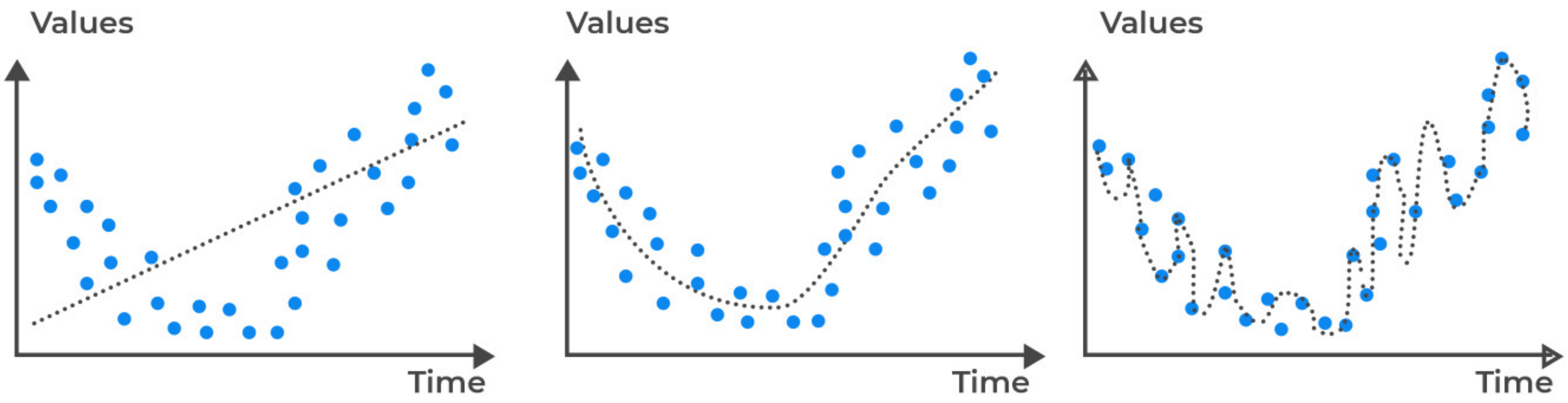
Machine Learning Part 2

Can we learn too well?

Generalization and Overfitting



Underfitted

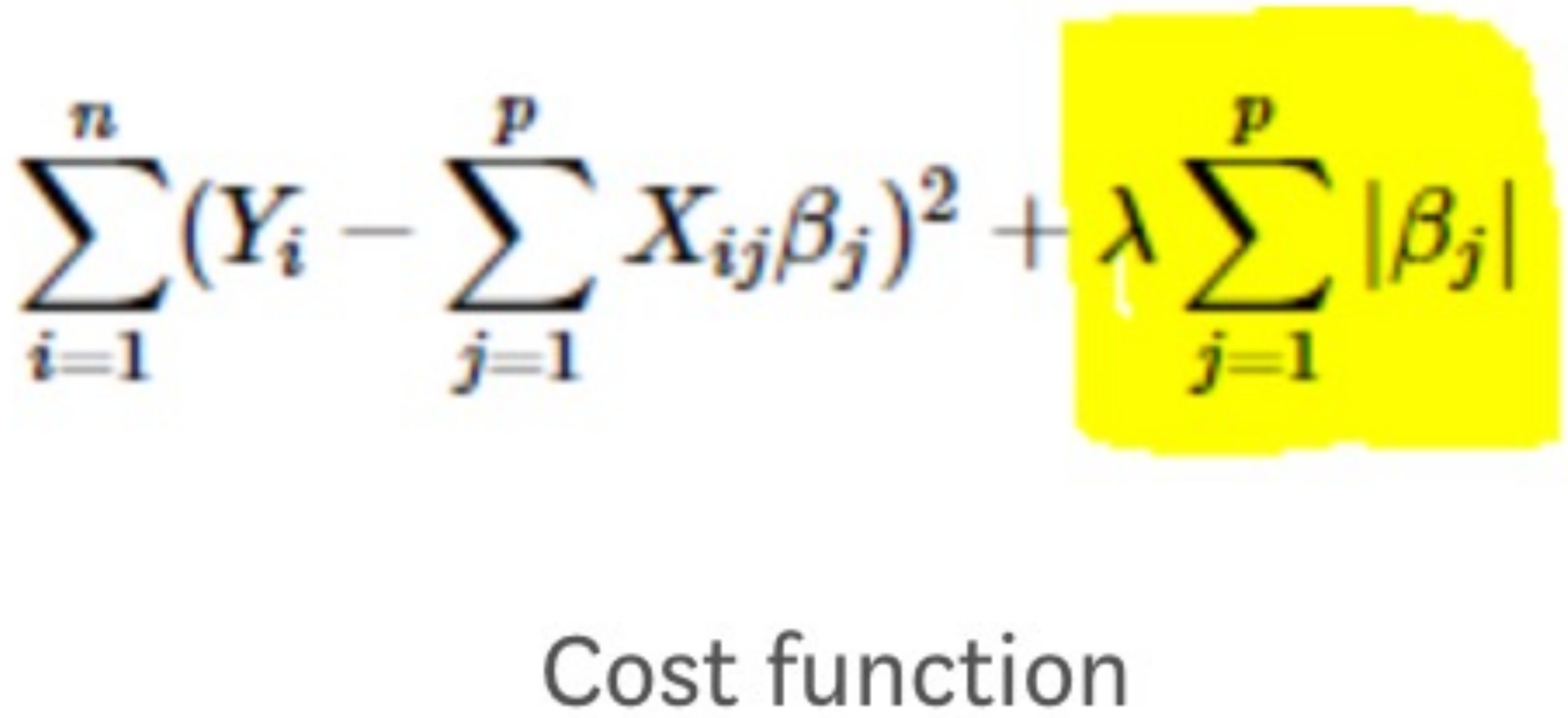
(High bias error)

Good Fit/Robust

(Balance between bias and variance)

Overfitted

(High variance error)

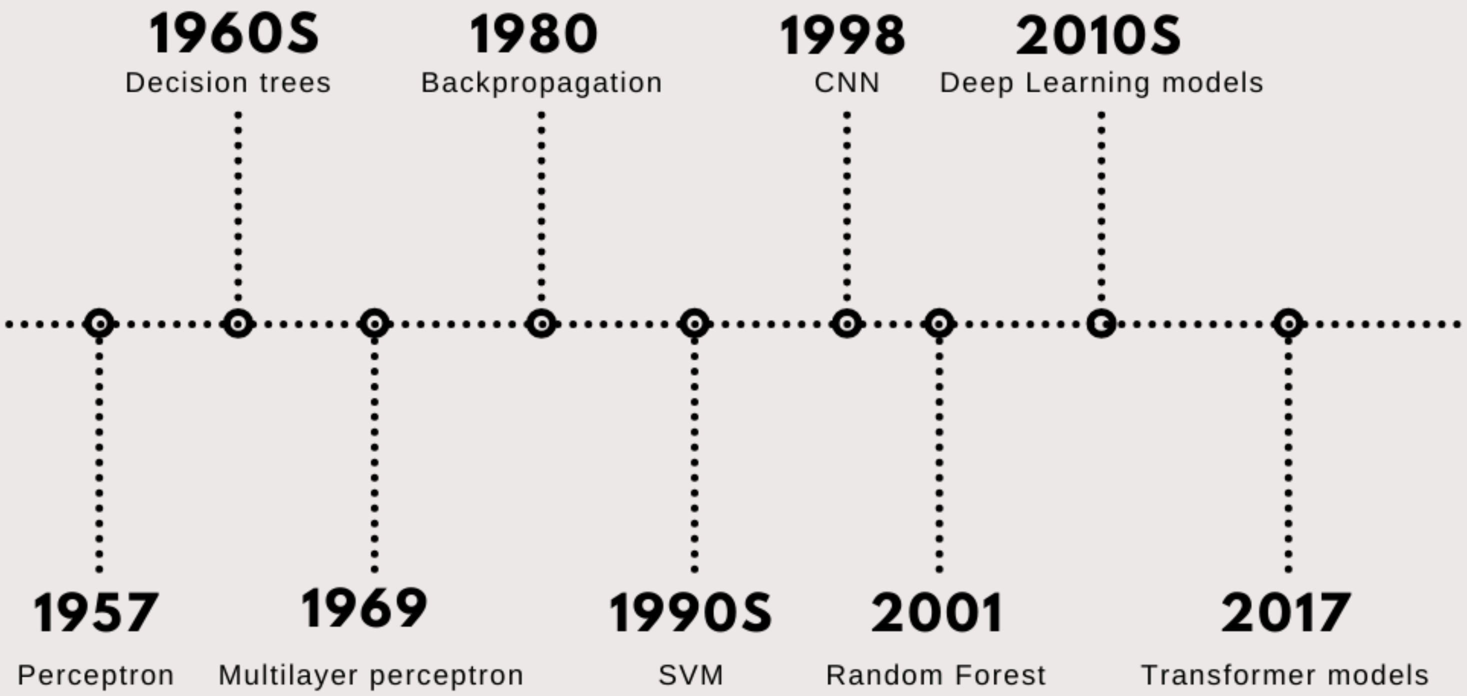


$$\sum_{i=1}^{n}(Y_i-\sum_{j=1}^{p}X_{ij}\beta_j)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

Cost function

Deep Learning

Important ML milestones



1957
Perceptron

1960S
Decision trees

1969
Multilayer perceptron

1980
Backpropagation

1990S
SVM

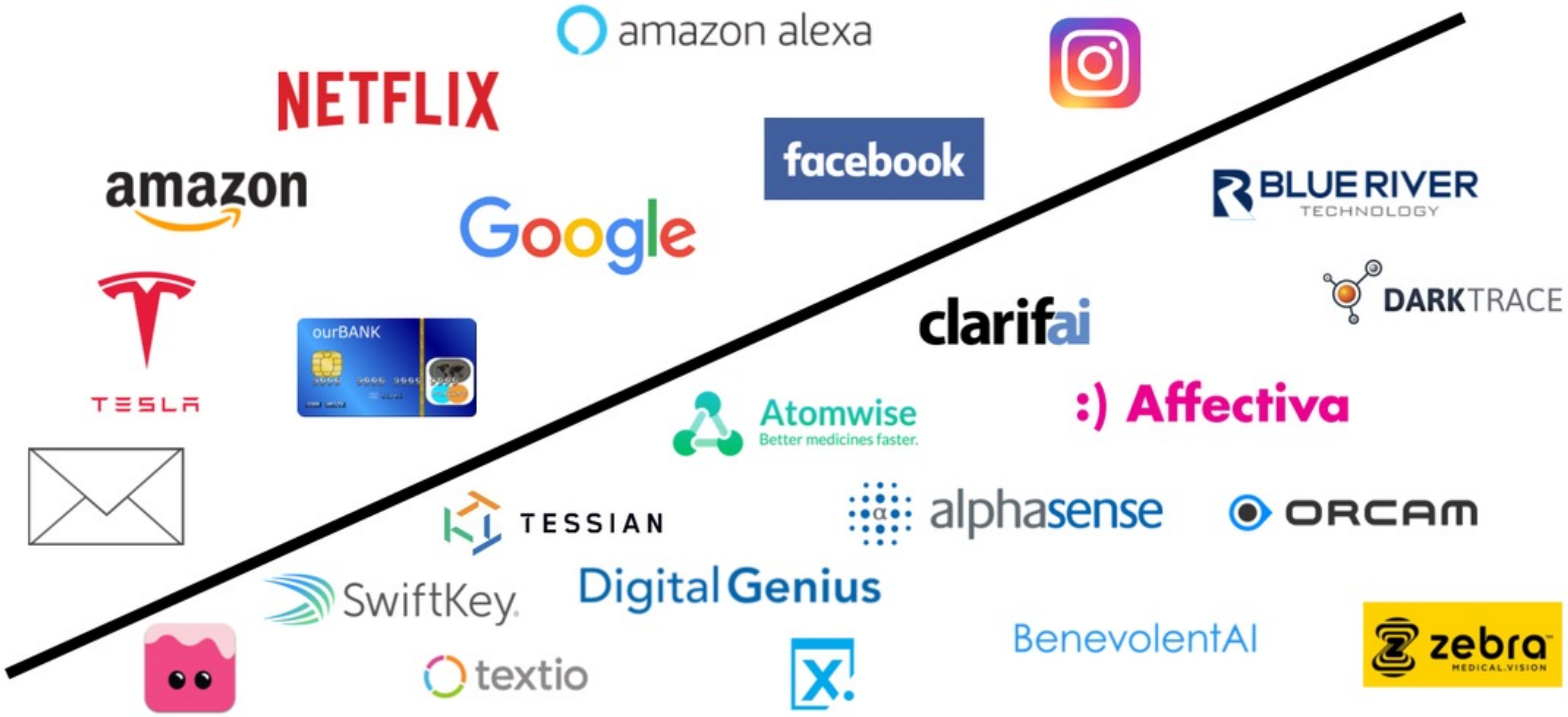
1998
CNN

2001
Random Forest

2010S
Deep Learning models

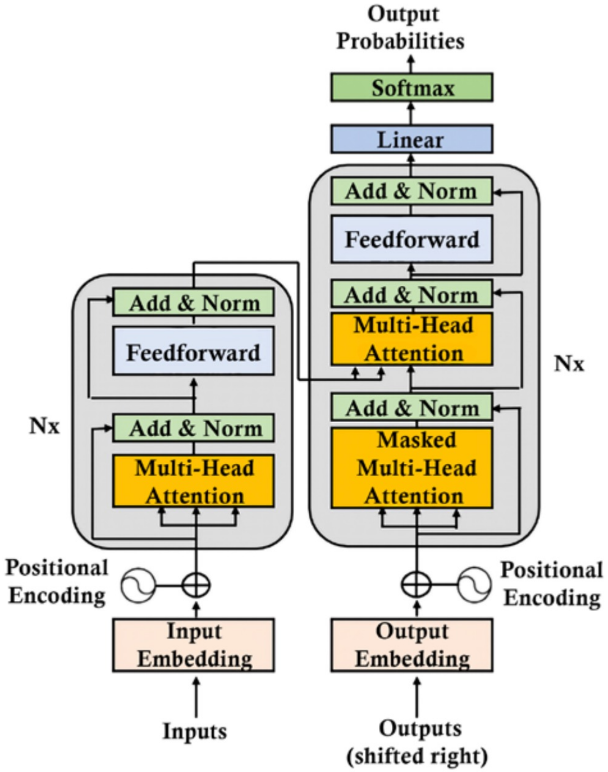
2017
Transformer models

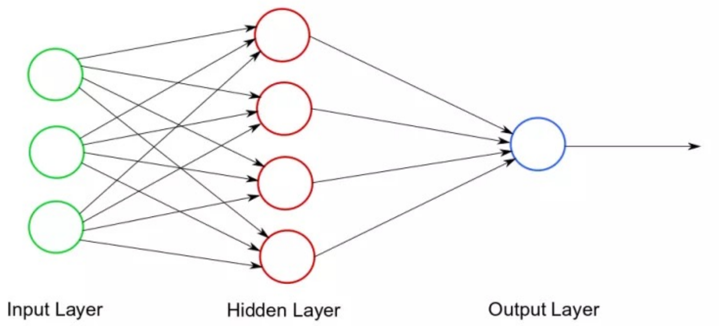
well-known

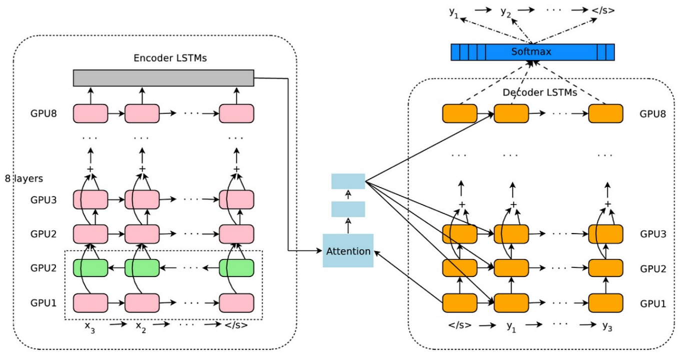


(maybe) less well-known

SELF DRIVING MODE
40

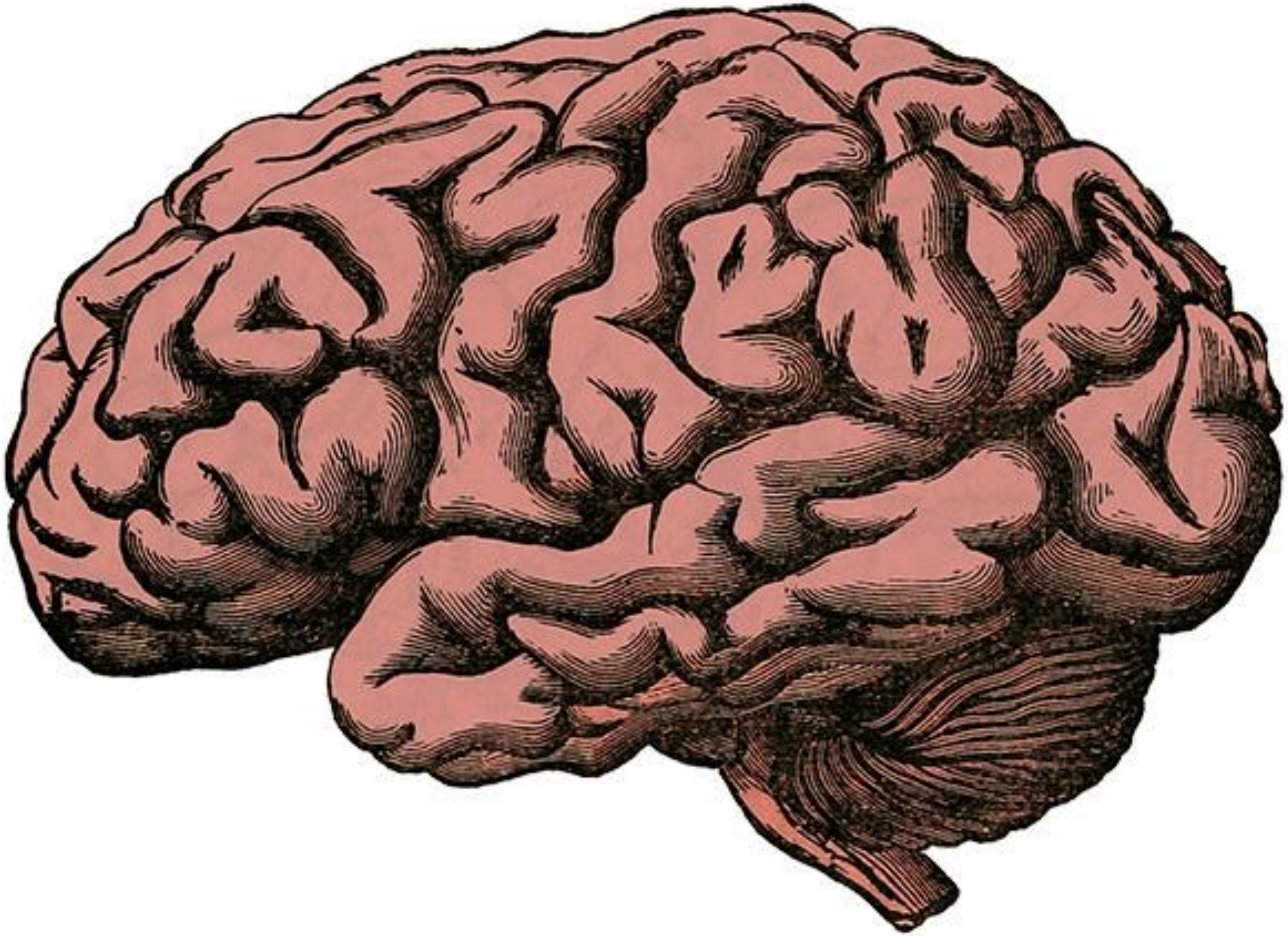

Output
Probabilities
Softmax
Linear
Add & Norm
Feedforward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feedforward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Output
Embedding
Positional
Encoding
Outputs
(shifted right)

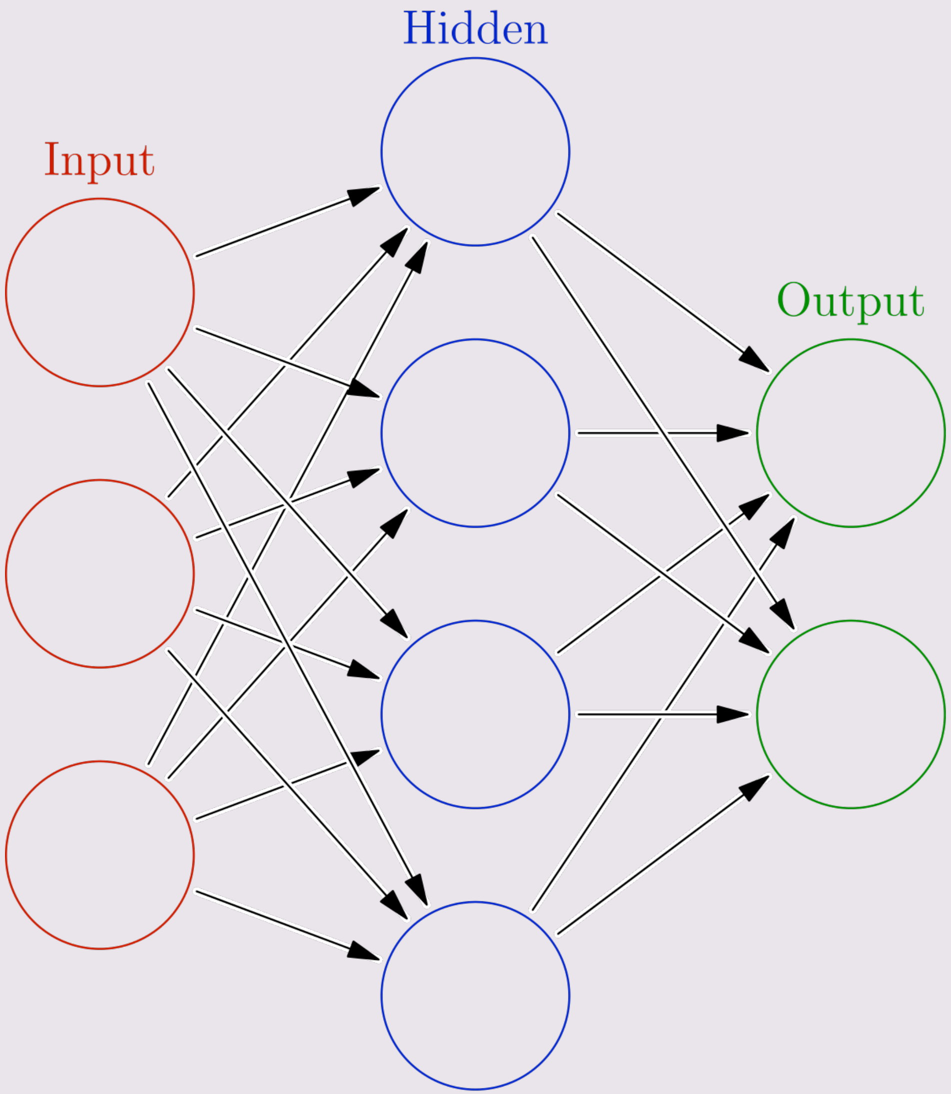

Input Layer
Hidden Layer
Output Layer


y1 → y2 → · · · → </s>
Softmax
Encoder LSTMs
Decoder LSTMs
GPU8
8 layers
GPU3
GPU2
GPU2
GPU1
x3 → x2 → · · · → </s>
Attention
GPU8
GPU3
GPU2
GPU1
</s> → y1 → · · · → y3

What is a Neural Network?

A model inspired by how brains works that has a bunch of neurons wired together.




Hidden
Input
Output

Why do we care?

Can learn complex (non linear) relationships with little human intervention

Where Deep Learning Shines?

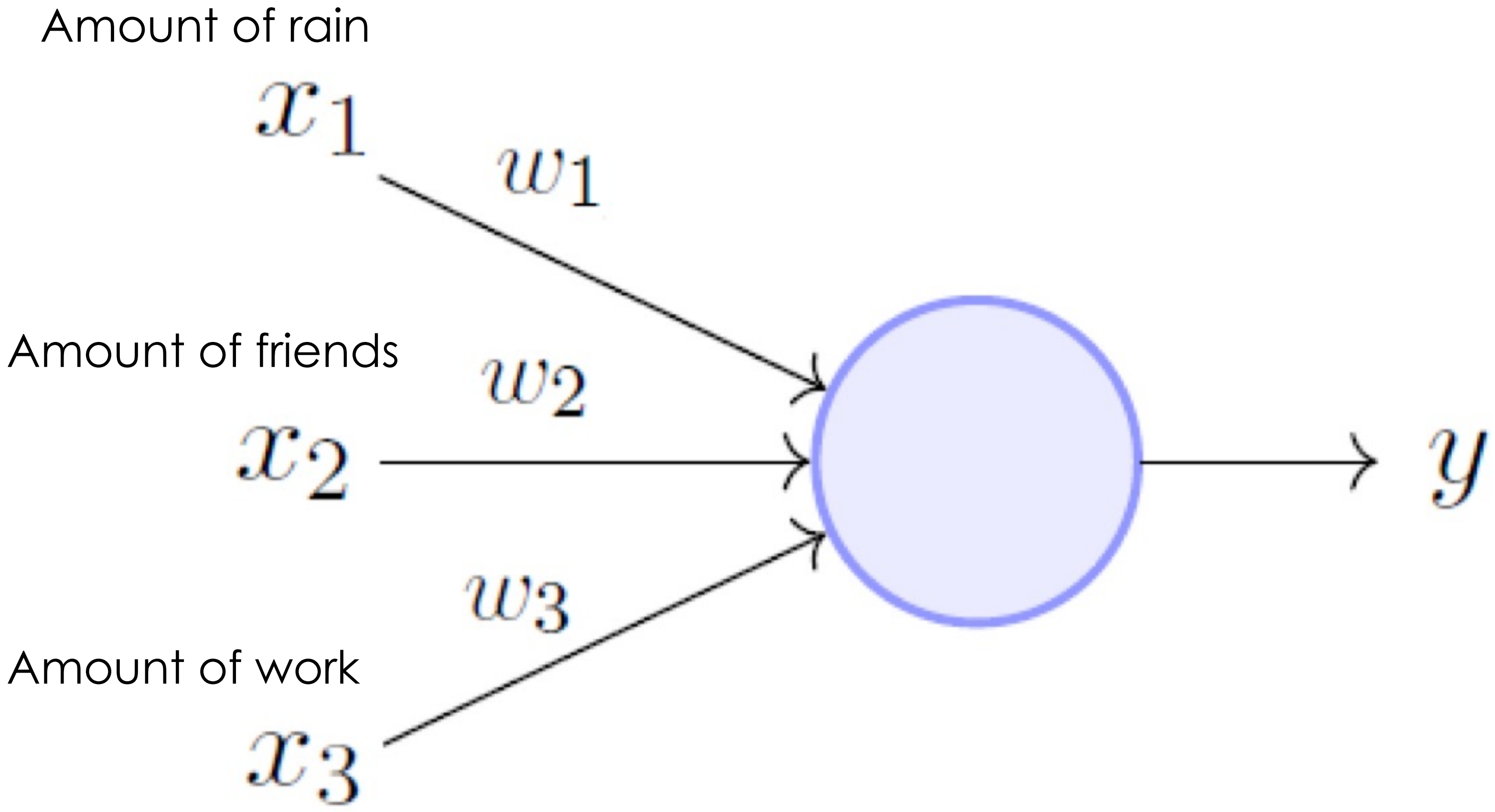
- Images
- Speech
- Text
- Game Playing

How to Build Neural Networks?

Perceptron

Thought Experiment?

- You want to decide whether or not to **darty**, and you only like to darty when 1) it's not raining, 2) you don't have much work or **(most importantly)** 3) when you have enough friends dartying. How can we represent this model as a perceptron (i.e. inputs, weights, biases)?



Amount of rain

x_1

w_1

Amount of friends

w_2

x_2

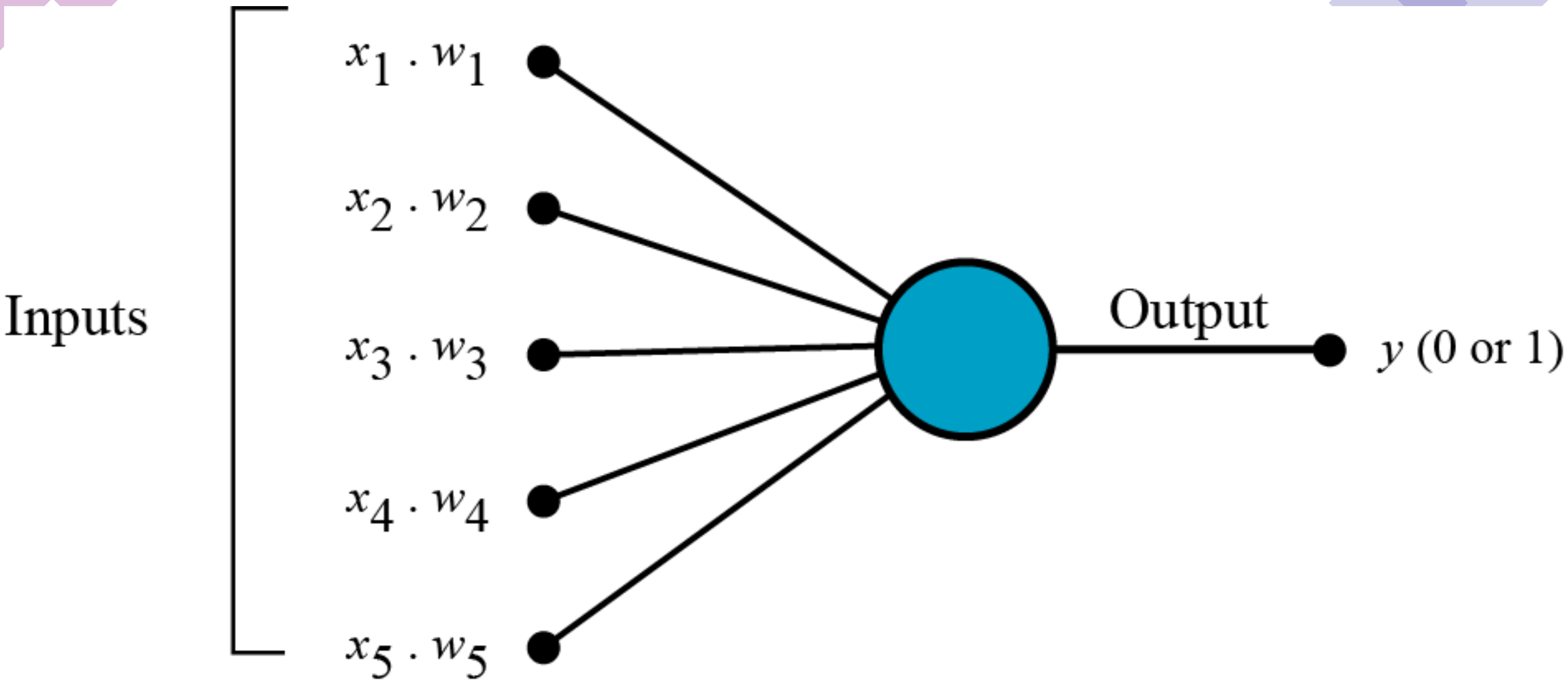
y

w_3

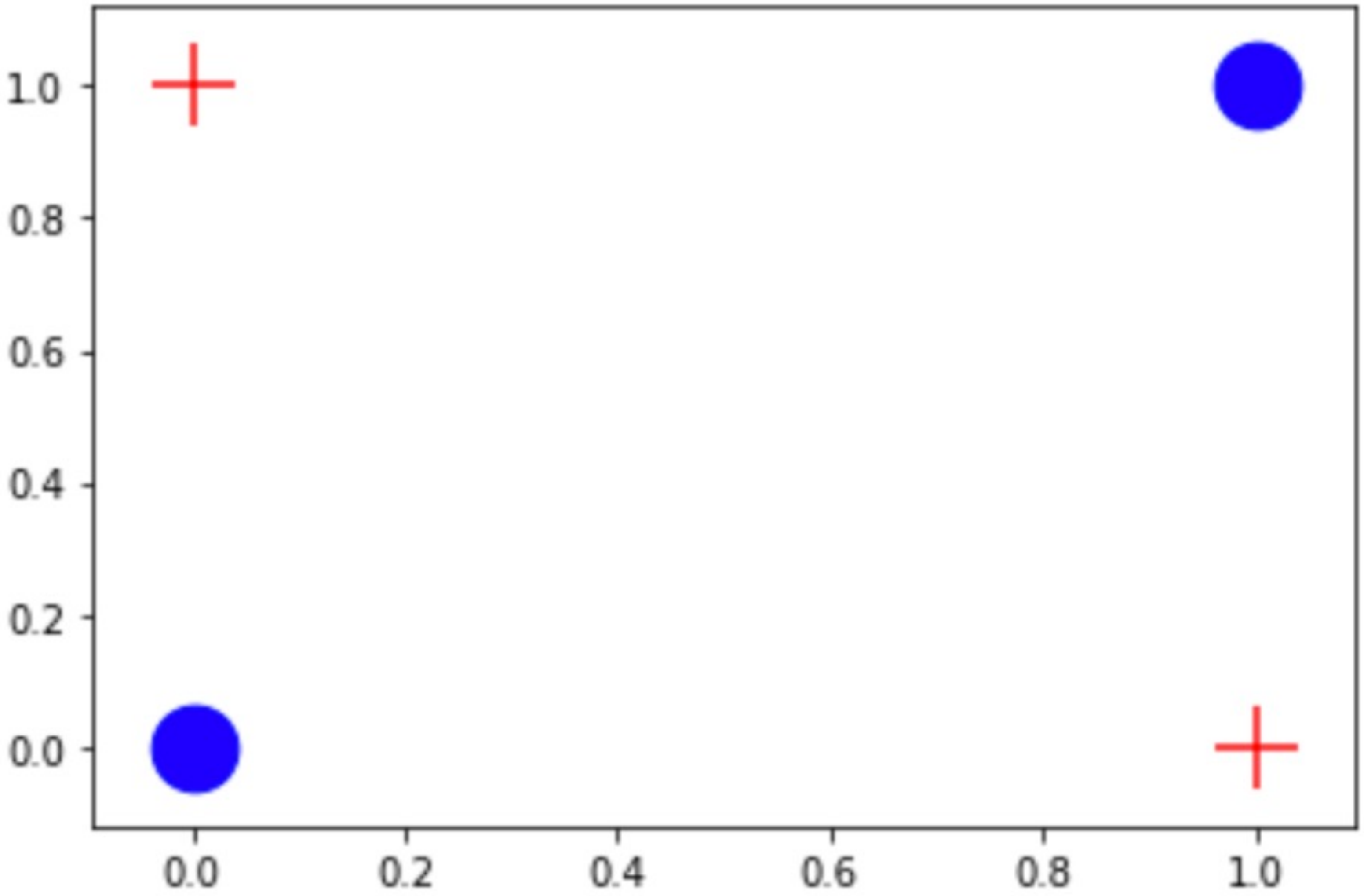
Amount of work

x_3

Perceptron Model (Minsky-Papert in 1969)

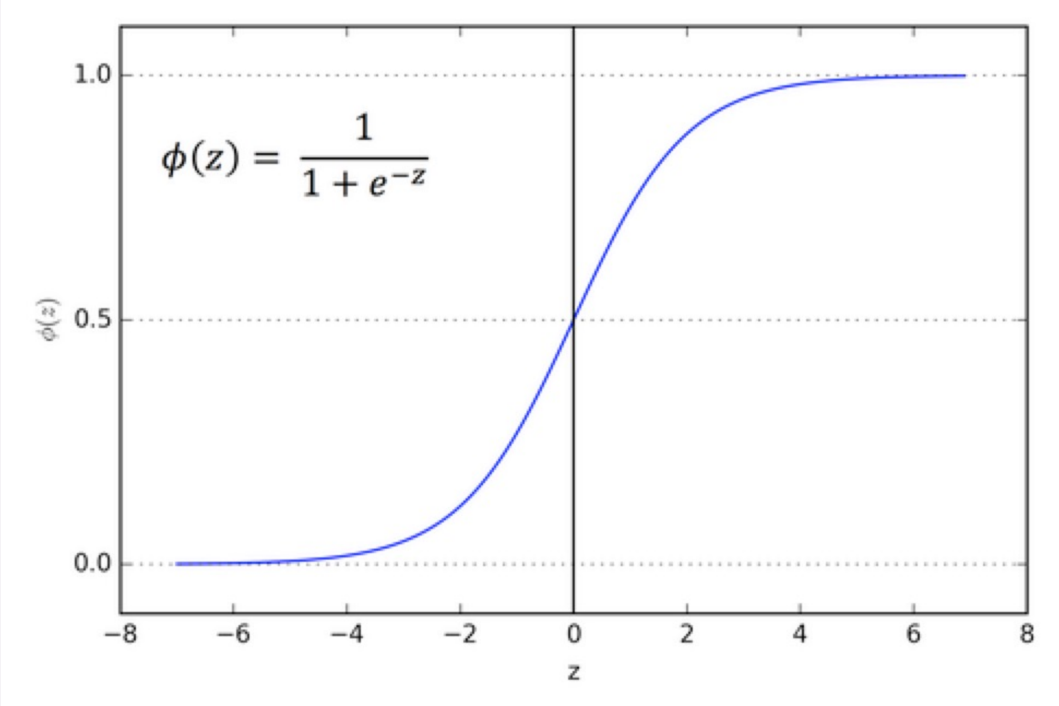

Inputs
$x_1 \cdot w_1$
$x_2 \cdot w_2$
$x_3 \cdot w_3$
$x_4 \cdot w_4$
$x_5 \cdot w_5$
Output
y (0 or 1)

What’s Wrong with the Perceptron?

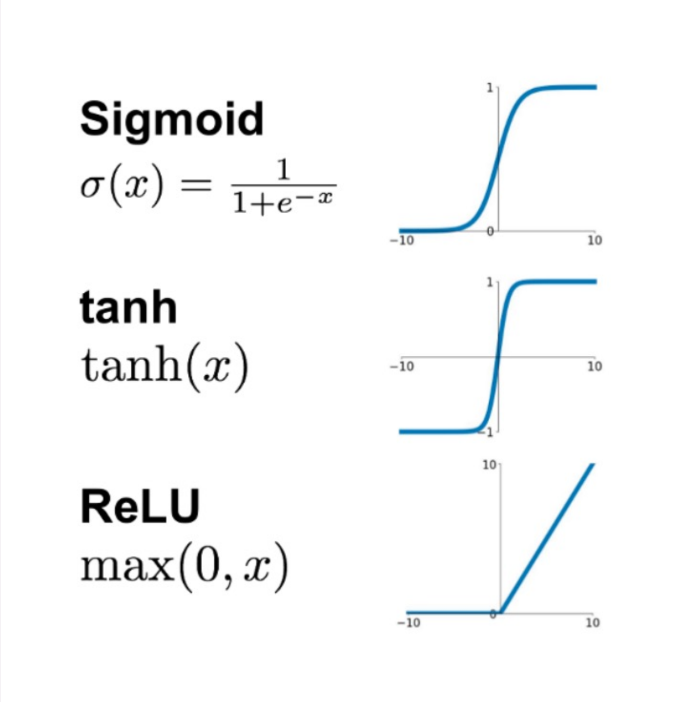

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Activation Functions

Function that takes a neuron output and says whether it should fire.



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



Sigmoid

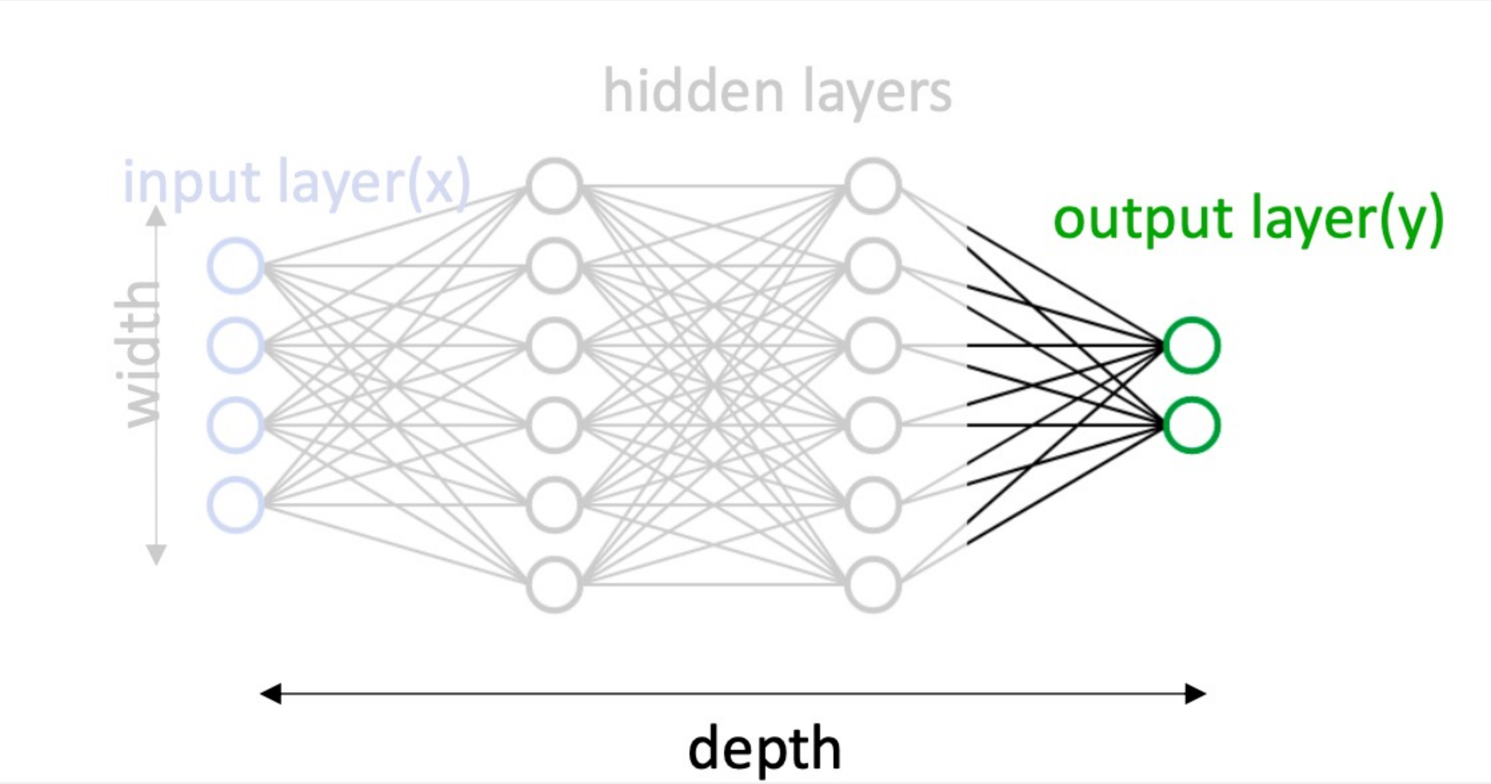
$\sigma(x) = \frac{1}{1+e^{-x}}$

tanh

$\tanh(x)$

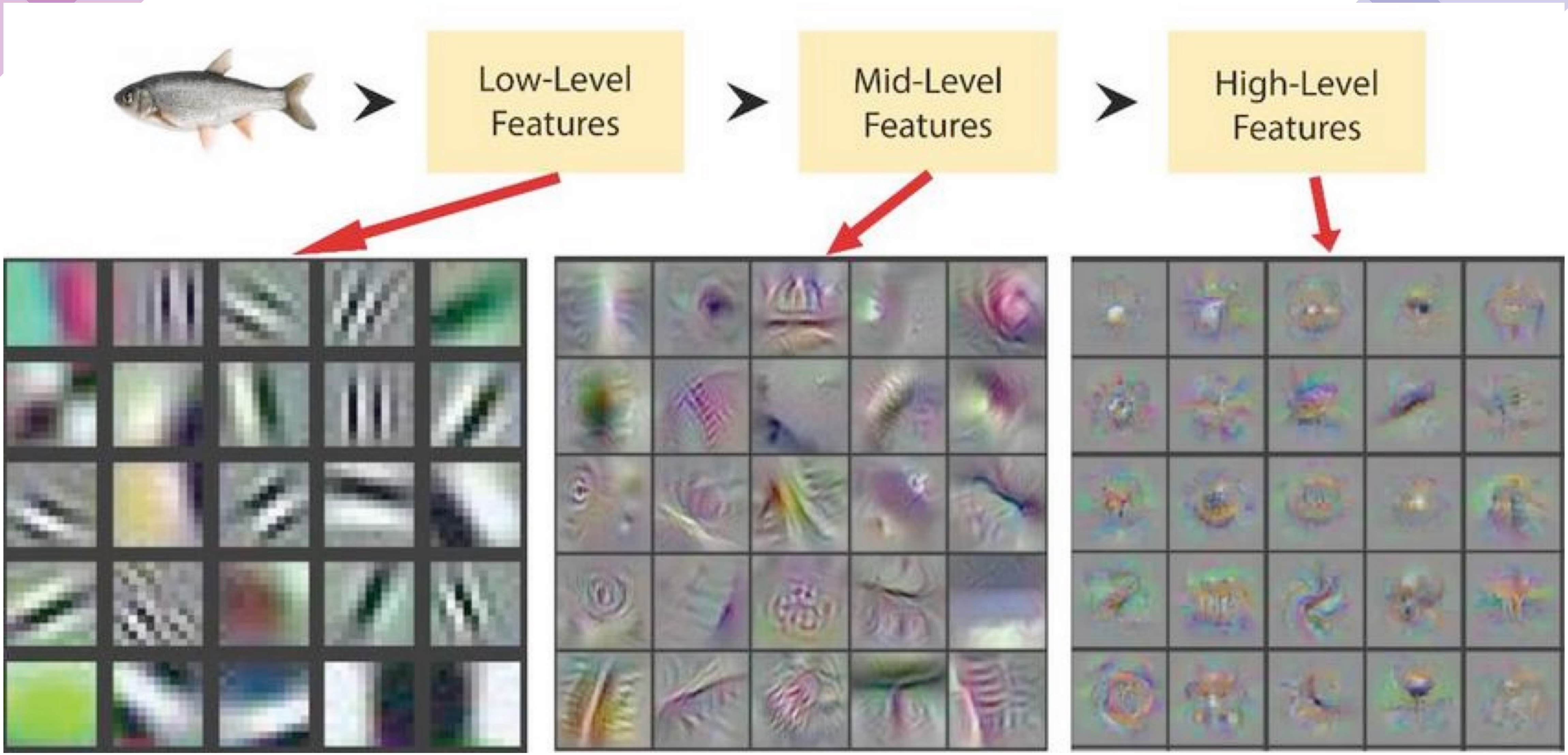
ReLU

$\max(0, x)$



Binary Classification – Sigmoid: Approximates over 0, 1
Multi-Class Classification – Softmax: a probability over all labels
Regression – Linear: no activation, so output can be in any range

Can Neural Networks learn anything?


Low-Level Features
Mid-Level Features
High-Level Features

How do we classify new instances?

How do we classify new instances?

By inputting our instances to the neural network and letting information flow to the output layer.

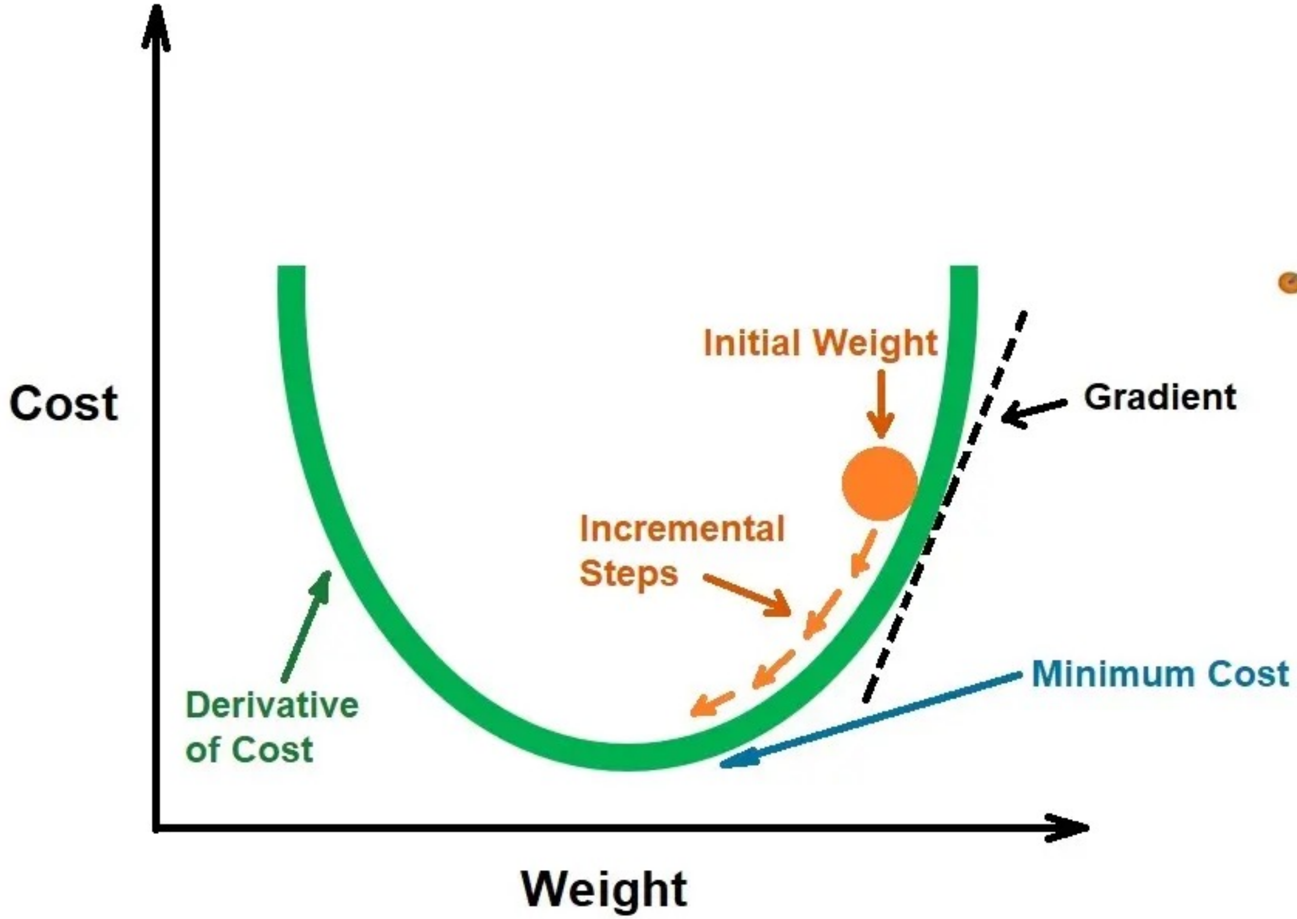
How do we find optimal network parameters?

By adjusting our weights and biases until our model stops making mistakes.

We still need the same components from ML –

Backpropogation

Gradient Descent



Deep Learning Training Process

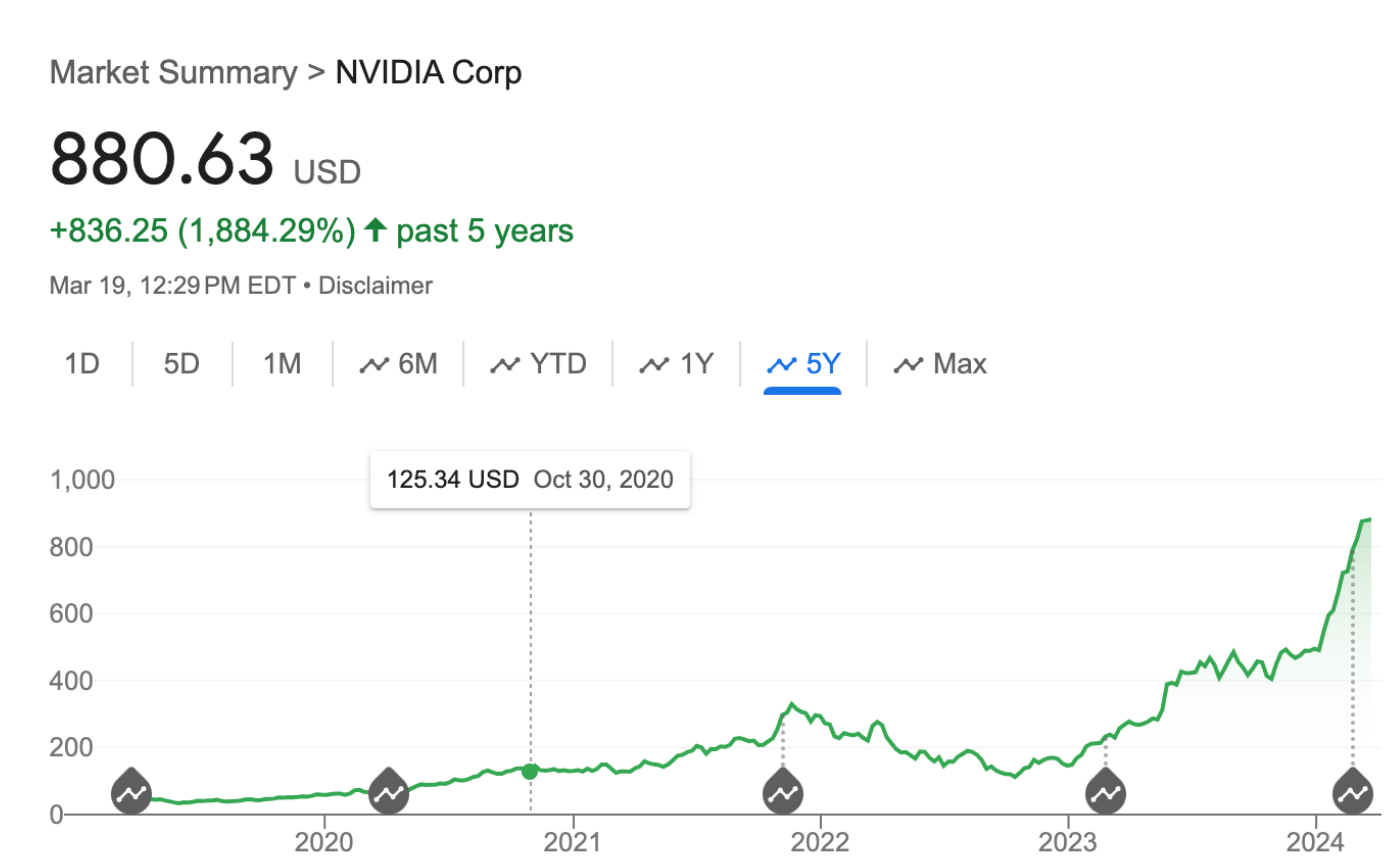
1. Initialize random parameters
2. Make prediction using train computing data to get loss function (forward pass)
3. Use **backpropagation** to gradients (backwards pass)
4. Use gradient descent to optimally update the parameters by a tiny amount
5. Repeat 2-4
6. Model converges

Problem with Implementations

1. Implementing backprop, gradient descent
2. Doing complicated calculus in high dimensions
3. Utilizing GPUs / CPU optimizations
4. Exporting to meaningful formats

Deep Learning Frameworks

1. Implementation of models & algorithms
2. GPU parallelization & CPU optimizations
3. Standardized model formats



Market Summary > NVIDIA Corp

880.63 USD

+836.25 (1,884.29%) ↑ past 5 years

Mar 19, 12:29 PM EDT • Disclaimer

PyTorch