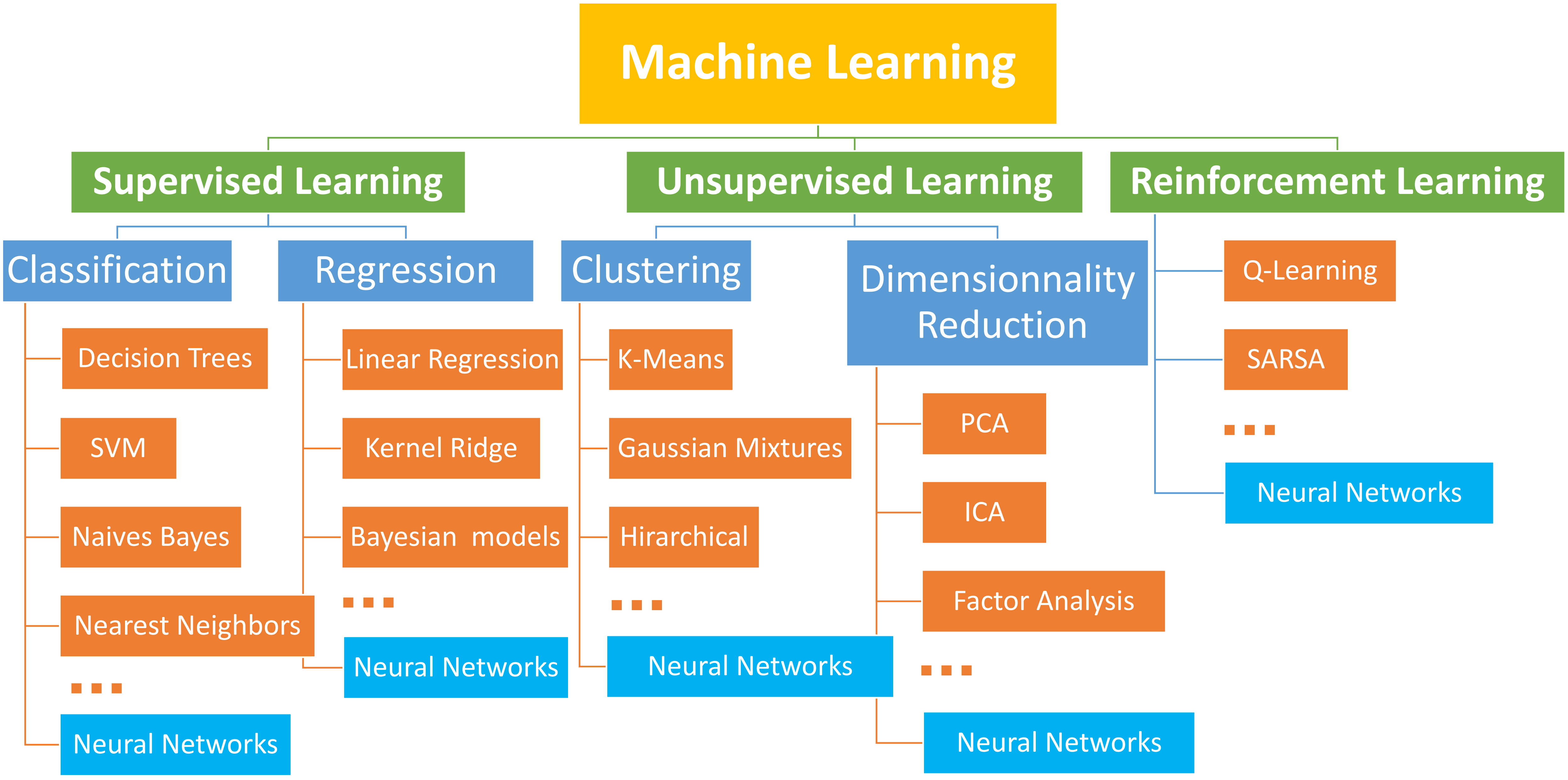
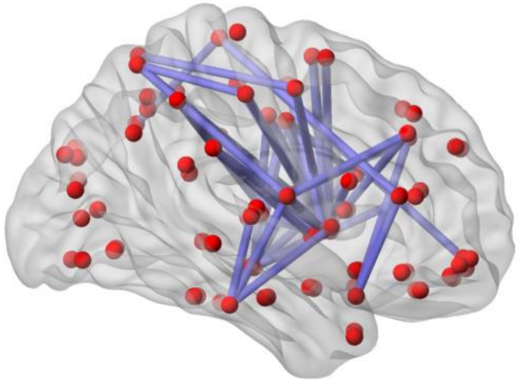
Artificial Intelligence

Artificial Neural Networks

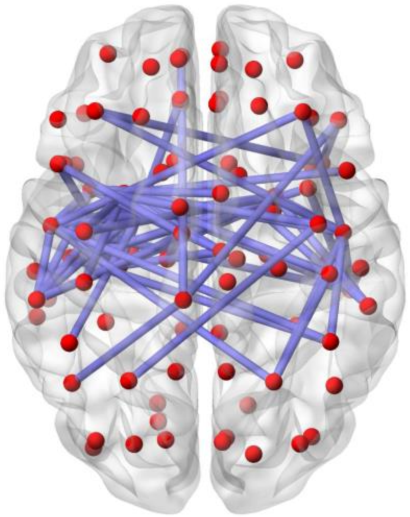
JUNIA ISEN / M1 / 2024-2025
Nacim Ihaddadene


Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Classification
Regression
Clustering
Dimensionnality Reduction
Q-Learning
SARSA
...
Neural Networks
Decision Trees
SVM
Naives Bayes
Nearest Neighbors
...
Neural Networks
Linear Regression
Kernel Ridge
Bayesian models
...
Neural Networks
K-Means
Gaussian Mixtures
Hirarchical
...
Neural Networks
PCA
ICA
Factor Analysis
...
Neural Networks

Artificial Neural Networks

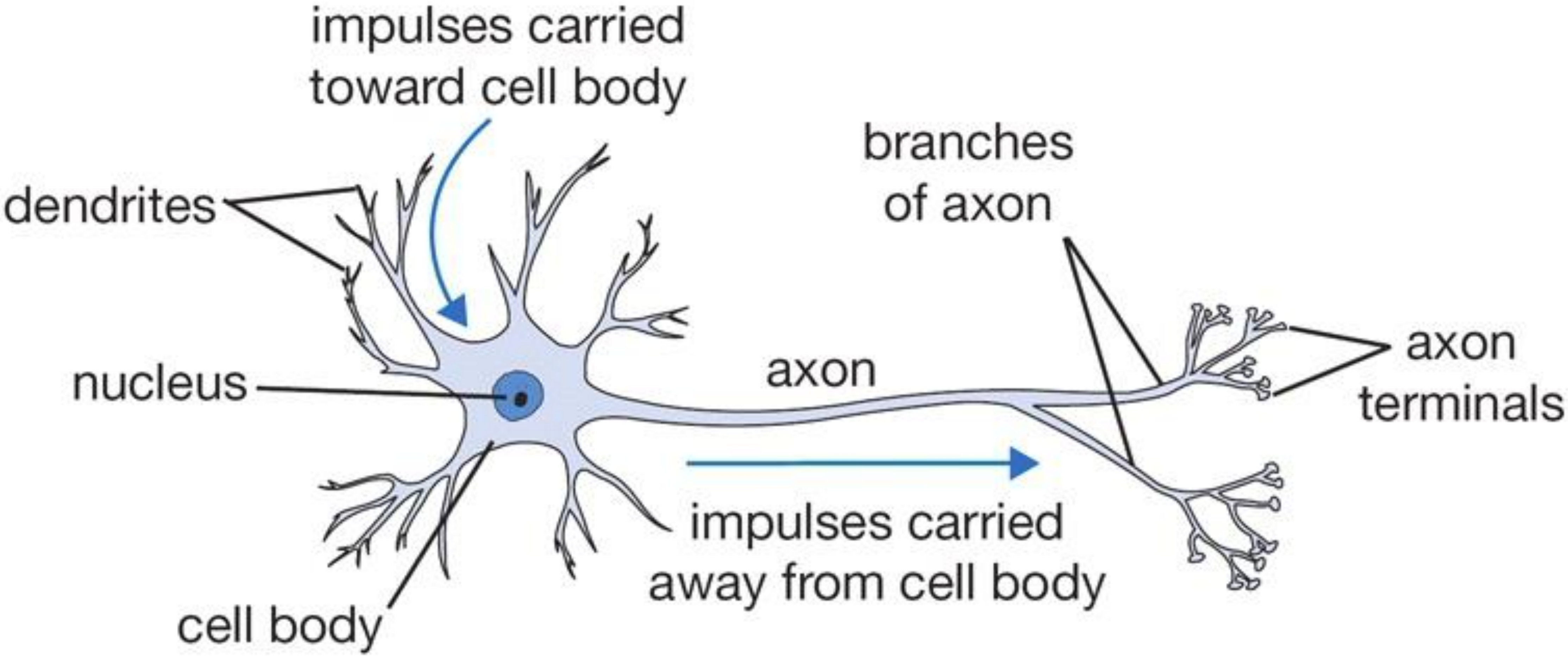


- A model of reasoning based on the human brain.



- Densely interconnected set of nerve cells (neurons).

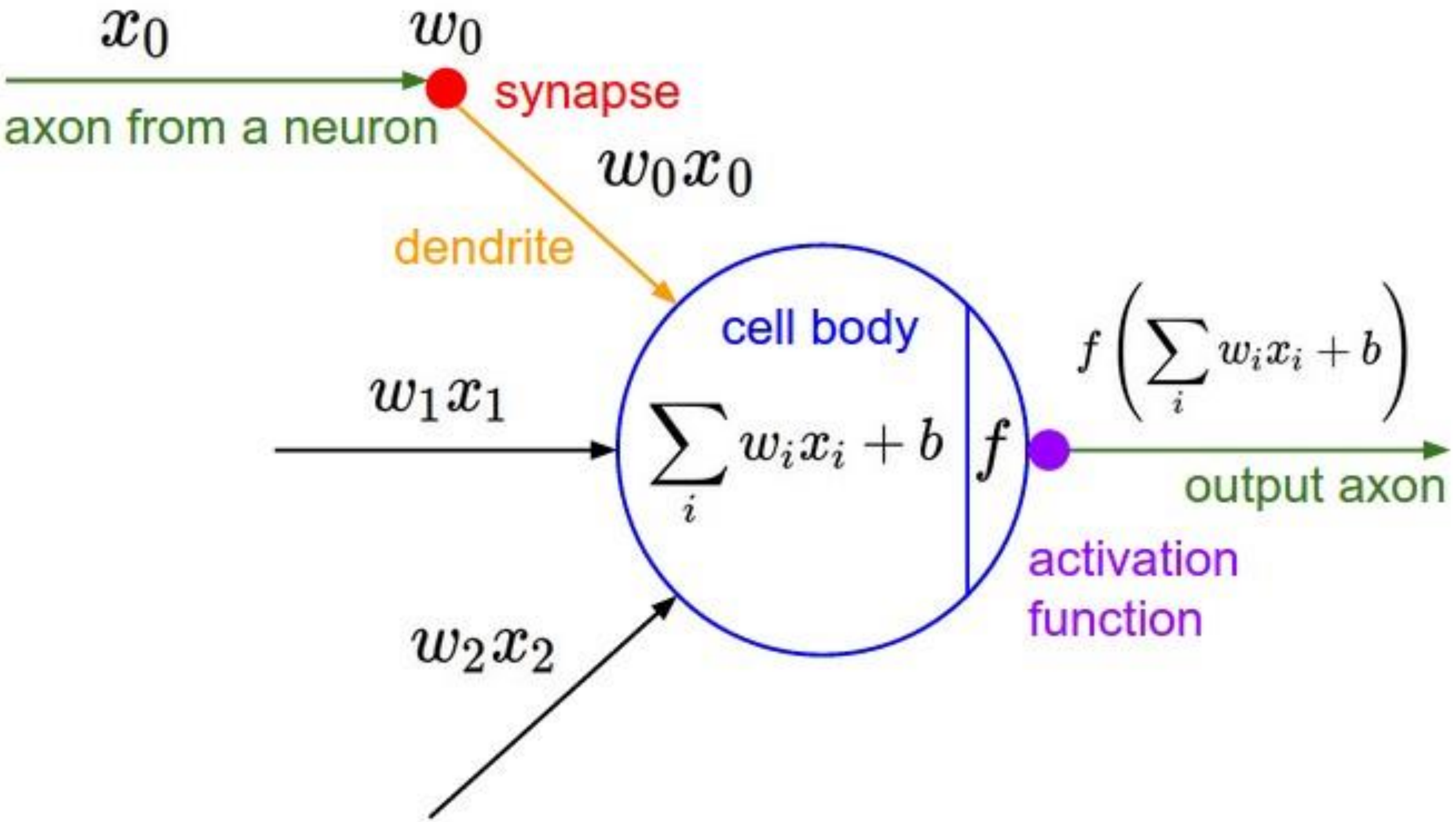
Inspired by biological neuron



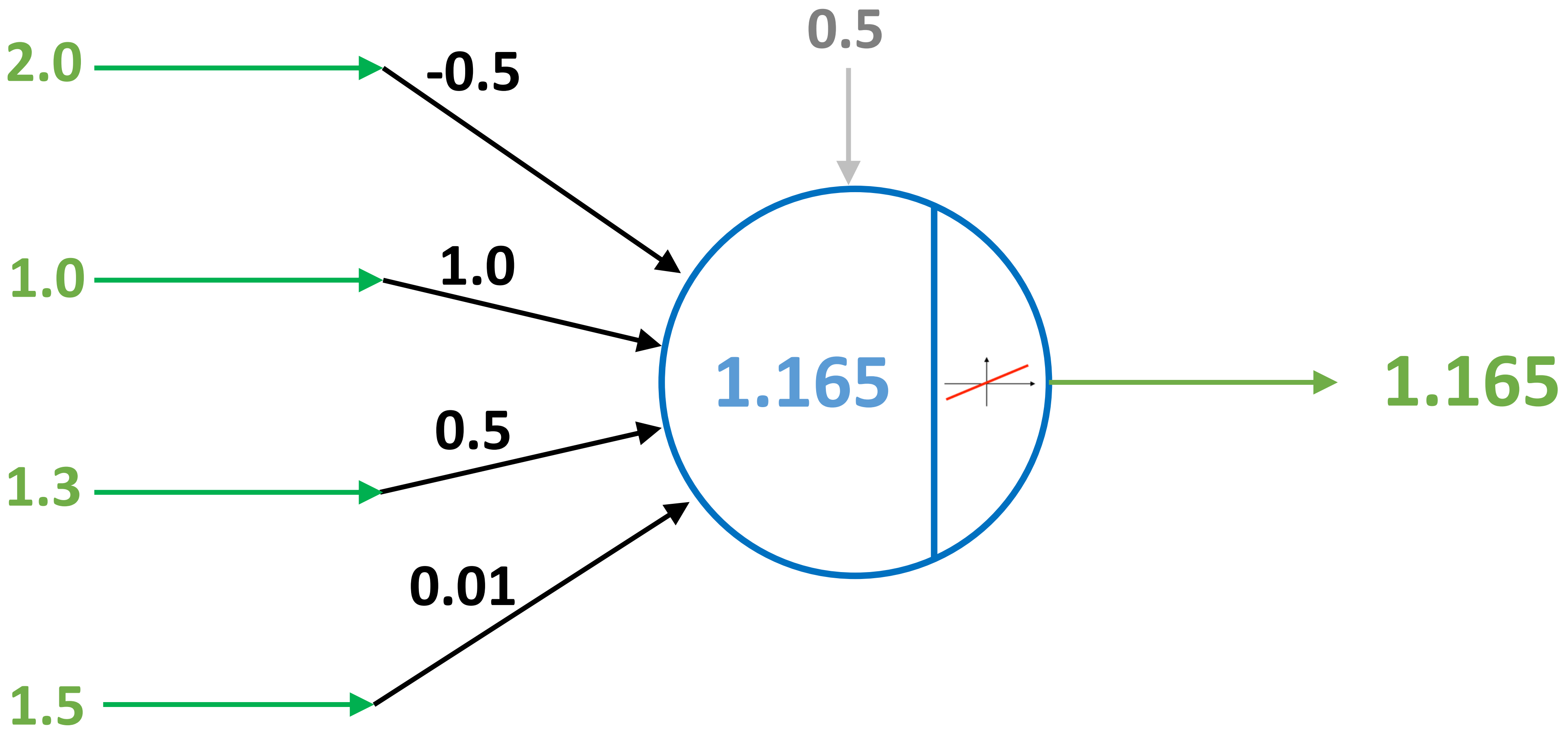
Average Human brain

- 100 billions neurons
- Each neurons is connected to other neurons with 10.000 synapses
- 100 to 1,000 trillion **synaptic connections***
- Signal sending time: 10^{-3} sec
- 70000 thoughts per day

Mathematical model (What is an artificial neuron?)

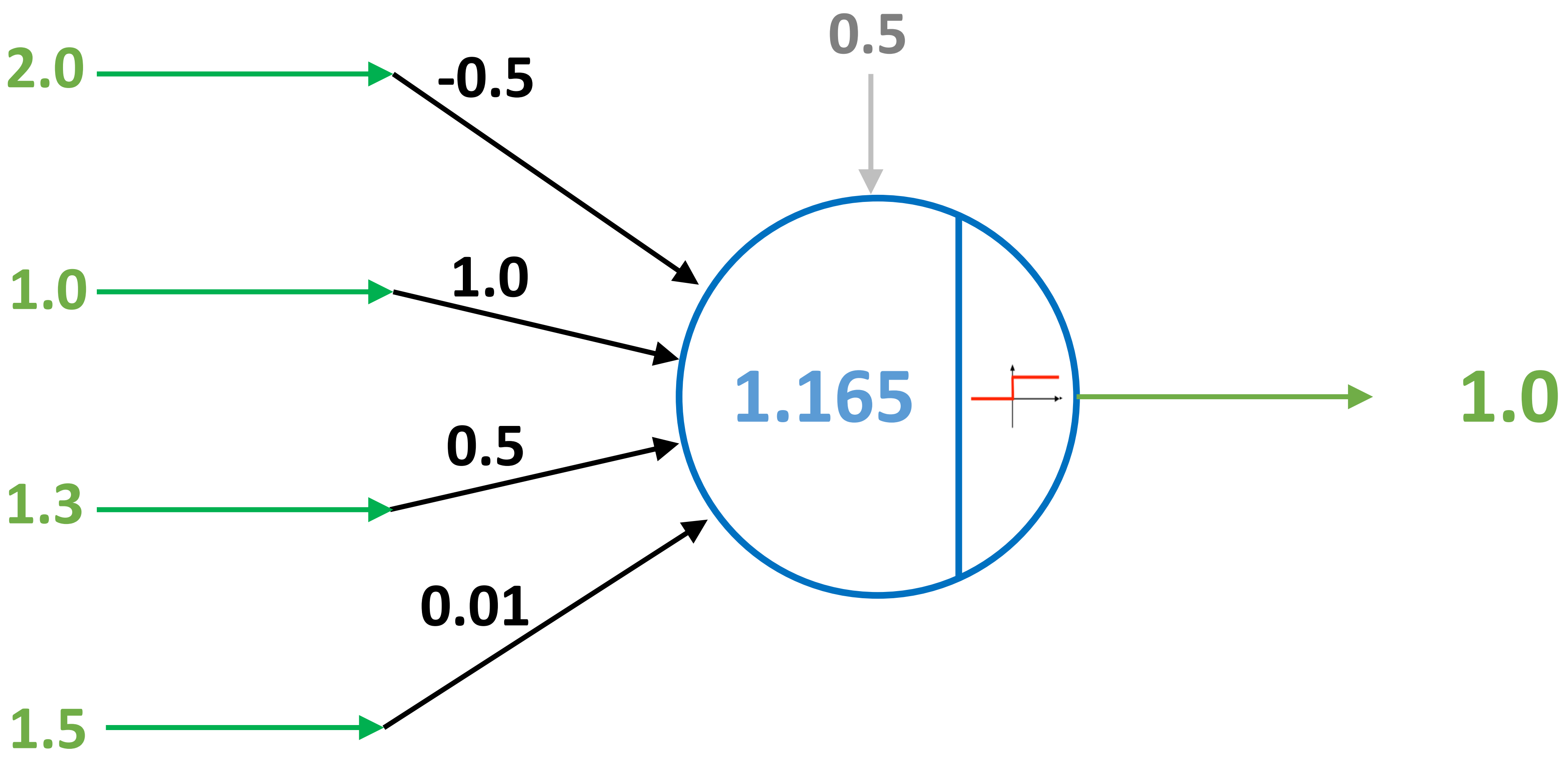


Example



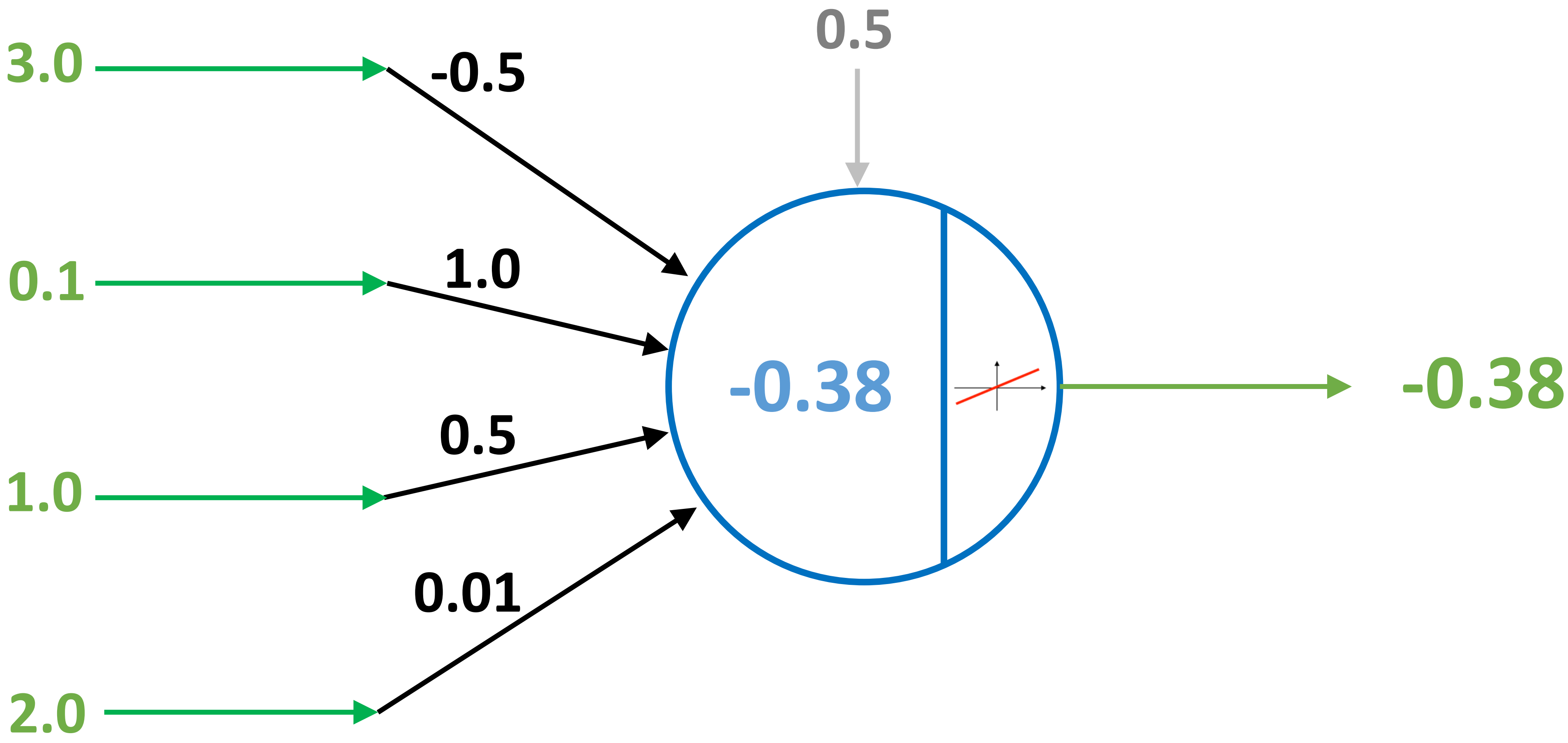
$2.0 \times -0.5 + 1.0 \times 1.0 + 1.3 \times 0.5 + 1.5 \times 0.01 + 0.5 = 1.165$

Example



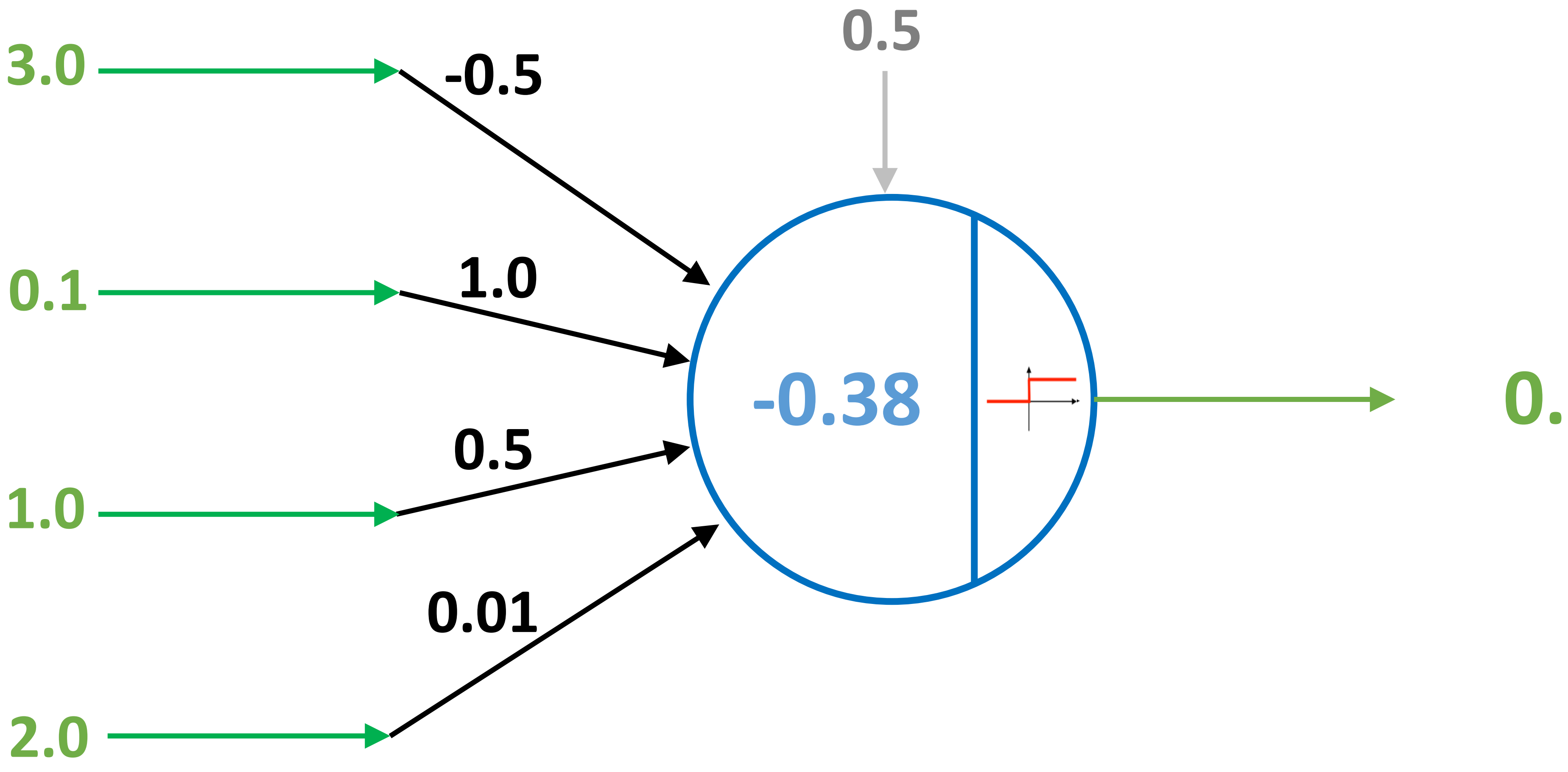
2.0 × -0.5 + 1.0 × 1.0 + 1.3 × 0.5 + 1.5 × 0.01 + 0.5 = 1.165

Example



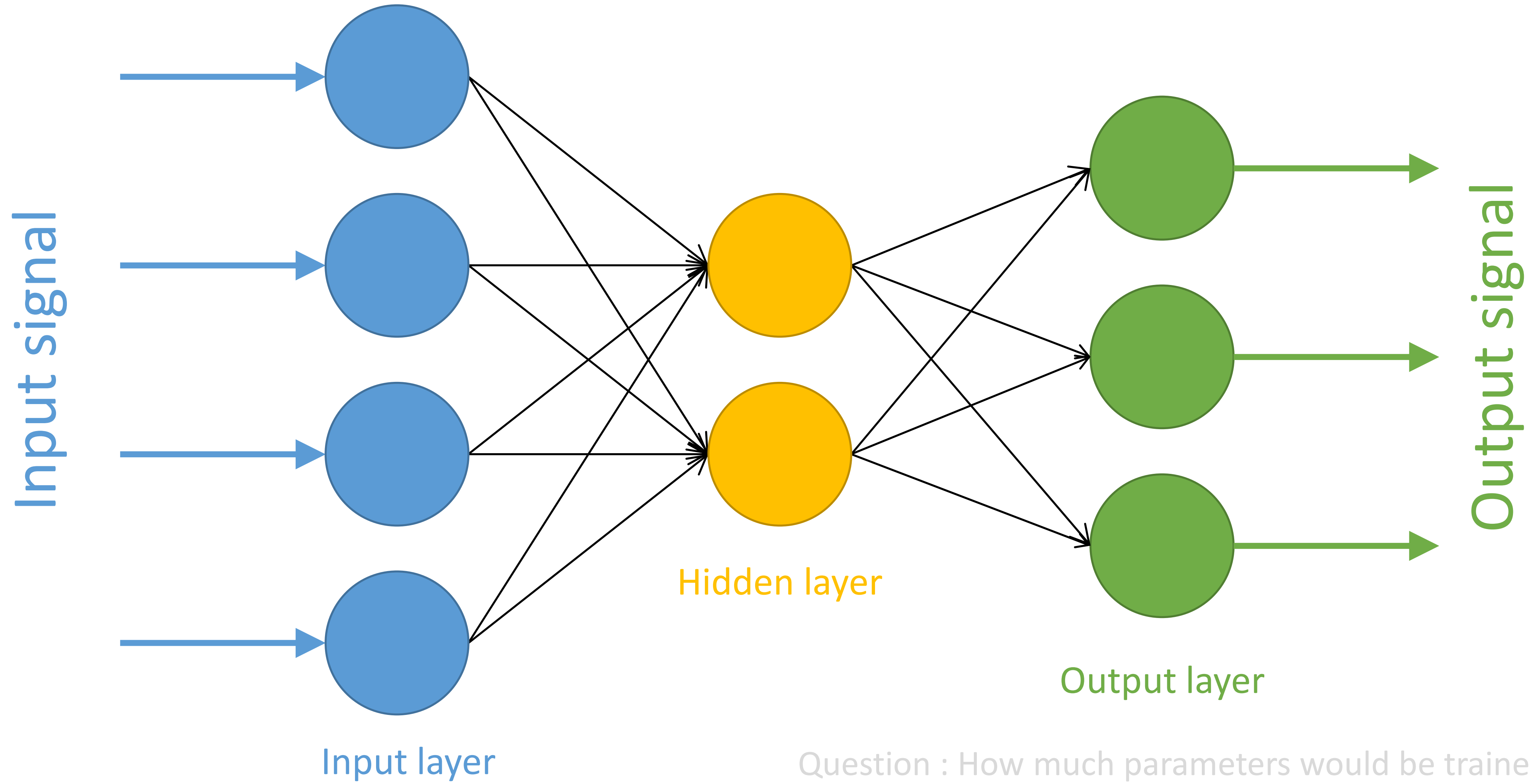
$3.0 \times -0.5 + 0.1 \times 1.0 + 1.0 \times 0.5 + 2.0 \times 0.01 + 0.5 = -0.38$

Example

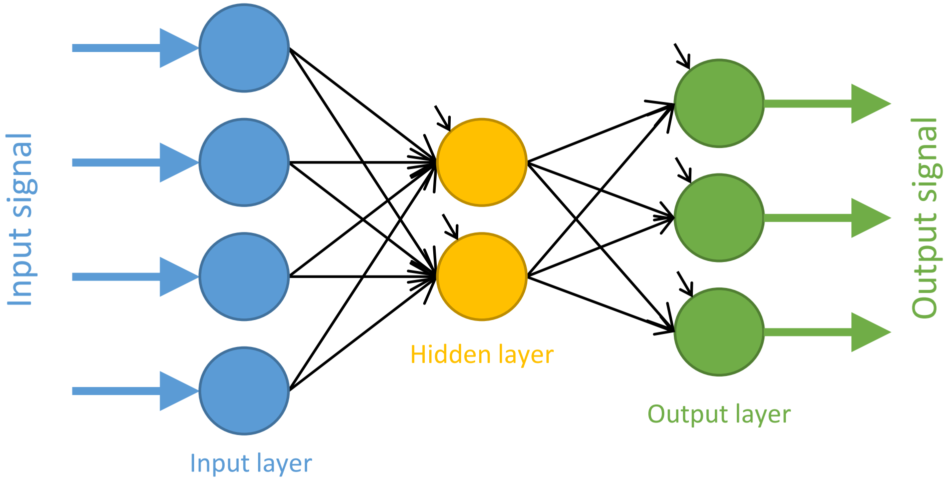


$3.0 \times -0.5 + 0.1 \times 1.0 + 1.0 \times 0.5 + 2.0 \times 0.01 + 0.5 = -0.38$

Architecture of neural networks



Question : How much parameters would be trained ?

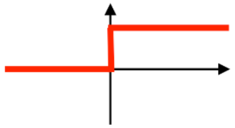
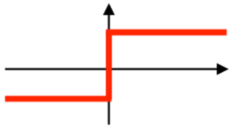
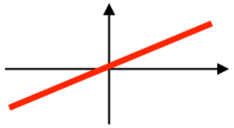
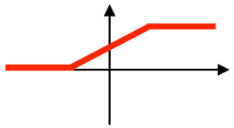
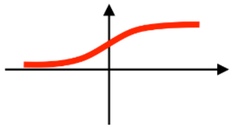
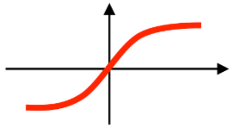
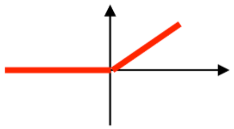
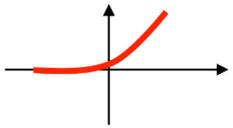


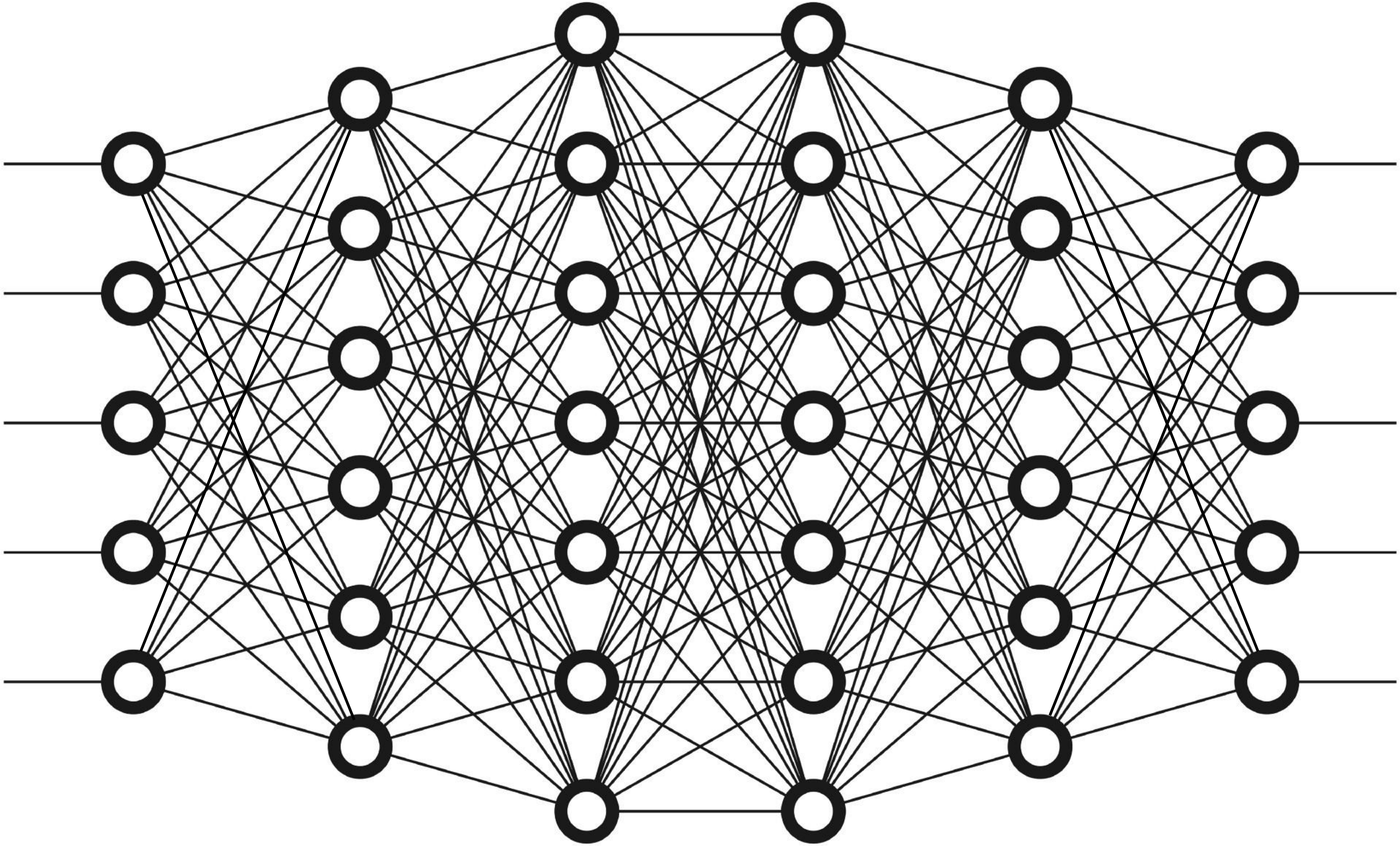

```
In [1]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense
        from tensorflow.keras import Input

        model = Sequential()
        model.add(Input(shape=(4,)))
        model.add(Dense(2, activation='relu'))
        model.add(Dense(3, activation='sigmoid'))
```

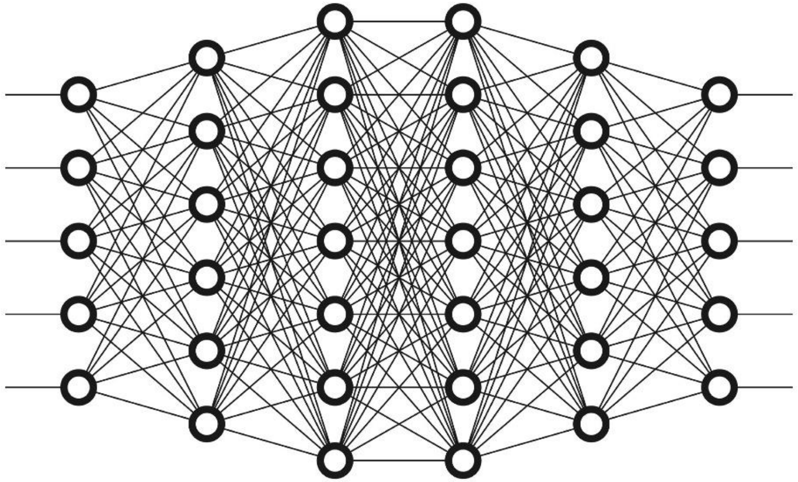
```
In [2]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 10

 dense_1 (Dense)             (None, 3)                 9

=================================================================
Total params: 19
Trainable params: 19
Non-trainable params: 0
_________________________________________________________________
```

Activation functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	



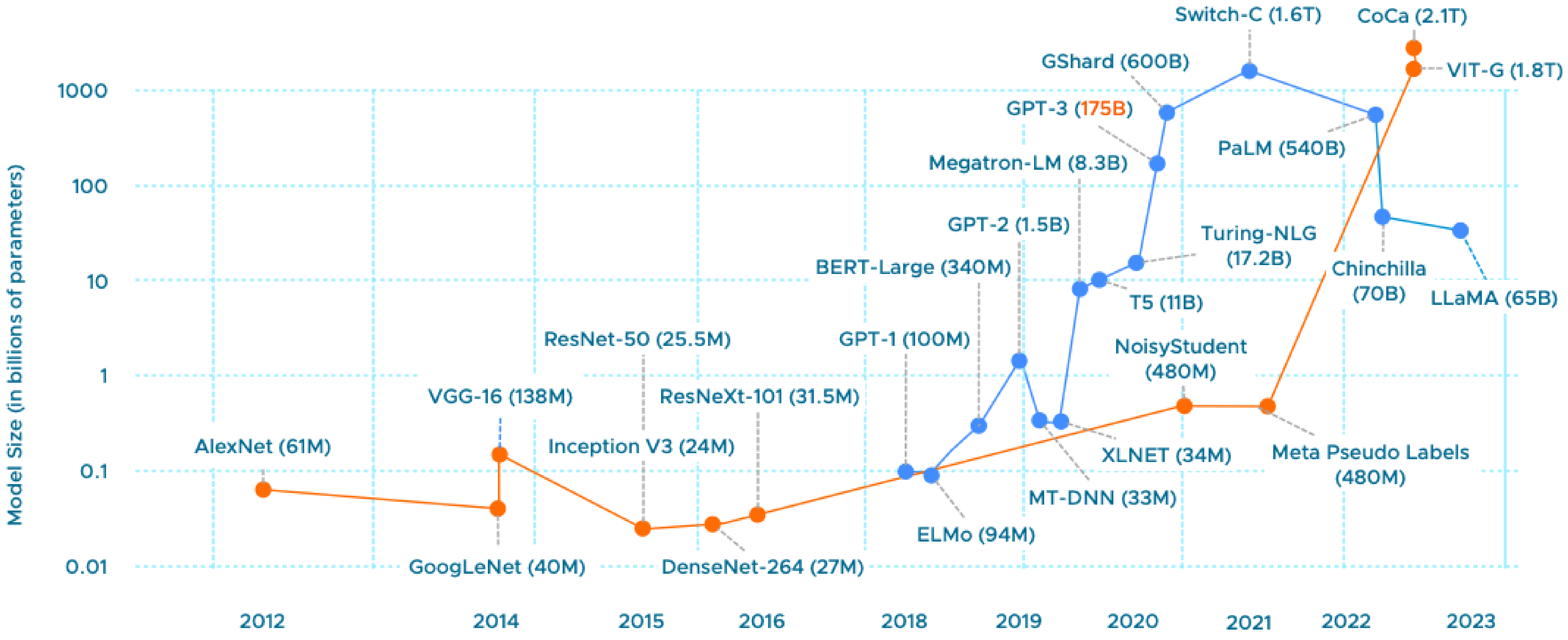


In [3]:
```
model = Sequential()
model.add(Input(shape=(5,)))
model.add(Dense(6))
model.add(Dense(7))
model.add(Dense(7))
model.add(Dense(6))
model.add(Dense(5))
```

In [4]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 6)                 36

 dense_3 (Dense)             (None, 7)                 49

 dense_4 (Dense)             (None, 7)                 56

 dense_5 (Dense)             (None, 6)                 48

 dense_6 (Dense)             (None, 5)                 35

=================================================================
Total params: 224
Trainable params: 224
Non-trainable params: 0
_________________________________________________________________
```

Example : Some popular Pre-trained models



Training of neural networks

- The network topology **is given:**
  - The user defines the number of layers,
  - and the number of neurons in each layer.
  - The user specifies an activation function for each layer.
    Same activation function is used at each hidden neuron of the same layer.

- **Learning (or training)** is the process of modifying (or calibrating) the weights of the neurons in order to produce a network that achieves the objective function
  - The network starts with randomly assigned weights
  - Multiple trainings starting from various randomly initalized weights might help

Training of neural networks

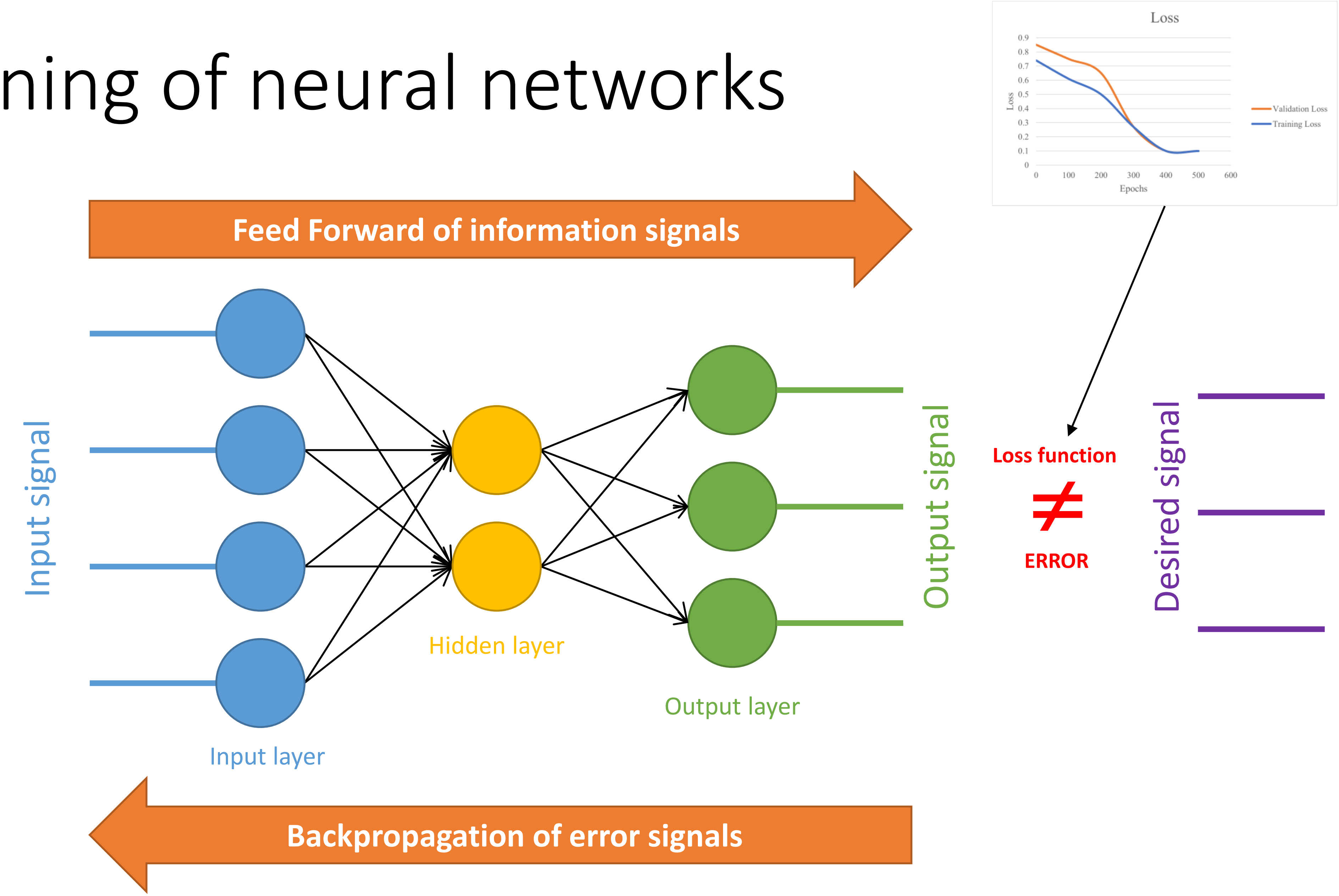
Train the neural network = Estimate the parameter values (Weights)

1. Forward propagation
   - An input vector propagates through the network

   → **Measure the current error**
2. Weight update (**backpropagation**)
   - The weights of the network will be changed/updated/adjusted in order to decrease the difference between the predicted and the real values

   → **Reduce that error**
3. **Repeat** until predicted output values and expected values agree

Training of neural networks

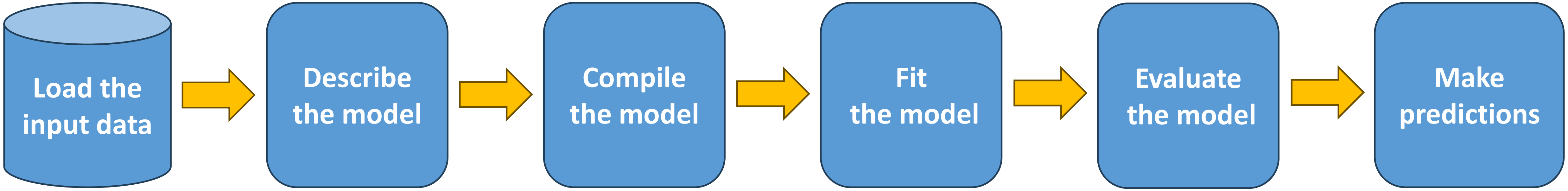


Train, Validation and Test datasets

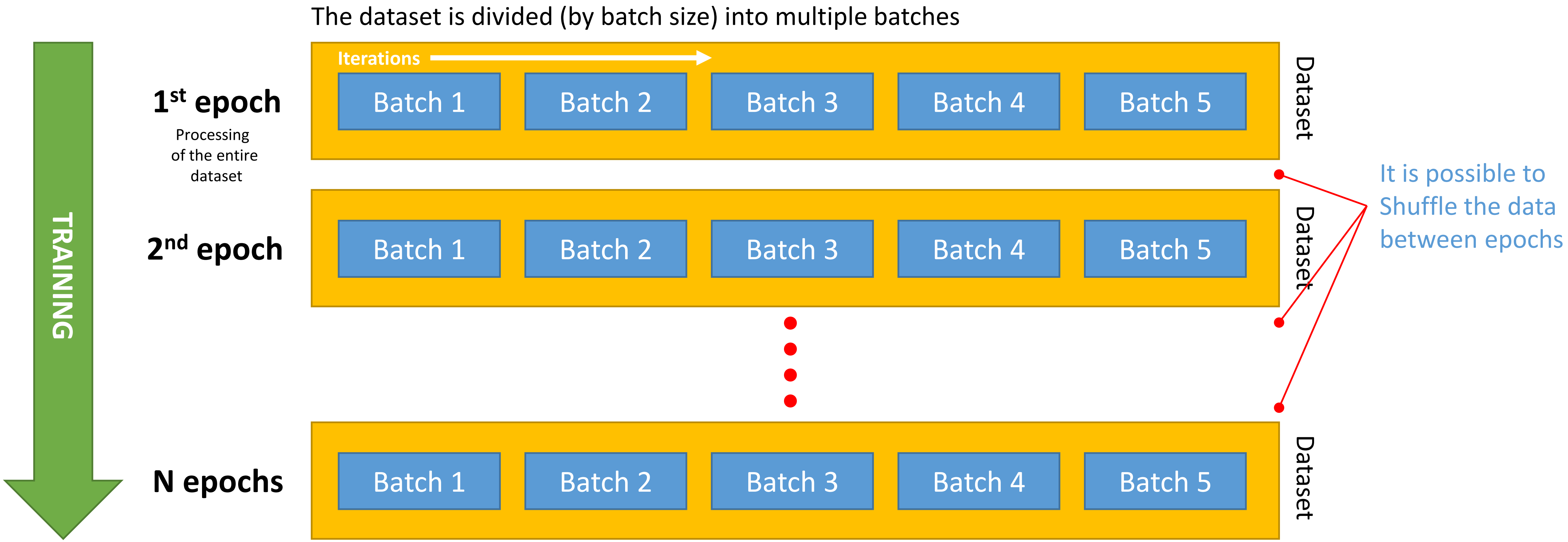
DATA

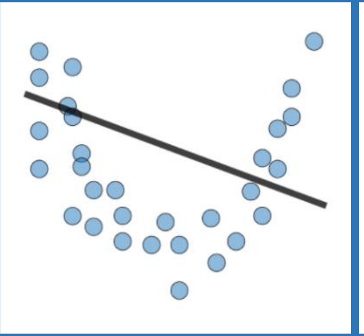
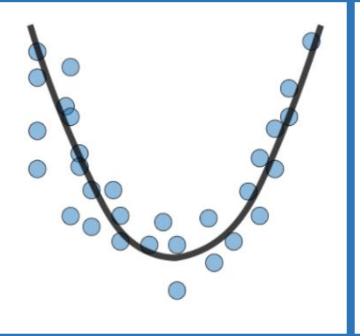
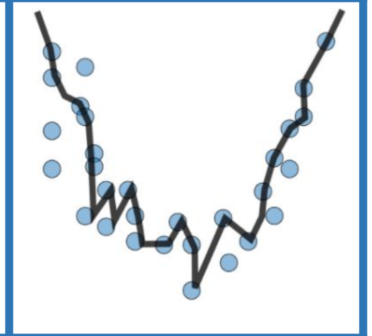
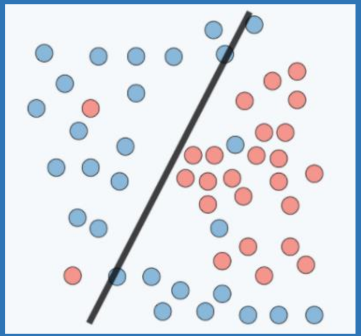
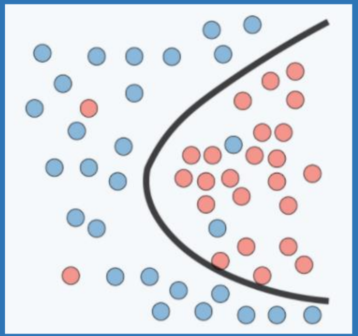
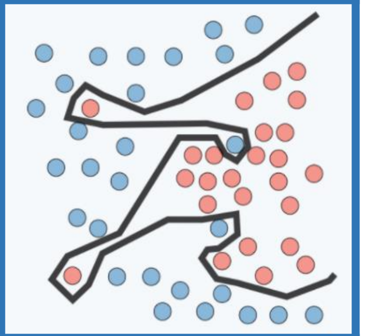
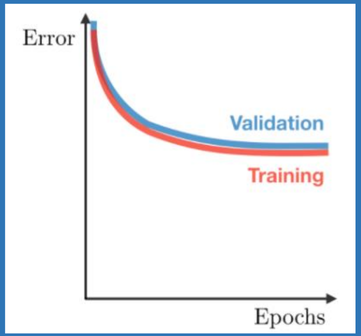
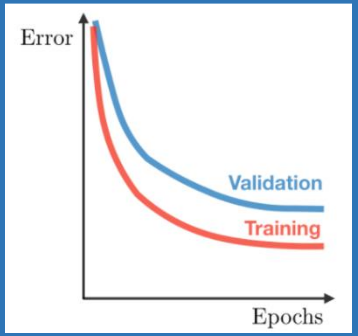
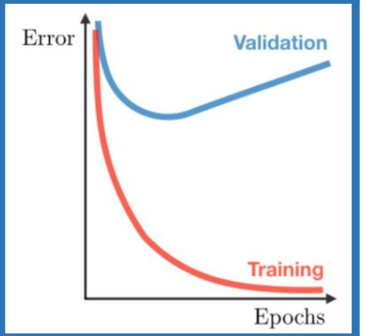
Training set	Validation set	Testing set
• Model is trained • Usually, 60% of the dataset	• Model is assessed • Avoid overfitting • Usually, 20% of the dataset	• Determine model accuracy • Unseen data • Usually, 20% of the dataset

Steps to create DL model



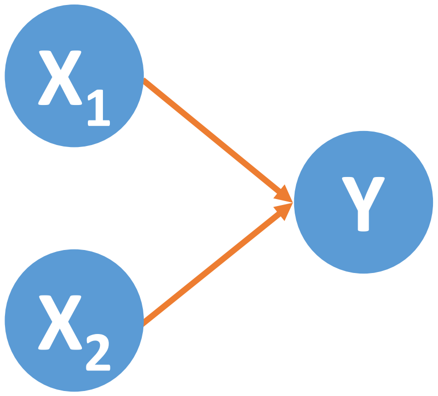
Epochs, Batch size and iterations…

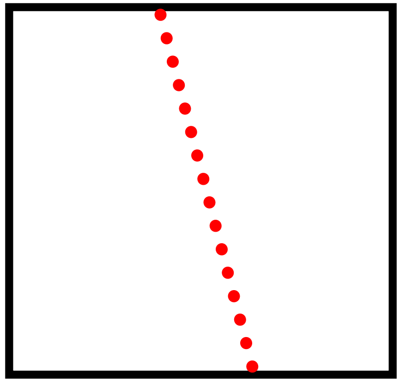


	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	• Complexify model • Add more features • Train longer		• Perform regularization • Get more data

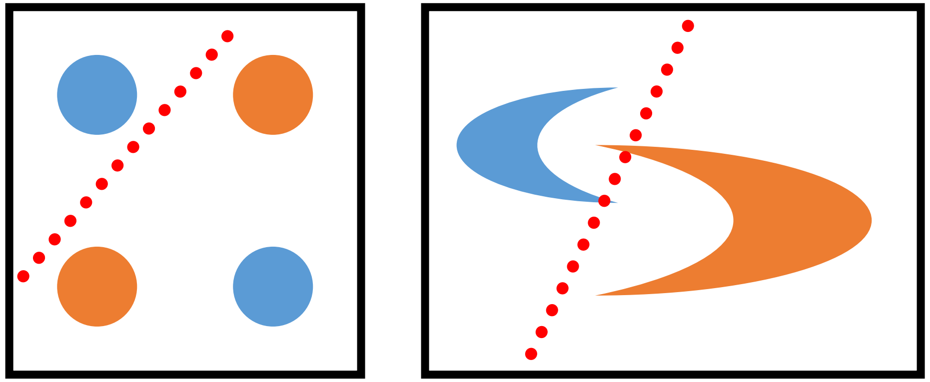
Decision Boundary

0 hidden layers (Linear classifier)





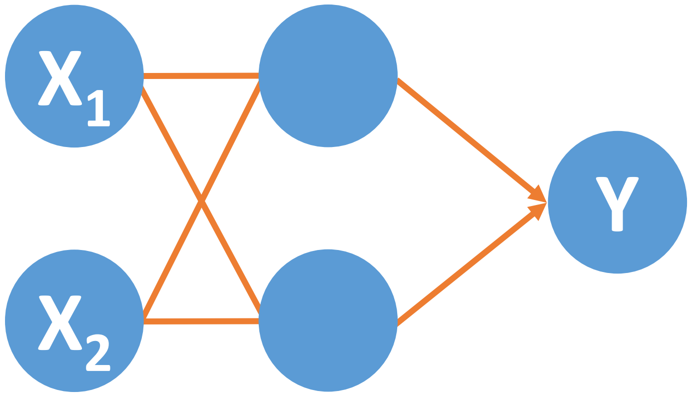
Hyperplanes

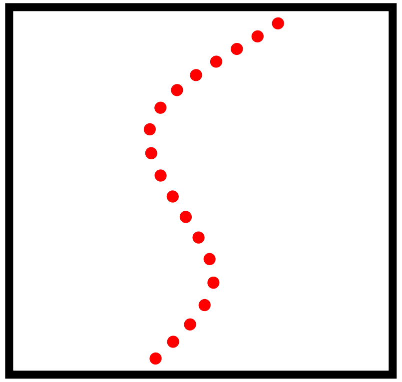


Example from Eric Postma via Jason Eisner

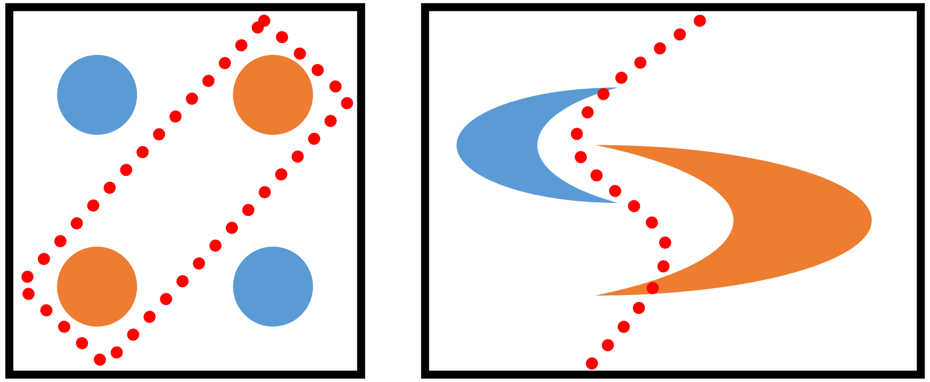
Decision Boundary

1 hidden layer





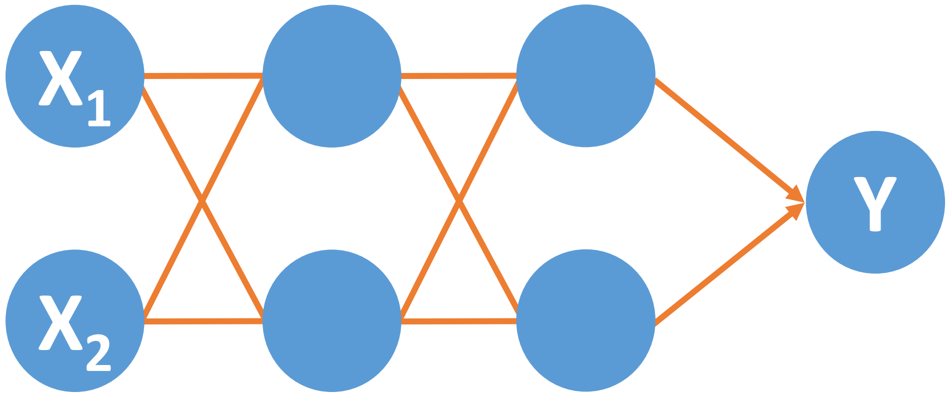
Boundary of convex region (open or closed)

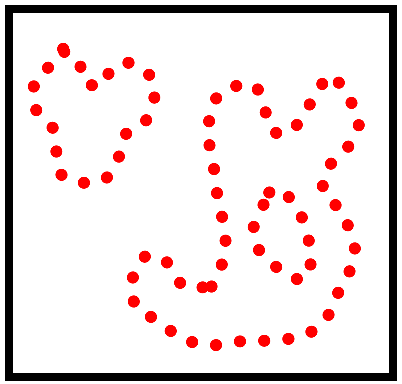


Example from Eric Postma via Jason Eisner

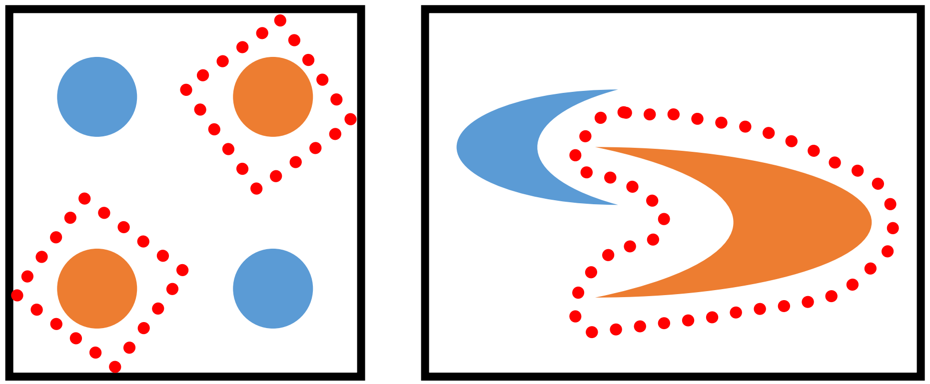
Decision Boundary

2 hidden layers



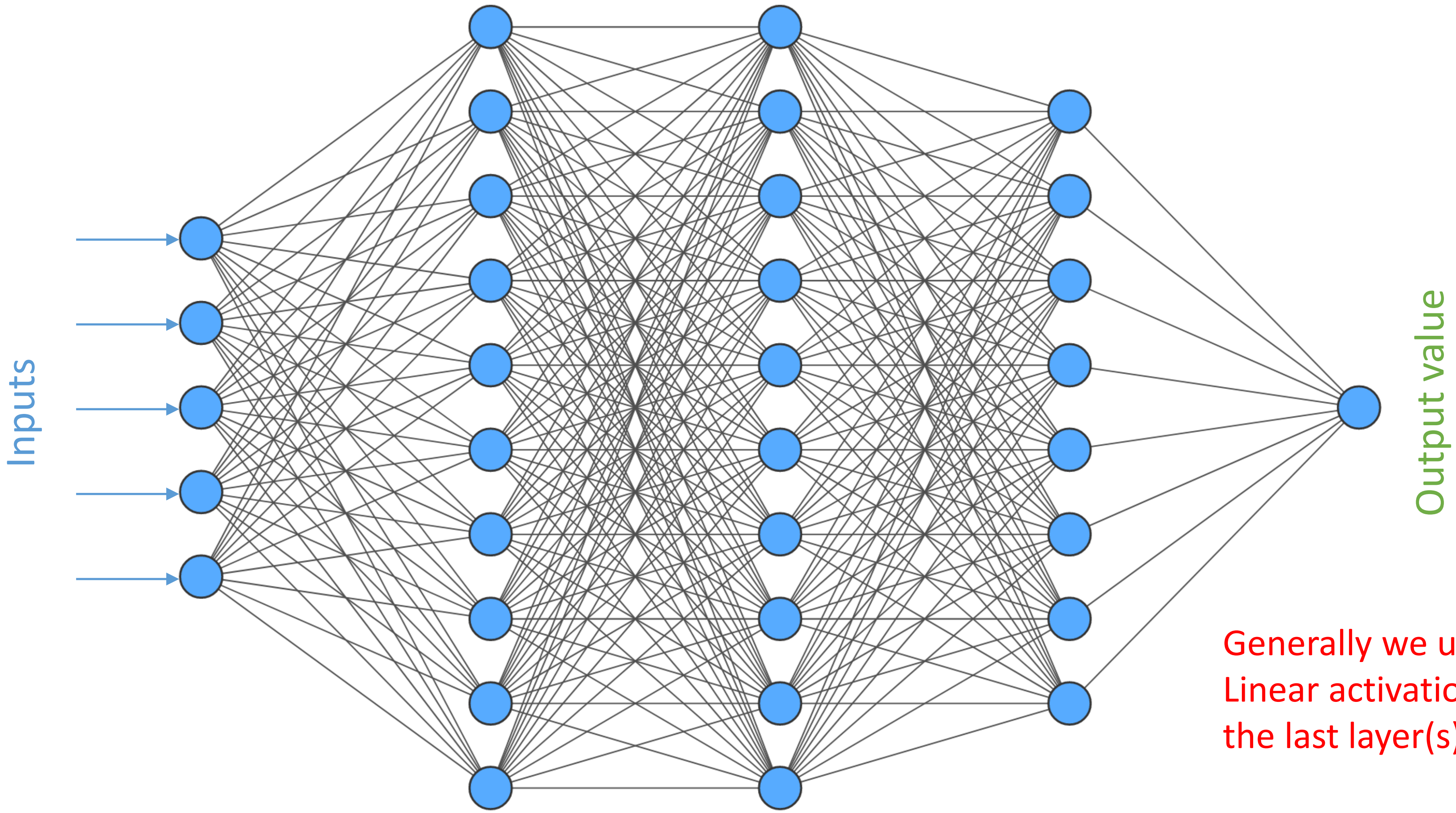


Combinations of convex regions



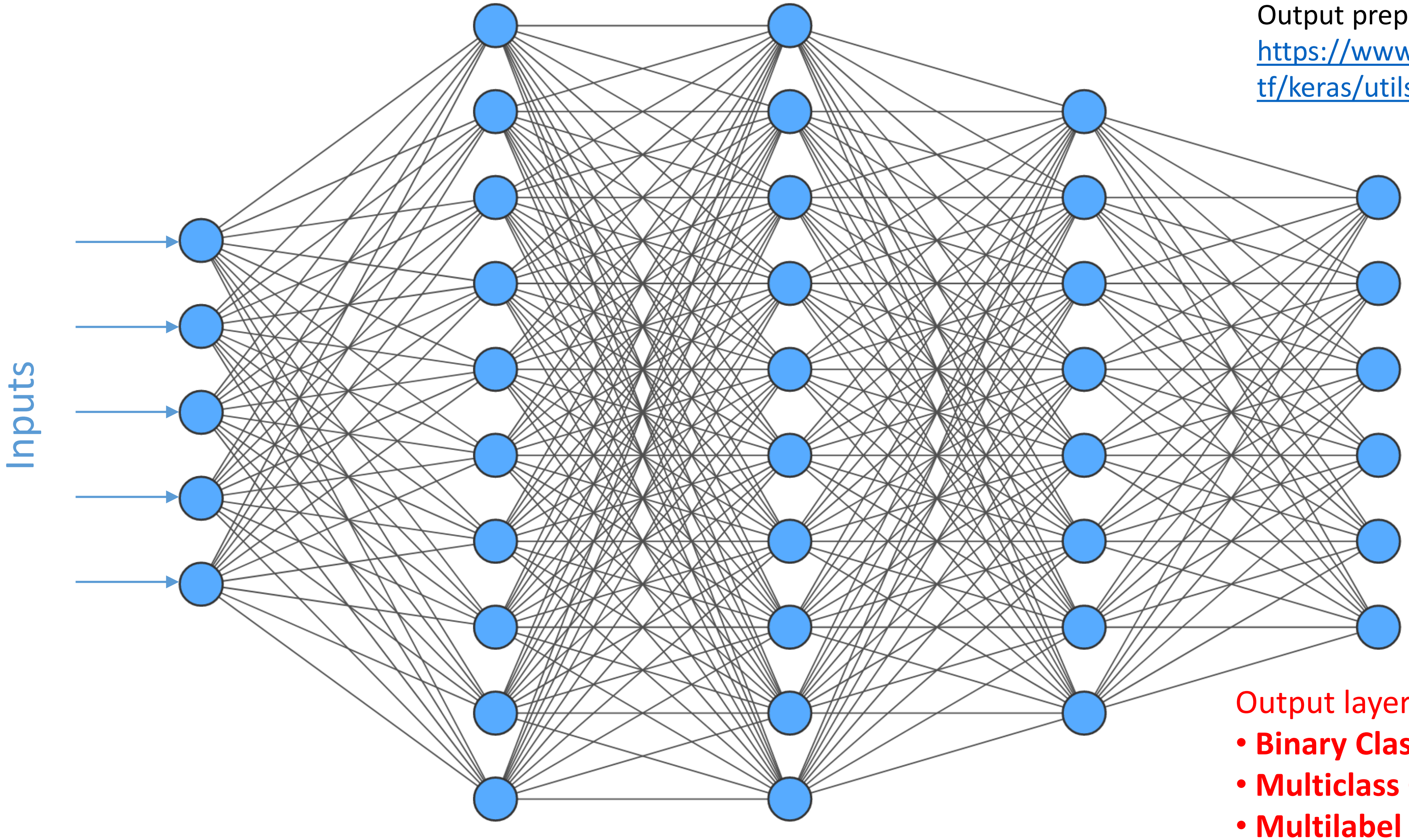
Example from Eric Postma via Jason Eisner

Neural networks for regression



Generally we use
Linear activation for
the last layer(s)

Neural networks for classification



Output preprocessing :
https://www.tensorflow.org/api_docs/python/tf/keras/utils/to_categorical

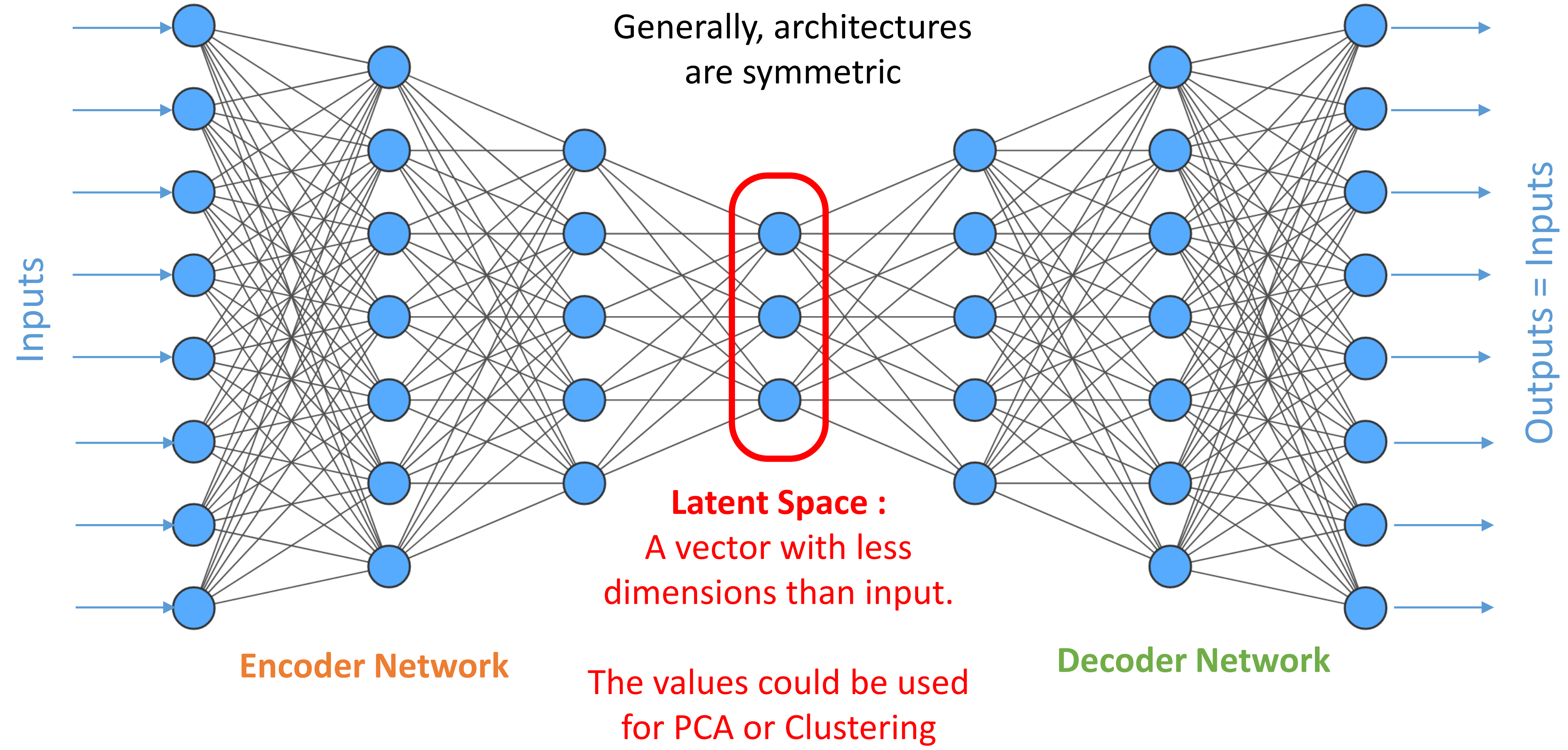
The number of neurons correponds to the number of classes

(For binary classification, we can use a unique layer)

Output layer activation function:

- **Binary Classification**: One node, sigmoid.
- **Multiclass Classification**: softmax activation.
- **Multilabel Classification**: sigmoid activation.

Neural networks for PCA or clustering

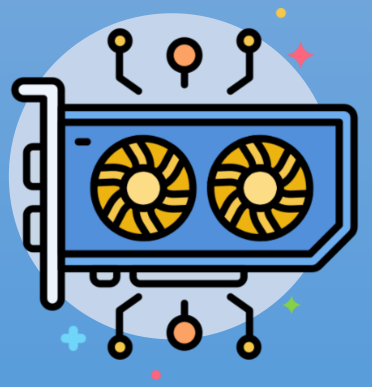


The power of Neural Networks

Deal withs any type of input and output data

Fit with any type of AI/ML problem

(Supervided, Unsupervised, Reinforcement)



Their embarrassingly parallel nature

Suitable for GPU/TPU Acceleration

Demos

Tutorials

Labwork