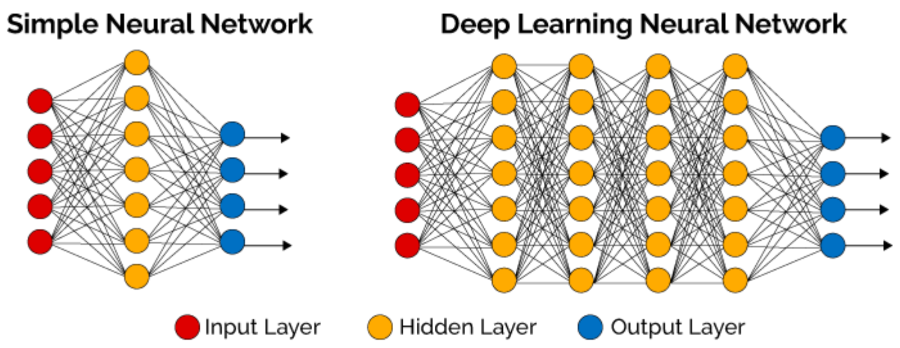
Introduction to Deep Learning

by NTU MLDA@EEE

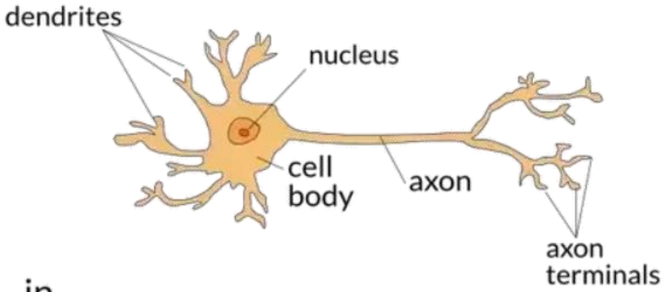
What is Deep Learning?

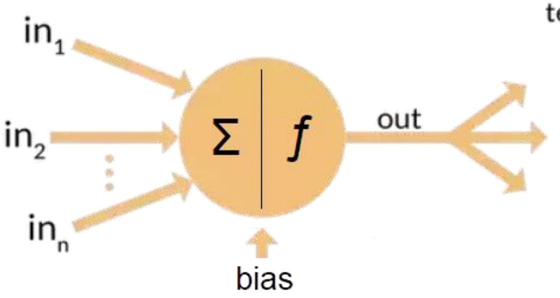


- Variety of definitions
- Inspired by human brains
- Usually in the form of **neural networks**
- Defined by a **massive number of parameters**
- Able to **represent complex data** / relationships
- Made possible due to advancement in computing technologies

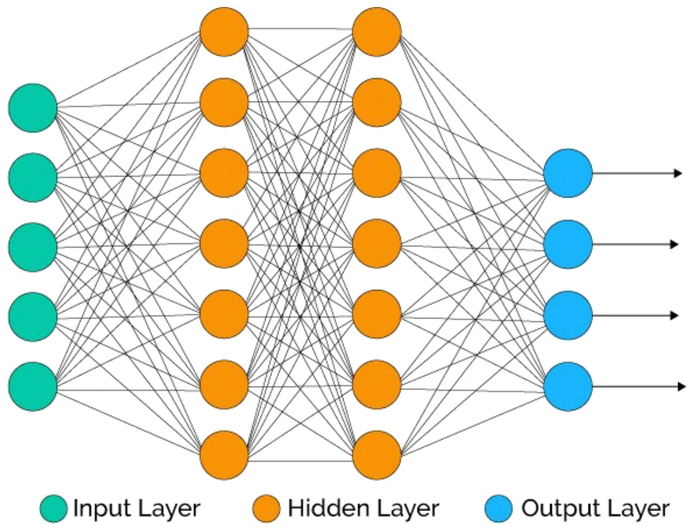
What is a neural network?

- Biologically inspired computing systems
- In human brain:
  - Dendrites receive signals from connected neurons
  - Received signal is processed and sent down the axon to neighboring neurons
- In artificial neural networks
  - A perceptron receives input (i.e. numbers) from other perceptrons
  - Inputs are transformed by a function and sent to connected perceptrons





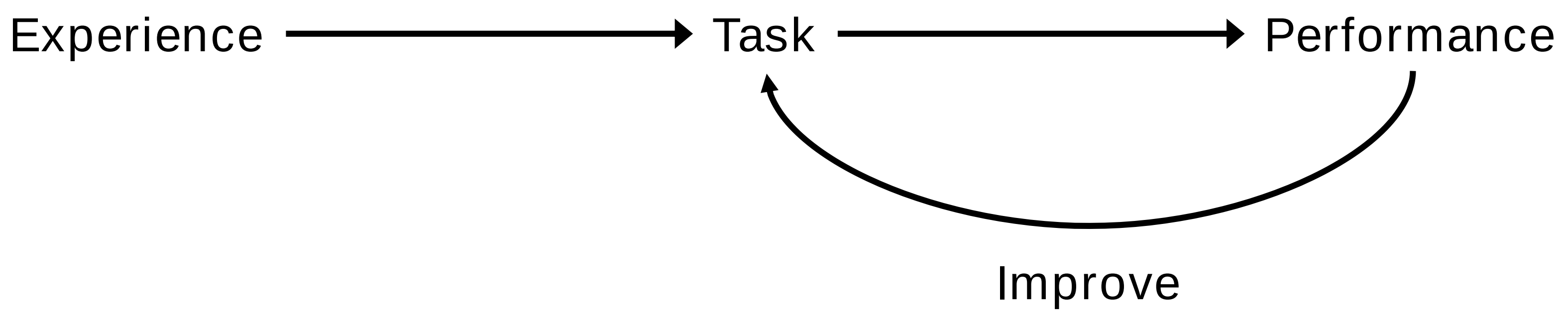
Artificial vs Biology



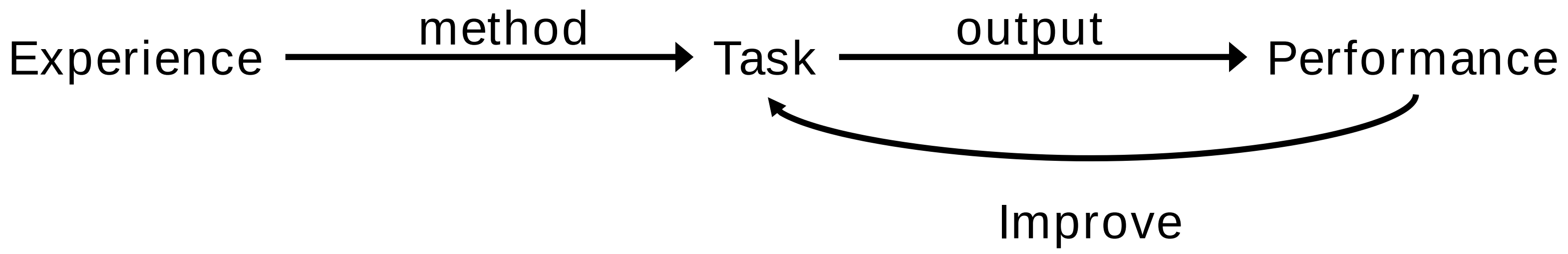
- Artificial neural networks are different from the human brain
- Human brain:
    - Neurons are connected in complex networks
    - About 86 billion neurons and >100 trillion connections
    - Dynamic in nature
- Artificial neural networks
    - Perceptrons are arranged in layers
    - Operates and learn in a predefined manner (and the size does not change)

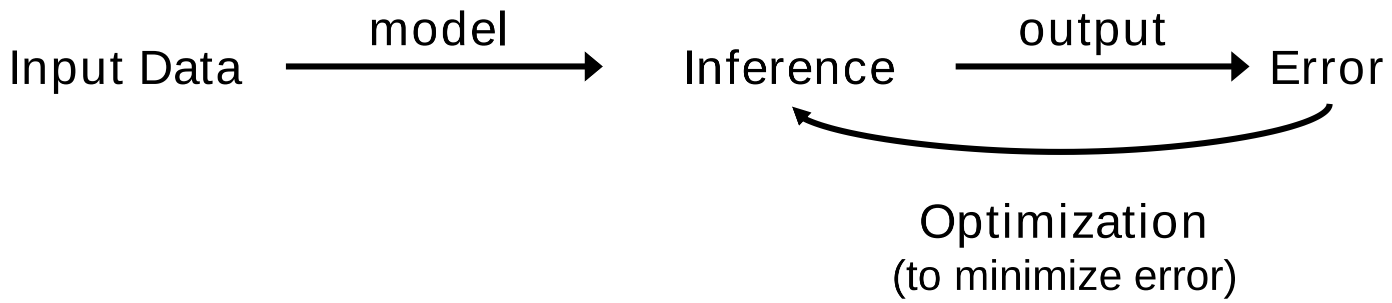
How do machines learn?

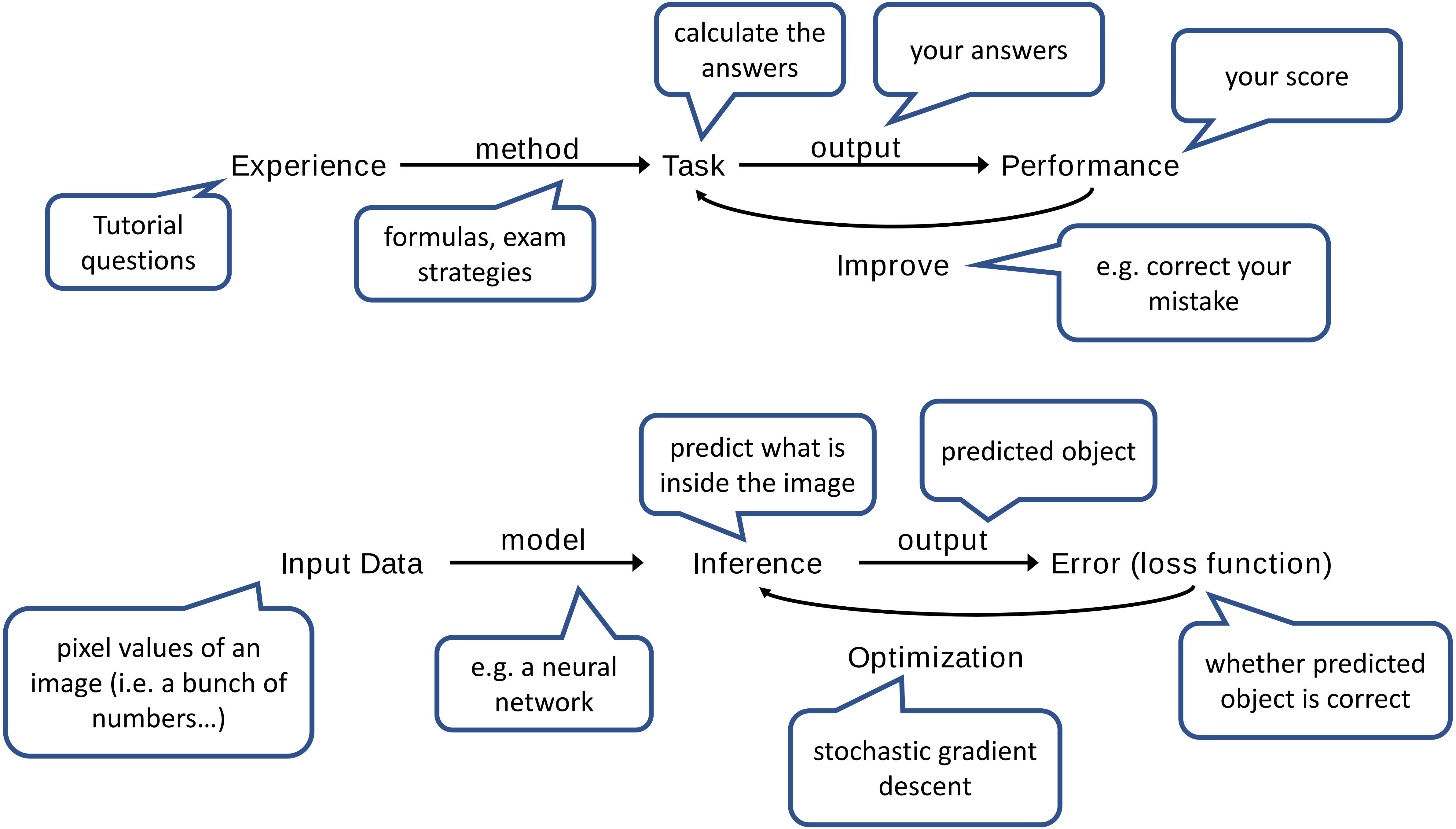
- A program is said to learn from experience E with respect to task T and performance measure P, if it’s performance at tasks in T, as measured by P, improves with experience E.



Some Terminologies

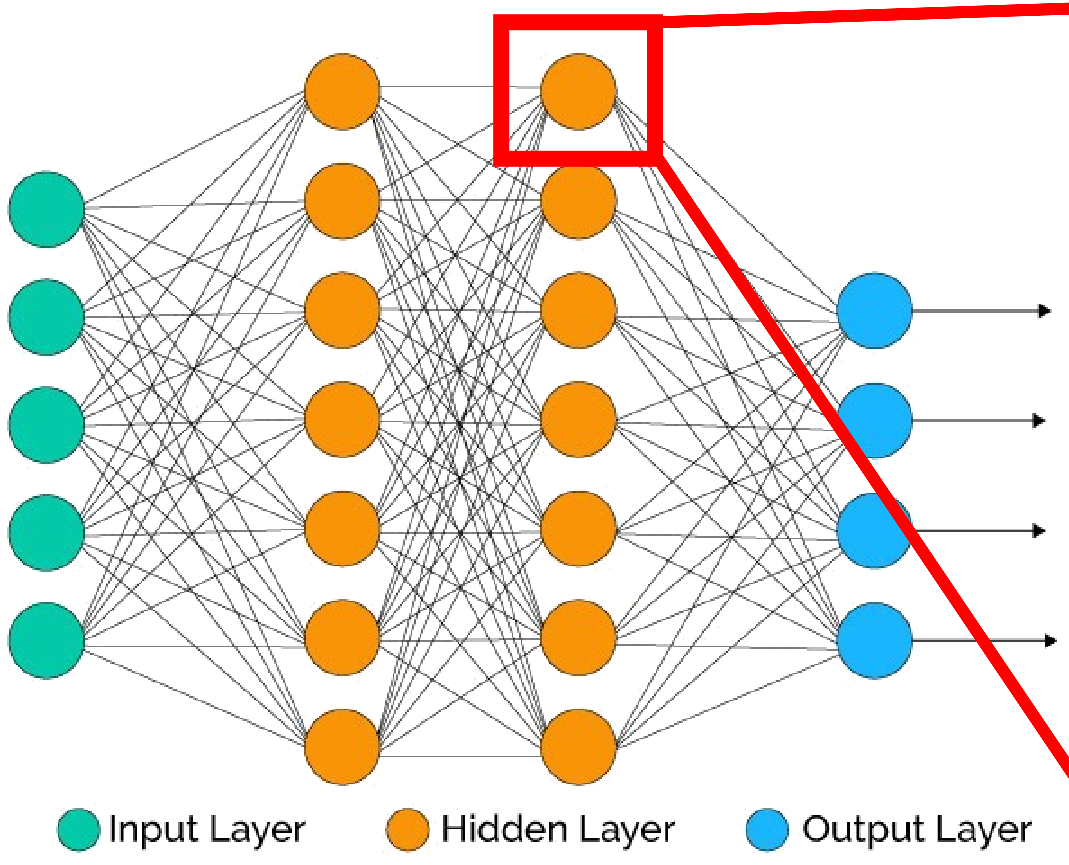


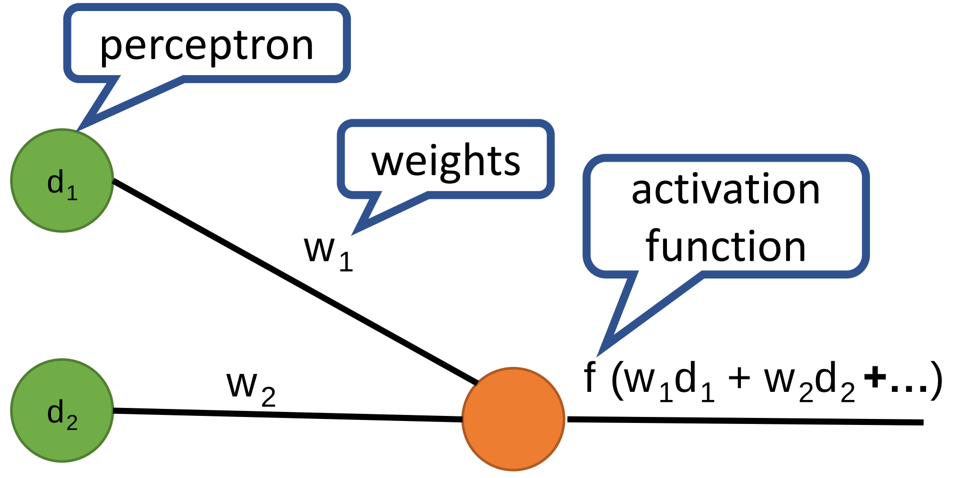



calculate the answers
your answers
your score
Experience
method
Task
output
Performance
Tutorial questions
formulas, exam strategies
Improve
e.g. correct your mistake
predict what is inside the image
predicted object
Input Data
model
Inference
output
Error (loss function)
pixel values of an image (i.e. a bunch of numbers…)
e.g. a neural network
Optimization
stochastic gradient descent
whether predicted object is correct

Neural Network Deep Dive

Structure of Neural Networks



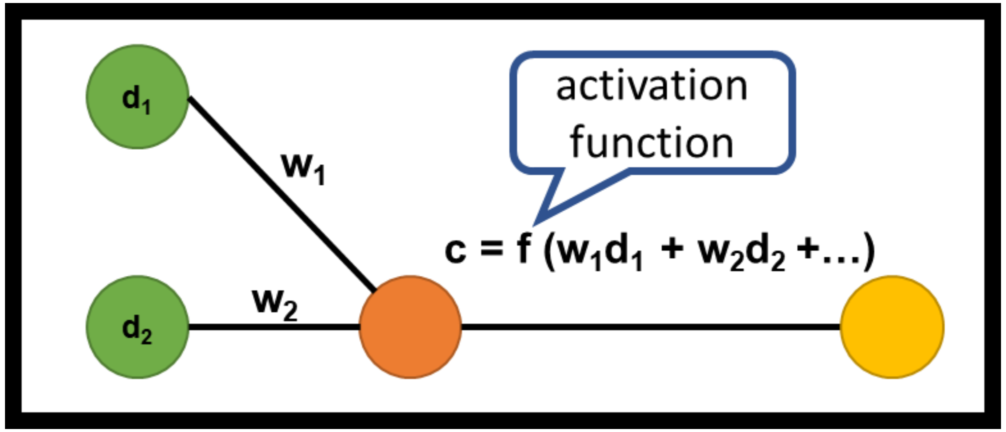


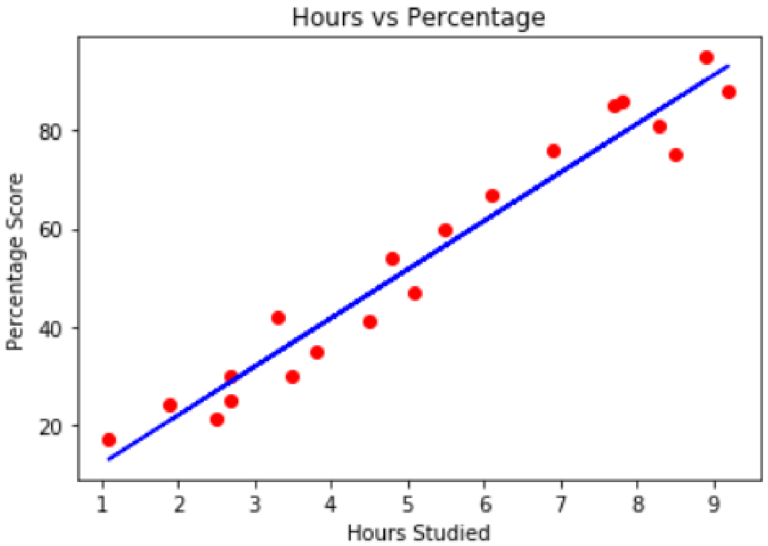
In most applications:

- connections between perceptrons are predefined
- activation function is predefined
- behavior of neural network is totally determined by its weights

Neural Network Deep Dive

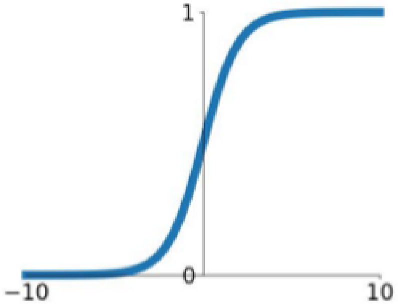
Why do we need an activation function?





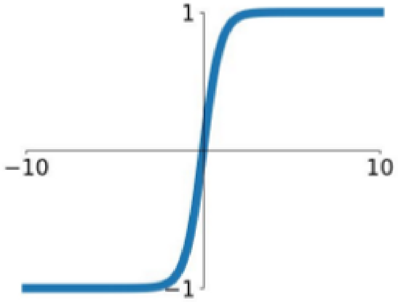
Sigmoid

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



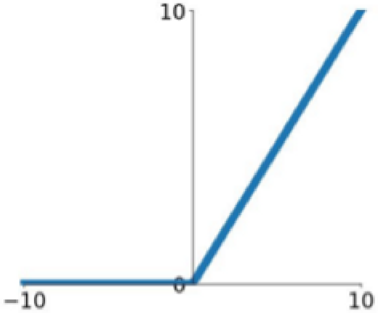
tanh

$$\tanh(x)$$



ReLU

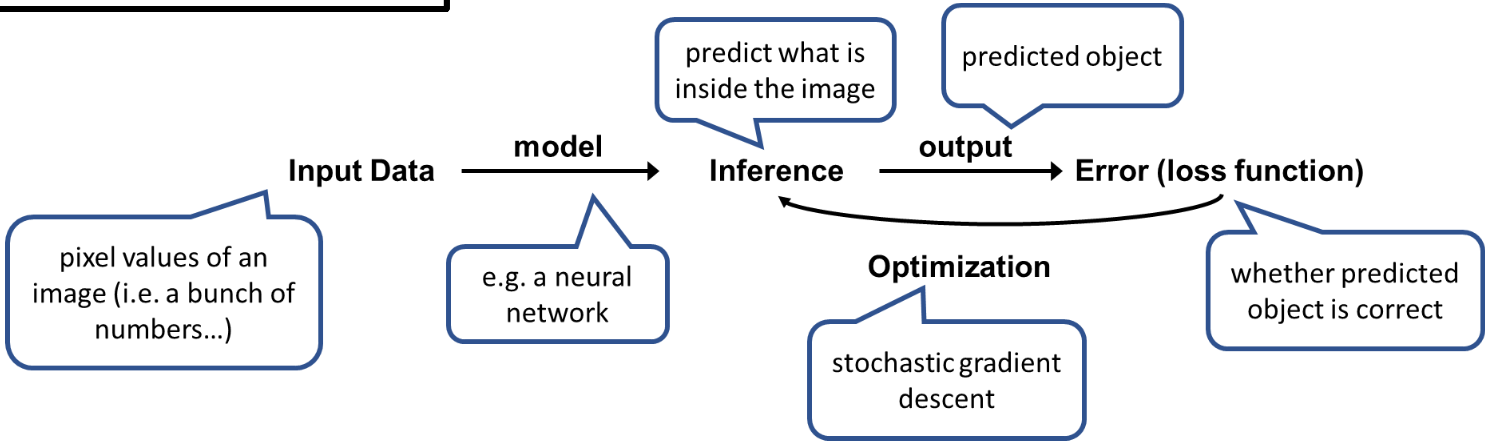
$$\max(0, x)$$



Goal of Neural Networks

- In supervised learning, we deal with parametric models
    - Parameters: variables that summarizes a set of data (the number of parameters is predefined by architecture of the neural network, but the values are learnt through optimization)
    - Hyperparameters: pre-chosen variables that define how the neural network learn (e.g. how fast it learns etc.)
- In short, the objective of training a neural network is to find a set of parameters that gives the minimum error

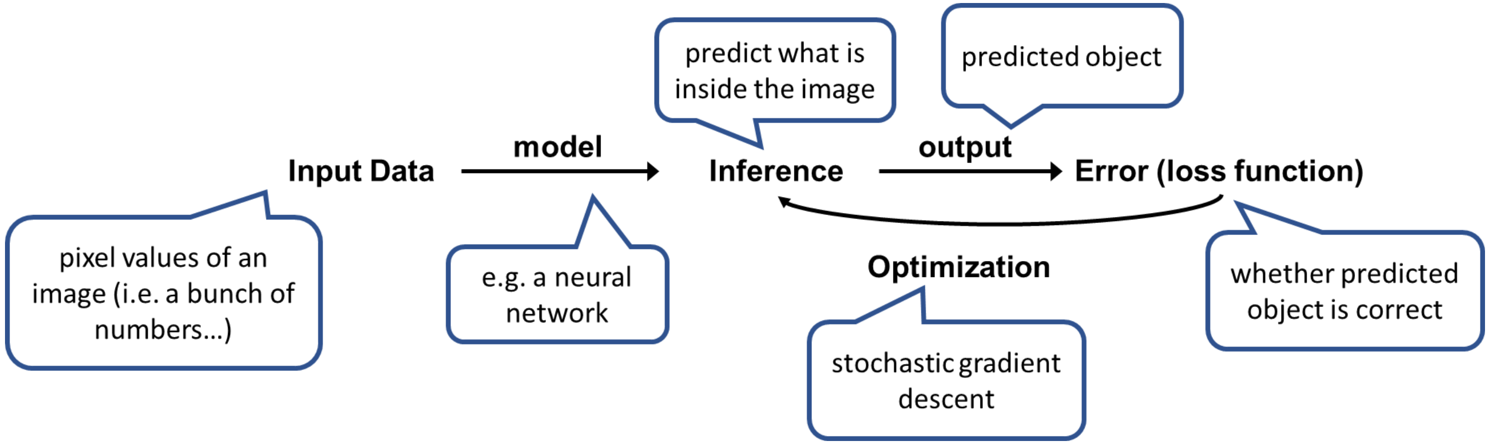
$$\boxed{\underset{E}{\operatorname{argmin}}(\theta_1, \dots, \theta_n)}$$



Neural Network Deep Dive

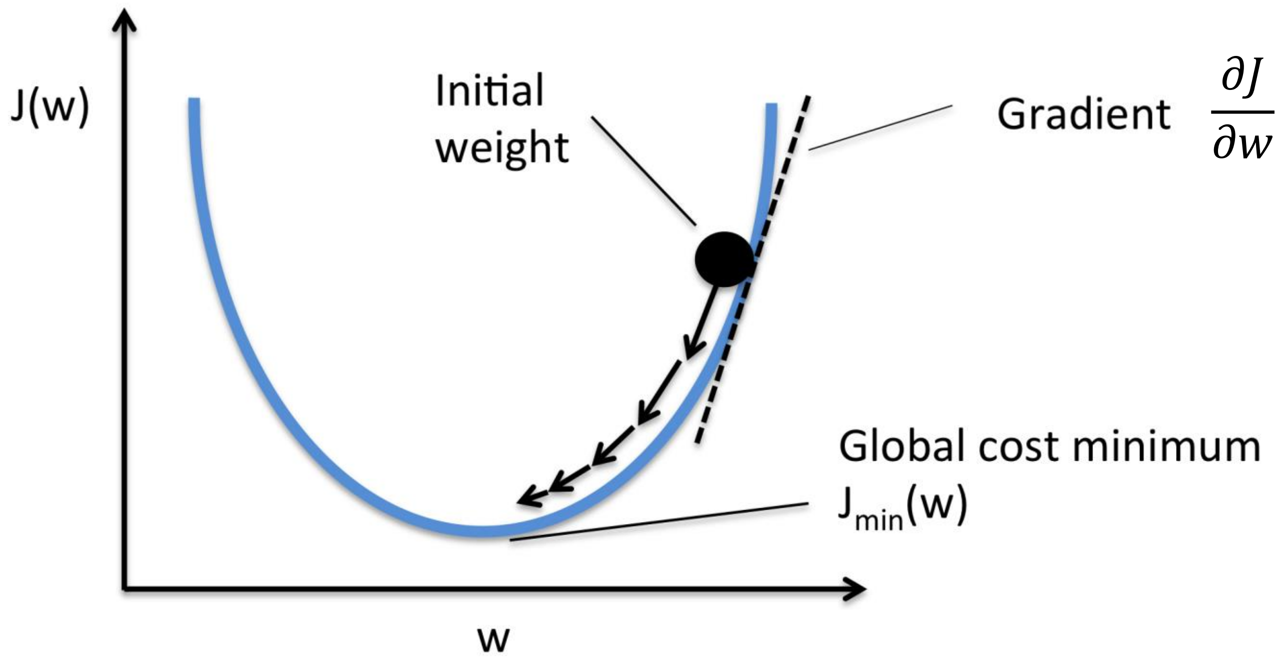
How Neural Networks learn?

- As mentioned, neural networks' final performance is 100% dependent on its weights
- learning = finding the best weights to use
- Neural networks are trained iteratively (i.e. one small step at a time)
  - Read a small batch of input data (can be as little as one but usually 32 – 256)
  - Make a prediction (i.e. inference)
  - Measure the error (by comparing the prediction with the correct answer)
  - Make small changes to the weights **accordingly (i.e. using an optimization method)**



Idealistic Stochastic Gradient Descent

- Most modern neural networks optimize its weight via stochastic gradient descent (or some variants of the SGD)
- Basic Idea:

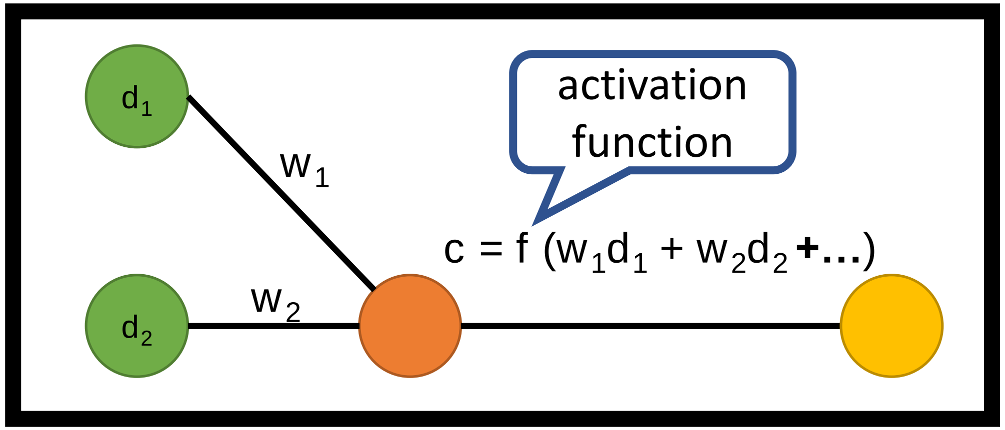


Weight Update Rule:

$$\boxed{w = w - \alpha \times \frac{\partial J}{\partial w}}$$

Neural Network Deep Dive

Choosing Activation Functions



Weight Update Rule:

$$w_1 = w_1 - \alpha \times \frac{\partial J}{\partial w_1}$$

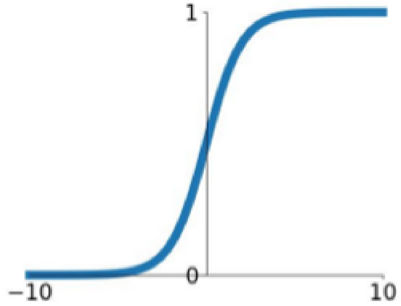
$$z = w_1 d_1 + w_2 d_2 + \ \ldots$$

$$c = f(z)$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial c} \times \frac{\partial c}{\partial z} \times \frac{\partial z}{\partial w_1}$$

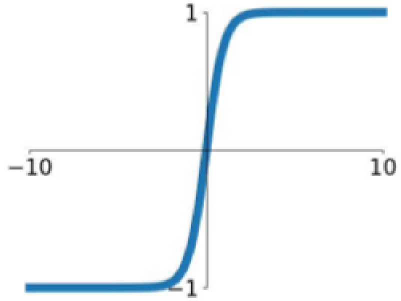
Sigmoid

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



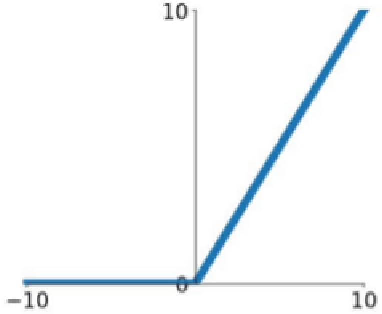
tanh

$$\tanh(x)$$



ReLU

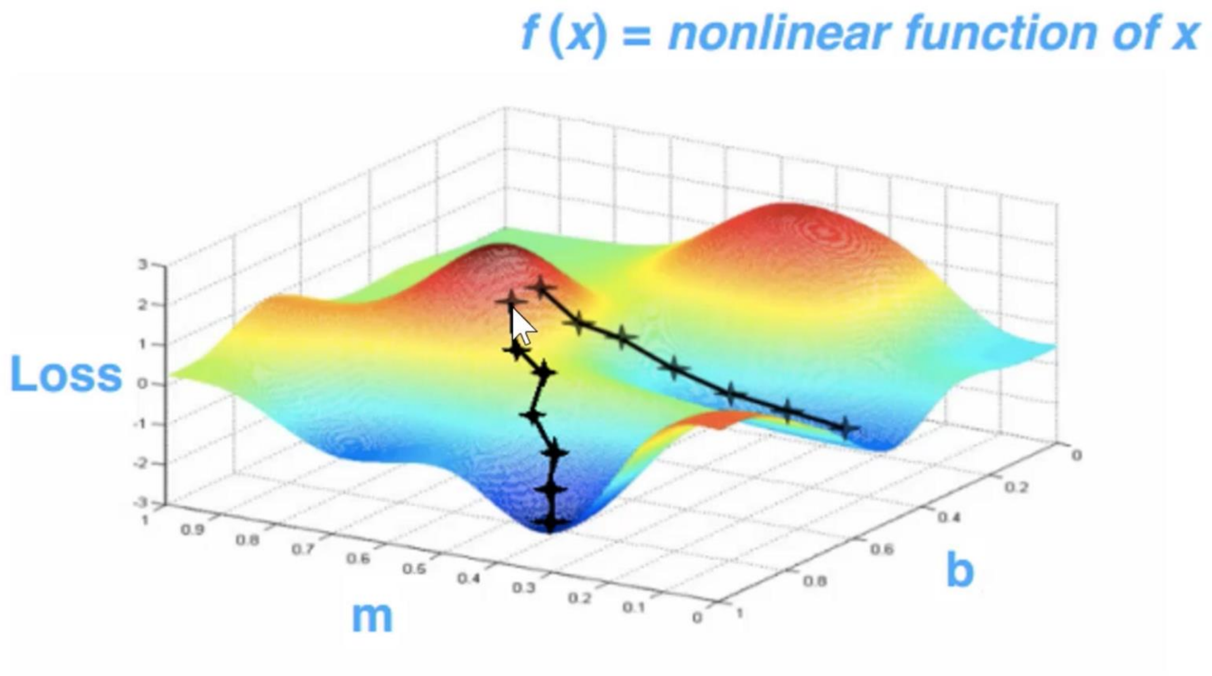
$$\max(0, x)$$



Neural Network Deep Dive

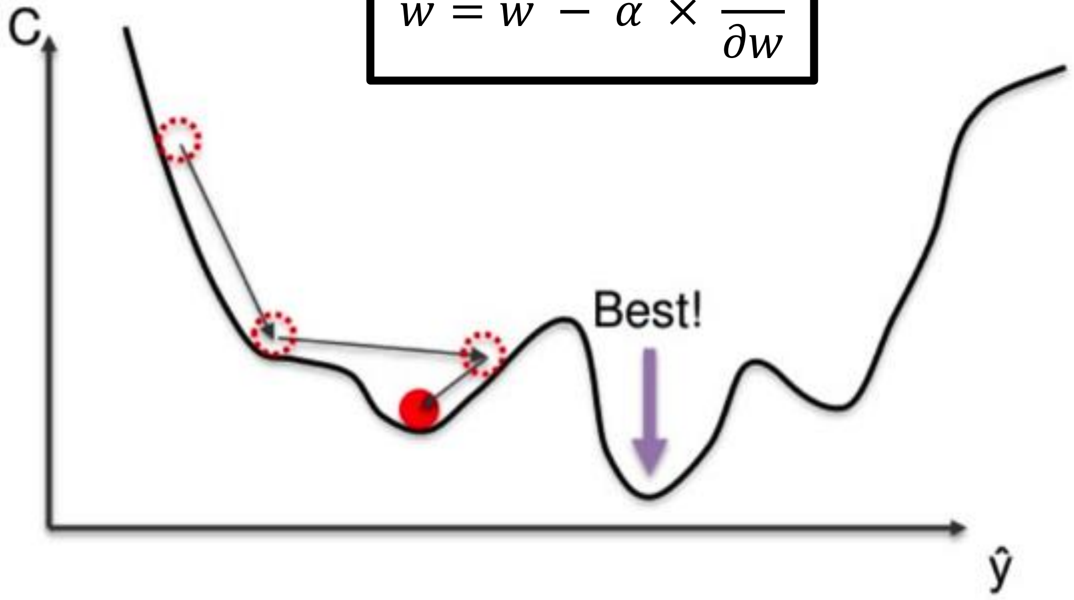
Outside the Fairy Tale

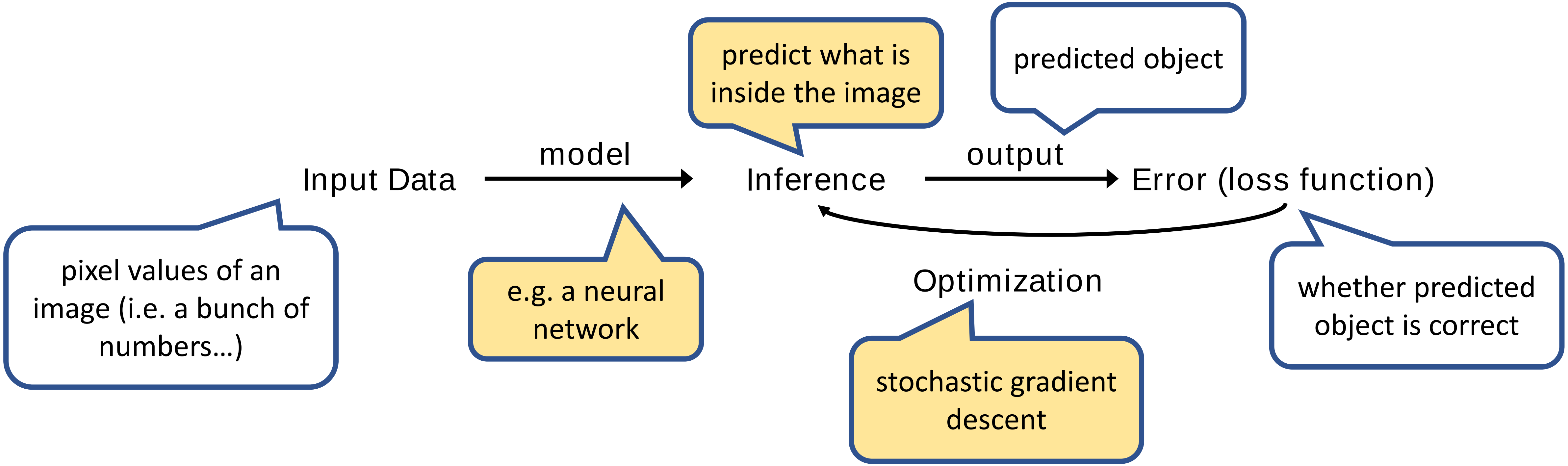
- In the real world, the loss function J is not necessarily concave with respect to the weights
- Hence, neural networks using SGD are trained to find local minima (i.e. a good approximation, not necessarily the best solution) to problems



Weight Update Rule:

$$\boxed{w = w - \alpha \times \frac{\partial J}{\partial w}}$$




predict what is inside the image
predicted object
model
output
Input Data
Inference
Error (loss function)
pixel values of an image (i.e. a bunch of numbers…)
e.g. a neural network
Optimization
whether predicted object is correct
stochastic gradient descent

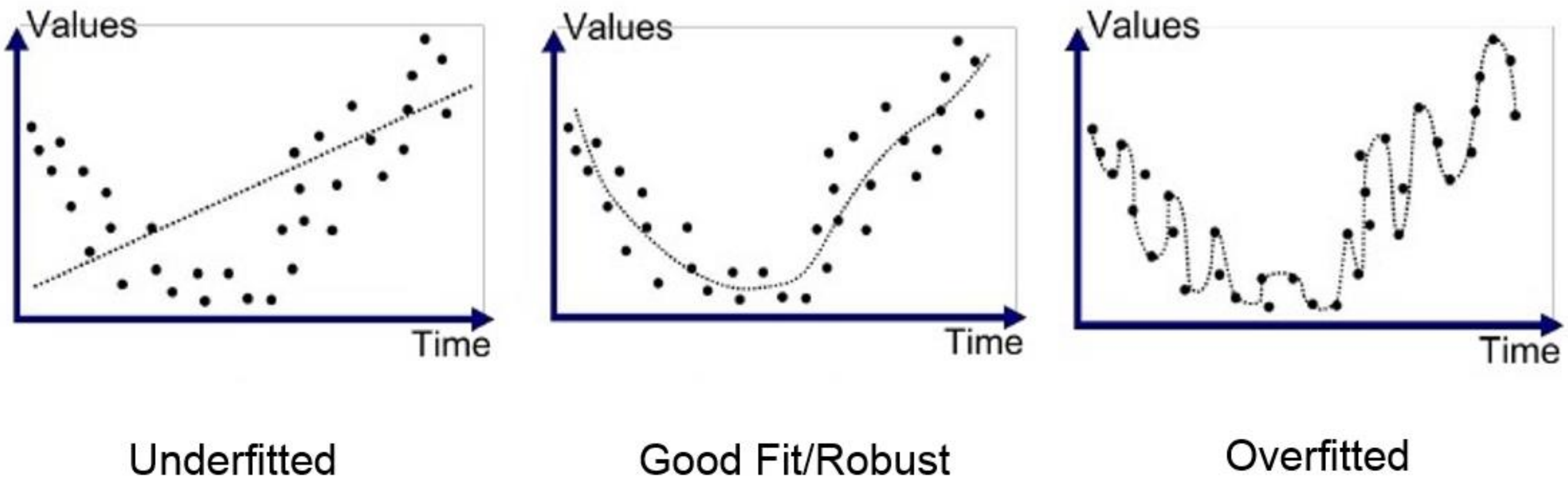
Neural Network Deep Dive

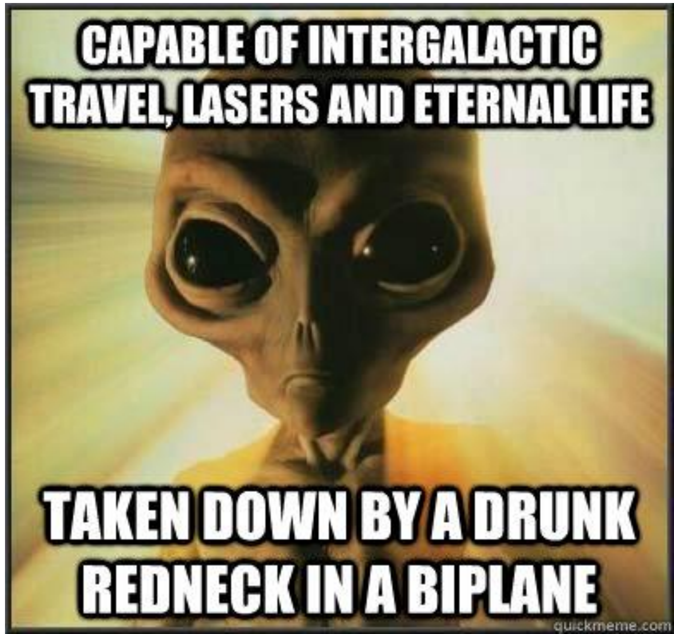
Simple demo

Neural Network Deep Dive

Capacity ≠ Performance

- When choosing the loss function, we cannot use the error as the only measure of performance, or else this will result in overfitting (i.e. not generalizing well)
- There are several common strategies against overfitting
    - Norm penalties
    - Data augmentation
    - Early Stoppage





Neural Network Deep Dive

Other Challenges

- How to initialize weights
- Prevent vanishing / exploding gradients
- Optimize training time
- Missing / skewed data
- Non-numerical data / time series data
- Data privacy