Lecture 2

2025-09-10

Neural representations: receptive field, tuning curve; Firing rate from spike train with convolution

- Online slides: lec02-represent.html
- Code: code02.ipynb, lec02-data.npz
- Homework: hw02.pdf, hw02-data.npz

Username: cns, Password: nycu2025

Laboratory of Computing Biological Networks

Top view of computational neuroscience

Computational neuroscience is the field of study in which mathematical tools and theories are used to investigate brain function. It can also incorporate diverse approaches from electrical engineering, computer science and physics in order to understand how the nervous system processes information.

nature.com

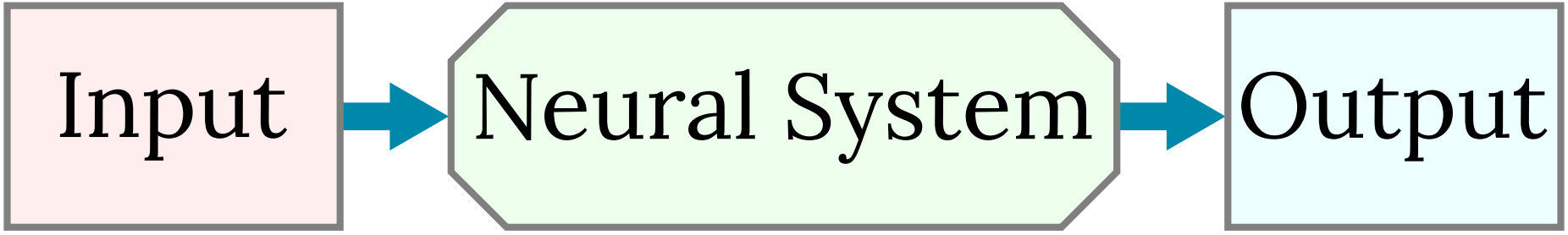
Theoretical analysis and computational modeling are important tools for characterizing what nervous systems do, determining how they function, and understanding why they operate in particular ways.

Dayan & Abbott 2001

What	⇒	Descriptive	⇒	Representation
How	⇒	Mechanistic	⇒	Model
Why	⇒	Interpretive	⇒	Function

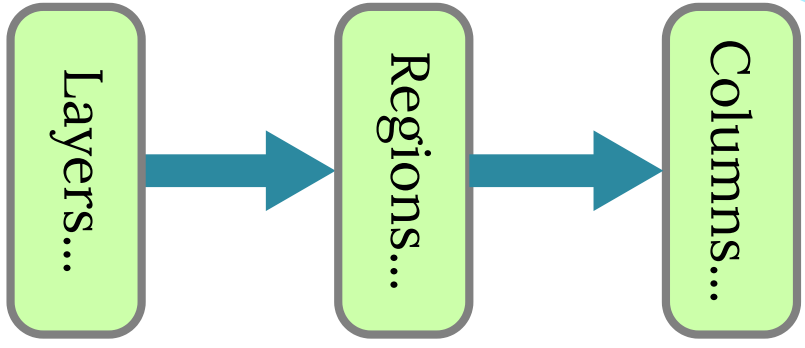
Role of a neural system

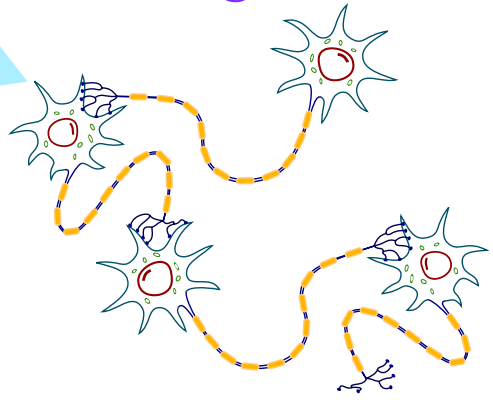
Information processing unit of an animal...



Mechanical: touch sound
Chemical: smell, taste
Radiant: light, heat

Somatic: voluntary, reflex
Autonomic: sympathetic, parasympathetic, endocrine
Emotion, thoughts and behaviors





Action potentials (spikes) are the internal language of neural systems

Representation: what a neuron is responding to

How an input causes or is represented by the activity of a neuron.

– *two notable ways of description:*

Receptive fields

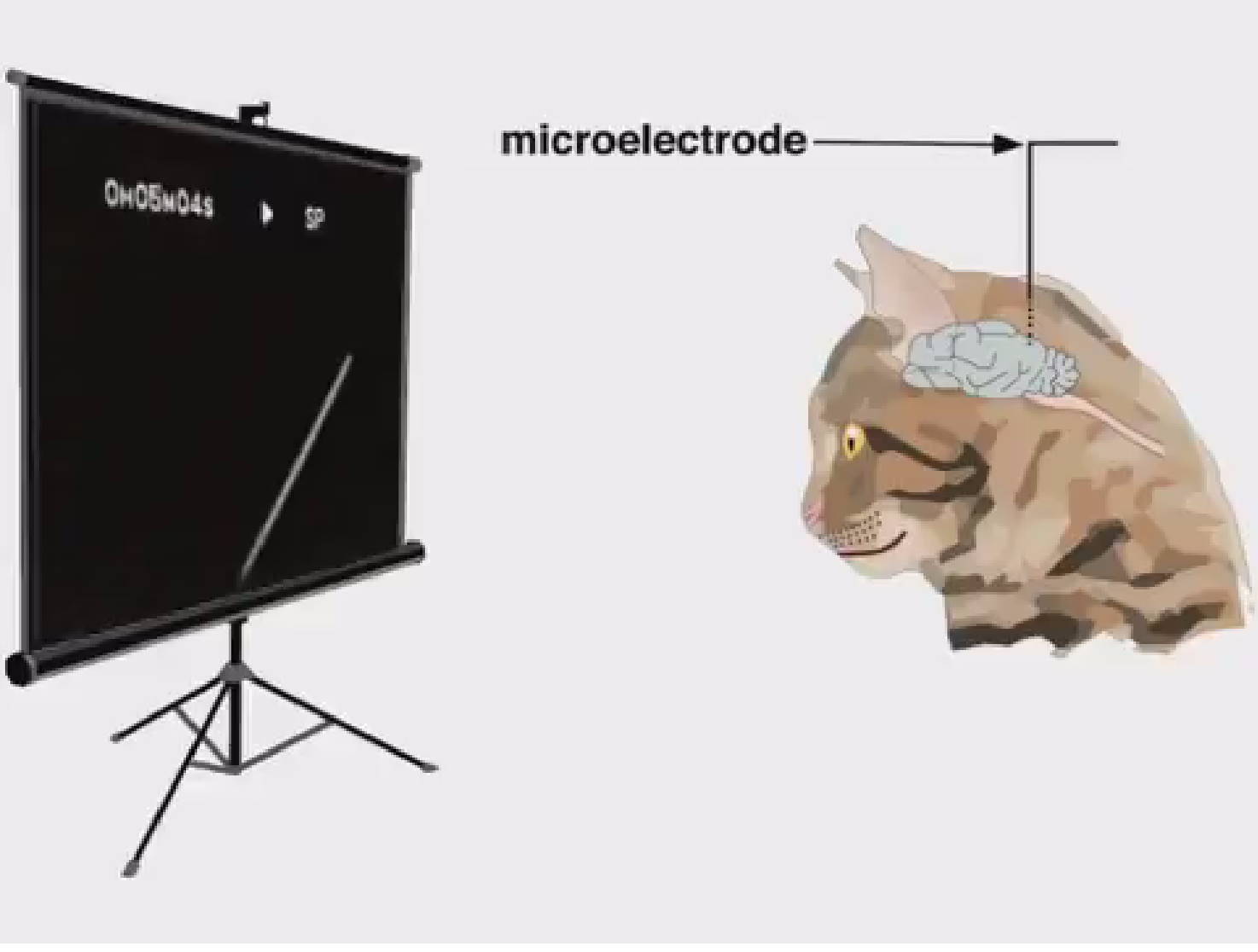
Locations of input that makes a neuron fire

Tuning curves

How response intensity of a neuron depends on an input variable

Receptive field of visual stimuli

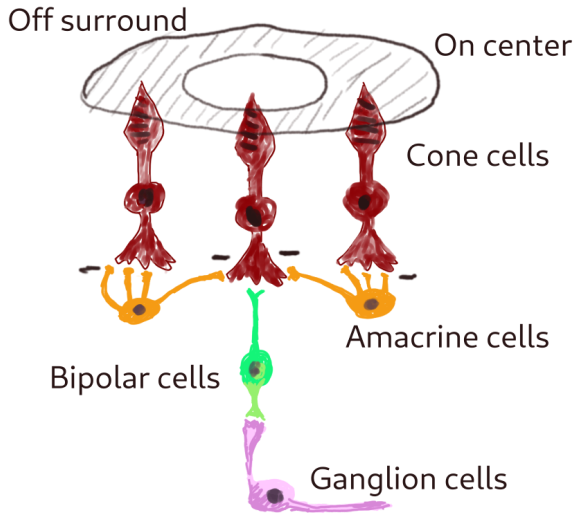
https://www.youtube.com/watch?v=OGxVfKJqX5E



David H. Hubel & Torsten Wiesel, 1959

Receptive field in visual system

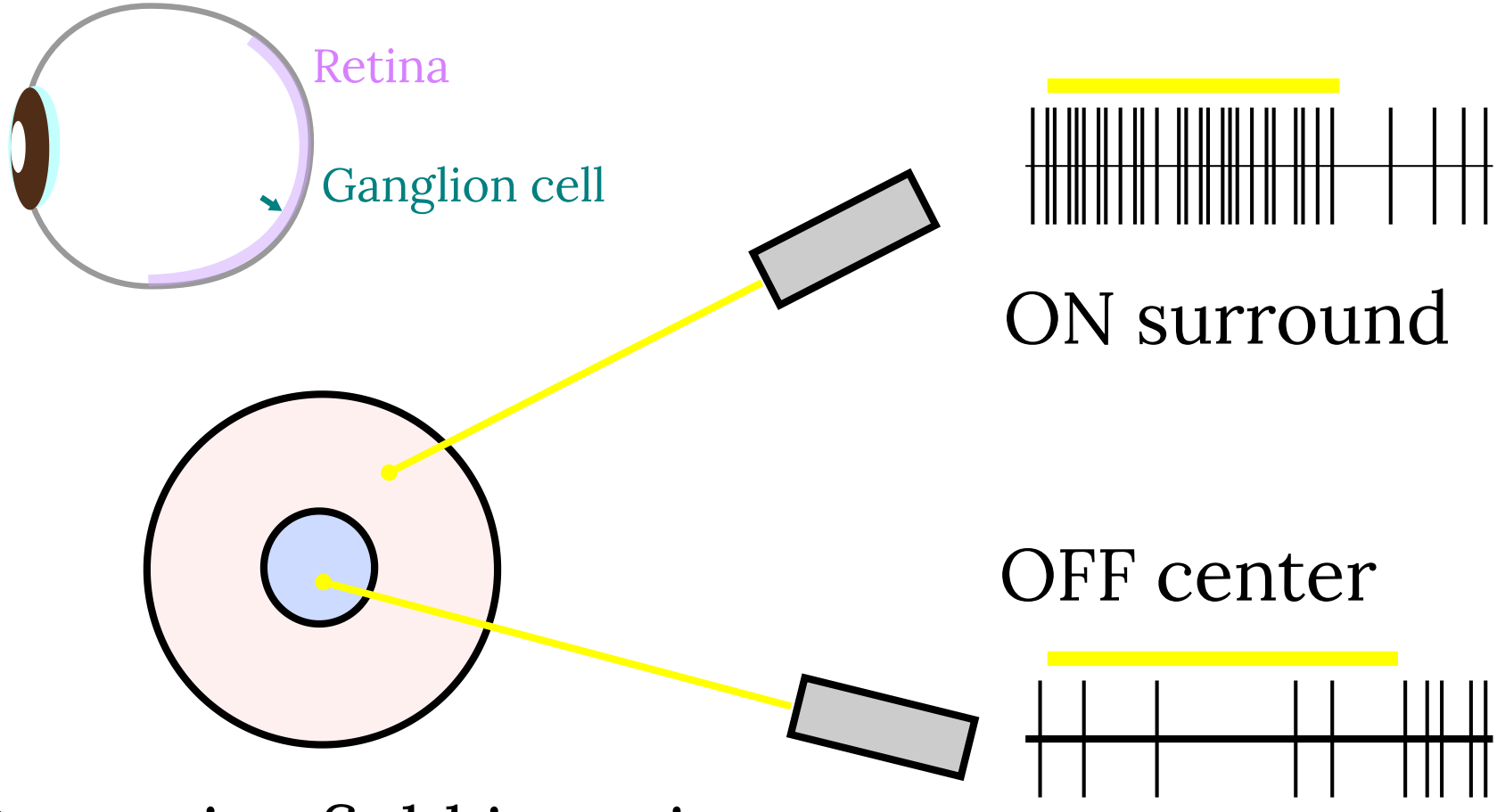
- RF at LGN: center–surround, circular
- RF at V1: elongated, orientation sensitive
- Convergence of neural signals
- Stacking receptive fields
- Efficient coding hypothesis
- Sparse coding, ICA, predictive coding
- Artificial neural networks, CNN



descriptive ⇒ mechanistic ⇒ interpretive

Receptive field of retinal ganglion cell

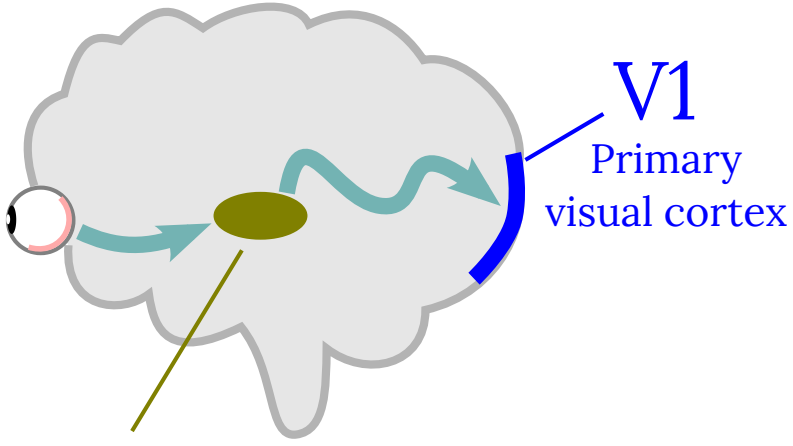
"What"

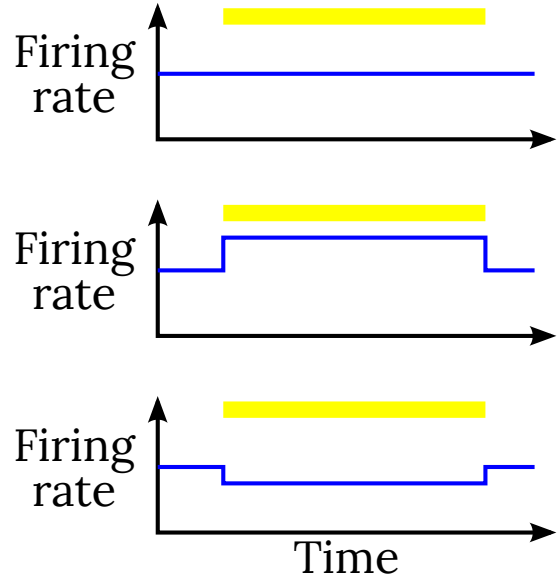


Similar RFs are found at LGN

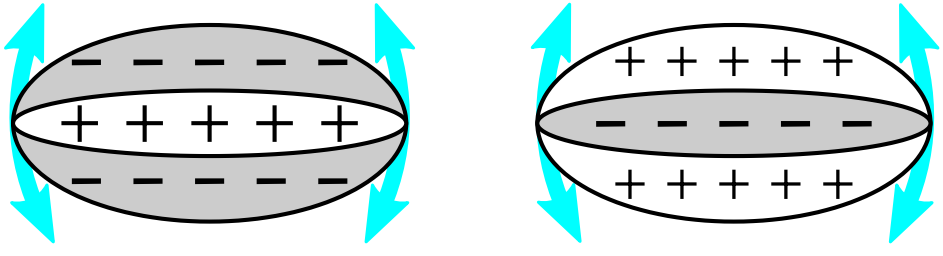
Receptive field of primary visual cortex

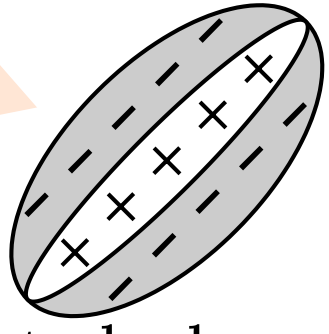
"What"





All possible orientation and location are found in the V1

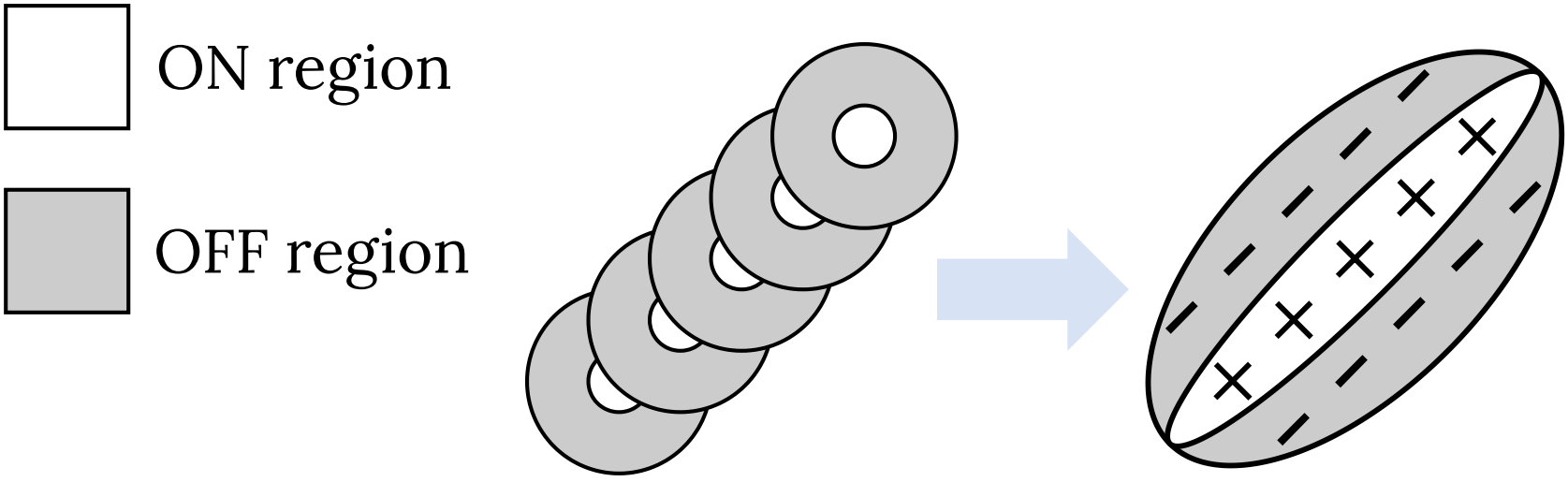




Oriented, elongated receptive field

Mechanistic view for RF formation?

"How"



Proposed by Hubel & Wiesel, output from specific combination of LGN cells are converging to drive a V1 neuron that has a RF that is a superposition of the LGN RFs.

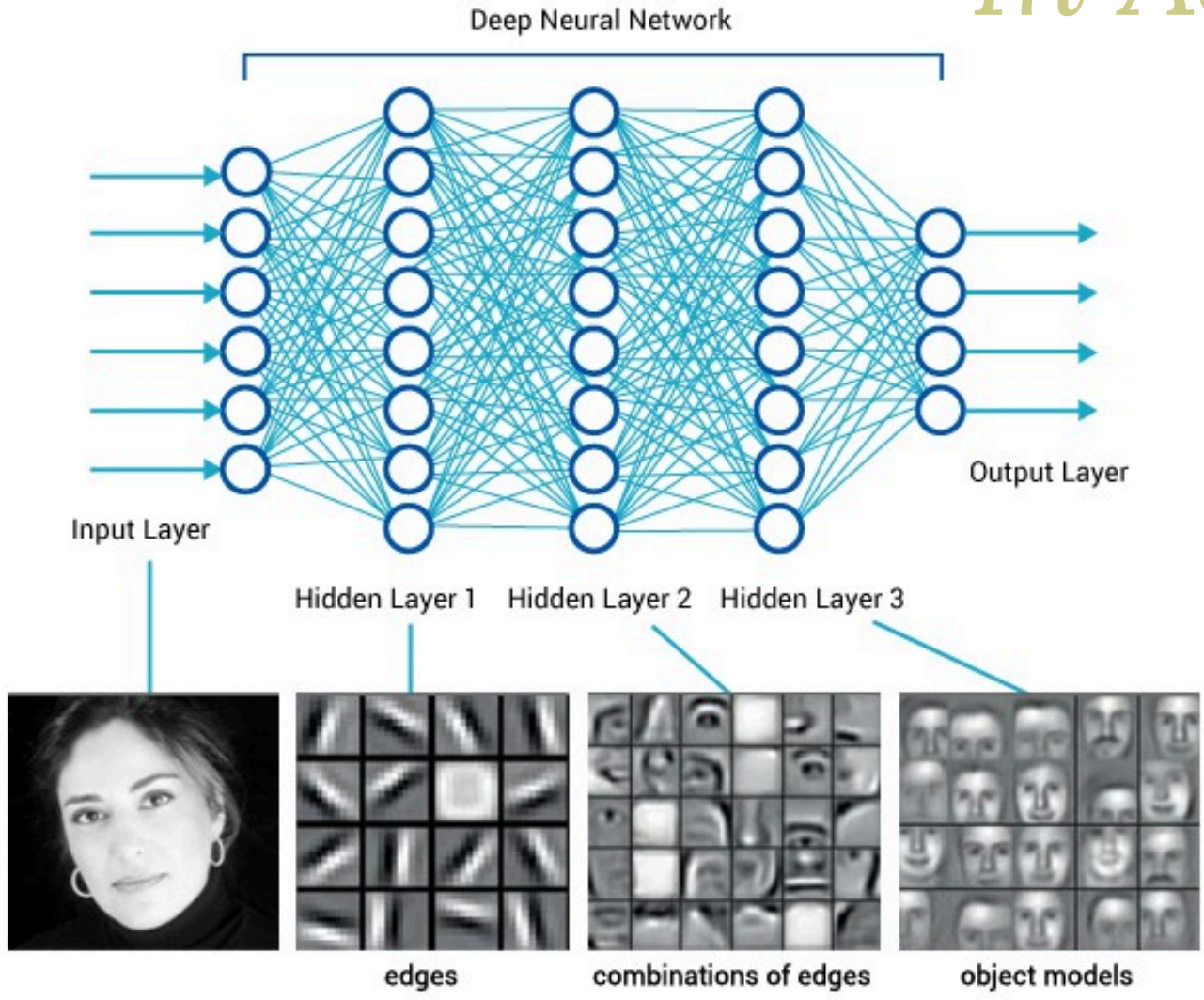
Possible reasons for the RF organization

"Why"

- Sparse coding
  Olshausen & Field (1996)
- Independent component analysis
  Bell & Sejnowski (1997)
- Predictive coding
  Rao & Ballard (1999)

CNN in deep learning

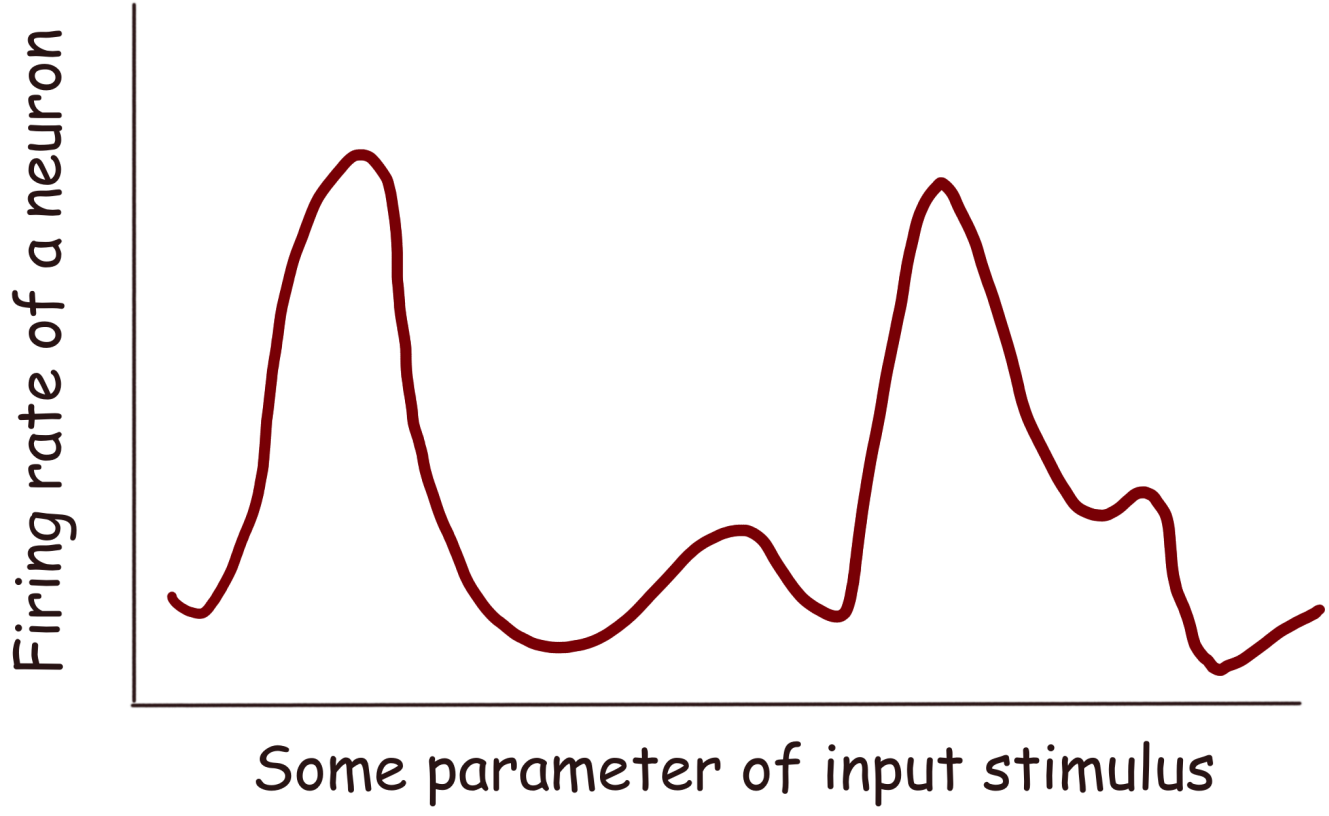
"In Action"



Sachdeva, medium.com (2017)

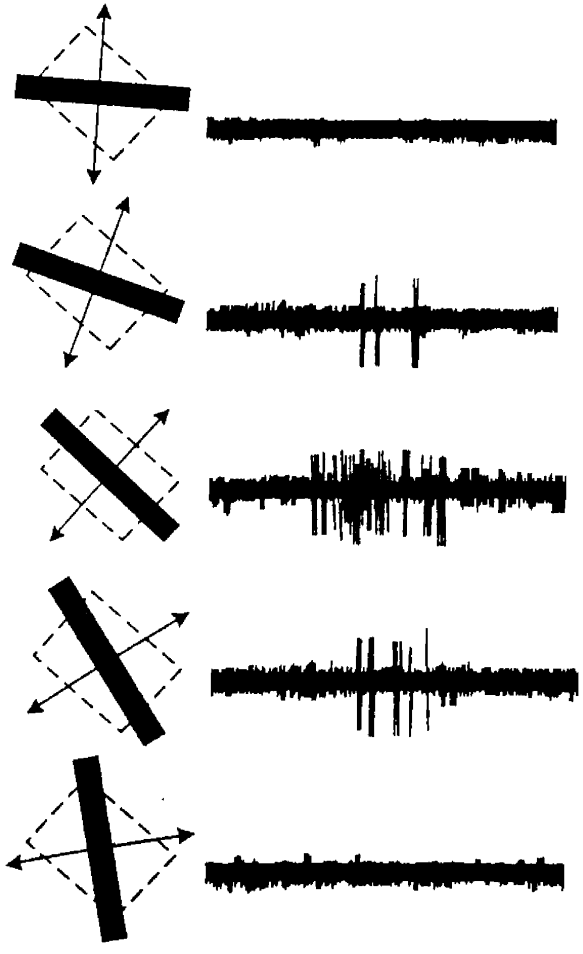
Tuning curve

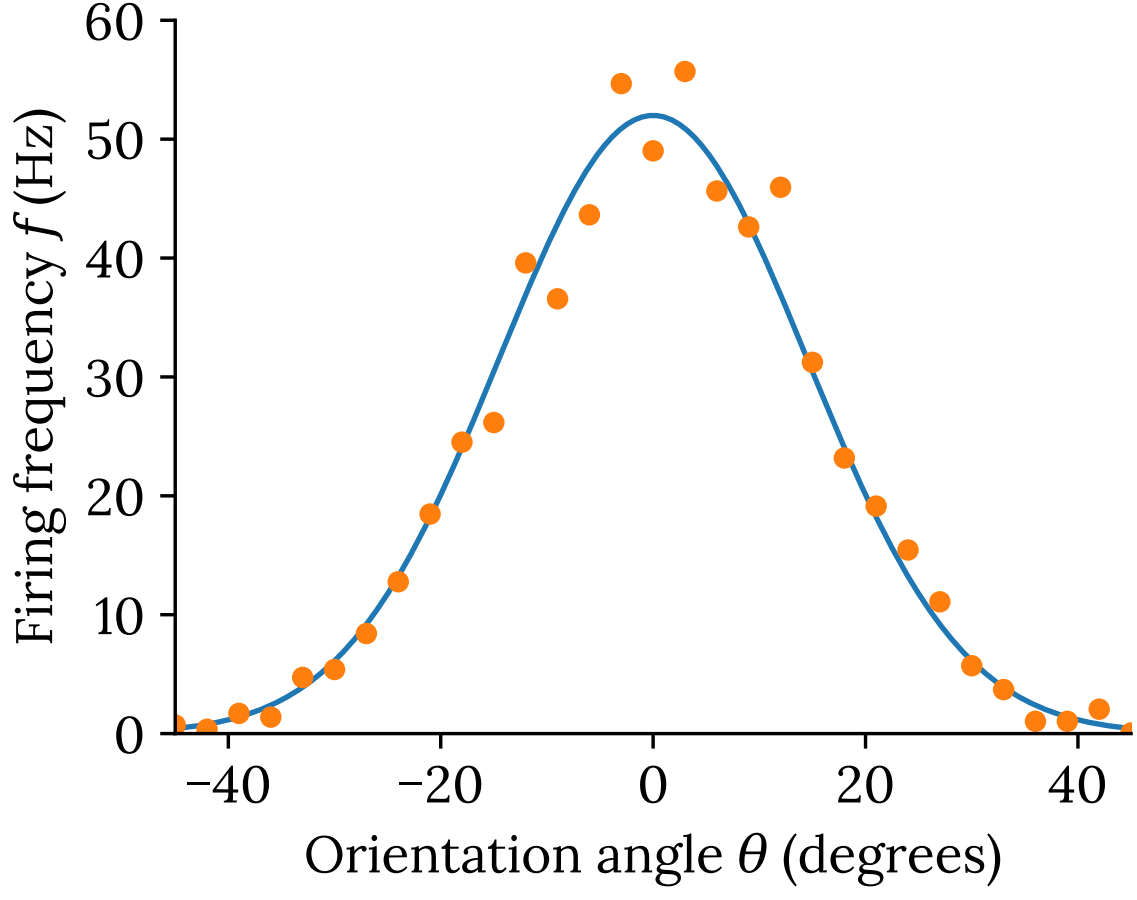
A plot that shows the average firing rate of a neuron as a function of a specific stimulus parameter.



Characterize the neural responses

Stimulus parameter: orientation of light bar

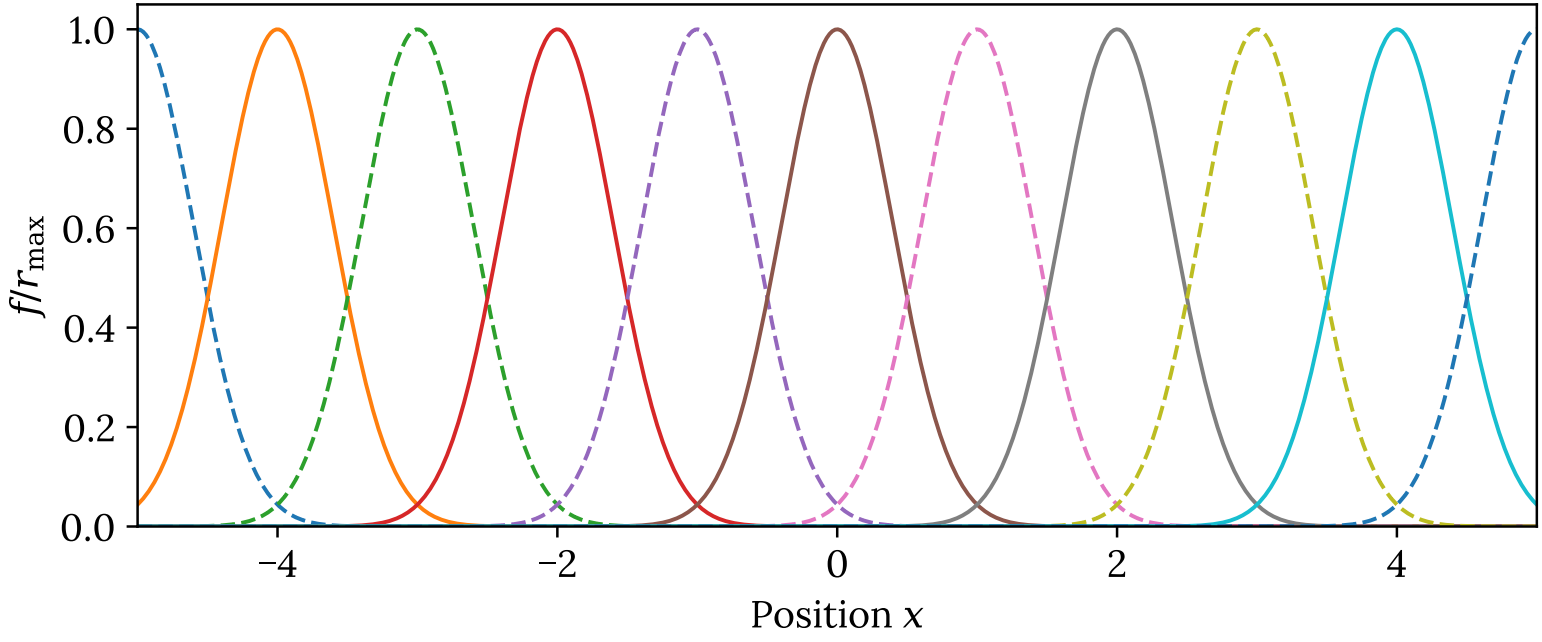




Dayan & Abbott (2001); Hubel & Wiesel (1968)

GPS in brains: Place cells

Stimulus parameter: Position of the animal



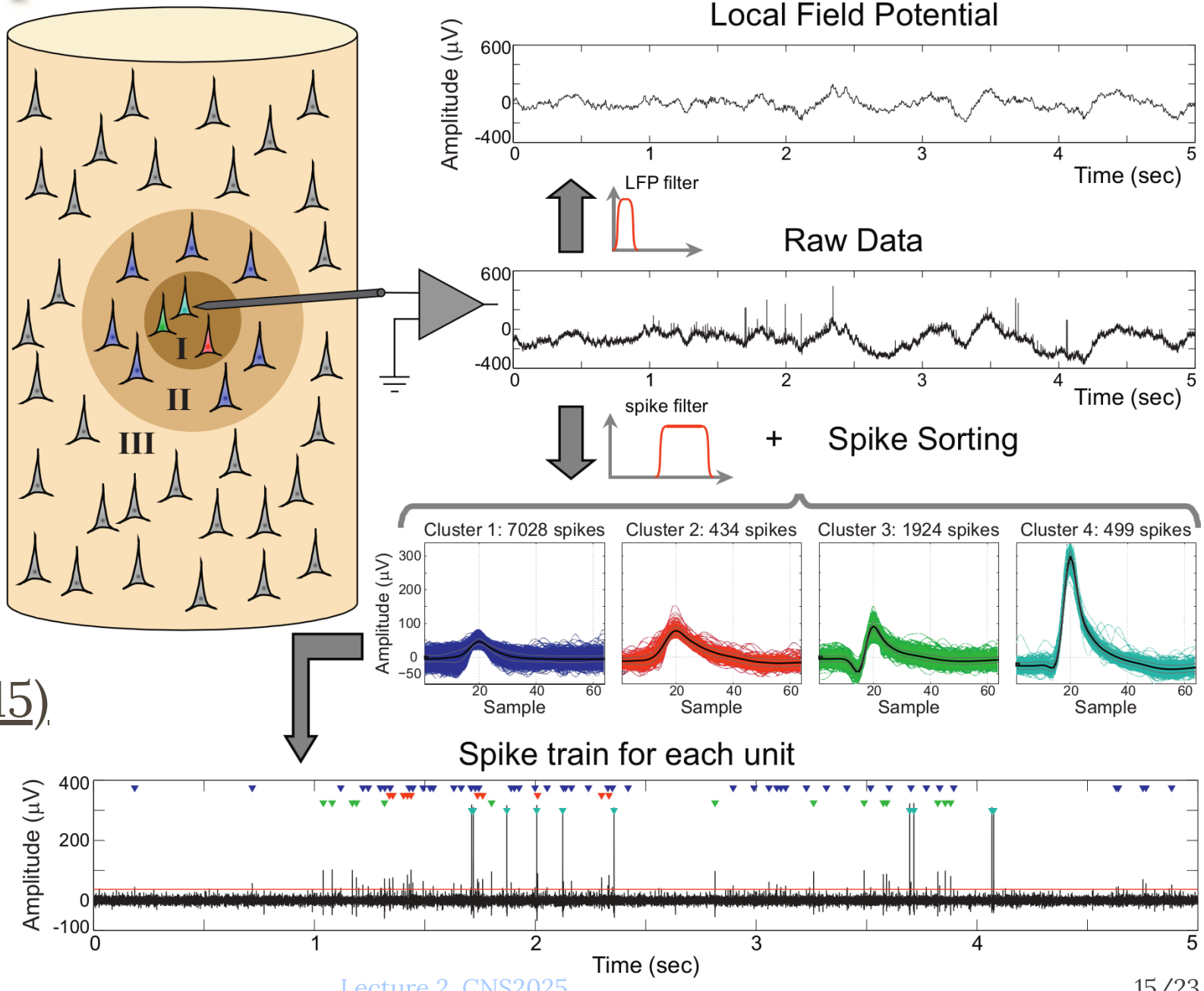
Different colors and line styles represent different cells. Each cell has a preferred position where it fires with maximum rate.

Spikes (or Action Potentials)

How do we quantify their properties?

How are spikes recorded?

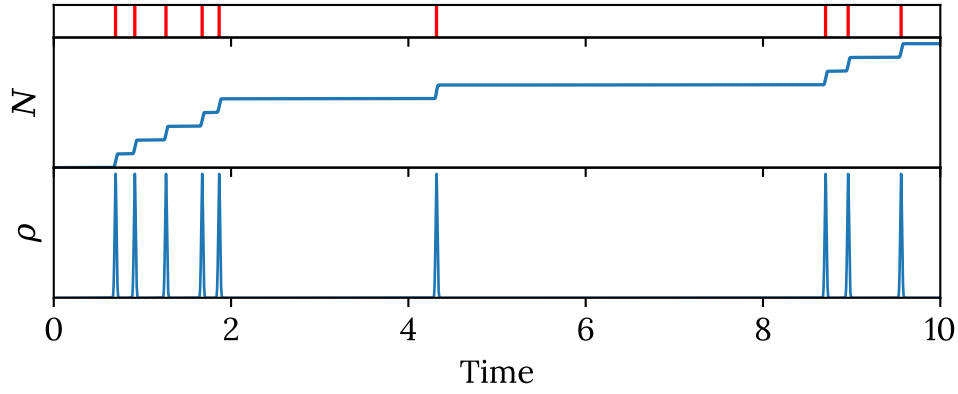
From
local field
to
spikes



Rey et al (2015)

Mathematical spike train

$$\rho(t) = \sum_i \delta(t - t_i) = \frac{d}{dt} N(t)$$



where $N(t)$ is total number of spikes up to time t, $\delta()$ is the Dirac delta function. The mean spike (firing) rate over T is

$$r = \frac{1}{T} \int_0^T d\tau \rho(\tau) = \frac{1}{T} [N(T) - N(0)]$$

Instantaneous firing rate is defined over a trial average ($\langle\rangle$)

$$r(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_t^{t+\Delta t} d\tau \langle \rho(\tau) \rangle = \langle \rho(\tau) \rangle$$

assuming the same protocol can be repeated for many times.

Computer representations of spiking data

Frame or bin representation:

$$\sigma_i = \pm 1$$
$$\sigma_i = 0 \text{ or } 1$$

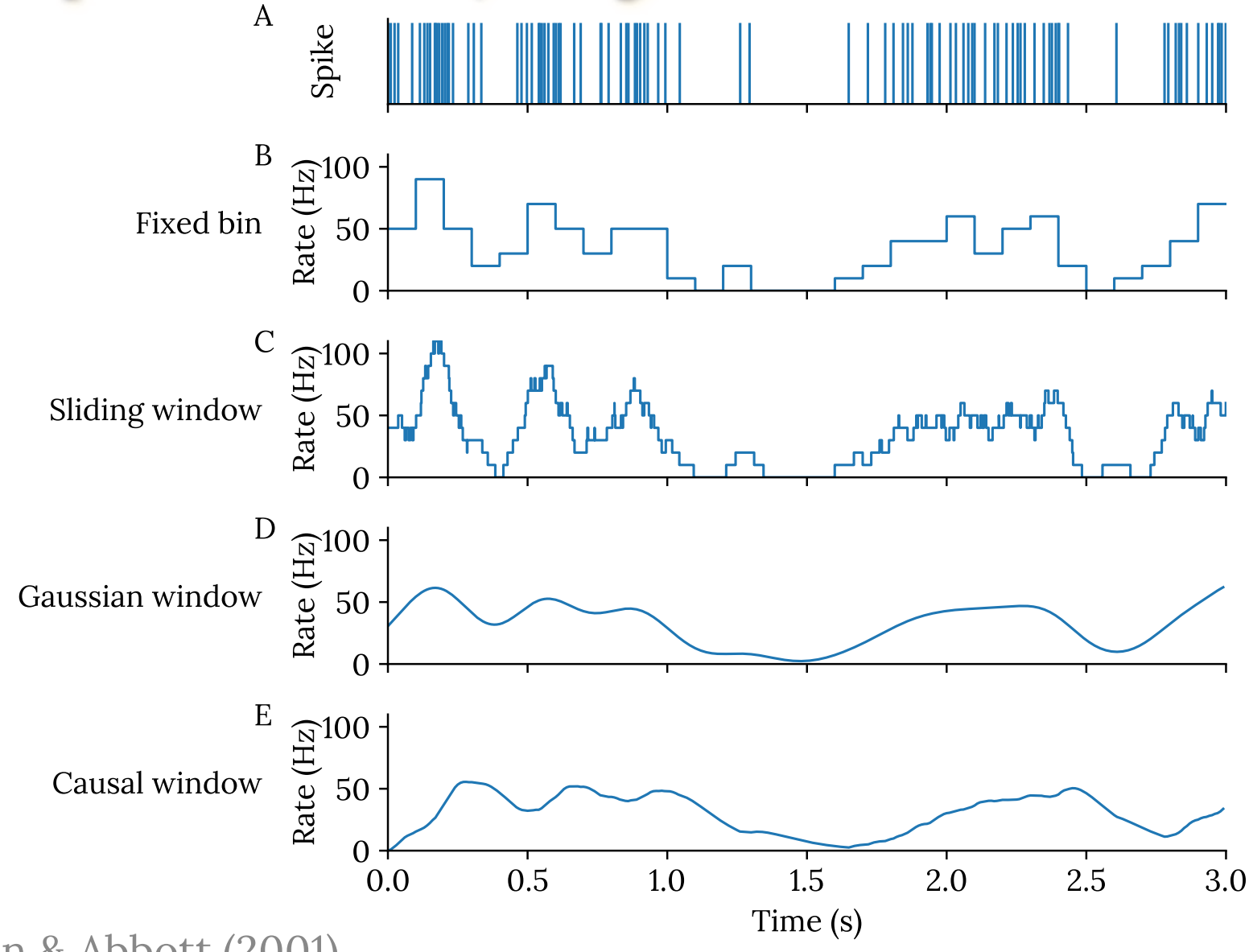
where i is the index of the frame or bin at time around $t = i\Delta t$. The value depends on whether there is a spike within. (Alternatively, spike count can be used for the frame or bin.) Data length scales with the recording time.

List of spiking time:

$$t_0, t_1, t_2, \ldots$$

where t_i is the ith firing time of the neurons. Data length scales with the number of recorded spikes.

From spike train to firing rate



Dayan & Abbott (2001)

Procedures for the firing rate

A. A spike train from a neuron in the inferotemporal cortex of a monkey recorded while that animal watched a video on a monitor under free viewing conditions.

B. Discrete-time firing rate obtained by binning time and counting spikes with $\Delta t = 100$ ms.

C. Approximate firing rate determined by sliding a rectangular window function along the spike train with $\Delta t = 100$ ms.

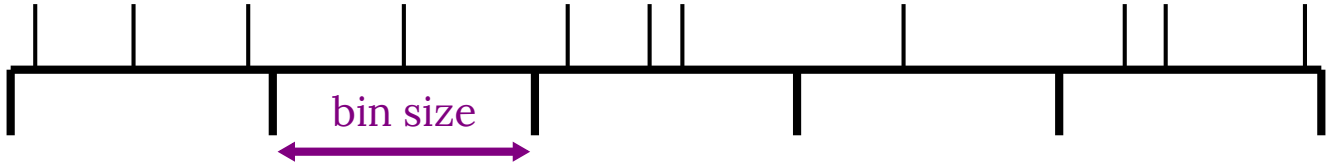
D. Approximate firing rate computed using a Gaussian window function with $\sigma_t = 100$ ms.

E. Approximate firing rate using the window function

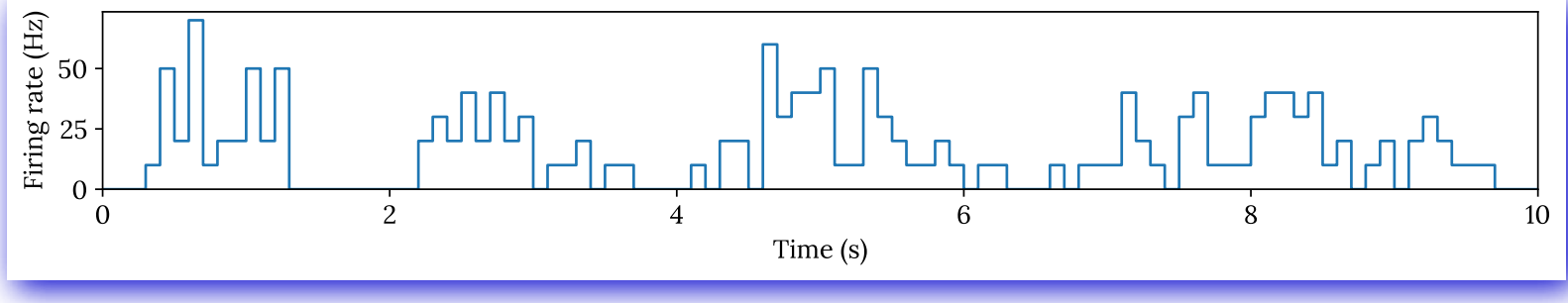
$$W(t) = [-\alpha^2 t \exp(\alpha t)]_+$$

where $[x]_+ = x$ if $x > 0$ else 0, with $1/\alpha = 100$ ms.

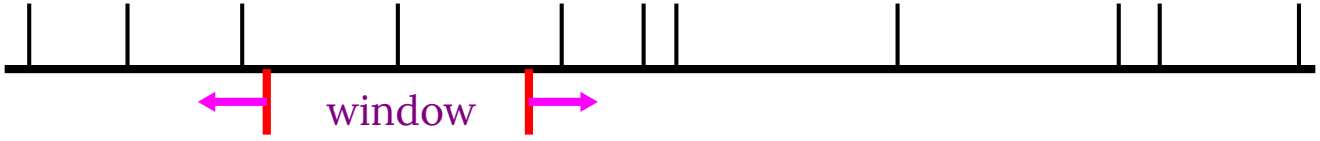
Fixed Binning



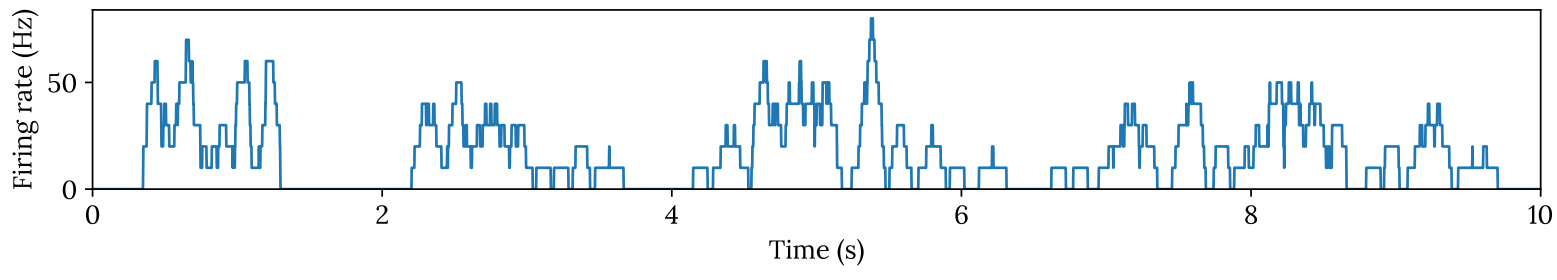

```
bin_size = 0.1
bin_spike_counts = np.zeros(int(10/bin_size)+1)
for t,s in zip(times, spike_counts):
  bin_spike_counts[int(t/bin_size)] += s
plt.figure(figsize=(12,1.5))
plt.step(
  (np.arange(len(bin_spike_counts)))*bin_size,
  bin_spike_counts/bin_size,where='post')
plt.xlim(0,10)
plt.ylim(0,None)
plt.xlabel('Time (s)')
plt.ylabel('Firing rate (Hz)')
```



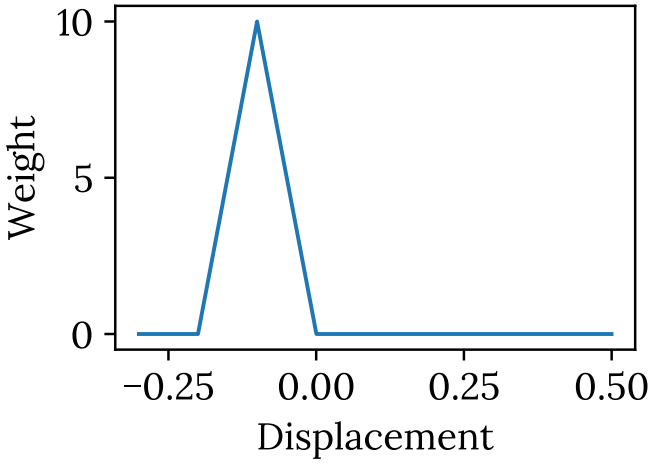
Sliding window




```
win_size = 0.1 # Size of sliding window
time_resolution = 0.001 # Spacing of time points
win_times = np.arange(0,10,time_resolution) # Time points
win_spike_counts = np.zeros(len(win_times)) # Spike counts in window
for t,s in zip(times, spike_counts): # Spike time
    # Find index range [i0,i1) of windows overlap the time
    i0 = max(0,int((t-win_size/2)/time_resolution))
    i1 = min(len(win_times),int((t+win_size/2)/time_resolution))
    win_spike_counts[i0:i1] += s
plt.figure(figsize=(12,1.5))
plt.plot(win_times,win_spike_counts/win_size)
plt.xlim(0,10)
plt.ylim(0,None)
plt.xlabel('Time (s)')
plt.ylabel('Firing rate (Hz)')
```

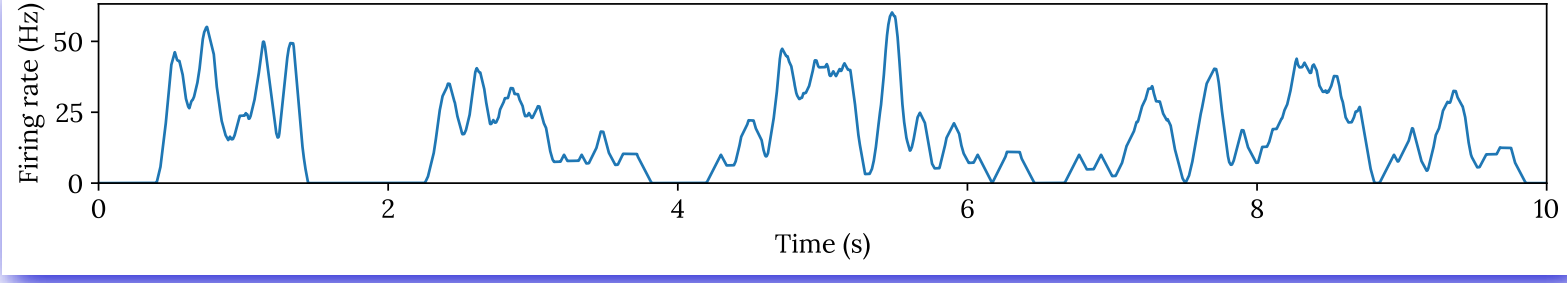


Convolution




```
def triangle_func(times,w):
    t = times+w
    return (t>-w)*(t<w)*(w-np.abs(t))
win_times = np.arange(-0.3,0.5,delta_t)
win_kernel = triangle_func(win_times,0.1)
# Normalize
win_kernel /= (delta_t*win_kernel.sum())
plt.plot(win_times,win_kernel)
plt.xlabel('Displacement')
plt.ylabel('Weight')
```

```
front_drop = (win_times>0).sum() # Number of points to be dropped from the front
estimated_rates = np.convolve(spike_counts,win_kernel[::-1])
estimated_rates = estimated_rates[front_drop:-len(win_kernel)+1+front_drop]
plt.figure(figsize=(12,1.5))
plt.plot(times, estimated_rates) # remove wkn//2 from beg
plt.xlim(0,10)
plt.ylim(0,None)
plt.xlabel('Time (s)')
plt.ylabel('Firing rate (Hz)')
```



Details of convolution

Arbitrary kernel function can be used. However since the discrete convolution is defined as: (See NumPy reference.)

$$(a * v)_n = \sum_{m=-\infty}^{\infty} a_m v_{n-m}$$

The ordering of v is reversed and multiplying with a. We will thus reverse our window when calling convolve.