

THE BIROBOTICS  
INSTITUTE



Scuola Superiore  
Sant'Anna

# Neuromorphic computing

## Robotics

M.Sc. programme in Computer Science

Lorenzo Vannucci

[lorenzo.vannucci@santannapisa.it](mailto:lorenzo.vannucci@santannapisa.it)

8 April 2019



# Outline

1. Introduction
2. Fundamentals of neuroscience
3. Simulating the brain
4. Software and hardware simulations
5. Robotic applications



# Outline

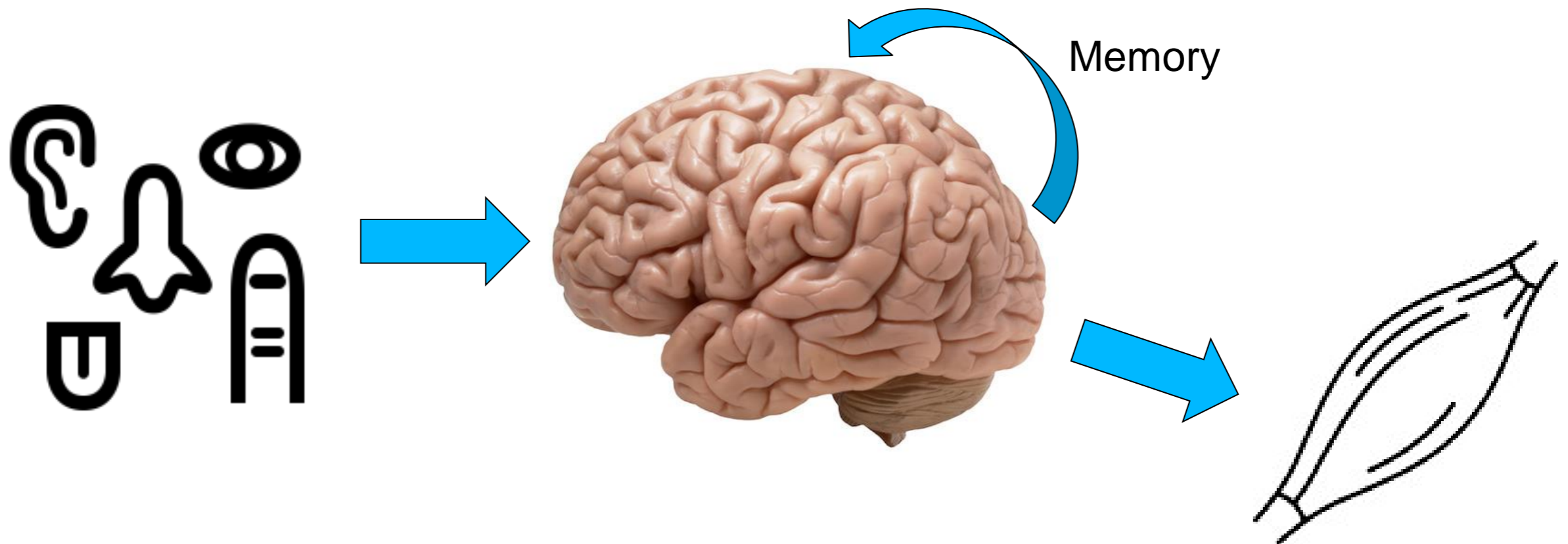
- 1. Introduction**
2. Fundamentals of neuroscience
3. Simulating the brain
4. Software and hardware simulations
5. Robotic applications



# What is neuromorphic computing?

We can define **neuromorphic computing** as the act of performing a computation in a manner similar to the brain.

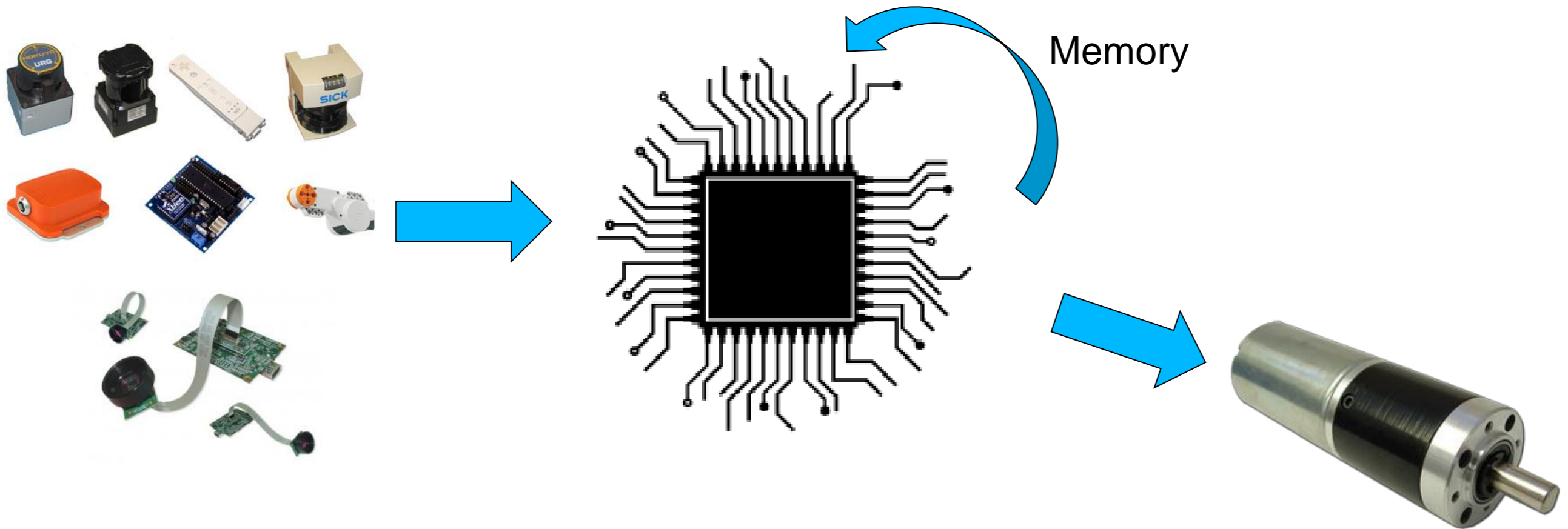
Our brain elaborates inputs coming from our sensors and produces outputs in term of generated motions and stored information.



# Why neuromorphic computing (in robotics)?

This kind of computing is very similar to what can be found in a robotic controller.

But the sensors and actuators are completely different, compared to the ones of humans and animals, thus the brain is substituted by a computer.



# Why neuromorphic computing (in robotics)?

Today, bio-inspired sensing and actuation technologies are starting to emerge.

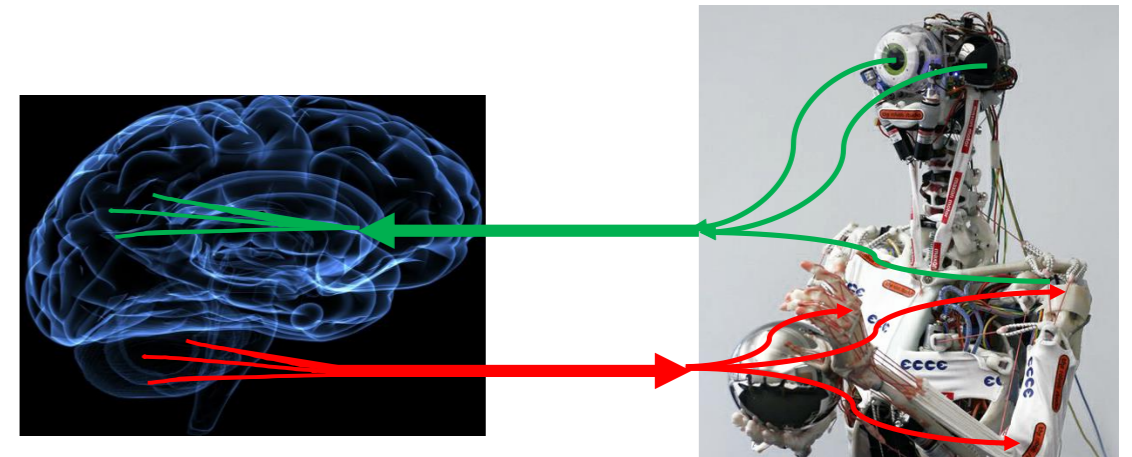


As such, neuromorphic computing can be used to control this new kind of robots.



# Why neuromorphic computing (in robotics)?

Moreover, we found out that classic control techniques fail when applied to complex robotic platforms.



Thus, we would like to apply neuromorphic computing to **give brains to robots.**



# Outline

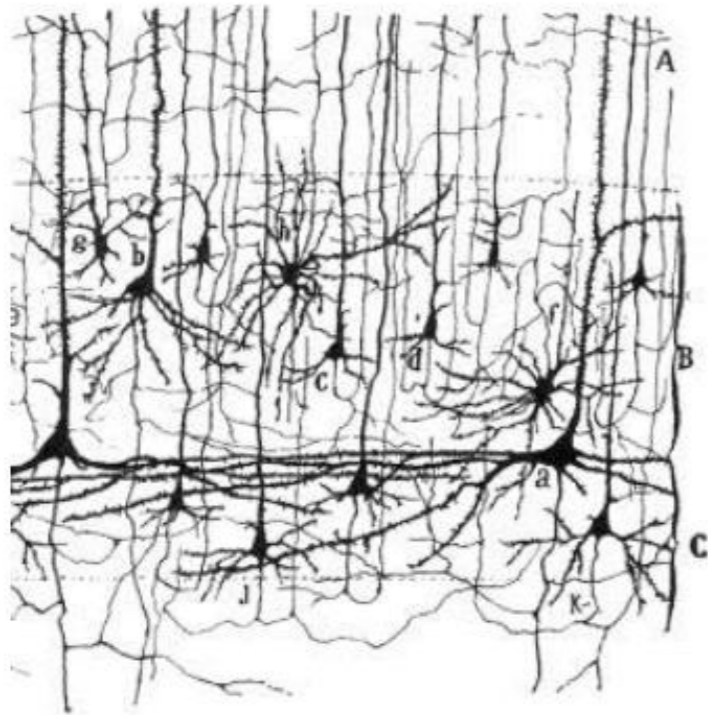
1. Introduction
- 2. Fundamentals of neuroscience**
3. Simulating the brain
4. Software and hardware simulations
5. Robotic applications



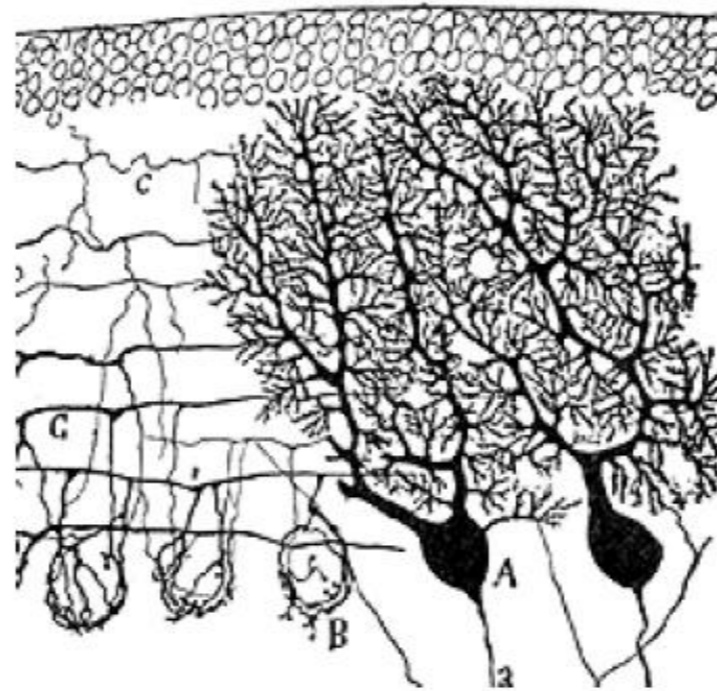


# Neuronal physiology

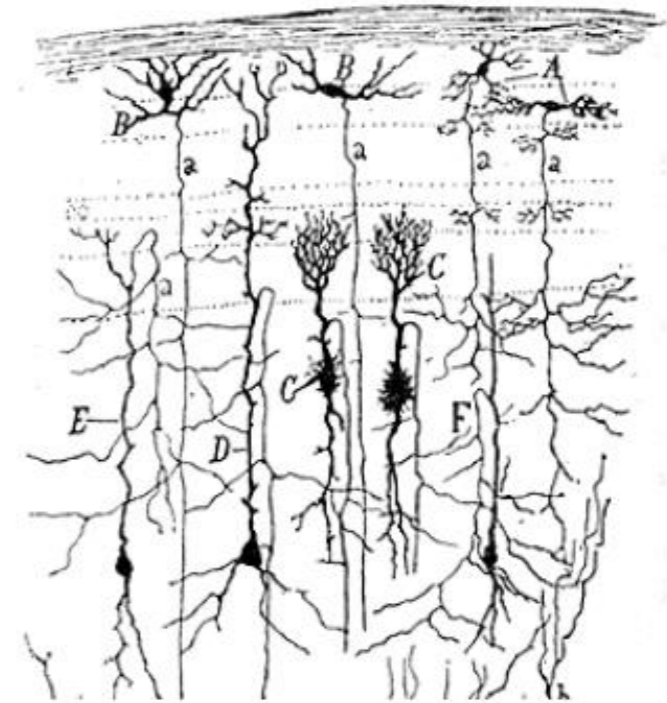
The neuron is the fundamental structural and functional unit of the brain.



**Visual Cortex**



**Cerebellum**



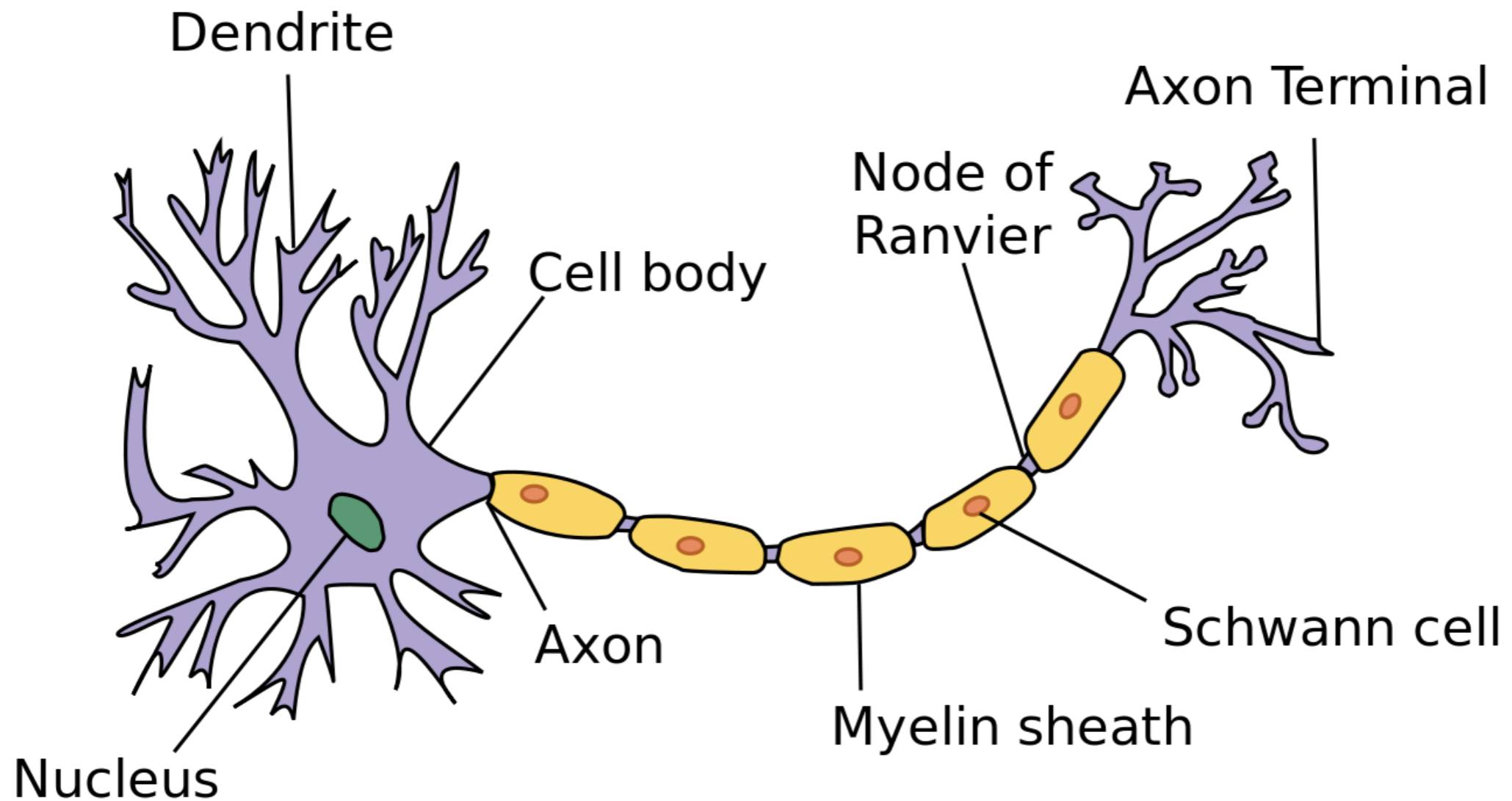
**Optic Tectum**

(Drawings by Ramón y Cajal, c. 1900)



# Neuronal physiology

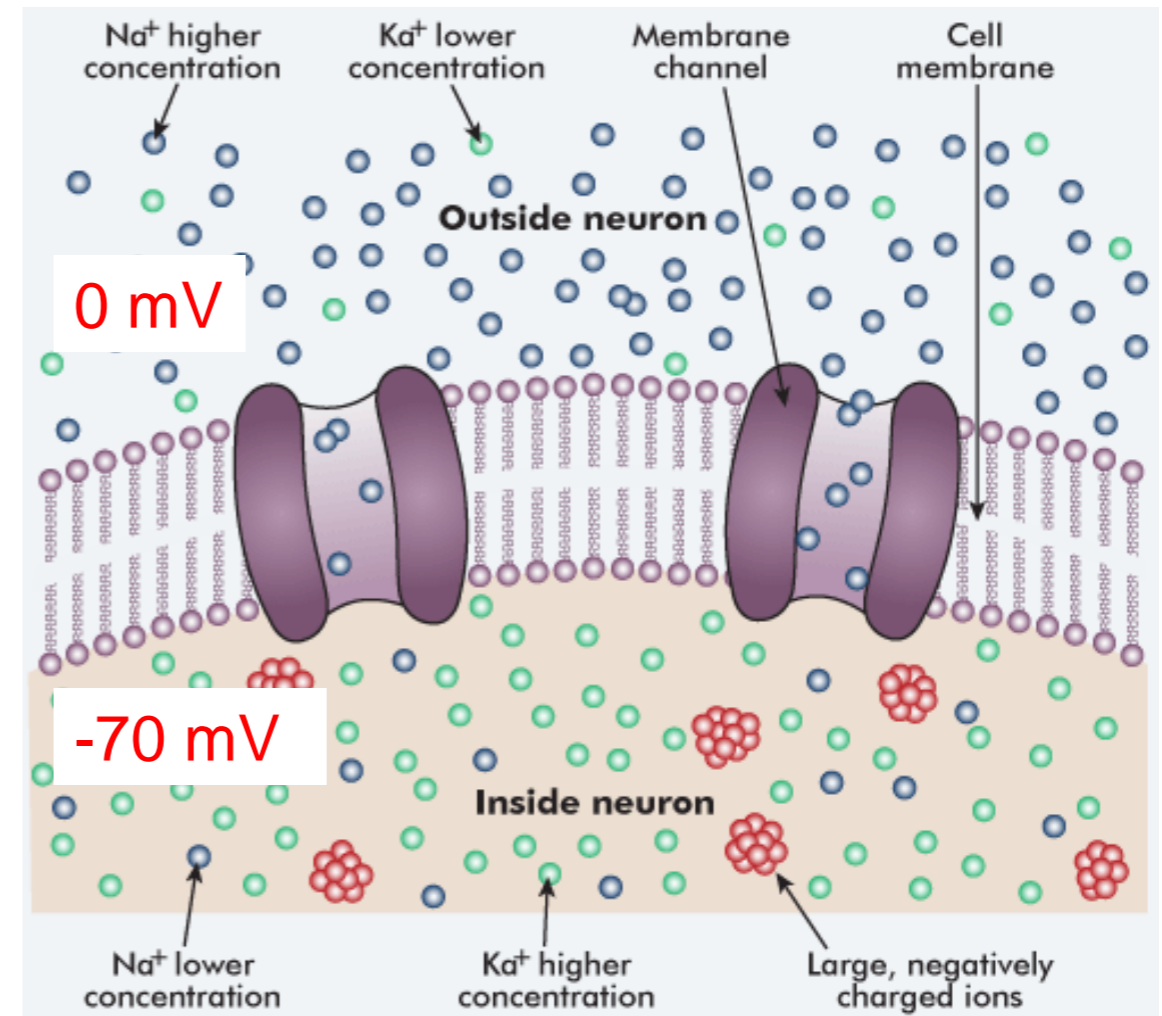
Many kind of neurons share the same cellular physiology.



# Neuronal physiology

Neuronal electrophysiological activity lies on the cell membrane.

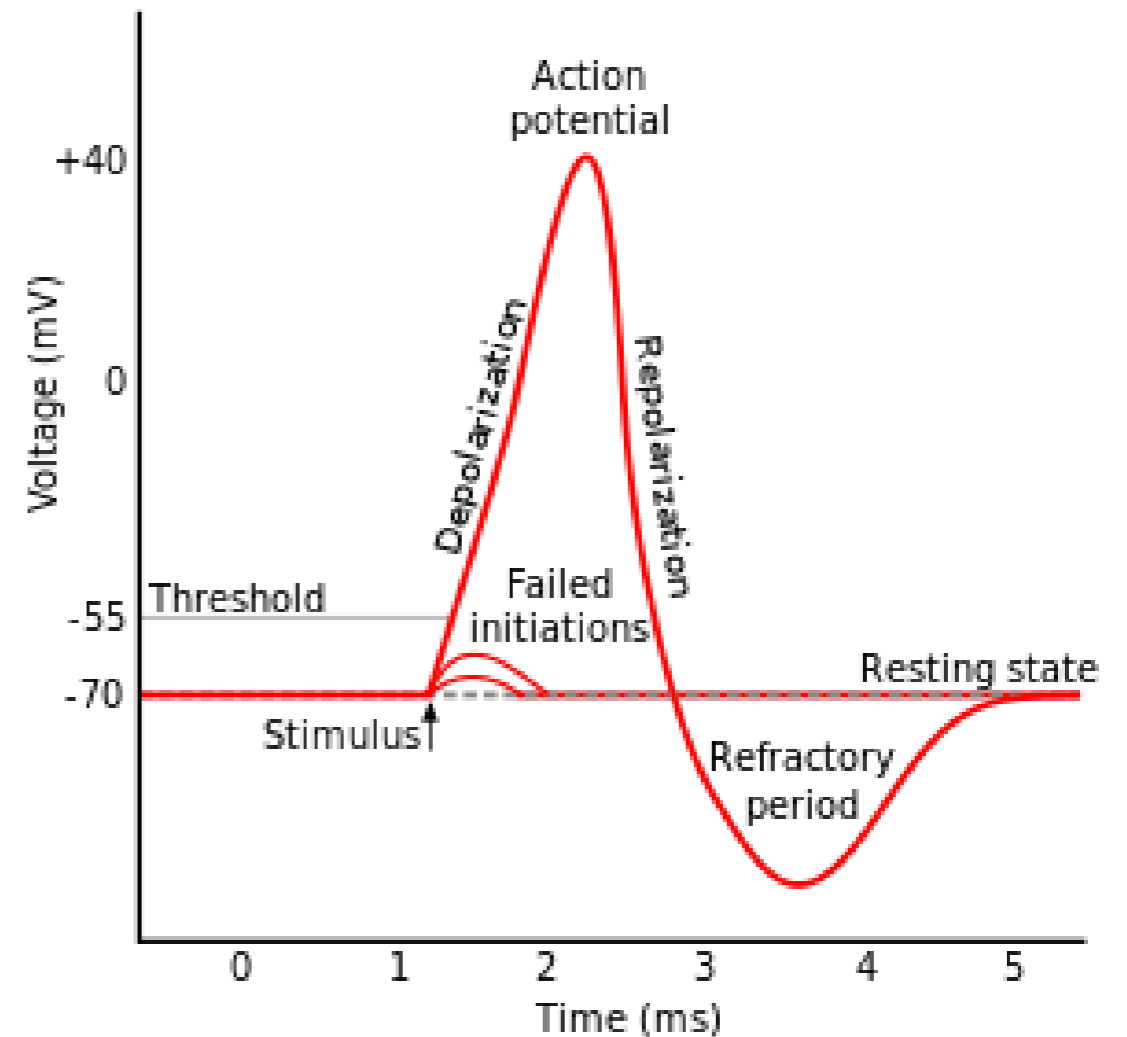
- Lipid bilayer, impermeable to charged ions.
- Ionic channels allow ions to flow in or out, selectively.
- The neuron maintains a **potential difference** across its membrane via the ionic pumps (expelling  $\text{Na}^+$  and allowing  $\text{K}^+$  in).
- When no external stimulus is present, we can refer to it as **resting potential**.



# Action potentials

The activity of a neuron (its “output”) is the **action potential** (or spike), generated by voltage-gated ionic channels.

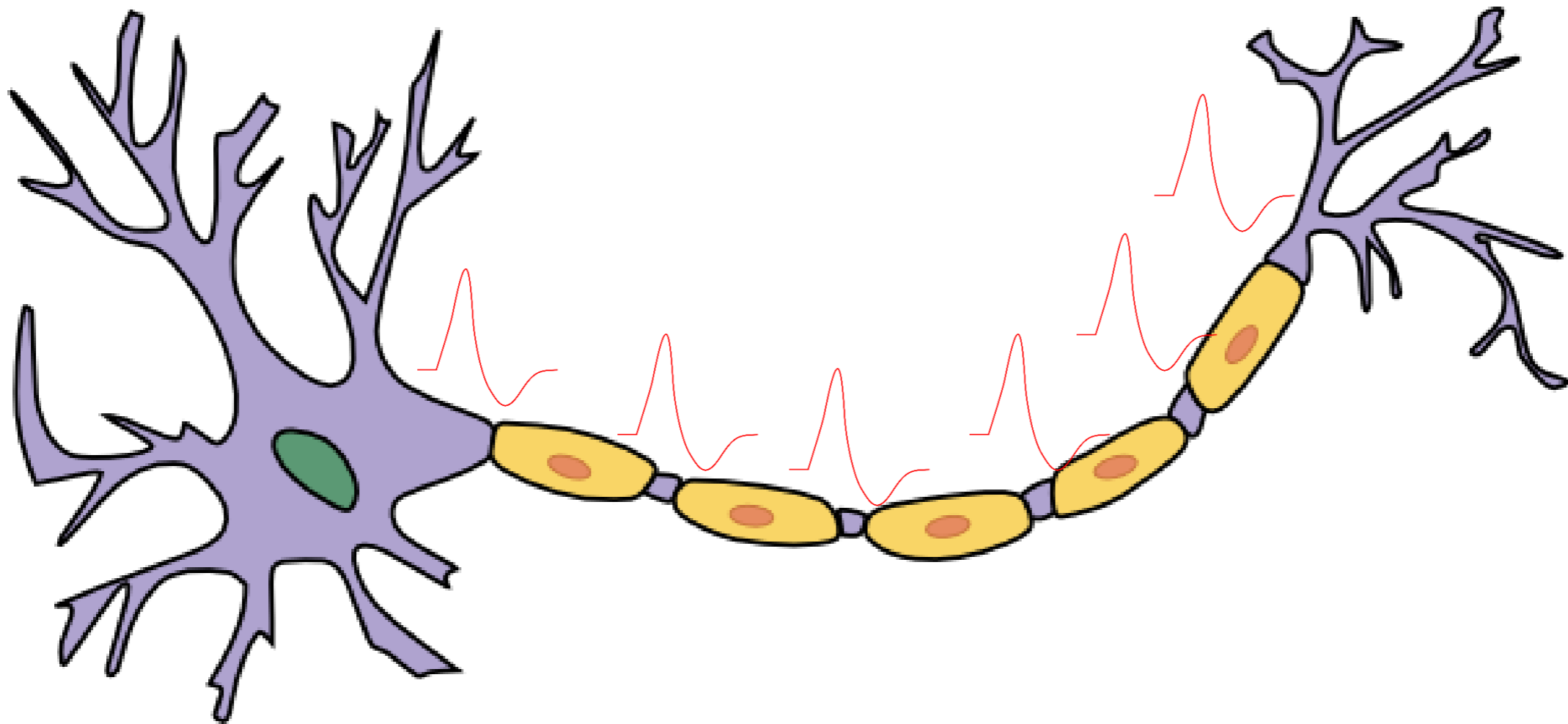
1. An external electric stimulus reach the membrane, depolarizing it.
2. Depolarization of the membrane opens  $\text{Na}^+$  channels ( $\rightarrow$  even more depolarization).
3. If membrane potential exceeds the **threshold potential**, an action potential occurs.
4. Afterwards, the membrane repolarize by expelling  $\text{K}^+$  ions and the neuron enters the refractory period.



# Action potentials

The action potential is transmitted through the axon towards other neurons.

Each non-myelinated section (node of Ranvier) replicates the spike.

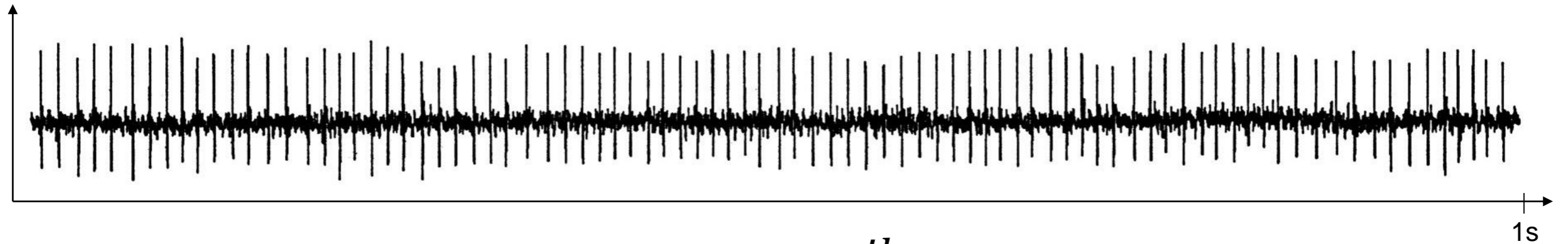


Propagation speed ranges from 1 to 100 m/s.



# Action potentials

The activity of a neuron is measured by computing its **firing rate**, expressed as the mean number of spikes per second.

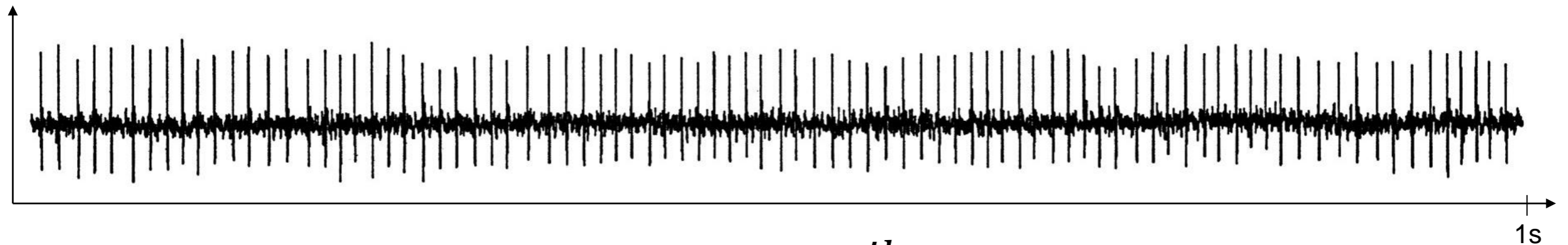


$$rate = \frac{n. spikes}{time}$$



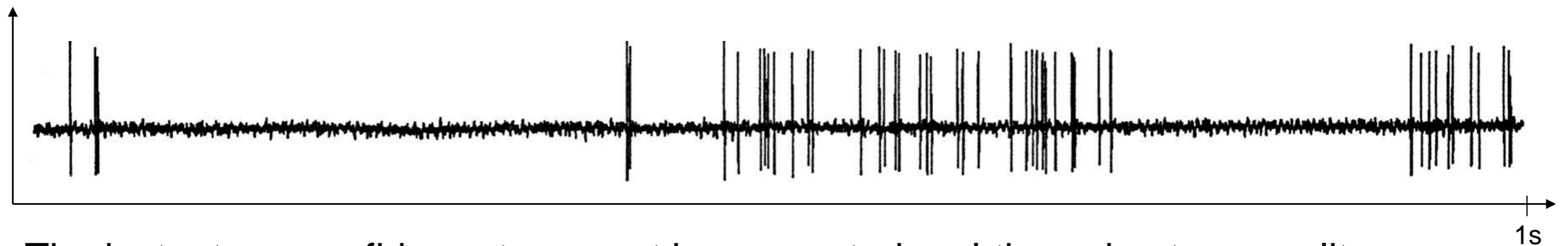
# Action potentials

The activity of a neuron is measured by computing its **firing rate**, expressed as the mean number of spikes per second.



$$rate = \frac{n. spikes}{time}$$

It is not always an easy task!



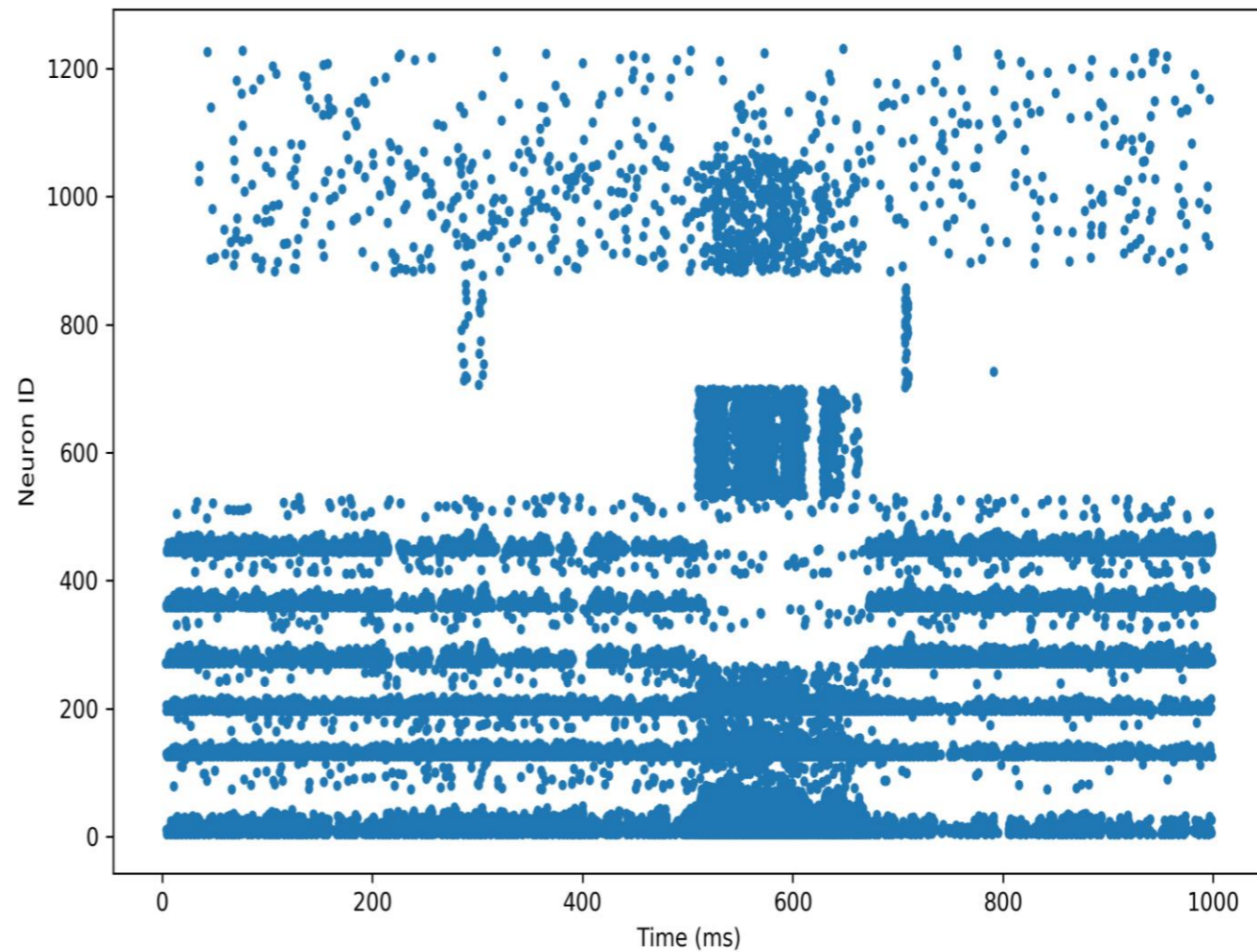
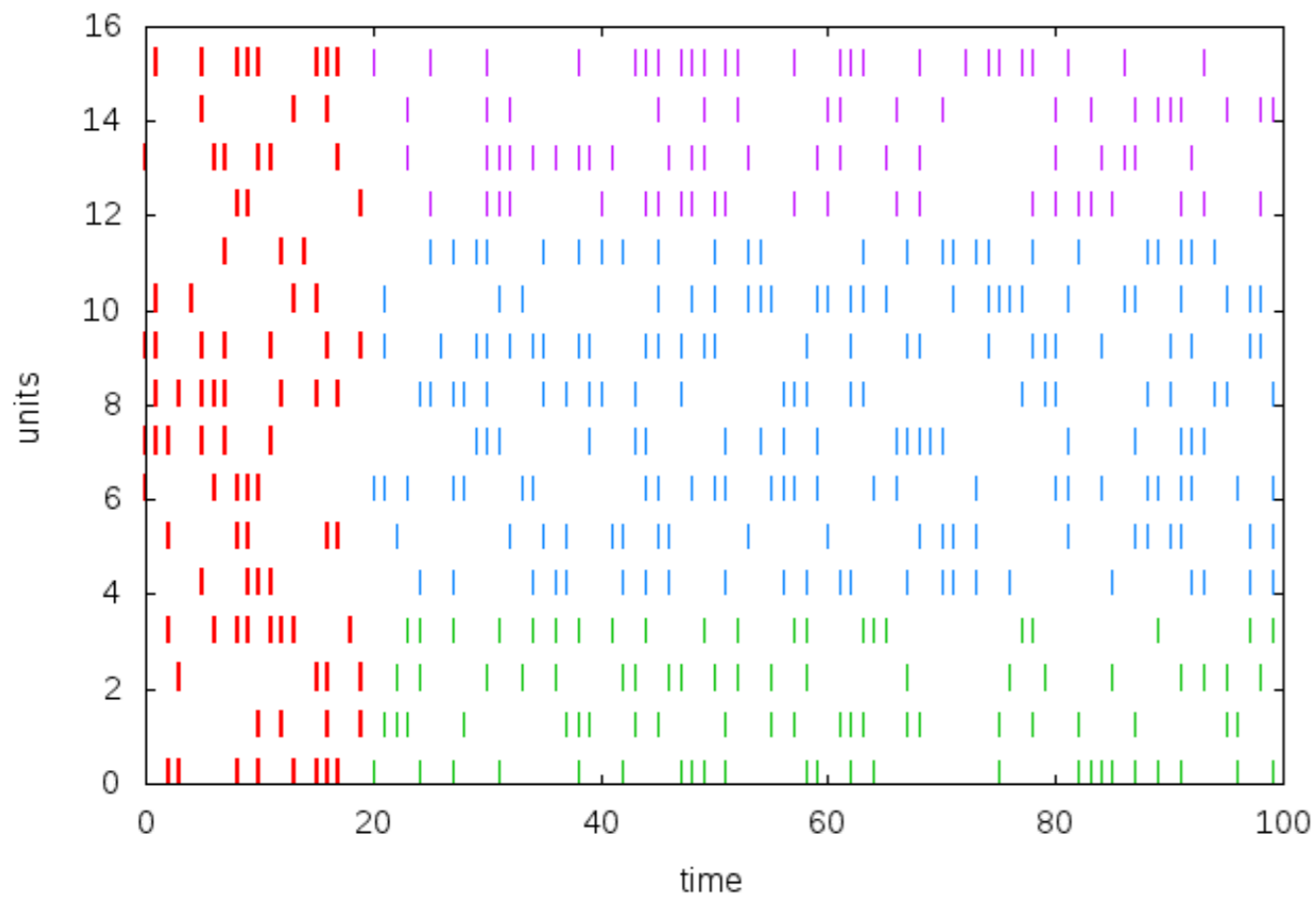
The instantaneous firing rate cannot be computed real-time, due to causality.



# Action potentials

Usually, we are interested in looking at the spike events, instead of the membrane potential, and for a high number of neurons (a population).

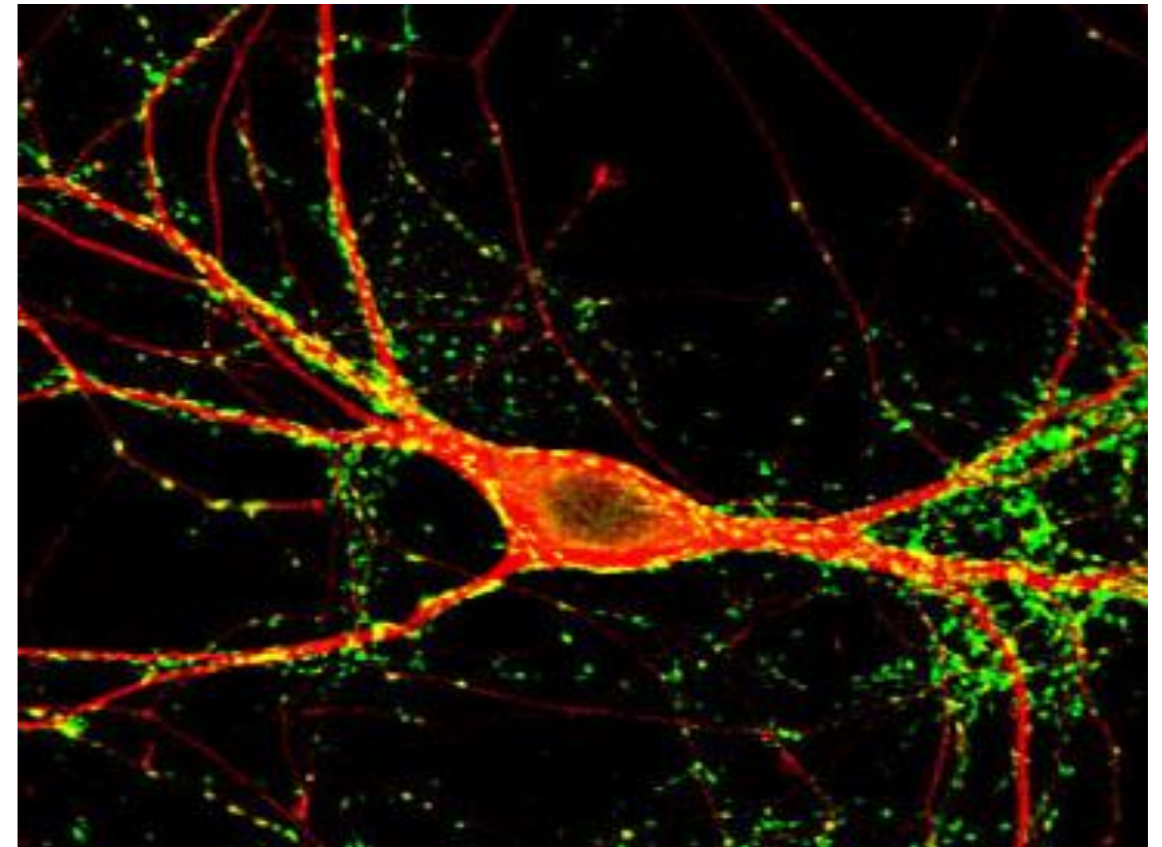
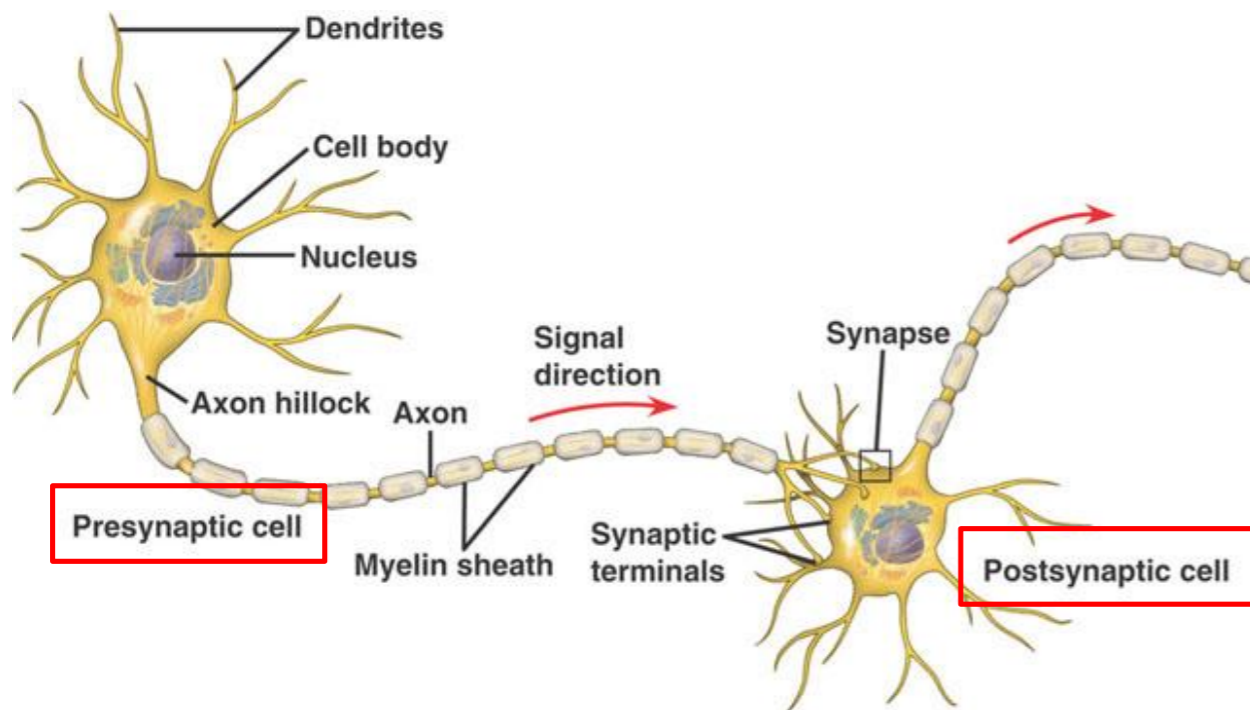
We can do so with raster plots.





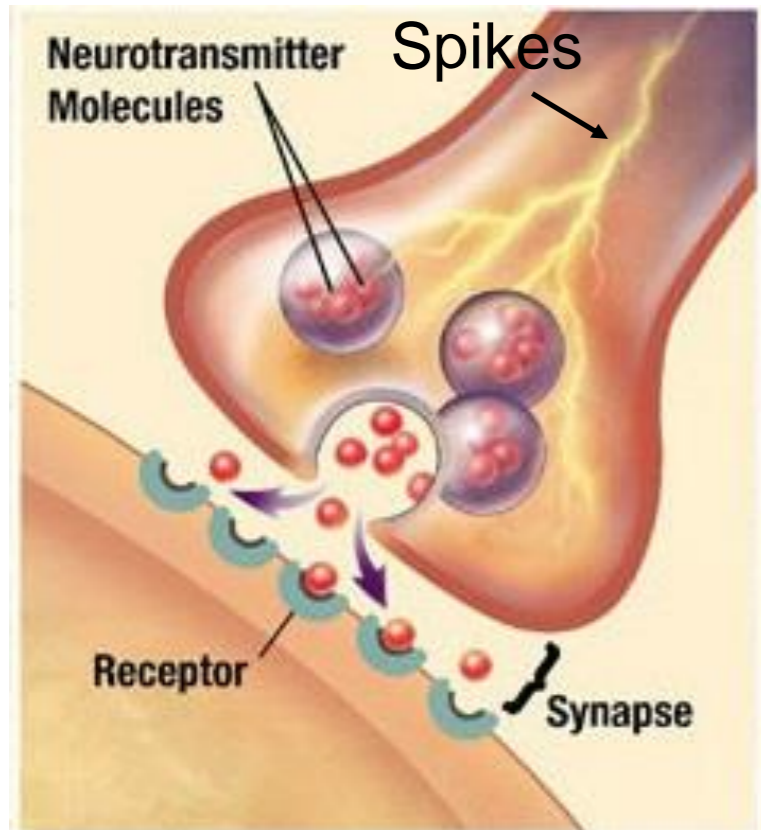
# Synapses

Axons and dendrites are connected through **synapses**. Each neuron has roughly 1000-10000 synapses.

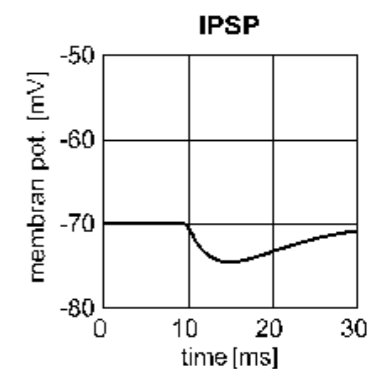
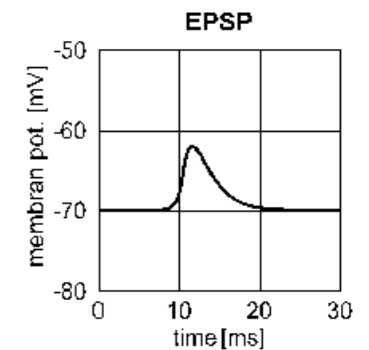


# Synapses

Synapses can be chemical or electrical, excitatory or inhibitory:



- a chemical **excitatory** synapse releases Glutamate  $\rightarrow$  opening of ion channels for  $\text{Na}^+$  influx  $\rightarrow$  membrane depolarization (membrane potential *increases*);
- a chemical **inhibitory** synapse releases GABA neurotransmitter  $\rightarrow$   $\text{K}^+$  leaves cell through ion channels  $\rightarrow$  membrane hyperpolarization (membrane potential *decreases*).

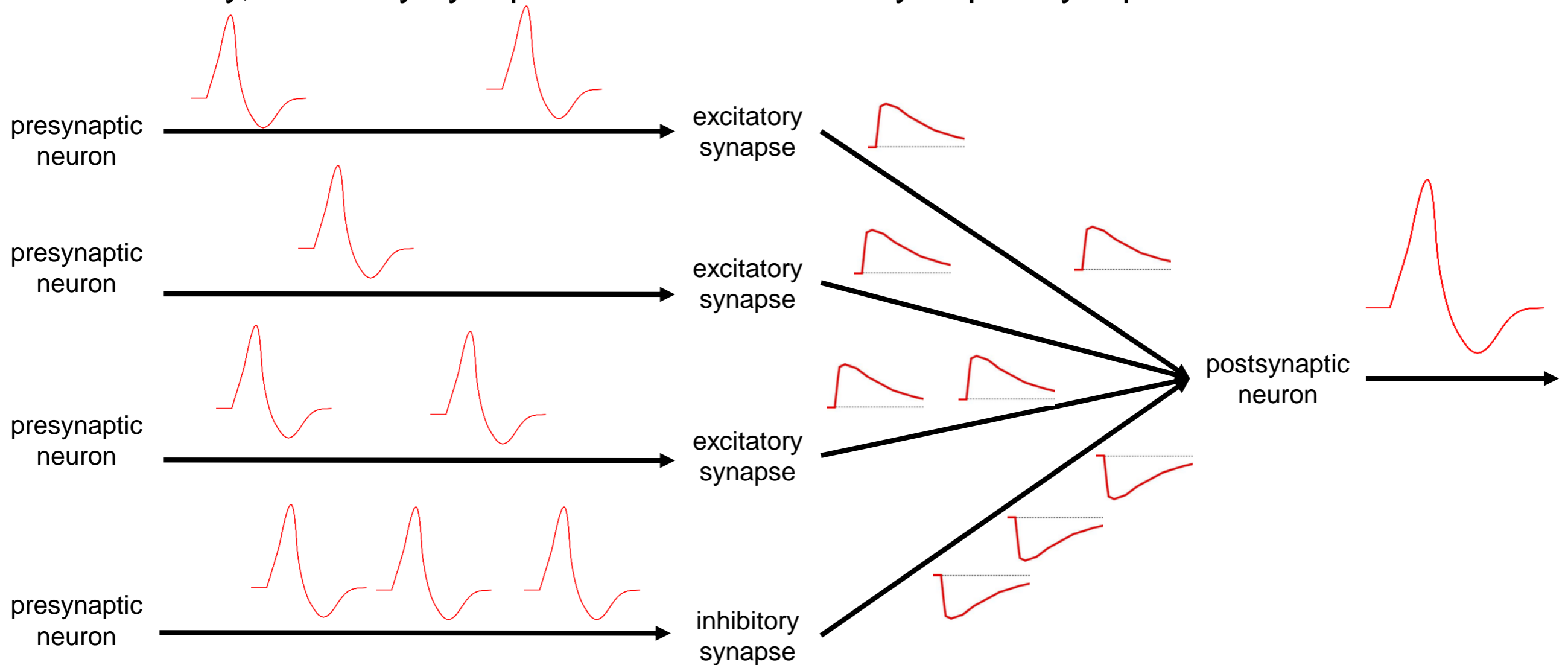


Every synapse, once reached by an action potential, generates a *postsynaptic current* (PSC) which turns in a *postsynaptic potential* (PSP).



# Synapses and action potentials

Each spike coming from presynaptic neurons and activating excitatory synapses contributes to the generation of an action potential in the postsynaptic neuron. Conversely, inhibitory synapses reduce the activity of postsynaptic neurons.



# Synaptic plasticity

Synapses are the basis for memory and **learning**.

If neuron A repeatedly takes part in making neuron B spike, then the synapse from a to B is strengthened and vice versa. This leads to two phenomena:

EPSP Before



Difference in spike times



EPSP After



Spike from A

Long term potentiation (LTP)



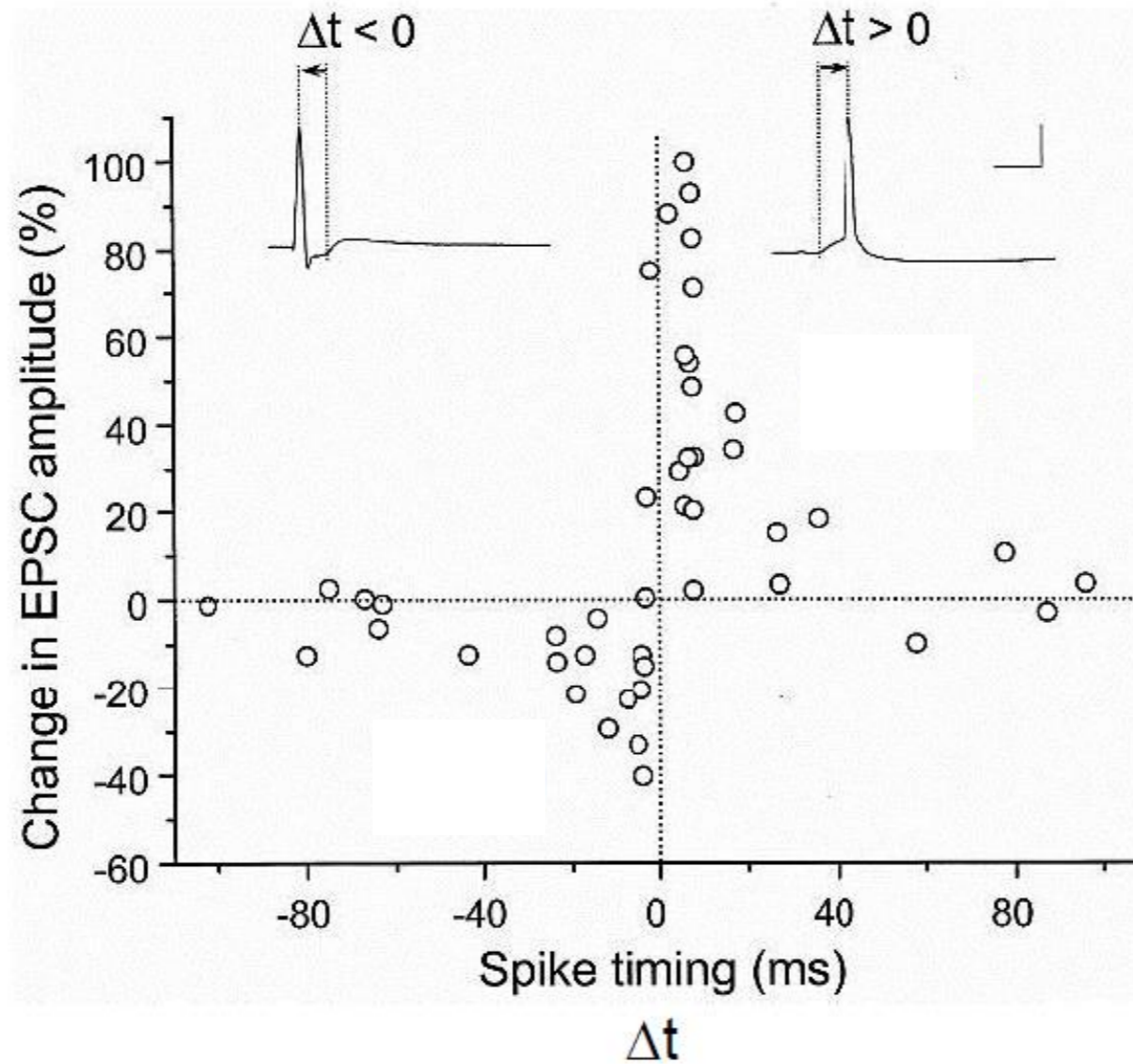
Spike from A

Long term depression (LTD)



# Synaptic plasticity

This adaptation mechanism depends on the timing of the EPSP and the action potential. Thus, it is called **Spike-Timing-Dependent Plasticity (STDP)**.



# Neural information processing

What kind of information can a neuron process?

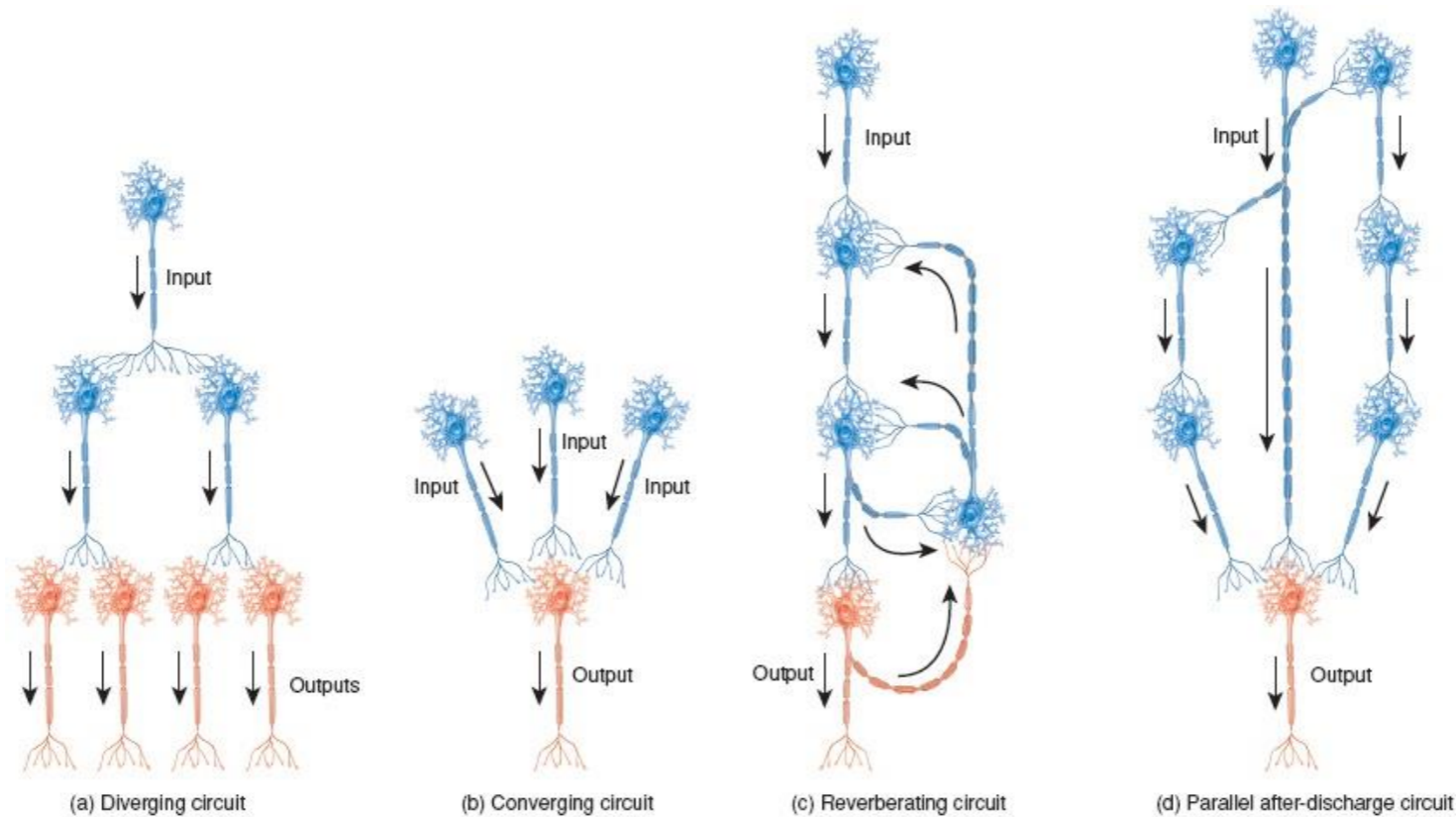


# Neural information processing

What kind of information can a neuron process?

**None!** (by himself)

Information is processed by means of the network topology and synaptic properties.

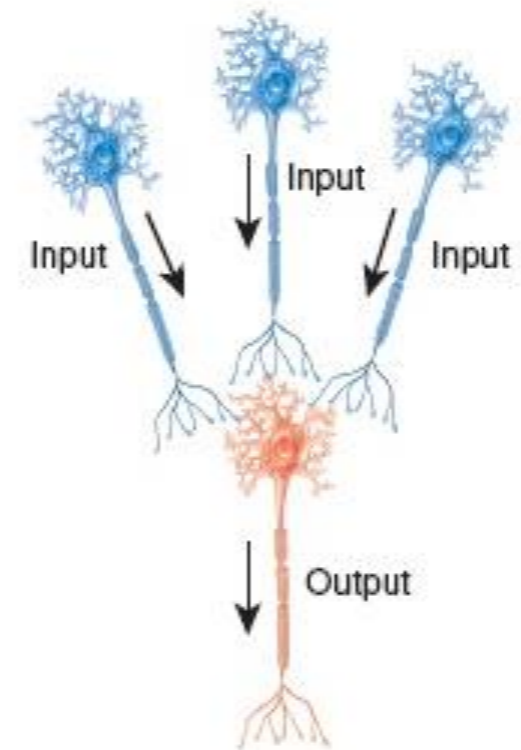


# Receptive fields

A simple way of processing information is the **receptive field** topology.

Each receptive field is made up of several input neurons and one output neuron that modulates the combination of their responses.

Receptive fields have been identified in the human brain to encode sensory information (auditory system, somatosensory system, visual system).

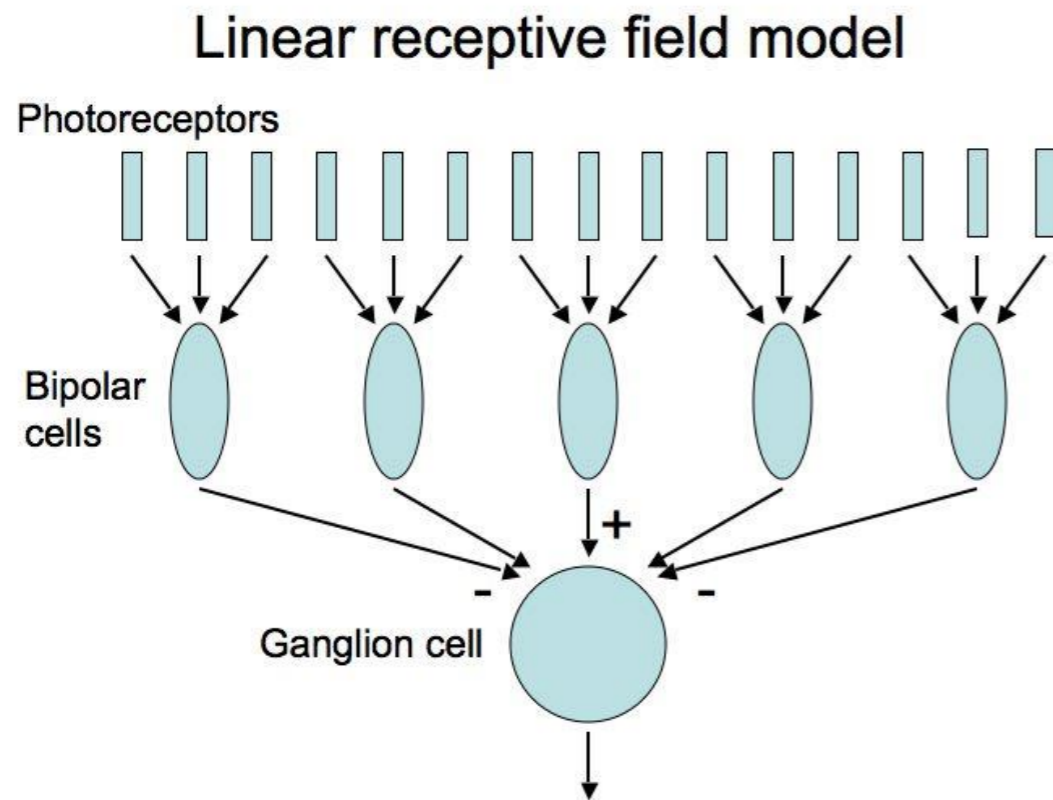




# Receptive fields

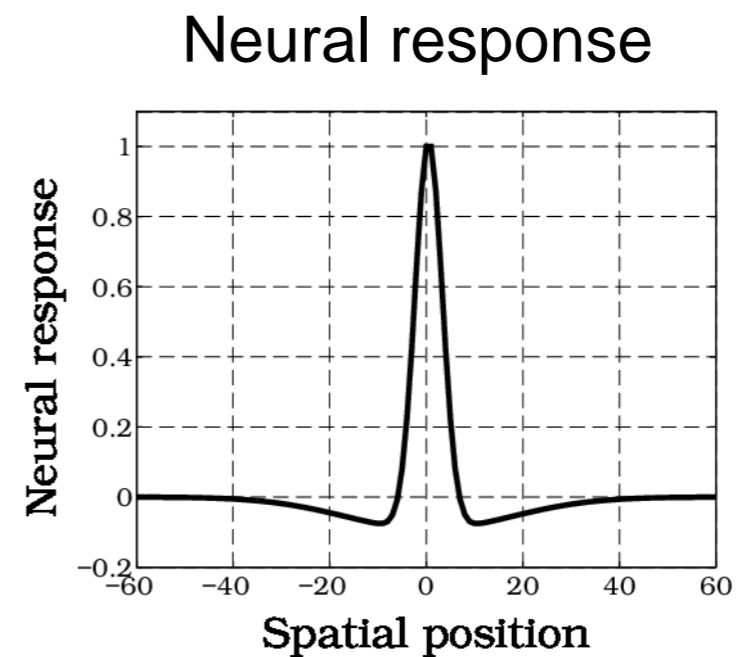
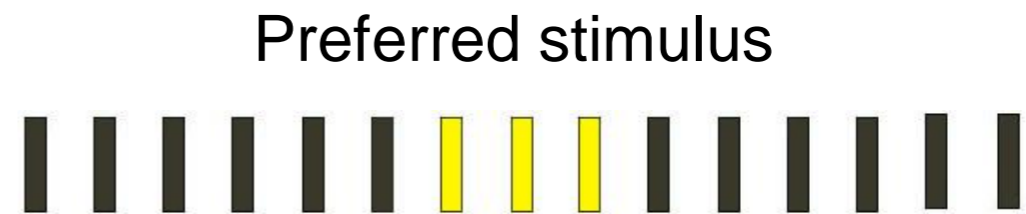
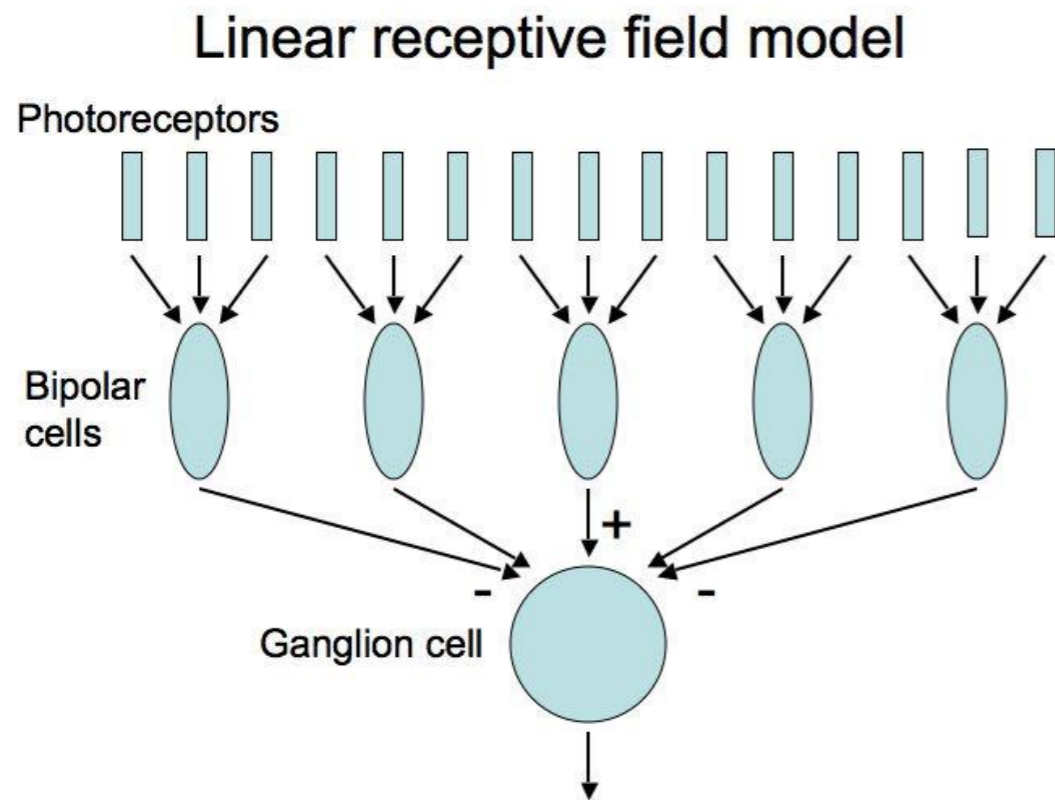
The retinal circuit implements receptive fields to process the image.

Preferred stimulus?



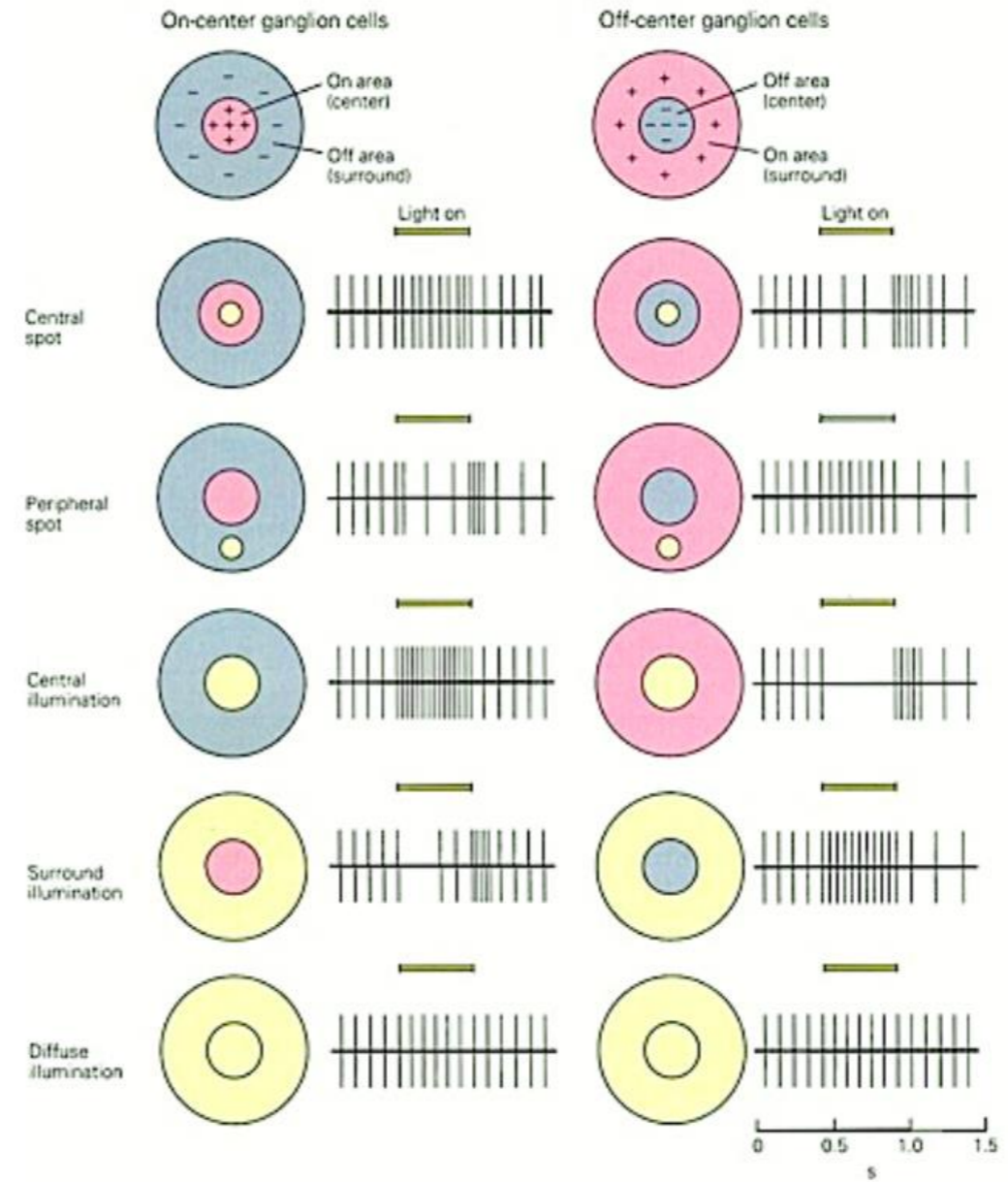
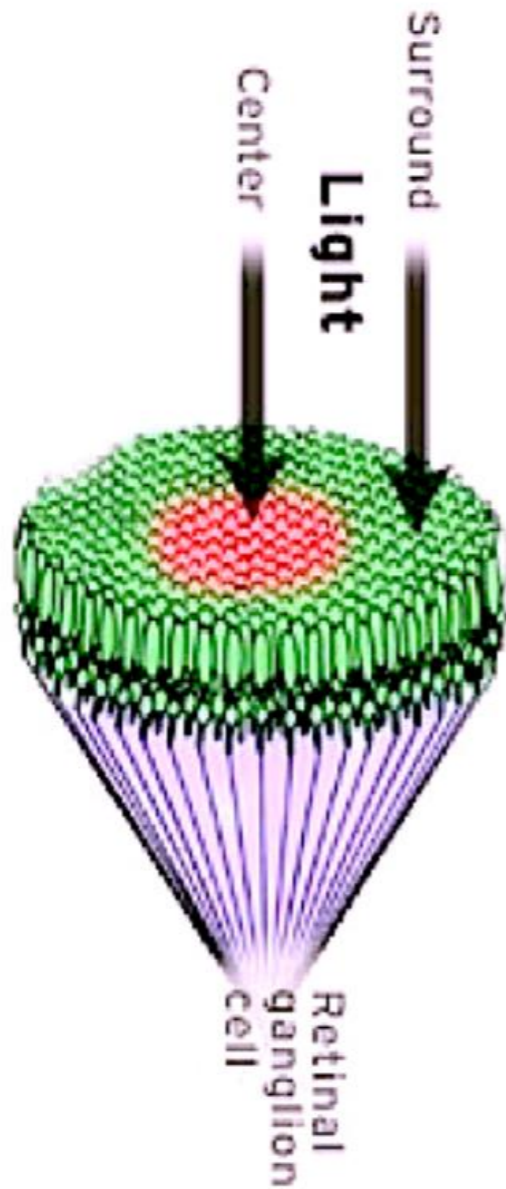
# Receptive fields

The retinal circuit implements receptive fields to process the image.



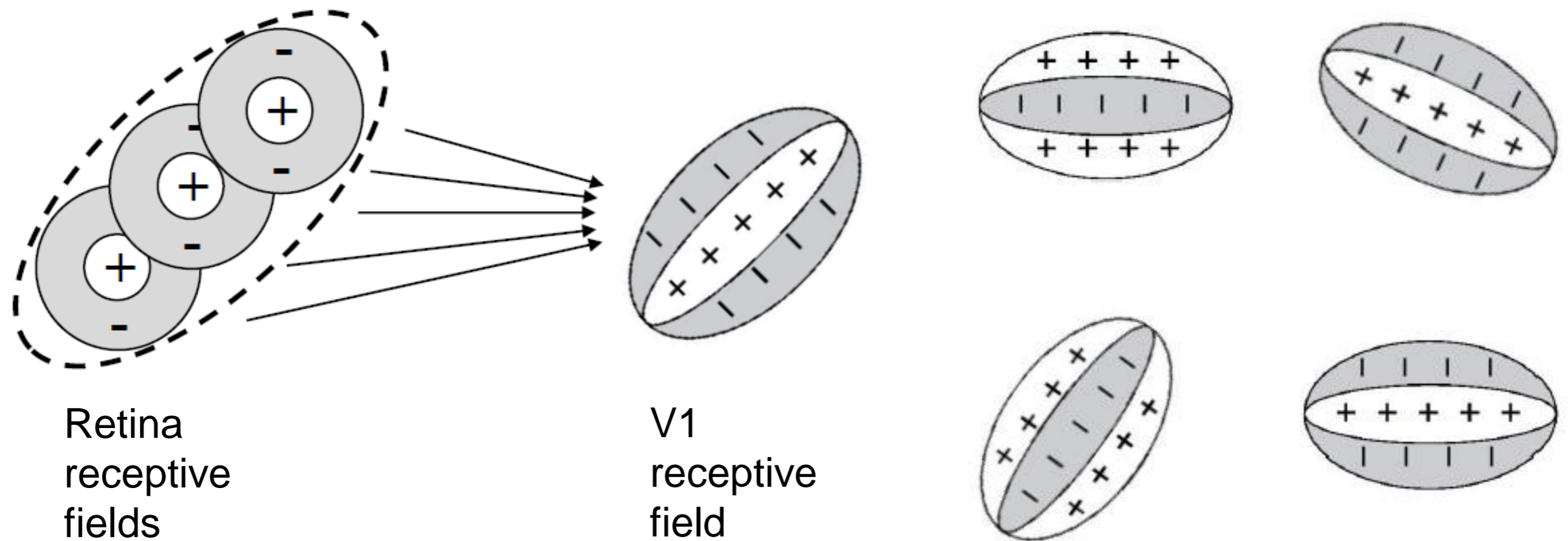
# Receptive fields

The retinal circuit implements receptive fields to process the image.



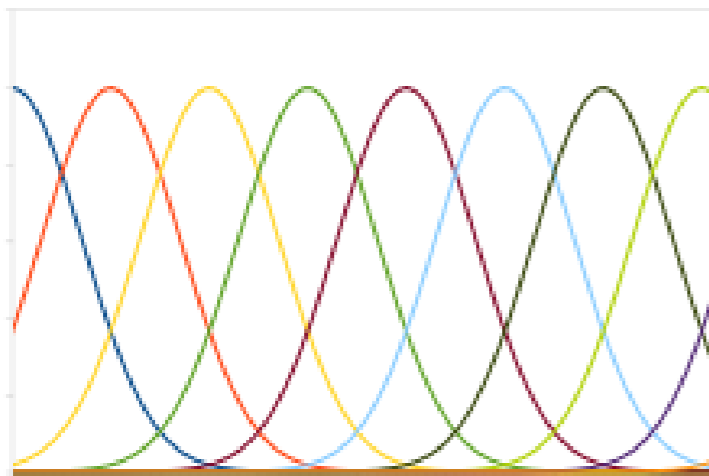
# Receptive fields

Receptive fields from the retina are in turn used to create oriented receptive fields in the visual cortex.

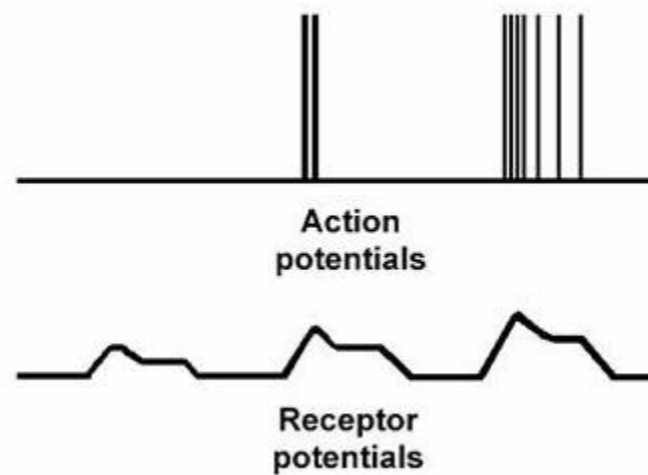


# Neural coding

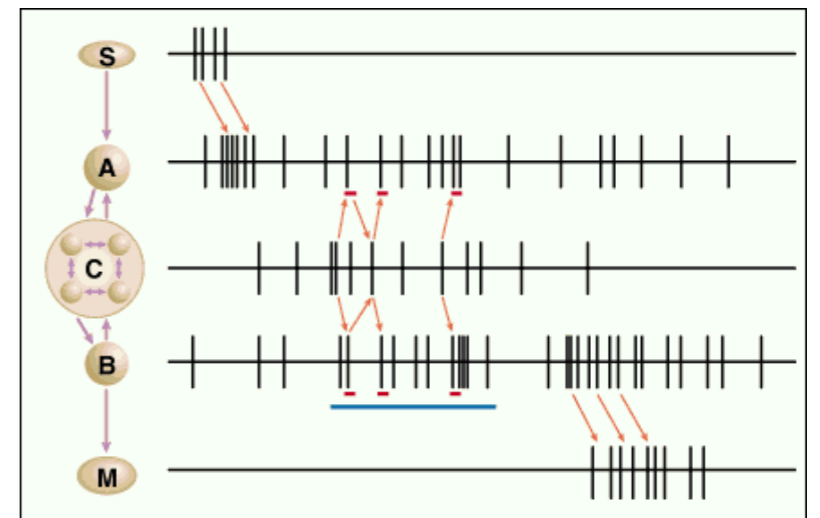
Each sensory input has its own dedicated brain areas that encode the information received. Different types of encoding are being used in the brain. The most well-understood are the following three:



population coding



rate coding

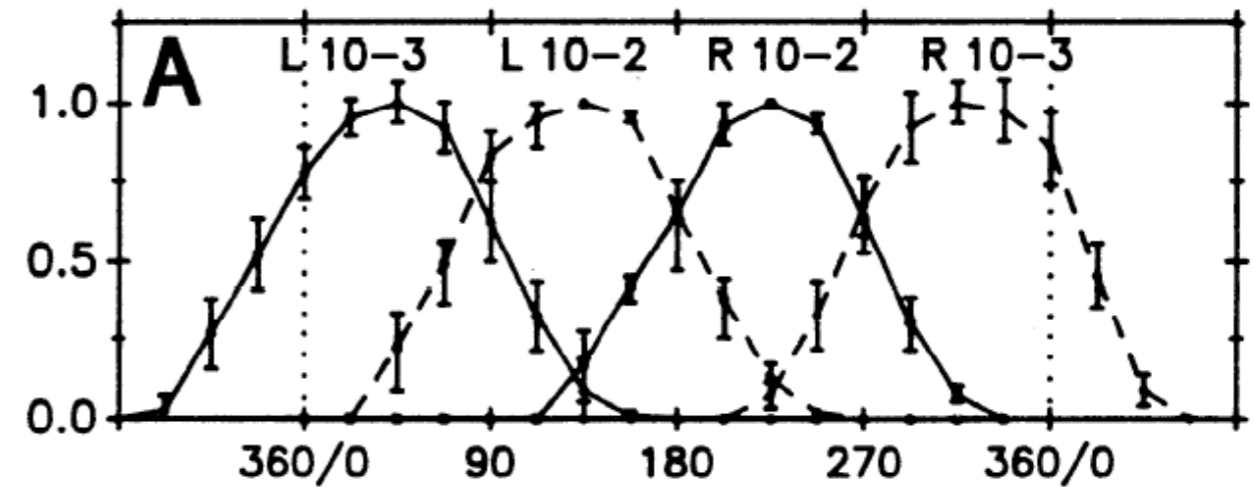
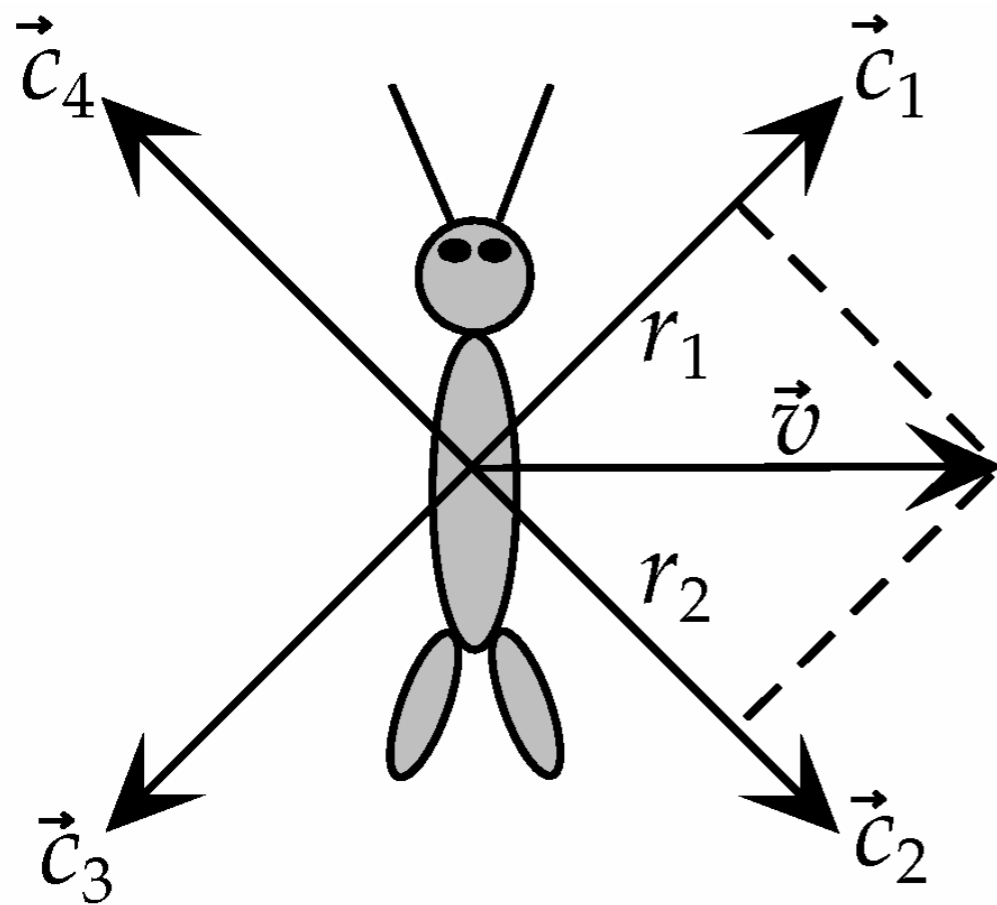


temporal coding



# Neural coding

In **population coding**, there is a population of neurons that respond differently to different values of the same sensory information. E.g. cricket cercal cells, some visual areas in the human brain.



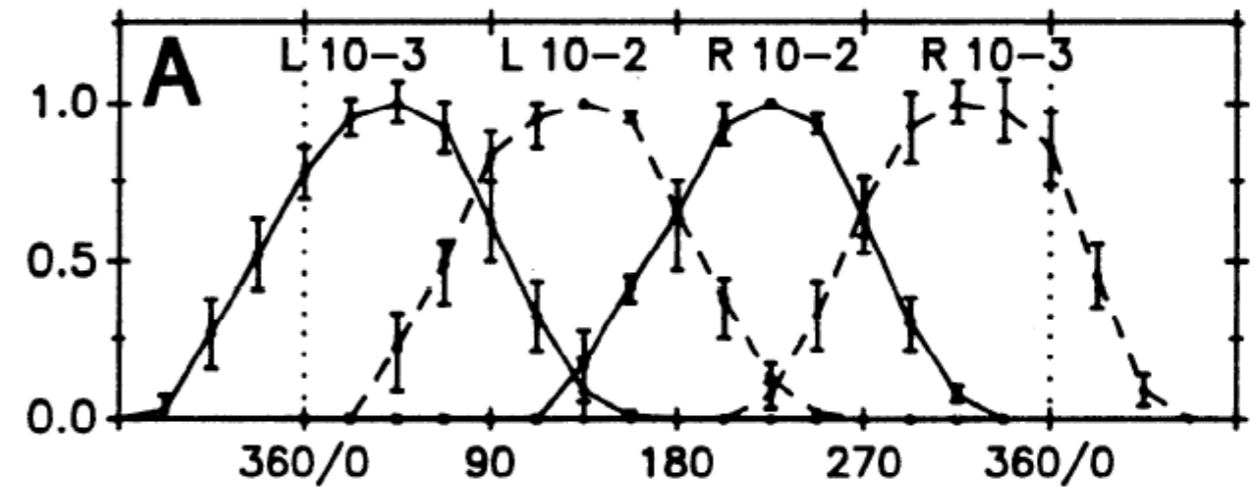
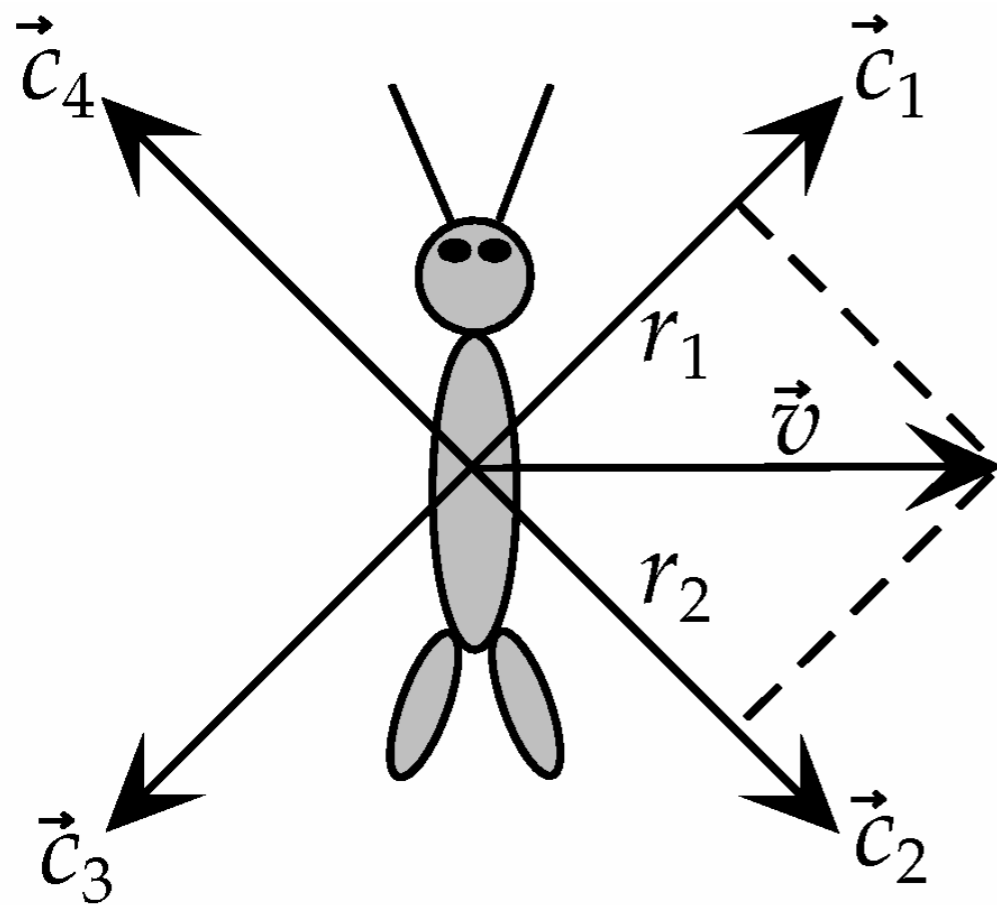
$$r_i = \vec{v} \cdot \vec{c}_i$$

The resulting value  $r$  is called the **population vector**.



# Neural coding

In **population coding**, there is a population of neurons that respond differently to different values of the same sensory information. E.g. cricket cercal cells, some visual areas in the human brain.



$$r_i = \vec{v} \cdot \vec{c}_i$$

The resulting value  $r$  is called the **population vector**.

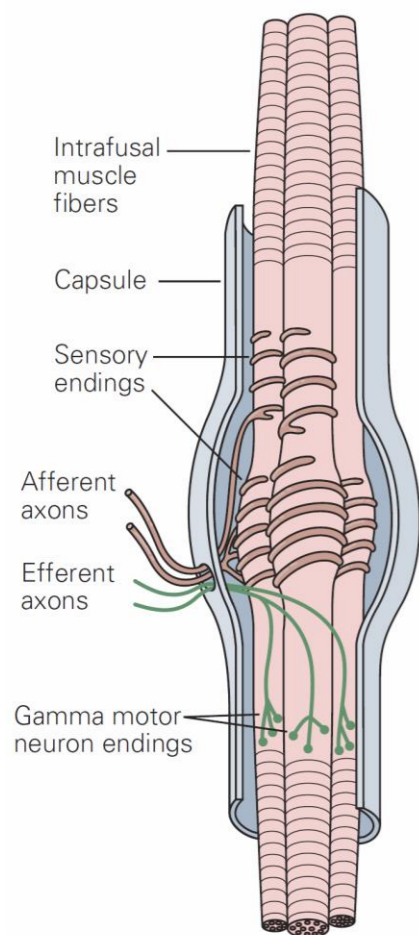
Is the representation efficient?  
Aren't  $c_1$  and  $c_2$  enough?



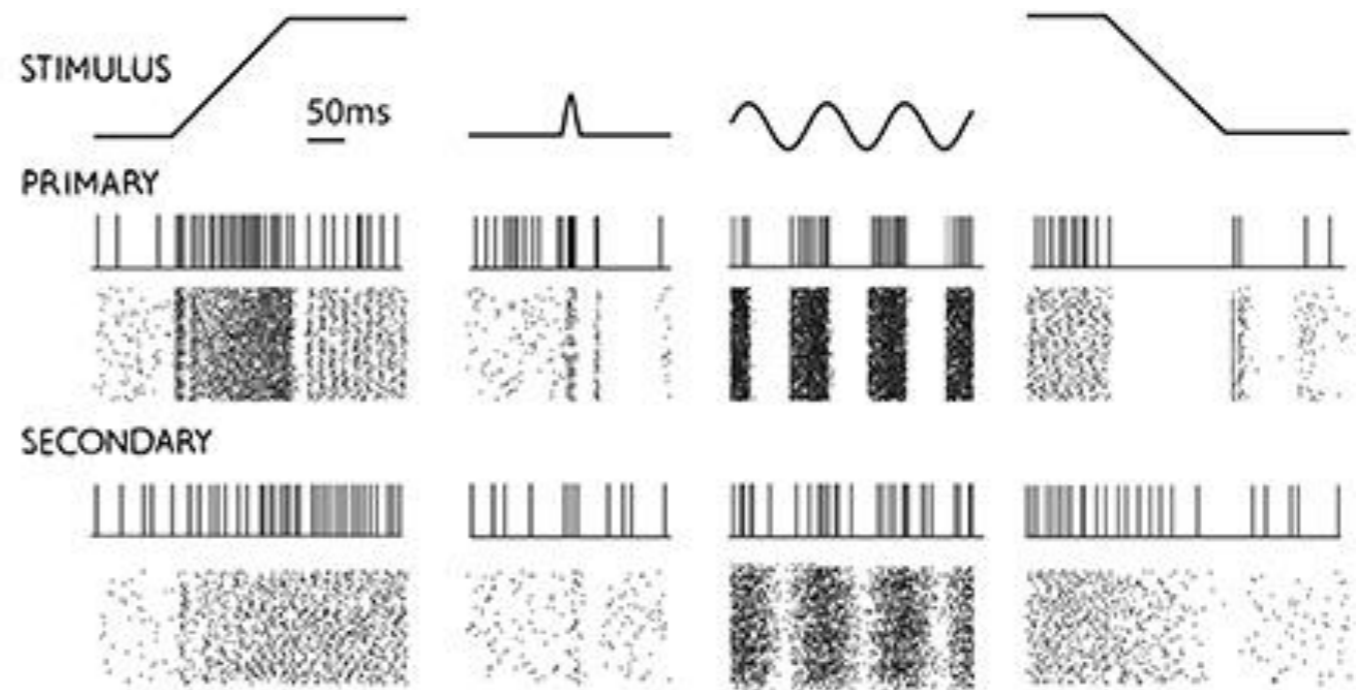
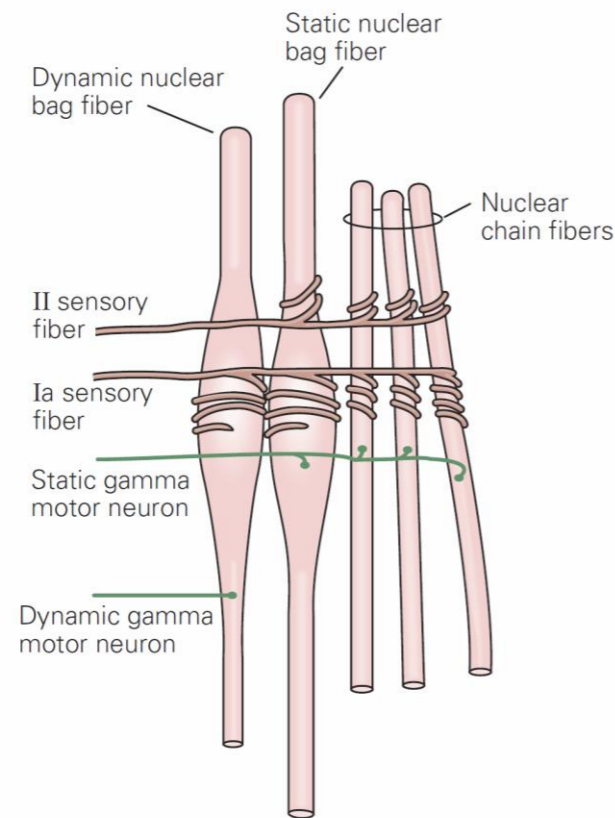
# Neural coding

In **rate coding**, all the information is encoded by directly translating it into firing rates. Thus, all neurons in the same population respond in the same manner to the same stimulus. This is common in many sensory afferents, e.g. mammalian muscle spindles.

A Muscle spindle



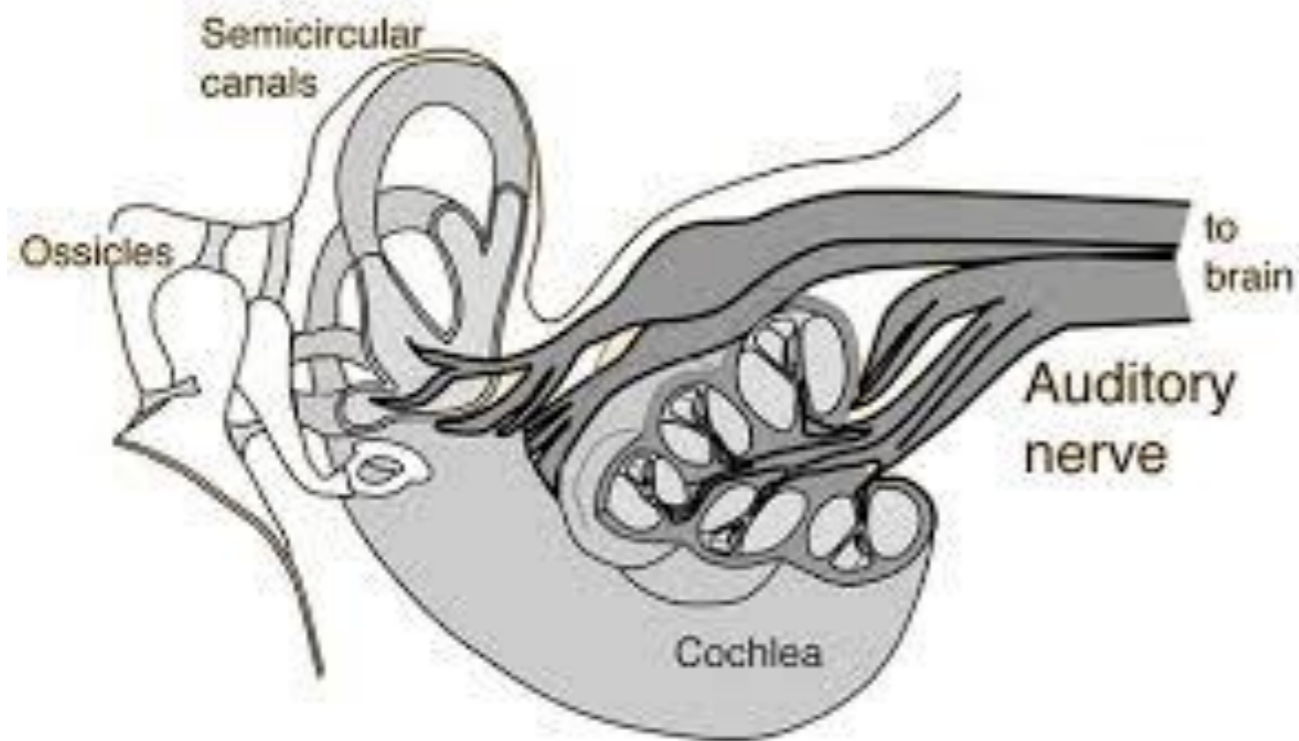
B Intrafusal fibers of the muscle spindle



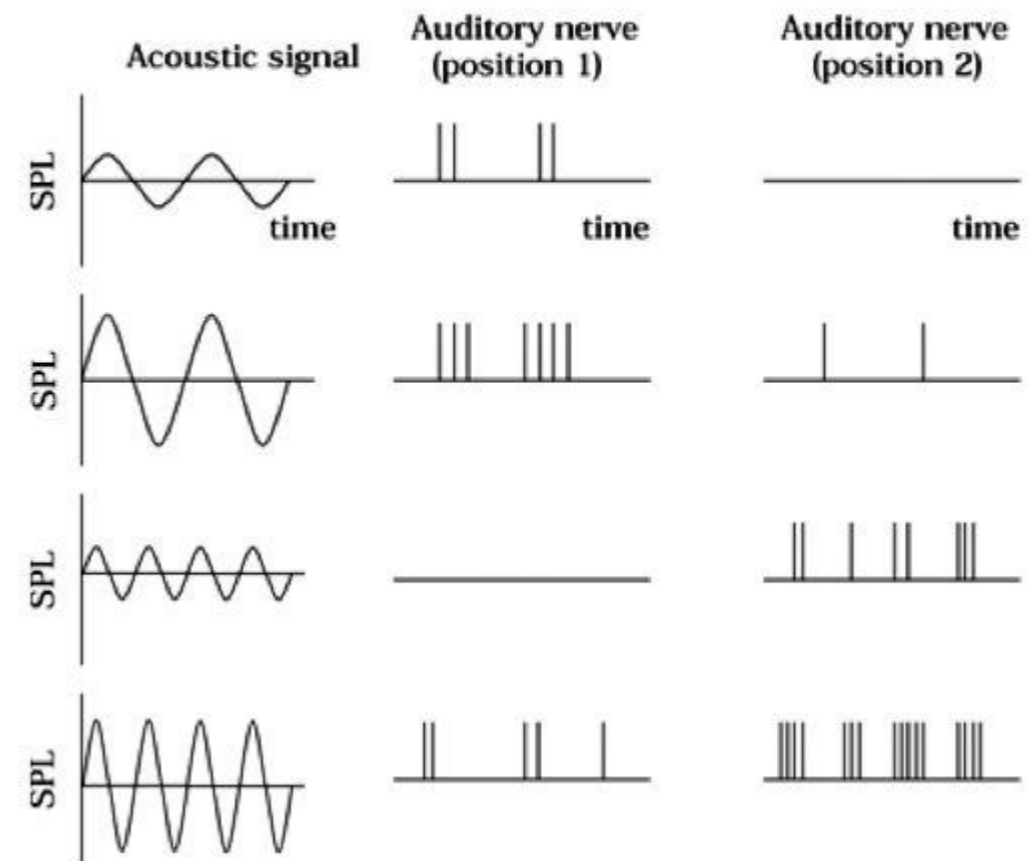


# Neural coding

A more complex encoding mechanism is **temporal coding**, where absolute or relative spike times are used. There are evidence for this kind of encoding in the auditory and gustative systems.



## Pitch and loudness



# Outline

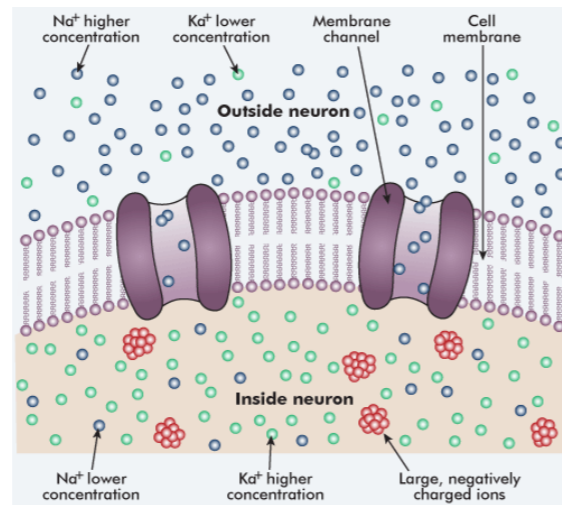
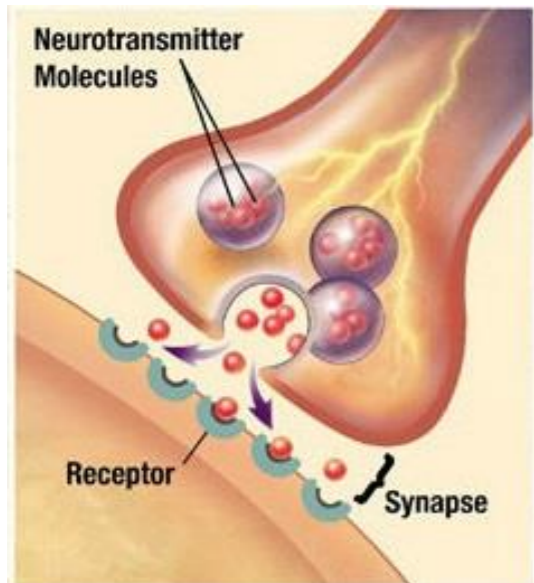
1. Introduction
2. Fundamentals of neuroscience
- 3. Simulating the brain**
4. Software and hardware simulations
5. Robotic applications



# Neuron abstractions

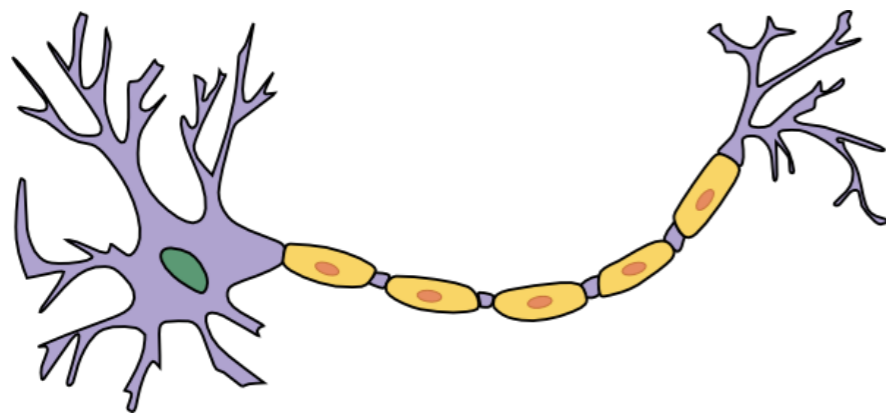
In order to simulate the behaviour of neural circuits we have to model the neuron dynamics.

Thus, we have to translate neurophysiologic properties into equations that we can implement.



## Abstract neuron models

- Rate-based
- Point neuron
- Detailed neuron



# Detailed neural abstraction

In these kind of models every aspect of the cell morphology is taken into account: diameter of the soma, length of the axon, position of synapses on the dendrites, distribution of ionic channels, neurotransmitter types, etc...

## Pros:

- very accurate
- can model any aspect of neural activity

## Cons:

- much knowledge is needed to model networks
- simulation times are high



Some detailed neural simulators exist, i.e. NEURON ([www.neuron.yale.edu/neuron](http://www.neuron.yale.edu/neuron)).

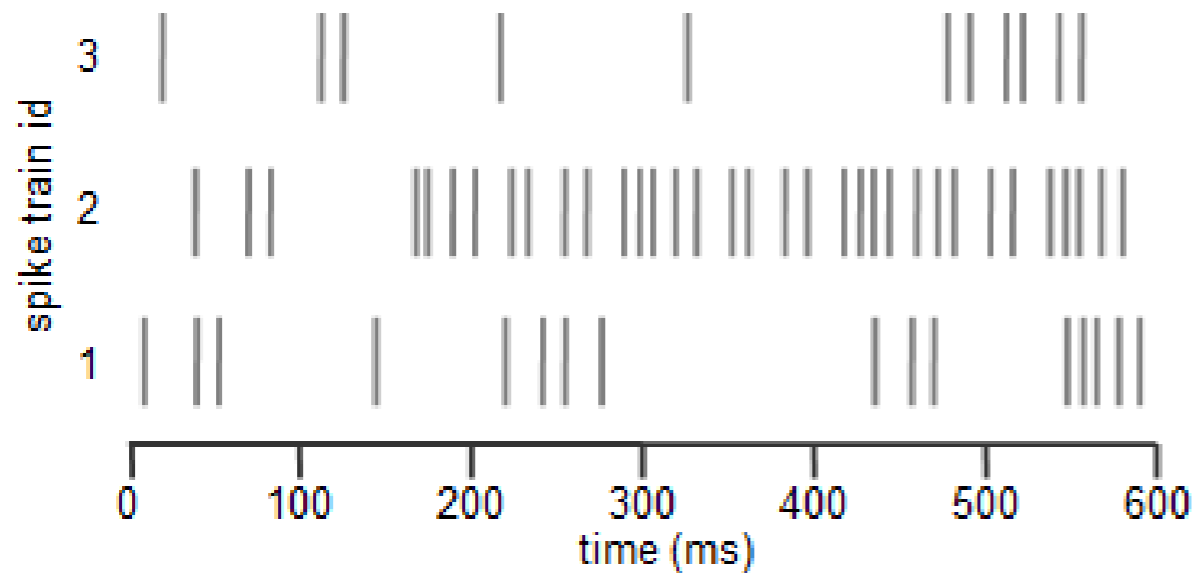
Too little abstraction!



# Rate-based abstractions

Each neuron produces spikes with a mean firing rate (in a time interval).

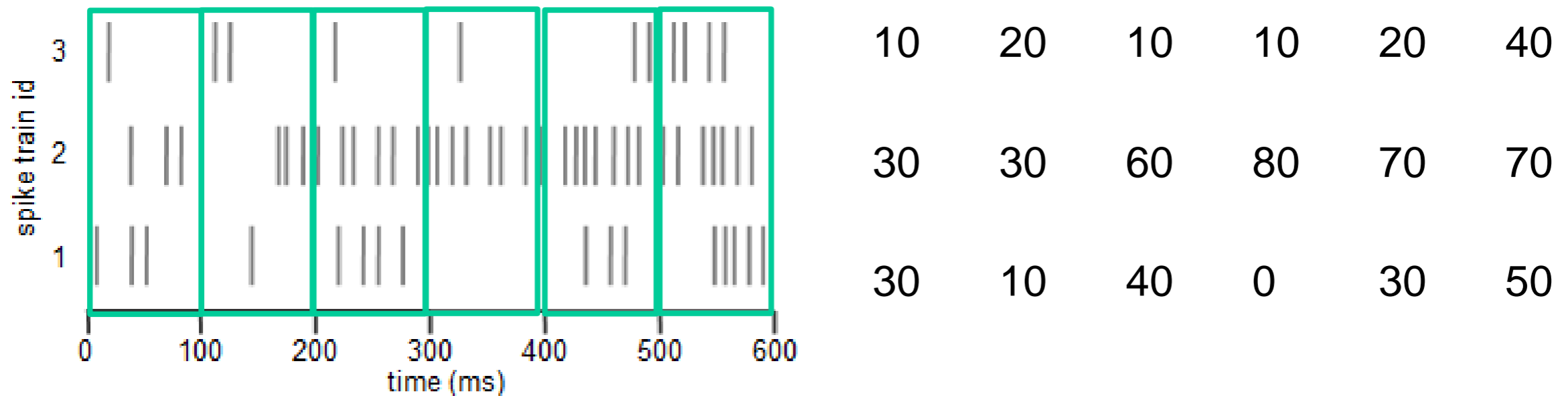
We can sample the firing rate by dividing spikes into bags:



# Rate-based abstractions

Each neuron produces spikes with a mean firing rate (in a time interval).

We can sample the firing rate by dividing spikes into bags:



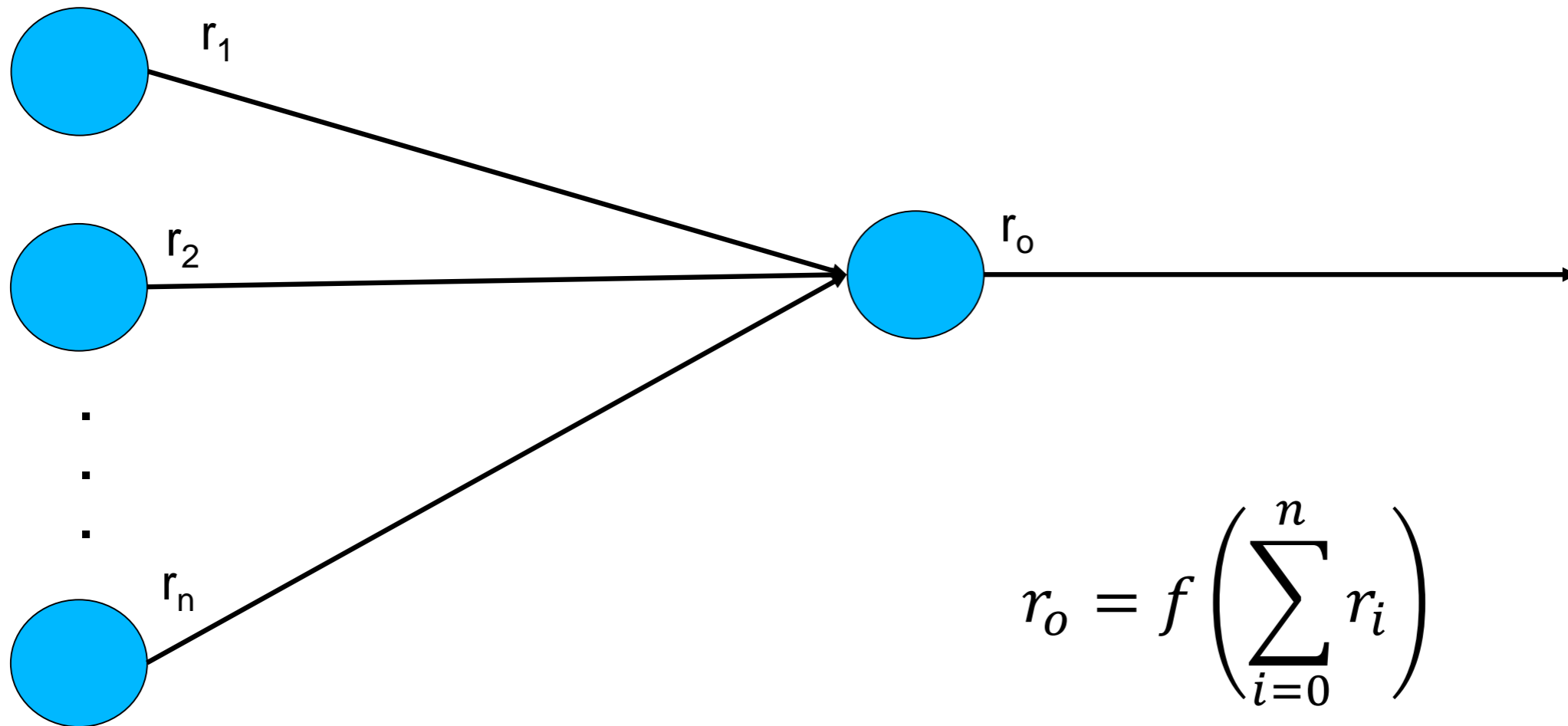
By doing so, we are:

- discretizing time
- forgetting about single action potential events



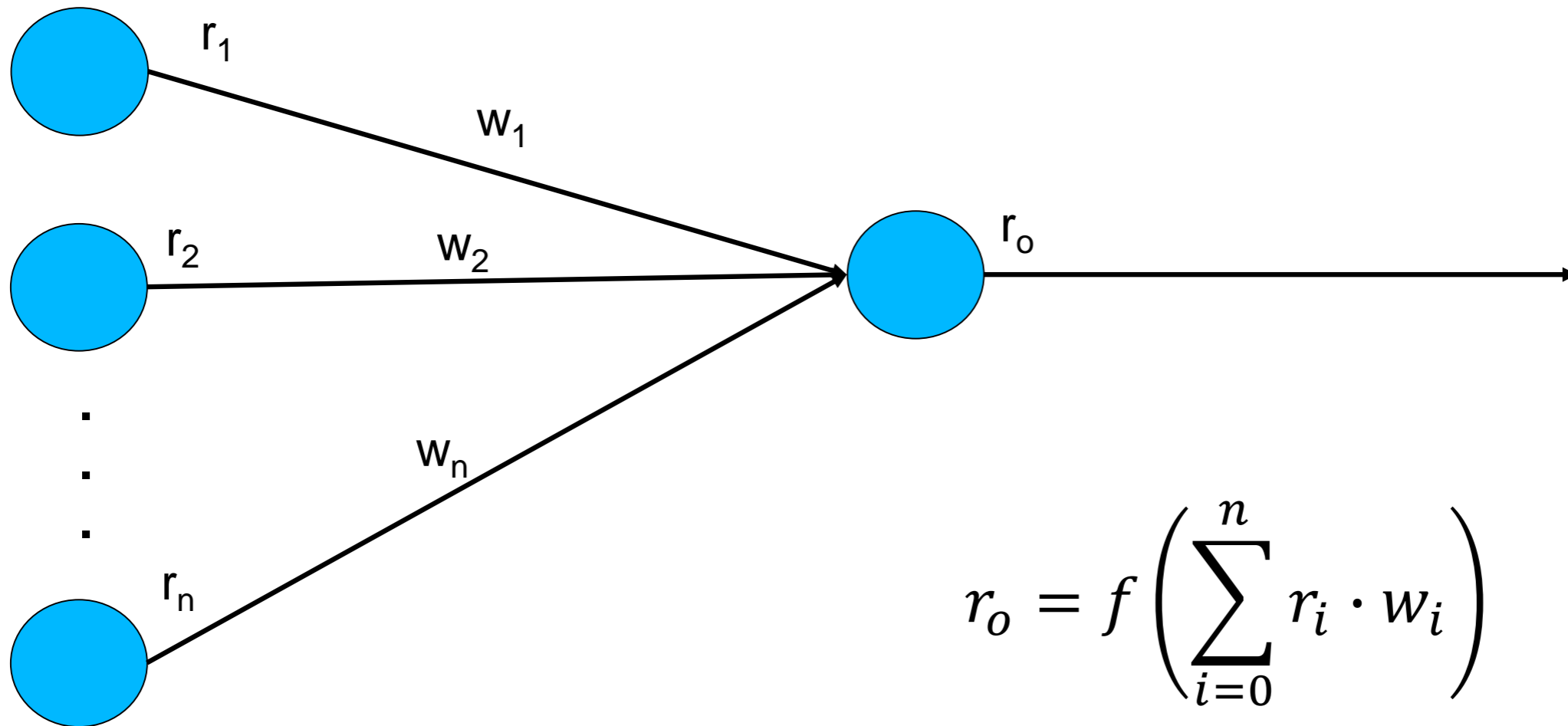
# Rate-based abstractions

Activity of a postsynaptic neuron can be computed as a function of the rates of presynaptic neurons.



# Rate-based abstractions

What about synapses? We can add weights on the connections.



$$r_o = f \left( \sum_{i=0}^n r_i \cdot w_i \right)$$

→ Rosenblatt's perceptron and Artificial Neural Networks.

Too much abstraction!

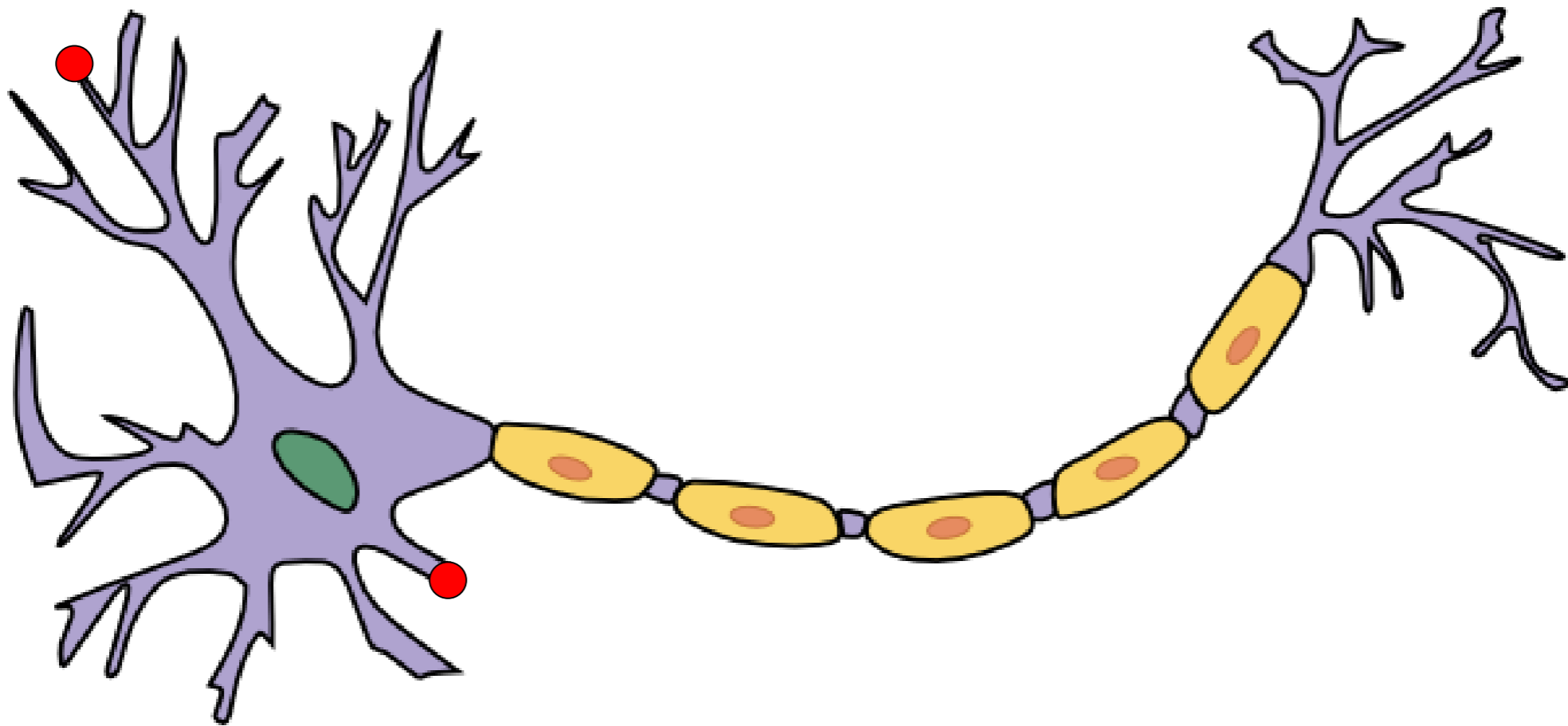




# Point neuron abstractions

Why are these called point-neuron abstractions?

Because we do not take into account the neuron morphology. Each neuron is dimensionless and currents propagate instantaneously from all the receiving synapses.



# Point neuron abstractions – neuron models

The neuron electrical properties can be described through electrical circuits:

- the lipidic membrane acts as a capacitor ( $C_m$ );
- all PSP can be summed up and represented as an external current generator ( $I_{ext}$ ).

We are interested in the voltage between the two termination of the capacitor (membrane potential,  $V_m$ ) and we also add the action potential rule:

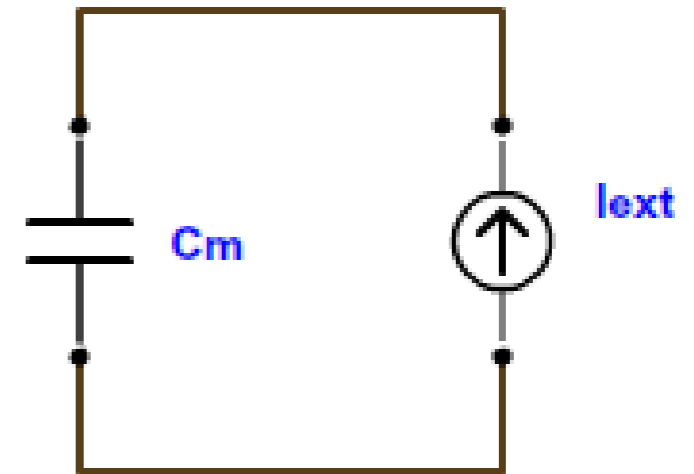
If  $V_m > V_{th}$  then  $V_m$  resets to  $V_{reset}$  and a spike is emitted.



# Point neuron abstractions – neuron models

A first circuit representing neural activity is the **Integrate and fire** model (IAF).

Kirchhoff's law:  $I_C(t) = I_{ext}(t)$



# Point neuron abstractions – neuron models

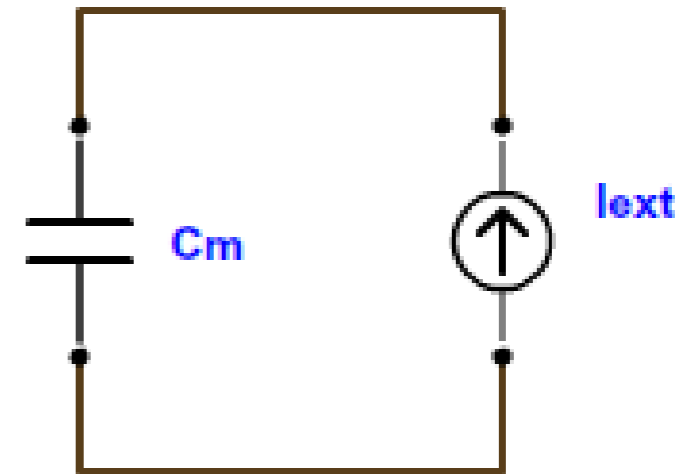
A first circuit representing neural activity is the **Integrate and fire** model (IAF).

Kirchhoff's law:  $I_C(t) = I_{ext}(t)$

$$Q(t) = C_m V_m(t)$$

By deriving the law  
of capacitance:

$$I_C(t) = C_m \frac{dV_m(t)}{dt}$$



# Point neuron abstractions – neuron models

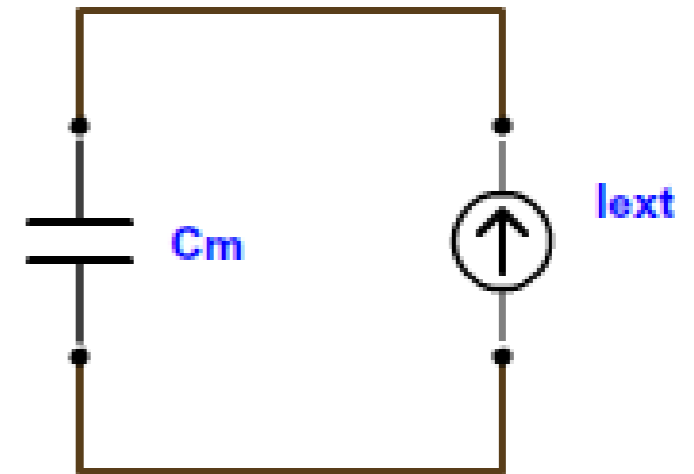
A first circuit representing neural activity is the **Integrate and fire** model (IAF).

Kirchhoff's law:  $I_C(t) = I_{ext}(t)$

$$Q(t) = C_m V_m(t)$$

By deriving the law of capacitance:

$$I_C(t) = C_m \frac{dV_m(t)}{dt}$$



Thus, we obtain:

$$\frac{dV_m(t)}{dt} = \frac{I_{ext}(t)}{C_m}$$



# Point neuron abstractions – simulation loop (I)

We can employ the differential equation to compute the dynamics of the membrane in a simulation loop, by discretizing time in small intervals.

```
T = 2000.0 // total simulation time, ms  
time = 0.0  
V = 0.0  
dt = 1.0 // simulation step, ms
```

```
while (time < T) {
```

```
    Iext = sum_external_currents()
```

```
    dVm = membrane_update(Iext)
```

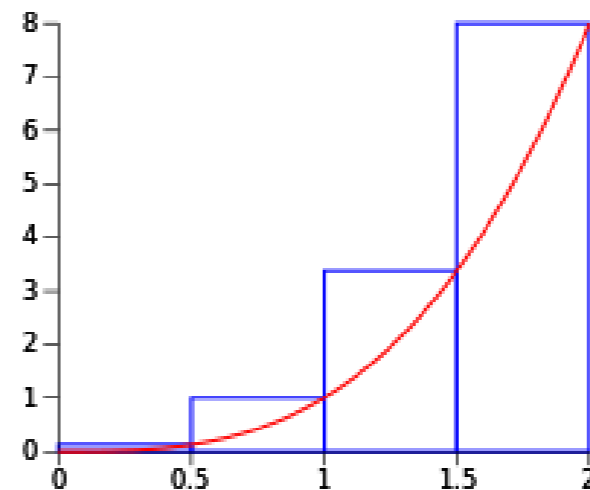
```
    V += dVm * dt // discrete integration
```

```
    if (V > Vth) emit_spike();
```

```
    time += dt
```

```
}
```

$$\frac{dV_m(t)}{dt} = \frac{I_{ext}(t)}{C_m}$$



Let's try it out!



# Point neuron abstractions – neuron models

Neurons have the **refractory period**, that must be taken into account for an accurate simulation. Otherwise, the firing rate will rise indefinitely.

without: 
$$r(I) = \frac{I}{C_m(V_{th} - V_{reset})} \quad \lim_{I \rightarrow +\infty} r(I) = +\infty$$



# Point neuron abstractions – neuron models

Neurons have the **refractory period**, that must be taken into account for an accurate simulation. Otherwise, the firing rate will rise indefinitely.

$$\text{without: } r(I) = \frac{I}{C_m(V_{th} - V_{reset})} \quad \lim_{I \rightarrow +\infty} r(I) = +\infty$$

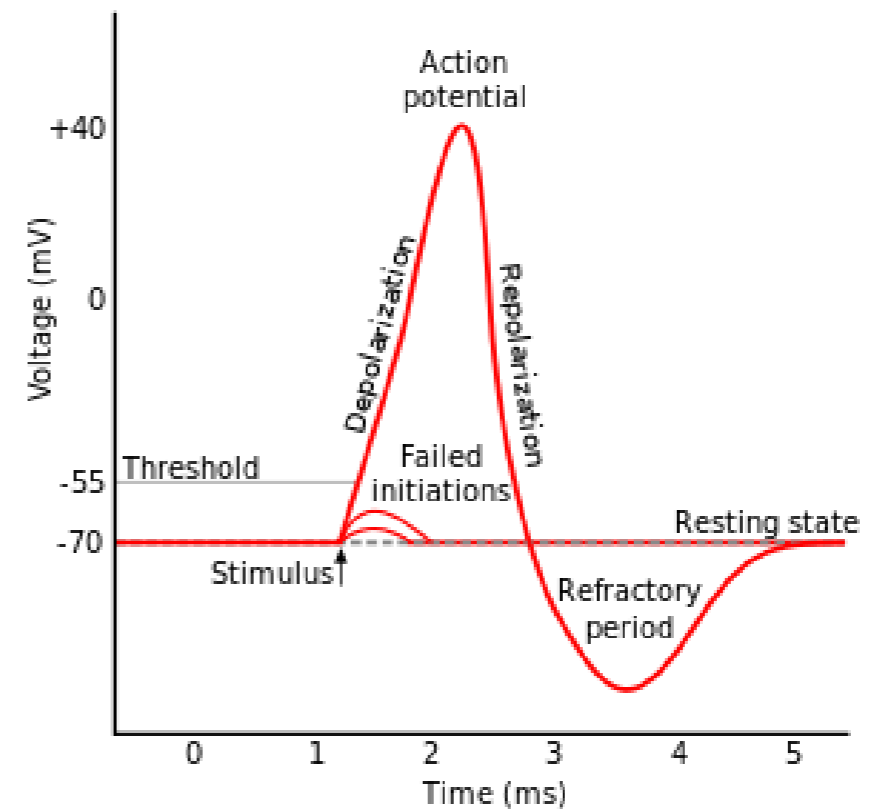
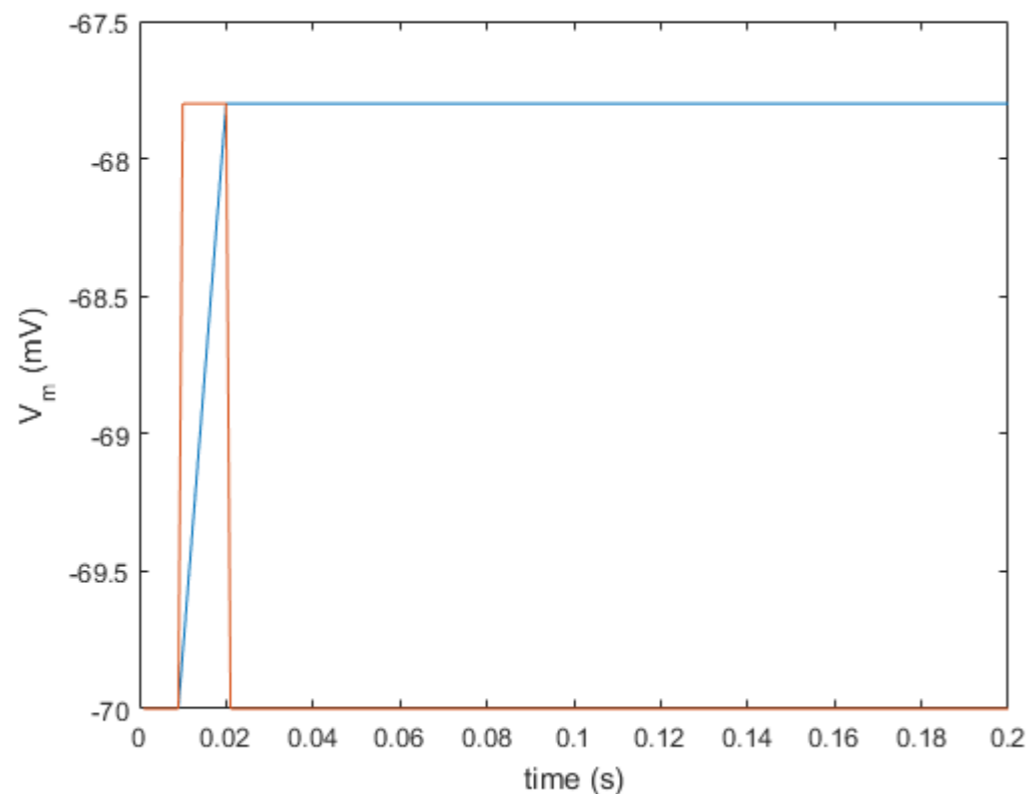
$$\text{with: } r(I) = \frac{I}{C_m(V_{th} - V_{reset}) + t_{ref}I} \quad \lim_{I \rightarrow +\infty} r(I) = \frac{1}{t_{ref}}$$





# Point neuron abstractions – neuron models

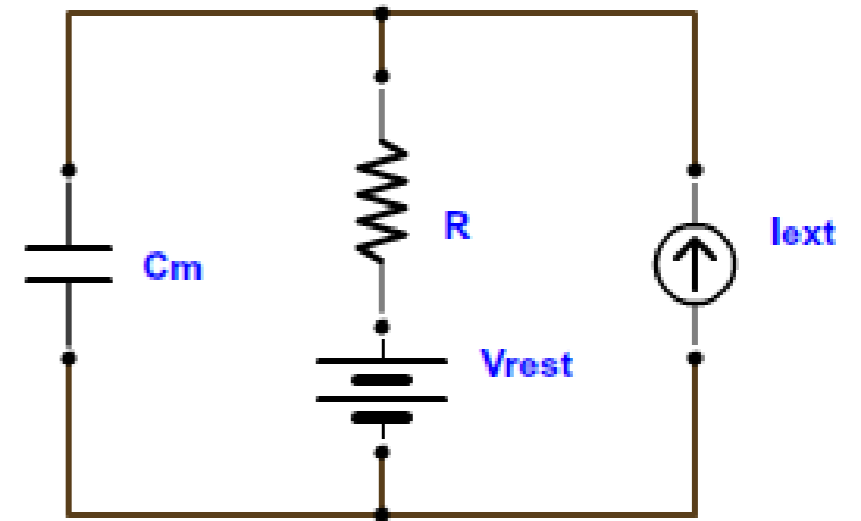
In the IAF model, the membrane continues to keep the gained potential, even if there is no external input current and the spike threshold is not reached. This is not true for the biological neuron.



# Point neuron abstractions – neuron models

The **Leaky integrate and fire** model (LIAF) adds a resistance in the circuit in order to model the leakage of charge. Moreover, a battery is added to represent the equilibrium potential of the cell membrane.

Kirchhoff's law:  $I_C(t) + I_R(t) = I_{ext}(t)$

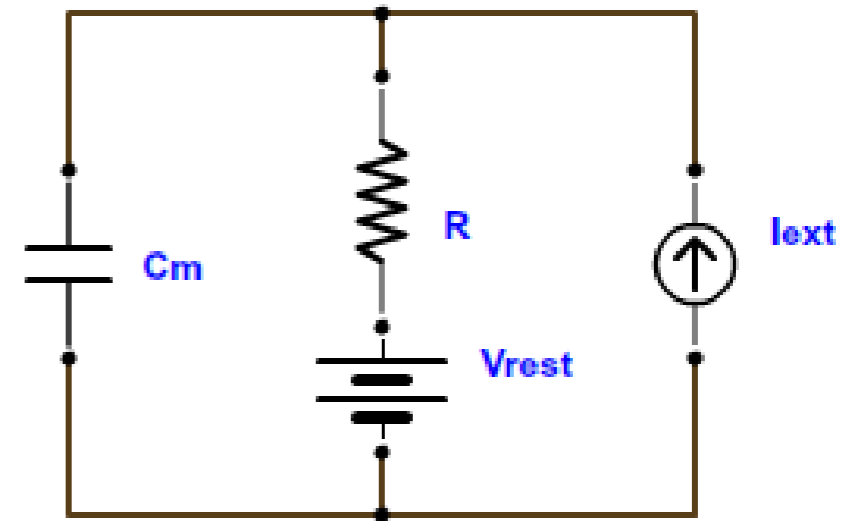


# Point neuron abstractions – neuron models

The **Leaky integrate and fire** model (LIAF) adds a resistance in the circuit in order to model the leakage of charge. Moreover, a battery is added to represent the equilibrium potential of the cell membrane.

Kirchhoff's law:  $I_C(t) + I_R(t) = I_{ext}(t)$

Ohm's law:  $I_R(t) = \frac{(V_m(t) - V_{rest})}{R}$

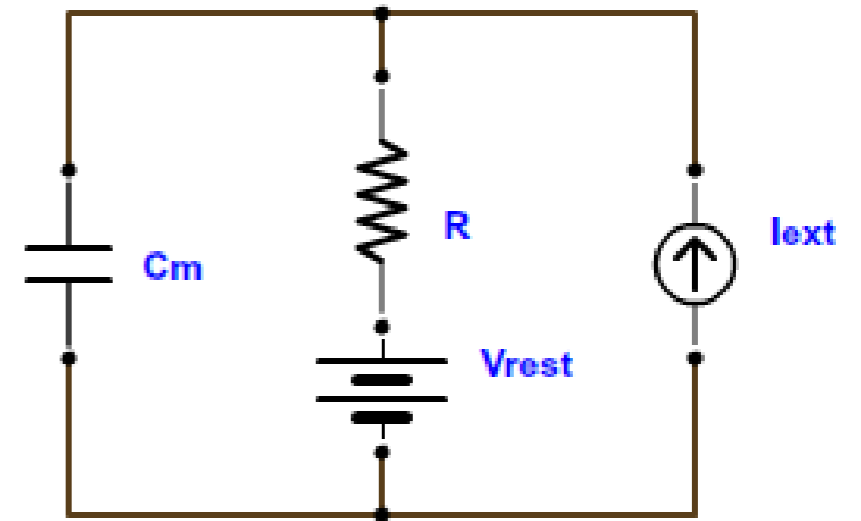


# Point neuron abstractions – neuron models

The **Leaky integrate and fire** model (LIAF) adds a resistance in the circuit in order to model the leakage of charge. Moreover, a battery is added to represent the equilibrium potential of the cell membrane.

Kirchhoff's law:  $I_C(t) + I_R(t) = I_{ext}(t)$

Ohm's law:  $I_R(t) = \frac{(V_m(t) - V_{rest})}{R}$



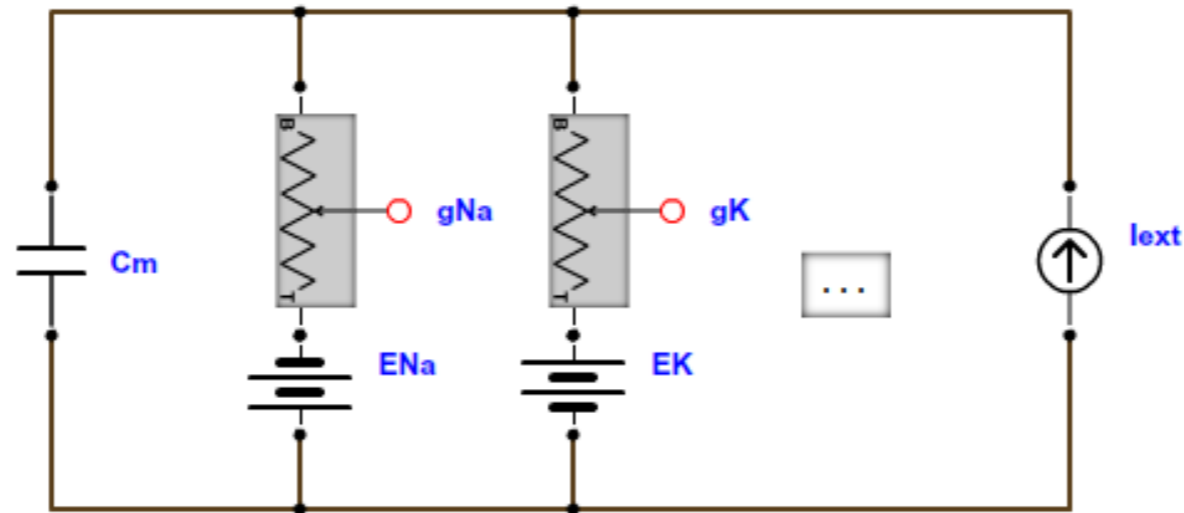
Thus, we obtain: 
$$\frac{dV_m(t)}{dt} = \frac{I_{ext}(t)}{C_m} - \frac{(V_m(t) - V_{rest})}{C_m R}$$



# Point neuron abstractions – neuron models

There are many others neuron models:

**Hodgkin–Huxley:** each ionic channel is modelled as a resistance-battery parallel circuit, with a probabilistic conductance.



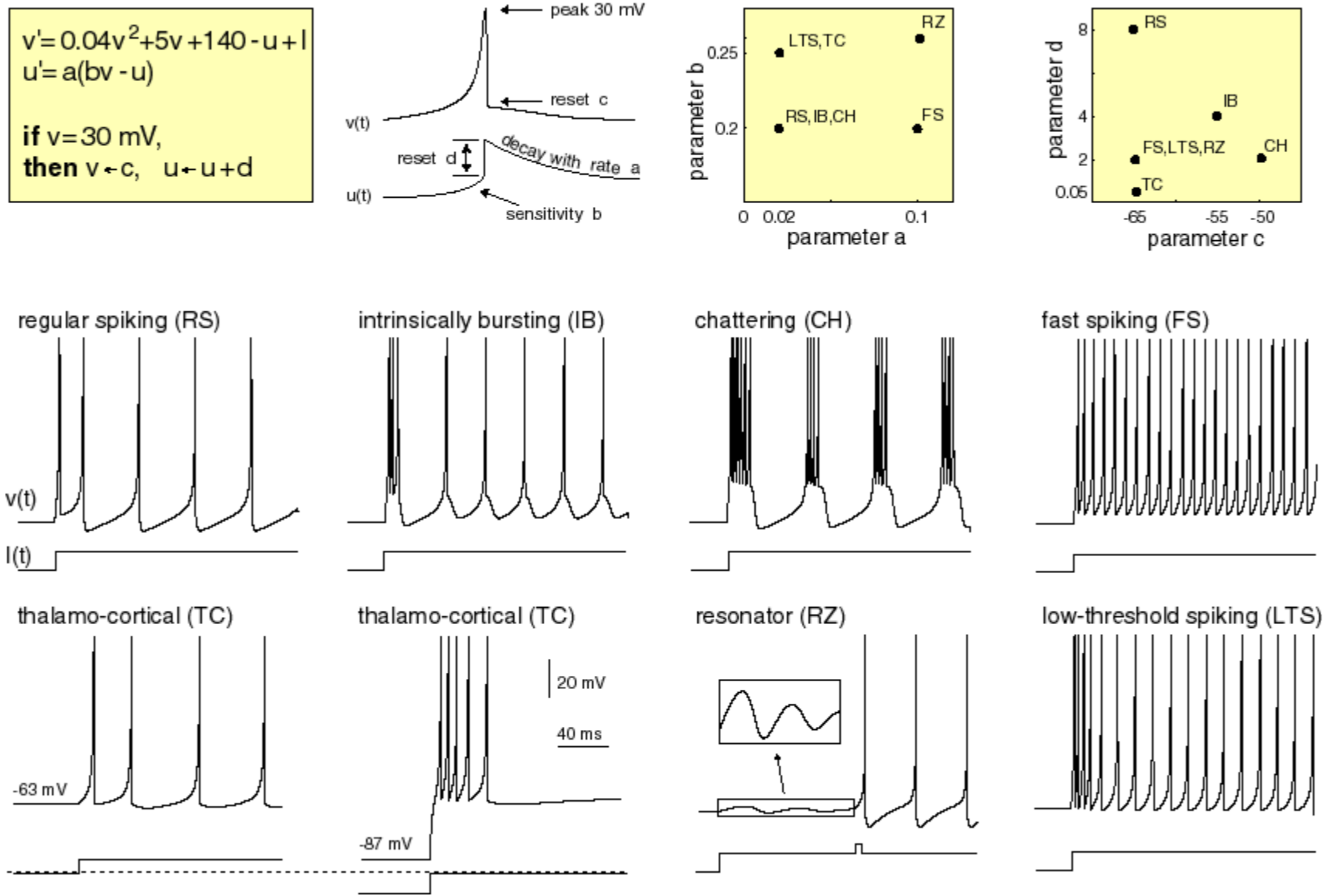
$$\frac{dV_m(t)}{dt} = \frac{I_{ext}(t)}{C_m} - \frac{1}{C_m} \sum_i g_i (V_m(t) - E_i)$$



# Point neuron abstractions – neuron models

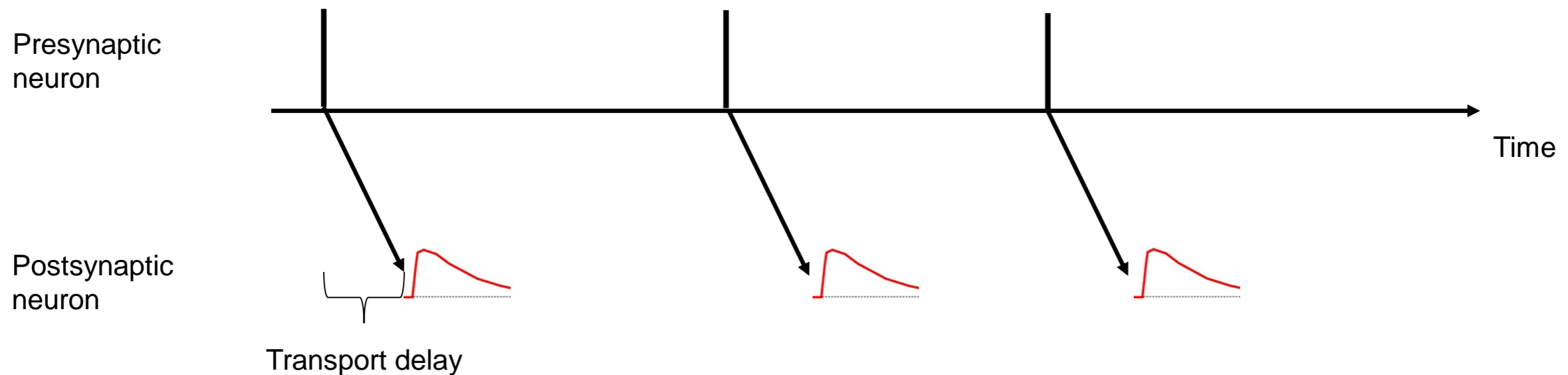
There are many others neuron models:

**Izhikevich:** two differential equations can model many different neuron behaviours.



# Point neuron abstractions – synapses models

Each action potential is transmitted as an event to all postsynaptic neurons connected, after a transmission delay (travel time on the axon). When such event is received a proper EPSC or IPSC is generated and added to the total input current.



Amongst the most common PSC types there is the **alpha-shaped** one:

$$I(t) = \frac{t}{\tau_s} e^{-\frac{t}{\tau_s}}$$



# Point neuron abstractions – synapses models

Each synapse has a weight that has two roles:

1. distinguishing between inhibitory and excitatory synapses by being negative or positive;
2. representing the strength of the connection between the two neurons.

Synaptic weights can be changed via rules implementing STDP, for example:

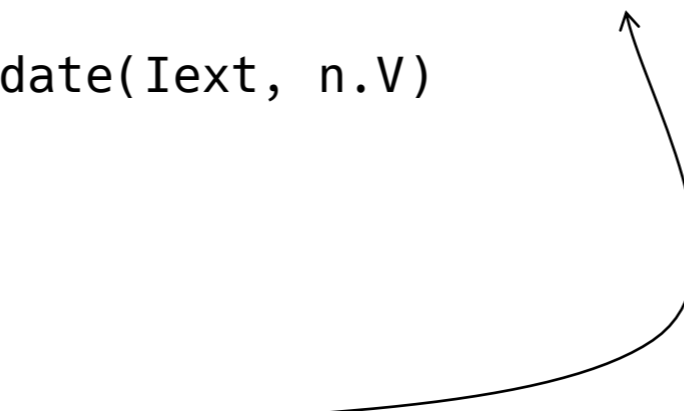
$$\Delta w_{ij} = \sum_f \sum_n W(t_i^f - t_j^n) \quad W(x) = \begin{cases} A_+ e^{(-\frac{x}{\tau_+})} \text{ for } x > 0 \\ -A_- e^{(-\frac{x}{\tau_-})} \text{ for } x < 0 \end{cases}$$





# Point neuron abstractions – simulation loop (II)

Given the previous equations we could in principle create a network simulation loop like the following:

```
while (time < T) {  
    foreach (n : neurons) {  
        Iext = n.sum_external_currents(n.received_spikes)  
        dVm = n.membrane_update(Iext, n.V)  
        n.V += dVm * dt  
        if (n.V > n.Vth) {  
            n.send_spike()  Send spike through delayed  
            and weighted connection  
            n.adjust_weights(n.received_spikes)  
        }  
    }  
    time += dt  
}
```



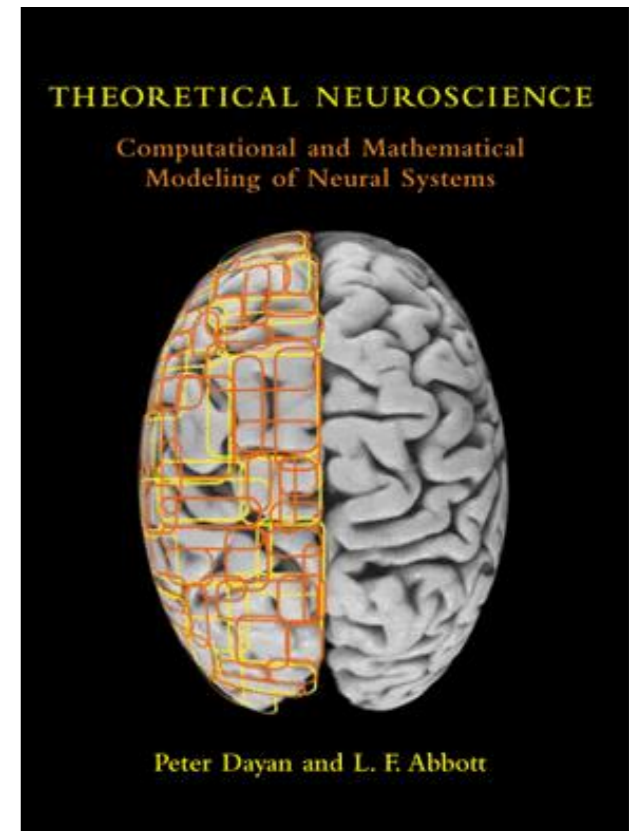
# Neuromorphic computing resources

## Neurology:

- “Principles of Neural Science” by Kandel et al.

## Computational neuroscience:

- “Theoretical Neuroscience: computational and mathematical modeling of neural systems” by Peter Dayan and Larry Abbott
- Computational Neuroscience on Coursera



## Other info (related):

- [lorenzo.vannucci@santannapisa.it](mailto:lorenzo.vannucci@santannapisa.it)



# Preparations for Hands-On session (15/04)

Installing the **Neurorobotics Platform** from:

<http://neurorobotics.net>

It works on Ubuntu 16.04 or  
with docker

We will also provide some live  
USB but they are not so  
reliable

