Reverse Engineering of AI Models

28 Sept. 2020

Mr. Jingli SHI

PhD at Auckland University of Technology

- Speaker: Mr. Jingli SHI
  - *PhD @ Auckland University of Technology, New Zealand*
  - Research: *natural language processing*

- Session 1 – NLP Models
  - Time: **15:00 – 16:00**
  - Course Aims: Understand low-level theory of AI model using XOR use case.

- Break
  - Time: **16:00 – 16:05**

- Session 2 – Demo
  - Time: **16:05 – 16:35**
  - Course Aims: Learn to implement AI model.

Outline

- Background
- AI Model Training Routine (XOR use case)
- Classic AI Models

Outline

- Background
  - AI Milestones
  - Who is Smarter?
  - Course goal
  - AI vs ML vs DL vs NLP
- AI Model Training Routine (XOR use case)
- Classic AI Models

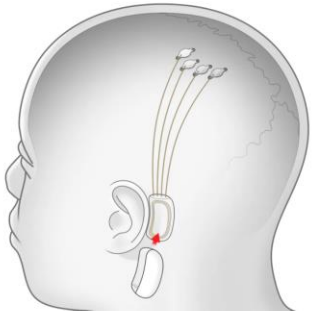
AI Advance Milestones

2016

2018

2017

2020



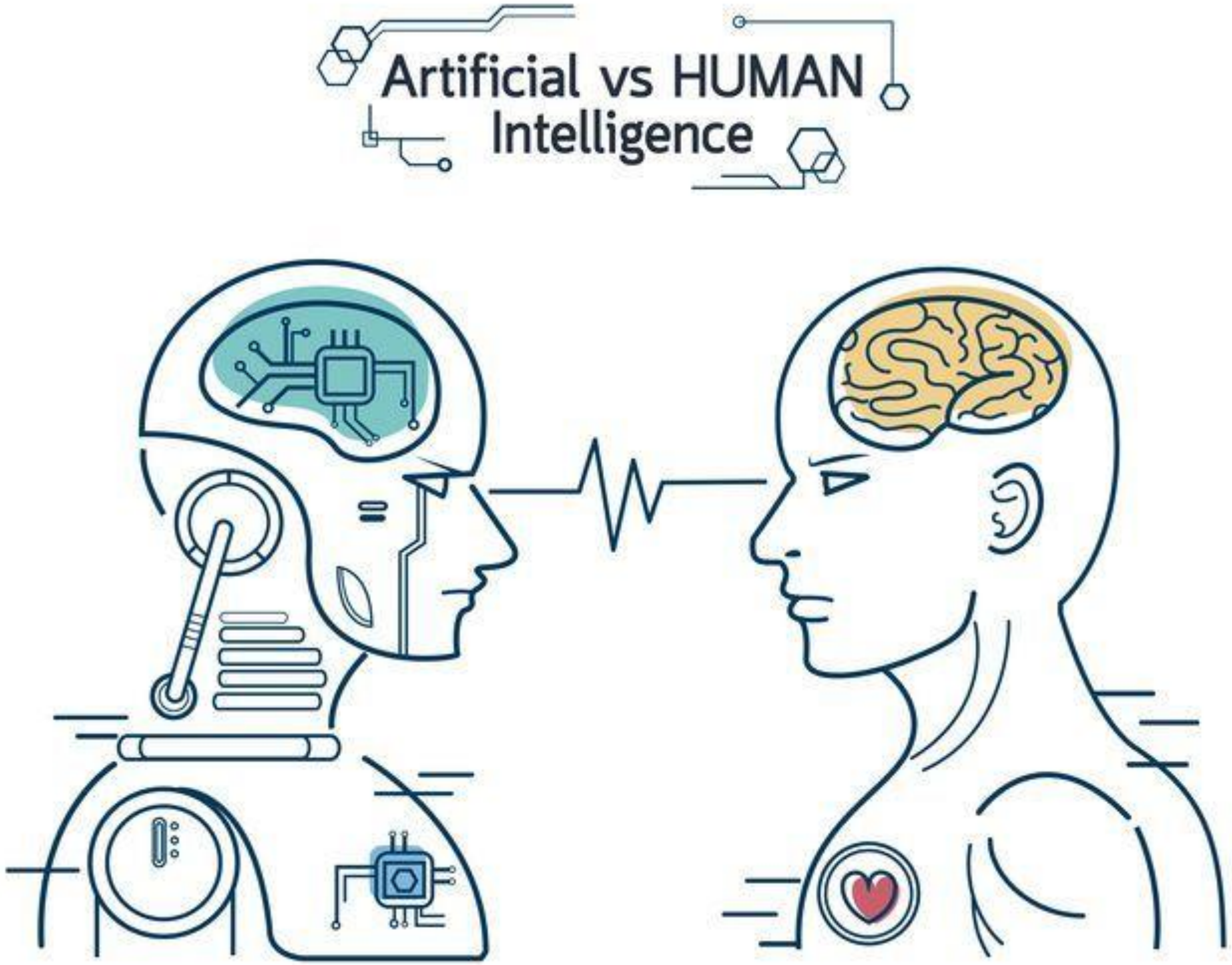
Neuralink by Elon Musk

Who is Smatter?

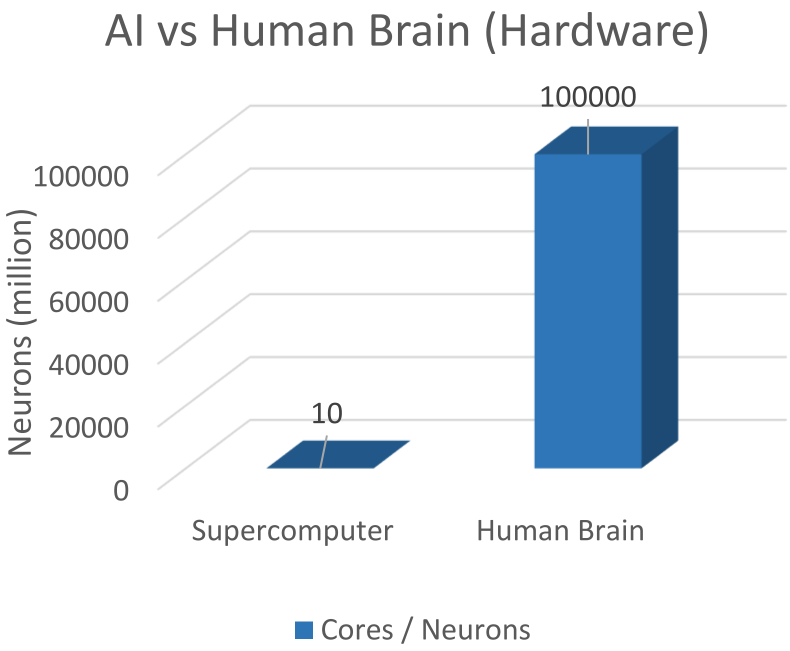
Estimation:

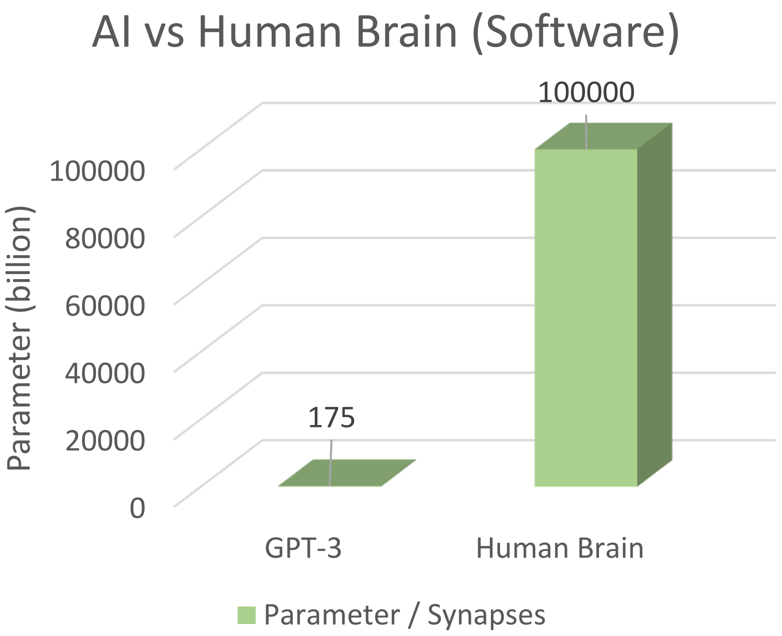
Robot surpass the capability of human brains around 2040.

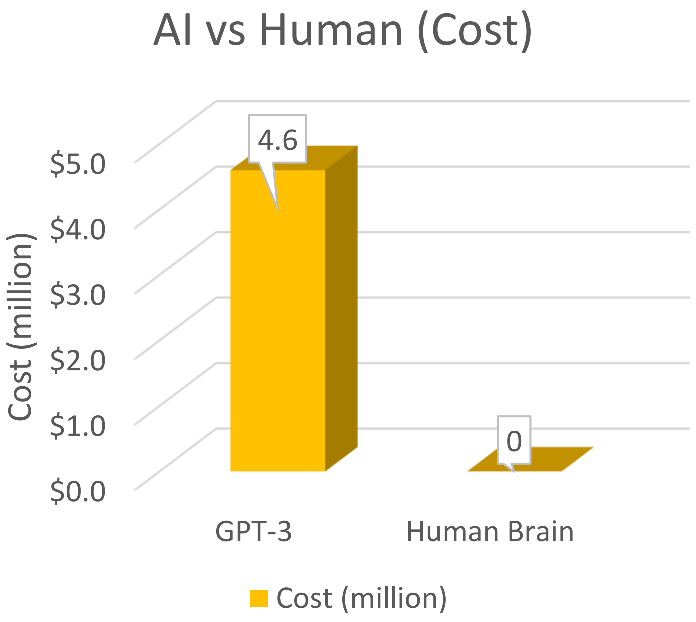
How about now?



AI vs Human

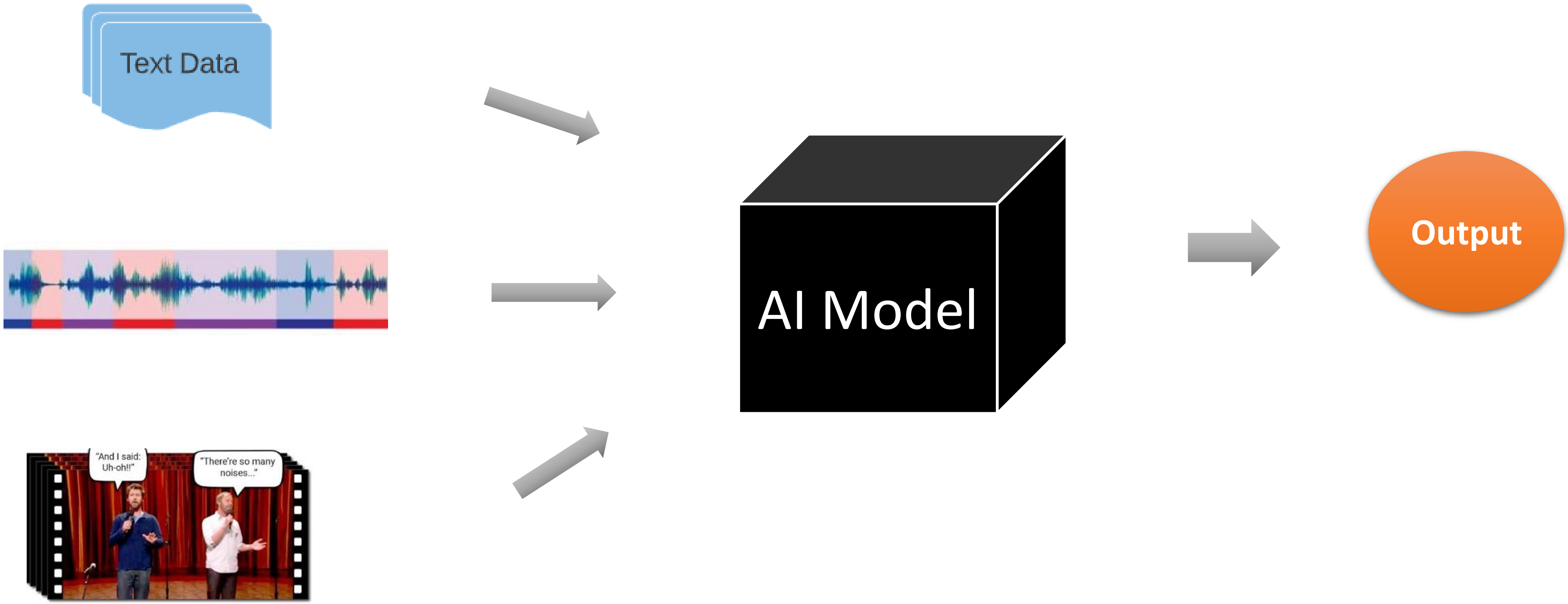




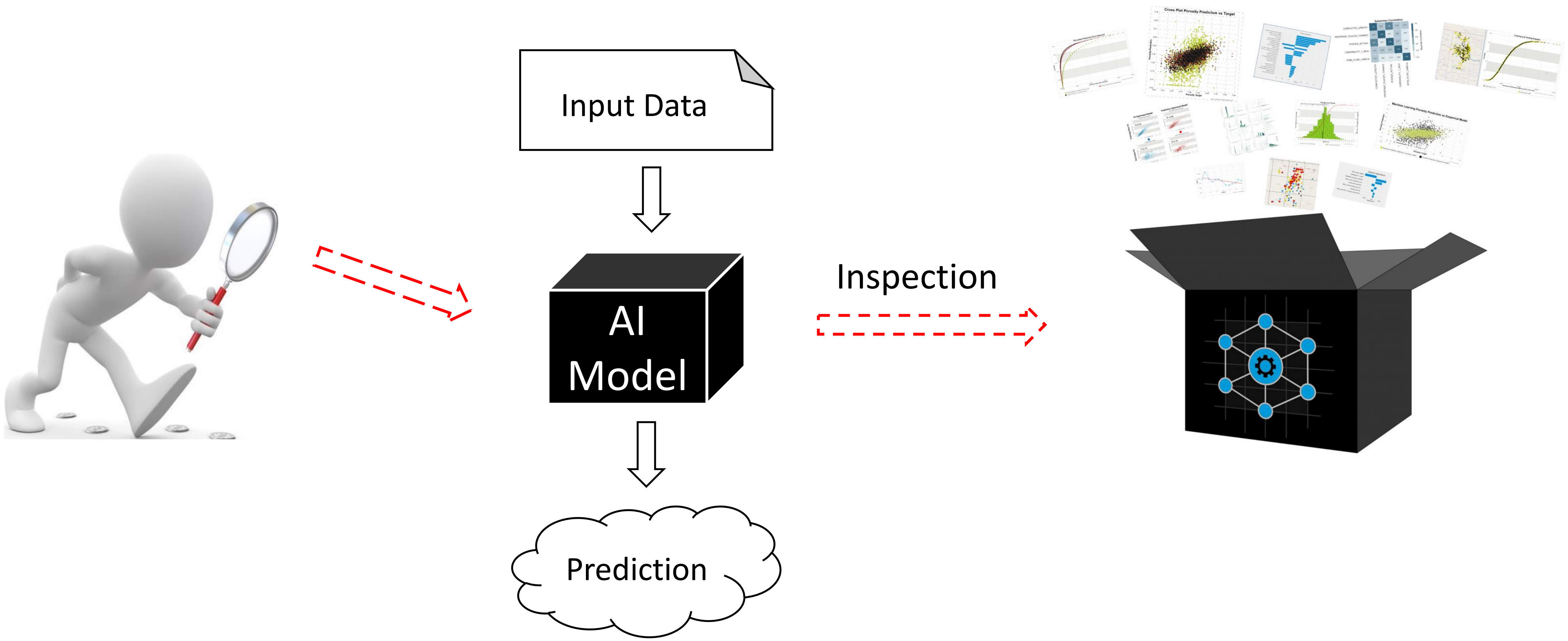


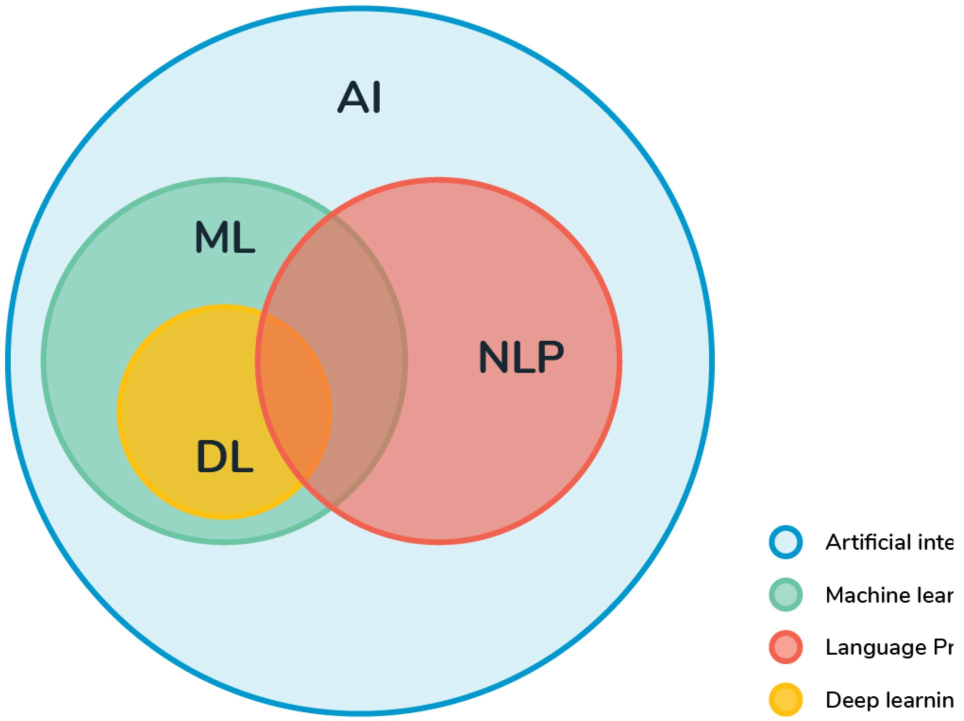
Source: https://brandminds.live/ai-versus-the-human-brain/#:~:text=%E2%80%9CArtificial%20intelligence%20(or%20AI),power%20of%20the%20human%20brain.&text=By%20comparisons%2C%20human%20brains%20can,to%20a%20single%20human%20brain.
https://www.youtube.com/watch?v=kpiY_LemaTc

AI (Blackbox)


Text Data
"And I said: Uh-oh!!"
"There're so many noises..."
AI Model
Output

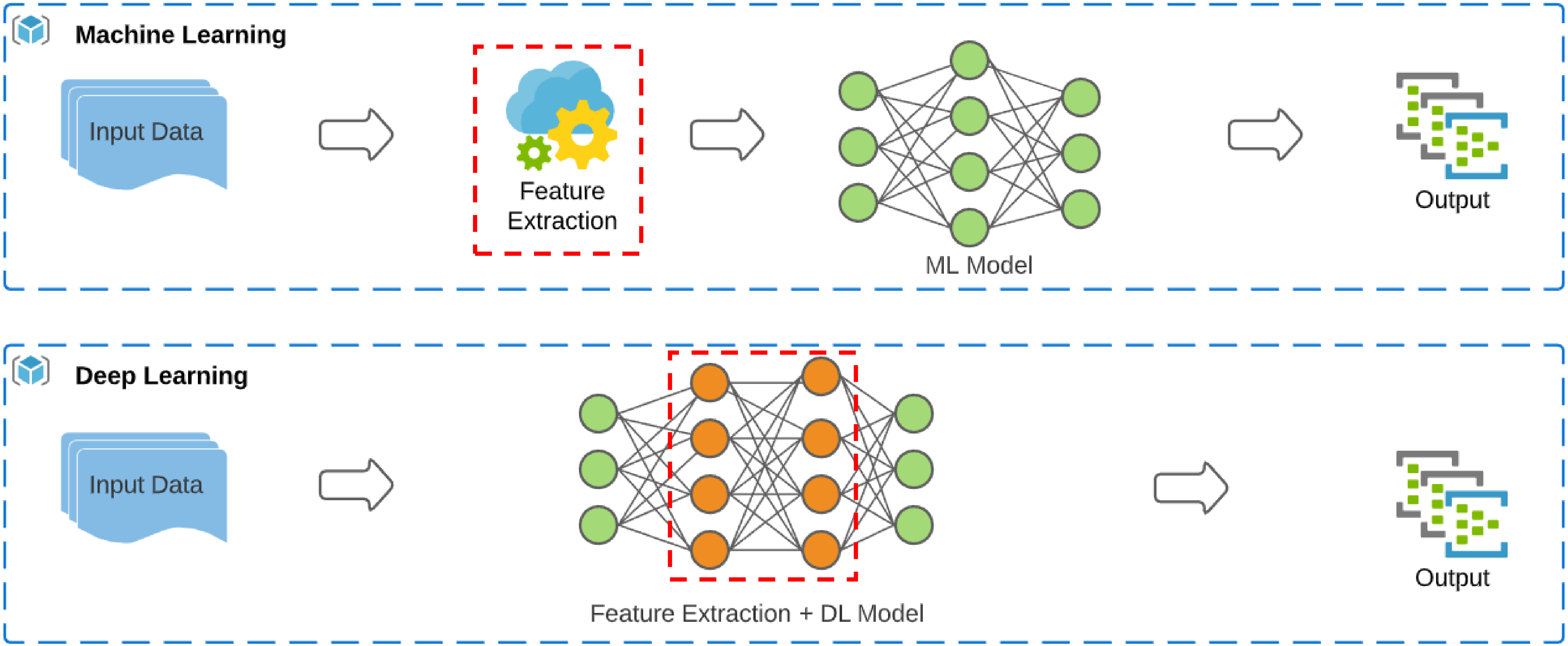
Blackbox Inspection





Background (AI vs ML vs DL vs NLP)

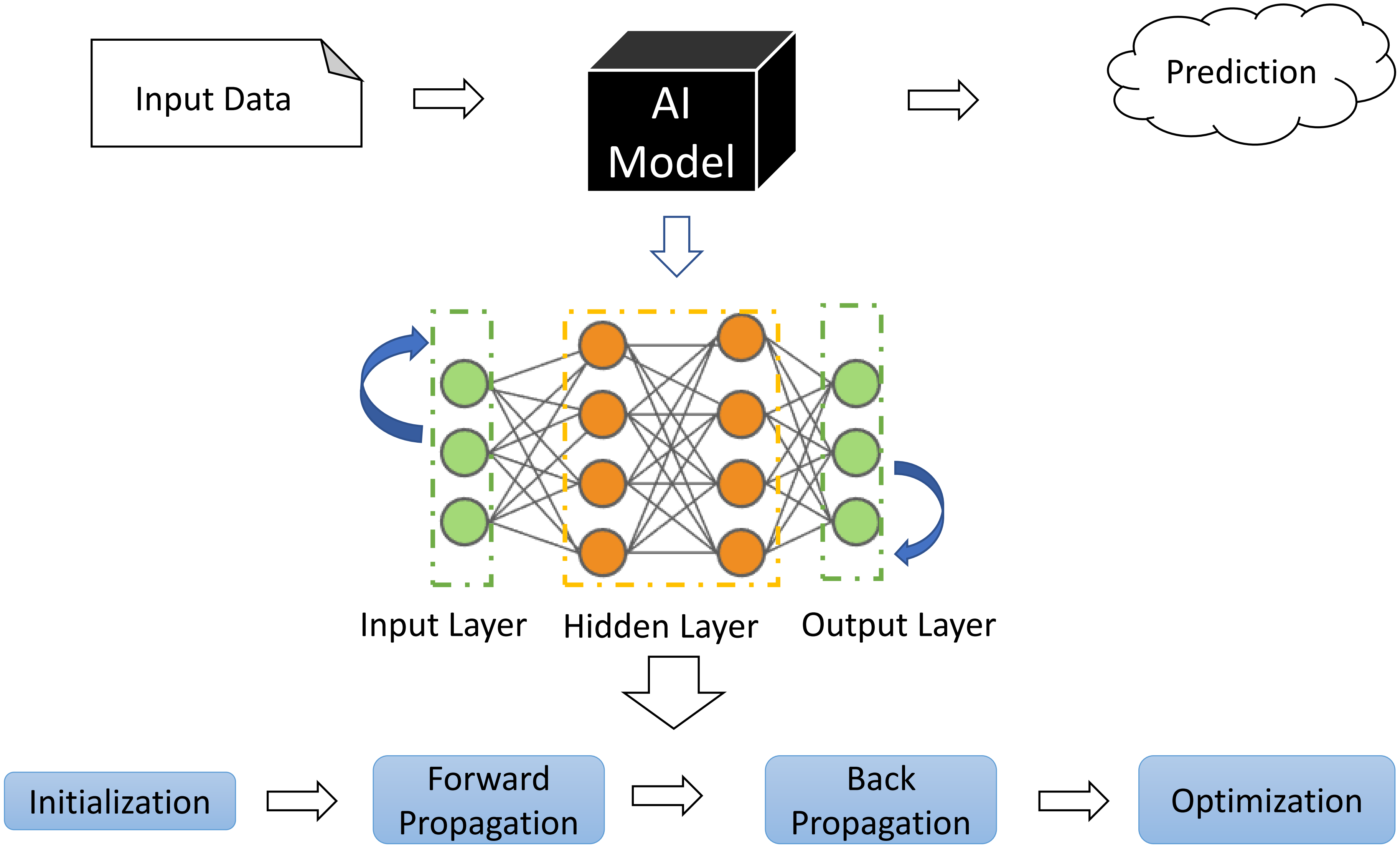
Machine Learning vs Deep Learning



Outline

- Background
- AI Model Training Routine (XOR use case)
  - Initialization
  - Forward Propagation
  - Backward Propagation
  - Optimization
- Classic AI Models

Explore Blackbox



AI Model Training Routine

For an AI model, the typical training routine is performing the following 4 steps **iteratively**.

Initialization

1. Initialize or update weights vector

Forward Propagation

2a. Multiply the weights vector with the inputs, sum the products.

2b. Put the sum through the activation function, e.g. sigmoid, tanh, ReLU, etc.

AI Model Training Routine

Back Propagation

3a. Compute the errors, i.e. difference between expected output and predictions

3b. Multiply the error with the derivatives to get the delta

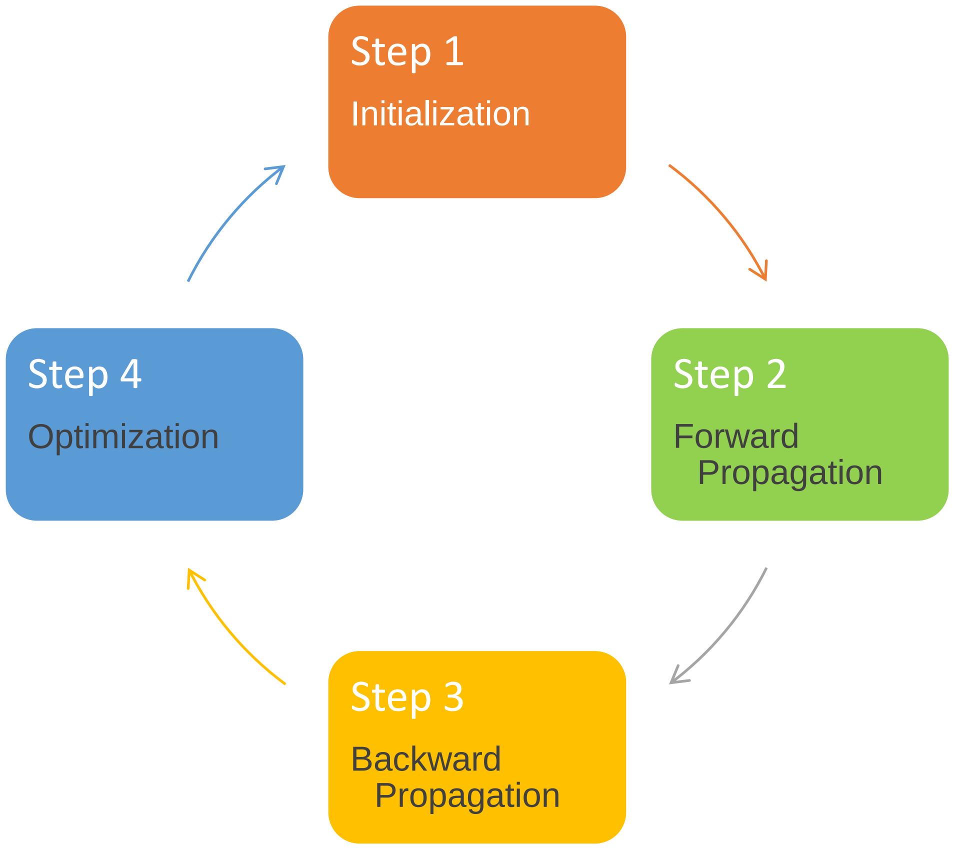
3c. Multiply the delta vector with the inputs, sum the product

Optimizer takes a step

4. Multiply the learning rate with the output of step 3c

Repeat 1-4 until *desired*

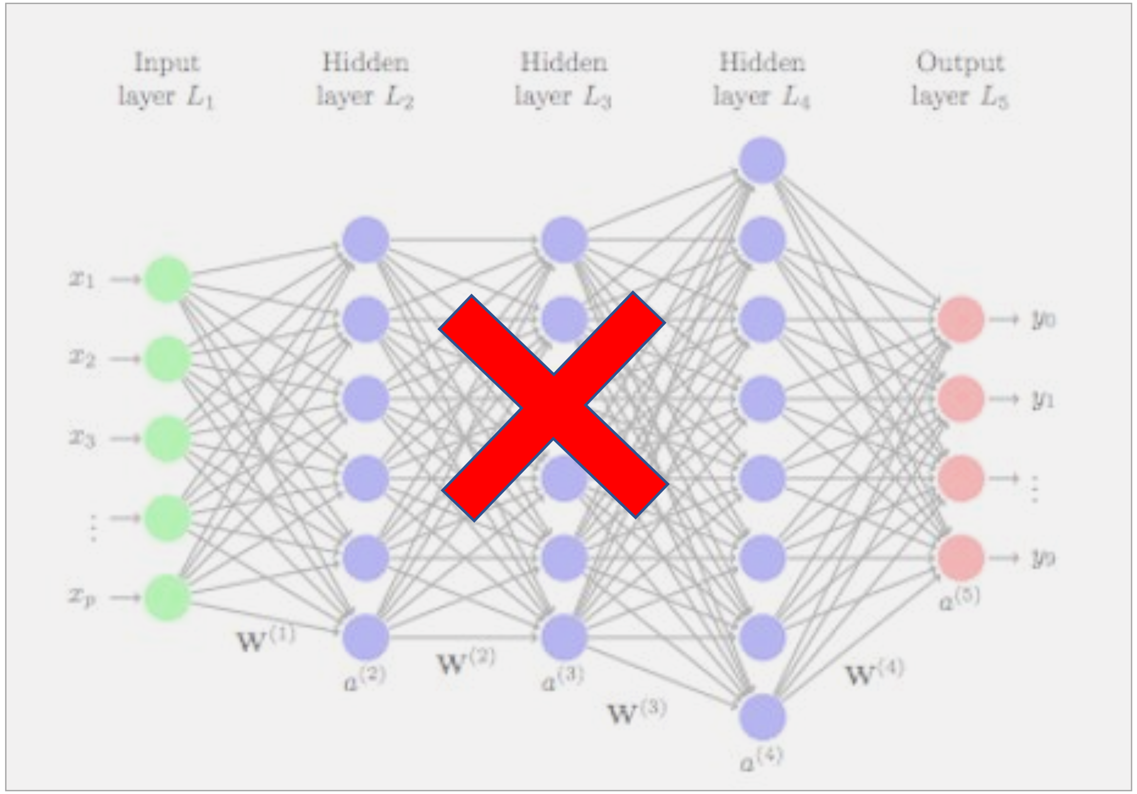
AI Model Training Routine

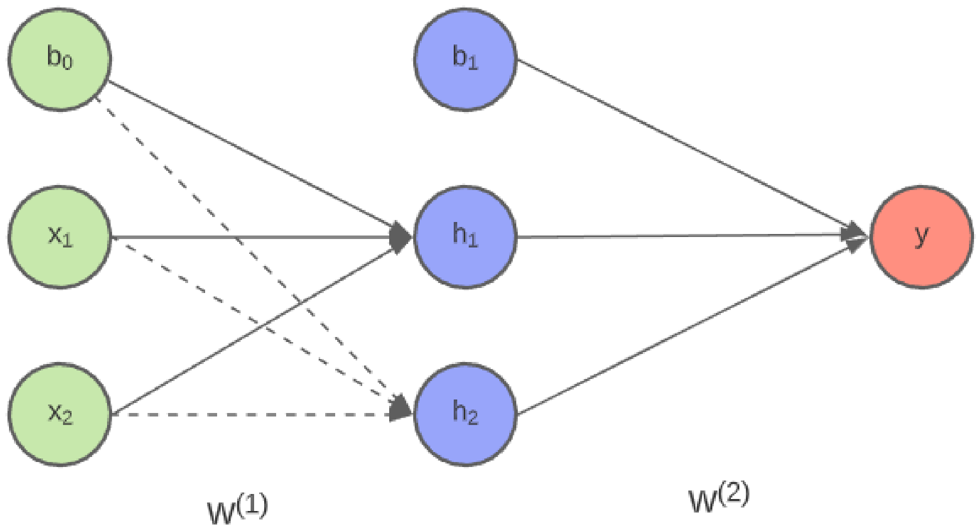


Stop criteria:

1. Loop end
2. No accuracy improvement

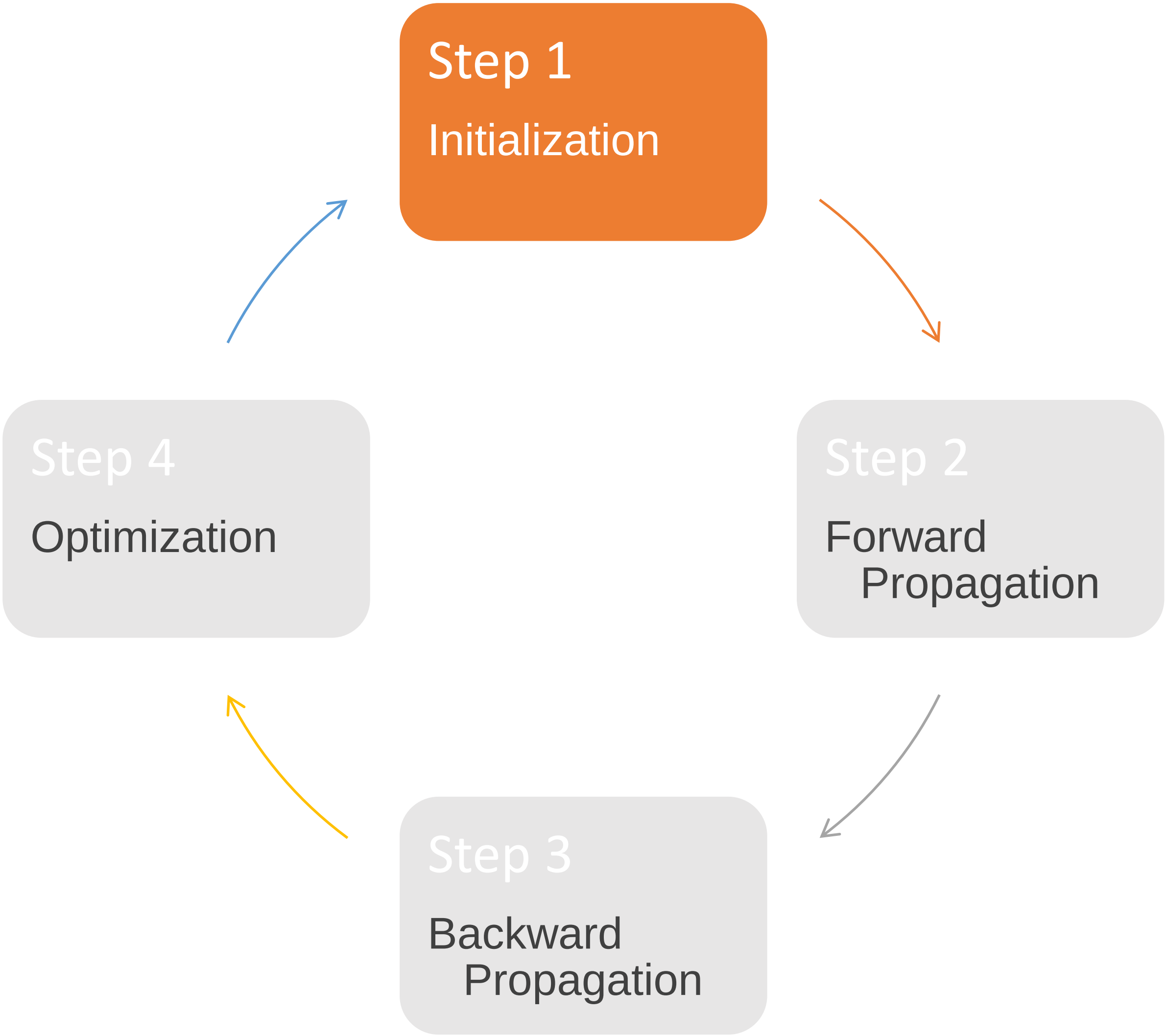
Simple Model Inspection





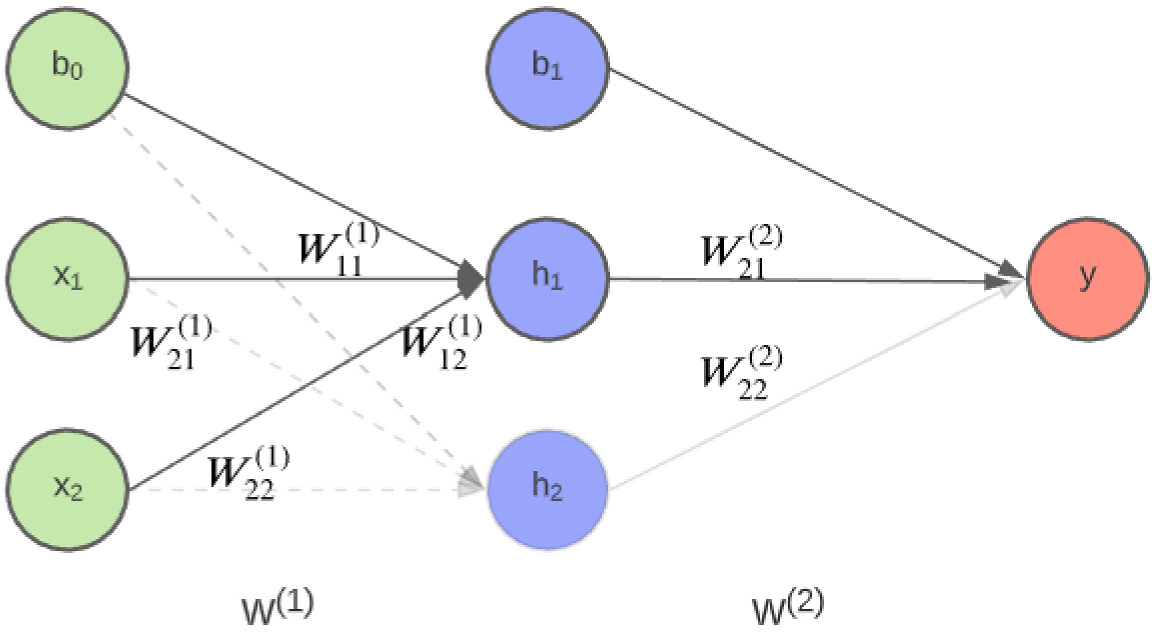
Source: https://uc-r.github.io/2018/04/09/feedforward-deep-models/

Initialization



XOR - Initialization

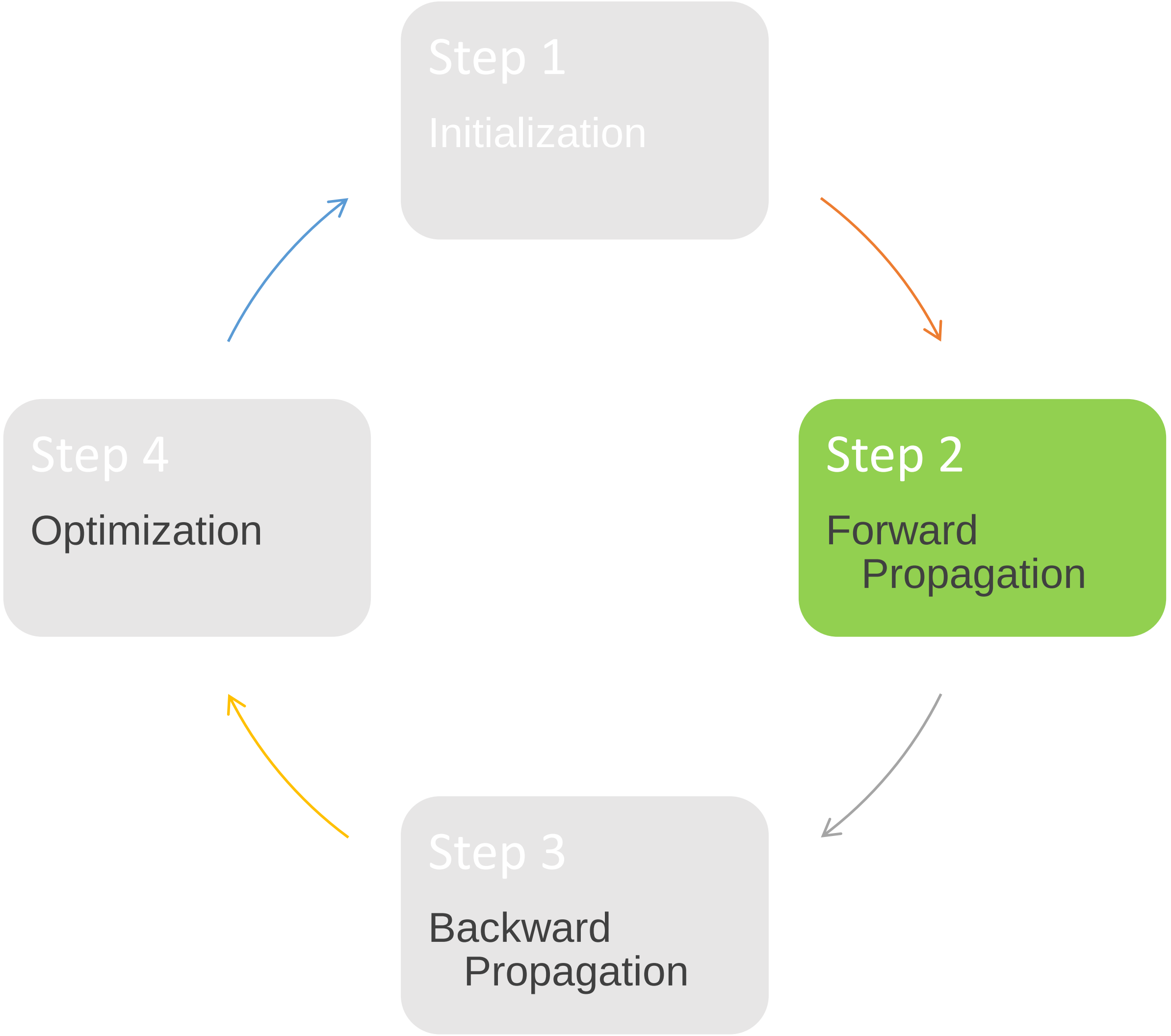
x1	x2	x1 XOR x2
0	0	0
0	1	1
1	0	1
1	1	0



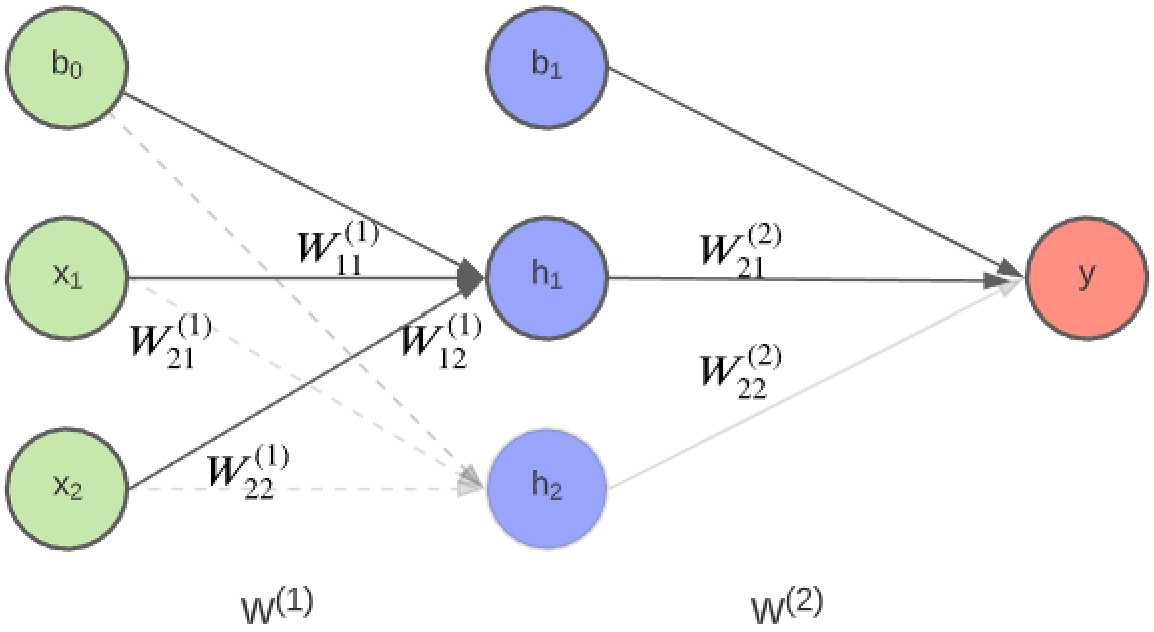
x1	x2	b0	W1_11	W1_12	W1_21	W1_22	b1	W2_21	W2_22
0	1	1	-1	1	1	-1	-1	1	1

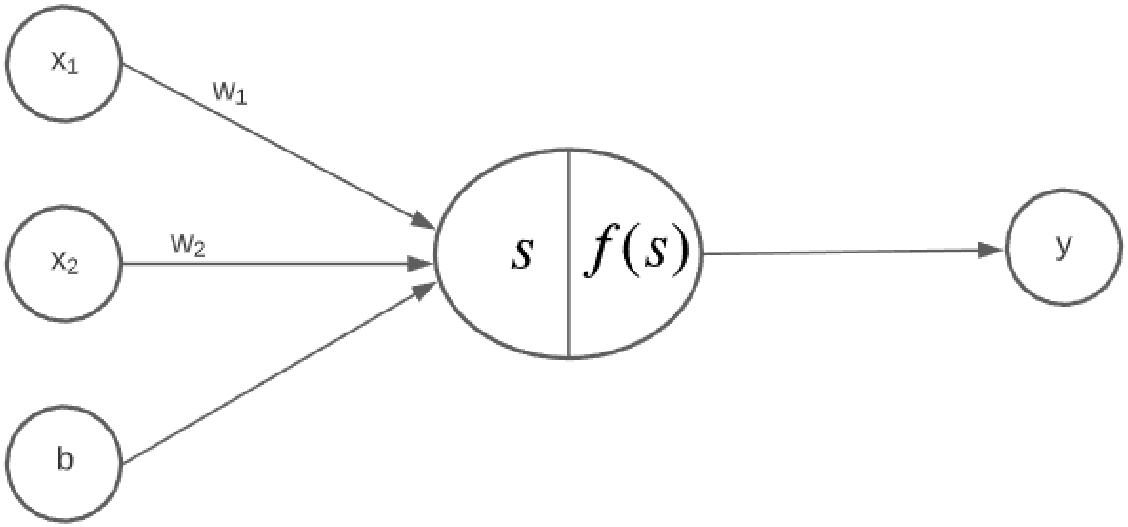
⋮

Forward Propagation



XOR - Forward Propagation





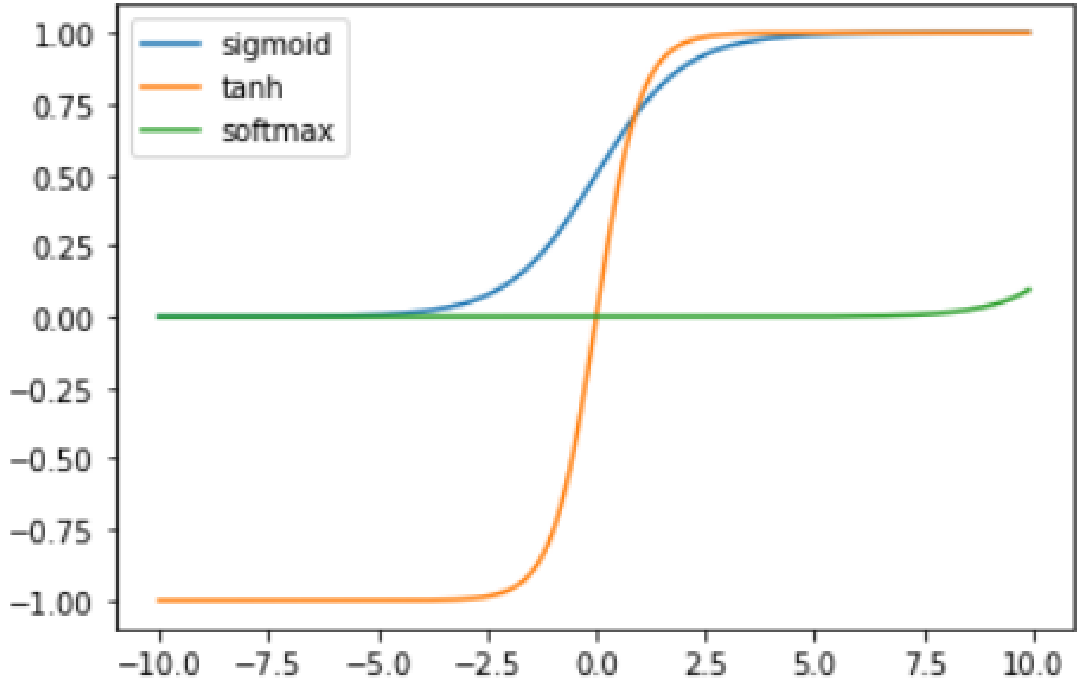
Preactivation Function

$$s = \sum w_i * x_i + b$$

Activation Function

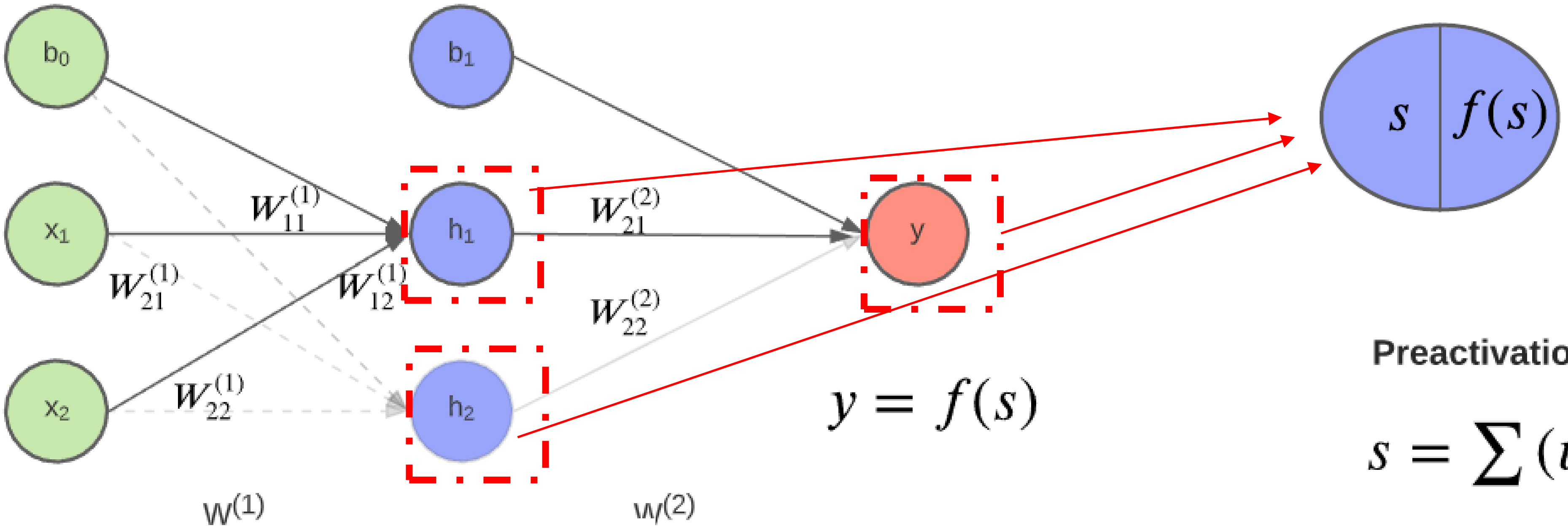
$$f(s) = \frac{1}{1+e^{-s}}$$

XOR - Activation Function



Activation function of a node defines the output of that node given an input or set of inputs. They help in keeping the value of the output from the neuron restricted to a certain limit as per our requirement.

XOR - Neuron Calculation



$$h_1 = f(s)$$

$$h_2 = f(s)$$

$$y = f(s)$$

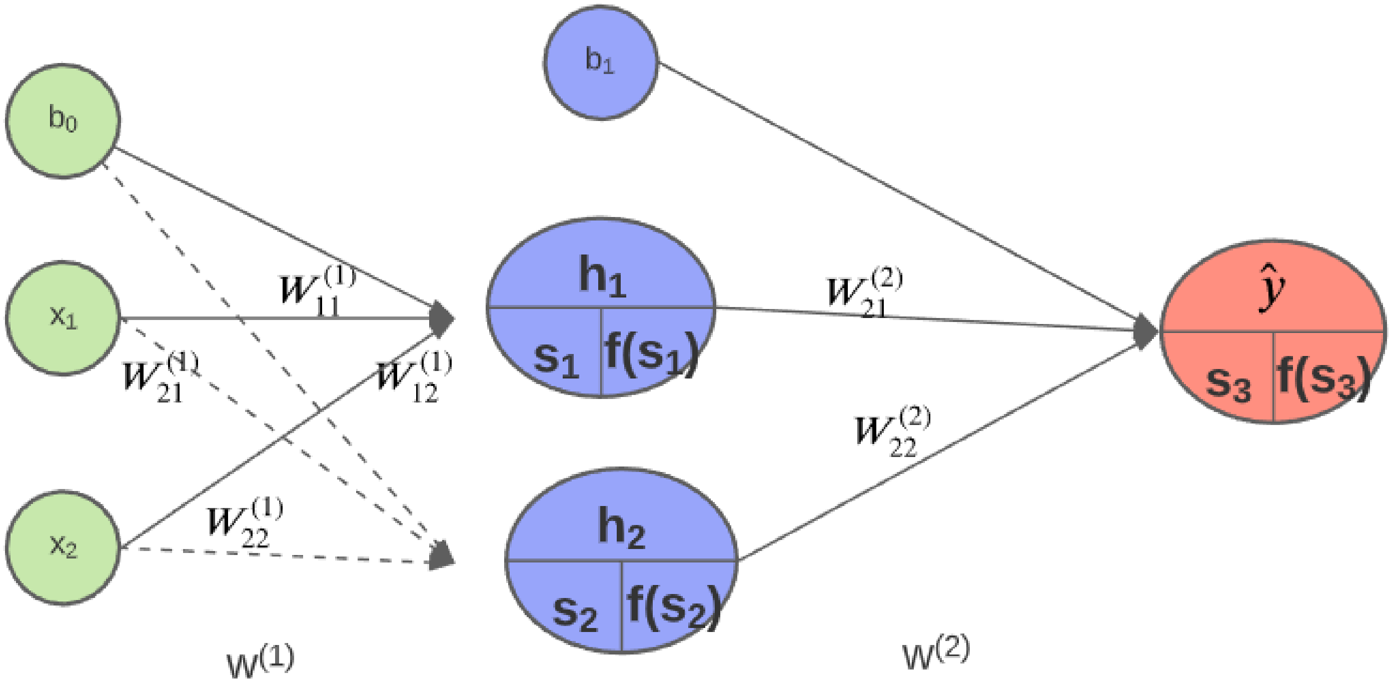
Preactivation Function

$$s = \sum (w_i * x_i) + b$$

Activation Function

$$f(s) = \frac{1}{1+e^{-s}}$$

XOR - Output Calculation



$$s_1 = \sum w_{1i}^{(1)} * x_i + b_0$$

$$h_1 = f(s_1) = sigmoid(s_1) = \frac{1}{1+e^{-s_1}}$$

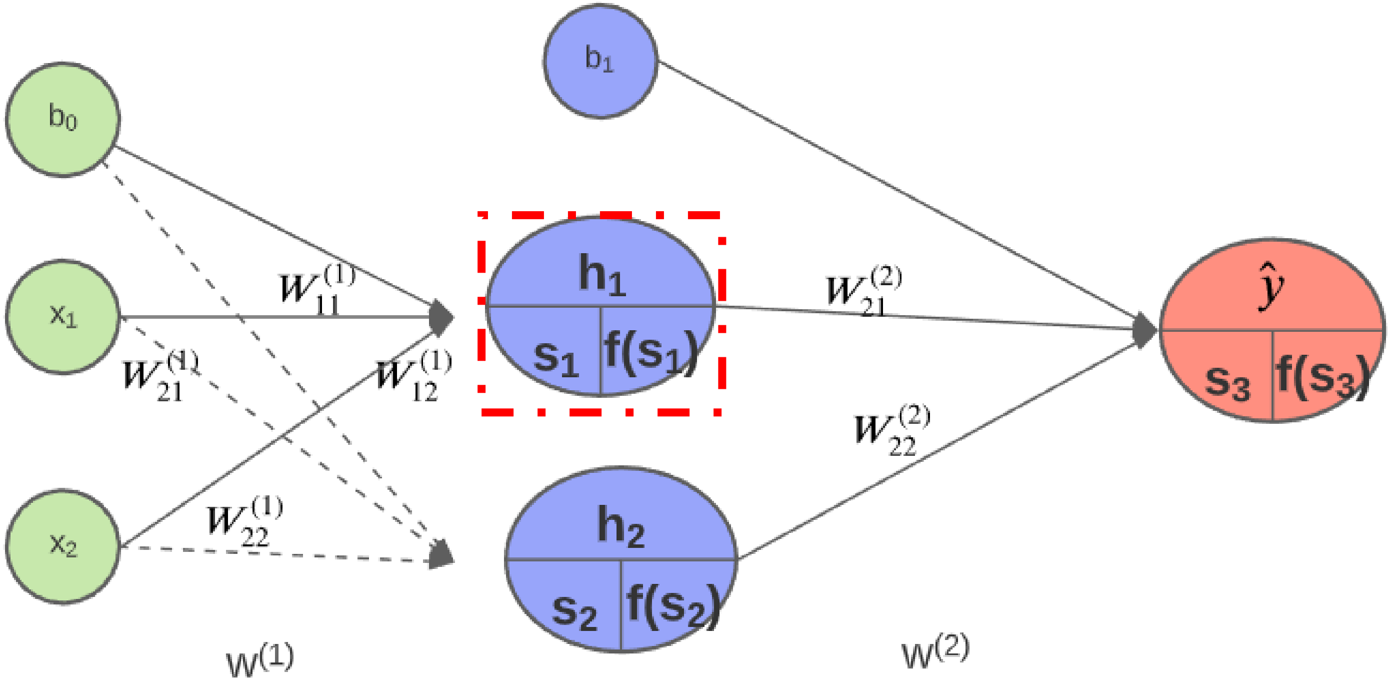
$$s_2 = \sum w_{2i}^{(1)} * x_i + b_0$$

$$h_2 = f(s_2) = sigmoid(s_2) = \frac{1}{1+e^{-s_2}}$$

$$S_3 = \sum w_{2i}^{2} * h_i + b_1$$

$$\hat{y} = f(s_3) = sigmoid(s_3) = \frac{1}{1+e^{-s_3}}$$

XOR – Forward Propagation



b0	W1_11	W1_12
1	-1	1

$$s_1 = \sum w_{1i}^{(1)} * x_i + b_0$$

$$s_1 = w_{11}^{(1)} * x_1 + w_{12}^{(1)} * x_2 + b_0$$

$$s_1 = (-1) * 0 + 1 * 1 + 1$$

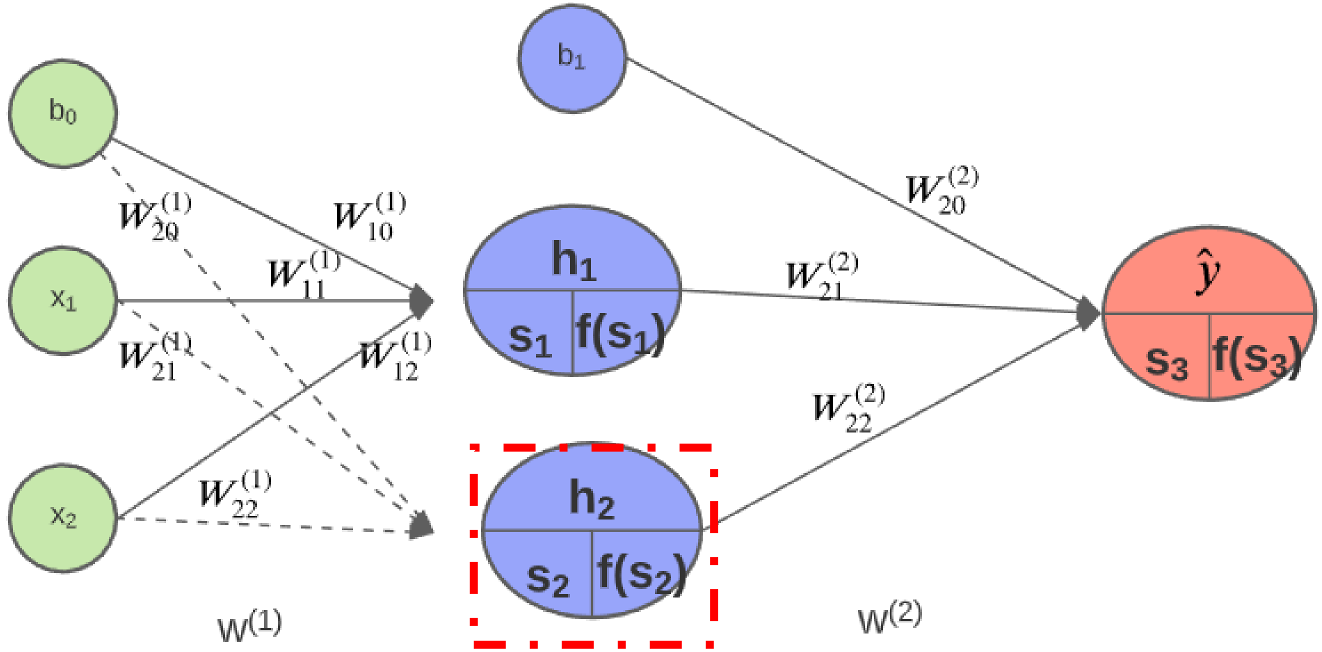
$$s_1 = 2$$

$$h_1 = f(s_1) = sigmoid(s_1) = \frac{1}{1+e^{-s_1}}$$

$$h_1 = 0.88$$

x_1	x_2	s_1	h1=f(s1)
0	1	2	0.88

XOR – Forward Propagation



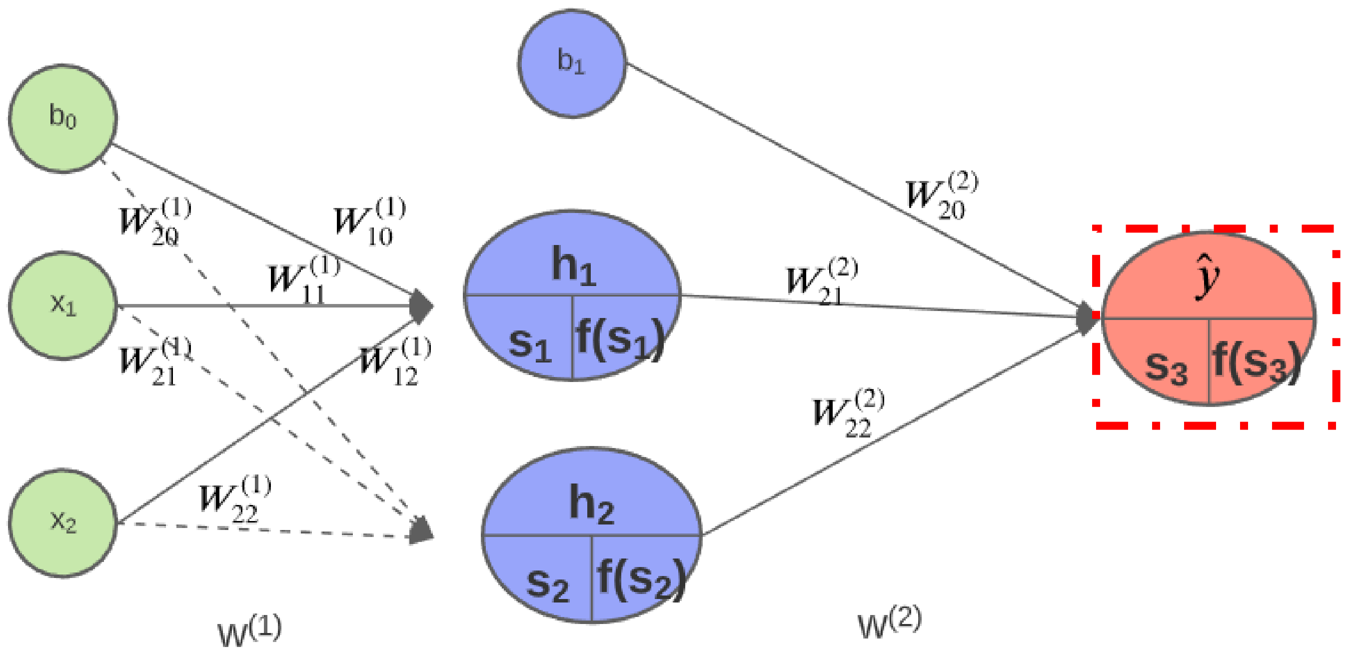
b0	W1_21	W1_22
1	1	-1

$$s_2 = w_{21}^{(2)} * x_1 + w_{22}^{(2)} * x_2 + b_0$$

$$h_2 = f(s_2) = sigmoid(s_2) = \frac{1}{1+e^{-s_2}}$$

x_1	x_2	s_1	h1=f(s1)	s2	h2=f(s2)
0	1	2	0.88	0	0.5

XOR – Forward Propagation



b1	W2_21	W2_22
-1	1	1

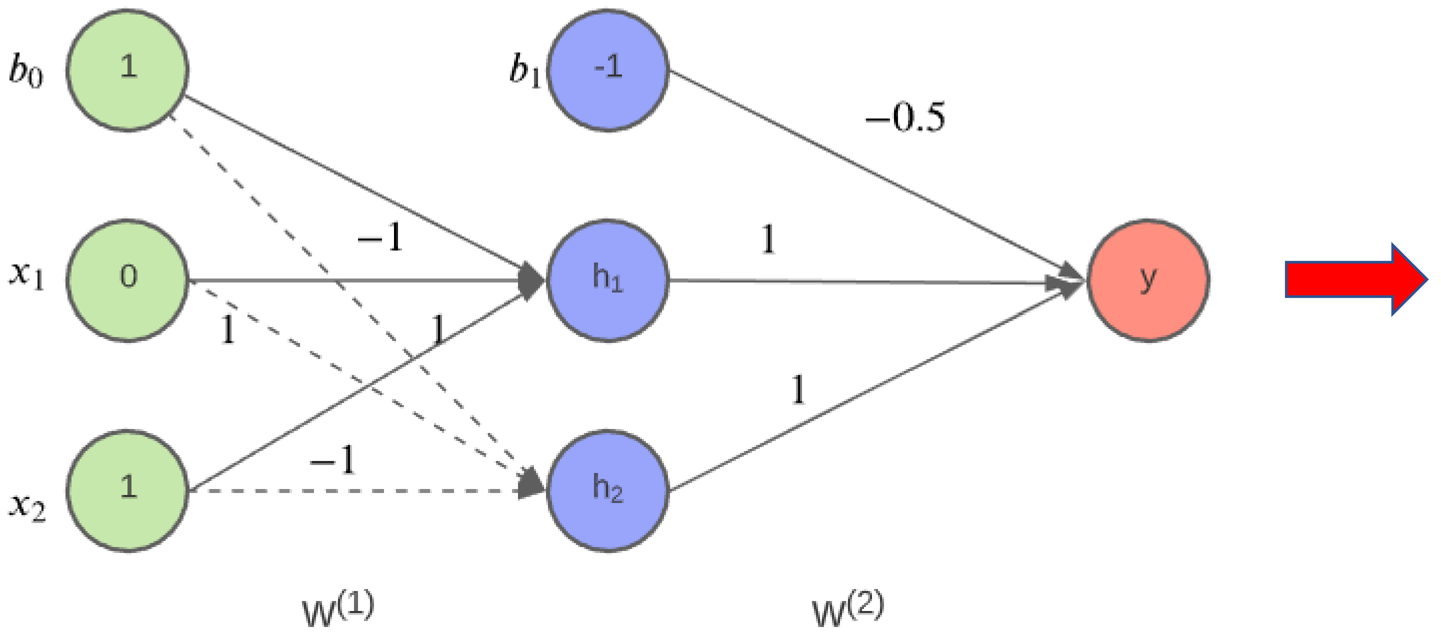
$$S_3 = \sum w_{2i}^{(2)} * h_i + b_1$$

$$s_3 = w_{21}^{(2)} * h_1 + w_{22}^{(2)} * h_2 + b_1$$

$$\hat{y} = f(s_3) = sigmoid(s_3) = \frac{1}{1+e^{-s_3}}$$

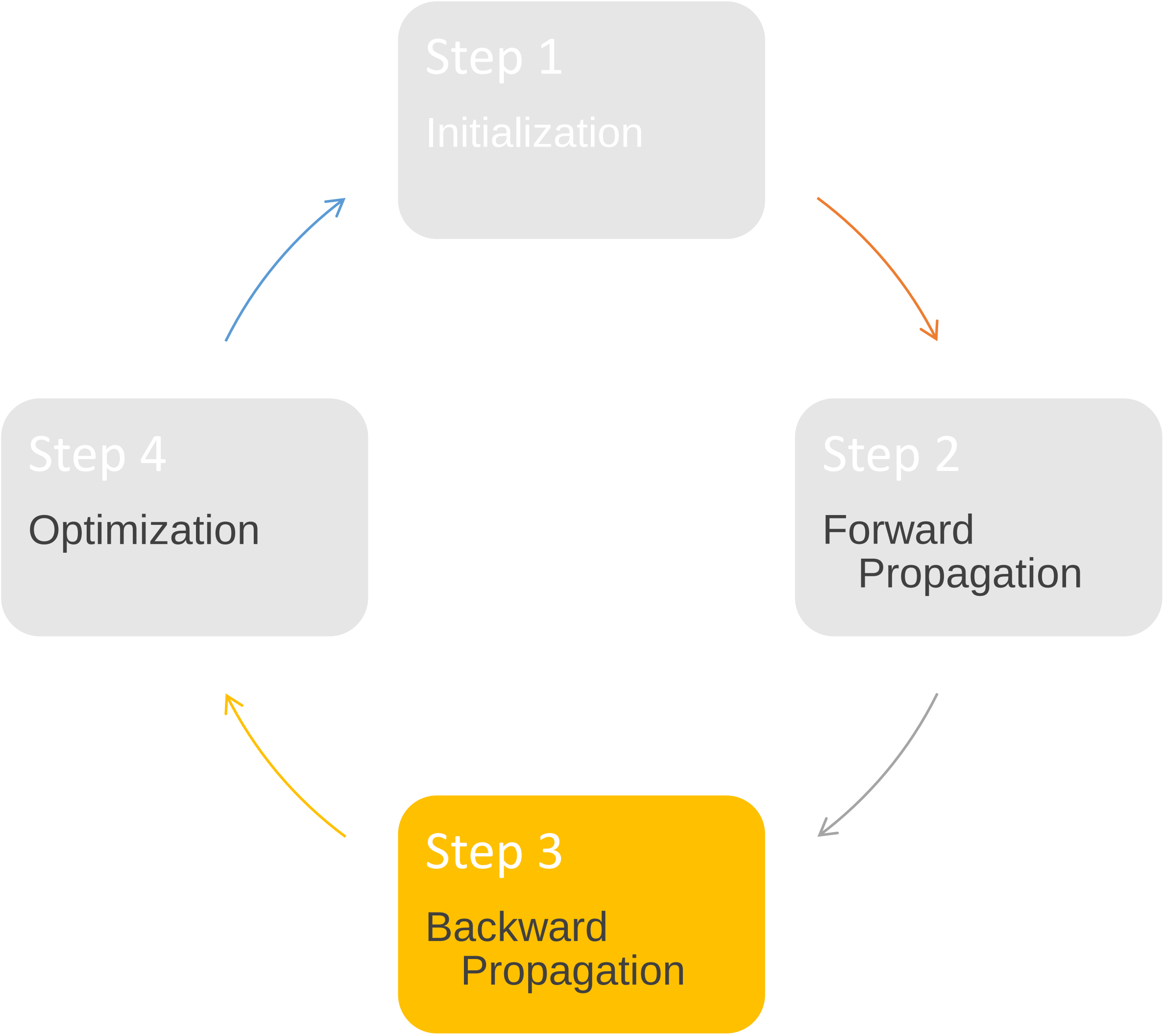
x_1	x_2	s_1	h1=f(s1)	s2	h2=f(s2)	s3	y^=f(s3)
0	1	2	0.88	0	0.5	0.38	0.59

XOR - Forward Propagation



x_1	x_2	y	y'
0	1	1	0.59

Backward Propagation

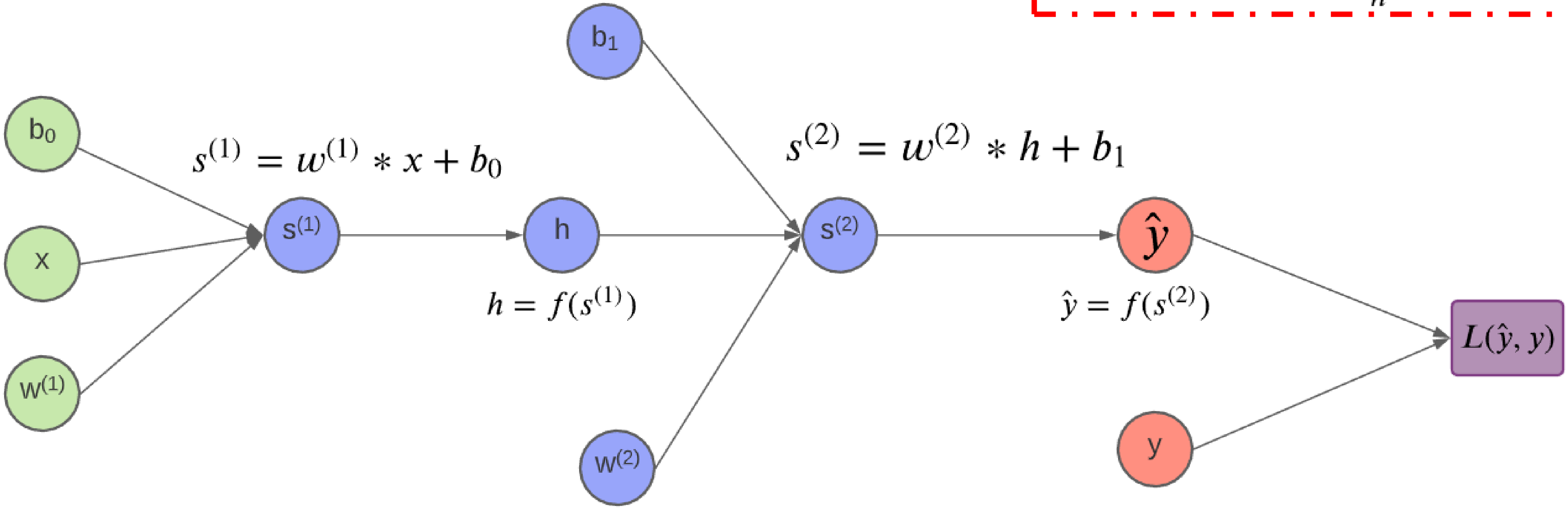


XOR – Backward Propagation

Cost Function

MAE: $L(\hat{y}, y) = \frac{1}{n}|\hat{y} - y|$

MSE: $L(\hat{y}, y) = \frac{1}{n}(\hat{y} - y)^2$



XOR – Loss/Cost Function

Target:
Minimum Difference

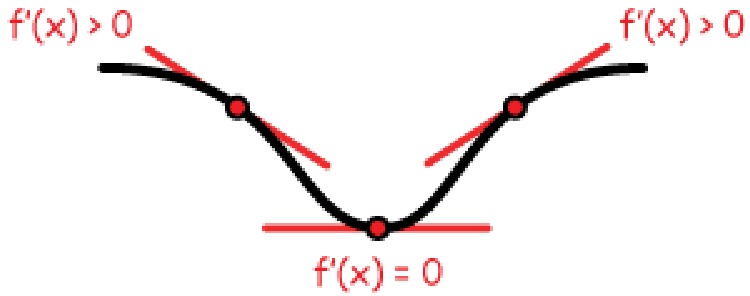
x_1	x_2	y	y’
0	1	1	0.59

$$L(\hat{y}, y) = (\hat{y} - y)^2 = F(s, h, w) = F(W)$$

Minimum

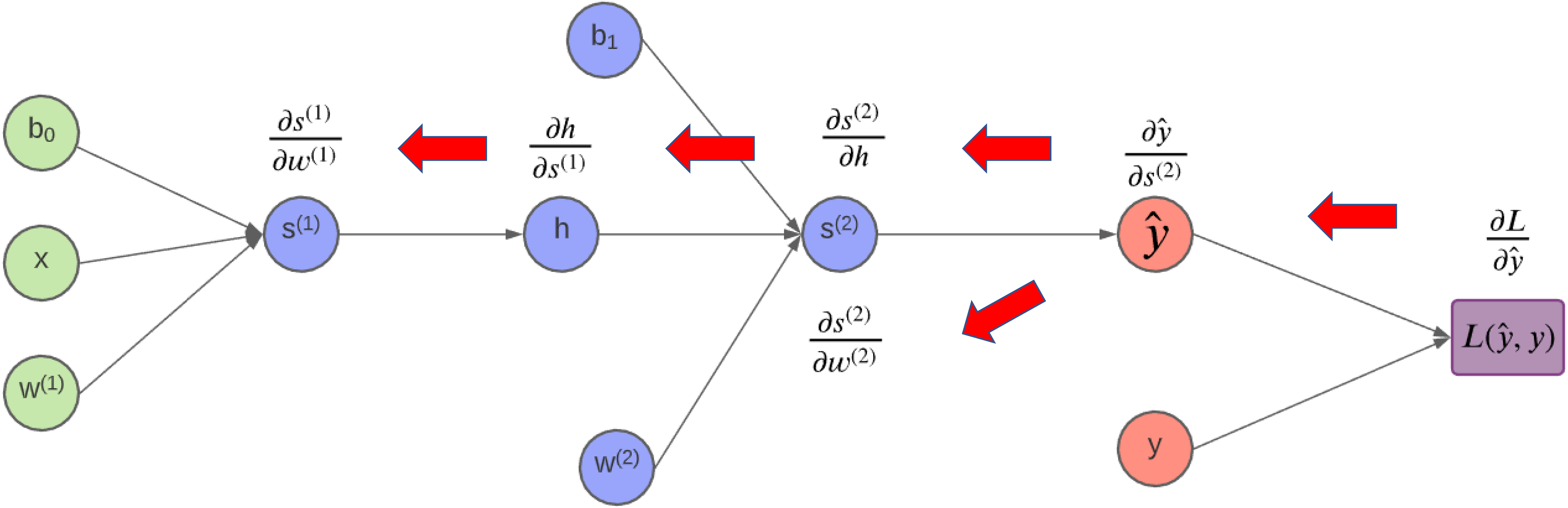
f′(x) negative on the left

f′(x) positive on the right

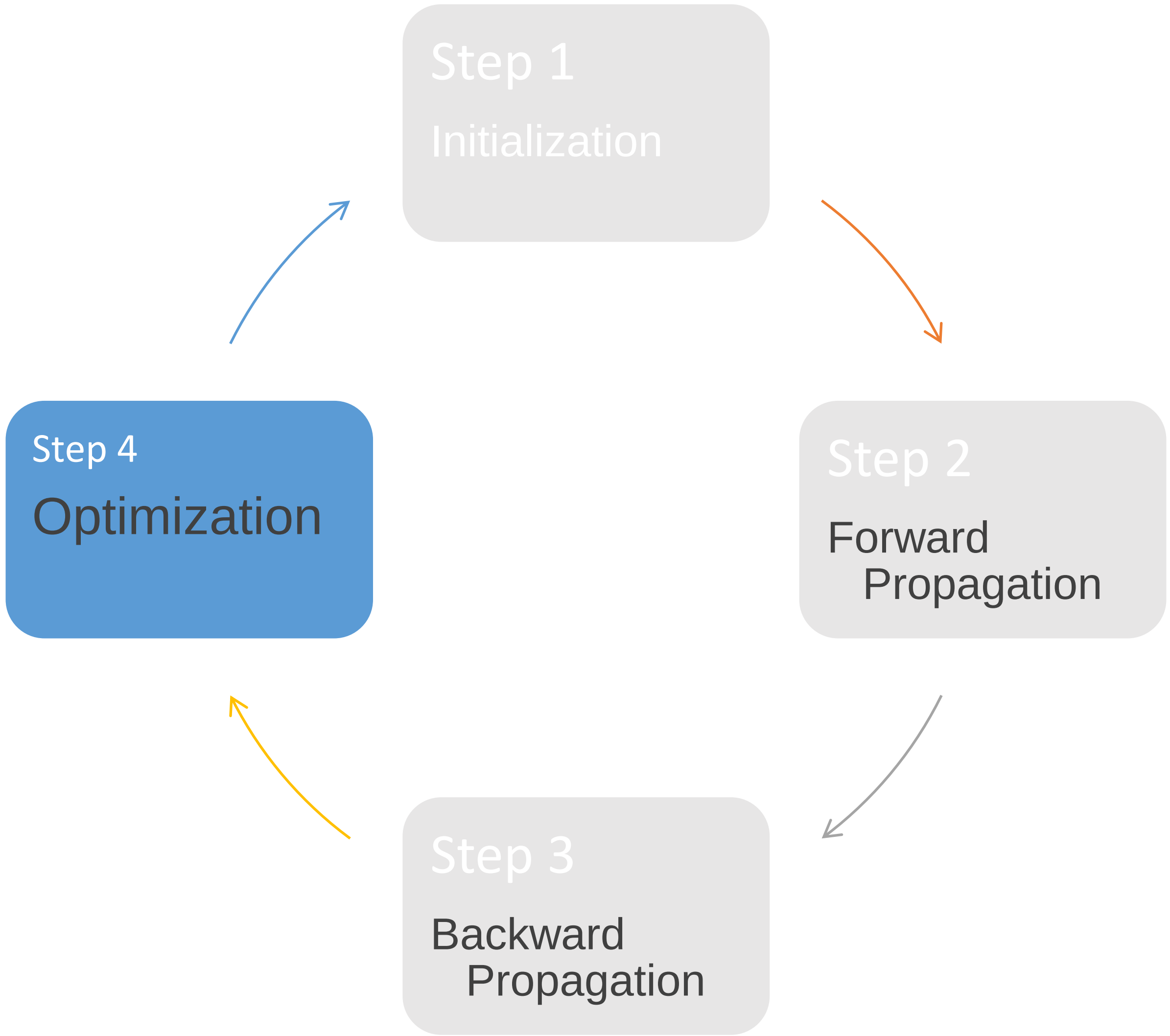


Source: https://xaktly.com/DerivativesIII.html

XOR - Gradient Descent

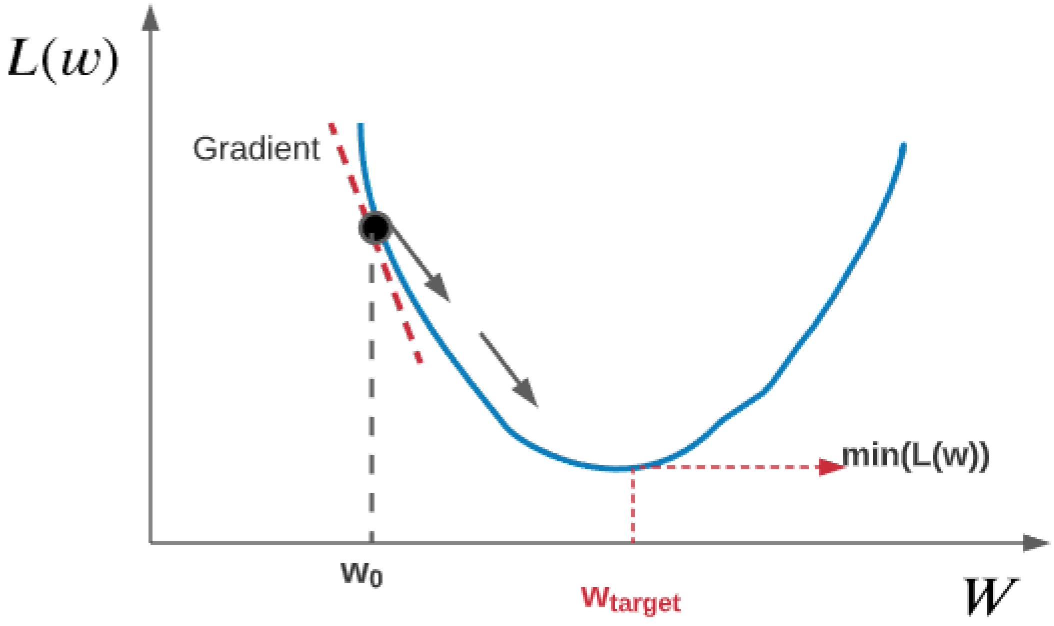


Optimization



XOR - Optimization

Loss Gradient



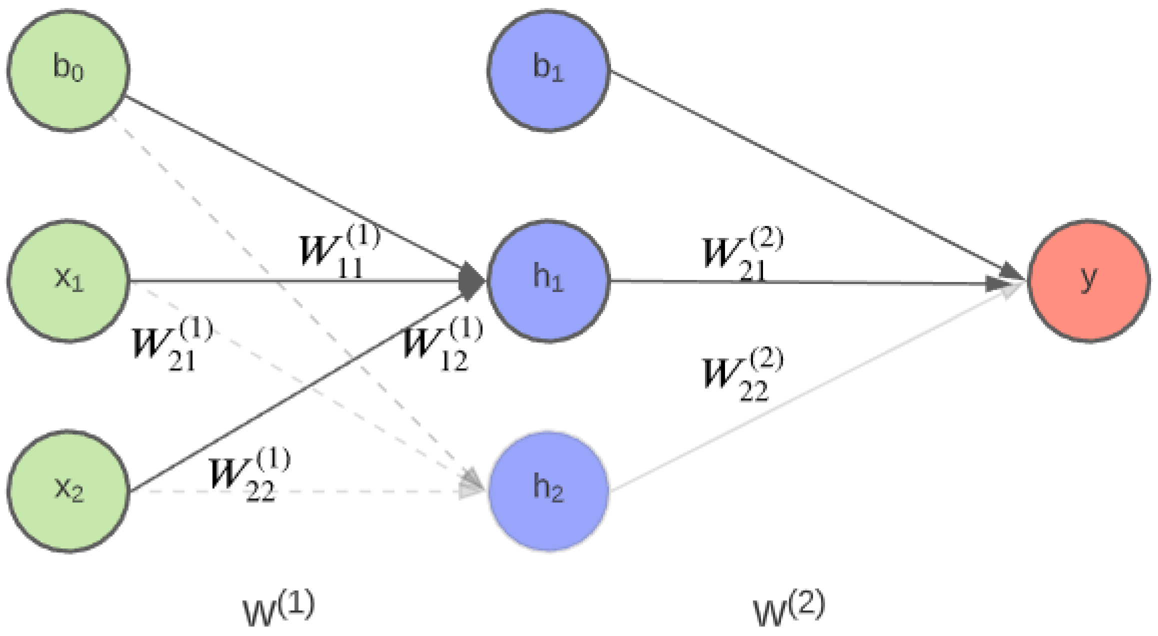
$$L(\hat{y}, y) = \frac{1}{n}(\hat{y} - y)^2 = F(w)$$

Gradient $\dfrac{\partial L}{\partial w}$

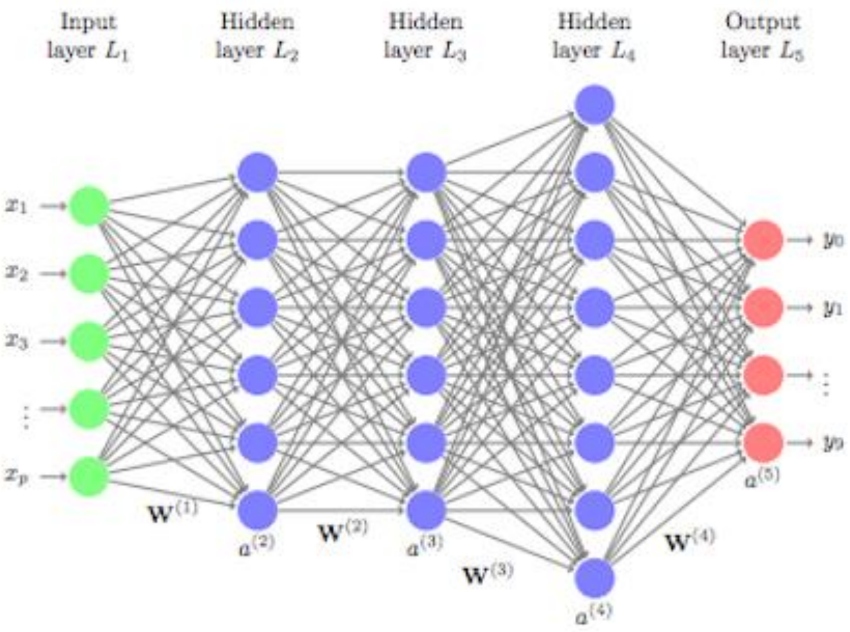
Updating Weight

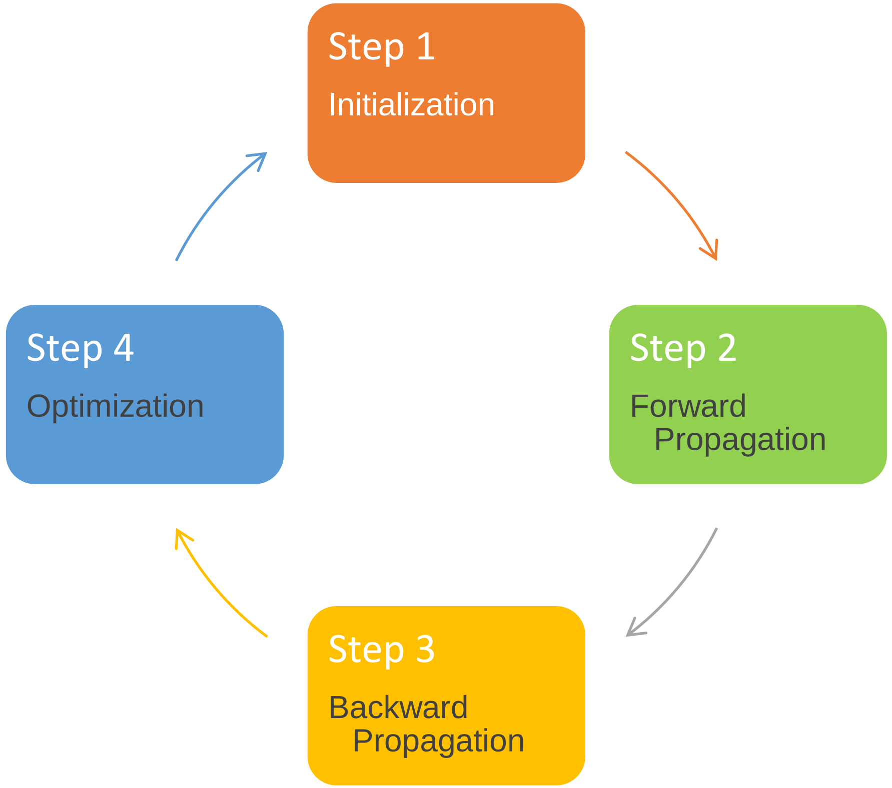
$$W_{t+1} = W_t - \alpha \frac{\partial L}{\partial w}$$

- W_{t+1} is new weights matrix
- W_t is old weights matrix
- α is learning rate



Conclusion


Input layer L_1
Hidden layer L_2
Hidden layer L_3
Hidden layer L_4
Output layer L_5
x_1
x_2
x_3
x_p
$\mathbf{W}^{(1)}$
$\mathbf{W}^{(2)}$
$\mathbf{W}^{(3)}$
$\mathbf{W}^{(4)}$
$a^{(2)}$
$a^{(3)}$
$a^{(4)}$
$a^{(5)}$
y_0
y_1
y_9


Step 1
Initialization
Step 2
Forward Propagation
Step 3
Backward Propagation
Step 4
Optimization

Outline

- Background
- AI Model Training Routine (XOR use case)
- Classic AI Models

Complicated AI Models

Layers	Activation Function	Cost/Loss Function	NN Type
1	Tanh	Mean Absolute (MA)	Convolution
10	Sigmoid	Mean Squared Error (MSE)	Recurrent
100	Softmax	Cross Entropy (CE)	Transformer

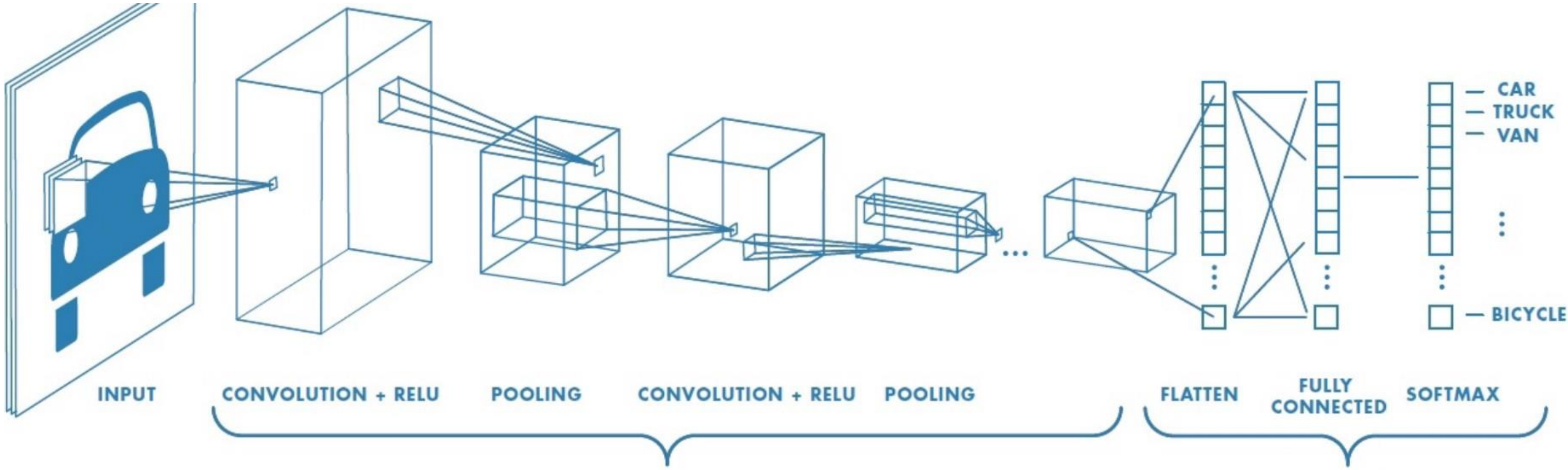
⋮

Classica AI Models

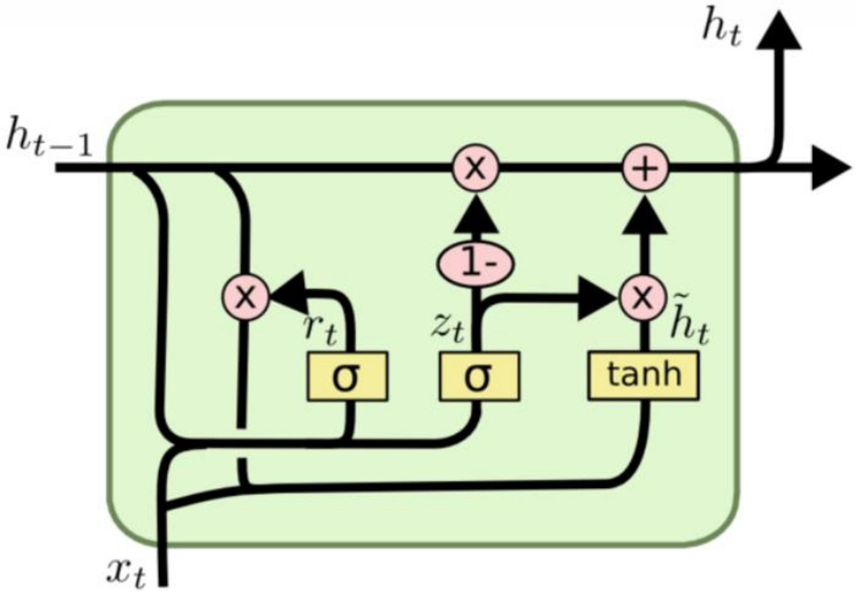
- CNN (Convolutional Neural Network)
- RNN (Recurrent Neural Network)
- GNN (Graph Neural Network)
- Transformer
- GAN (Generative Adversarial Network)
- …

CV: CNN (Convolutional Neural Network)

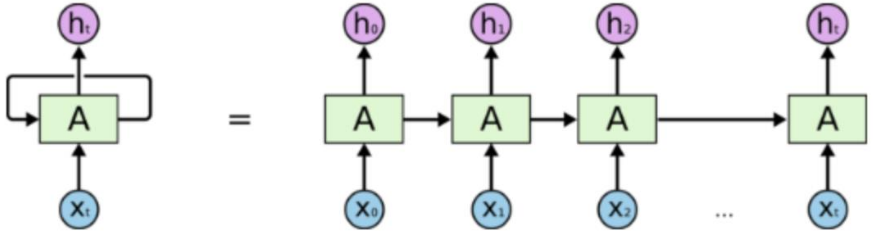
- Input: image
- Application: CV (object classification, object detection, …)



NLP: RNN & LSTM



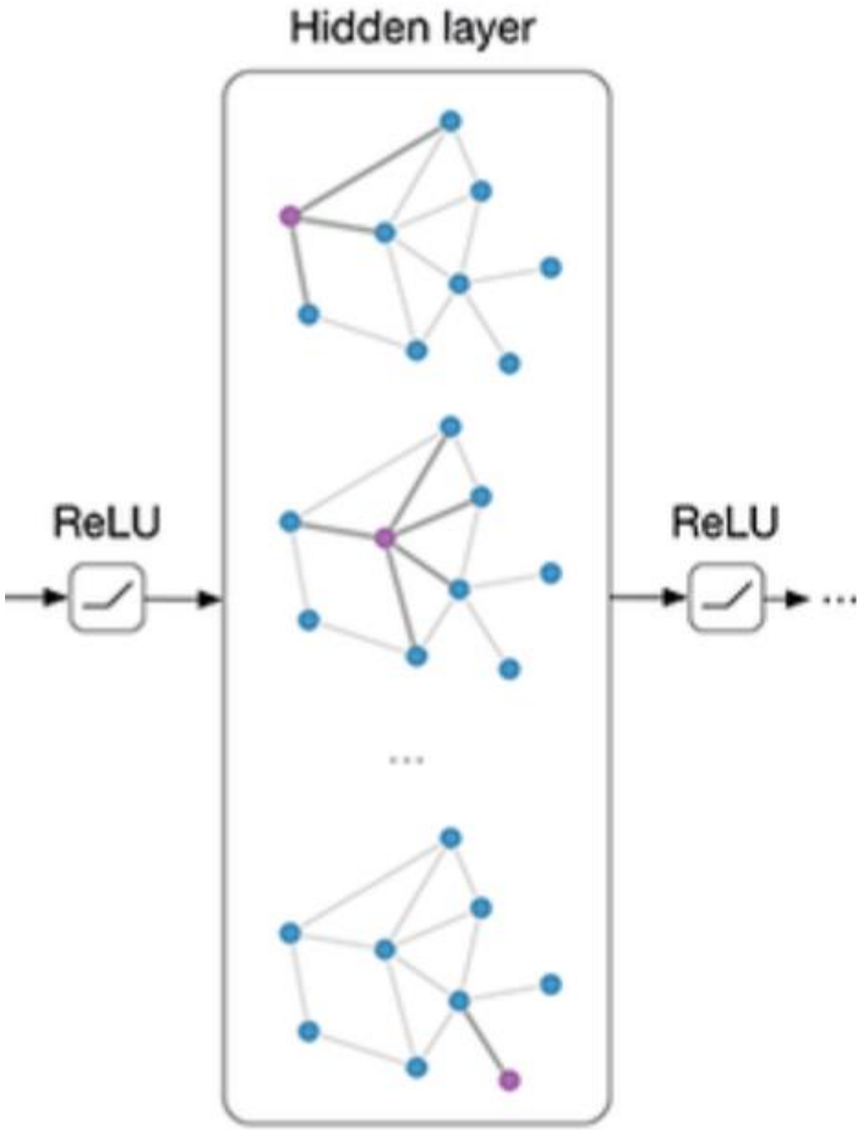
- Input: text sequence
- Application: NLP (machine translation, classification, sentiment analysis, …)



GNN

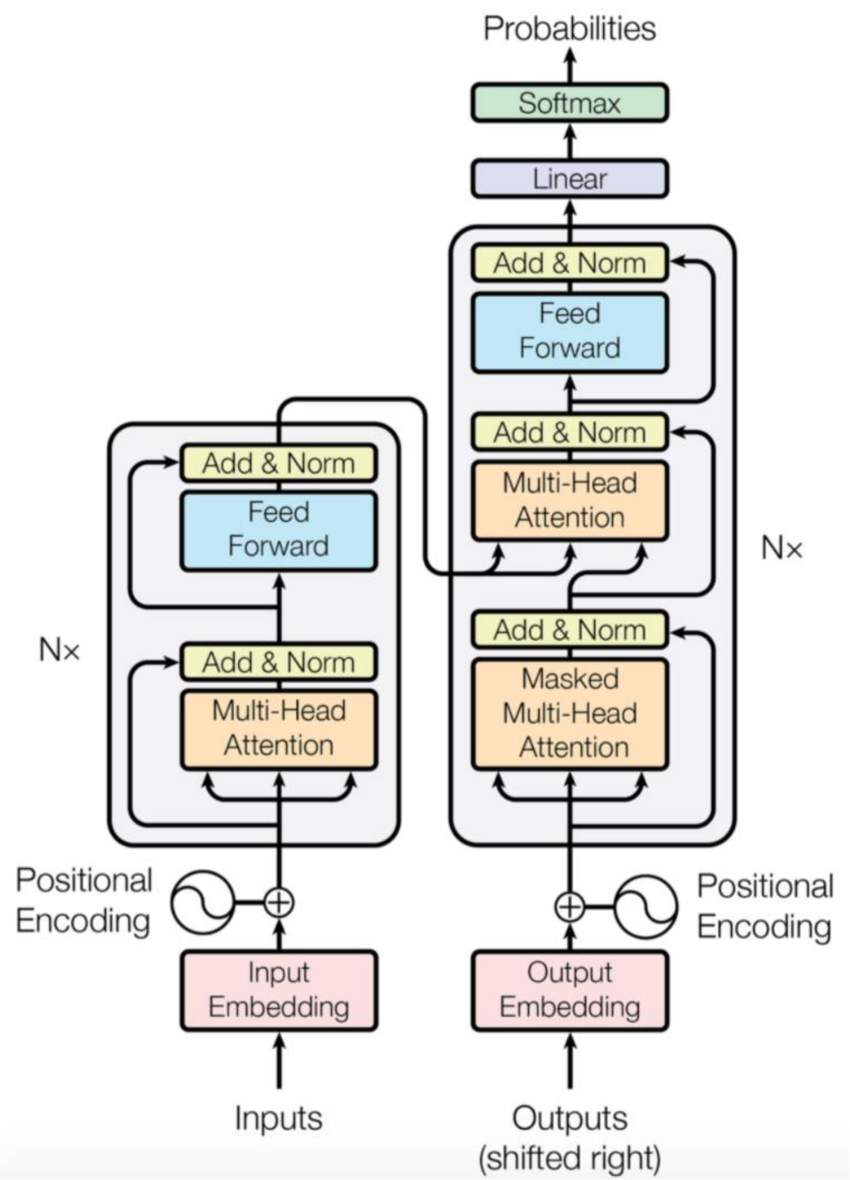
Input: graph structure (map data, nano-scale molecules)

Application: drug discovery, route optimization



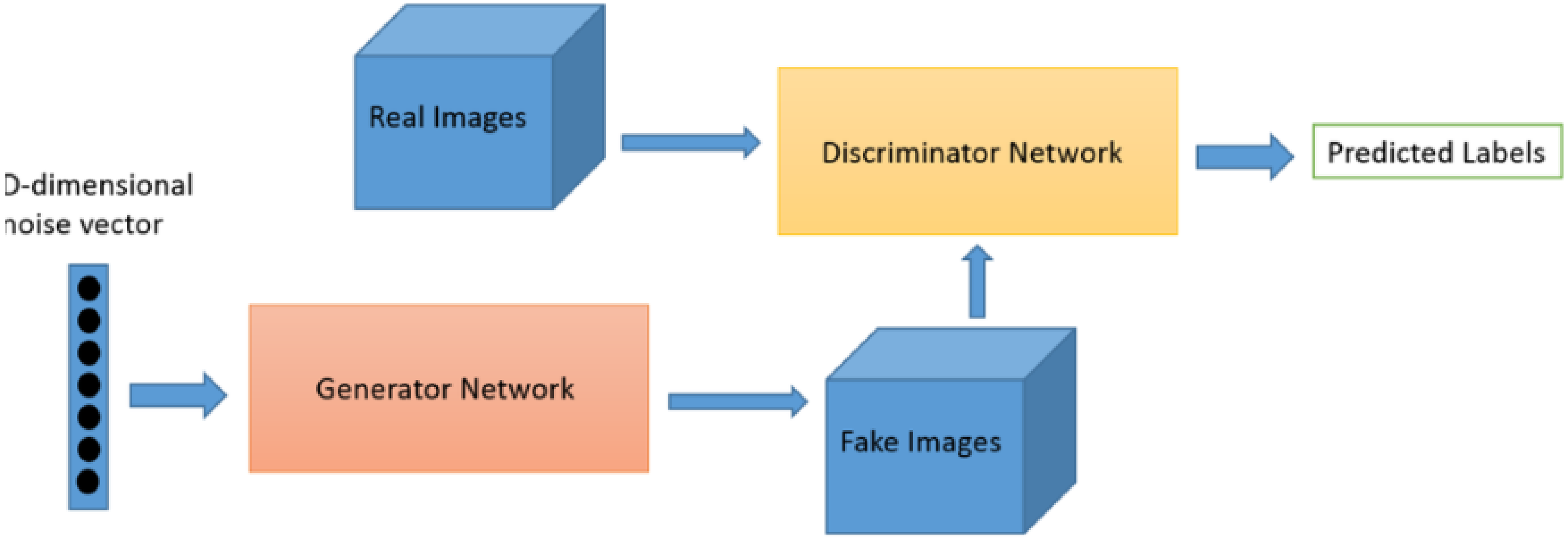
CV&NLP - Transformer

- Input: image or text
- Application: transfer learning



CV- GAN

- Input: photos, paintings …
- Application: generating image, constructing 3D models from images, …



Demo

Q & A