z - y$$

$$\frac{\partial L}{\partial w} = \frac{\partial f(xw)}{\partial w} \frac{\partial L}{\partial f(xw)}$$



BACK PROPAGATION

DERIVATIVE CHAIN RULE

$$x \xrightarrow{w} f(xw) \longrightarrow y \quad L = z - y$$

$$\frac{\partial L}{\partial w} = \frac{\partial f(xw)}{\partial w} \frac{\partial L}{\partial f(xw)}$$
$$\frac{\partial L}{\partial w} = \frac{\partial(xw)}{\partial w} \frac{\partial f(xw)}{\partial(xw)} \frac{\partial L}{\partial f(xw)}$$



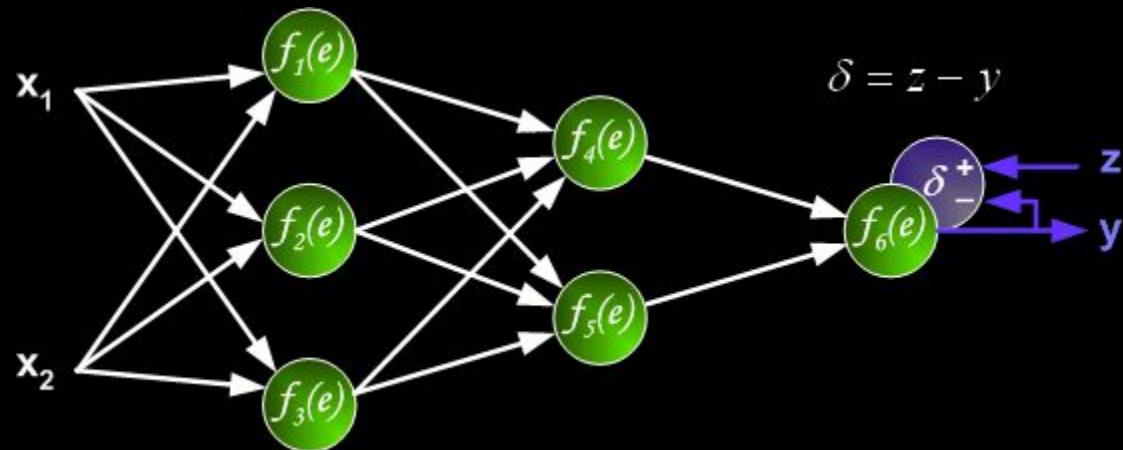
BACK PROPAGATION

DERIVATIVE CHAIN RULE

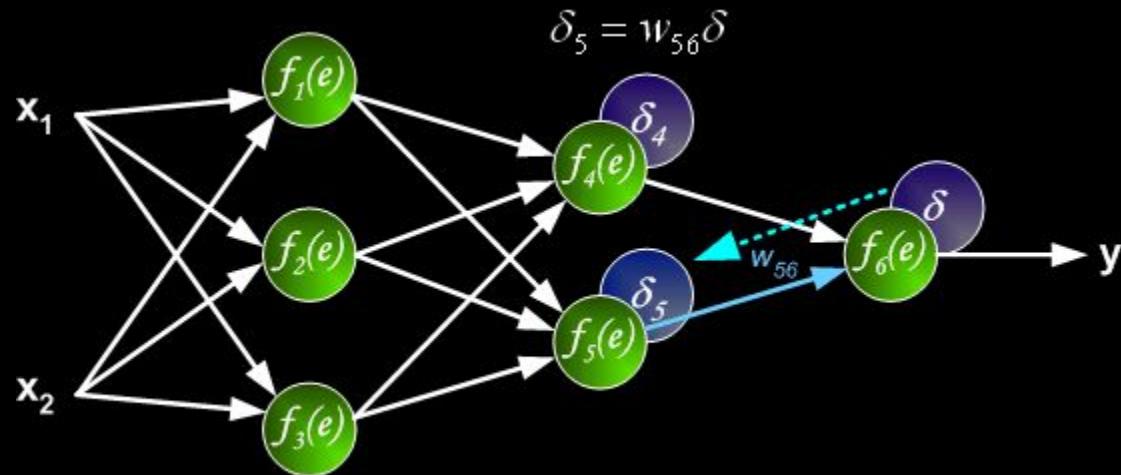
$$x \xrightarrow{w} f(xw) \longrightarrow y \quad L = z - y$$

$$\begin{aligned}\frac{\partial L}{\partial w} &= \frac{\partial f(xw)}{\partial w} \frac{\partial L}{\partial f(xw)} \\ \frac{\partial L}{\partial w} &= \frac{\partial(xw)}{\partial w} \frac{\partial f(xw)}{\partial(xw)} \frac{\partial L}{\partial f(xw)} \\ \frac{\partial L}{\partial w} &= x \frac{\partial f(e)}{\partial(e)} \delta\end{aligned}$$

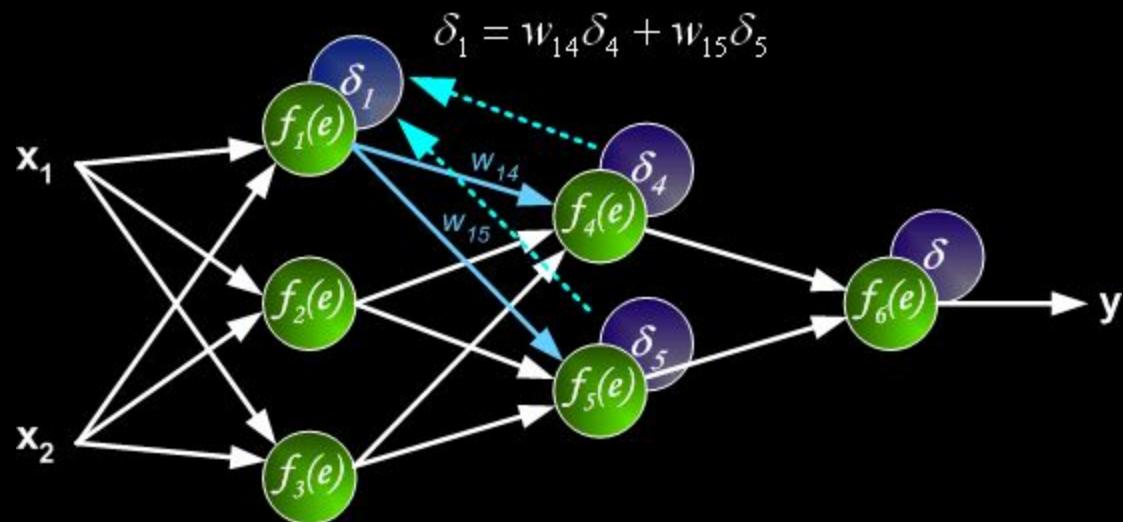
COMPUTE ERROR



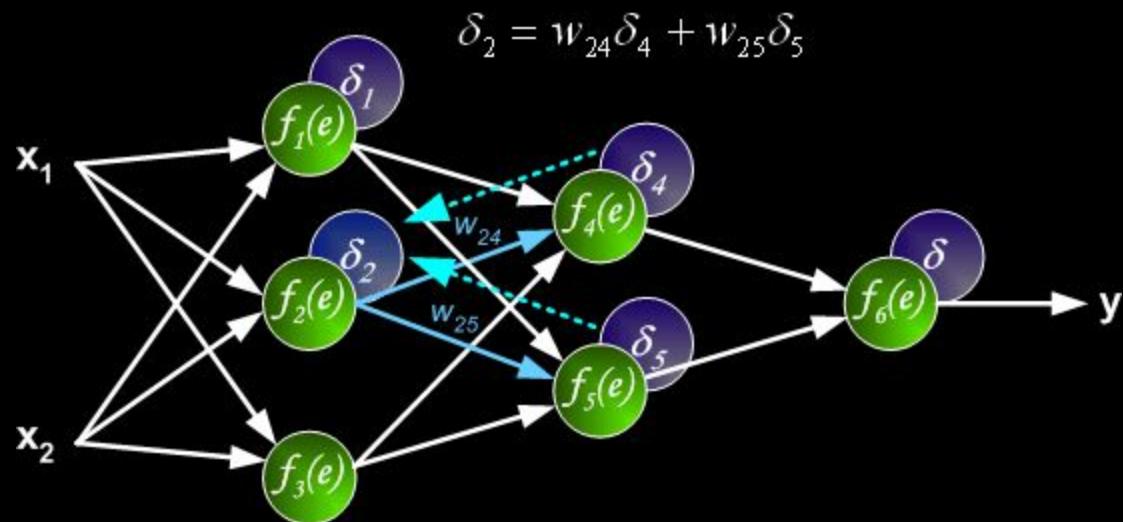
COMPUTE ERROR



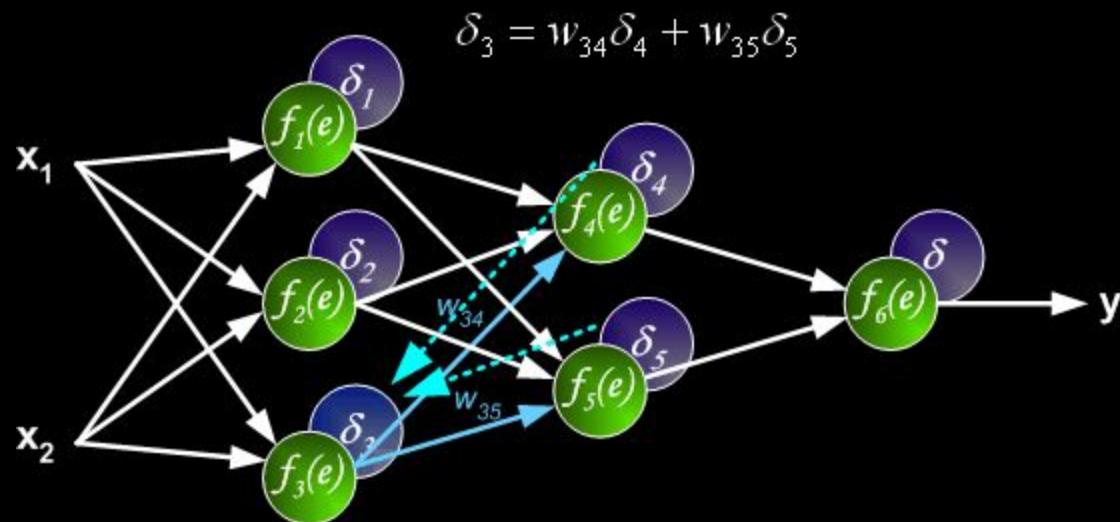
COMPUTE ERROR



COMPUTE ERROR



COMPUTE ERROR





BACK PROPAGATION

Update
weights



BACK PROPAGATION

$$\frac{\partial L}{\partial w} = x \frac{\partial f(e)}{\partial(e)} \delta$$

$$w = w - \eta \frac{\partial L}{\partial w}$$

Update
weights



BACK PROPAGATION

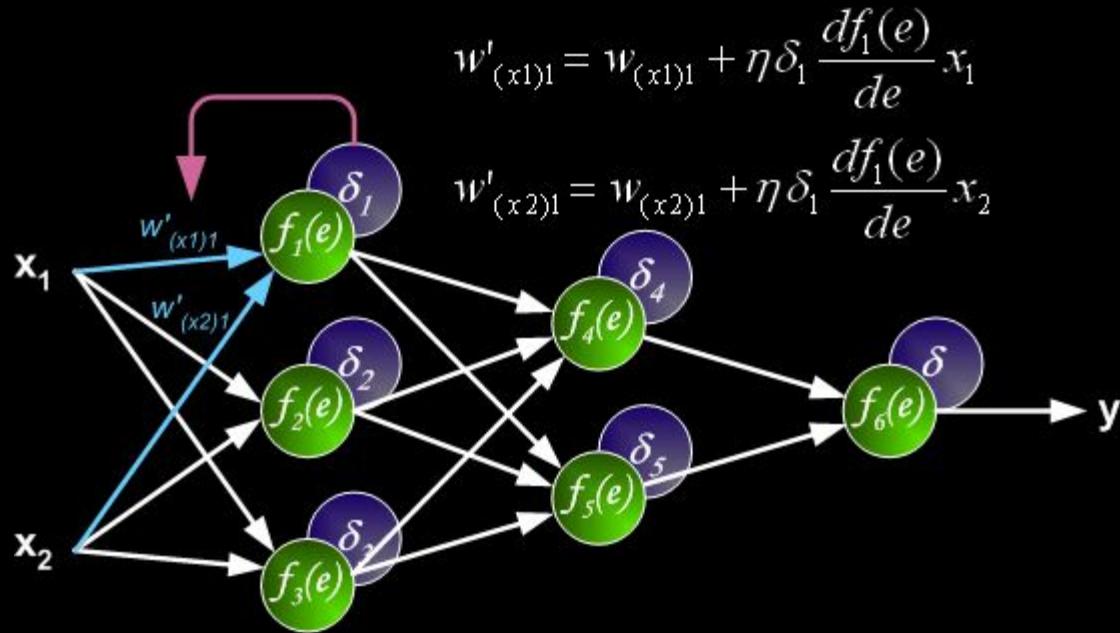
$$\frac{\partial L}{\partial w} = x \frac{\partial f(e)}{\partial(e)} \delta$$

$$w = w - \eta \frac{\partial L}{\partial w}$$

Update
weights

$$w = w - \eta x \frac{\partial f(e)}{\partial(e)} \delta$$

UPDATE WEIGHTS

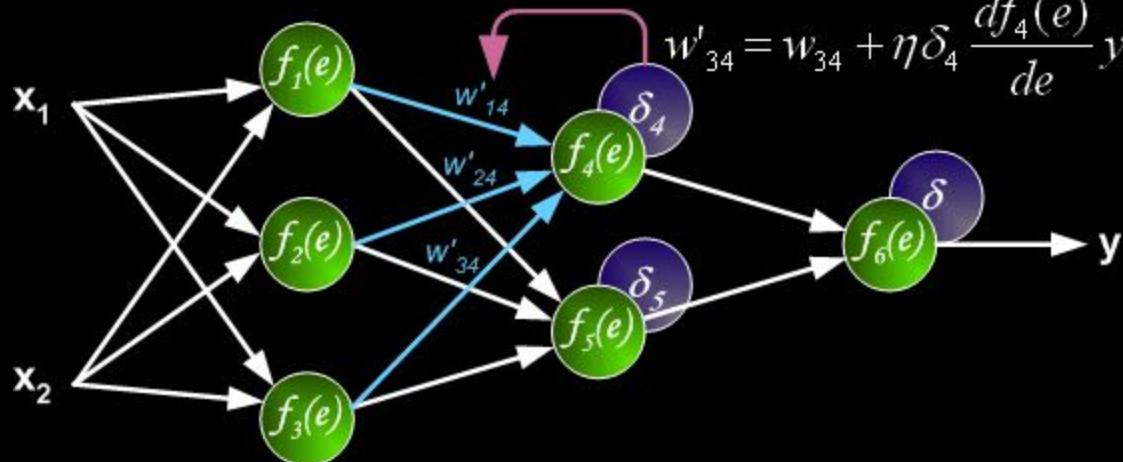


UPDATE WEIGHTS

$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$

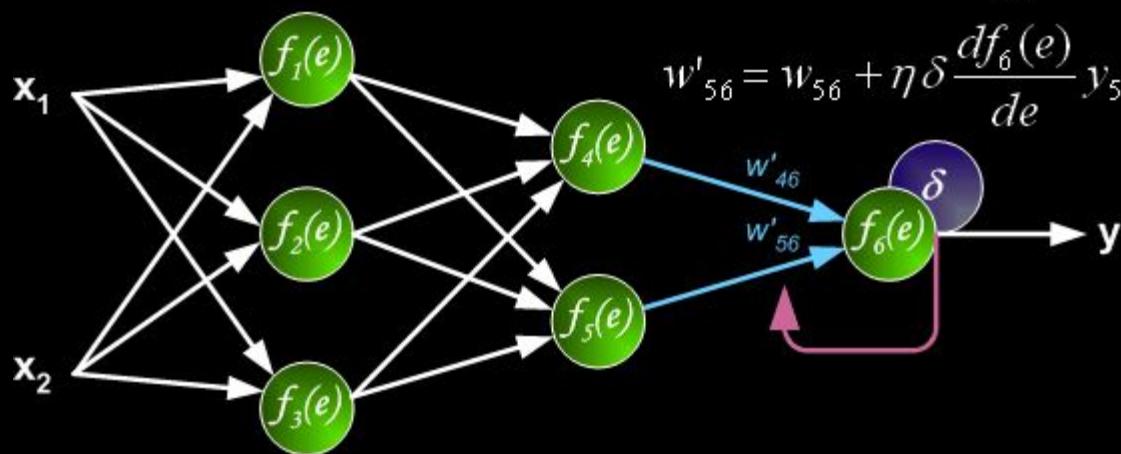
$$w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3$$



UPDATE WEIGHTS

$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$



SUMMARY

1. RANDOM WEIGHTS
2. FORWARD PASS
3. COMPUTE COST
4. BACKWARD PASS
5. UPDATE WEIGHTS

...

N. PROFIT???

SUMMARY

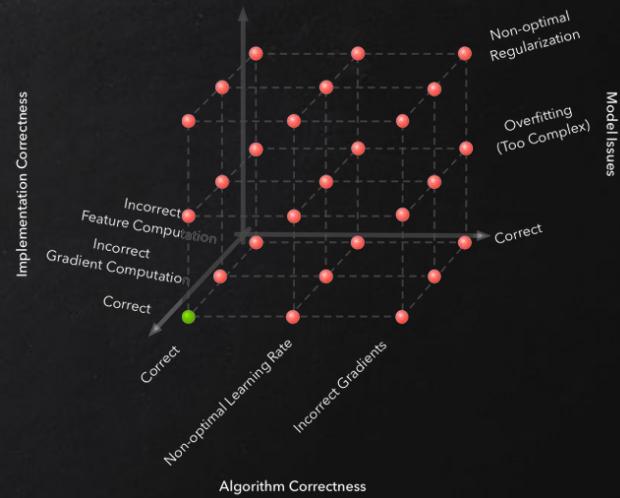
NOT SO FAST



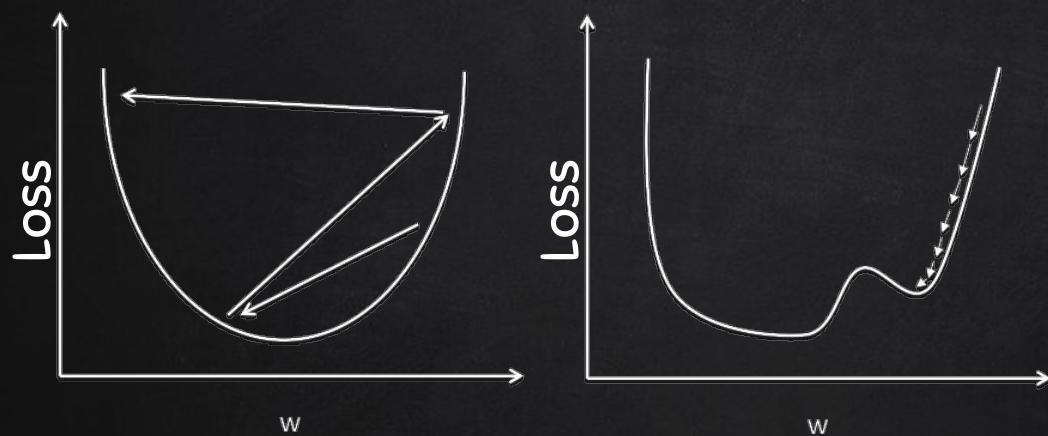
1. RANDOM WEIGHTS
2. FORWARD PASS
3. COMPUTE COST
4. BACKWARD PASS
5. UPDATE WEIGHTS

...

N. PROFIT???



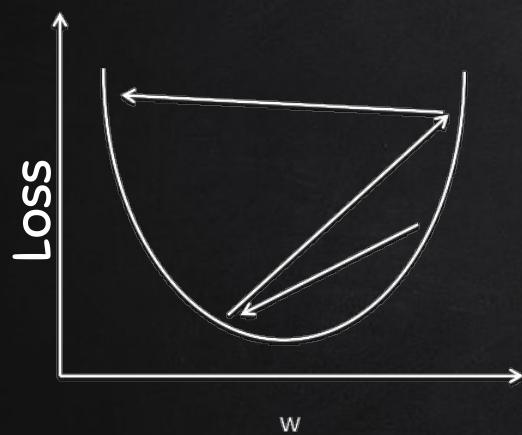
LEARNING RATE



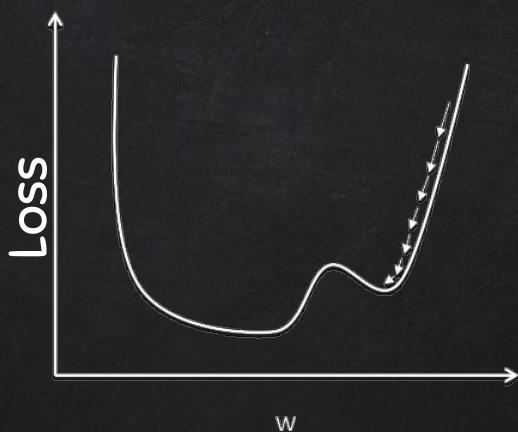
Large learning rate: Overshooting.

Small learning rate: Many iterations
until convergence and trapping in
local minima.

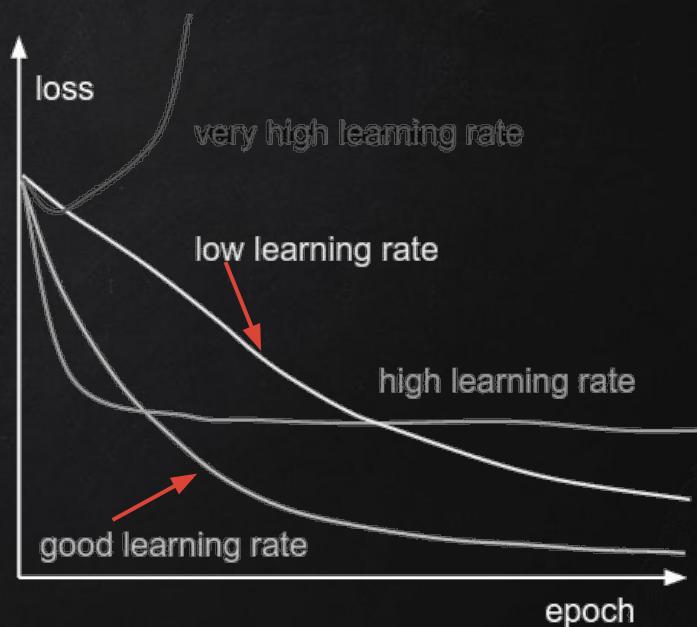
LEARNING RATE



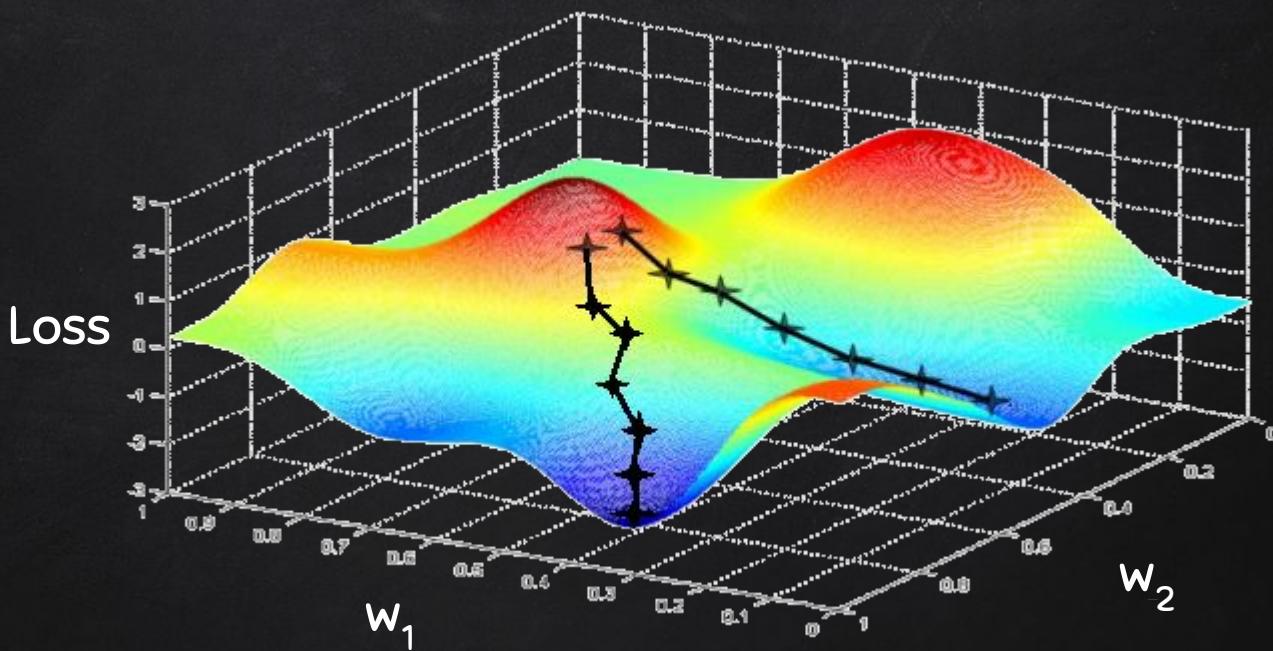
Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.



GRADIENT DESCENT



GRADIENT DESCENT



$$\Delta W = \eta \frac{1}{N} \sum \delta W$$

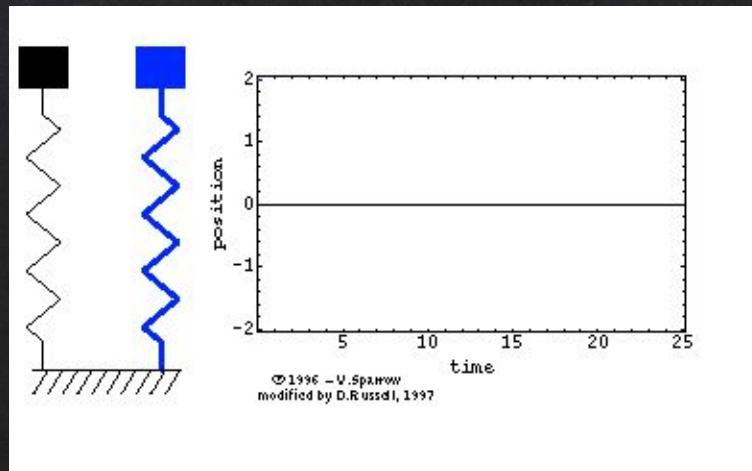
STOCHASTIC GRADIENT DESCENT



Mini-batch weight
update #1

Mini-batch weight
update #2

STOCHASTIC GRADIENT DESCENT WITH MOMENTUM



Demo: Why Momentum really works?

CHOICES

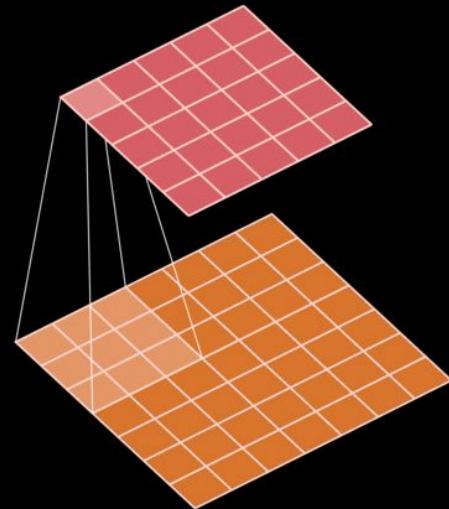
INITIALIZATIONS	ACTIVATION	COST	OPTIMIZERS
- UNIFORM	- TANH	- CROSS	- SGD
- GAUSSIAN	- SIGMOID	- ENTROPY	- MOMENTUM
- TRUNNORM	- RELU	- MSE	- RMSPROP
- GLOROTUNI	- PRELU	- L1 LOSS	- ADAGRAD
- XAVIER	- LRELU	- KL DIV	- ADADELTA
- KAIMING	- SWISH	- JS DIV	- ADAM

3.

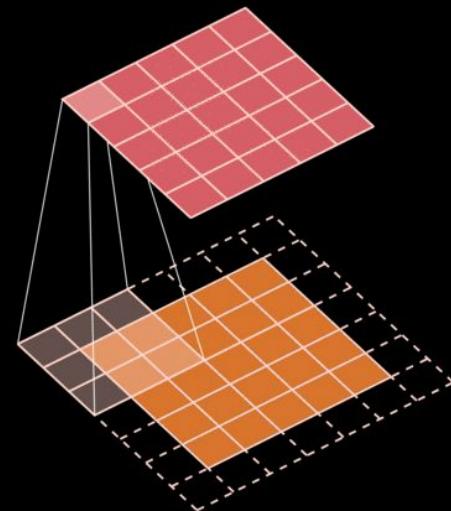
CONVOLUTION NETWORKS

Demo Convolution

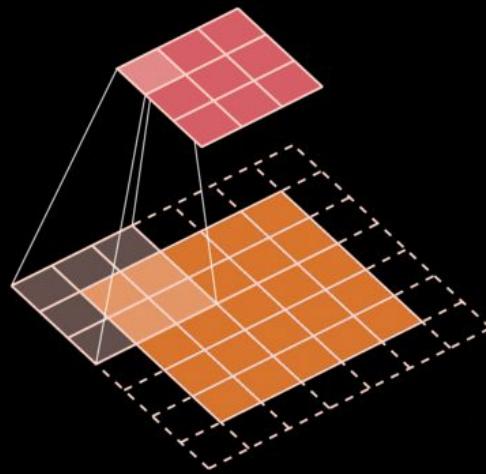
CONVOLUTION



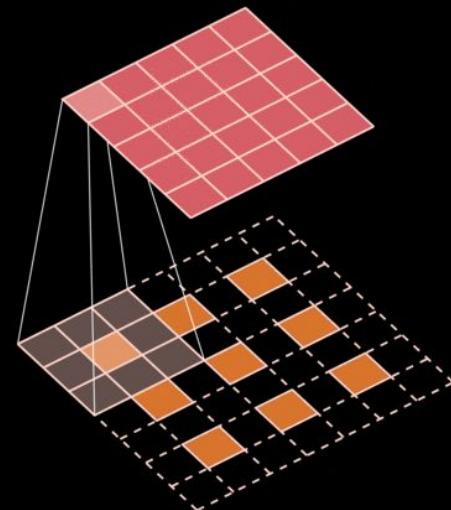
CONVOLUTION WITH PADDING



CONVOLUTION WITH PADDING + STRIDES



DECONVOLUTION / CONVOLUTION TRANSPOSED



MAX POOL

0	1	4	9
3	2	5	8
1	2	3	1
3	1	7	4

Size: 2x2, Stride: 2



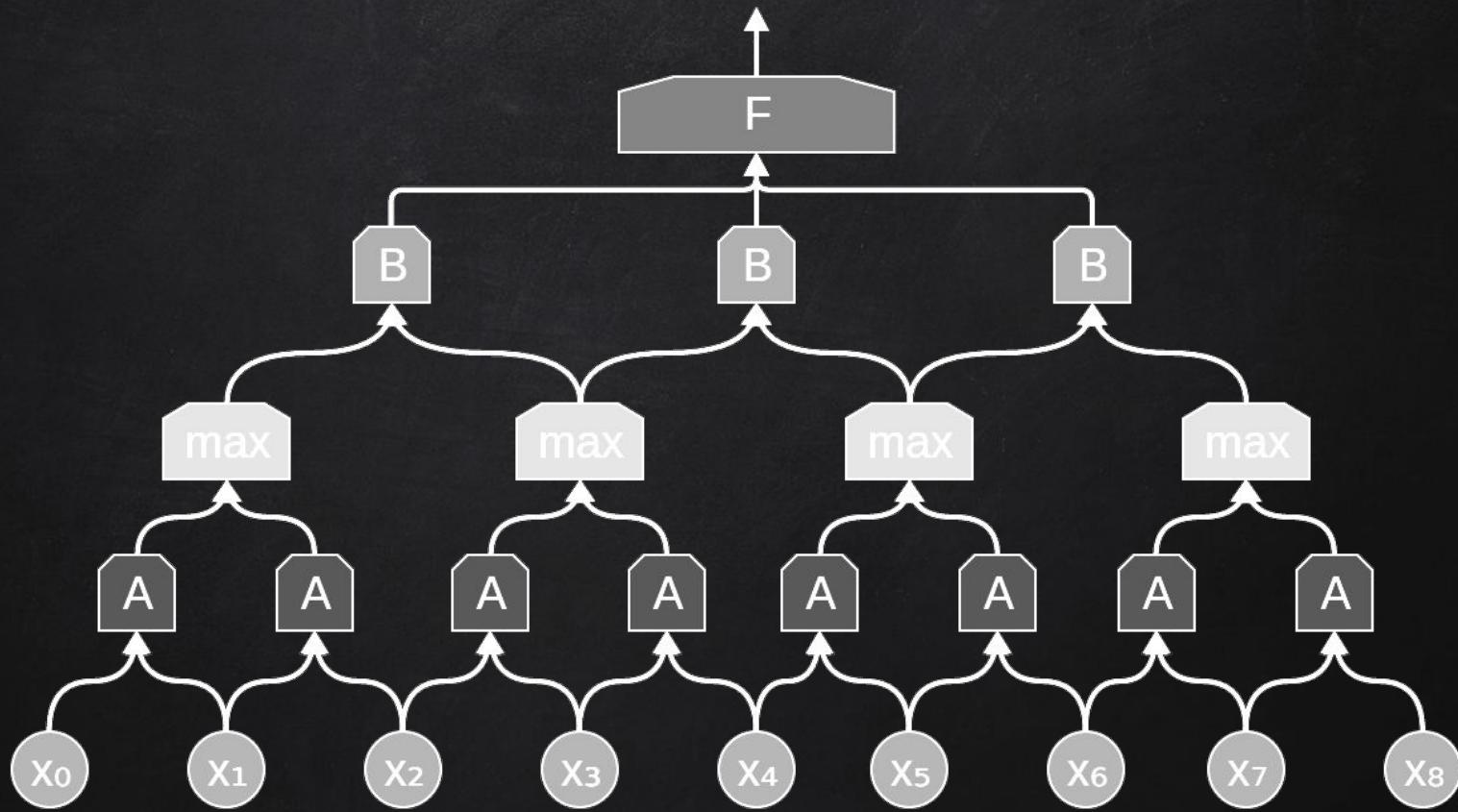
3	9
3	7

GEOFF HINTON ON POOLING----

THE POOLING OPERATION USED IN CONVOLUTIONAL NEURAL
NETWORKS IS A **BIG MISTAKE** AND THE FACT THAT IT WORKS SO
WELL IS A DISASTER.

IF THE POOLS DO NOT OVERLAP, POOLING LOSES VALUABLE INFORMATION ABOUT WHERE THINGS ARE. WE NEED THIS
INFORMATION TO DETECT PRECISE RELATIONSHIPS BETWEEN THE PARTS OF AN OBJECT.

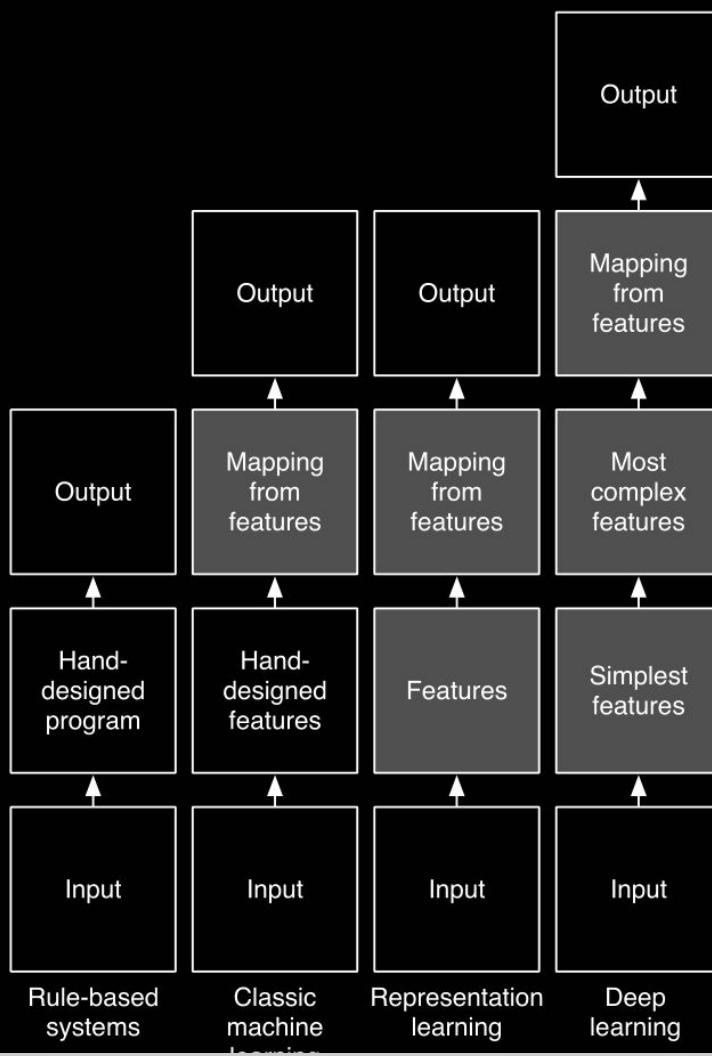
CONVOLUTIONAL NEURAL NETWORK

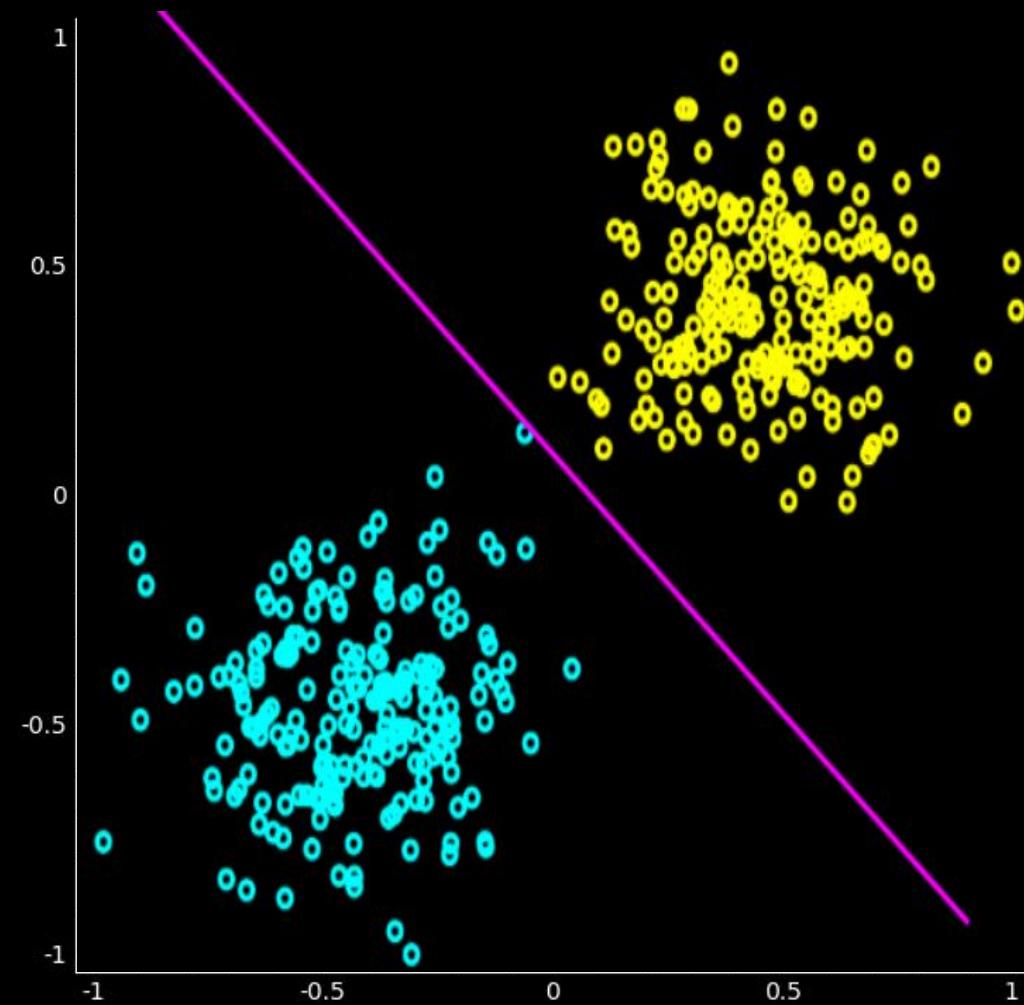


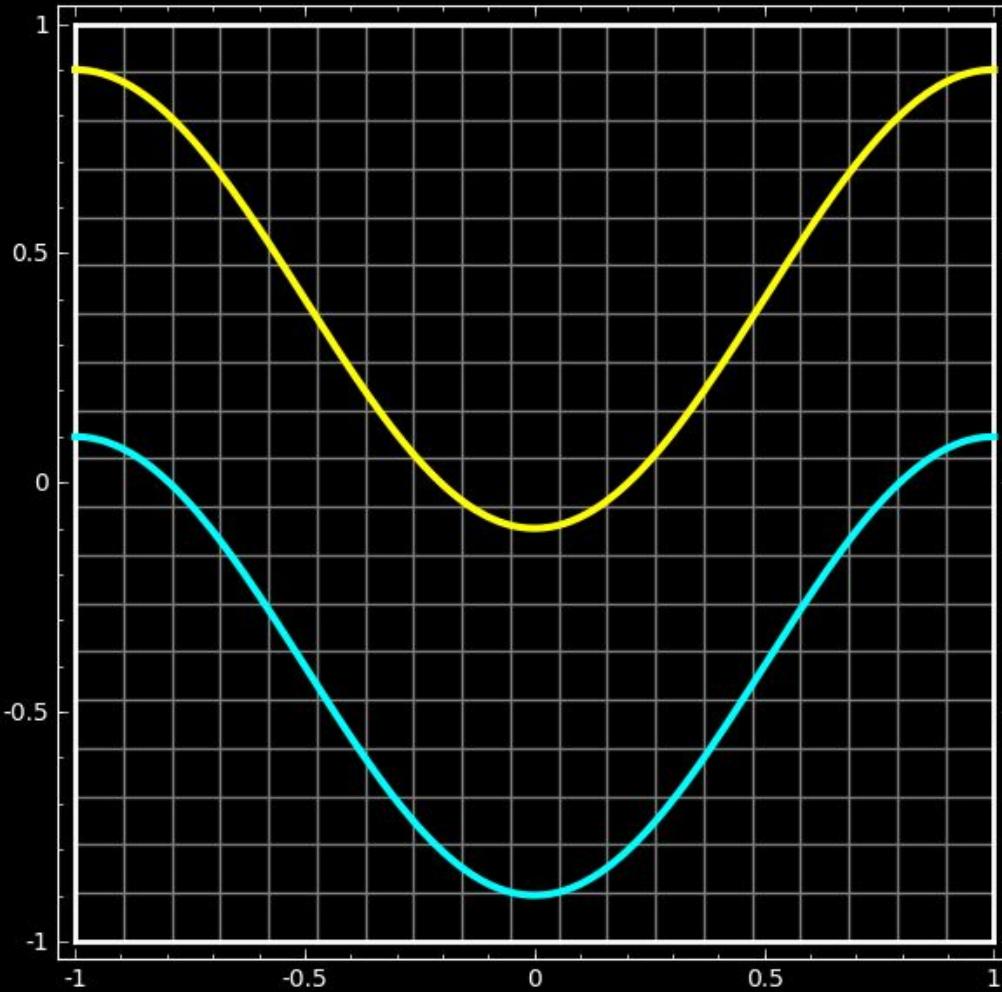


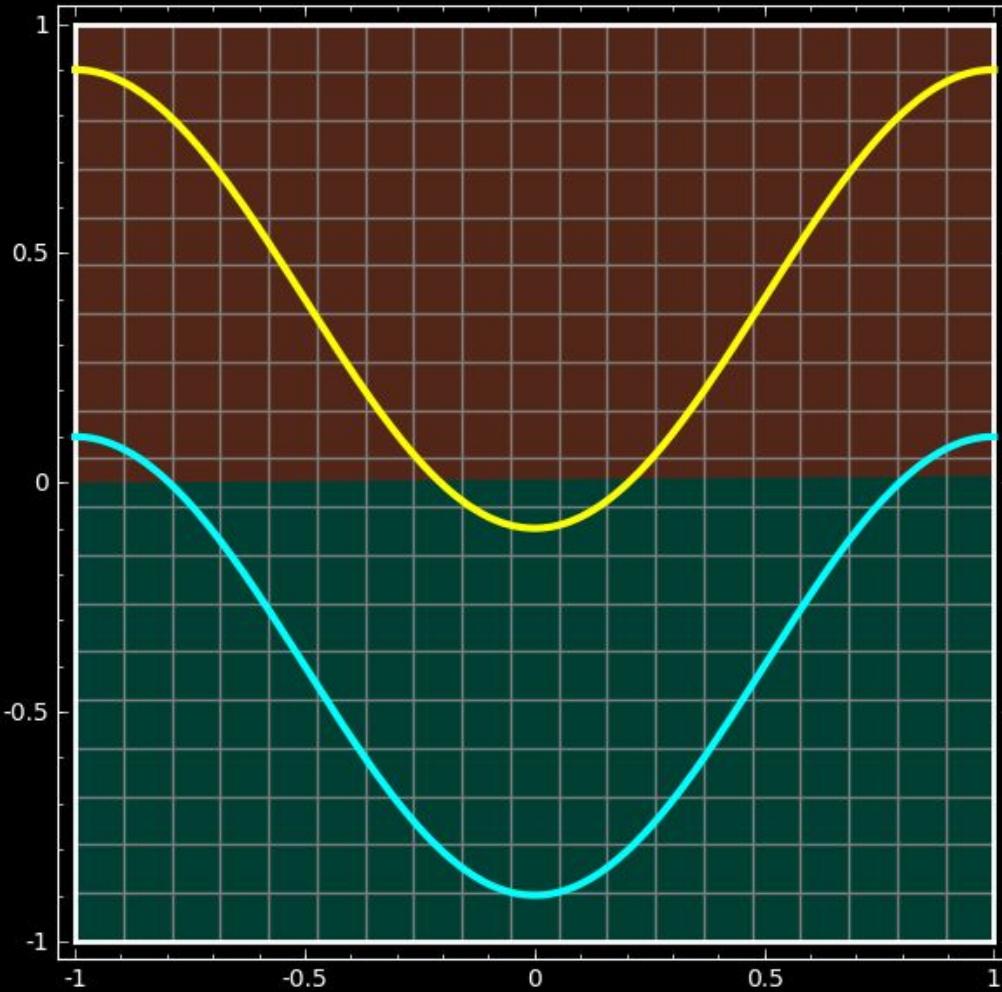
But what do the layers do?

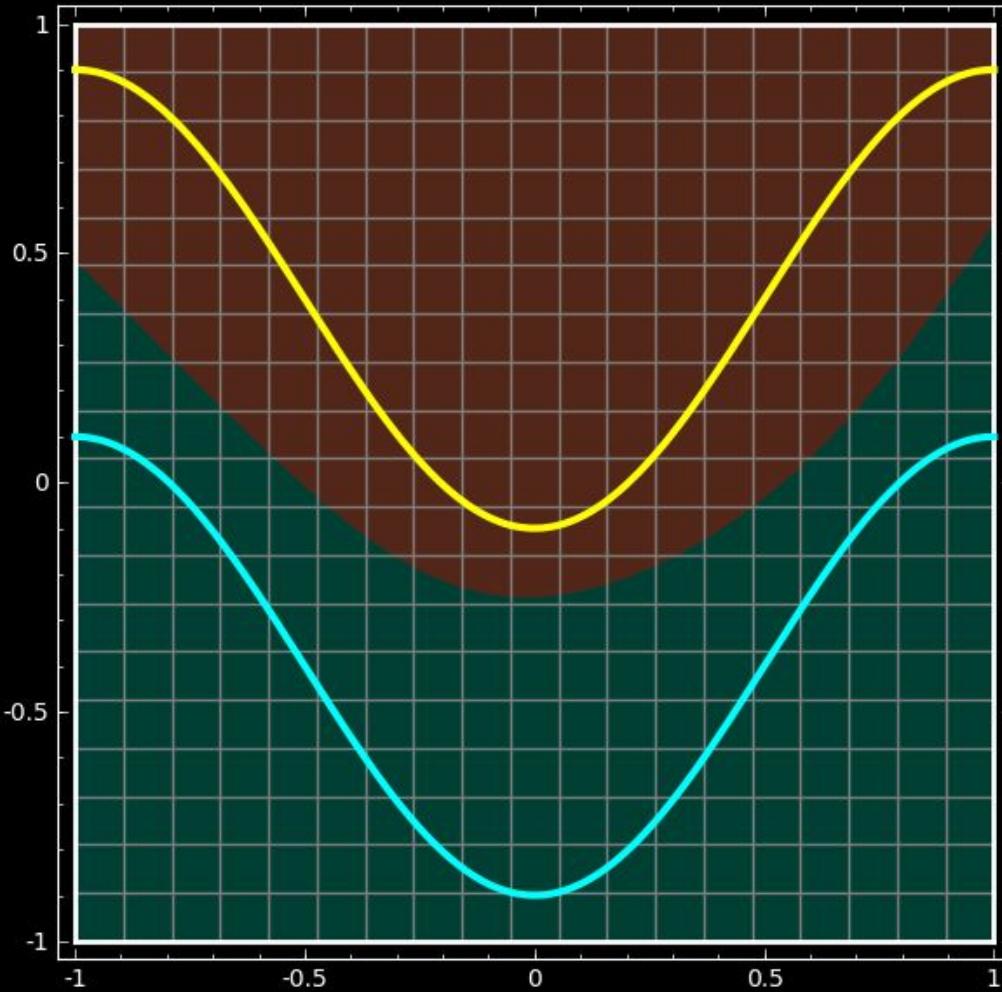
How many layers do I need?

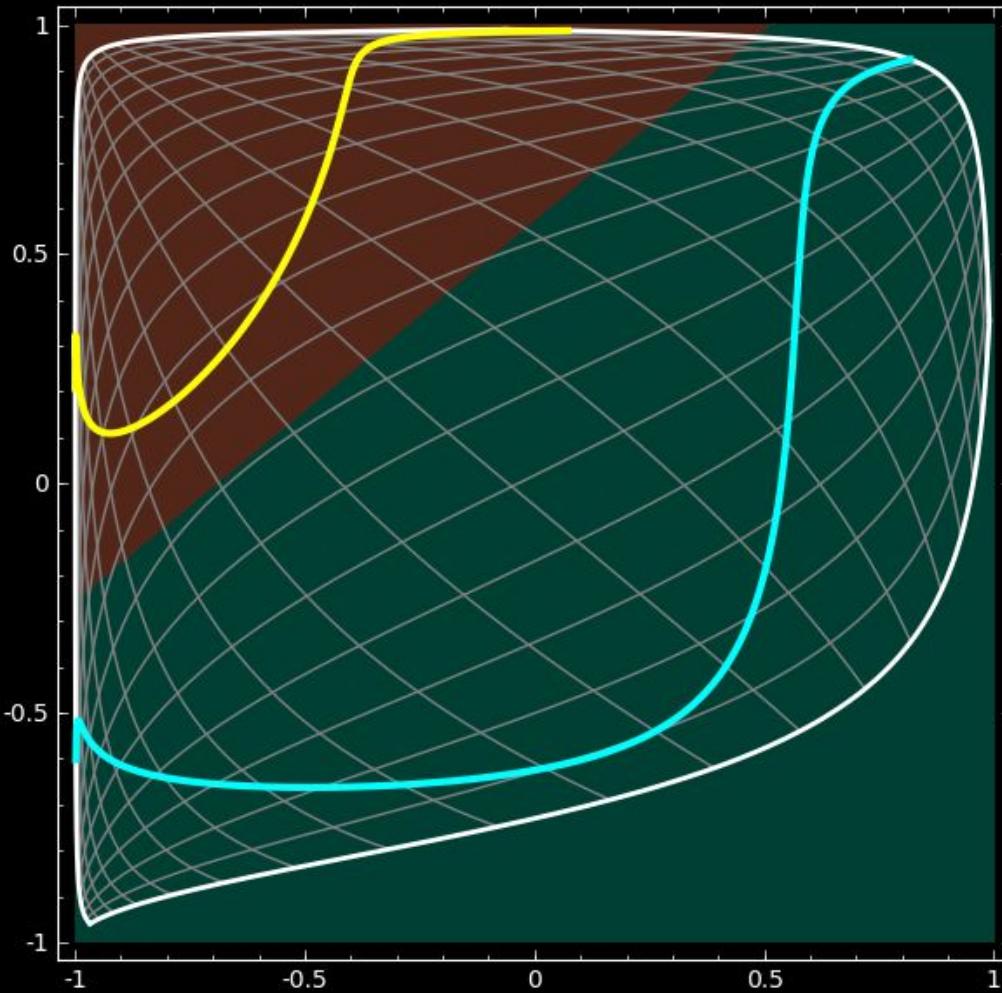






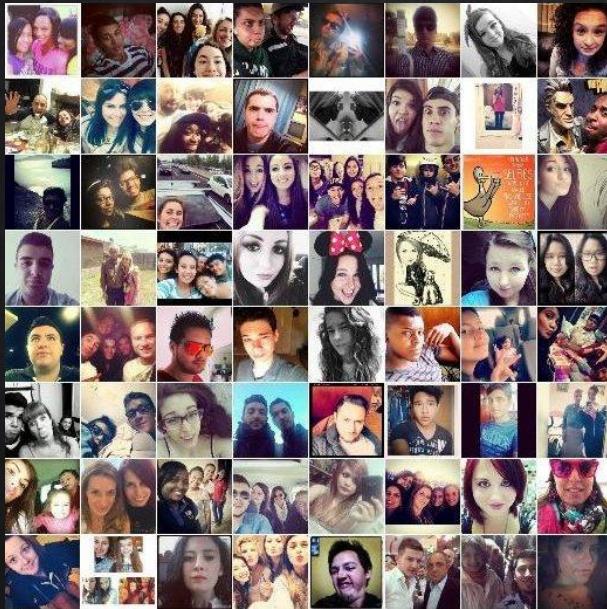






SELFIE !

<= OR =>



Any questions?

You can find me at

iamaaditya.github.io

aprakash@brandeis.edu



THANKS!

CREDITS

- ✗ Presentation template by [SlidesCarnival](#)
- ✗ Awesome diagrams on CNN by <http://colah.github.io/>
- ✗ Formulas from, you know where, [wikipedia](#)
- ✗ Selfie <http://karpathy.github.io/2015/10/25/selfie/>
- ✗ [Neural Networks and Deep Learning](#)
- ✗ [Going Deeper – GoogLeNet](#)
- ✗ Back propagation images with formulas, [Mariusz Bernacki](#)
- ✗ Geoff Hinton, guy who started all this -- [NIPS 2015 tutorial](#)
- ✗ Yann LeCun, guy who made it popular, [Deep Representations](#)
- ✗ [CS231n](#) Lecture Slides
- ✗ Nervana presentation on [DL](#)

Deep Learning



What society thinks I do



What my friends think I do



What other computer



What mathematicians think I do



What I think I do

FROM TENSORFLOW
IMPORT *

What I actually do