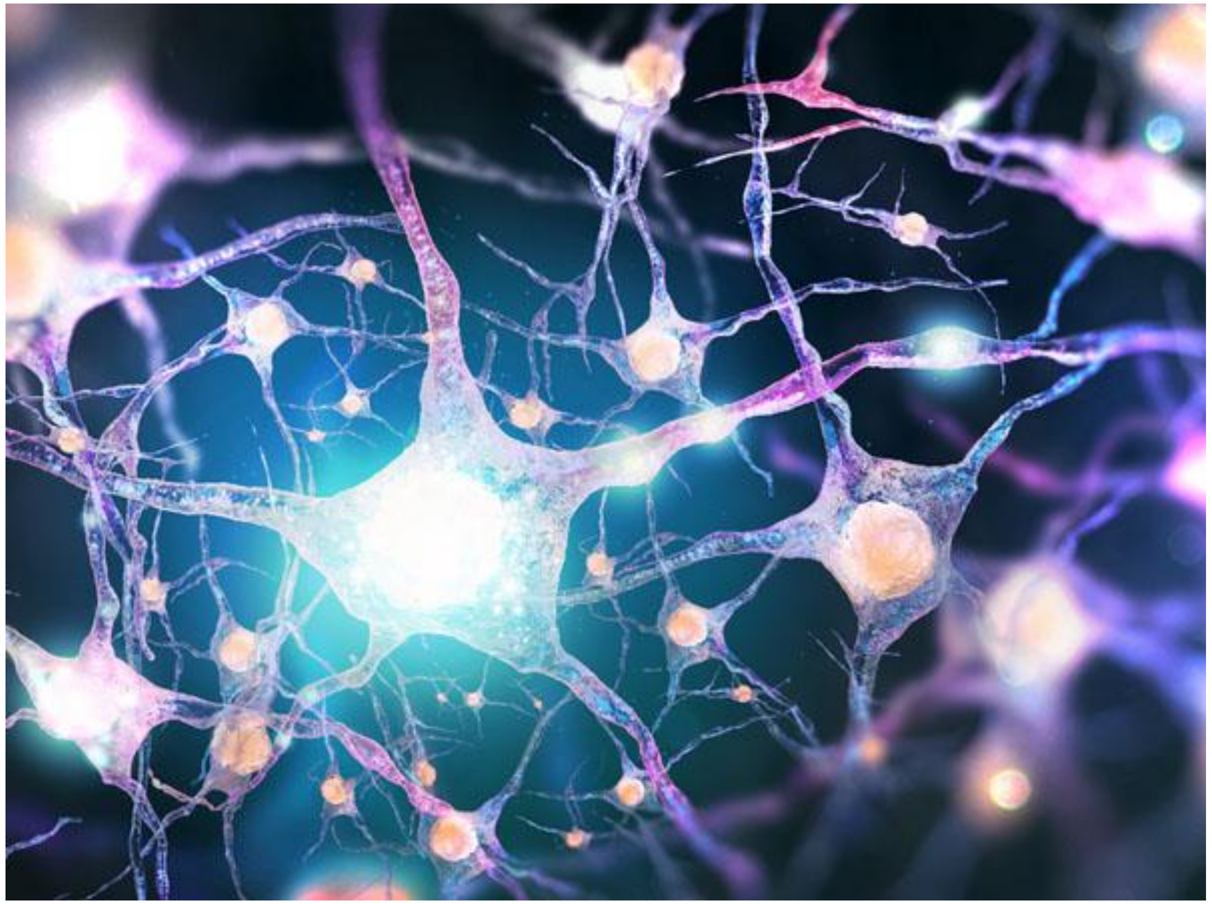
Lecture 3: Introduction to Reinforcement Learning

GEARS Program
Summer 2024

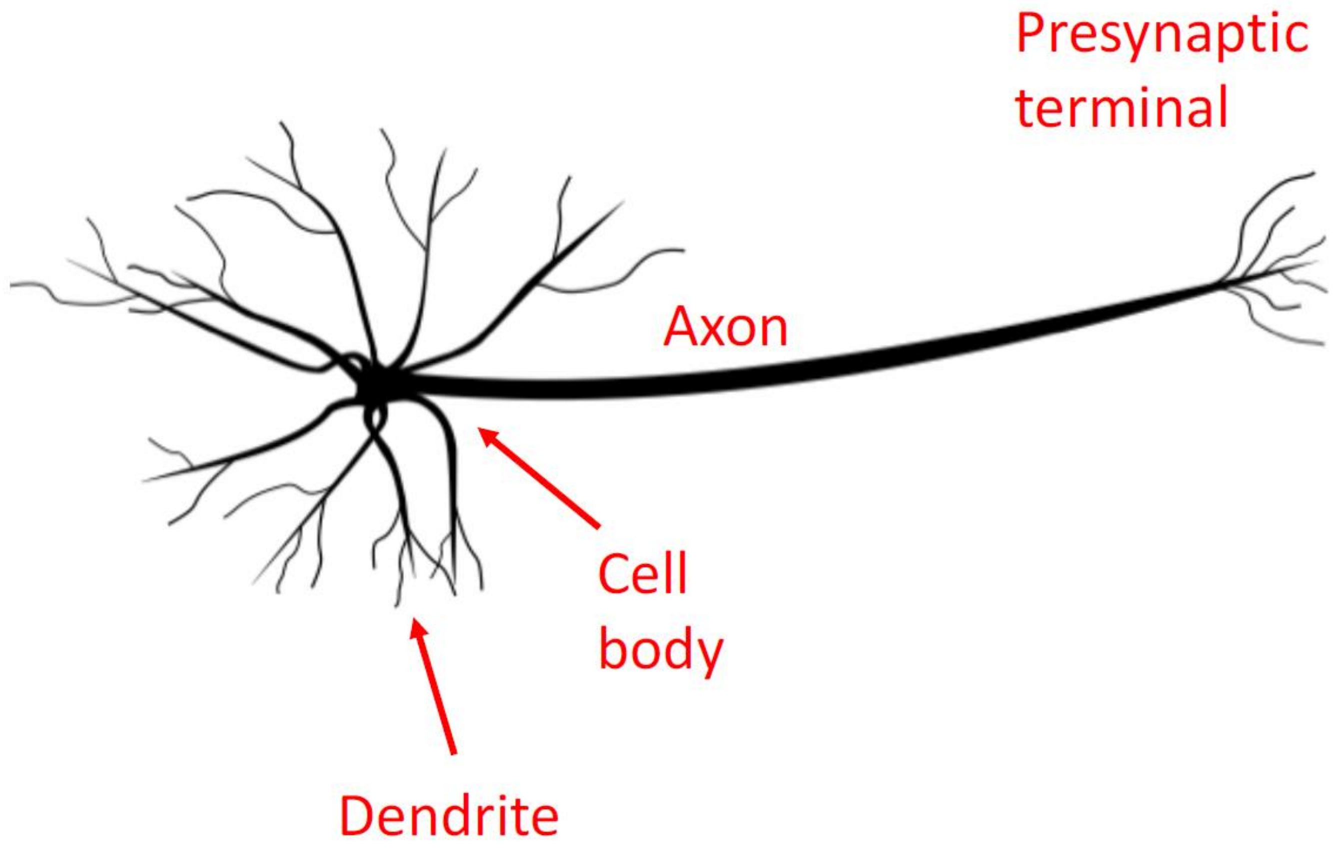
Ivan Lopez-Sanchez, PhD

The Biological Neural Network

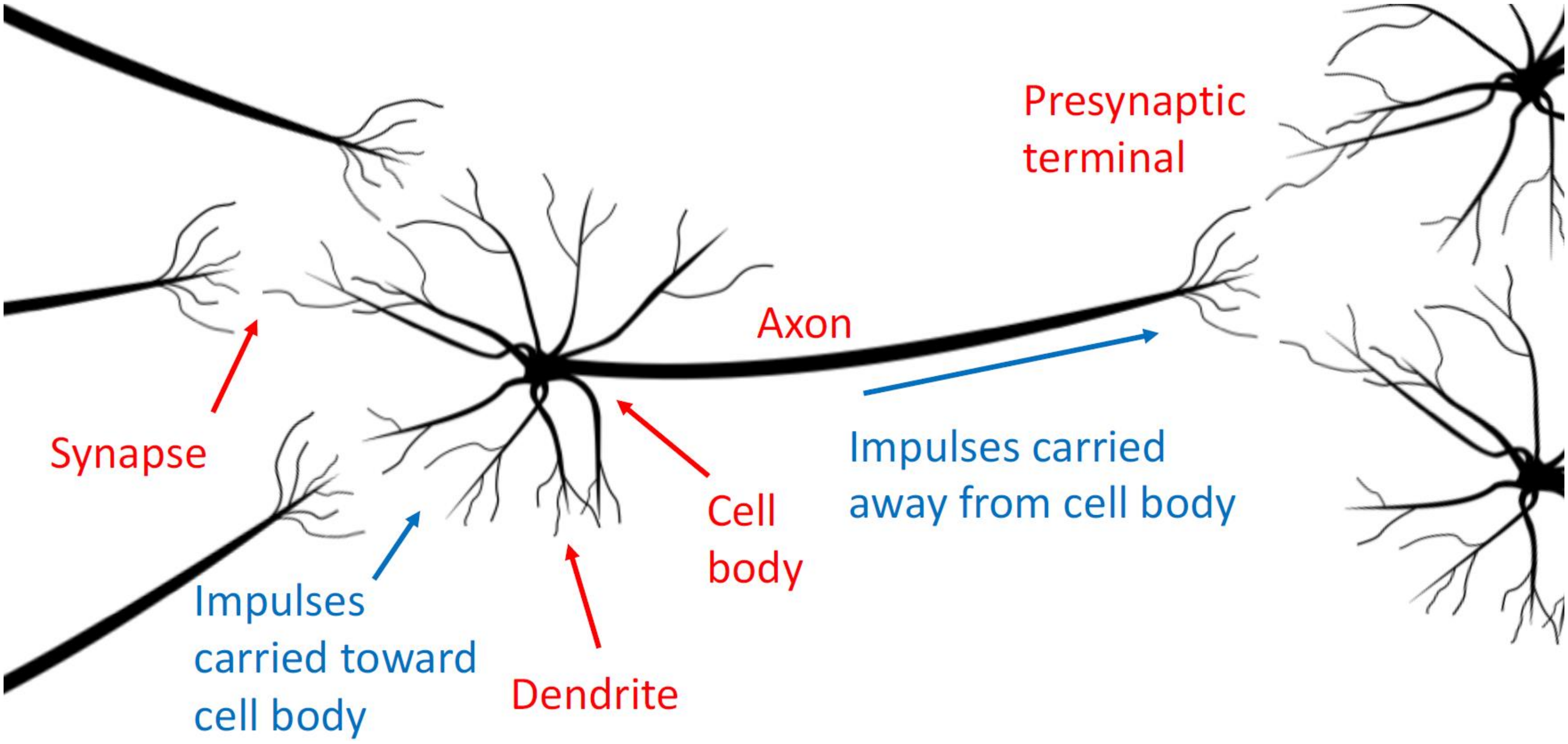
- Our brain is made of huge amounts of interconnected neurons.
- These neurons are our processing units.



The Biological Neural Network

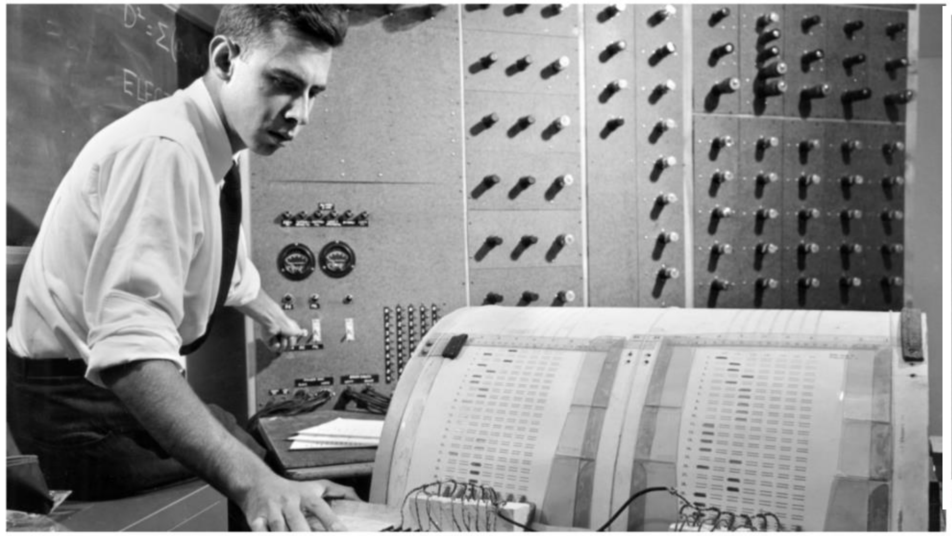


The Biological Neural Network


Presynaptic terminal
Axon
Synapse
Impulses carried away from cell body
Cell body
Impulses carried toward cell body
Dendrite

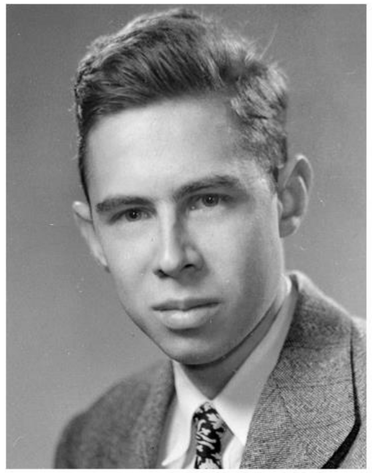
The first Artificial Neural Networks (ANNs)

- History of the ANNs stems from the **1940s**, the decade of the first electronic computer.
- However, the first important step took place in **1958** when **Rosenblatt introduced the first concrete neural model**, the perceptron. Rosenblatt also took part in constructing the first successful neurocomputer, the Mark I Perceptron.

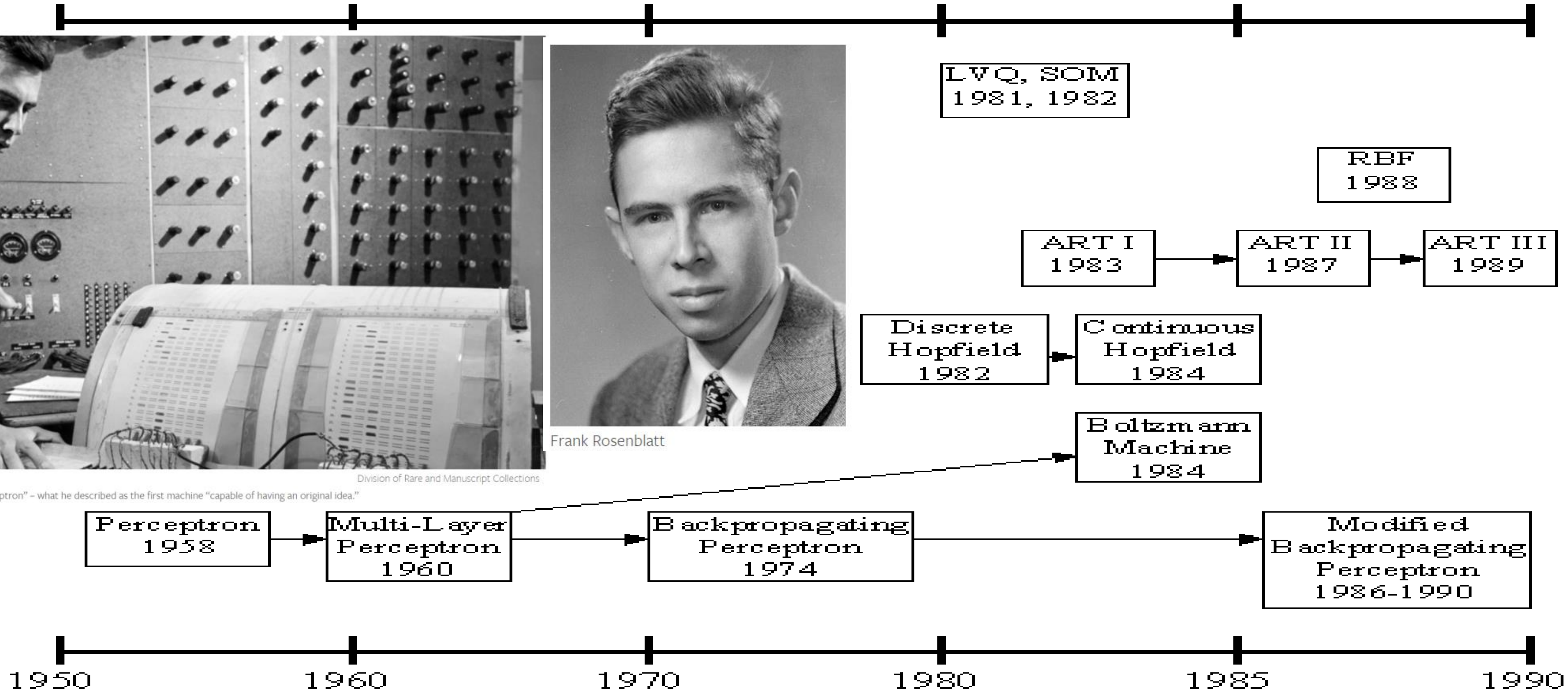


Division of Rare and Manuscript Collections

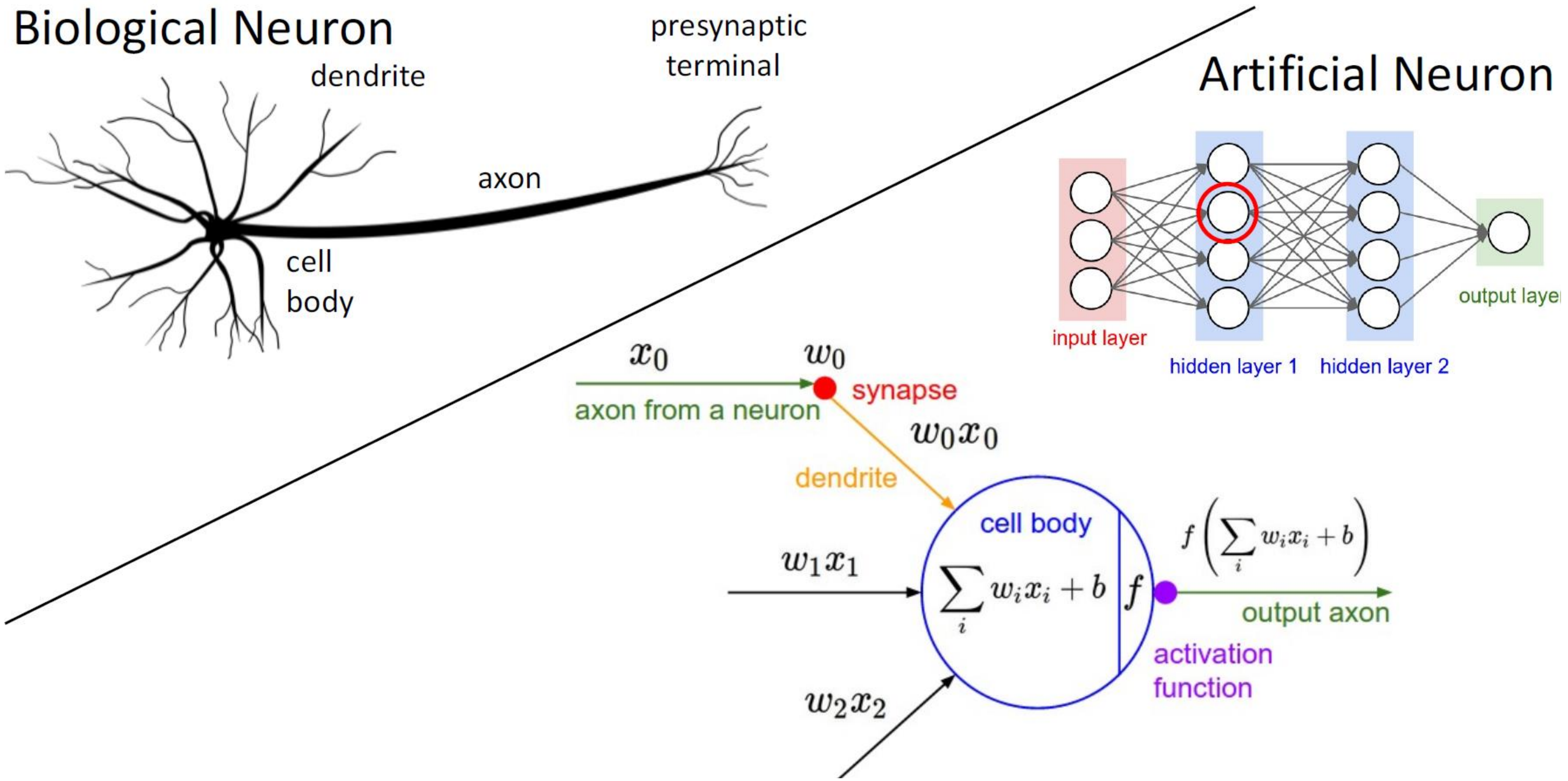
Frank Rosenblatt '50, Ph.D. '56, works on the "perceptron" – what he described as the first machine "capable of having an original idea."



Frank Rosenblatt

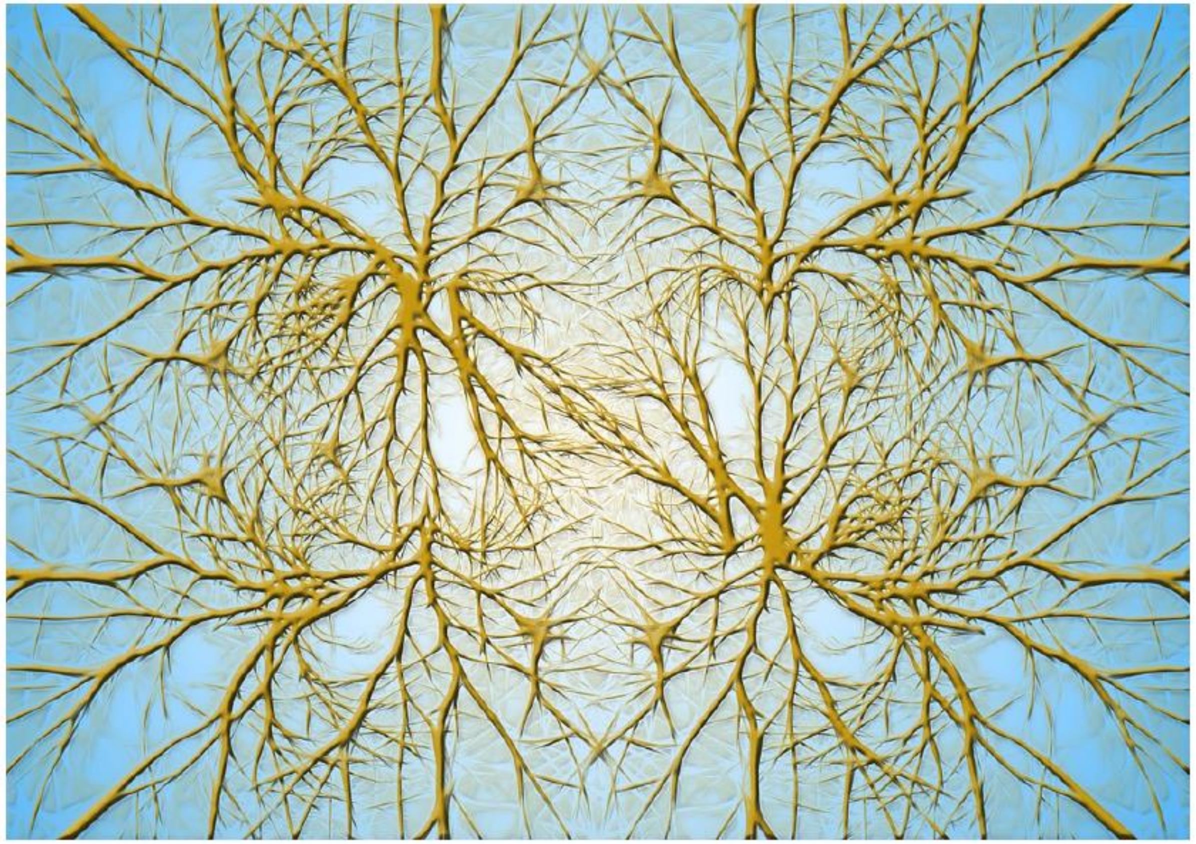


The relation between…

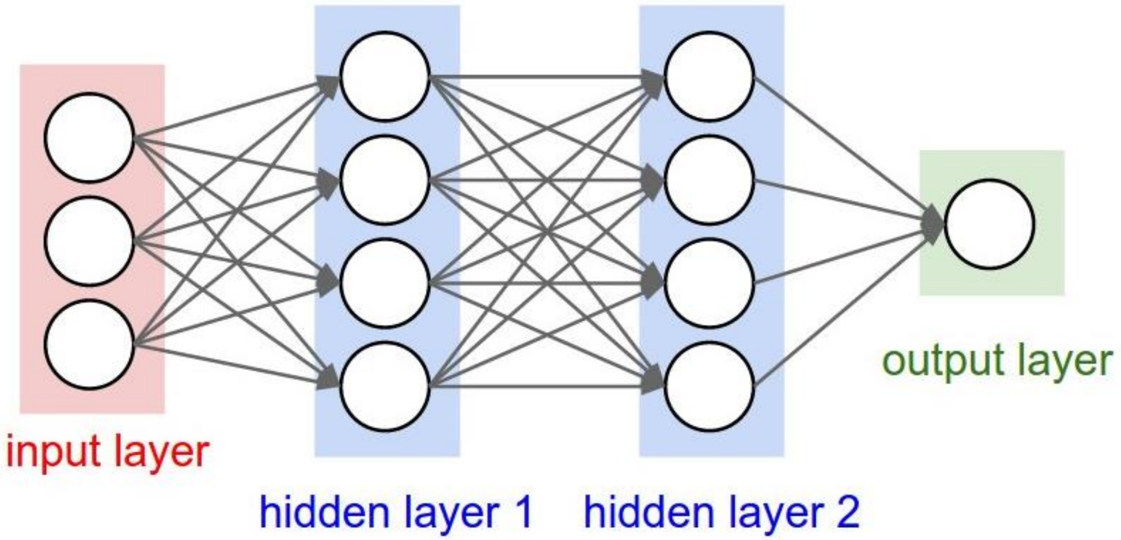


The relation between…

Biological Neurons:
Complex connectivity patterns

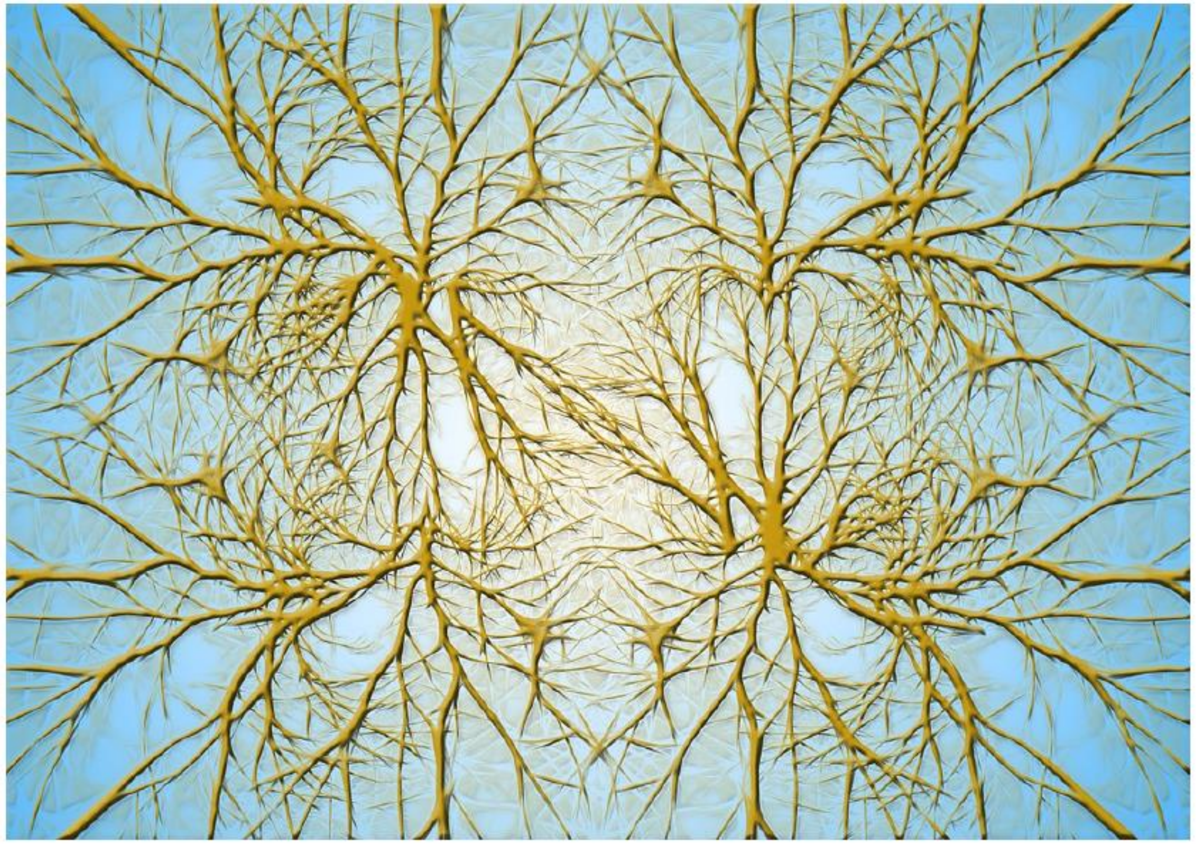


Neurons in a neural network:
Organized into regular layers for computational efficiency

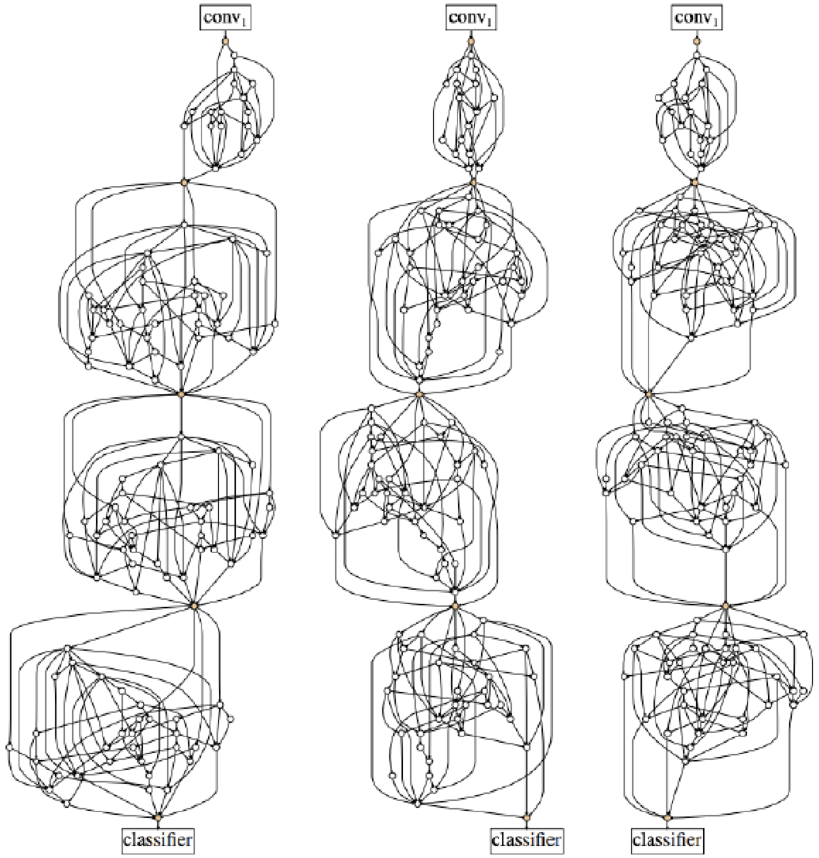


The relation between…

Biological Neurons:
Complex connectivity patterns

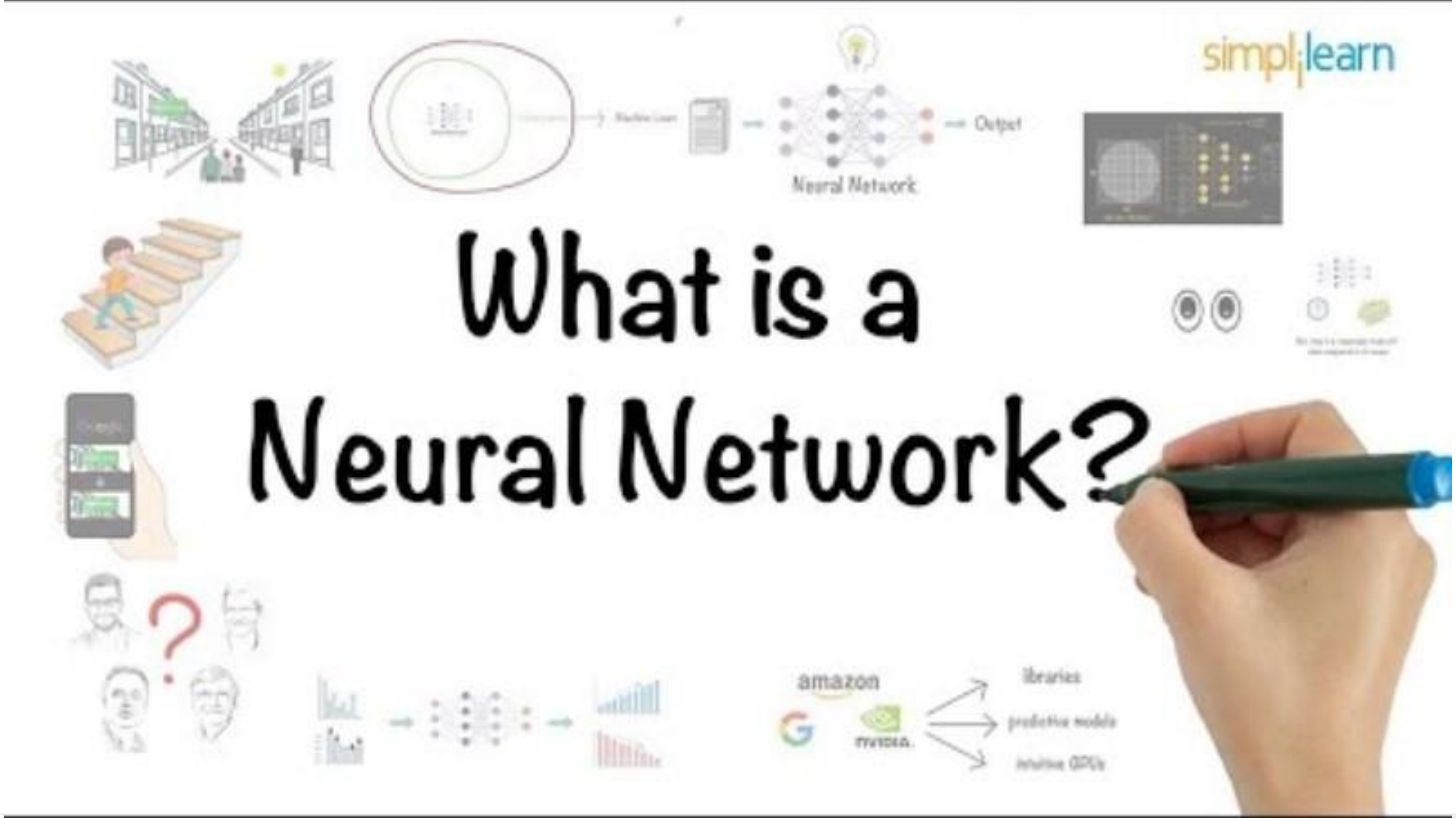


But neural networks with random connections can work too!



Xie, Saining, et al. "Exploring randomly wired neural networks for image recognition." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

The Artificial Neural Network

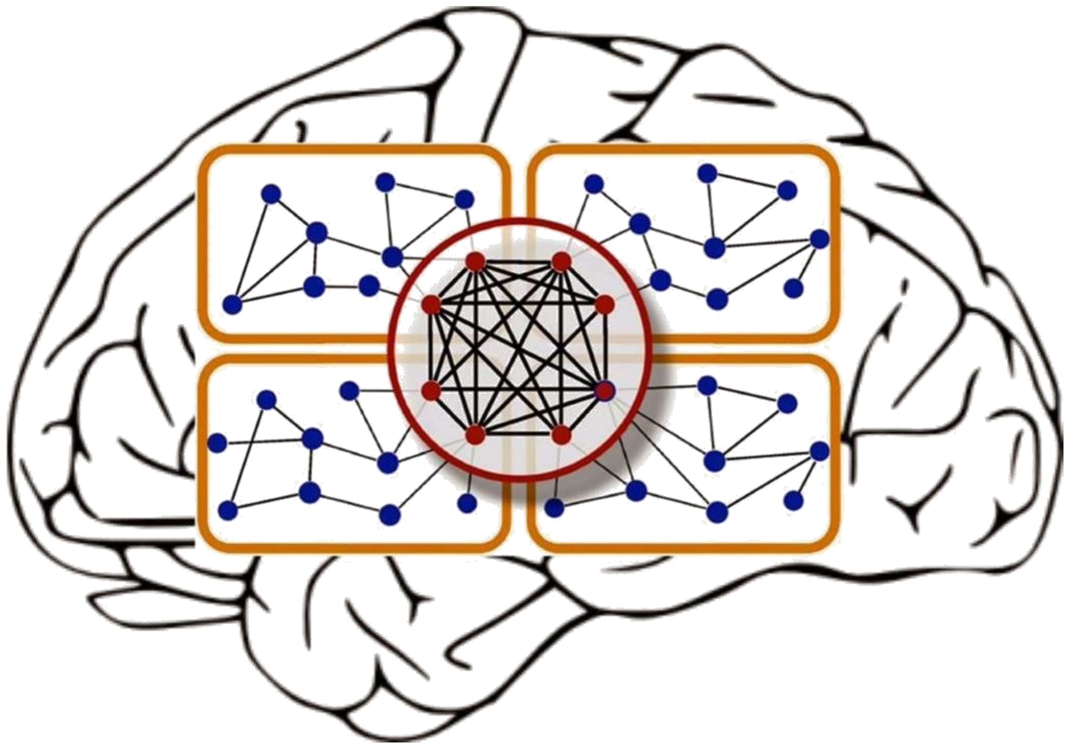


https://youtu.be/bfmFfD2RIcg?si=kWdVyVmD7T9mEeRc

Why ANN?

There are two basic reasons why we are interested in building artificial neural networks (ANNs):

- **Technical viewpoint:** Some problems such as character recognition or the prediction of future states of a system require massively parallel and adaptive processing.
- **Biological viewpoint:** ANNs can be used to replicate and simulate components of the human (or animal) brain, thereby giving us insight into natural information processing.



The Artificial Neuron: Perceptron

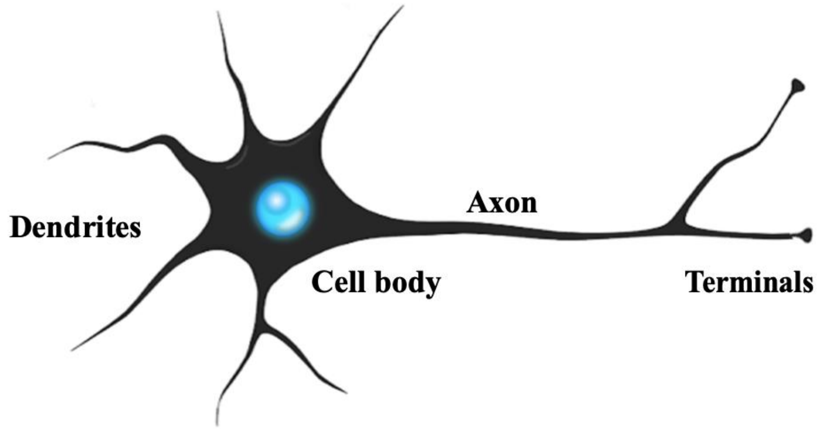
The “building blocks” of neural networks are the **neurons**.

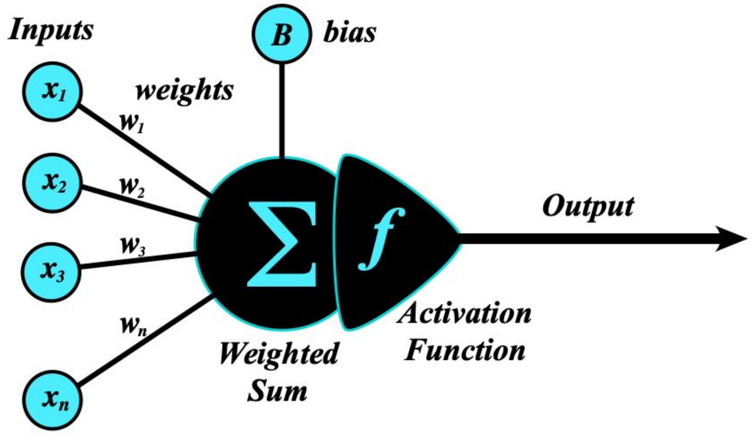
- Artificial Neurons are also called units or nodes. However, in the ANNs field are better known as **perceptron**.

Basically, each perceptron:

- Receives input from many other neurons.
- Changes its internal state (activation) based on the current input.
- Sends one output signal to many other neurons, possibly including its input neurons (recurrent network).

$$y = \left(\sum_{i=1}^{n} w_i x_i + b \right) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$





The single layer ANN

- Is the simplest NN and it is know as Functional Link Neural Network.
- It consist of a single layer of perceptrons.

$$W_\eta^\top \boldsymbol{\sigma}_\eta (V_\eta^\top \boldsymbol{\gamma}_\eta)$$

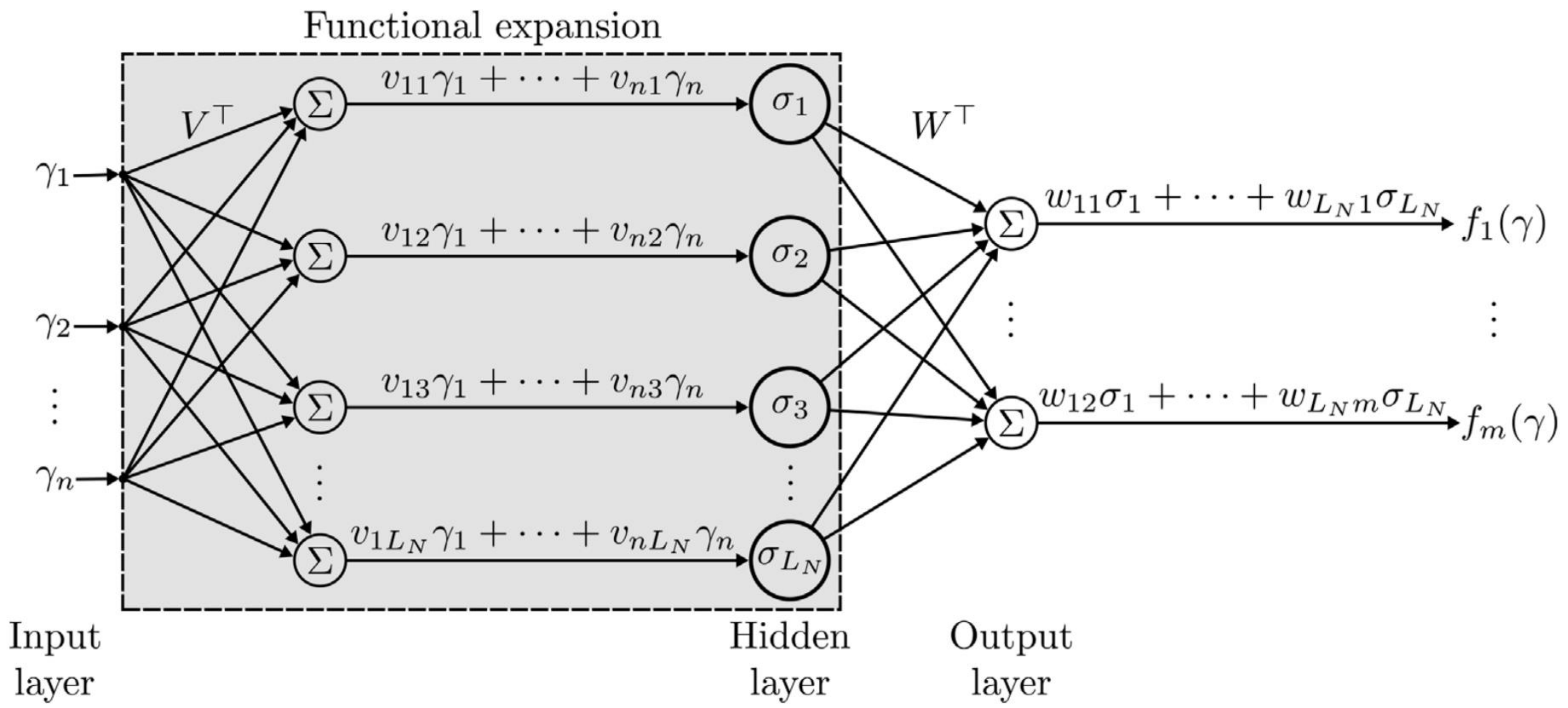
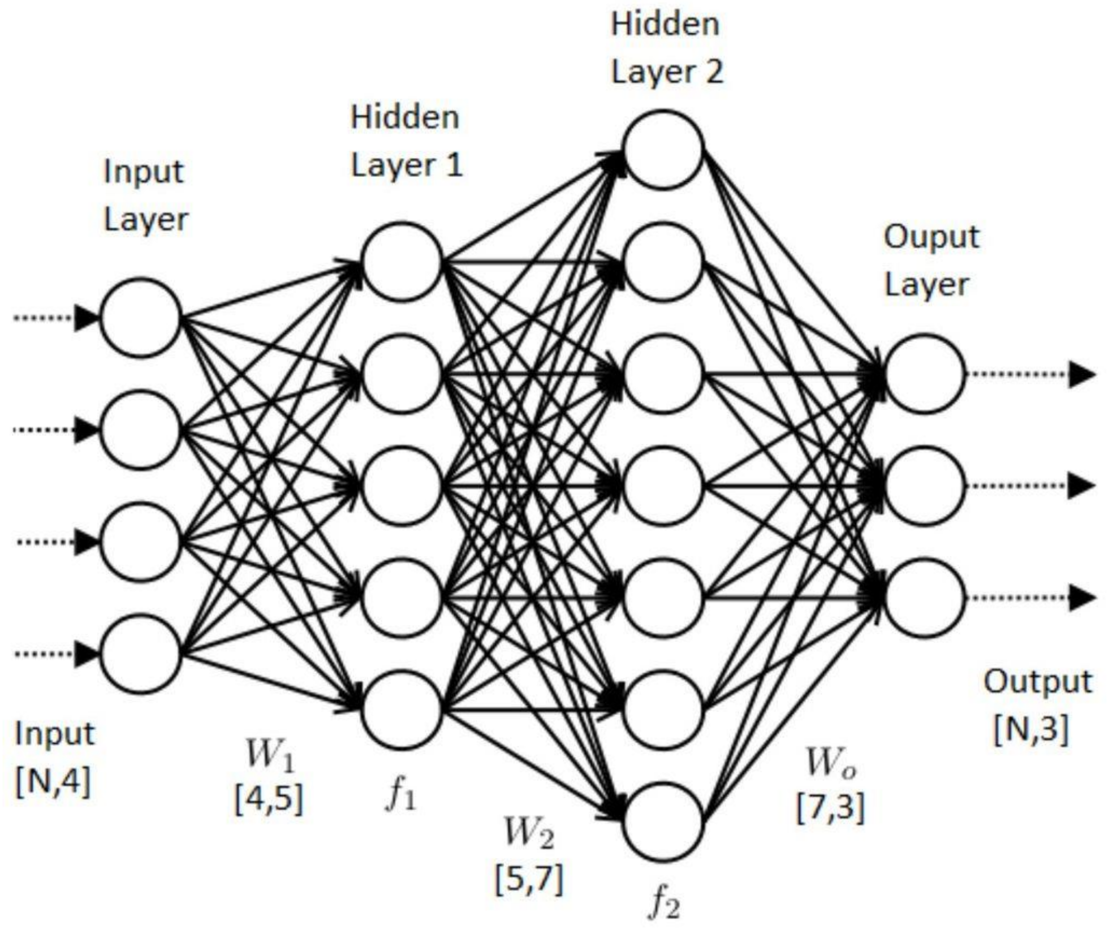

Functional expansion
$V^\top$
$W^\top$
γ_1
γ_2
γ_n
$v_{11}\gamma_1 + \cdots + v_{n1}\gamma_n$
$v_{12}\gamma_1 + \cdots + v_{n2}\gamma_n$
$v_{13}\gamma_1 + \cdots + v_{n3}\gamma_n$
$v_{1L_N}\gamma_1 + \cdots + v_{nL_N}\gamma_n$
σ_1
σ_2
σ_3
σ_{L_N}
$w_{11}\sigma_1 + \cdots + w_{L_N 1}\sigma_{L_N}$
$w_{12}\sigma_1 + \cdots + w_{L_N m}\sigma_{L_N}$
$f_1(\gamma)$
$f_m(\gamma)$
Input layer
Hidden layer
Output layer

Fig. 3. Diagram of a FLNN where $\boldsymbol{\gamma} \in \mathbb{R}^n$ is the NN's extended input vector, $V \in \mathbb{R}^{n \times L_N}$ and $W \in \mathbb{R}^{L_N \times m}$ are the input and output weights matrices, respectively, $\boldsymbol{\sigma} \in \mathbb{R}^{L_N}$ is the vector of activation functions, $\boldsymbol{f}(\boldsymbol{\gamma}) \in \mathbb{R}^m$ NN's output vector, and $L_N \in \mathbb{R}$ is the number of neurons in the hidden layer.

Lopez-Sanchez, Ivan, Ricardo Pérez-Alcocer, and Javier Moreno-Valenzuela. "Trajectory tracking double two-loop adaptive neural network control for a Quadrotor." Journal of the Franklin Institute 360.5 (2023): 3770-3799.

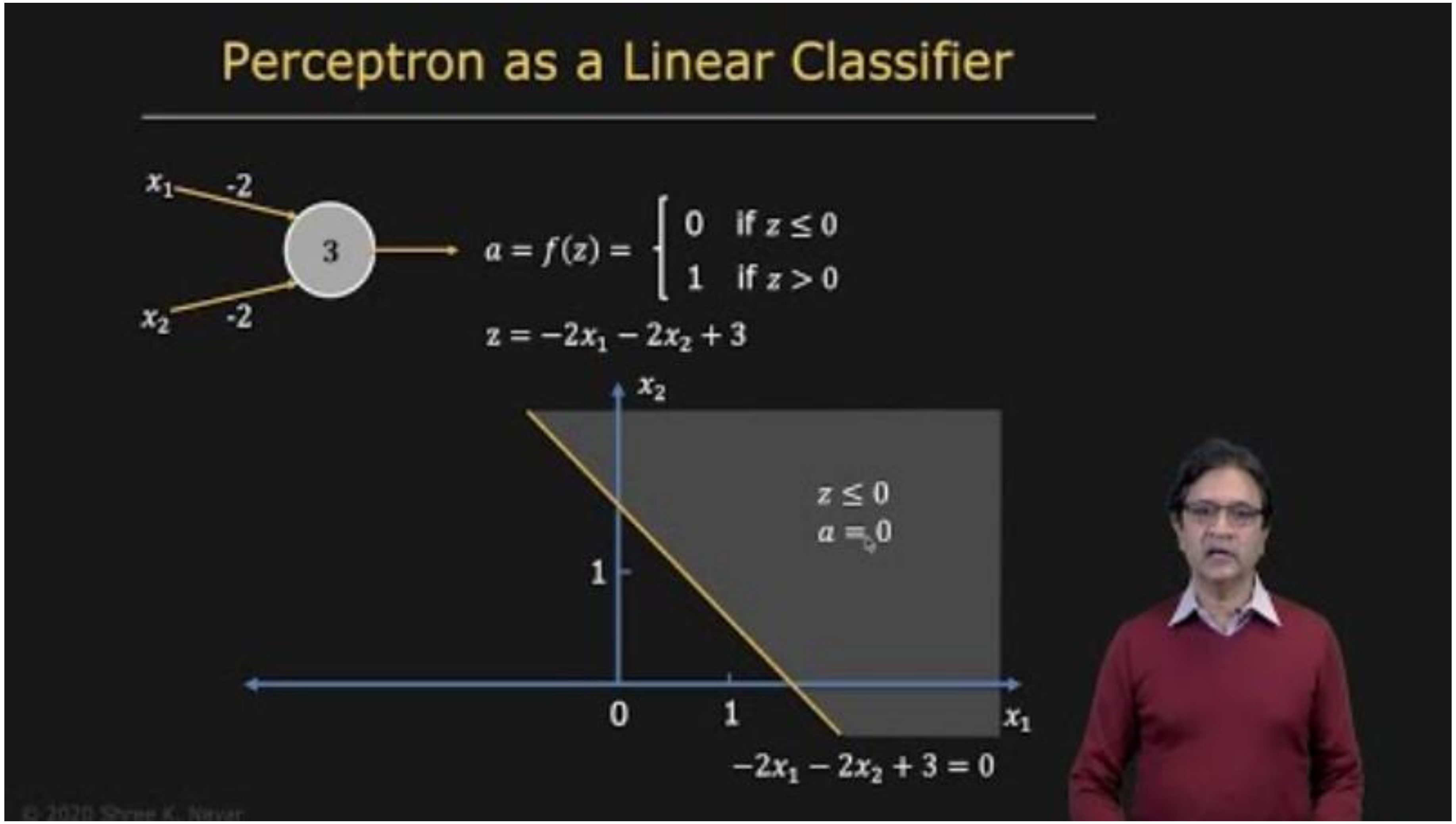
Then, an ANN

- Consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections.



The Artificial Neuron: Perceptron

The “building blocks” of neural networks are the **neurons.** https://www.youtube.com/watch?v=OFbnpY_k7js



Perceptron as a Linear Classifier

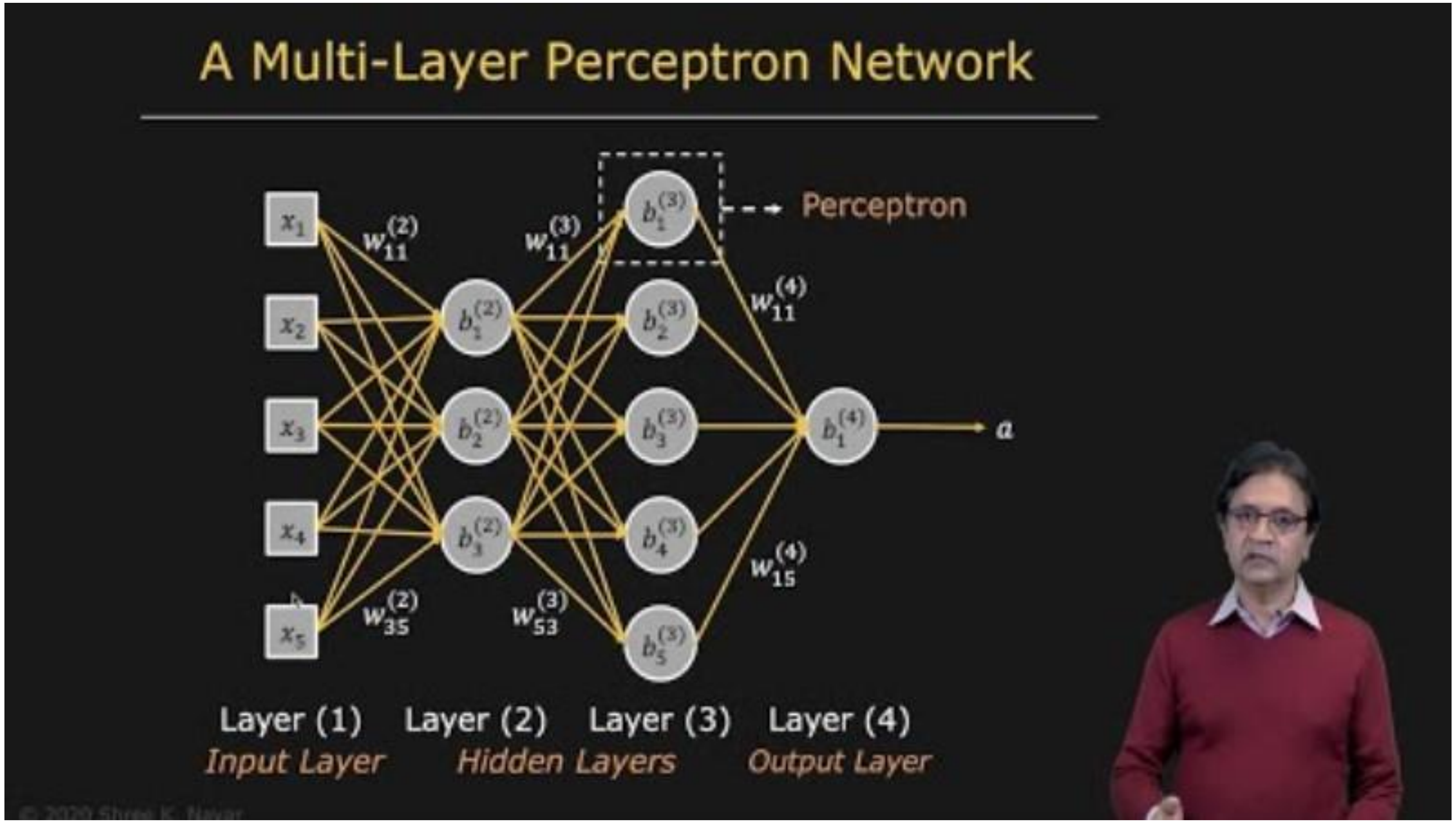
$$a = f(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

$$z = -2x_1 - 2x_2 + 3$$

$$-2x_1 - 2x_2 + 3 = 0$$

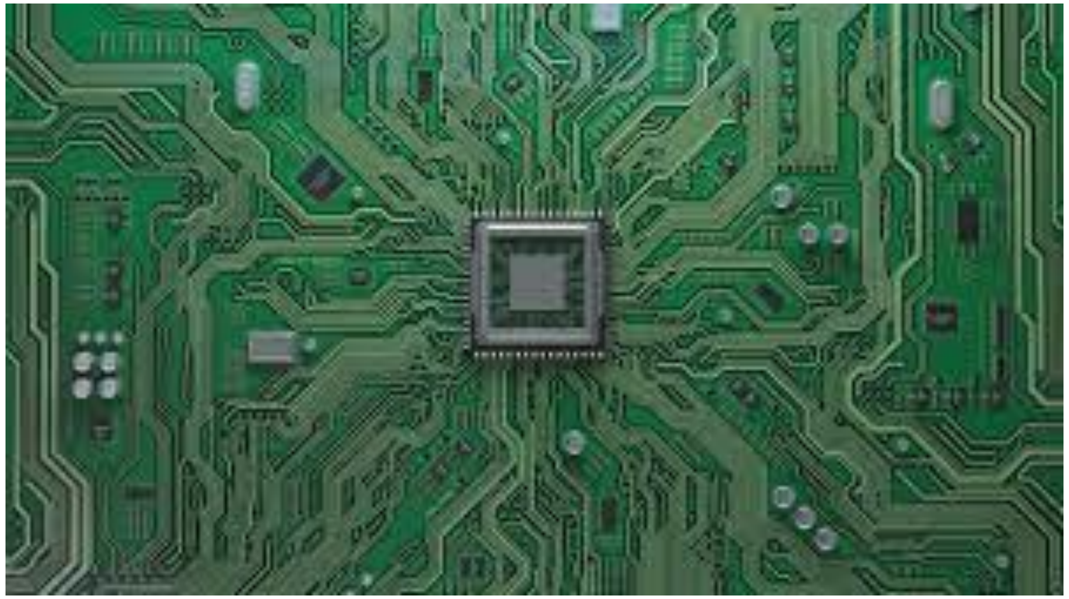
Multilayer Perceptron (MLP) NN

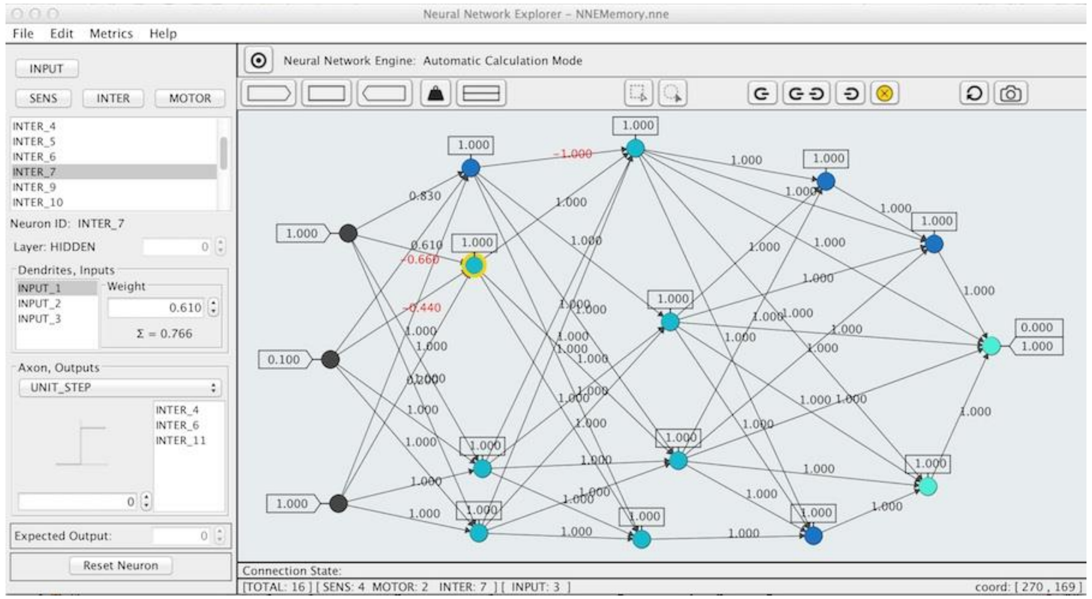
The base of artificial neural networks are the MLPs. https://www.youtube.com/watch?v=torNuKNLwBE



In general words

- An ANN is either a hardware implementation or a computer program which strives to simulate the information processing capabilities of its biological exemplar.
- ANNs are typically composed of a great number of interconnected artificial neurons.
- The artificial neurons are simplified models of their biological counterparts.
- ANN is a technique for solving problems by constructing software that works like our brains.





Neural Network Engine: Automatic Calculation Mode

Neuron ID: INTER_7

Layer: HIDDEN

Dendrites, Inputs

	Weight
INPUT_1	0.610
INPUT_2	
INPUT_3	

Σ = 0.766

Axon, Outputs

UNIT_STEP

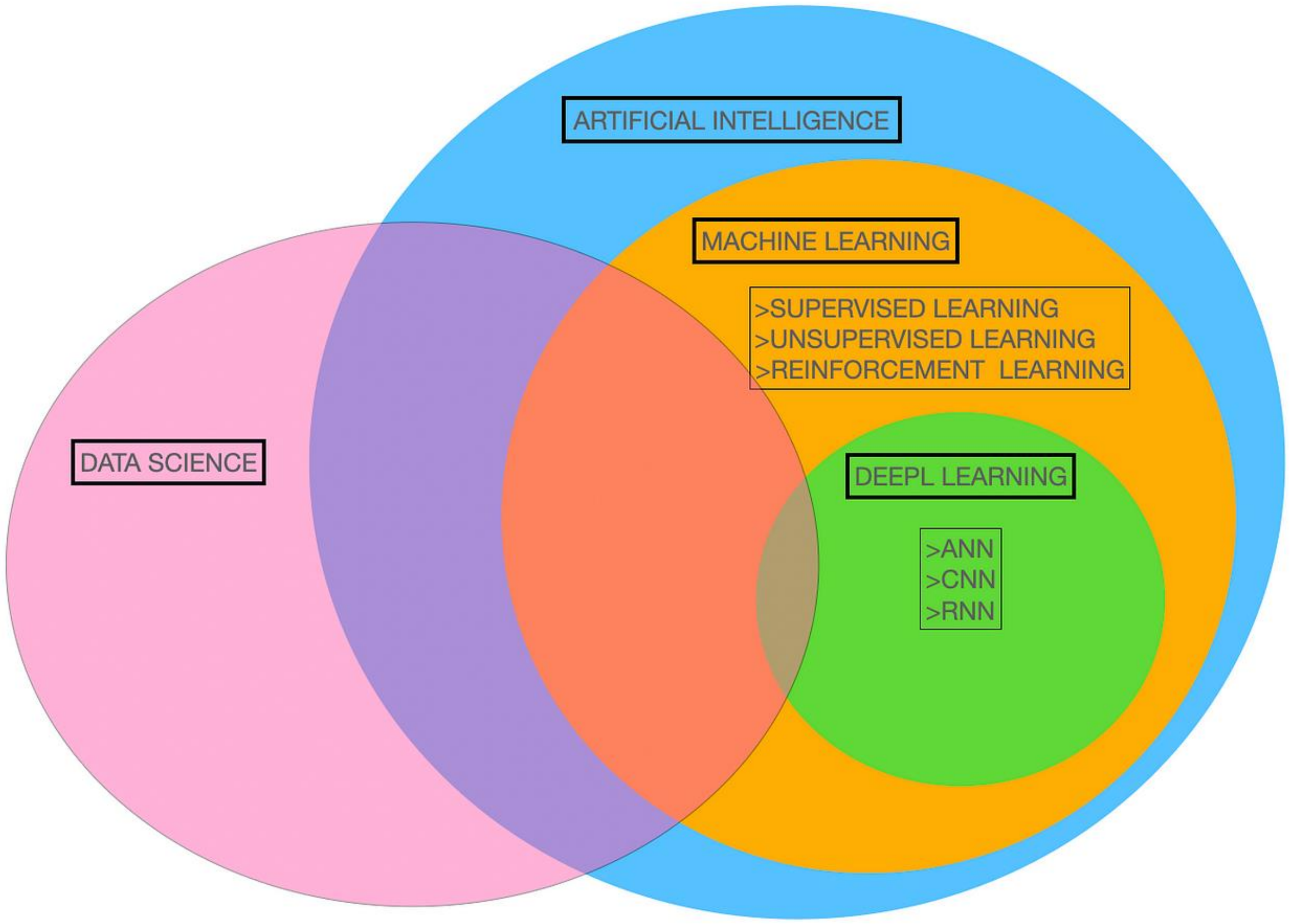
INTER_4
INTER_6
INTER_11

Expected Output: 0

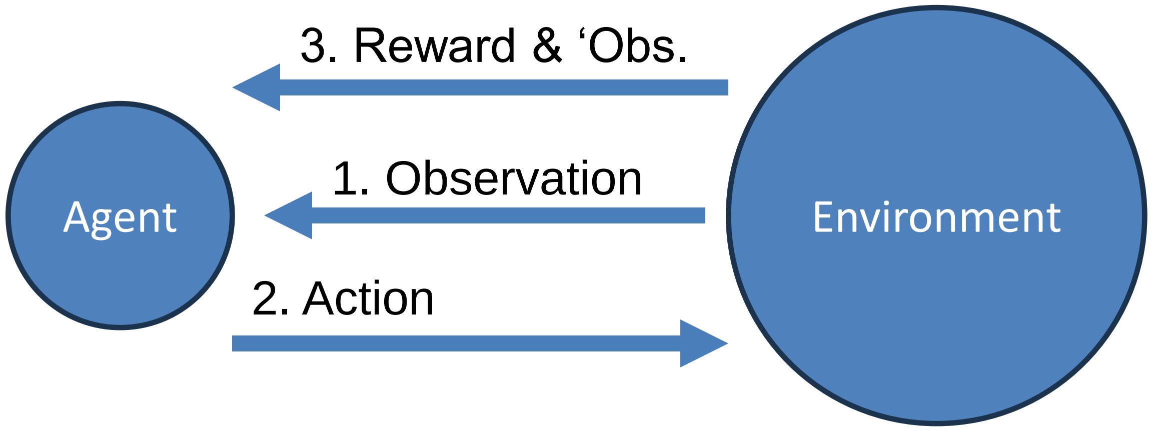
Connection State:
[TOTAL: 16] [SENS: 4 MOTOR: 2 INTER: 7] [INPUT: 3]

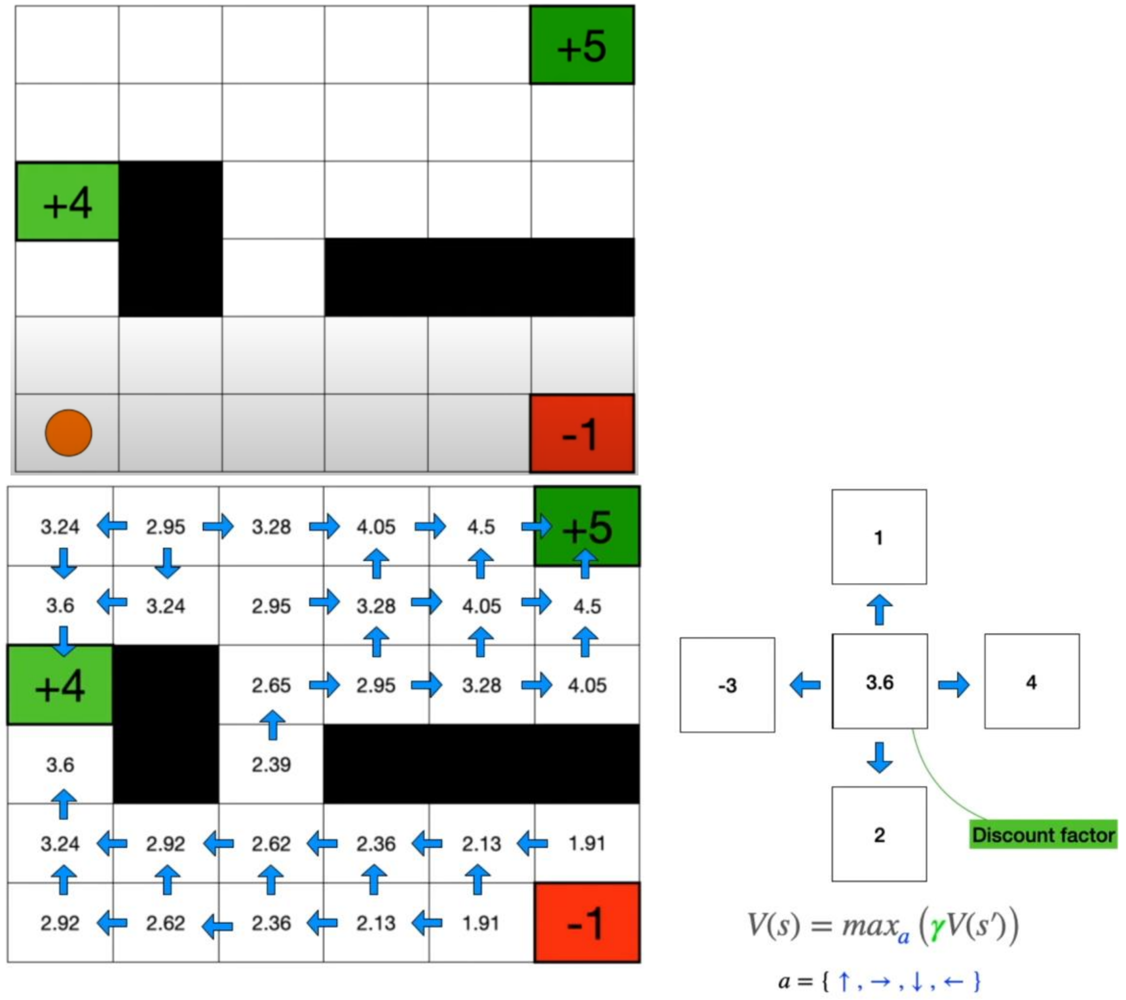
coord: [270 , 169]

Reinforcement learning



General interaction in RL





$$V(s) = max_a\left(\gamma V(s')\right)$$

$$a = \{\uparrow, \rightarrow, \downarrow, \leftarrow\}$$

Agent

- Entity that learns to make decisions by interacting with an environment. It is form by:
  - Policy
  - Value functions
  - Model of the environment (optional)

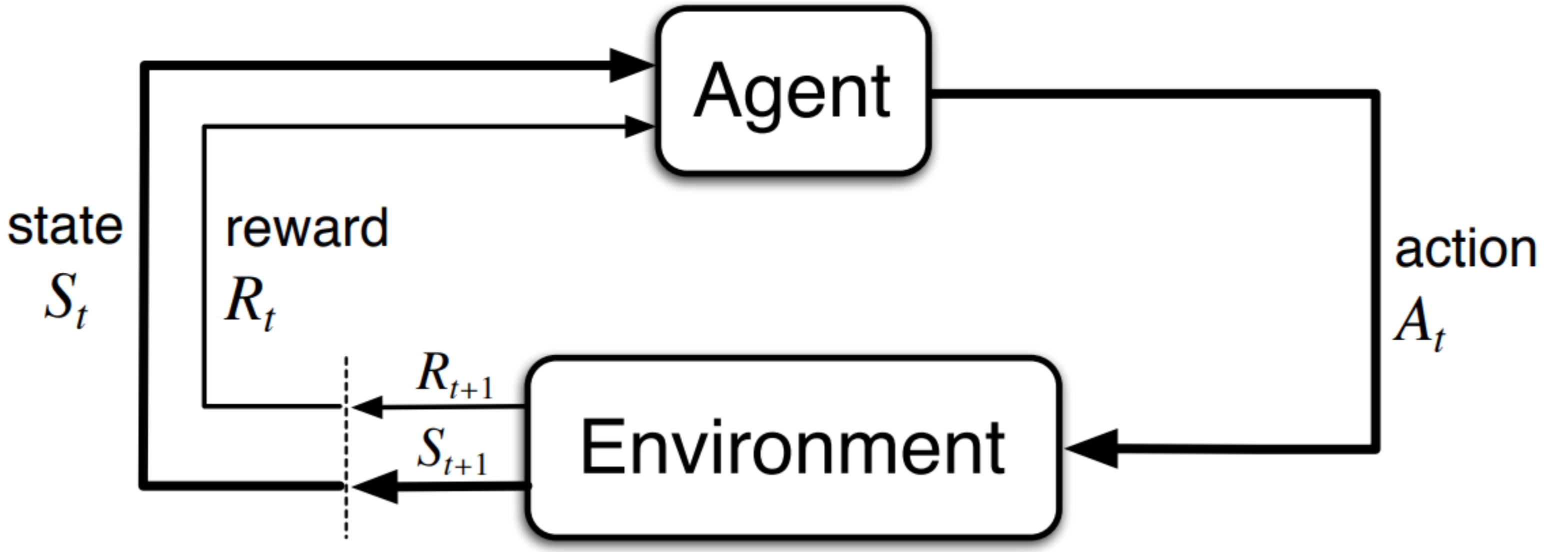
Its goal is to optimize its behavior and maximize cumulative rewards over time.

It makes decisions based on its observations of the environment and receives feedback in the form of rewards.

The learning process involves exploring the environment, exploiting known information, and updating the policy based on feedback received from the environment.

Policy

- Policy (π)
  - The policy is a strategy used by the agent to decide which action to take in a given state. It can be deterministic (always selecting the same action for a state) or stochastic (selecting actions according to a probability distribution).
- The policy can be represented as a mapping from states to actions, $\pi : S \rightarrow A$, or as a probability distribution over actions, $\pi(a|s)$.



Value function

- The value function estimates the expected cumulative reward that can be obtained from a state (or state-action pair).

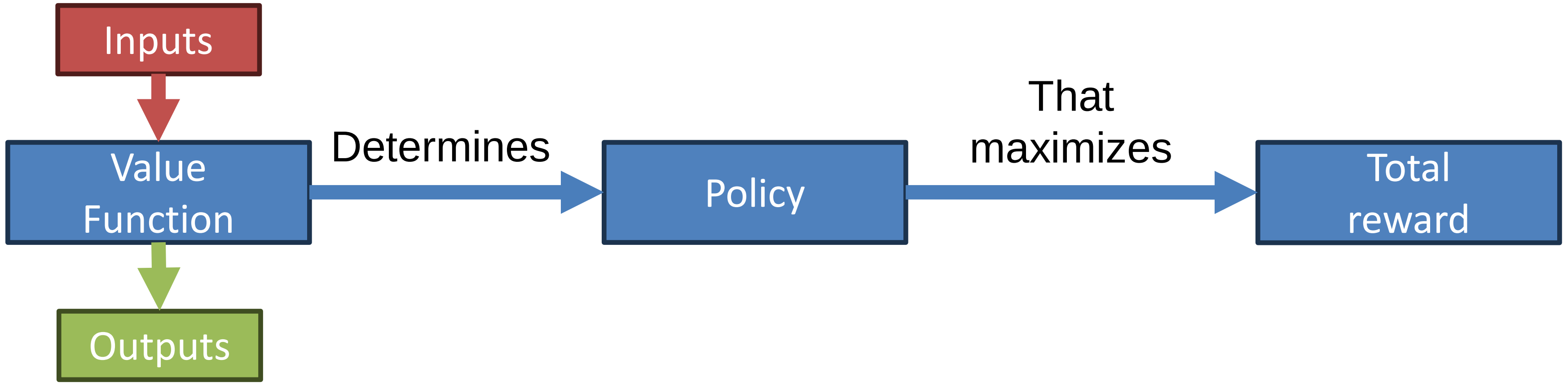
- State Value Function (V): The expected cumulative reward starting from state s and following the policy π.

$$V^{\pi}(s)=E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R_{t+1}|S_t=s\right]$$

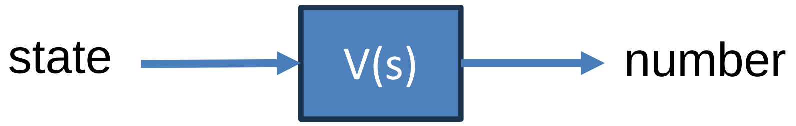
- Action Value Function (Q): The expected cumulative reward starting from state s, taking action a, and following the policy π.

$$Q^{\pi}(s,a)=E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R_{t+1}|S_t=s,A_t=a\right]$$

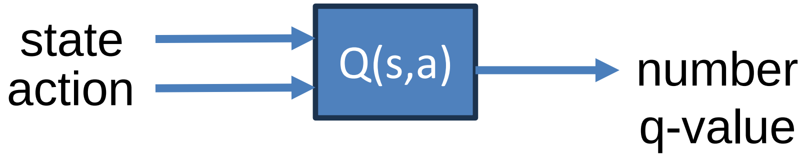
Value functions



State value function



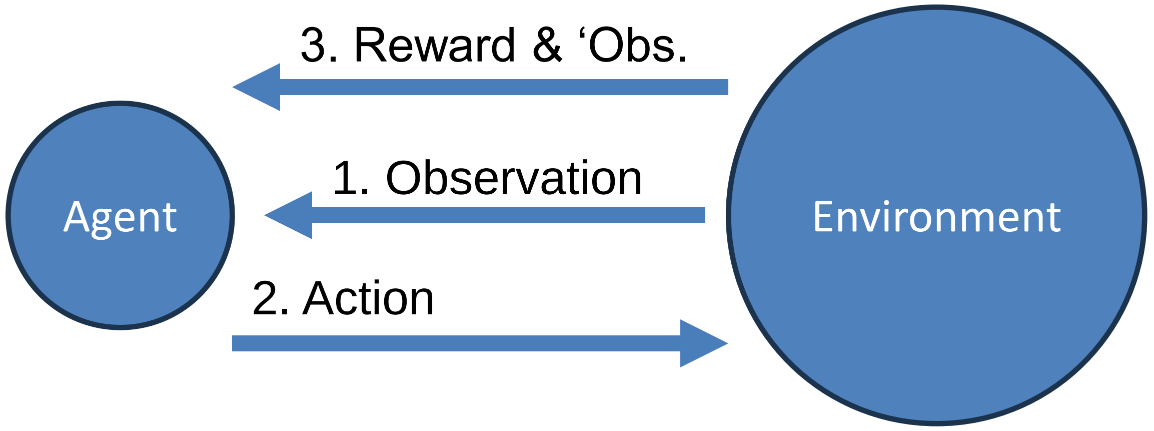
State-action value function



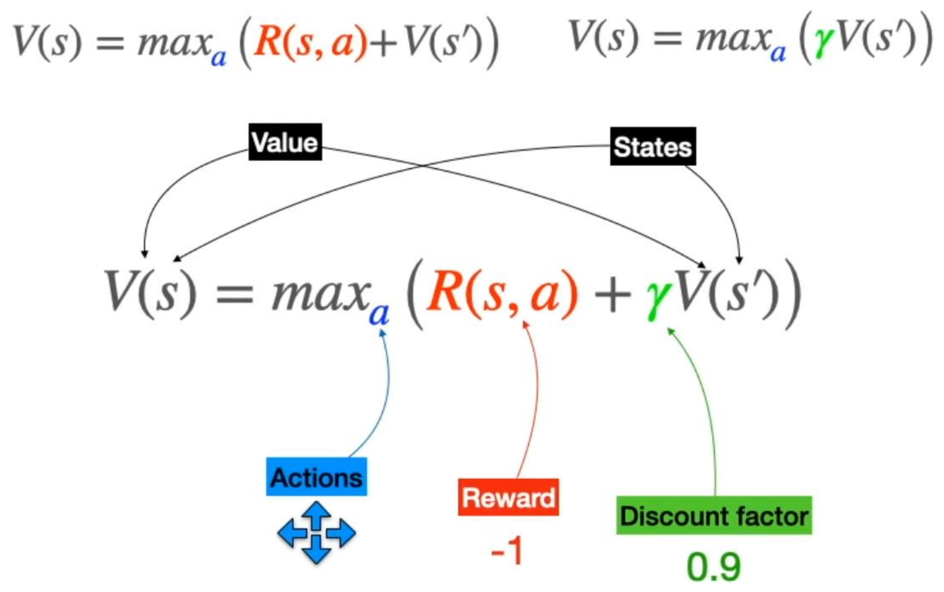
- State value function let us know how good is to be in certain state.

- State-action value functions let us know how good is to take certain action given the actual state.

Interaction of the elements

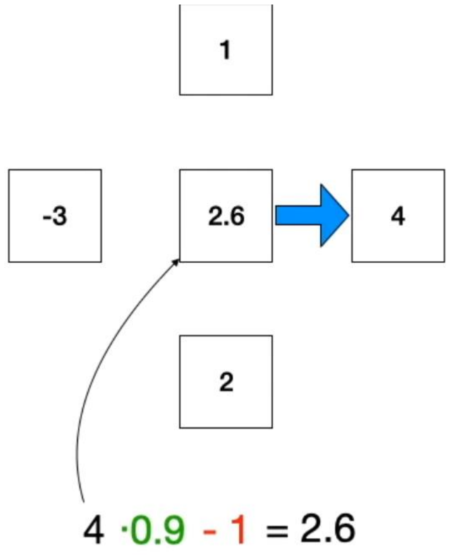


Bellman’s equation example

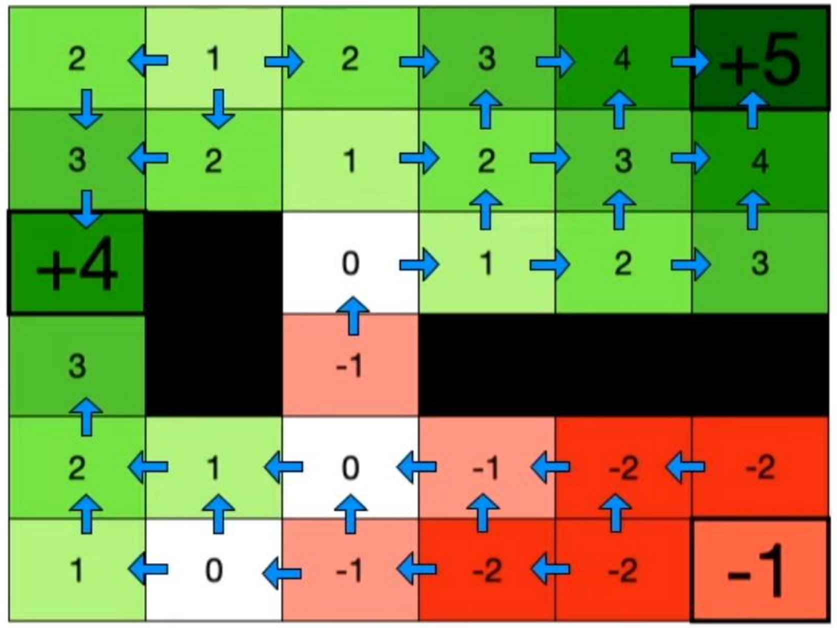


$$V(s) = max_a\left(R(s,a) + V(s')\right) \qquad V(s) = max_a\left(\gamma V(s')\right)$$

$$V(s) = max_a\left(R(s,a) + \gamma V(s')\right)$$



$4 \cdot 0.9 - 1 = 2.6$



Q-Learning

- It is about the total reward
- The Q-function:
  - States
  - Actions
  - Total cumulative reward

$$R_T = Q(s,a)$$

- The policy is the function that maps from states to actions $a = \pi(s)$
- Putting them together $R_T = Q_\pi(s,a)$

- The reward function
  - Discount factor

$$R_T = R_1 + R_2 + R_3 + \cdots + R_n$$

$$R_T = R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots + \gamma^n R_{n+1} = \sum_{i=0}^{n} \gamma^i R_{i+1}$$

- The Bellman's Equation (for this purpose)

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r(s',a') + \gamma \max_a Q(s',a) - Q(s,a))$$

Deep Q-Network (DQN)

Deep reinforcement learning algorithm.

- Uses a neural network to approximate the Q-values. The network takes the state as input and outputs Q-values for all possible actions.
- Uses experience replay, where experiences (state, action, reward, next state) are stored in a replay buffer. During training, random mini-batches are sampled from this buffer to break correlation between consecutive experiences and stabilize training.
- Employs a target network (a copy of the main network), which is updated less frequently. This helps to stabilize training by providing consistent target Q-values.
- The neural network is trained using a gradient descent algorithm, minimizing the mean squared error between the predicted Q-values and the target Q-values:

$$L = \left(r(s,a) + \gamma \max_{a'} Q^{*}(s',a') - Q(s,a)\right)^{2}$$

- It is suitable for environments with large or continuous state spaces, where the Q-table would be impractical. The use of neural networks allows it to generalize across similar states and actions.

Q-Learning VS DQN

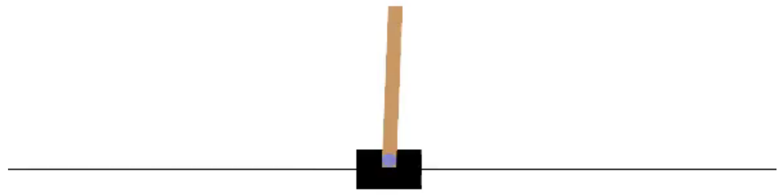
	Q-learning	Deep Q-Network
Q-values representation	Q-table	Neural Network
Scalability	Small and discrete state-action spaces	Large or continuous state spaces
Learning stability	Stable only on small state action spaces	Stable training given its experience replay and target network feature
Update mechanism	Bellman’s equation	Gradient descent and loss minimization (other algorithms can be employed)
Computational Requirements	Low computational power and memory consumption	High computational resources needed
When to use it	When dealing with small, discrete state-action spaces	When dealing with large or continuous state-action spaces

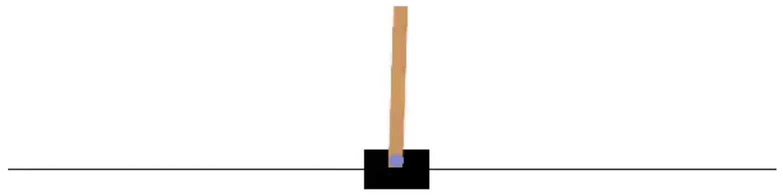
Cart-pole example

Objective

1. Understand the basic concepts in reinforcement learning:
    1. Environment
    2. Agent
    3. States
    4. Observation
    5. Action
    6. Reward
    7. Policy

2. Compare the performance of the trained policy regarding
    1. Number of neurons
    2. Number of network hidden layers
    3. Training episodes

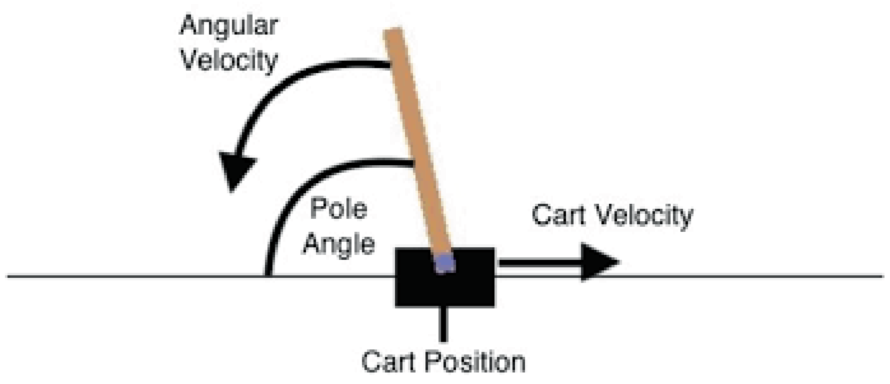


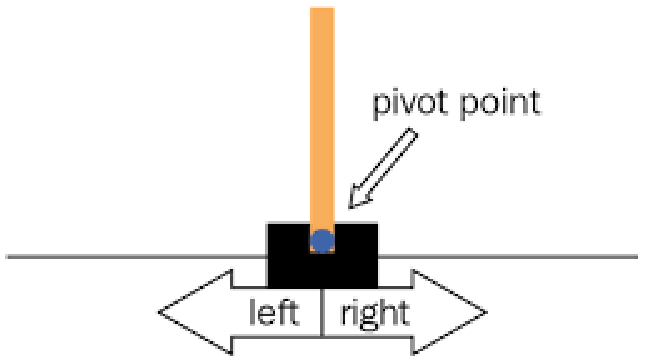


Using Google Colab and OpenAI Gym

Cart-pole Environment

- **Cart and Pole Dynamics:**
  - **Cart: A small cart that can move left or right along a frictionless track.**
  - **Pole: A pole attached to the cart by an unactuated rotational joint. The pole starts upright and can fall to either side.**
- **State Space (four-dimensional vector):**
  - **cart_position: The position of the cart on the track.**
  - **cart_velocity: The velocity of the cart.**
  - **pole_angle: The angle of the pole from the vertical.**
  - **pole_velocity_at_tip: The velocity of the tip of the pole.**
- **Action Space (discrete with two possible actions):**
  - **0: Apply a force to the cart to move it to the left.**
  - **1: Apply a force to the cart to move it to the right.**
- **Reward:**
  - **The agent receives a reward of +1 for every time step the pole remains upright and within the allowed boundaries.**
- **Termination Conditions (episode):**
  - **The pole angle exceeds ±12 degrees from the vertical.**
  - **The cart position exceeds ±2.4 units from the center.**
  - **The episode length reaches a maximum of 200 steps (500 steps in our example).**





Homework 1

Based on the Examples obtained in the Google Colab Notebook.

1. Compare the performance and reward values per episode for different models employing the same agent (Deep Q-Network). Please describe your findings in text and attach figures and videos, if necessary.
   1. Training episodes: Model with 2 hidden layers and 24 neurons per layer with relu activation funcions for the hidden layers and a linear activation function for the output layer.
      1. 5,000 training episodes.
      2. 10,000 training episodes.
      3. 20,000 training episodes.
   2. Number of neurons per layer: Model with 2 hidden layers with relu activation functions for the hidden layers and a linear activation function for the output layer, 5000 training episodes.
      1. 24 neurons per layer
      2. 32 neurons per layer
      3. 40 neurons per layer
   3. Number of hidden layers: Model with 24 neurons per layer with relu activation funcions for the hidden layers and a linear activation function for the output layer, 5,000 training episodes.
      1. 2 hidden layers
      2. 4 hidden layers
      3. 6 hidden layers

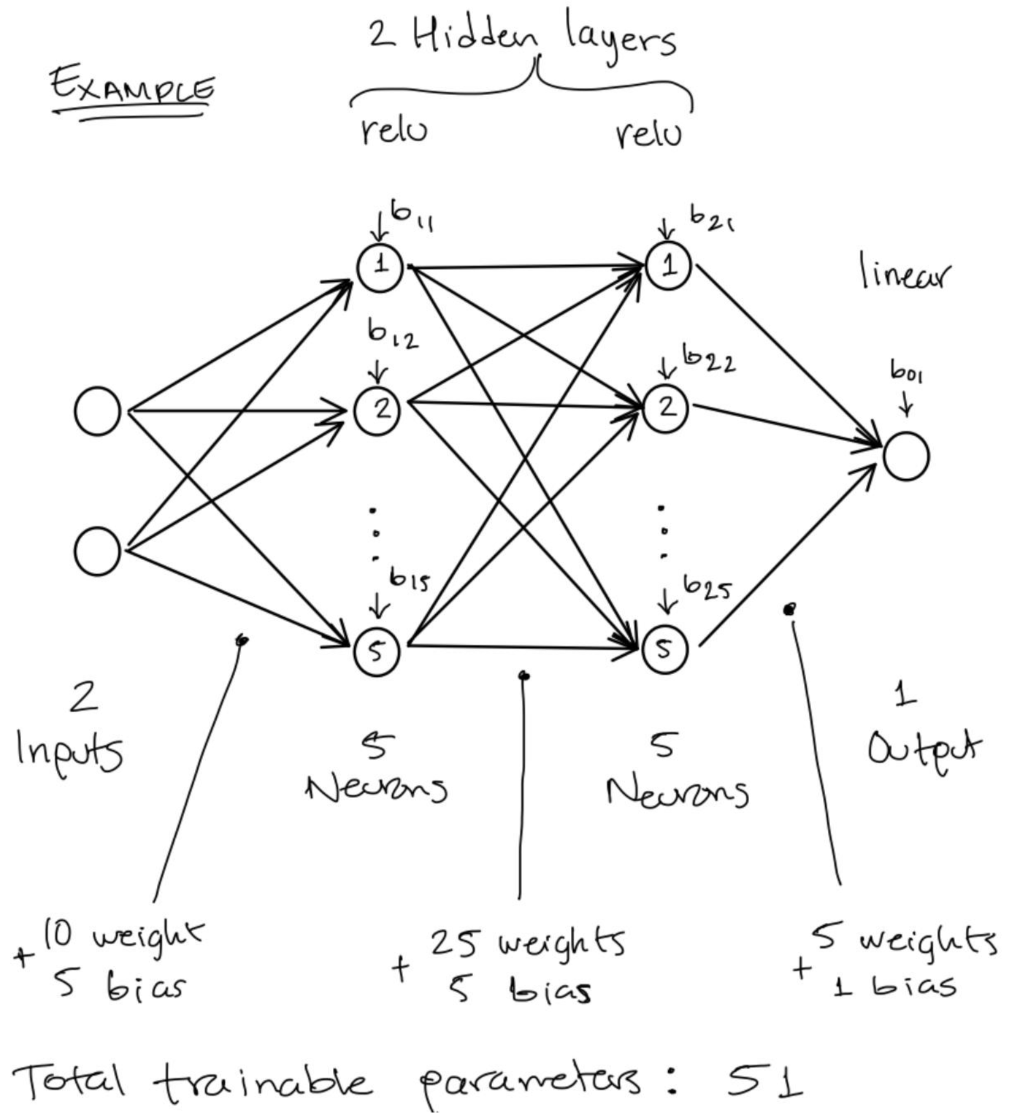
HINT: you can use several Google Colab notebooks simultaneously to accelerate the process of training each model.

Homework 1

Based on the Examples obtained in the Google Colab Notebook.

1. Submit a brief answer to the questions below using both text and figures.
   1. Diagram of each of the neural network architectures (without repeating them) specifying:
      1. Number of inputs
      2. Number of outputs
      3. Weights between each layer (trainable parameters)
      4. Biases (per layer)
      5. Total of trainable parameters

HINT: You can use any software to make the diagrams (or can be done by hand).


EXAMPLE
2 Hidden layers
relu
relu
linear
b_{11}
b_{12}
b_{15}
b_{21}
b_{22}
b_{25}
b_{01}
1
2
5
1
2
5
2 Inputs
5 Neurons
5 Neurons
1 Output
+ 10 weight 5 bias
+ 25 weights 5 bias
+ 5 weights 1 bias
Total trainable parameters : 51