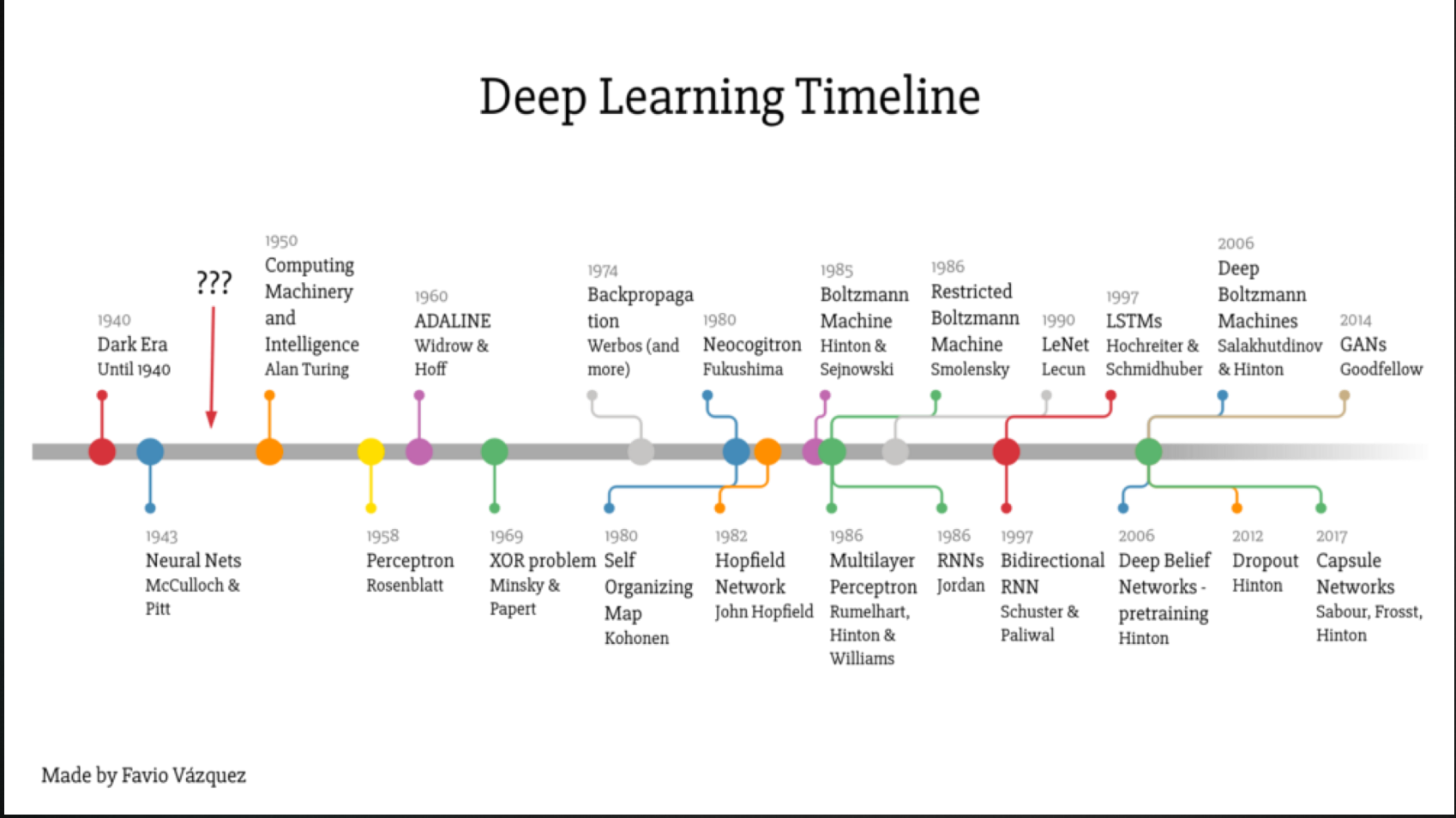
Neural networks

GUILLEM GUIGÓ I COROMINAS

A bit of history



2004 - Oh, Kyoung-Su, and Keechul Jung. "GPU implementation of neural networks." *Pattern Recognition* 37.6 (2004): 1311-1314.

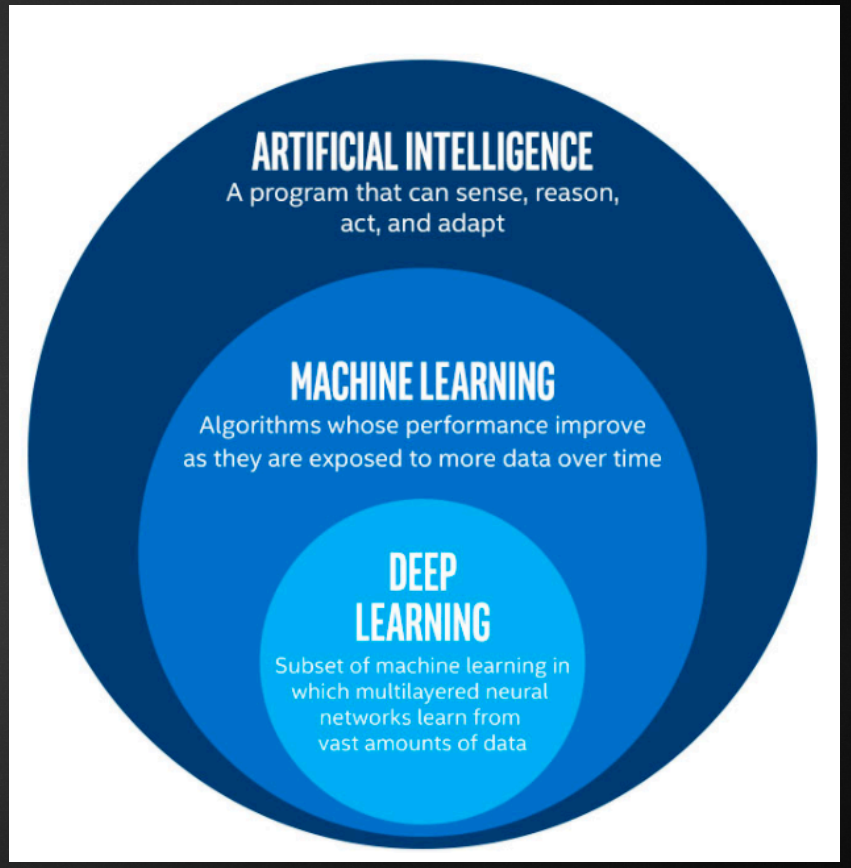
2012 - AlexNet: top 5 error of 15.8% on ImageNet

2015 - ResNet: 3.57% error on the ImageNet test set. Won the 1st place on the ILSVRC 2015 classification task.

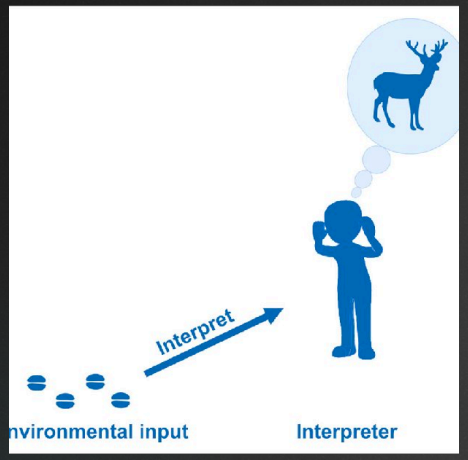
2017 - Transformers introduced by Google Brain

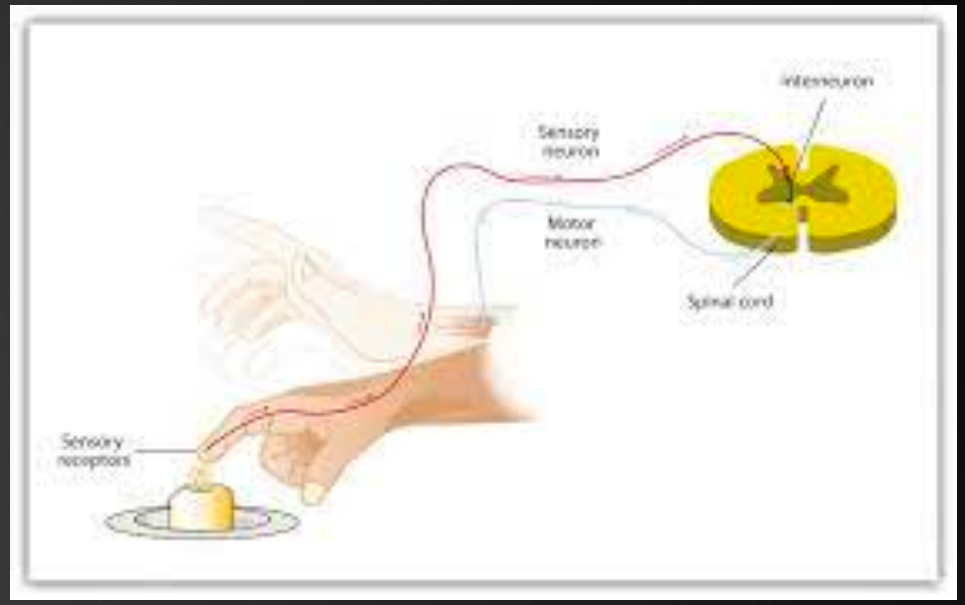
Deep learning

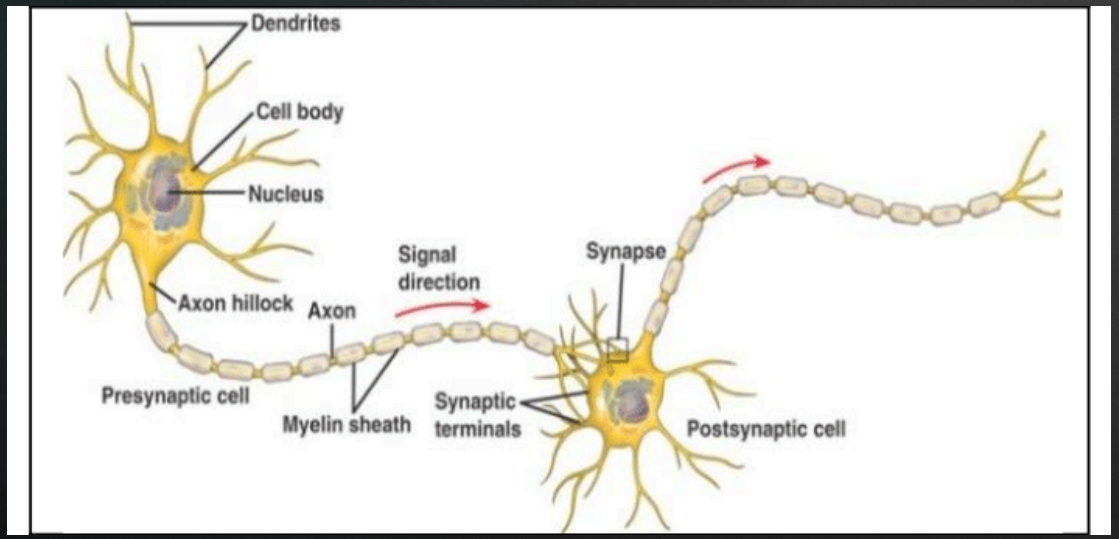
- **Specialized field of Machine Learning that is based on the training of Deep Artificial Neural Networks (ANNs)**
- Popular when using large datasets, e.g. images, text, audio
- **ANNs models are inspired by the way human brain "functions"**
- Human brain consists of billions of neurons that communicate with each other using electrical and chemical signals to enable humans to see, feel and make decisions



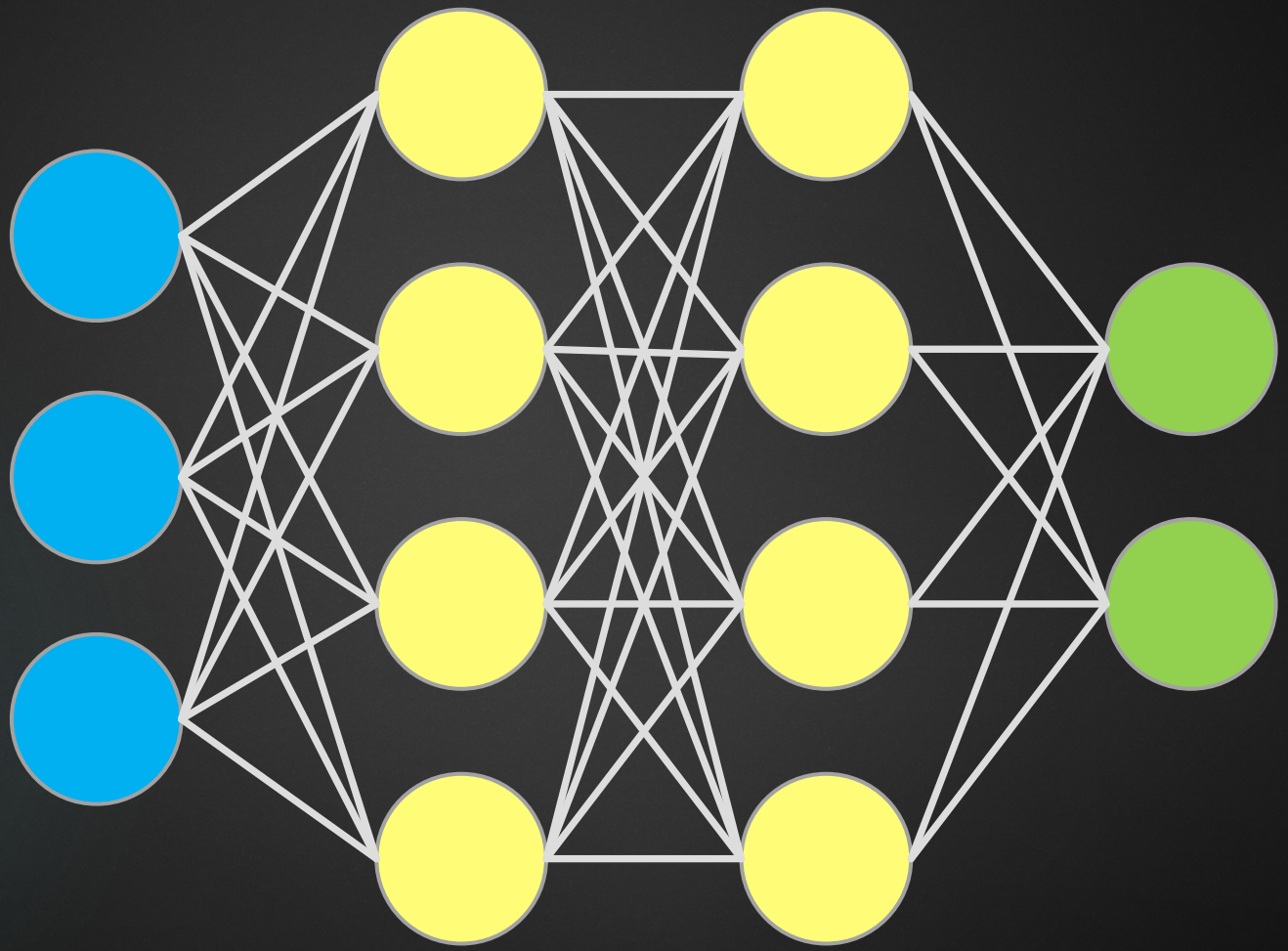
The biological neuron







Artificial Neural Network

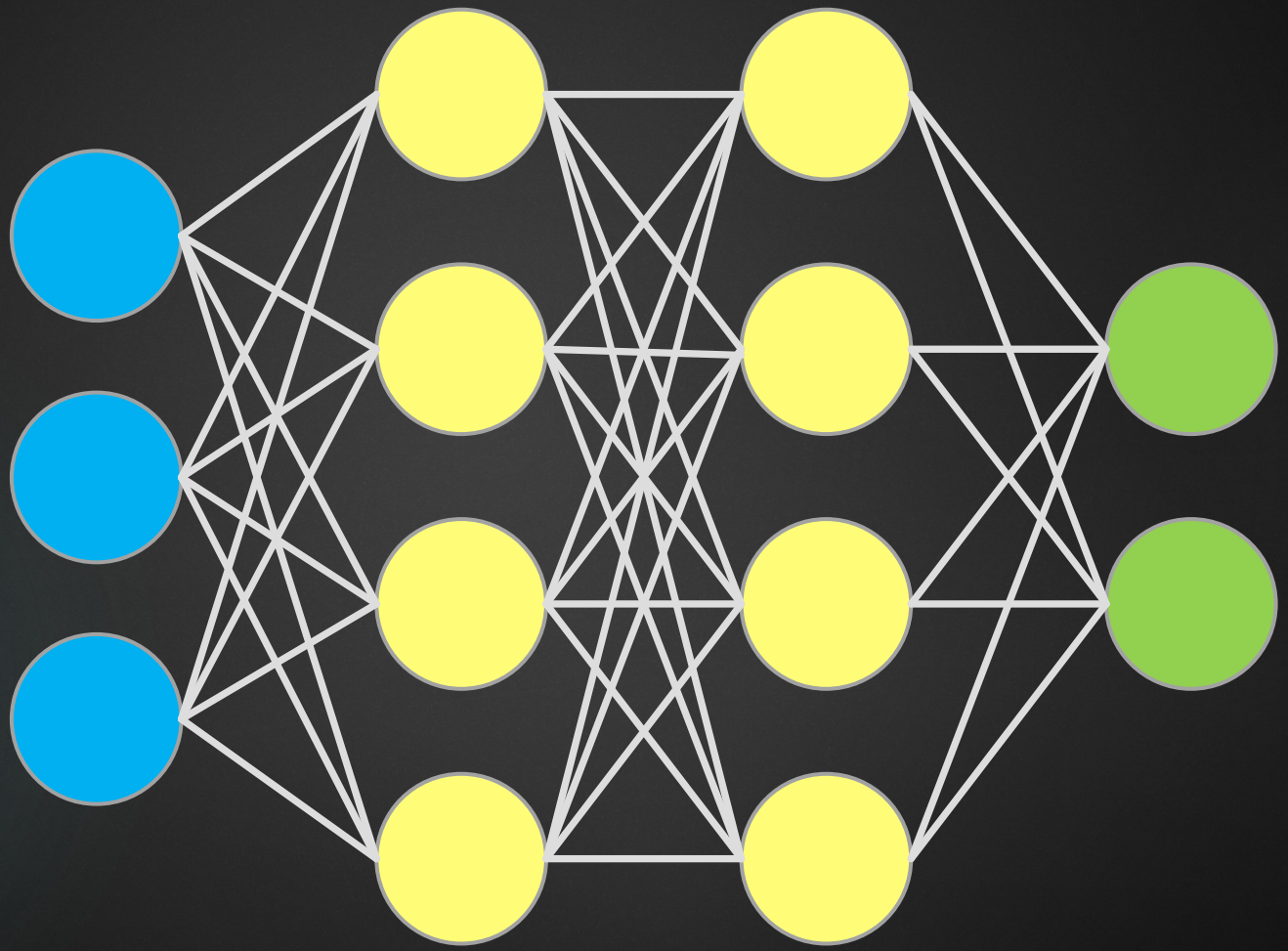


Artificial Neural Network

Humidity: 60%

Temperature: 17°C

Pressure: 102HPa



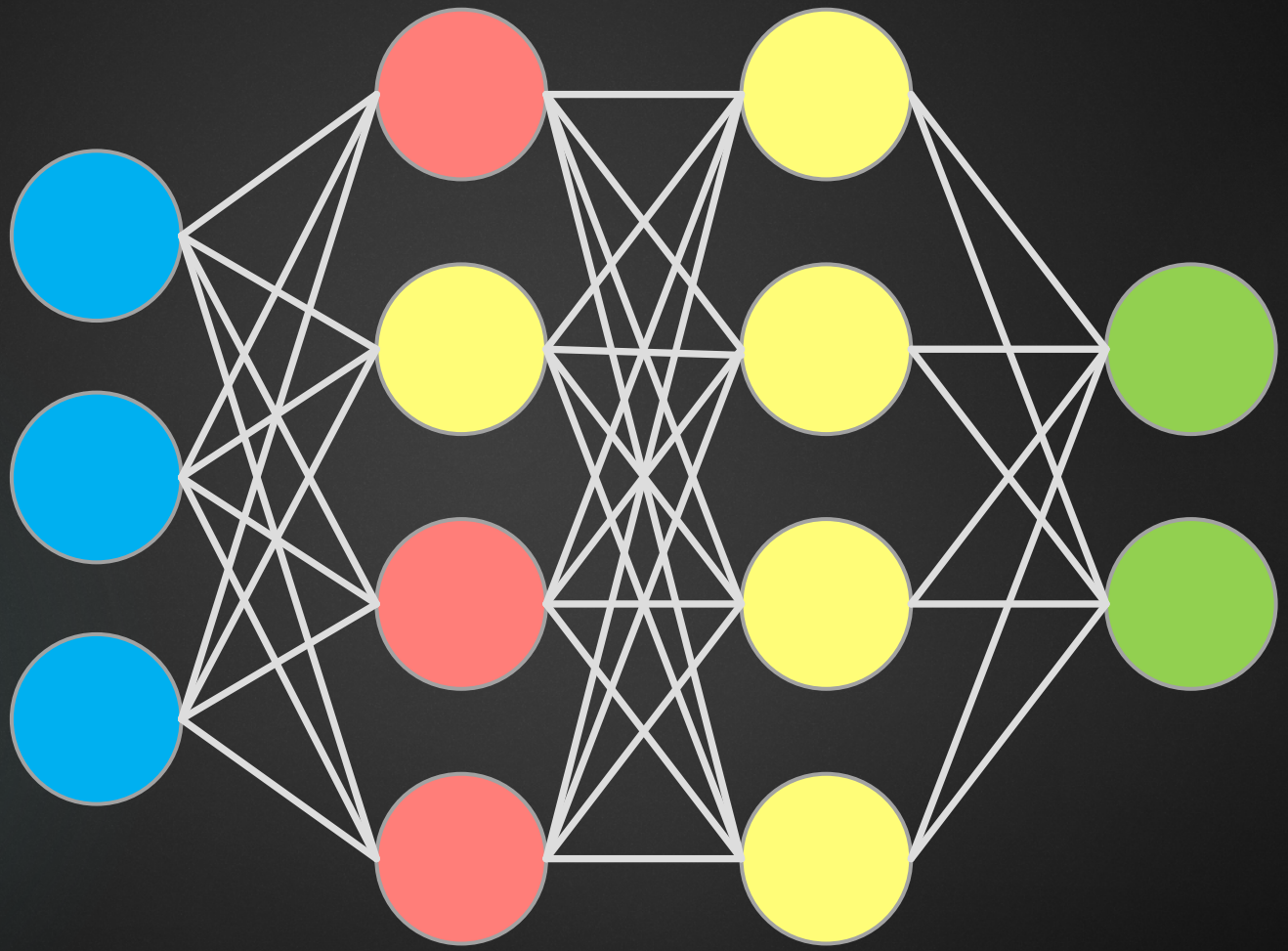
input Hidden layers output

Artificial Neural Network

Humidity: 60%

Temperature: 17°C

Pressure: 102HPa



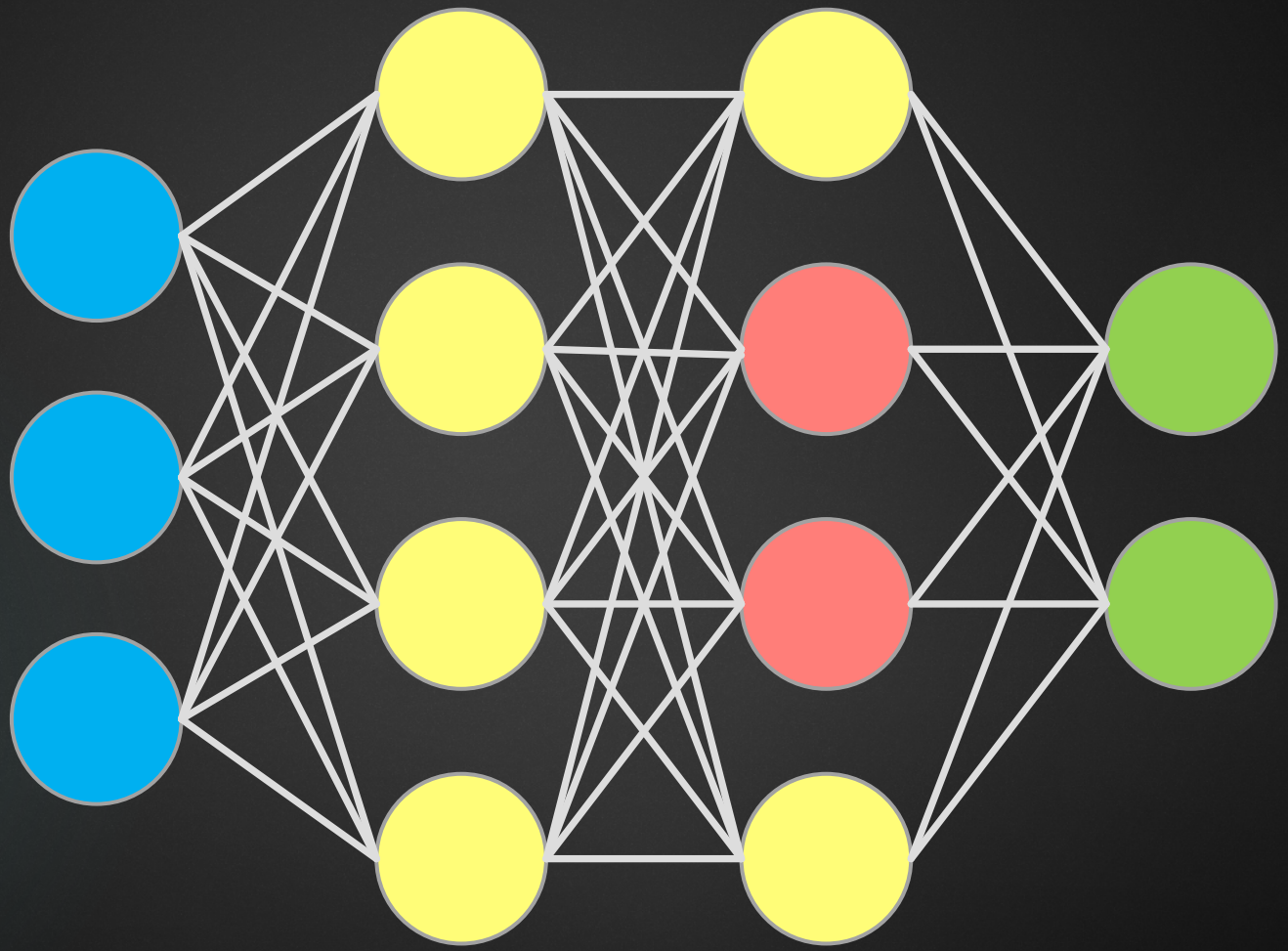
input Hidden layers output

Artificial Neural Network

Humidity: 60%

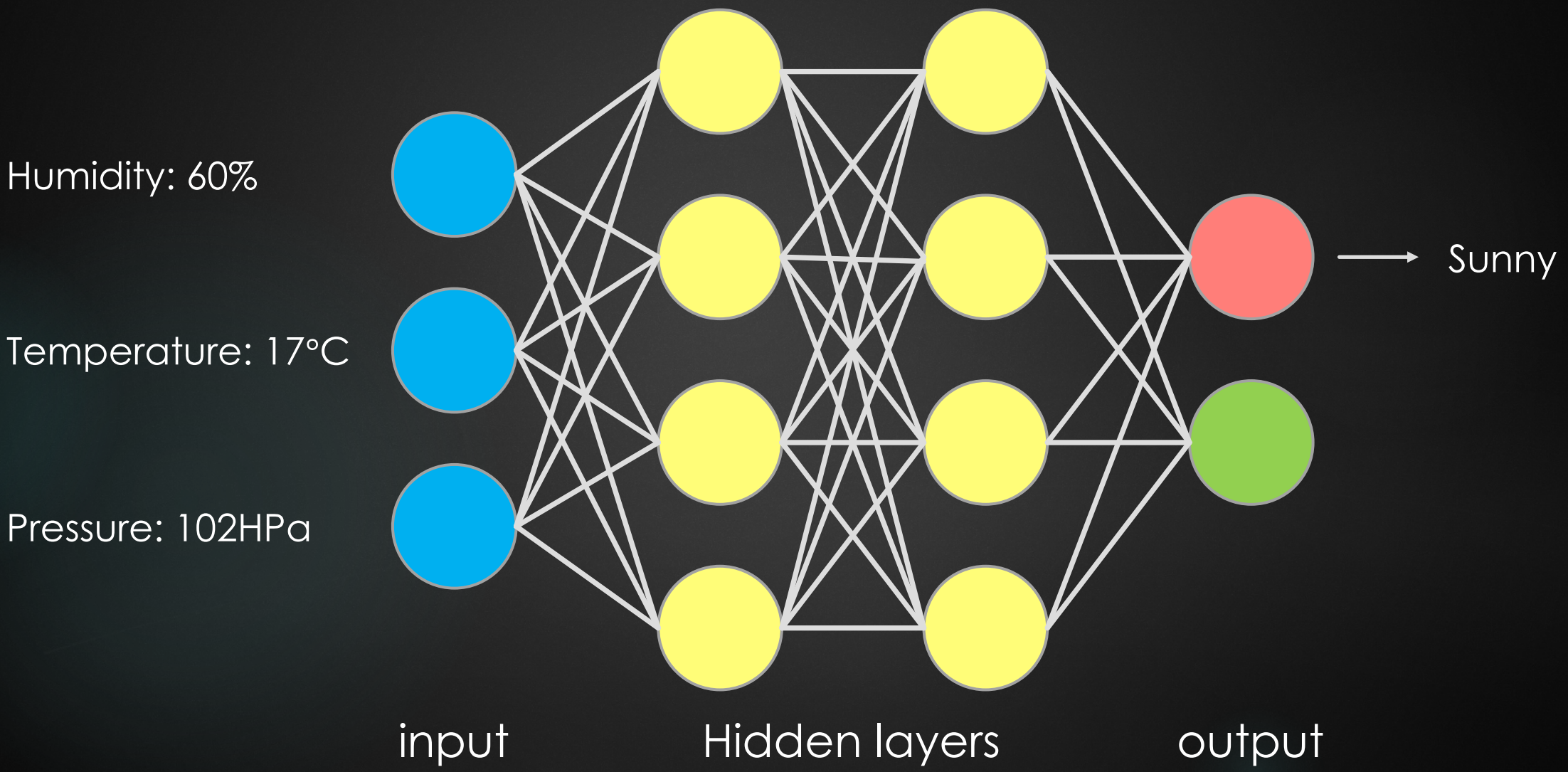
Temperature: 17°C

Pressure: 102HPa



input Hidden layers output

Artificial Neural Network

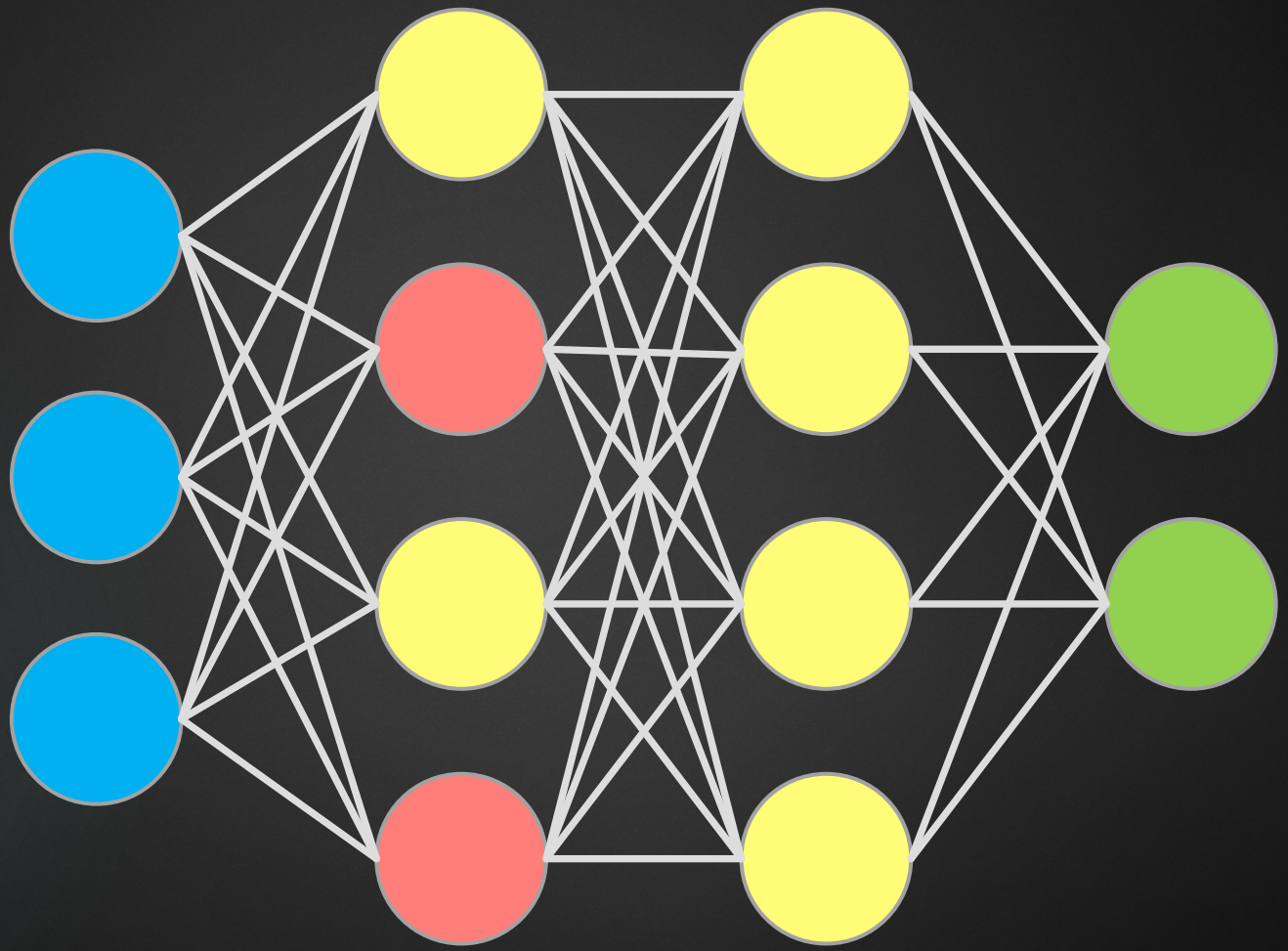


Artificial Neural Network

Humidity: 90%

Temperature: 16°C

Pressure: 100HPa



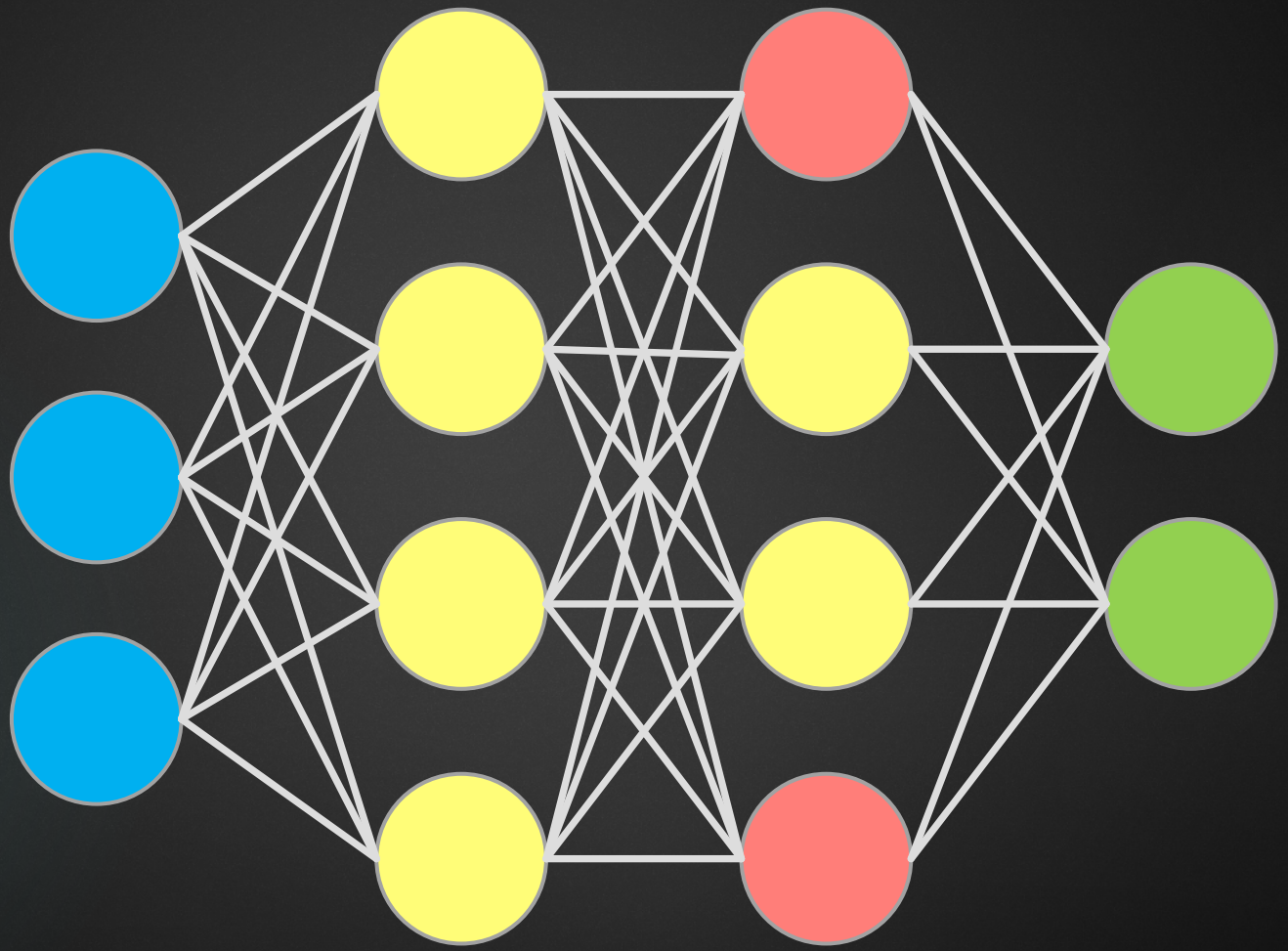
input Hidden layers output

Artificial Neural Network

Humidity: 90%

Temperature: 16°C

Pressure: 100HPa



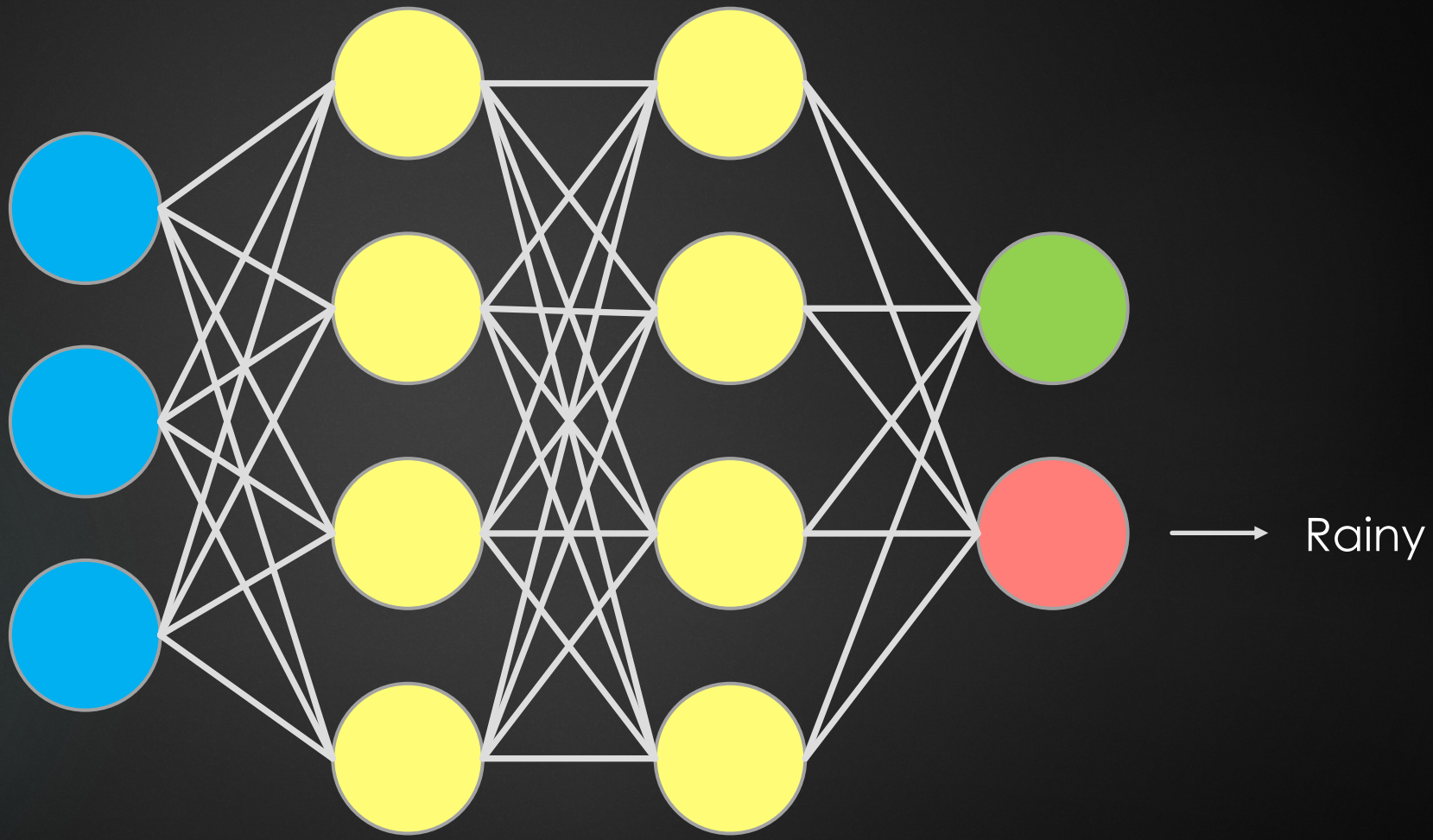
input Hidden layers output

Artificial Neural Network

Humidity: 90%

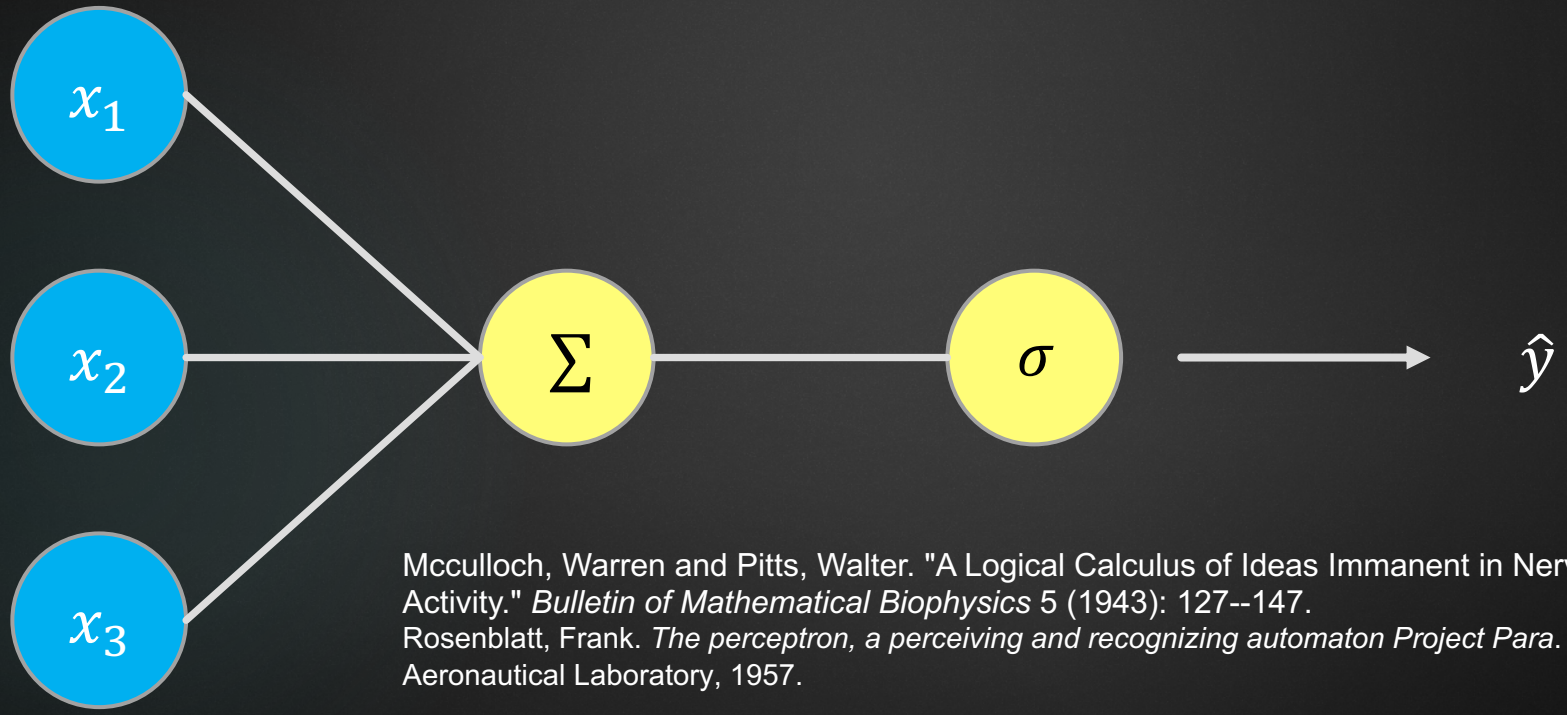
Temperature: 16°C

Pressure: 100HPa



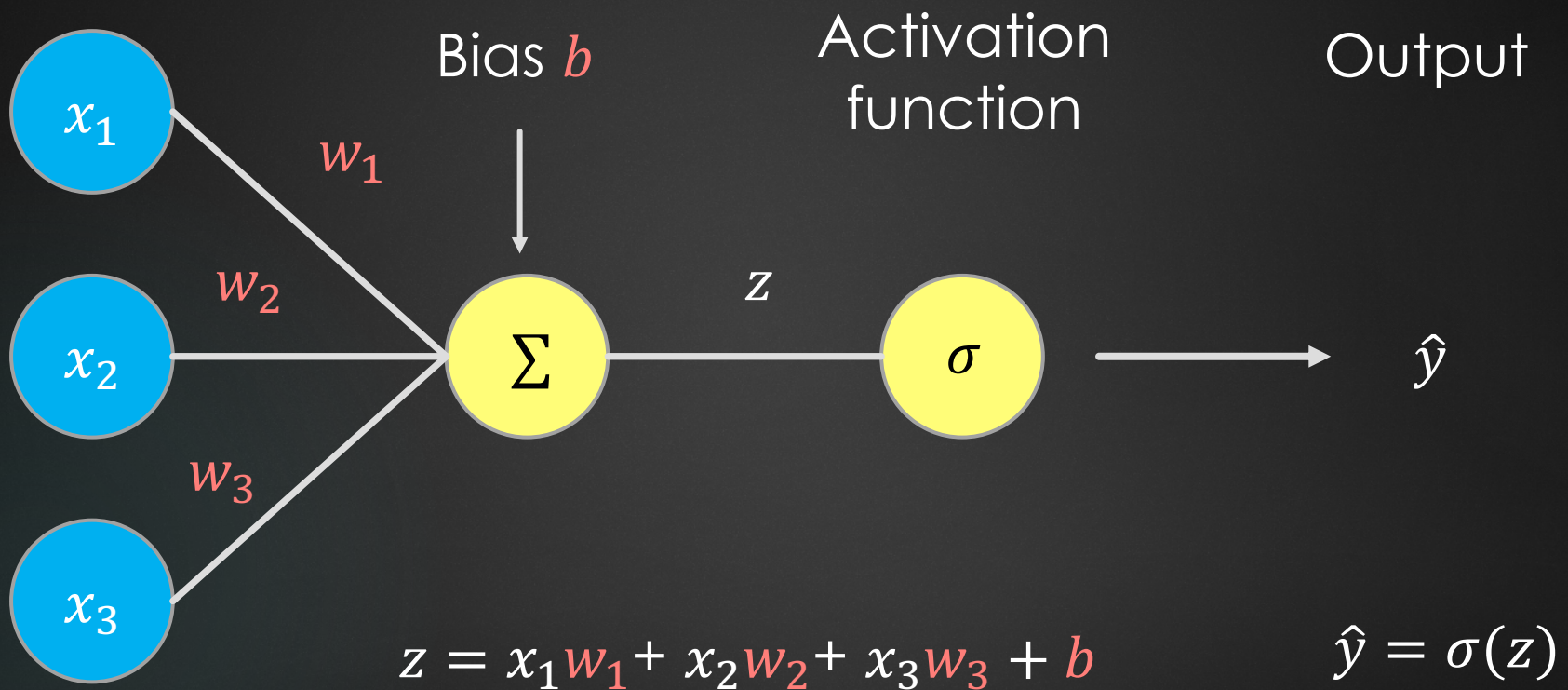
The perceptron

- McCulloch and Pits (1943) - Description of the model
- Frank Rosenblatt (1957) – Machine implementation



Mcculloch, Warren and Pitts, Walter. "A Logical Calculus of Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics* 5 (1943): 127--147.
Rosenblatt, Frank. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

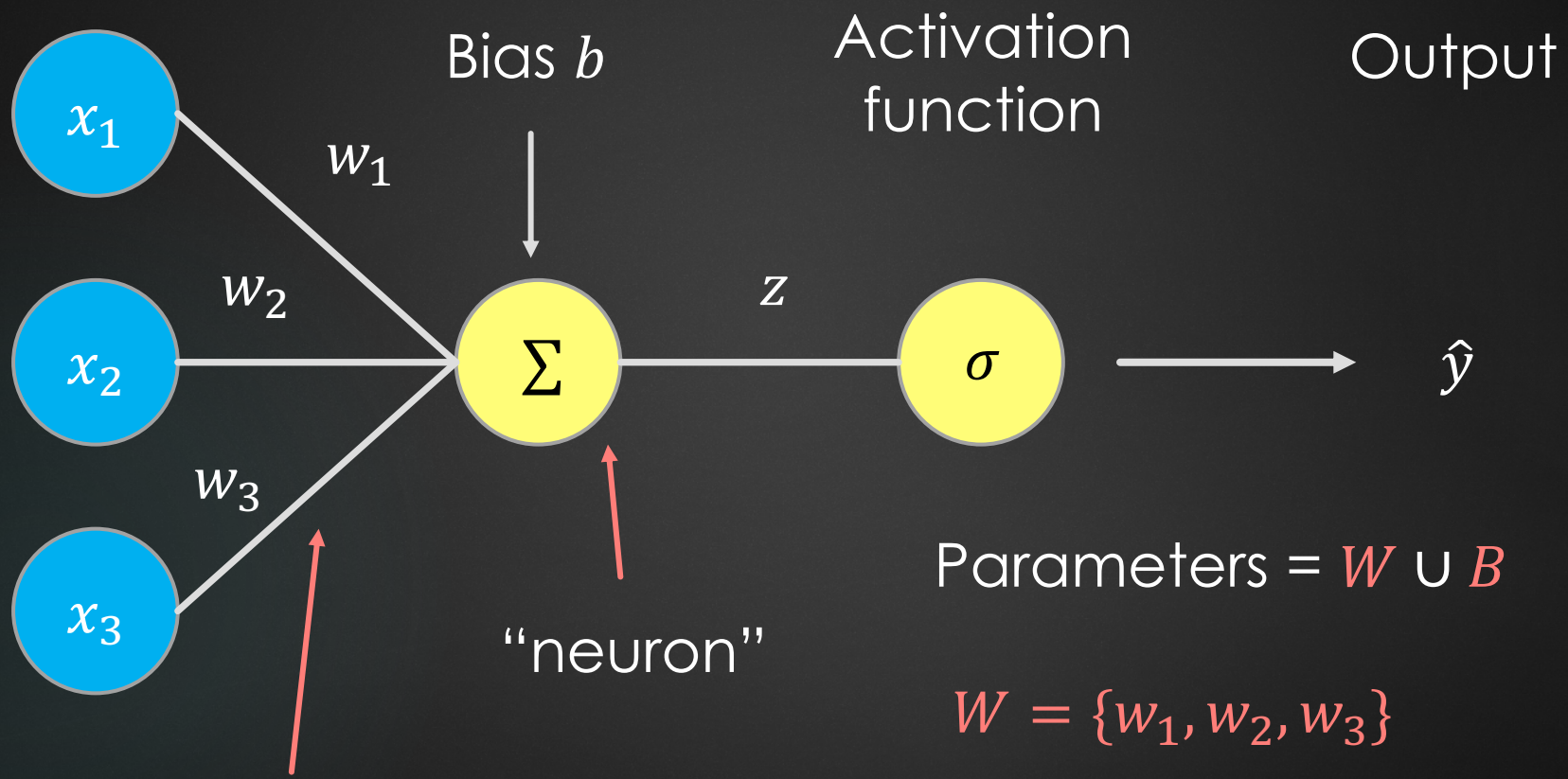
The perceptron



$$z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$\hat{y} = \sigma(z)$$

The perceptron

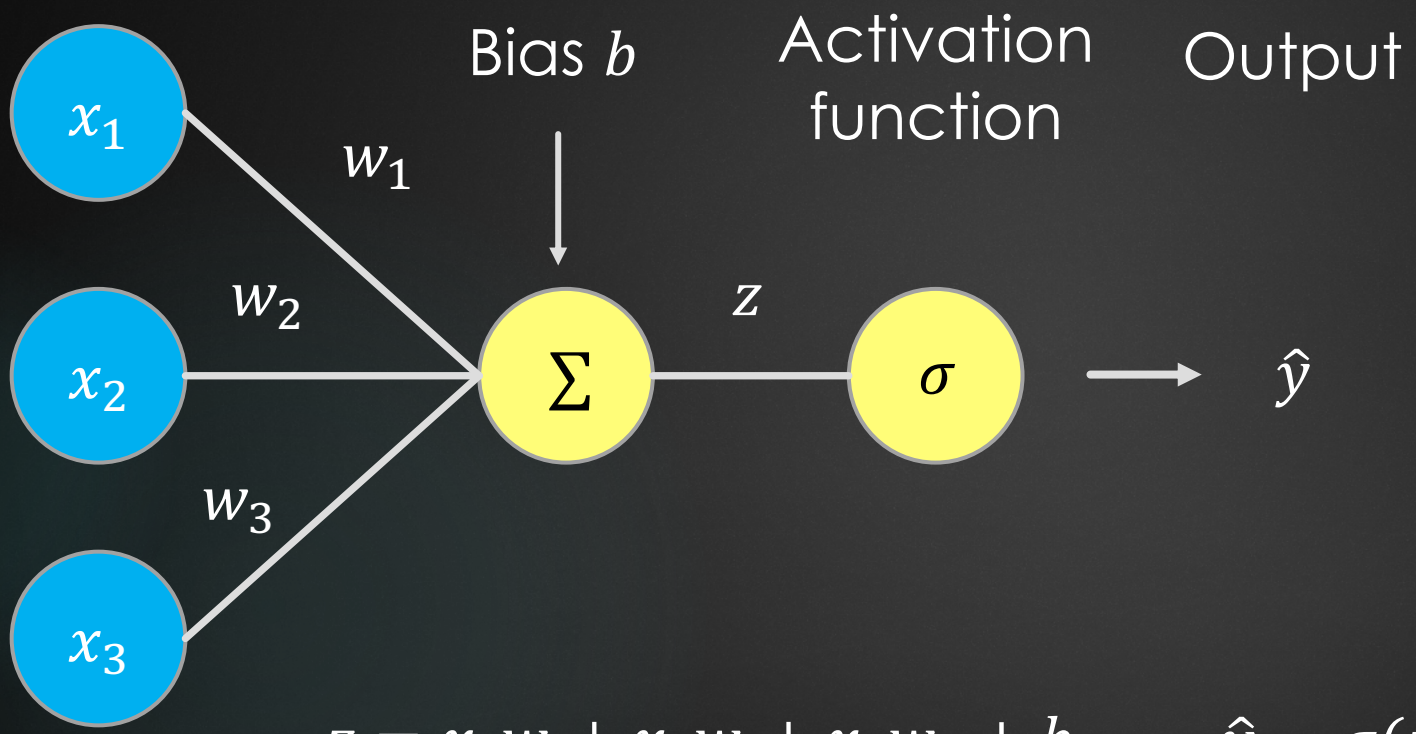


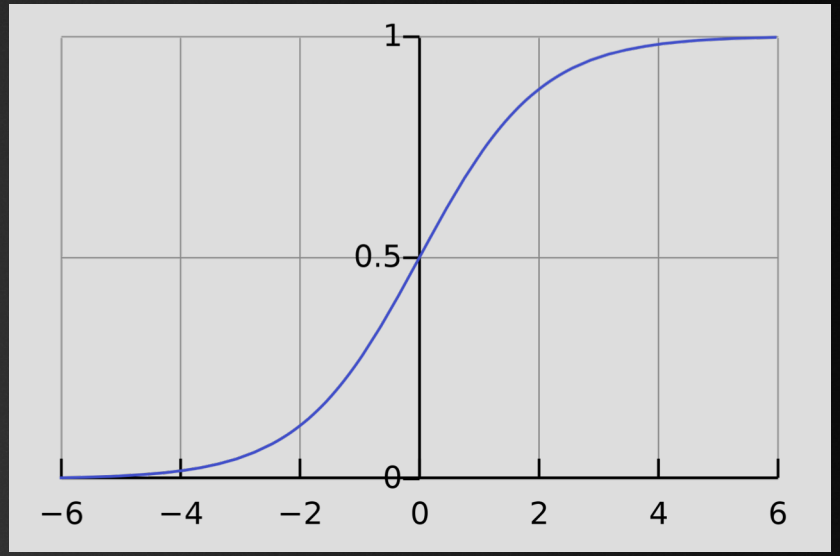
Parameters = $W \cup B$

$W = \{w_1, w_2, w_3\}$

$B = \{b\}$

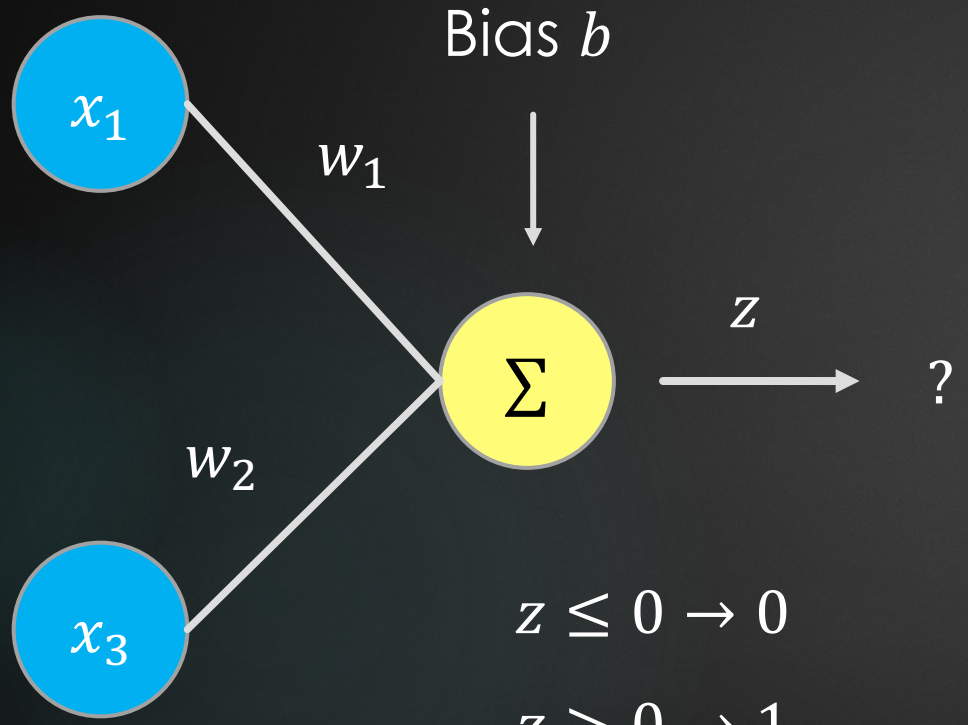
The perceptron



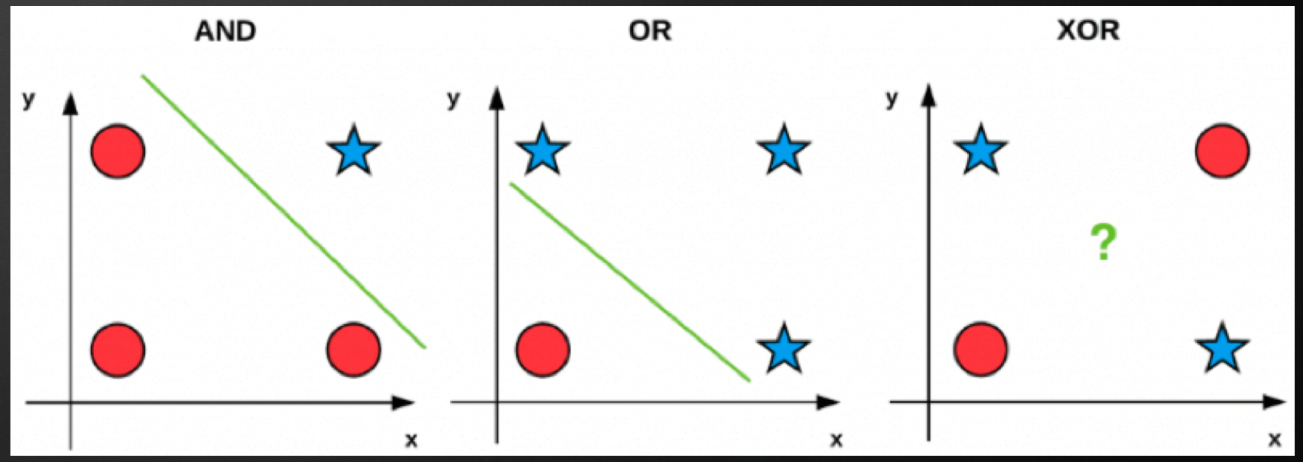


$$z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b \qquad \hat{y} = \sigma(z) \qquad \sigma(z) = \frac{1}{1 + e^{-z}}$$

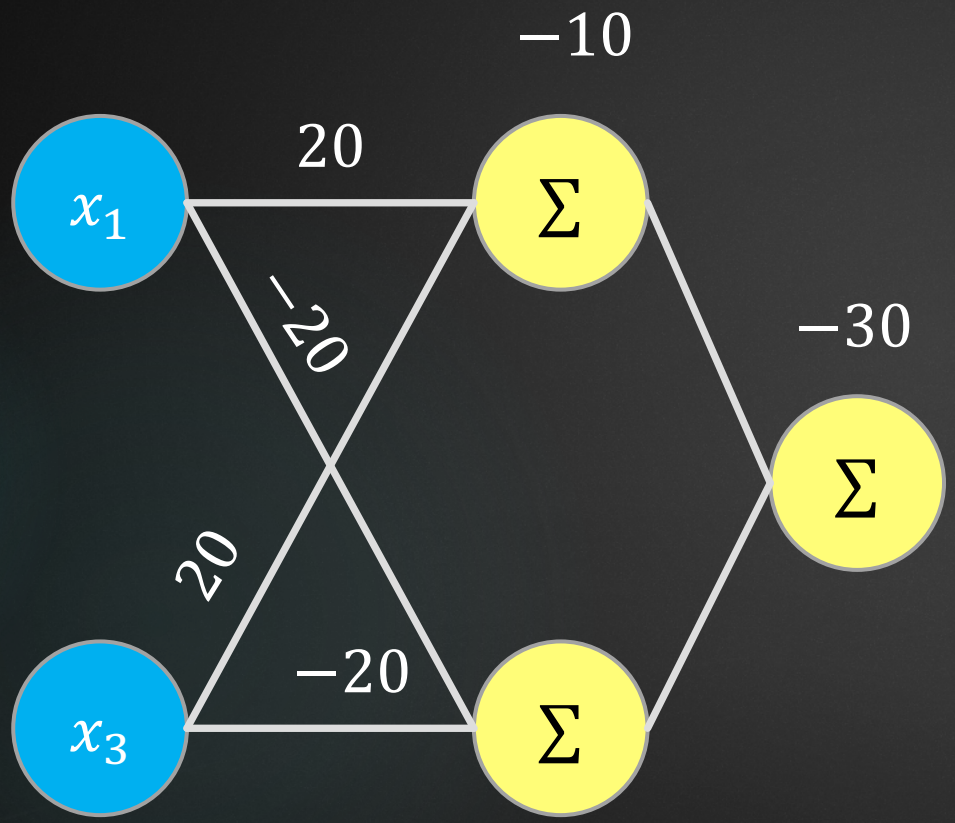
The perceptron and the XOR problem

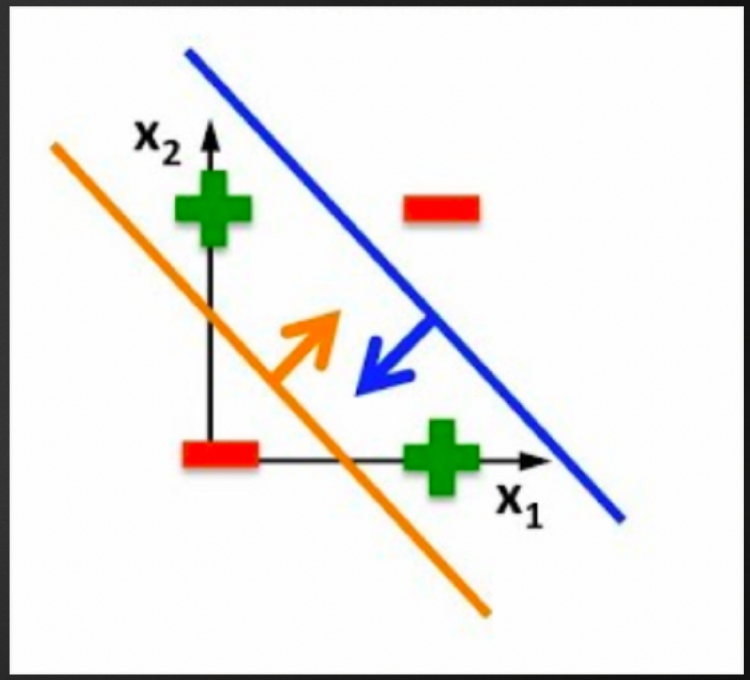


x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0



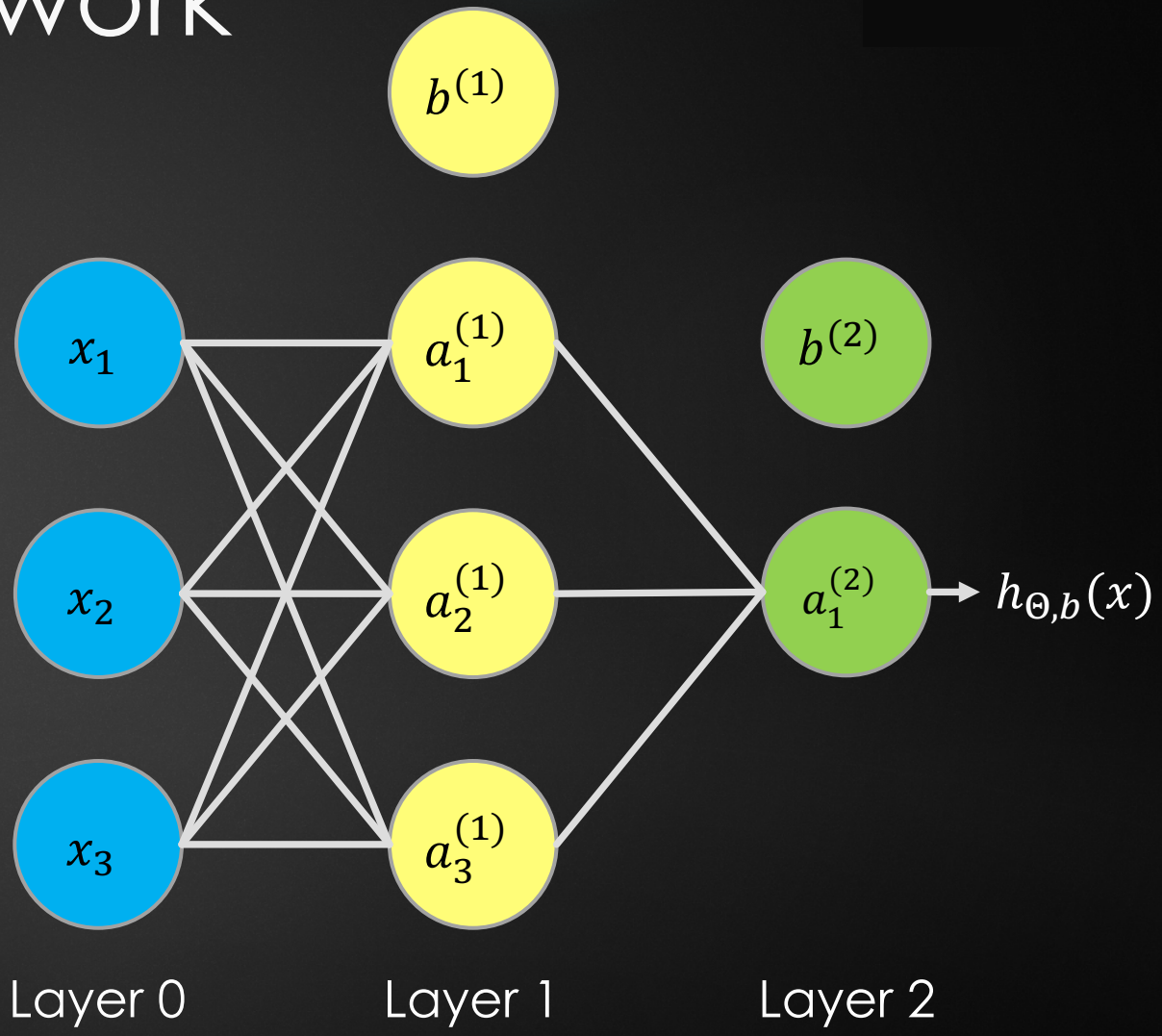
The perceptron and the XOR problem





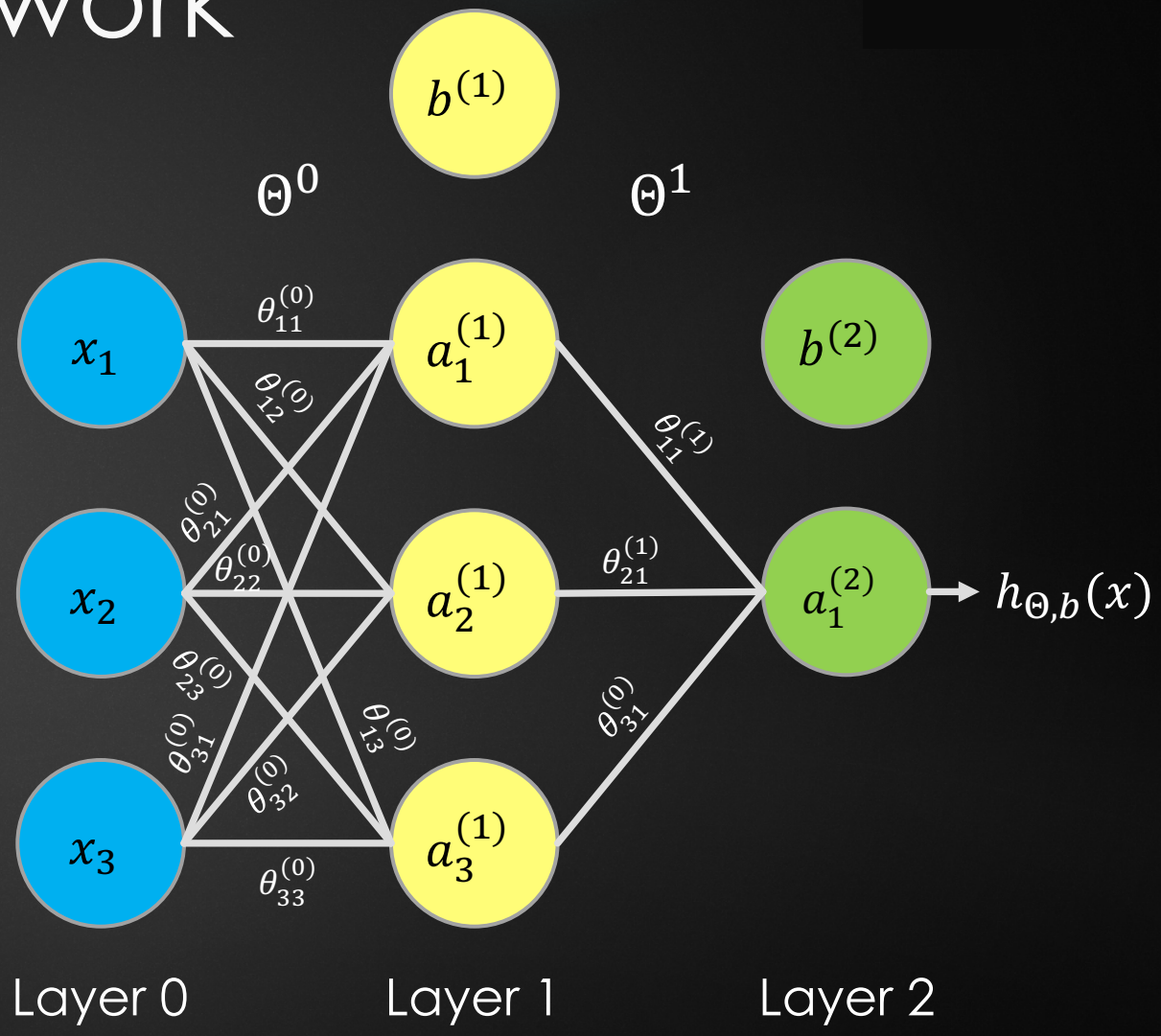
Building a neural network

- **Input layer (layer 0):** this layer consists of the input features x_1, x_2 and x_3
- **Hidden layer (layer 1):** contains values that we don't observe in the training set
- **Output layer (layer 2):** it has a neuron/units (can have more units) that outputs the final value computed by the hypothesis

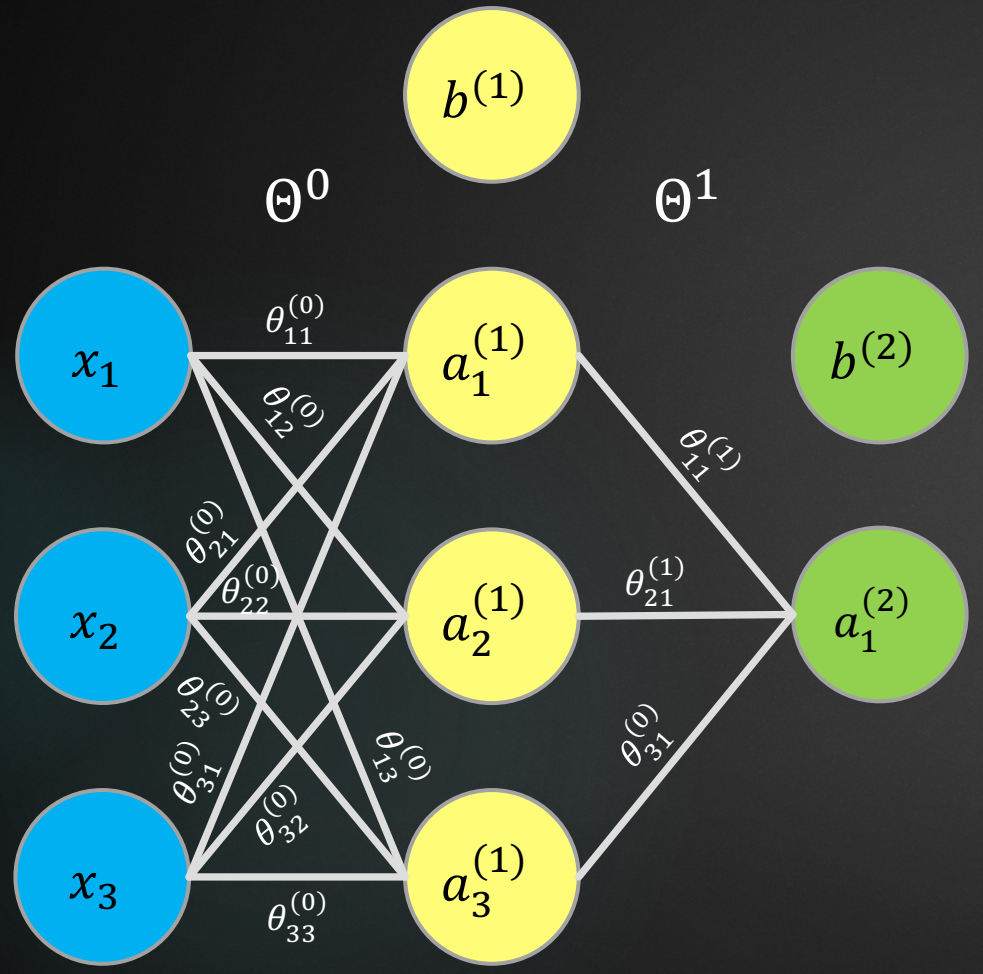


Building a neural network

- $a_i^{(j)}$ - Activation unit i in layer j
- $\Theta^{(j)}$ - Matrix of weights that controls the function mapping from j to layer $j + 1$
- $b^{(j)}$ - Bias vector in layer j



Building a neural network



$$a_1^{(1)} = g(\theta_{11}^{(0)}x_1 + \theta_{21}^{(0)}x_2 + \theta_{31}^{(0)}x_3 + b_1^{(1)})$$

$$a_2^{(1)} = g(\theta_{12}^{(0)}x_1 + \theta_{22}^{(0)}x_2 + \theta_{32}^{(0)}x_3 + b_2^{(1)})$$

$$a_3^{(1)} = g(\theta_{13}^{(0)}x_1 + \theta_{23}^{(0)}x_2 + \theta_{33}^{(0)}x_3 + b_3^{(1)})$$

$$h_{\Theta,b}(x) = a_1^{(2)} = \theta_{11}^{(1)}a_1^{(1)} + \theta_{21}^{(1)}a_2^{(1)} + \theta_{31}^{(1)}a_3^{(1)} + b_1^{(2)}$$

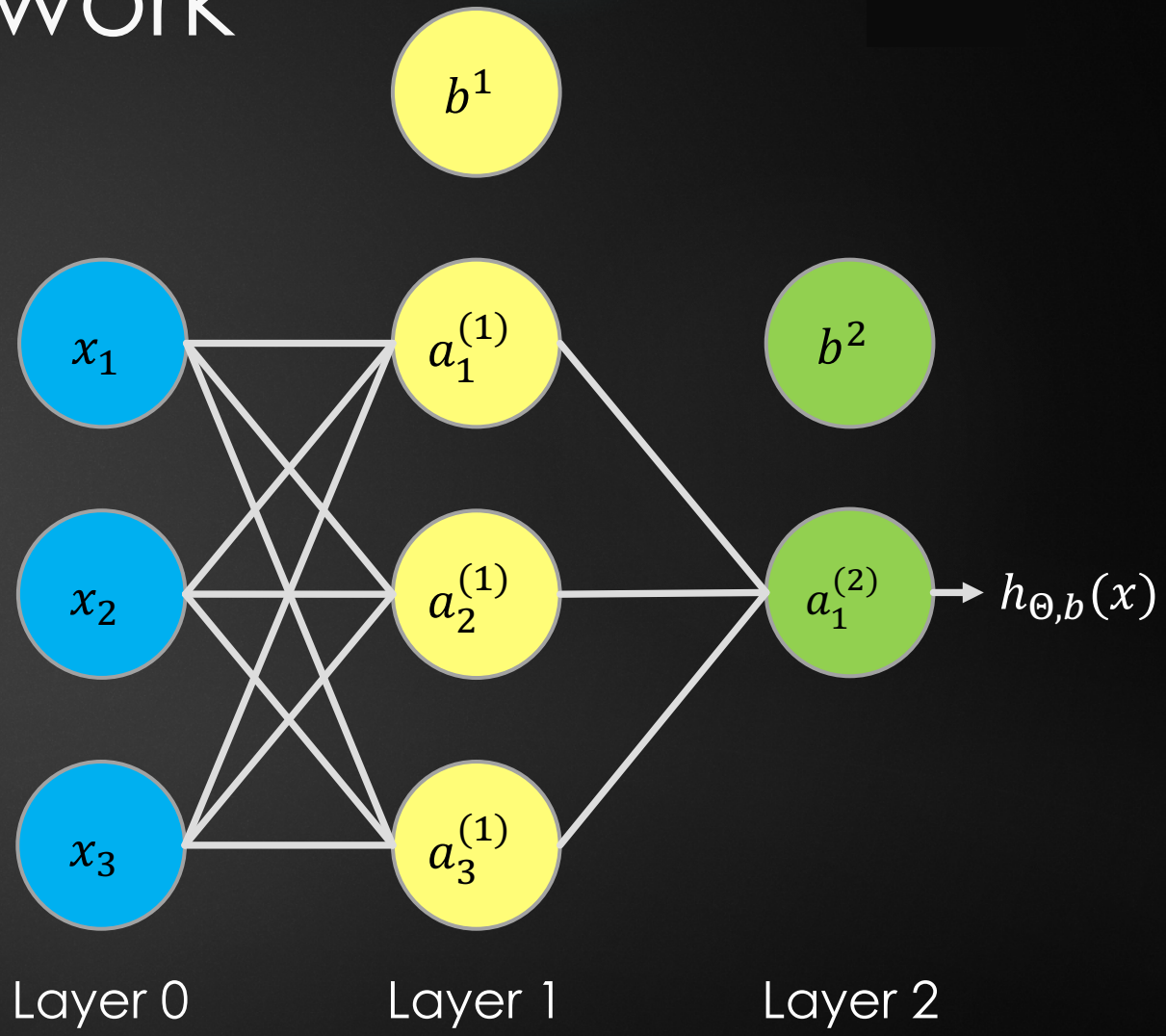
$$\Theta^{(0)} = \begin{pmatrix} \theta_{11}^{(0)} & \theta_{12}^{(0)} & \theta_{13}^{(0)} \\ \theta_{21}^{(0)} & \theta_{22}^{(0)} & \theta_{23}^{(0)} \\ \theta_{31}^{(0)} & \theta_{32}^{(0)} & \theta_{33}^{(0)} \end{pmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix}$$

$$a^{(1)} = \begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{bmatrix} = (\Theta^{(0)})^T \cdot x + b^{(1)}$$

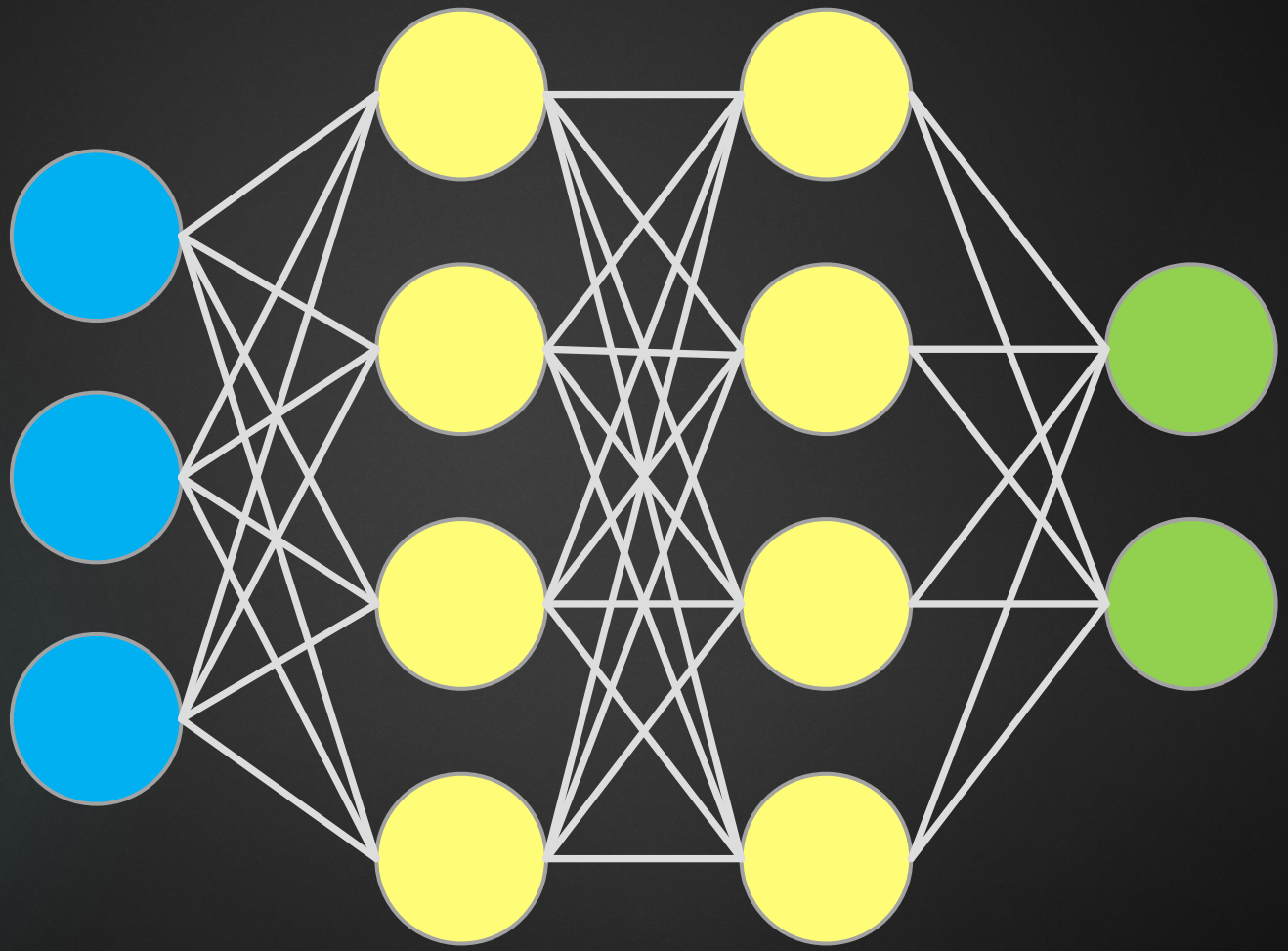
$$a^{(j)} = (\Theta^{(j-1)})^T \cdot a^{(j-1)} + b^{(j)}$$

Building a neural network

- In this example the computation of $h_{\Theta,b}(x)$ is known as **forward propagation**
- It starts with the multiplication of the input units with the weights of the first layer.
- It propagates to the hidden layer computing the activation function of the second layer.
- The propagation follows with the activation of the output layer



Artificial Neural Networks



Images sources

- Deep learning timeline: https://noeliagorod.com/2019/06/26/deep-learning-timeline/
- AI diagram: http://danieljhand.com/the-relationship-between-artificial-intelligence-ai-machine-learning-ml-and-deep-learning-dl.html
- Environmental input: https://designsforlearning.nu/articles/10.16993/dfl.118
- Neuron sketch: https://evaprofebio.jimdofree.com/biology-and-geology-3rd-eso/ud-5-sistema-nervioso-y-endocrino/2-2-impulso-nervioso-y-sinapsis/
- Reflex act: https://www.opentextbooks.org.hk/ditatopic/26420
- Sigmoid function: https://en.wikipedia.org/wiki/Sigmoid_function#/media/File:Logistic-curve.svg
- AND-OR-XOR: https://pyimagesearch.com/2021/05/06/implementing-the-perceptron-neural-network-with-python/
- XOR decision boundary: https://ynebula.tistory.com/22