What is artificial intelligence?

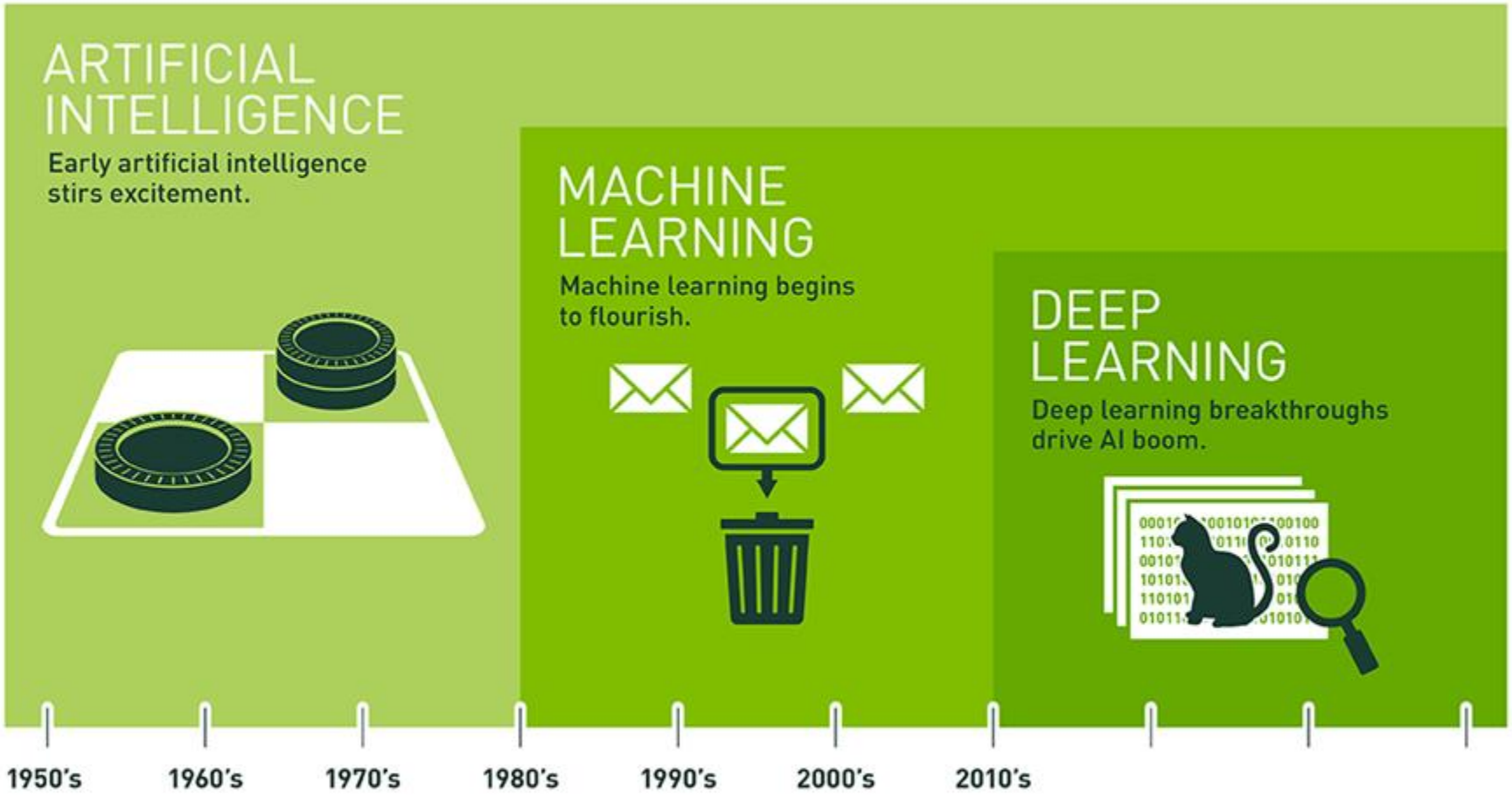
Artificial intelligence is the ability of a computer to perform tasks commonly associated with intelligent beings.

What is machine learning?

Machine learning is the study of algorithms that learn from examples and experience instead of relying on hard-coded rules and make predictions on new data.

What is deep learning?

Deep learning is a subfield of machine learning focusing on learning data representations as successive layers of increasingly meaningful representations.



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Image from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

“Traditional” machine learning:

handcrafted features → learned classifier → **cat**

Deep, “end-to-end” learning:

learned low-level features → learned mid-level features → learned high-level features → learned classifier → **cat**

Why Deep Learning Now?

- Big Data
  - Larger Datasets
  - Easier Collection and Storage
- Hardware
  - GPUs
  - Massively Parallelizable Training
- Software
  - Improved Techniques
  - New Models
  - Toolboxes

IMAGENET

+

GPU and TPU

+

PYTORCH

Microsoft CNTK

Caffe2

dmlc mxnet

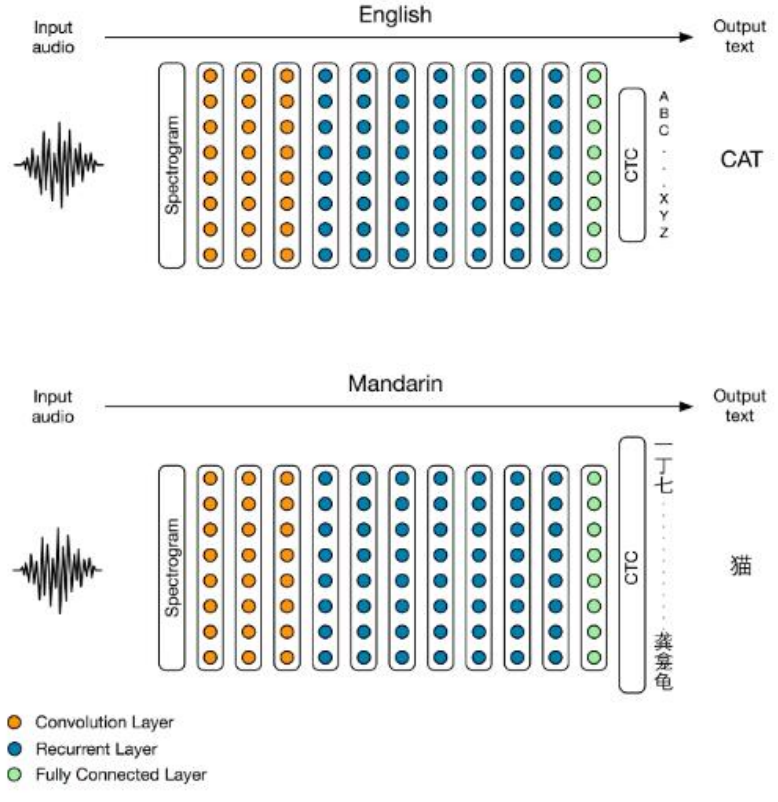
gensim

spaCy

theano


Deep
Learning
BOOM!!!

DL Today: Speech-to-Text

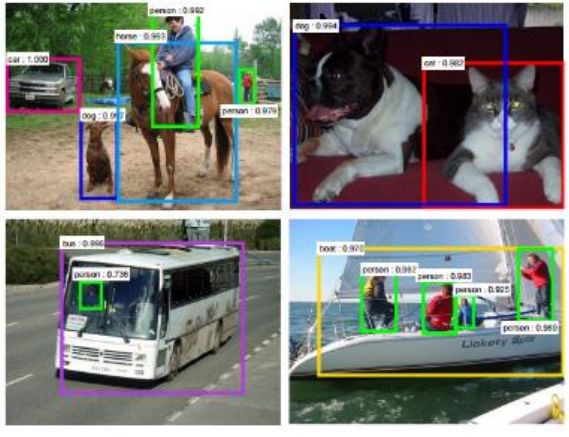


[Baidu 2014]

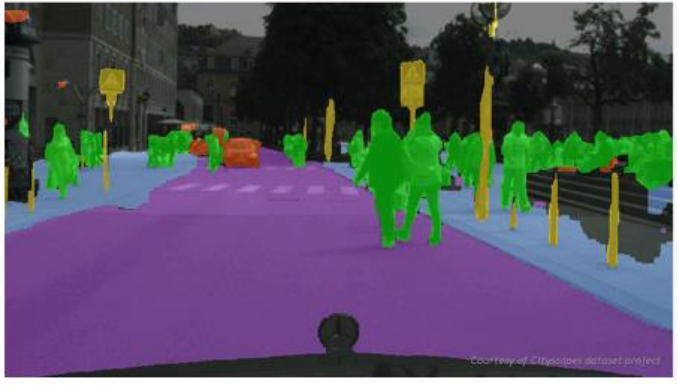
DL Today: Vision

[Krizhevsky 2012]

[Ciresan et al. 2013]



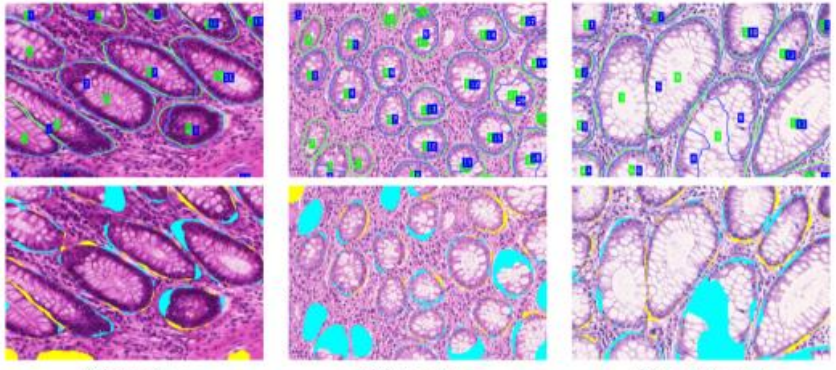
[Faster R-CNN - Ren 2015]



[NVIDIA dev blog]

DL Today: Vision

[Stanford 2017]



[Nvidia Dev Blog 2017]

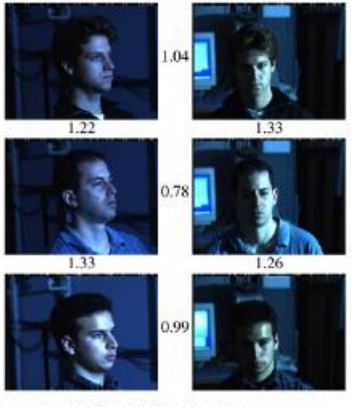
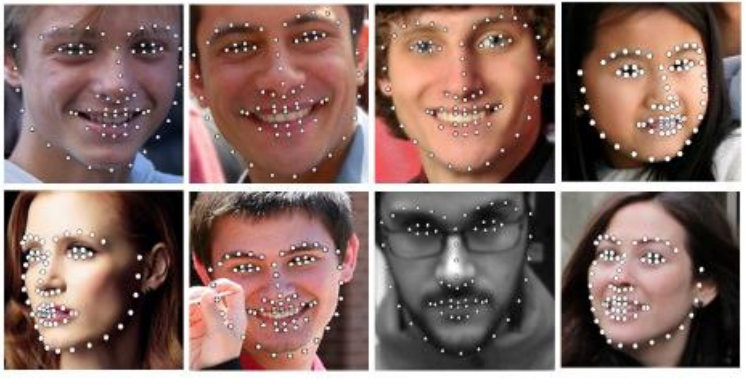


Figure 1. **Illumination and Pose invariance.**

[FaceNet - Google 2015]



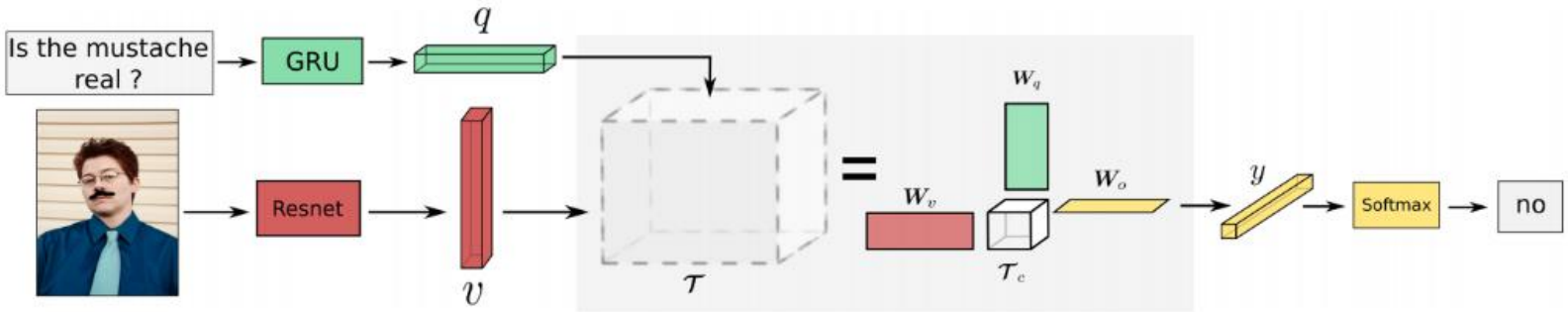
[Facial landmark detection CUHK 2014]

DL Today: NLP

[Google Inbox Smart Reply]

[Amazon Echo / Alexa]

DL Today: Vision + NLP



[VQA - Mutan 2017]

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

[Karpathy 2015]

DL Today: Generative models

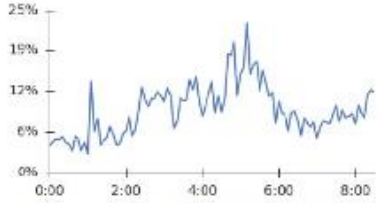
Sampled celebrities [Nvidia 2017]

StackGAN v2 [Zhang 2017]

DL for AI in games

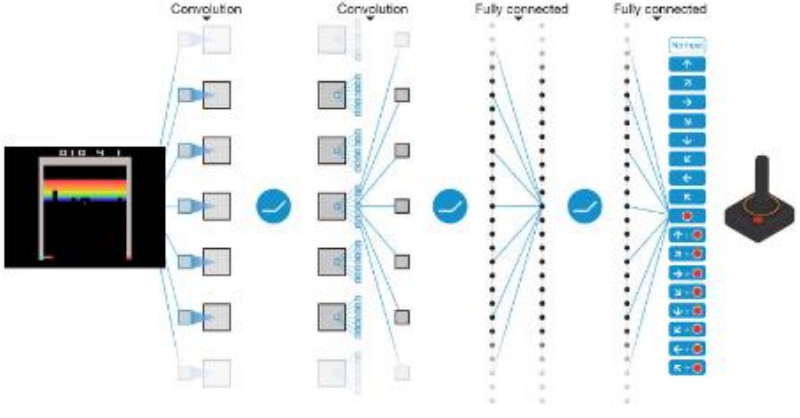
w 20/30/0, b 8/40/2

A10: English Opening



1...e5 g3 d5 cxd5 ♘f6 ♗g2 ♘xd5 ♘f3

[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]

[Starcraft 2 for AI research]

AlphaFold – Protein Structure Prediction

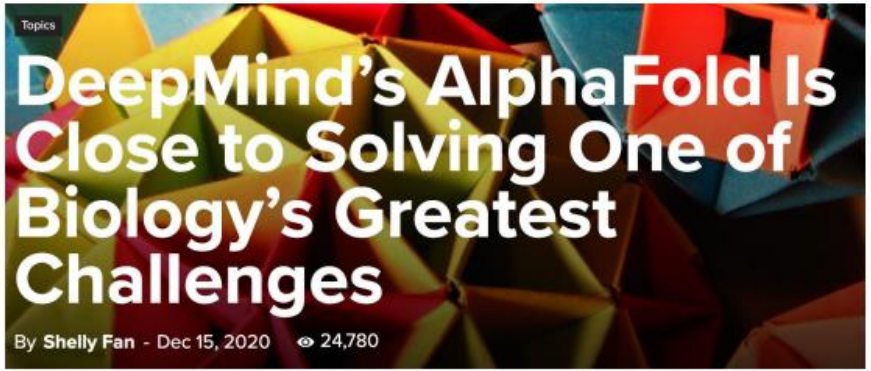


Image credit: SingularityHub

AlphaFold's AI could change the world of biological science as we know it

DeepMind's latest AI breakthrough can accurately predict the way proteins fold

Has Artificial Intelligence 'Solved' Biology's Protein-Folding Problem?

12-14-20

DeepMind's latest AI breakthrough could turbocharge drug discovery

ChatGPT - Large Language Models

Microsoft Confirms Its $10 Billion Investment Into ChatGPT, Changing How Microsoft Competes With Google, Apple And Other Tech Giants

generate python code to add two arrays and then reverse the second half of resultant array

Sure, here's some Python code that should do what you're asking for:

python Copy code

```python
import numpy as np

# Define two arrays
array1 = np.array([1, 2, 3, 4])
array2 = np.array([5, 6, 7, 8])

# Add the two arrays
result = array1 + array2

# Reverse the second half of the result array
mid = len(result) // 2
result[mid:] = result[mid:][::-1]

# Print the final result
print(result)
```

World

ChatGPT breaks record with 100 million users – and investors come flocking

Bianke Neethling • 15 Feb 2023

write twinkle twinkle little stars for PhD students

Sure, here's my take on "Twinkle Twinkle Little Star" for PhD students:

Verse 1:
Research, research, day and night
In the lab or desk in sight
Coding, writing, always learning
Data sets and papers churning

Chorus:
PhD, PhD, you'll go far
Publish papers like little stars
Innovation, curiosity
PhD, your destiny

Verse 2:
Reading, analyzing, questioning
Theories tested, new things bring
In the field or at the screen
You're pursuing what's never been seen

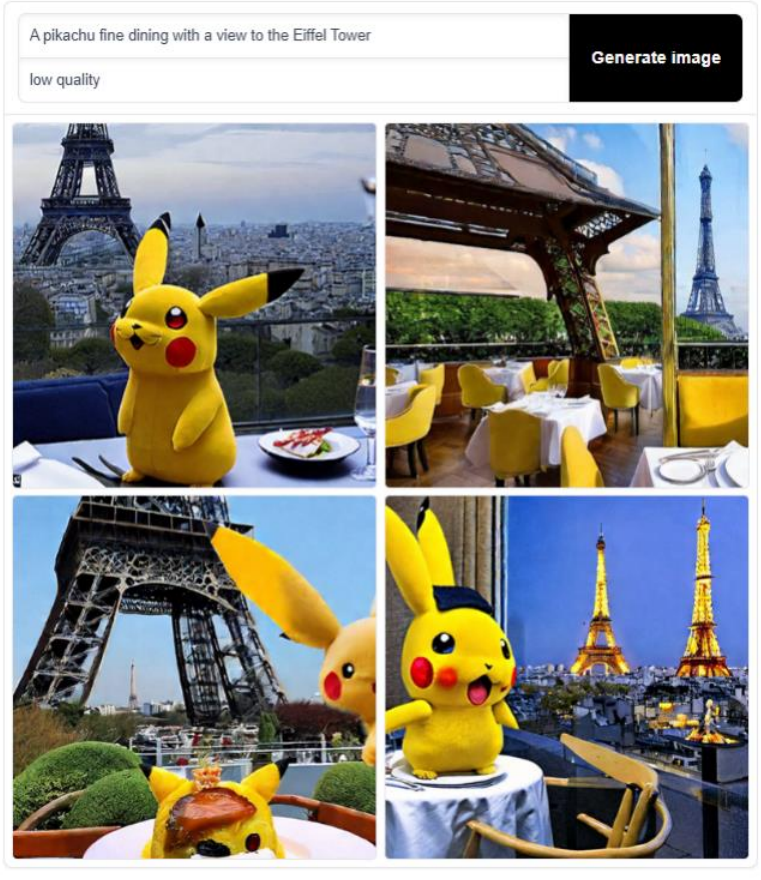
Chorus:
PhD, PhD, you'll go far
Publish papers like little stars
Innovation, curiosity
PhD, your destiny

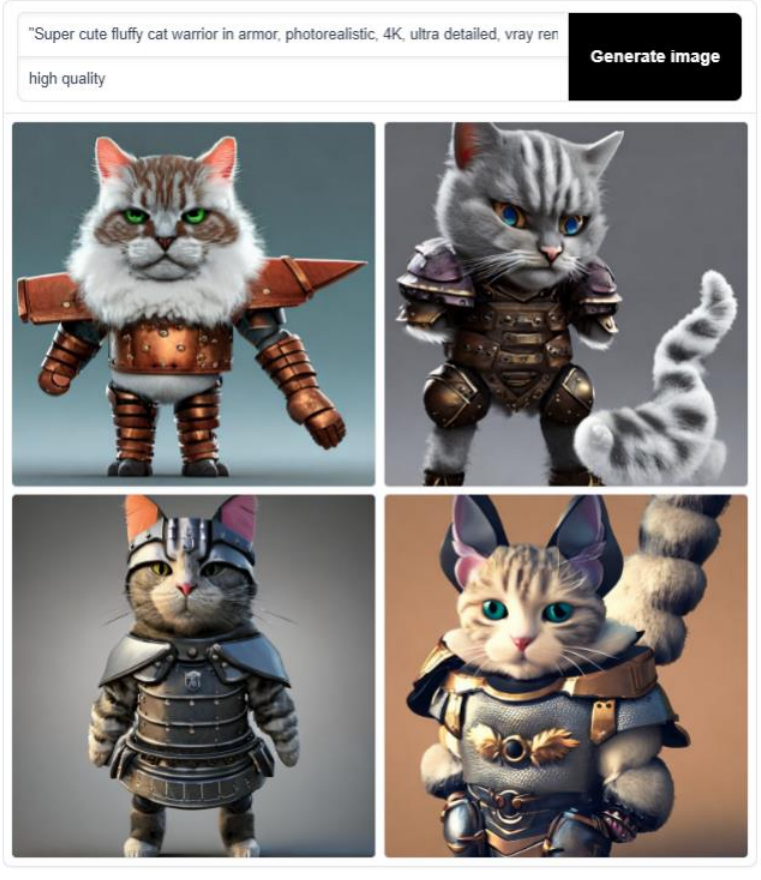
Verse 3:
Conferences, seminars, debates
Sharing knowledge, learning fates
With peers, advisors, mentors too
Collaboration helps you through

Regenerate response

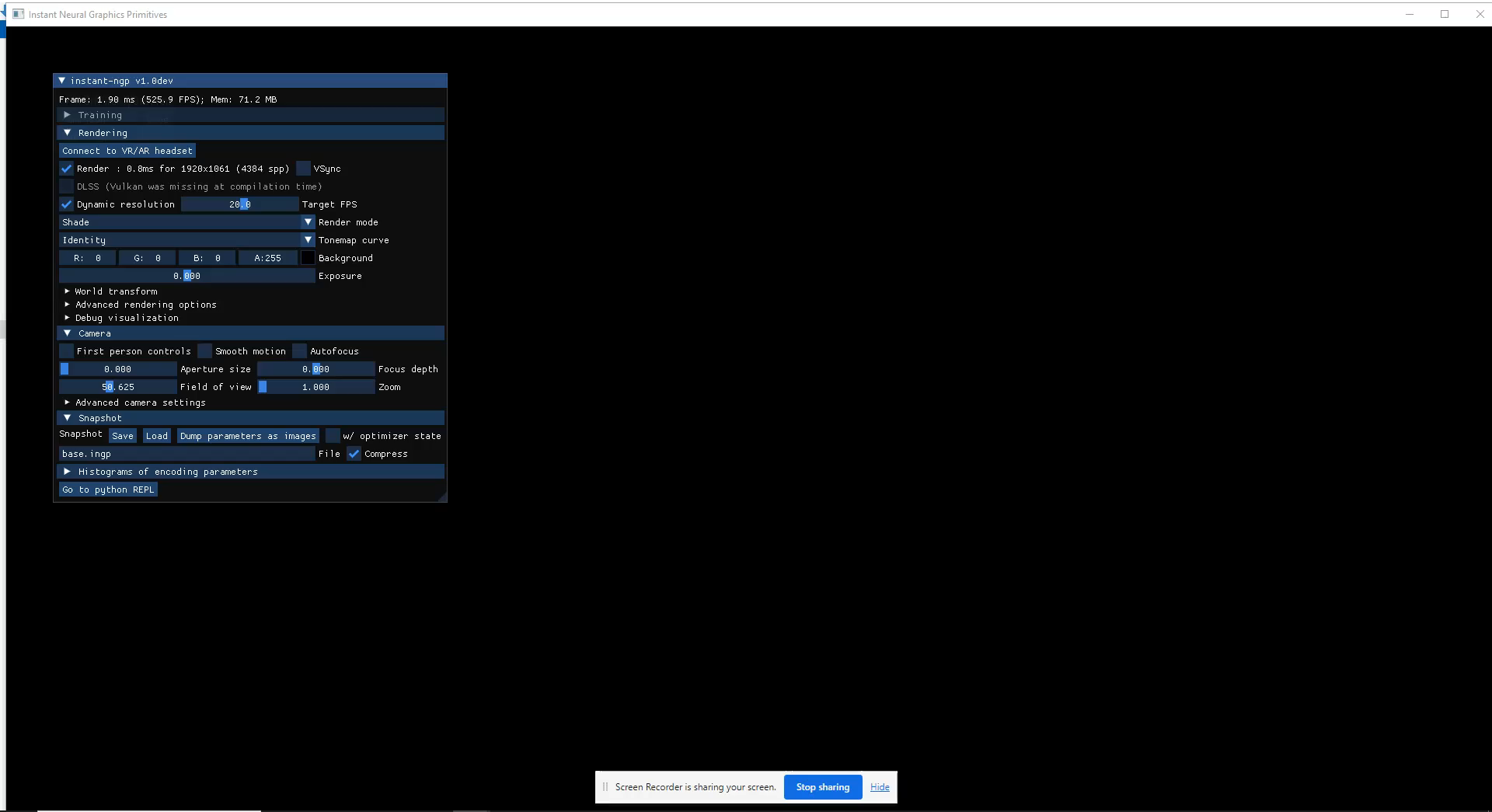
Google poised to release chatbot technology after ChatGPT success

Stable Diffusion - Generative Models





3D model generation



History of AI: Major Milestones

Table 1: Major milestones that will be covered in this paper

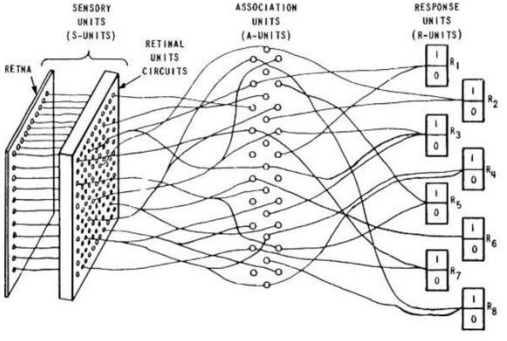
Year	Contributer	Contribution
300 BC	Aristotle	introduced Associationism, started the history of human's attempt to understand brain.
1873	Alexander Bain	introduced Neural Groupings as the earliest models of neural network, inspired Hebbian Learning Rule.
1943	McCulloch & Pitts	introduced MCP Model, which is considered as the ancestor of Artificial Neural Model.
1949	Donald Hebb	considered as the father of neural networks, introduced Hebbian Learning Rule, which lays the foundation of modern neural network.
1958	Frank Rosenblatt	introduced the first perceptron, which highly resembles modern perceptron.
1974	Paul Werbos	introduced Backpropagation
1980	Teuvo Kohonen	introduced Self Organizing Map
	Kunihiko Fukushima	introduced Neocogitron, which inspired Convolutional Neural Network
1982	John Hopfield	introduced Hopfield Network
1985	Hilton & Sejnowski	introduced Boltzmann Machine
1986	Paul Smolensky	introduced Harmonium, which is later known as Restricted Boltzmann Machine
	Michael I. Jordan	defined and introduced Recurrent Neural Network
1990	Yann LeCun	introduced LeNet, showed the possibility of deep neural networks in practice
1997	Schuster & Paliwal	introduced Bidirectional Recurrent Neural Network
	Hochreiter & Schmidhuber	introduced LSTM, solved the problem of vanishing gradient in recurrent neural networks
2006	Geoffrey Hinton	introduced Deep Belief Networks, also introduced layer-wise pretraining technique, opened current deep learning era.
2009	Salakhutdinov & Hinton	introduced Deep Boltzmann Machines
2012	Geoffrey Hinton	introduced Dropout, an efficient way of training neural networks

From: Wang & Raj: On the Origin of Deep Learning (2017)

History of AI: Major Milestones

Dartmouth AI Conference 1956

Early neural networks



Perceptron, one of the first neural network architectures

Rosenblatt 1957

F. Rosenblatt

1957

The Perceptron, 1957

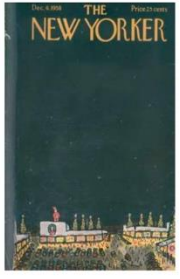
History of AI: Major Milestones

Early hype

"First serious rival to the human brain even devised."

"Remarkable machine capable of what amounts to thought"

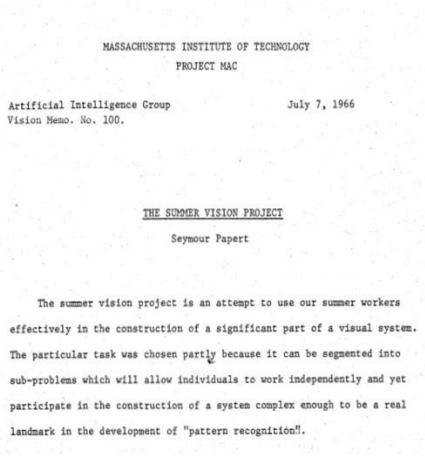
— The New Yorker



1958

Manson, Stewart, Gill 1958

Early hype



MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group
Vision Memo. No. 100.

July 7, 1966

THE SUMMER VISION PROJECT

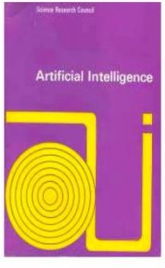
Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

Papert 1966

The Lighthill Report

"Most workers in AI research and in related fields confess to a pronounced feeling of disappointment in what has been achieved in the past twenty-five years. […] In no part of the field have the discoveries made so far produced the major impact that was then promised."



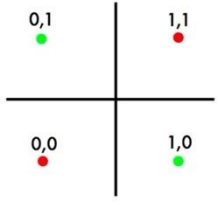
J. Lighthill

1972

Lighthill 1972

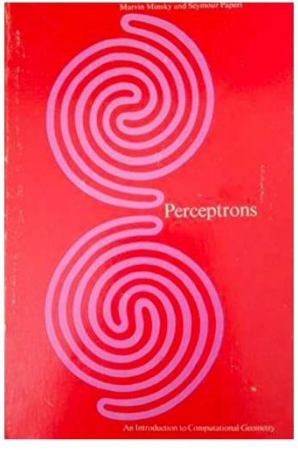
History of AI: Major Milestones

The "XOR Affair"



XOR

"[simple] perceptron cannot represent even the XOR function"



M. Minsky

S. Papert

1969

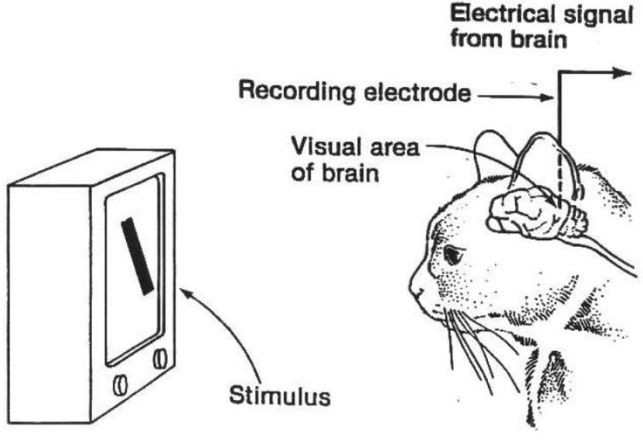
Minsky, Papert 1969

The XOR Affair, 1969

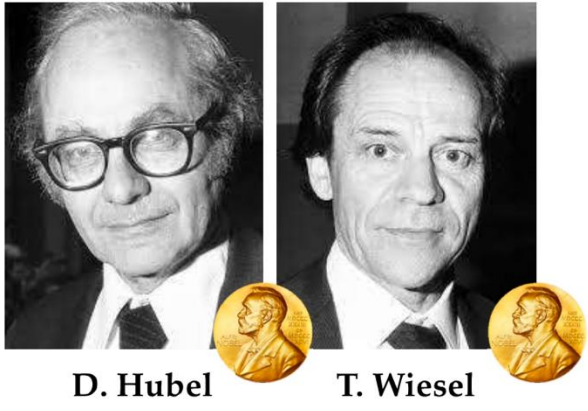
History of AI: Major Milestones

History of AI: Major Milestones

Secrets of the visual cortex



Experiments of Hubel and Wiesel that established the structure of the visual cortex



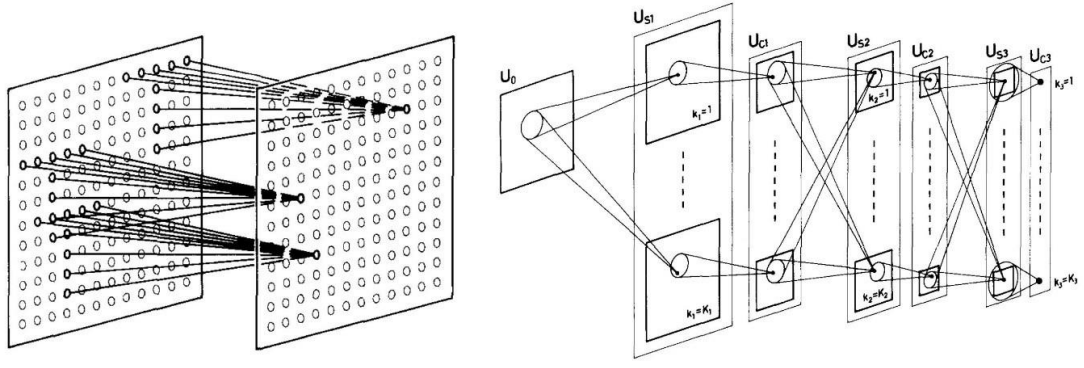
D. Hubel T. Wiesel

1959

Hubel, Wiesel 1959; 1962

History of AI: Major Milestones

Neocognitron



Neocognitron, an early geometric neural network

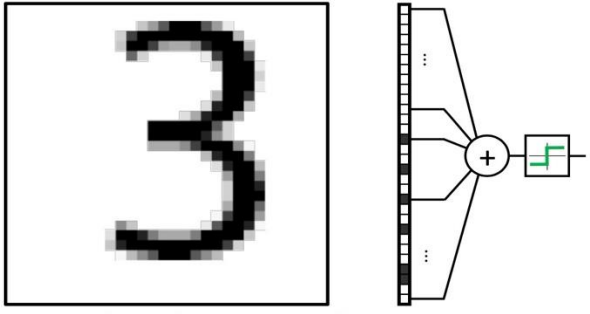
K. Fukushima

1980

Fukushima 1980

Neocognitron, 1980

History of AI: Major Milestones



input image input vector

"The response of [Perceptrons] was severely affected by the shift in position […] of the input patterns. Hence, their ability for pattern recognition was not so high."— Fukushima

Neocognitron

- Deep neural network (7 layers tested)
- Local connectivity ("receptive fields")
- Nonlinear filters with shared weights (S-layers)
- Average pooling (C-layers)
- ReLU activation function
- "Self-organised" (unsupervised) **– no backprop yet!**

Fukushima 1980

K. Fukushima

1980

Neocognitron, 1980

History of AI: Major Milestones

How to train your neural network?

F. Rosenblatt

Perceptron learning rule (1 layer)

A. Ivakhnenko

Group method of data handling

S. Linnainmaa

P. Werbos

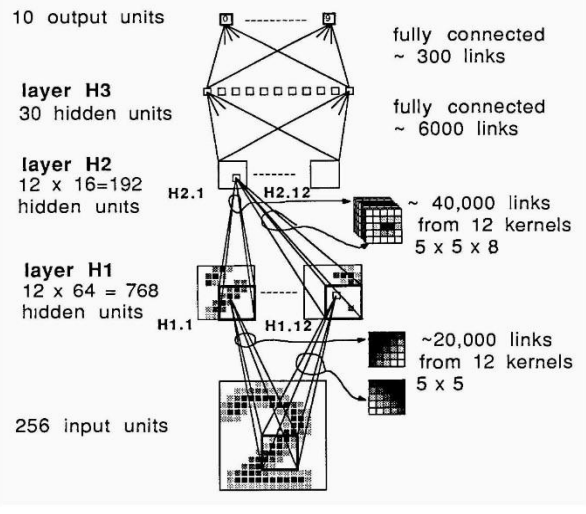
Backpropagation

D. Rumelhart

Rosenblatt 1957; Ivakhnenko, Lapa 1966; Linnainmaa 1970; Werbos 1982; Rumelhart et al. 1986

History of AI: Major Milestones

Convolutional neural networks



First version of a CNN

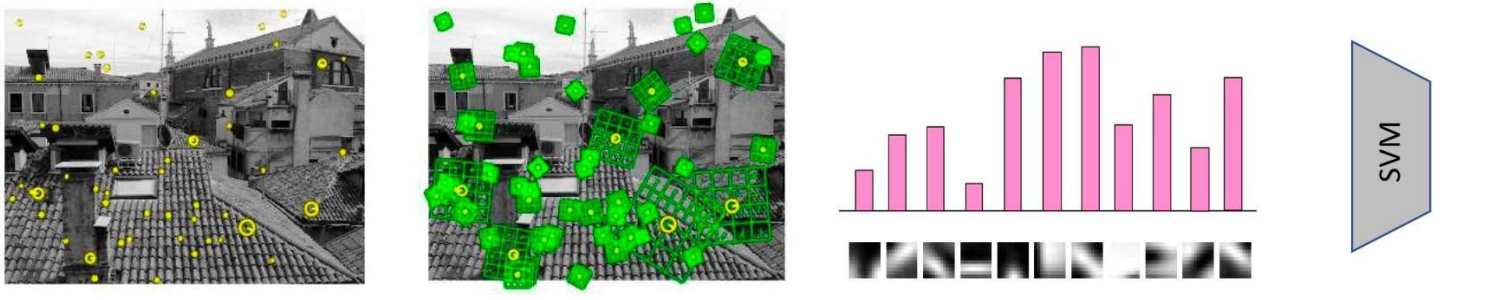
AT&T DSP-32C
capable of 125m floating point multiply-accumulate operations/sec

Y. LeCun

LeCun et al. 1989

History of AI: Major Milestones

Computer vision in the 2000s



Feature detection **Feature description** **Feature aggregation** **Classification**

A typical image classification pipeline from the 2000s

Sivic, Zisserman 2003

Common Techniques for Feature Detection Before 2000

- **Harris Corner Detection :** Detects corners in an image by analyzing changes in intensity in different directions.

- **Canny Edge Detection:** Detects edges in an image by finding areas of rapid intensity change.

- **Scale-Invariant Feature Transform (SIFT):** Detects keypoints that are invariant to scale and rotation changes.

Handcrafter features/kernel matrices

Original

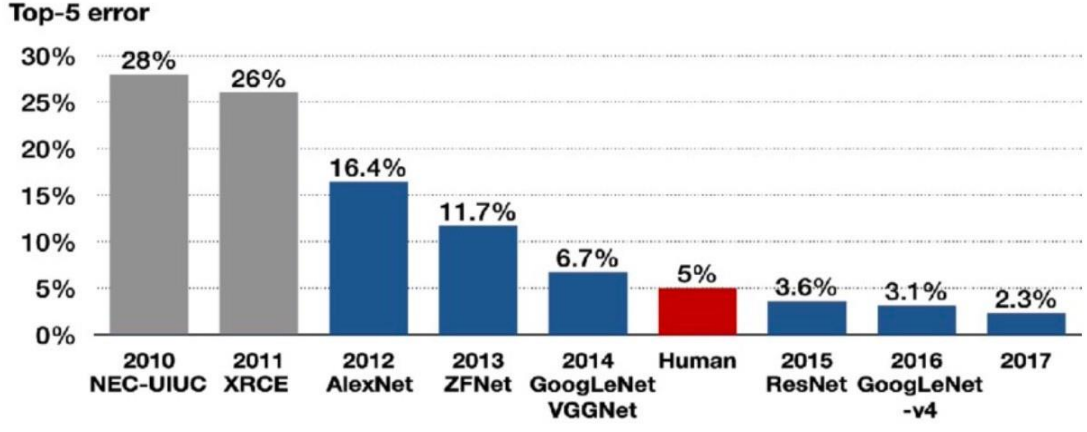
Sharpen

Edge Detect

"Strong" Edge Detect

History of AI: Major Milestones

ImageNet

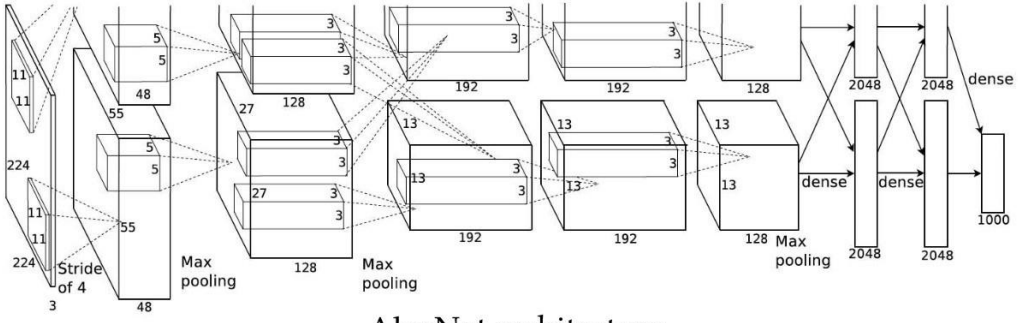


L. Fei-Fei

AlexNet beating all "handcrafted" approaches on ImageNet benchmark—the moment of truth for computer vision

History of AI: Major Milestones

AlexNet



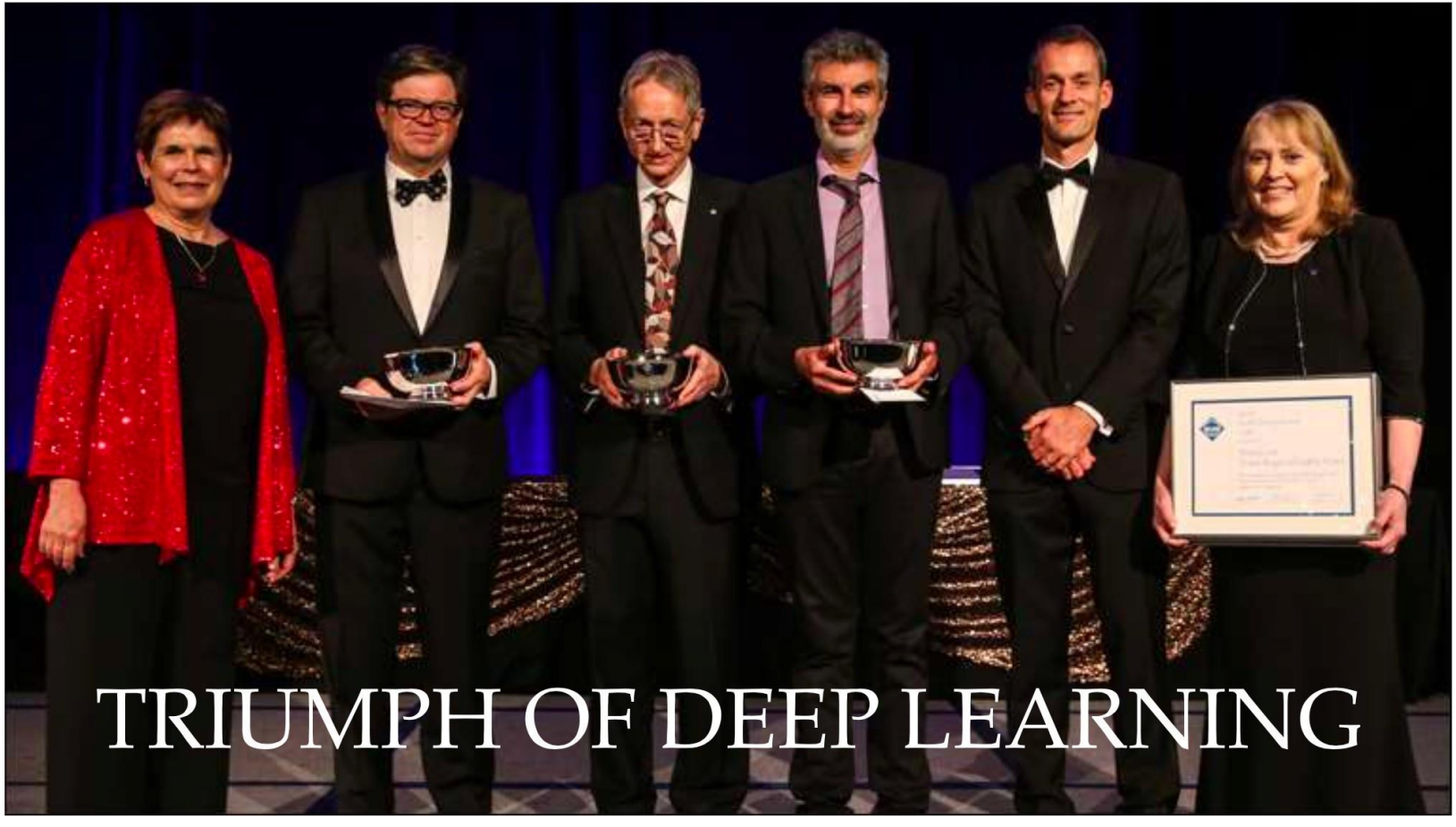
AlexNet architecture

Nvdia GTX 580 GPU capable of
~200G FLOP / sec

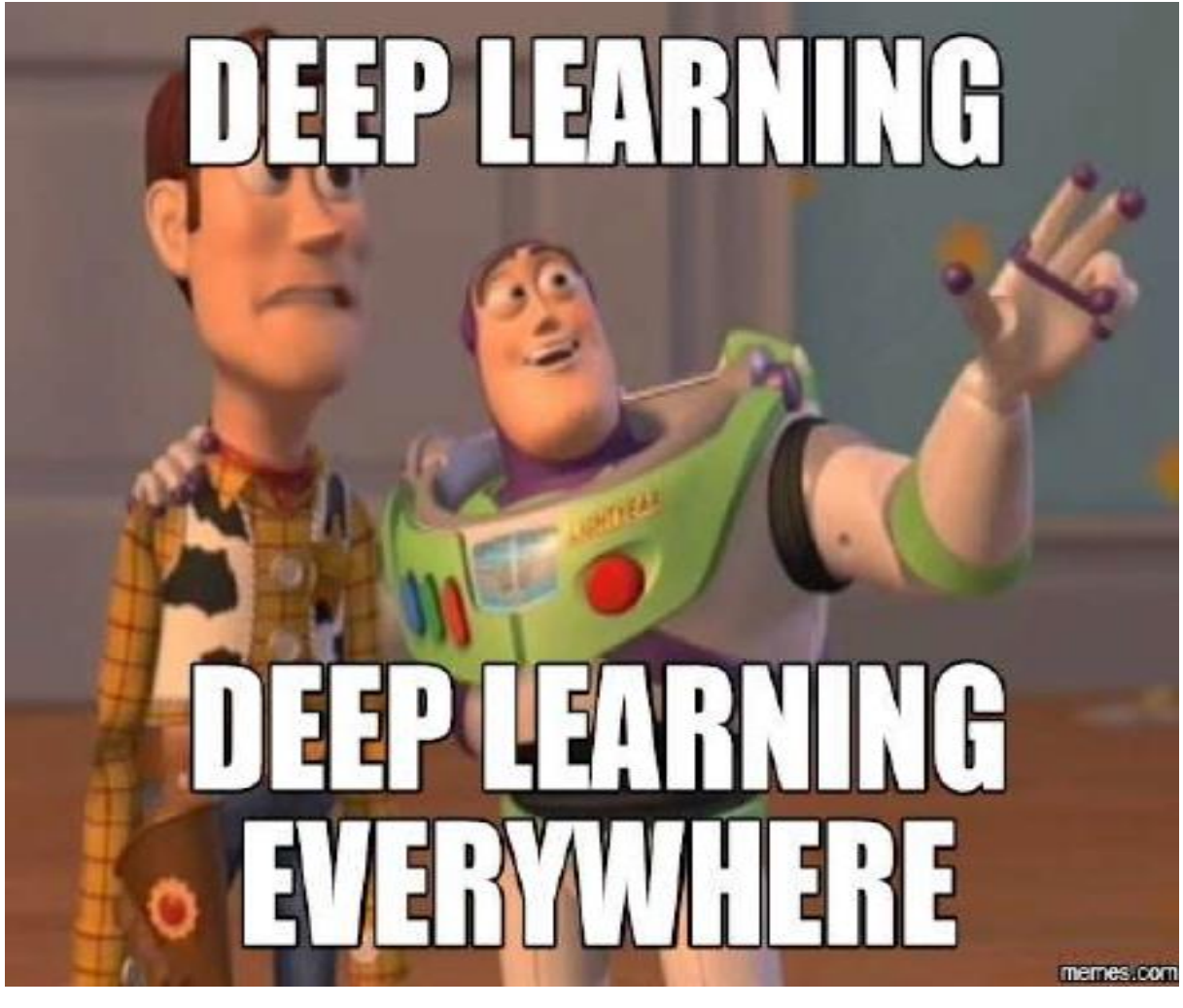
A. Krizhevsky

Krizhevsky et al. 2012

History of AI: Major Milestones



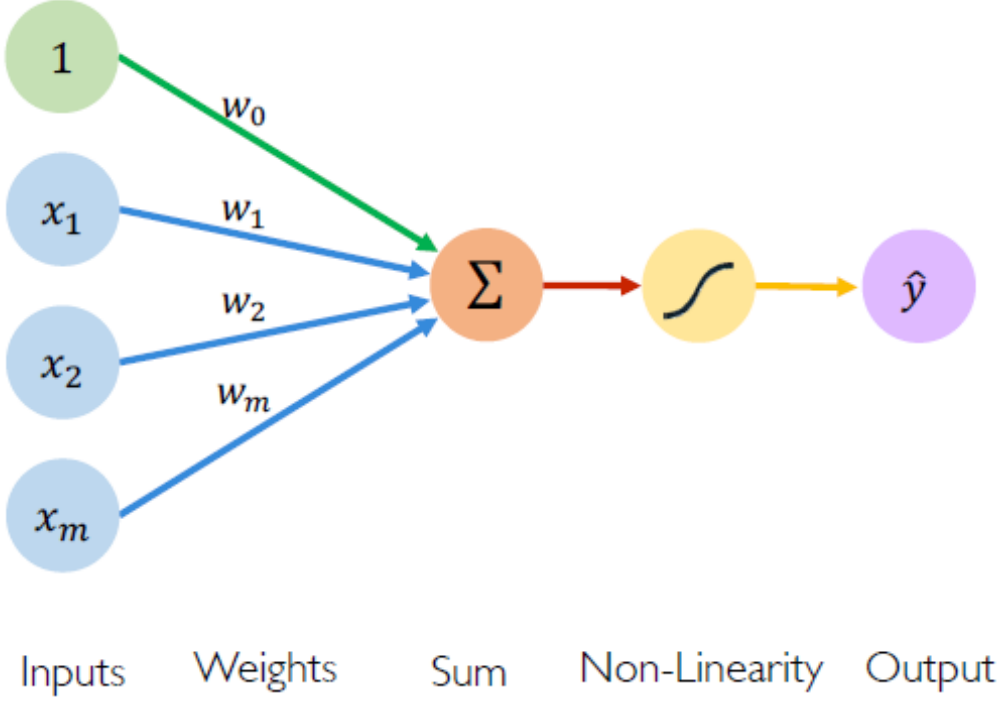
Deep Learning Memes



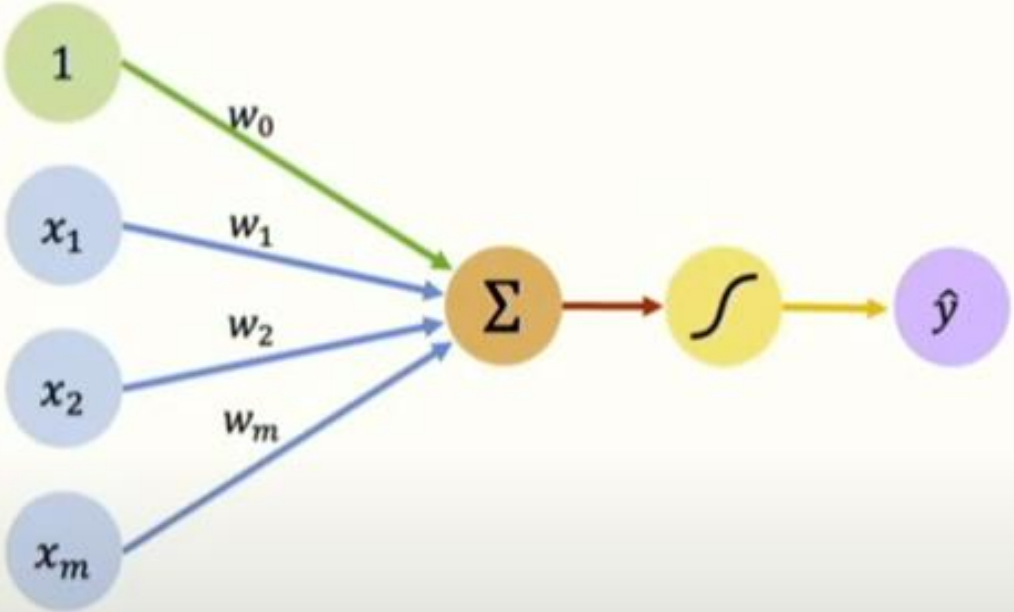
The Perceptron

The structural building block of deep learning

The Perceptron: Forward Propagation



The Perceptron: Forward Propagation



Inputs Weights Sum Non-Linearity Output

Output

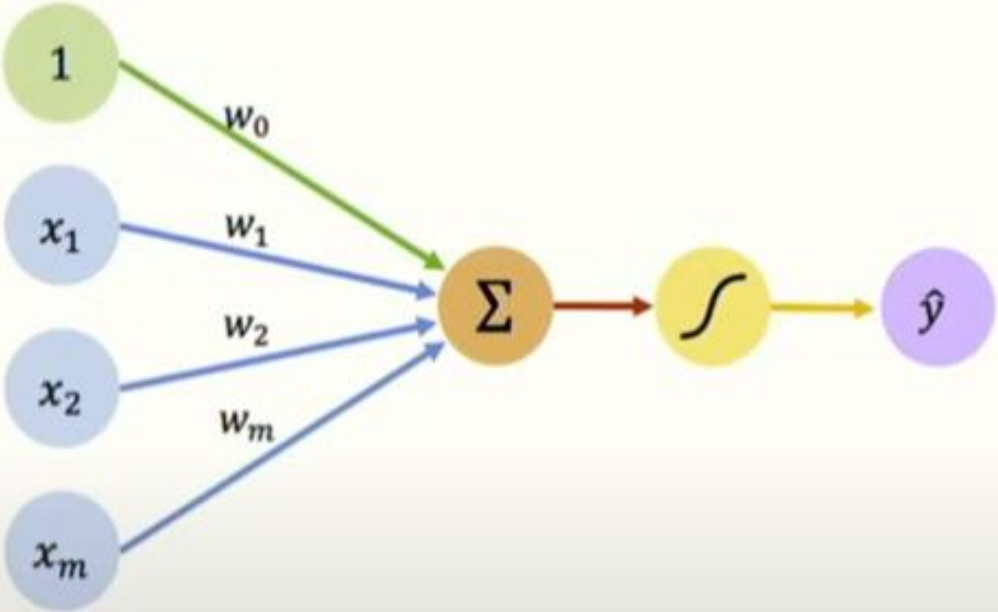
Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right)$$

Non-linear activation function

Bias

The Perceptron: Forward Propagation



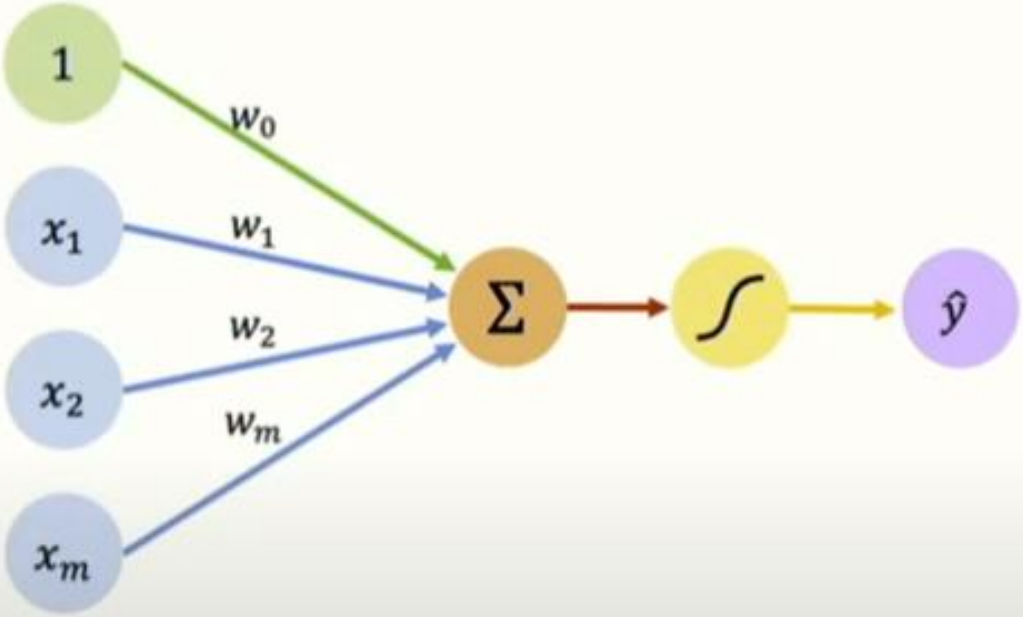
Inputs Weights Sum Non-Linearity Output

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right)$$

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

The Perceptron: Forward Propagation



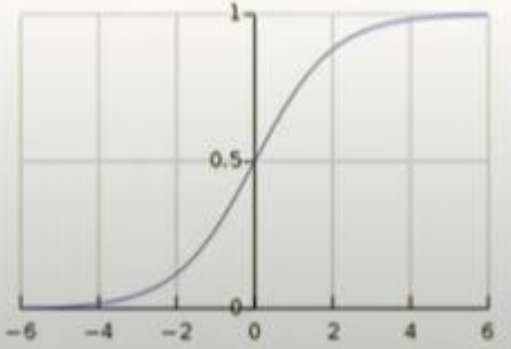
Inputs Weights Sum Non-Linearity Output

Activation Functions

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

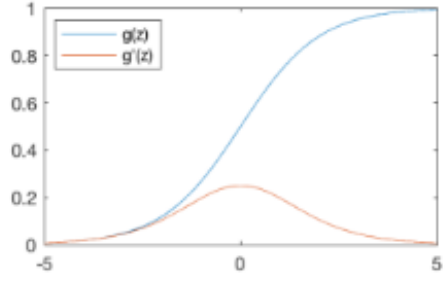
- Example: sigmoid function

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



Activation Functions

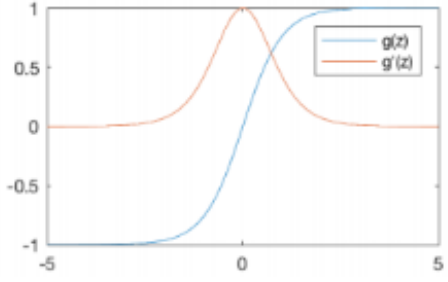
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

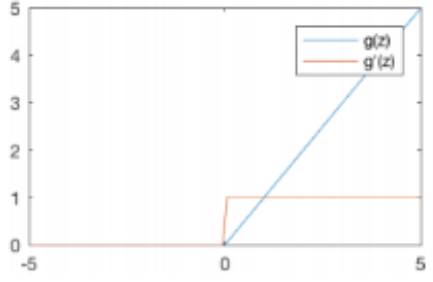
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

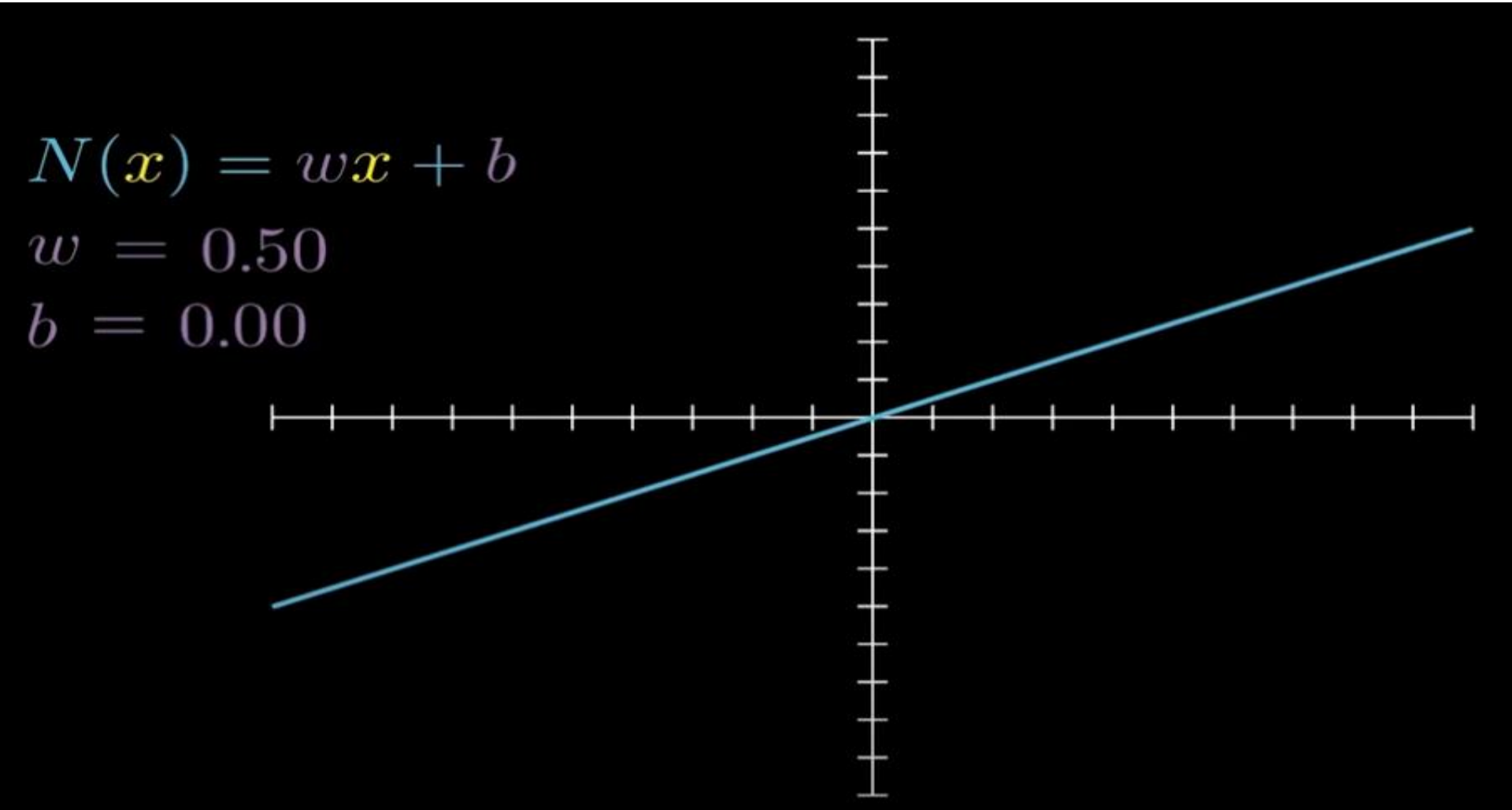
Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

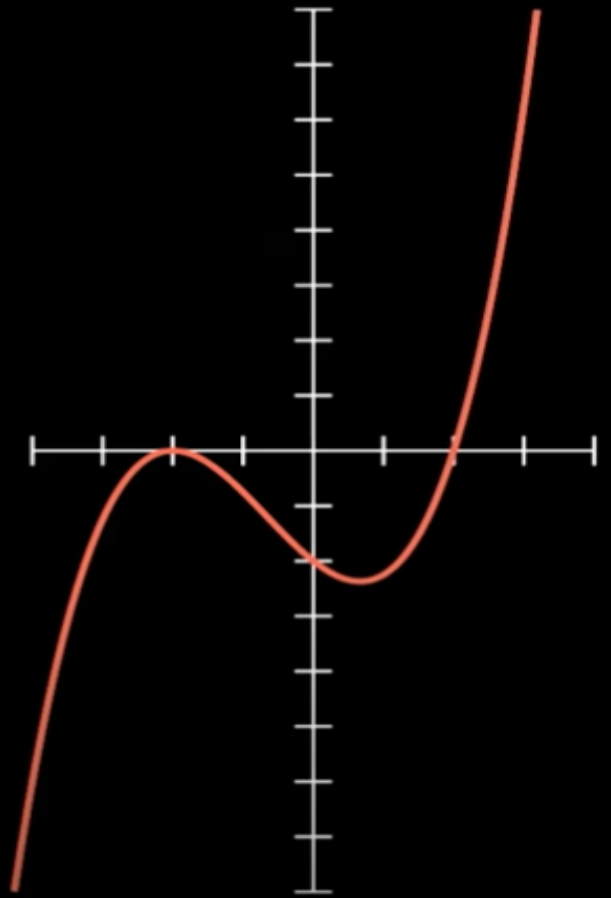
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Neuron as a linear function



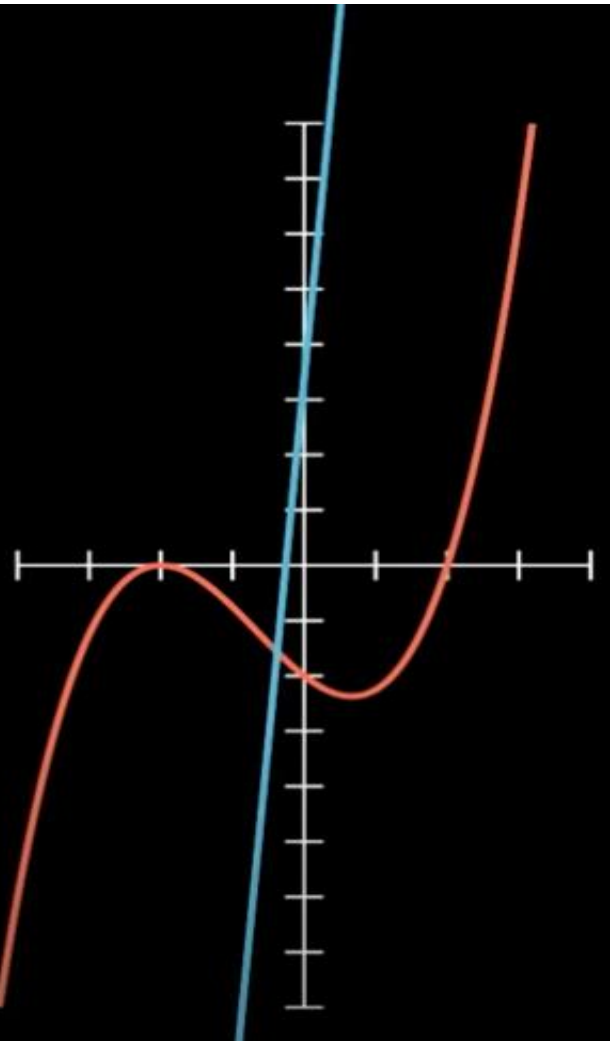
Example: Estimating a target function using multiple neurons

$$N_0(x) = -5x + -7.7$$
$$N_1(x) = -1.2x + -1.3$$
$$N_2(x) = 1.2x + 1$$
$$N_3(x) = 1.2x + -0.2$$
$$N_4(x) = 2x + -1.1$$
$$N_5(x) = 5x + -5$$

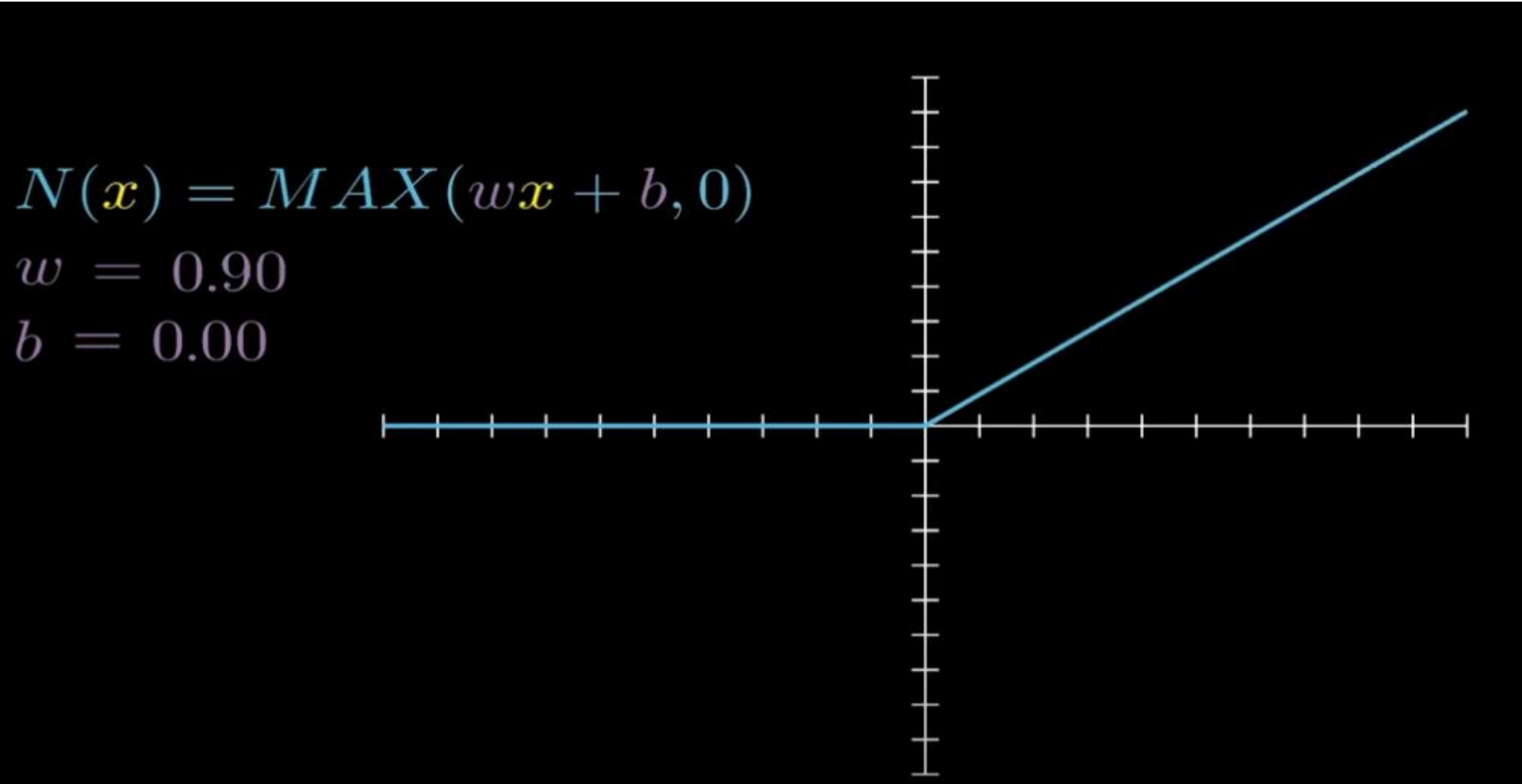


No Non-linearity

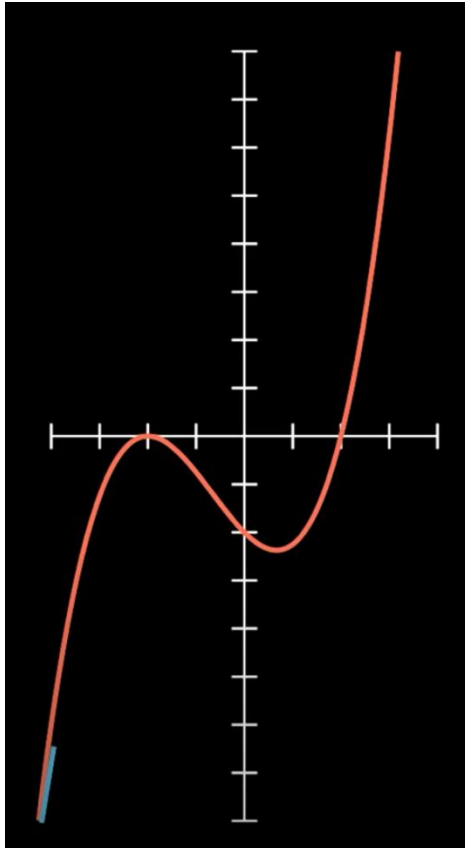
$$-1(-5x + -7.7)+$$
$$-1(-1.2x + -1.3)+$$
$$-1(1.2x + 1)+$$
$$1(1.2x + -0.2)+$$
$$1(2x + -1.1)+$$
$$1(5x + -5)$$

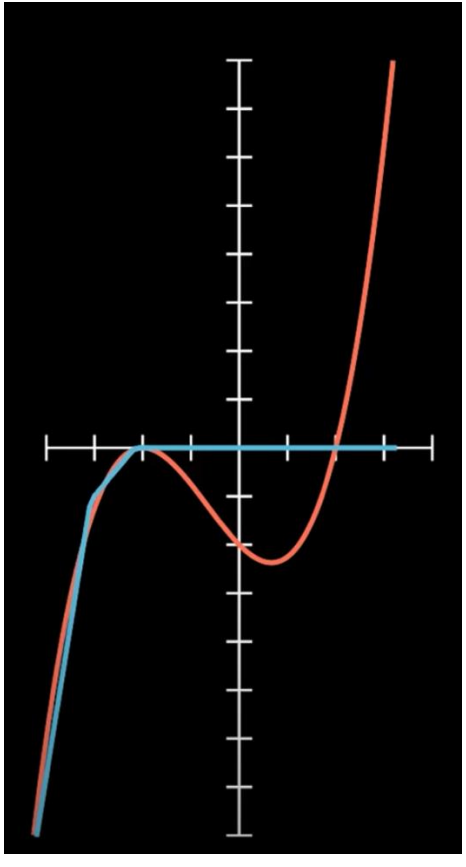


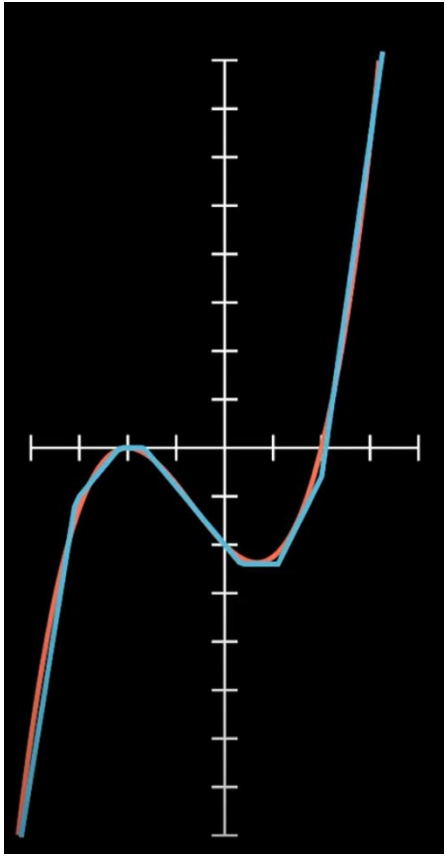
Inducing Non-linearity using ReLU



Non-linearity

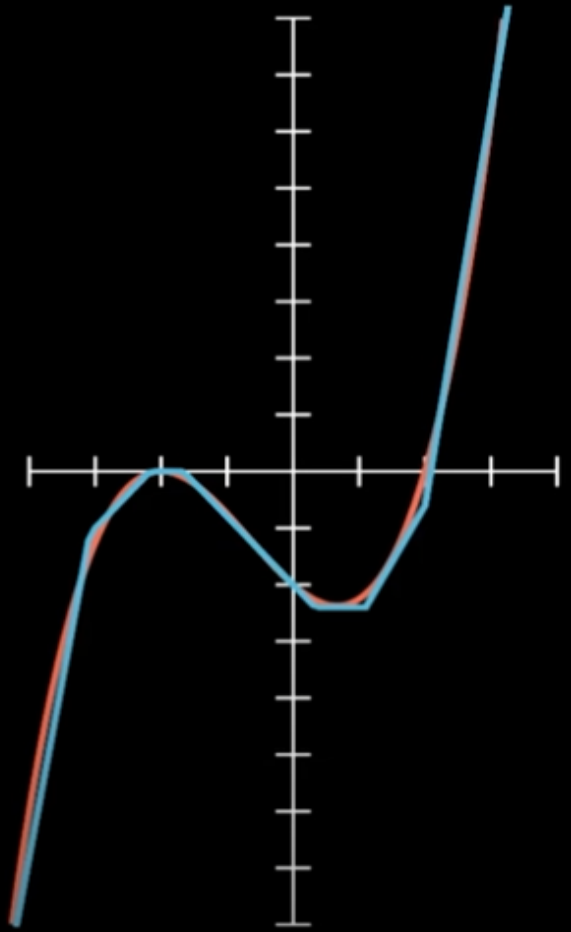






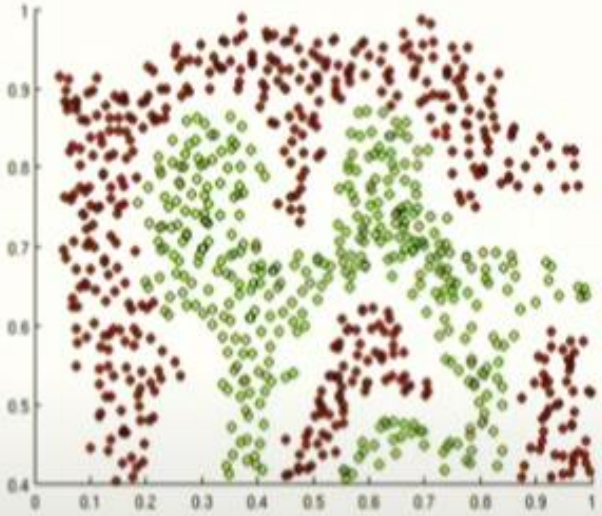
Non-linearity

$$-max(-5x + -7.7, 0) +$$
$$-max(-1.2x + -1.3, 0) +$$
$$-max(1.2x + 1, 0) +$$
$$max(1.2x + -0.2, 0) +$$
$$max(2x + -1.1, 0) +$$
$$max(5x + -5, 0)$$



Importance of Activation Functions

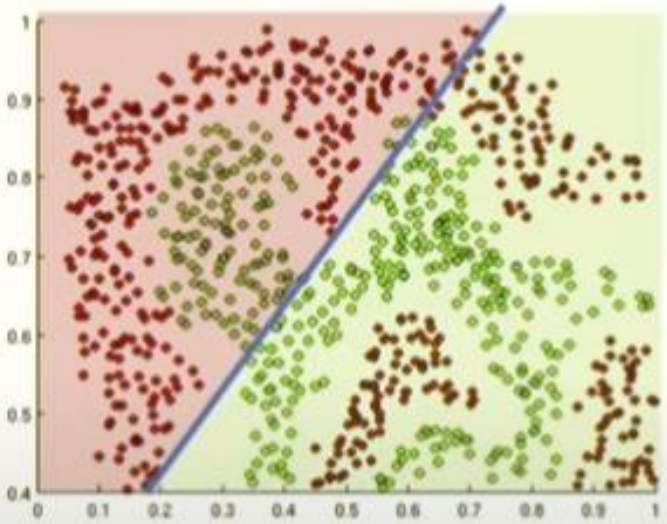
The purpose of activation functions is to ***introduce non-linearities*** *into the network*



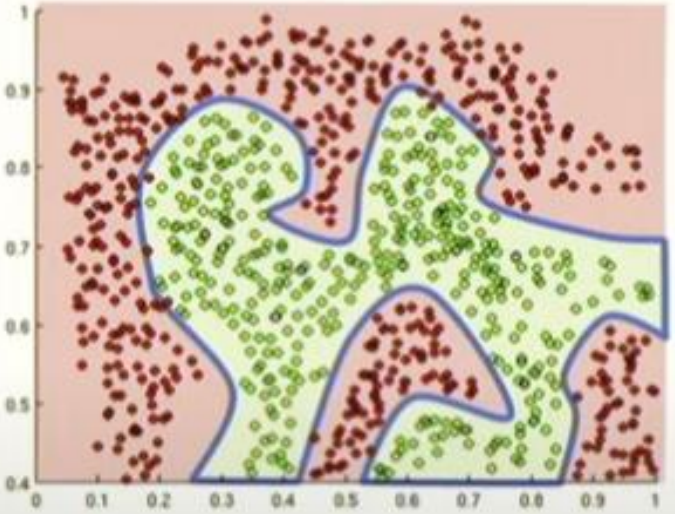
What if we wanted to build a neural network to distinguish green vs red points?

Importance of Activation Functions

The purpose of activation functions is to ***introduce non-linearities*** *into the network*

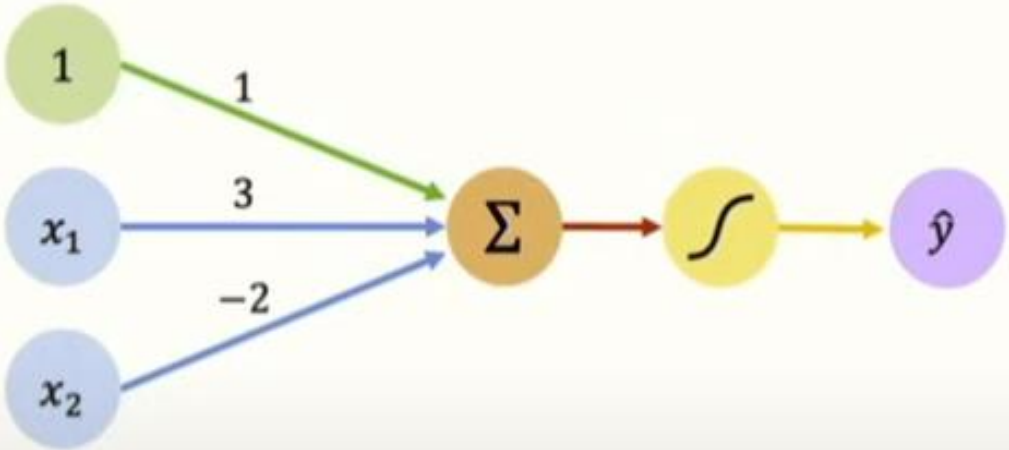


Linear activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

The Perceptron: Example

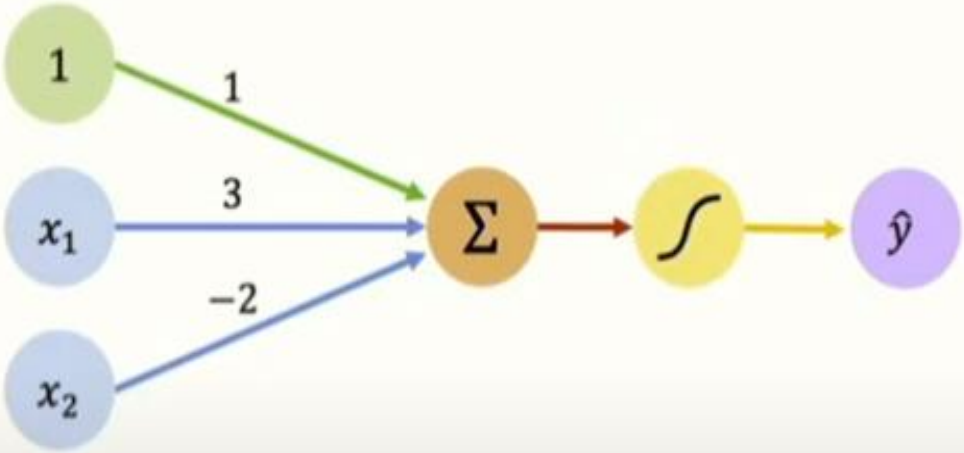


We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

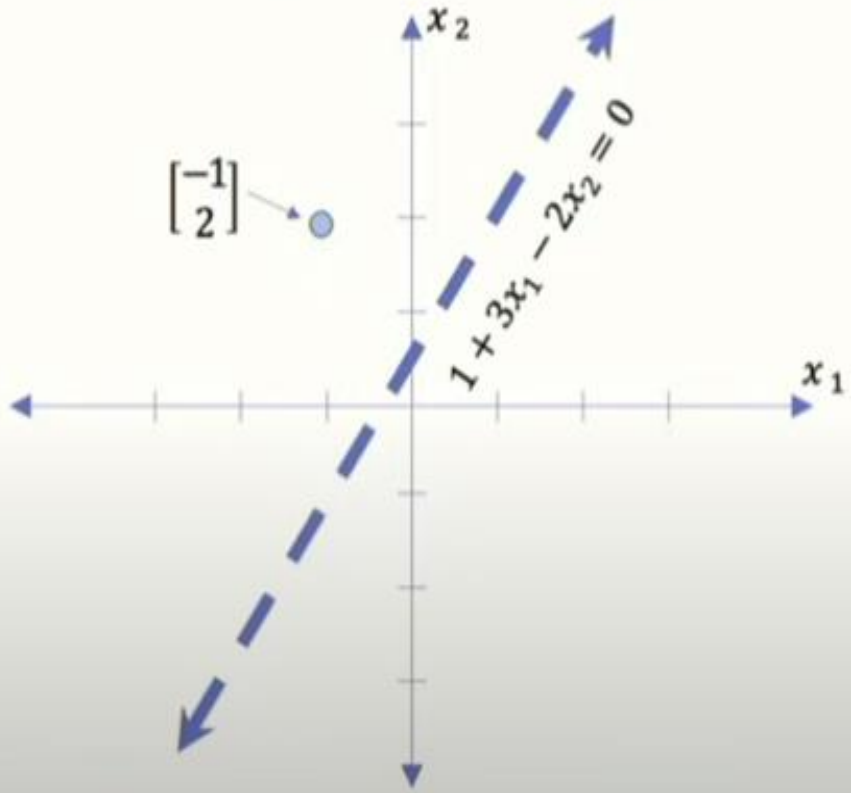
$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

This is just a line in 2D!

The Perceptron: Example



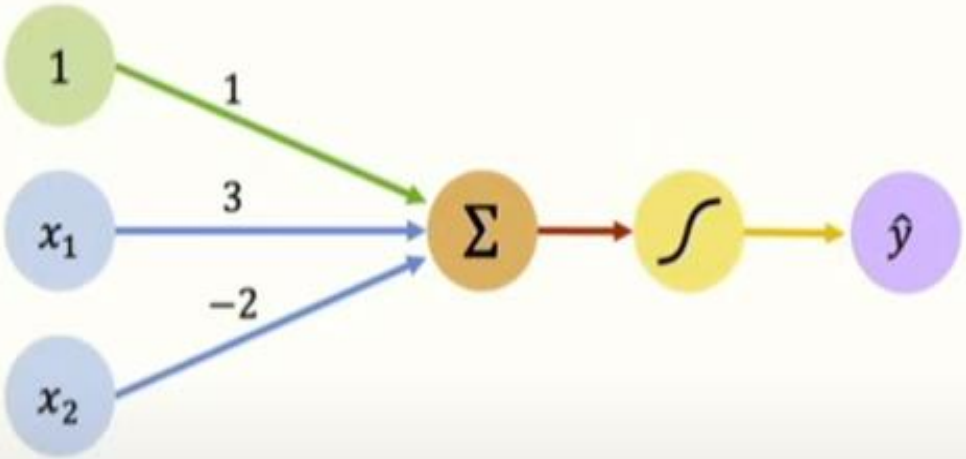
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



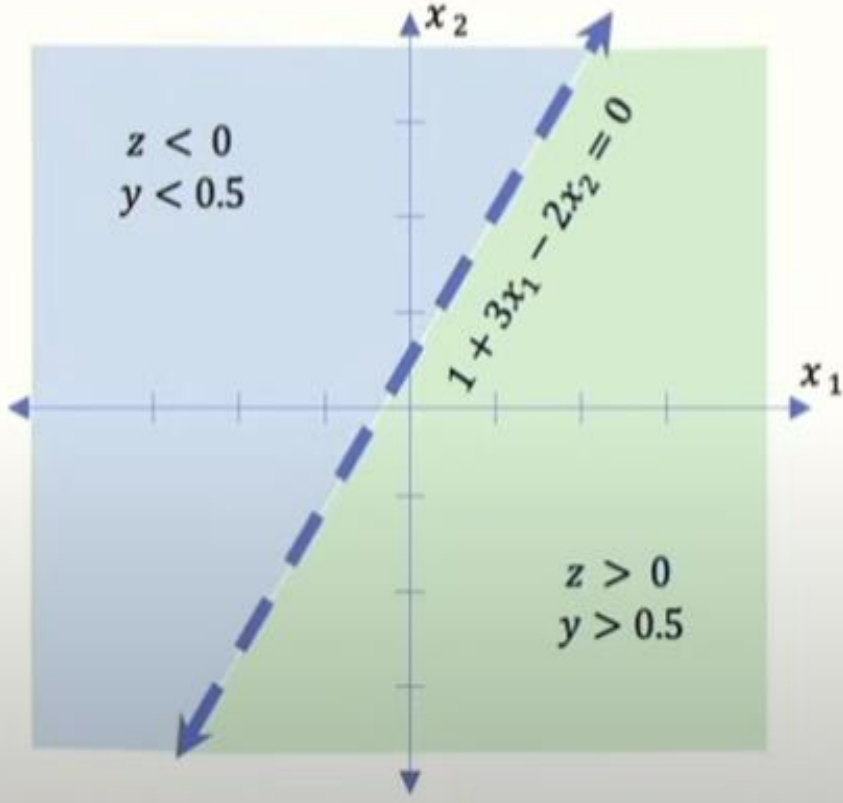
Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$\hat{y} = g\big(1 + (3 * -1) - (2 * 2)\big)$$
$$= g(-6) \approx 0.002$$

The Perceptron: Example

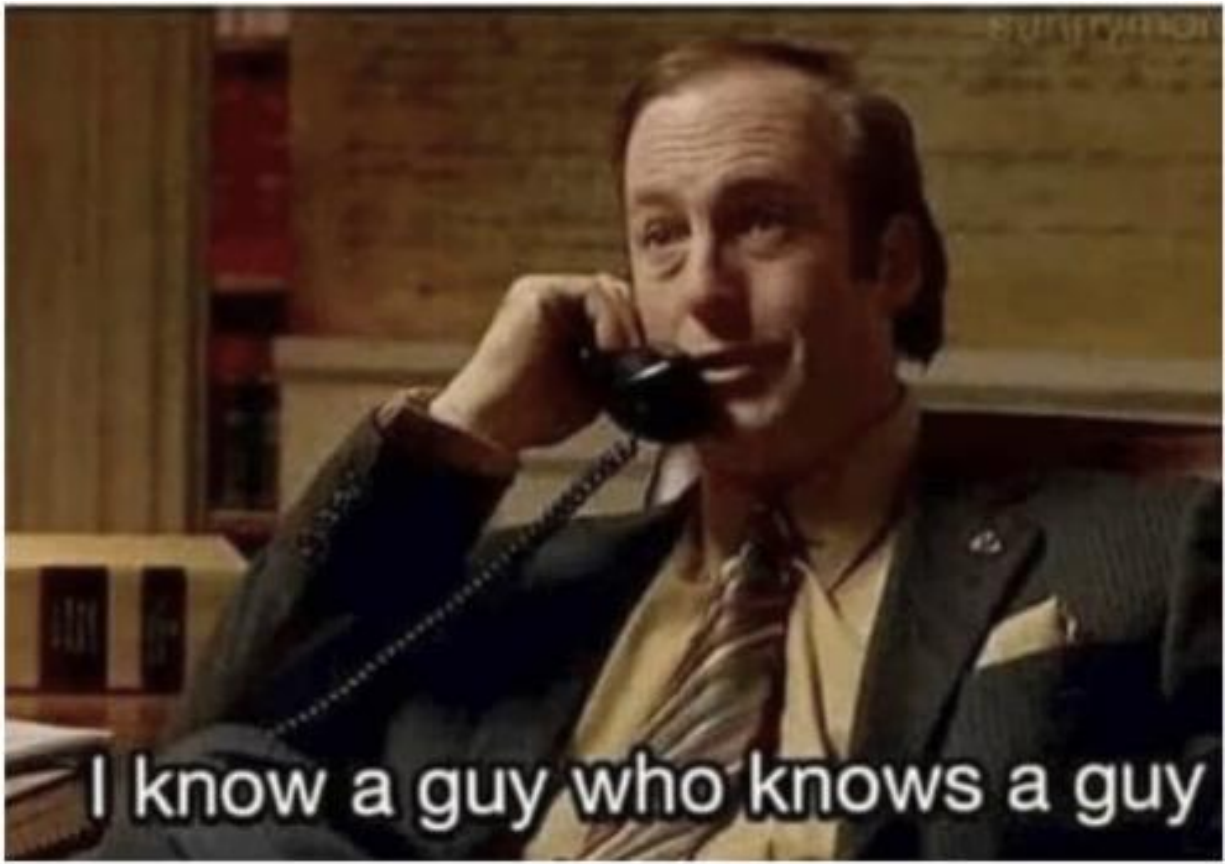


$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



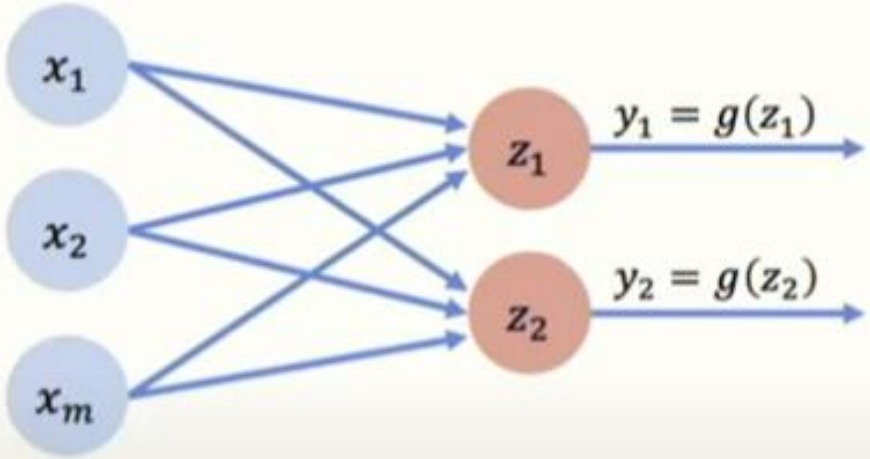
Building Neural Networks with Perceptrons

Deep Learning Memes



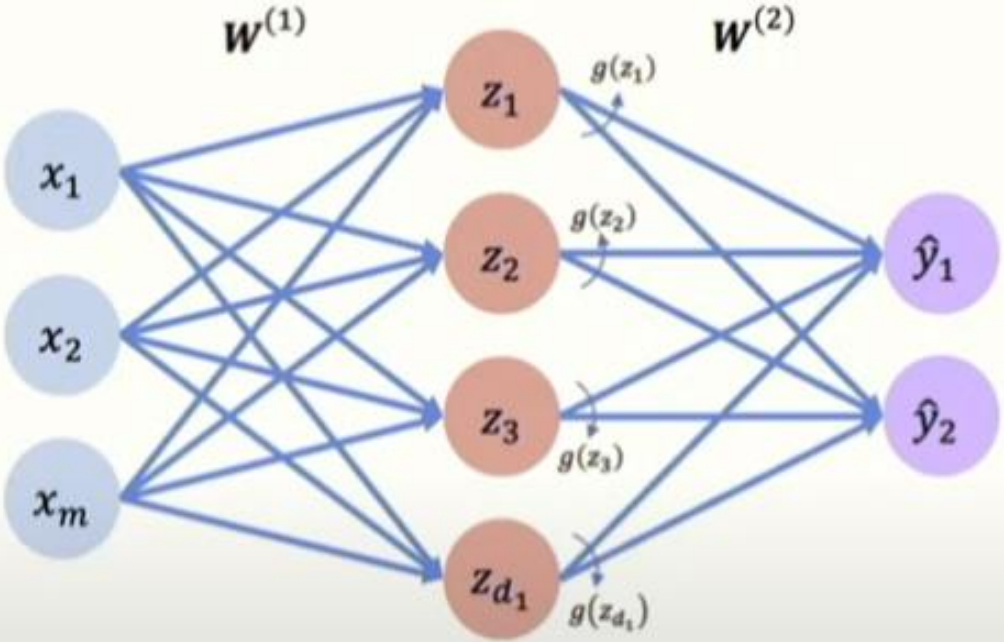
What the job of a neuron in an ANN really looks like.

Multi Output Perceptron



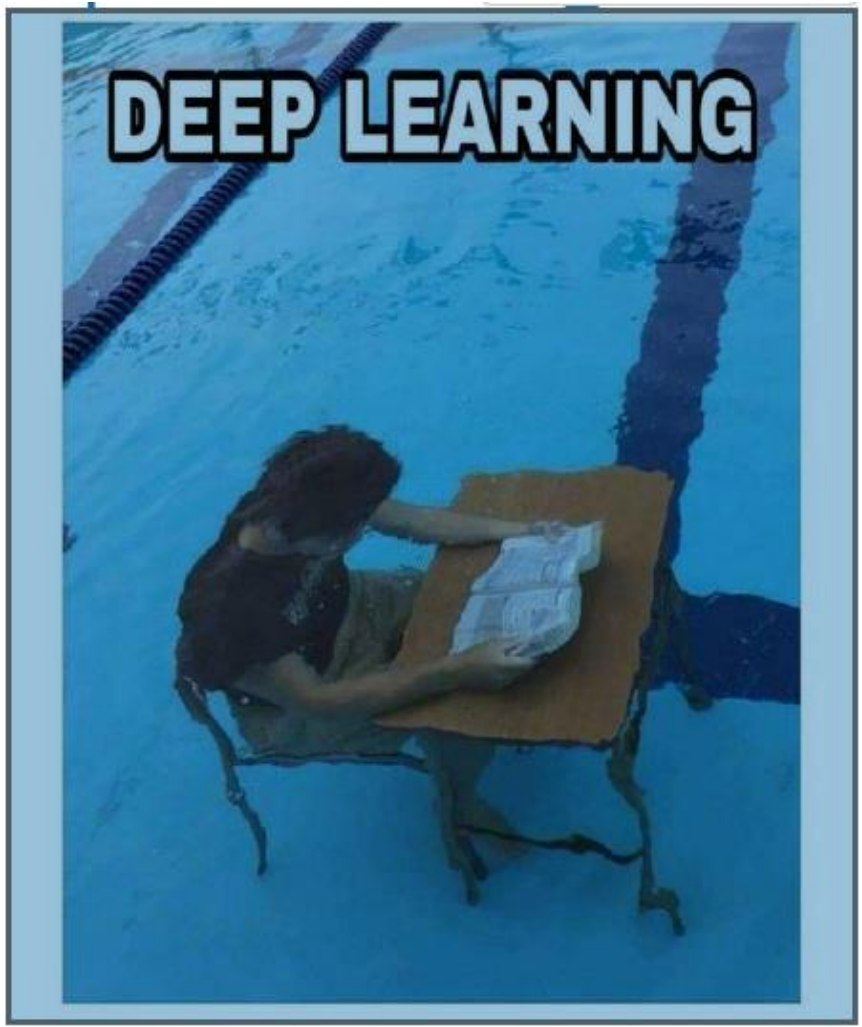
$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

Single Layer Neural Network

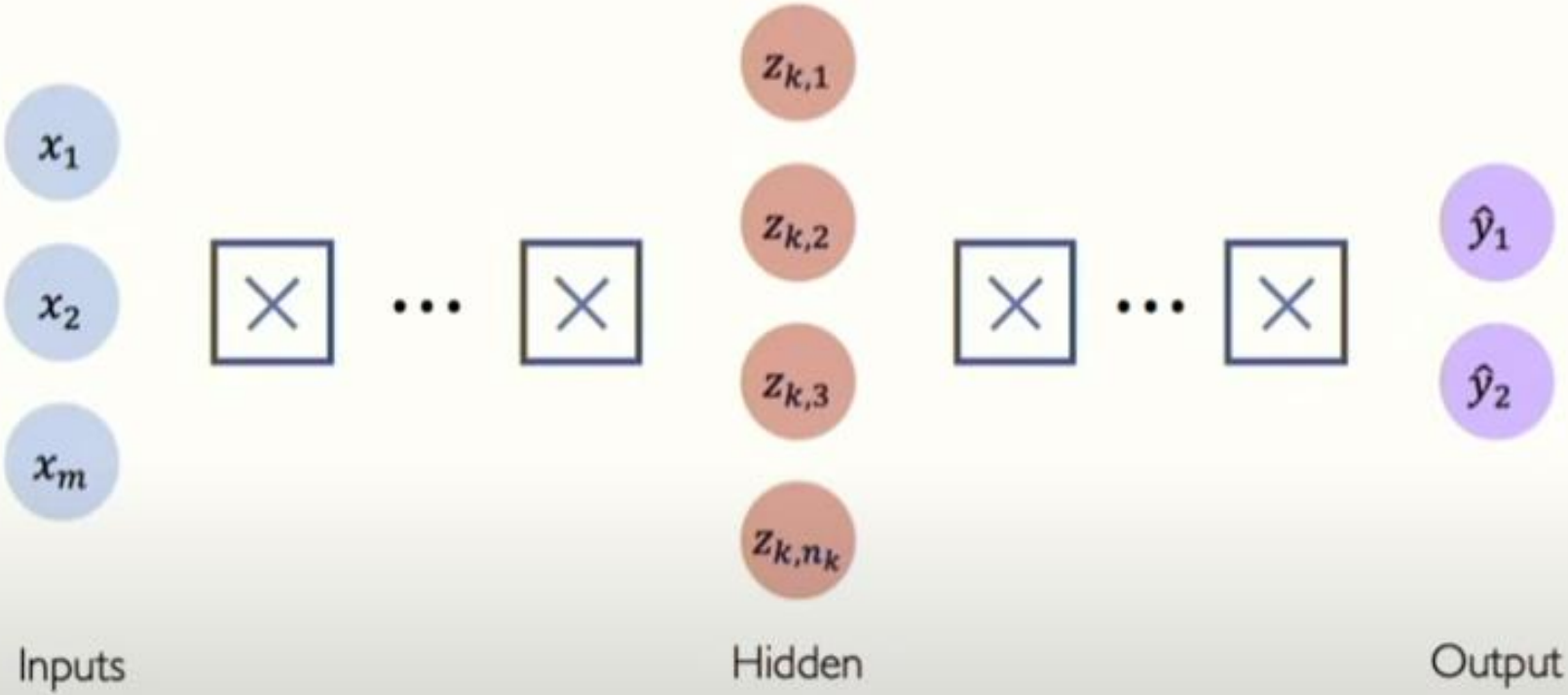


$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,i}^{(1)} \qquad \hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} g(z_j) \, w_{j,i}^{(2)}\right)$$

What's Deep in Deep Learning



Deep Neural Network



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

Example Problem

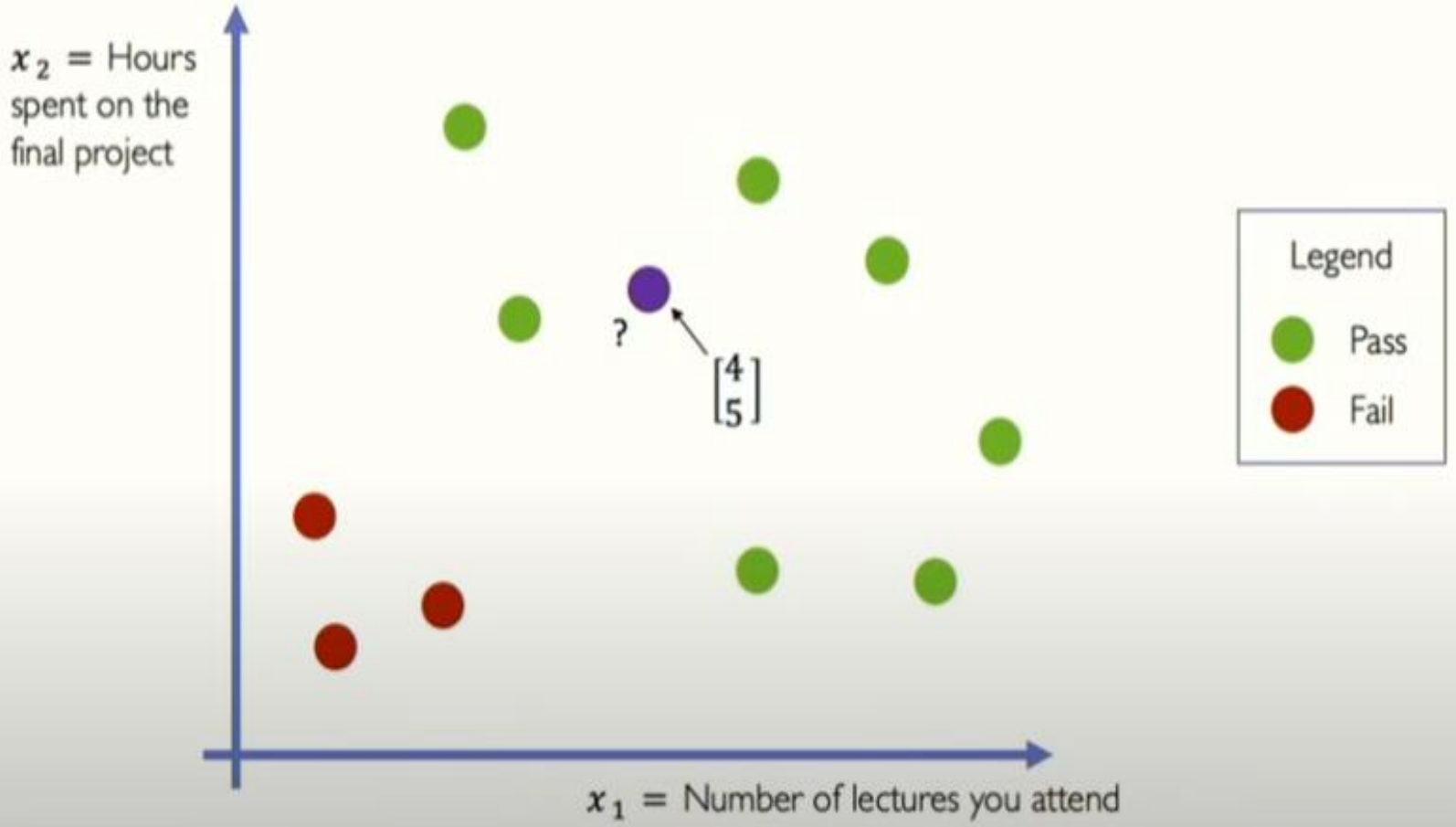
Will I pass this class?

Let's start with a simple two feature model

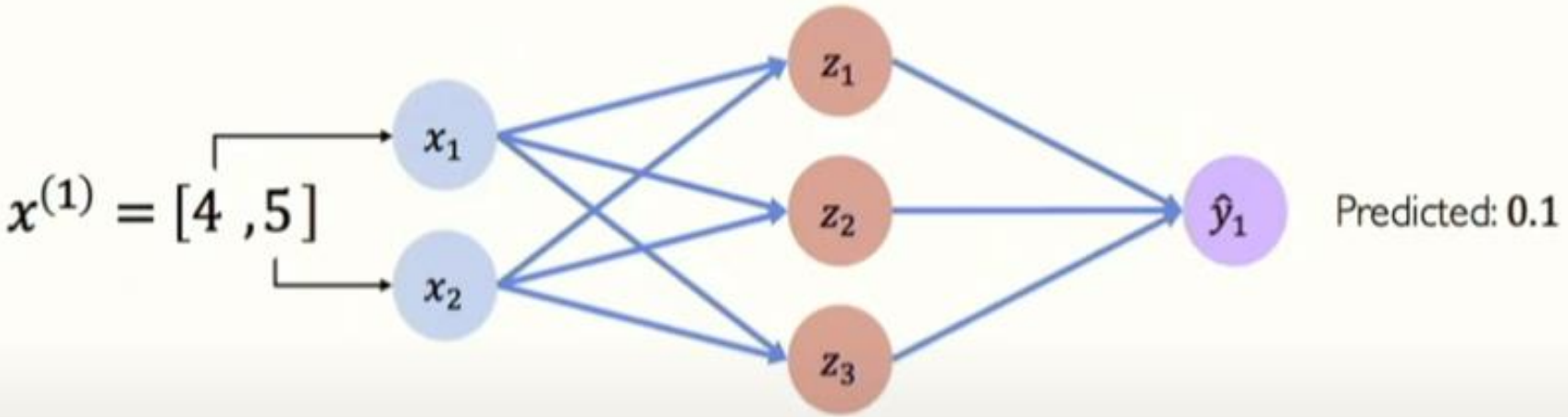
x_1 = Number of lectures you attend

x_2 = Hours spent on the final project

Example Problem: Will I pass this class?

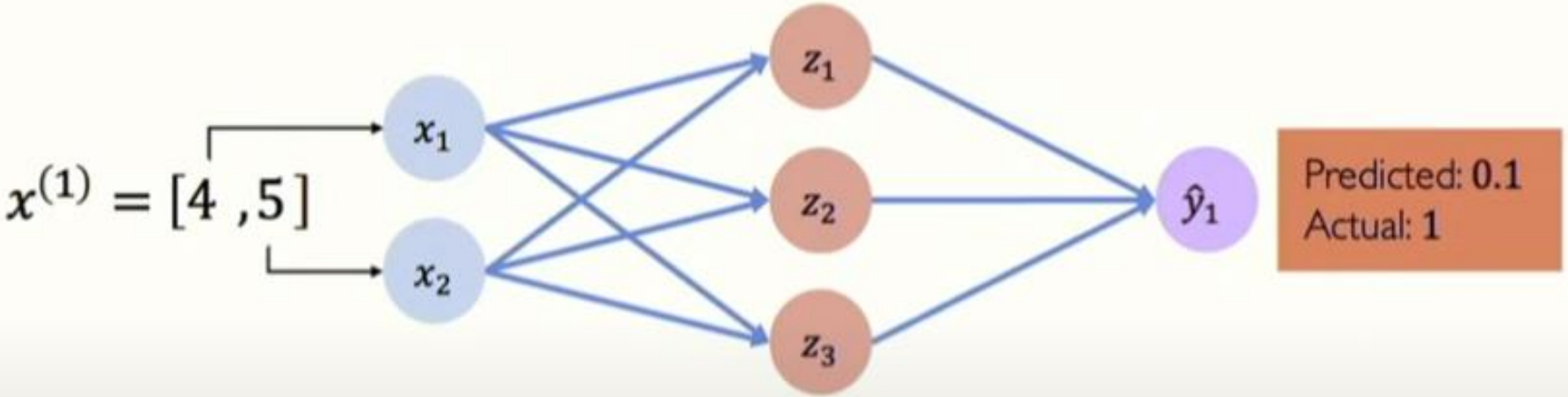


Example Problem: Will I pass this class?



Quantifying Loss

*The **loss** of our network measures the cost incurred from incorrect predictions*

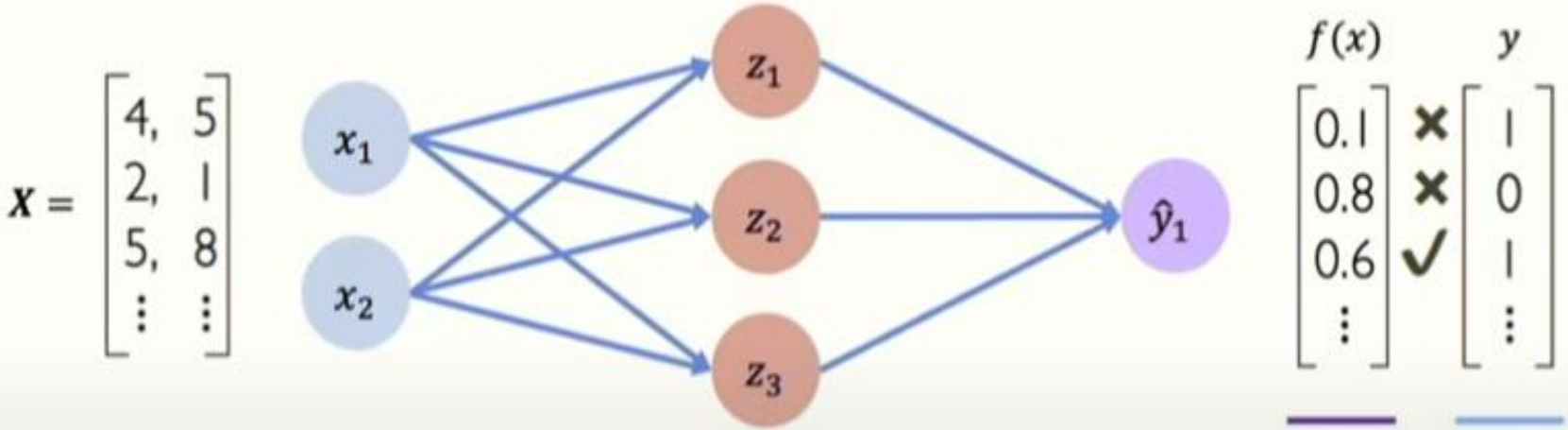


$$\mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

Predicted Actual

Empirical Loss

*The **empirical loss** measures the total loss over our entire dataset*



Also known as:
- Objective function
- Cost function
- Empirical Risk

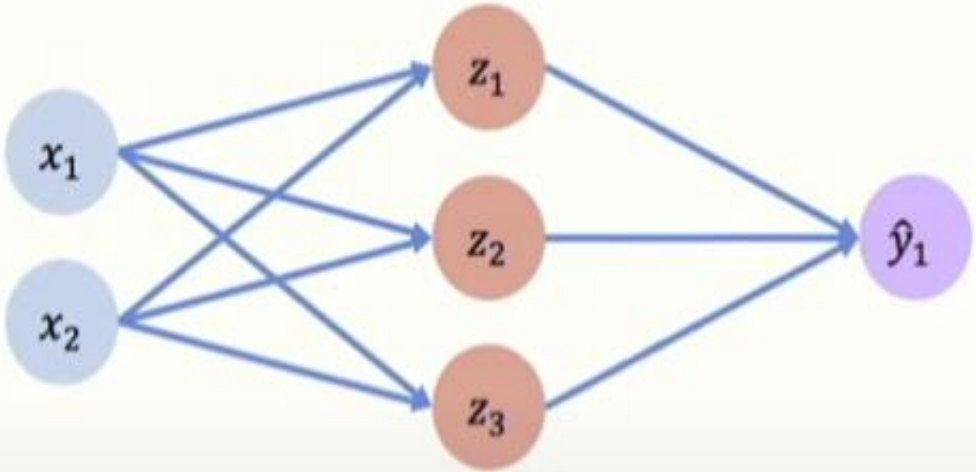
$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

Predicted Actual

Binary Cross Entropy Loss

Cross entropy loss *can be used with models that output a probability between 0 and 1*

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$f(x)$ y

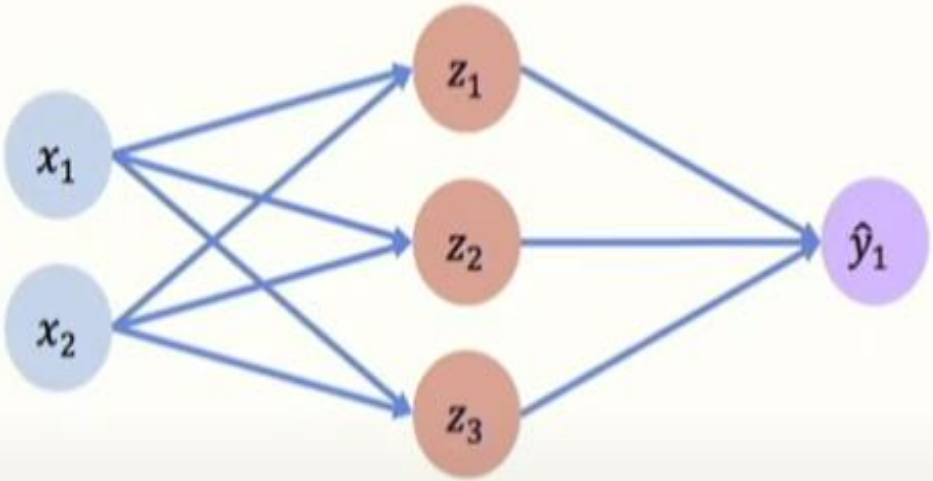
$$\begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

$$J(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big) + (1-\underbrace{y^{(i)}}_{\text{Actual}}) \log\Big(1-\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)$$

Mean Squared Error Loss

Mean squared error loss *can be used with regression models that output continuous real numbers*

$$X = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$f(x)$		y
30	✗	90
80	✗	20
85	✓	95
⋮		⋮

Final Grades (percentage)

$$J(W) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)}; W\right)\right)^2$$

Actual Predicted

Loss Optimization

We want to find the network weights that ***achieve the lowest loss***

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} J(\boldsymbol{W})$$

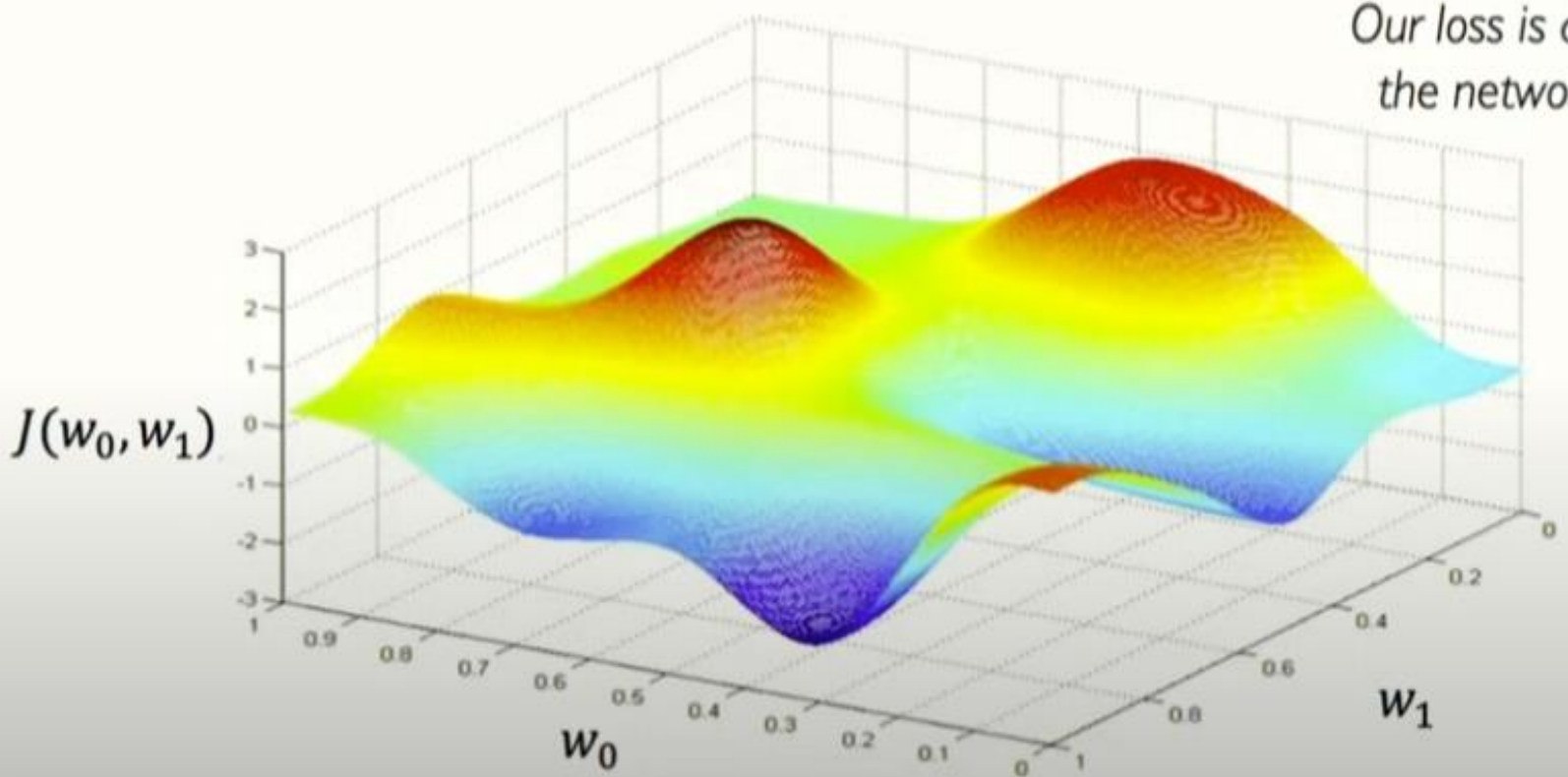
Remember:

$$\boldsymbol{W} = \left\{\boldsymbol{W}^{(0)}, \boldsymbol{W}^{(1)}, \ldots\right\}$$

Loss Optimization

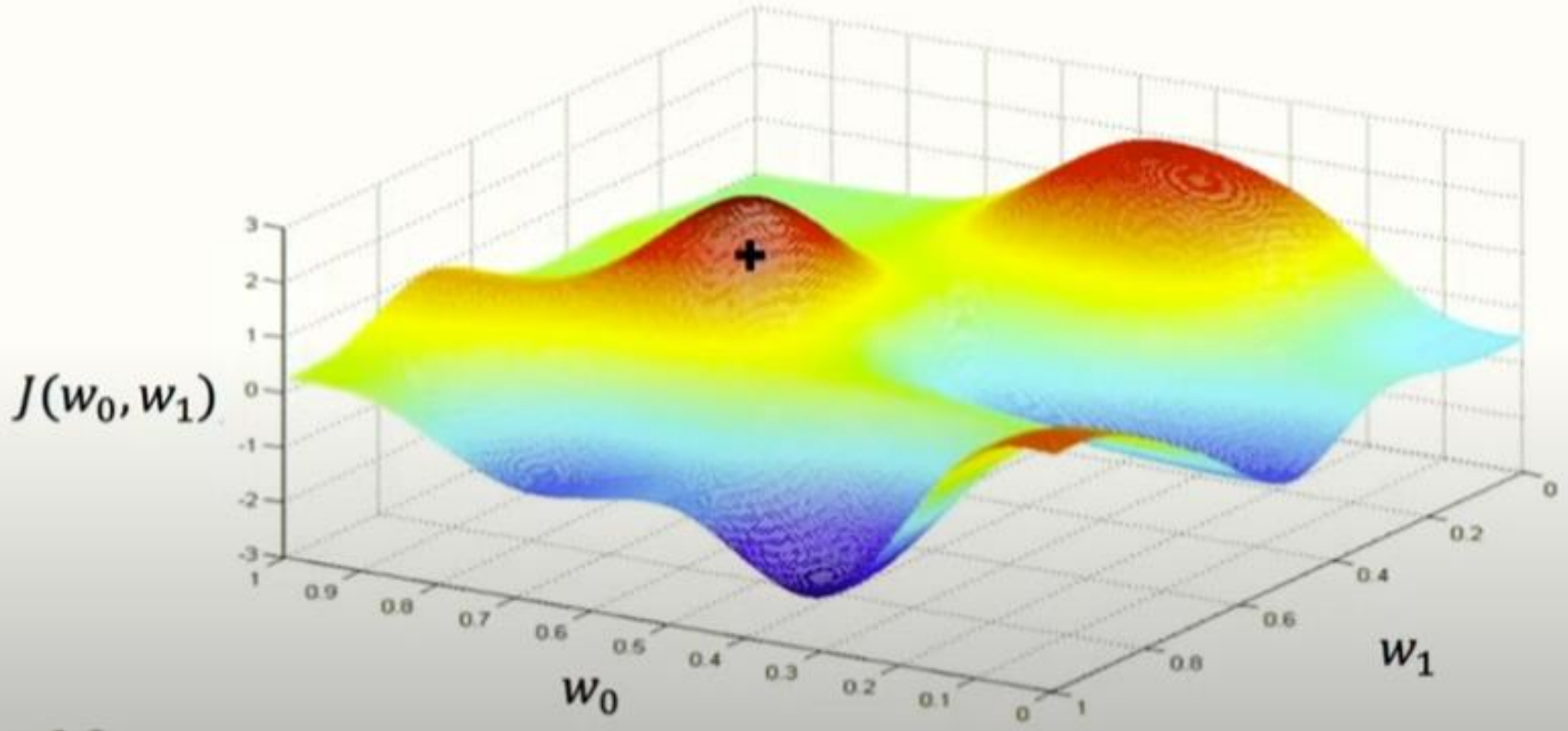
$$W^* = \underset{W}{\operatorname{argmin}} J(W)$$

Remember:
Our loss is a function of the network weights!



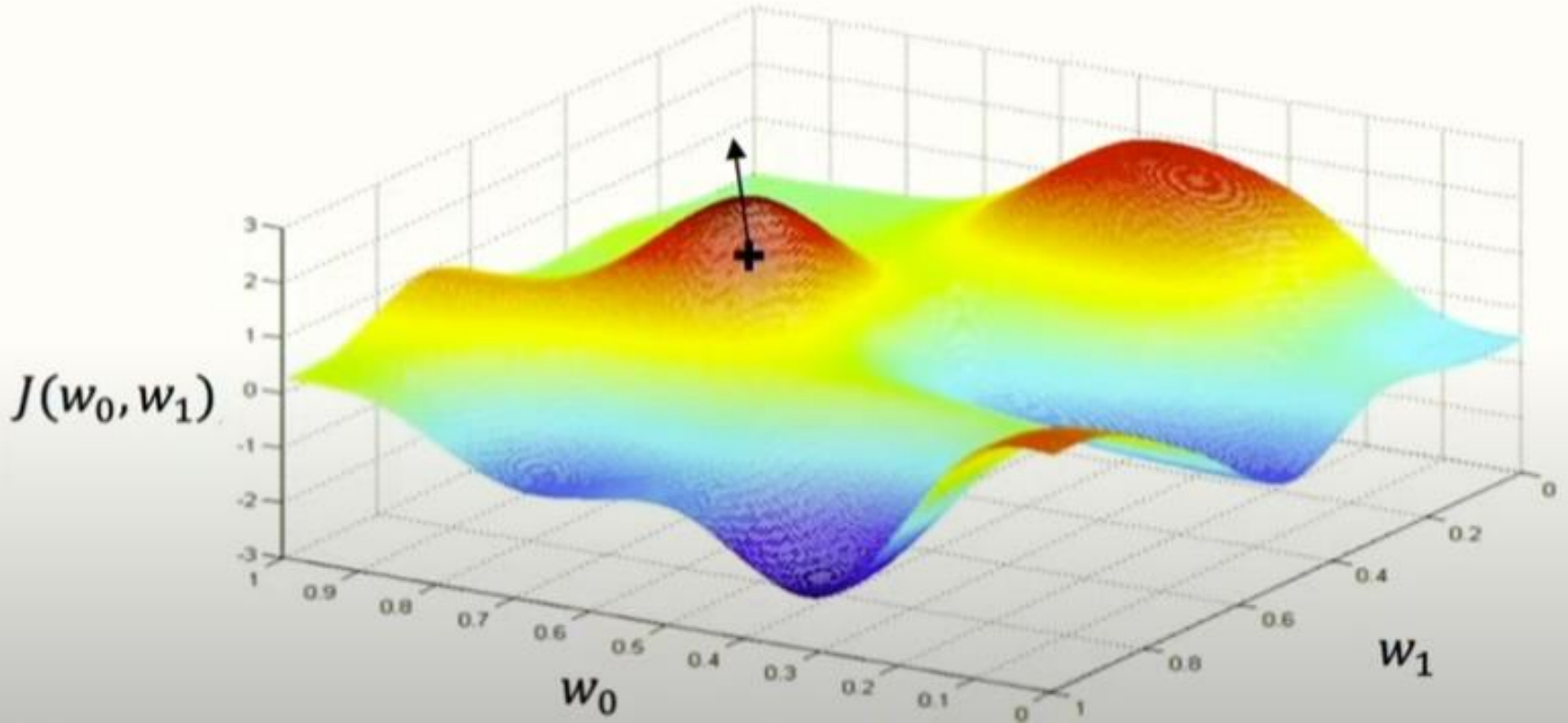
Loss Optimization

Randomly pick an initial (w_0, w_1)



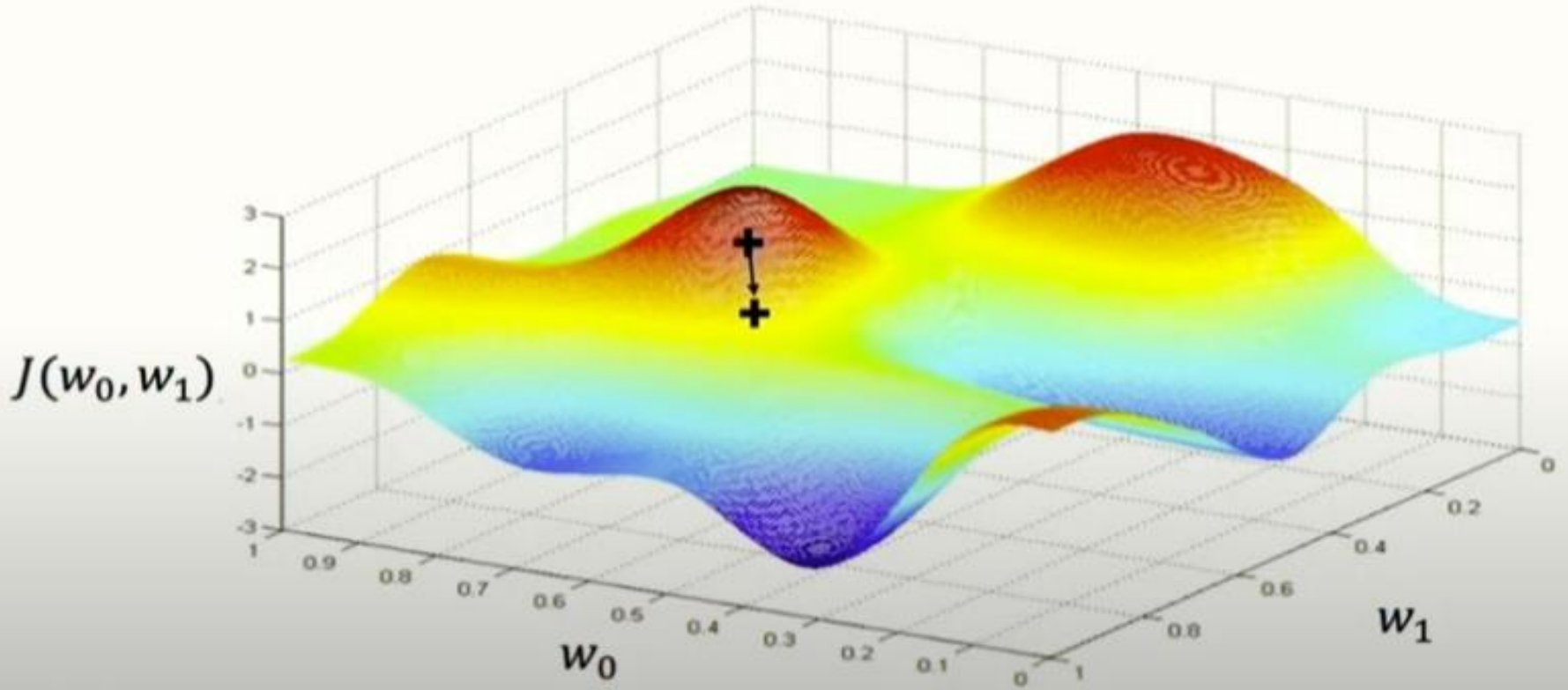
Loss Optimization

Compute gradient, $\frac{\partial J(W)}{\partial W}$



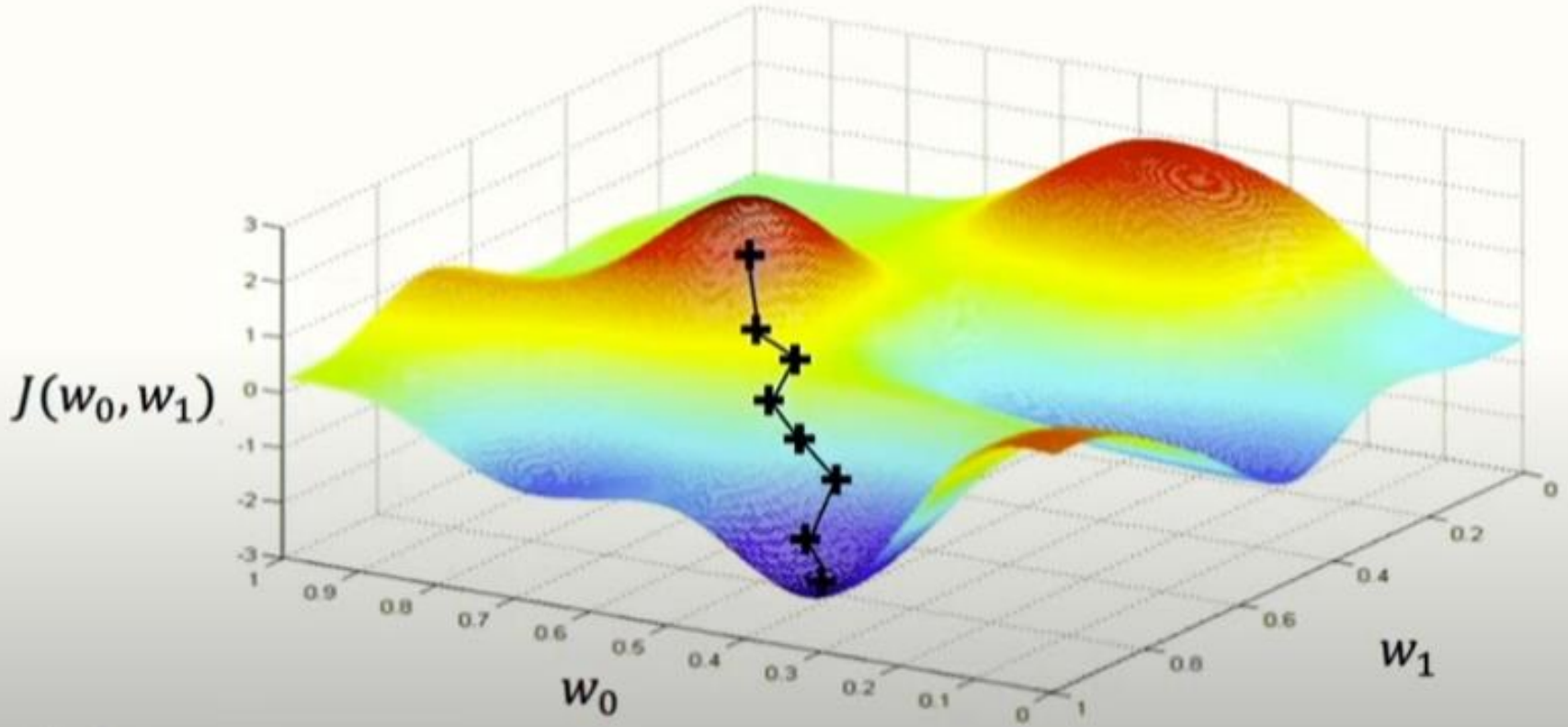
Loss Optimization

Take small step in opposite direction of gradient



Gradient Descent

Repeat until convergence

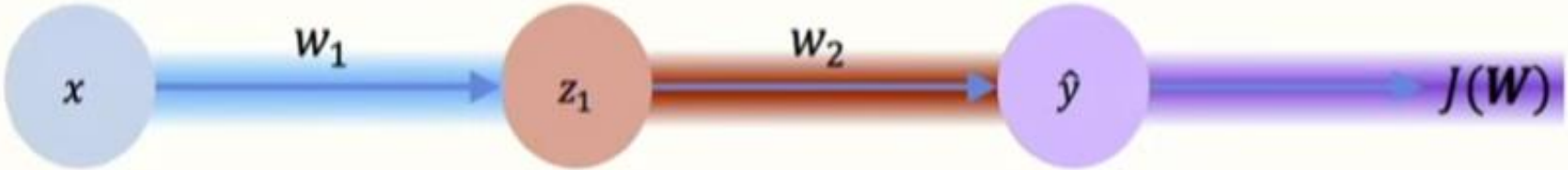


Gradient Descent

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
4. Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Return weights

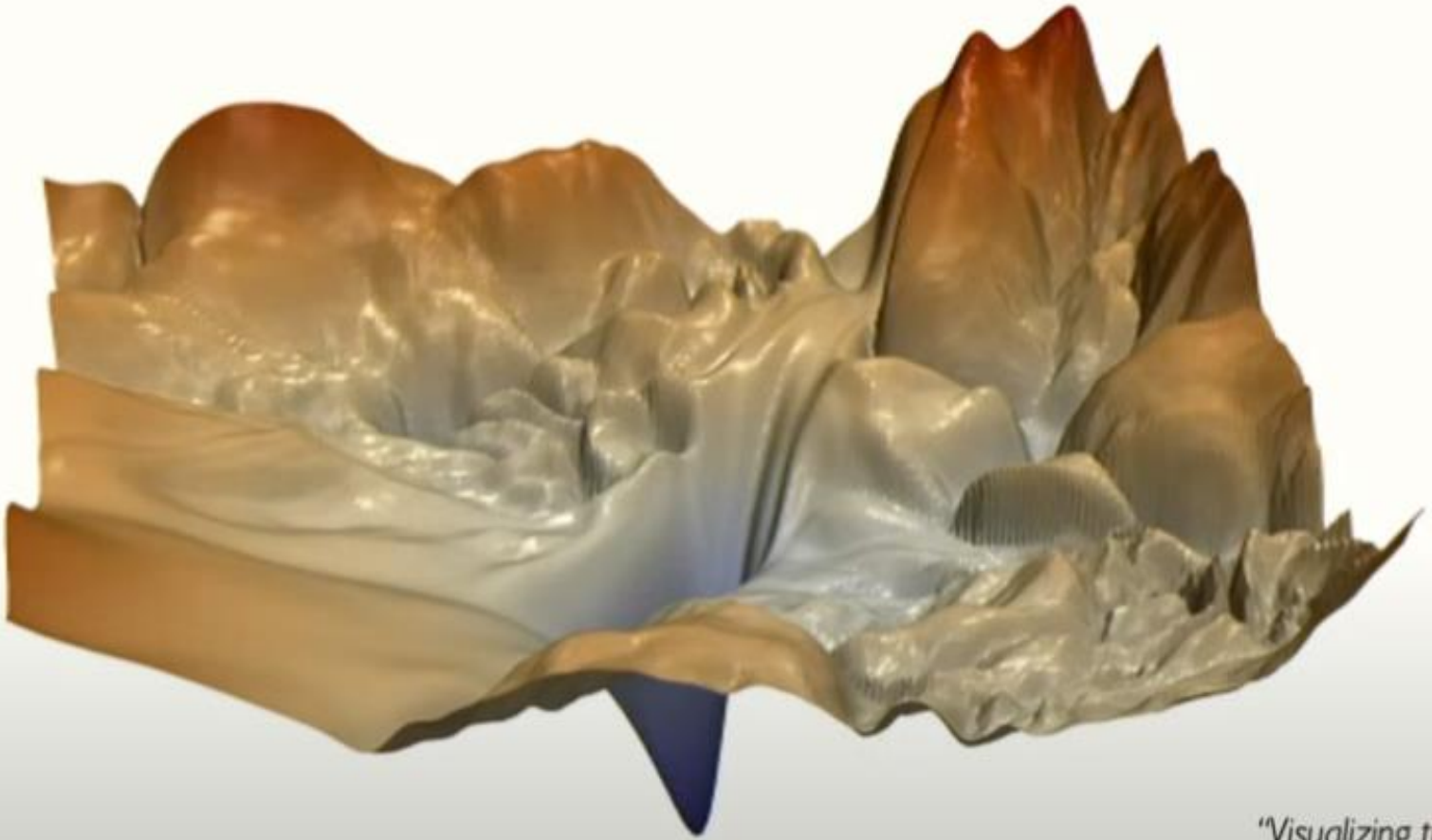
Computing Gradients: Backpropagation



$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

Repeat this for ***every weight in the network*** *using gradients from later layers*

Training Neural Networks is Difficult



"Visualizing the loss landscape of neural nets". Dec 2017.

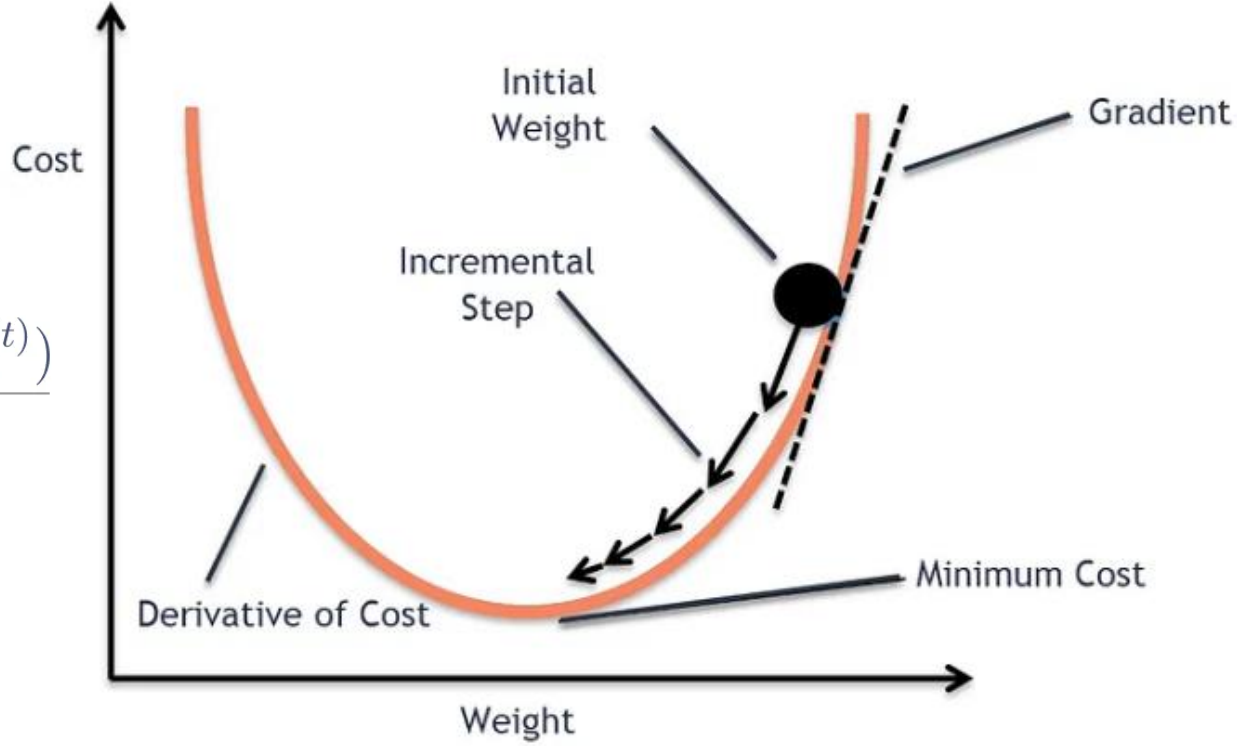
Loss Optimization

$$\underbrace{\mathrm{w}^{(t+1)}}_{\text{position of next iteration}} = \mathrm{w}^{(t)} \underbrace{- \alpha \nabla f(\mathrm{w}^{(t)})}_{\text{step}}$$

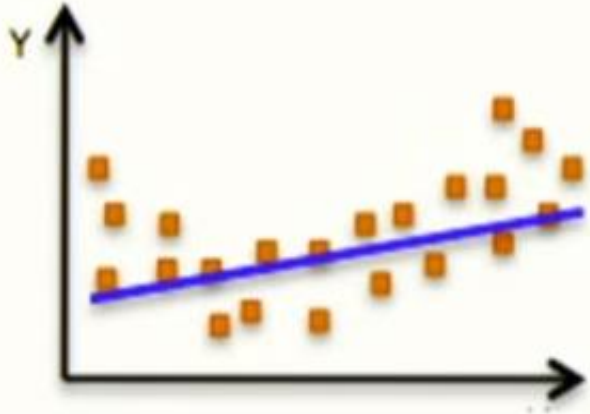
learning rate

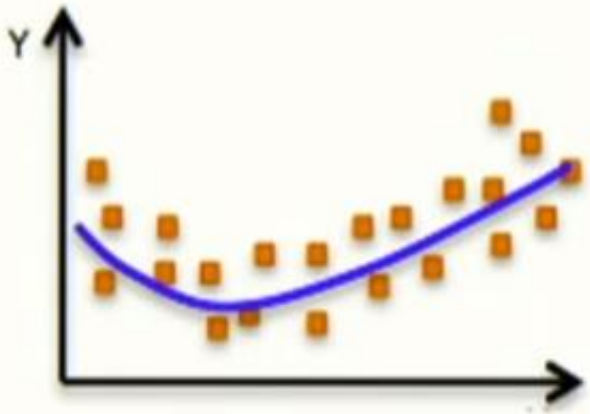
position of previous step

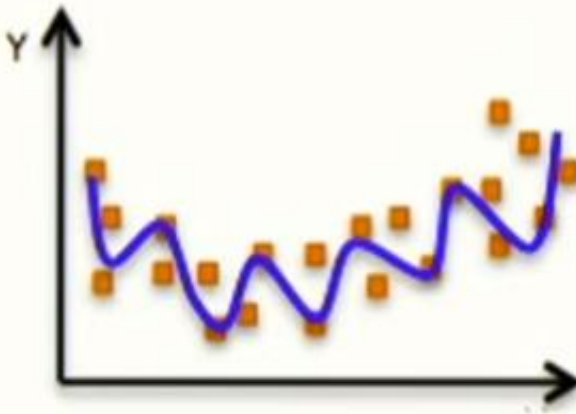
Gradient descent



The Problem of Overfitting







Underfitting
Model does not have capacity to fully learn the data

Ideal fit

Overfitting
Too complex, extra parameters does not generalize well

Regularization

What is it?

Technique that constrains our optimization problem to discourage complex models

Regularization

What is it?

Technique that constrains our optimization problem to discourage complex models

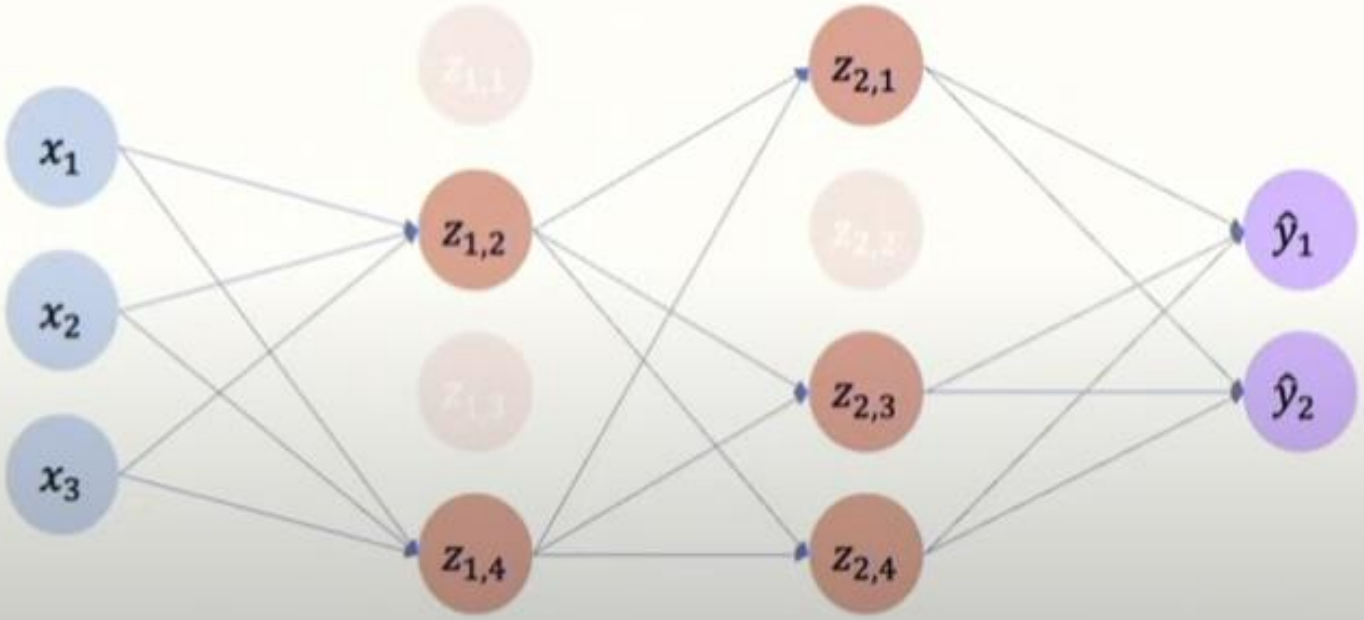
Why do we need it?

Improve generalization of our model on unseen data

Regularization 1: Dropout

- During training, randomly set some activations to 0
  - Typically 'drop' 50% of activations in layer
  - Forces network to not rely on any 1 node

```
tf.keras.layers.Dropout(p=0.5)
```



Regularization 2: Early Stopping

- Stop training before we have a chance to overfit

Deep Learning Memes

Deep Learning

What society thinks I do

What my friends think I do

What other computer scientists think I do

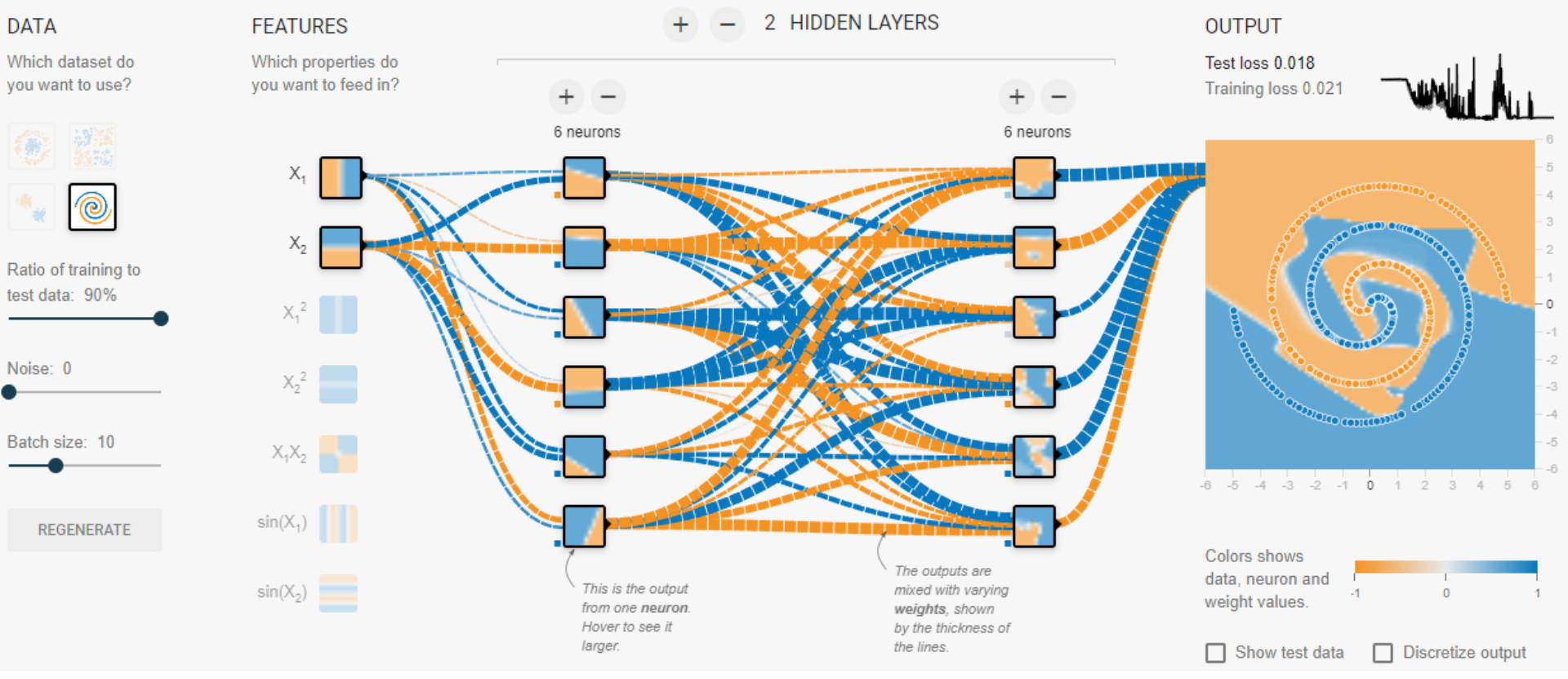
What mathematicians think I do

What I think I do

```
from theano import *
```

What I actually do

Fully Connect Artifical Neural Network



[Tensorflow Neural Network Playground]

Thank You.