Spiking Neural Networks

Timothée Masquelier
timothee.masquelier@cnrs.fr

ISAE Feb 2021

UNIVERSITÉ TOULOUSE III PAUL SABATIER

What is computational neuroscience?

Simply put, it is the field that is concerned with how the brain computes. The word “compute” (…) refers to the operations that must be carried out to perform cognitive functions. (…) Computational neuroscience (…) seeks a mechanistic understanding of these operations, to the point that they could potentially be simulated on a computer. Romain Brette

What I cannot create, I do not understand. Richard Feynman, 1985

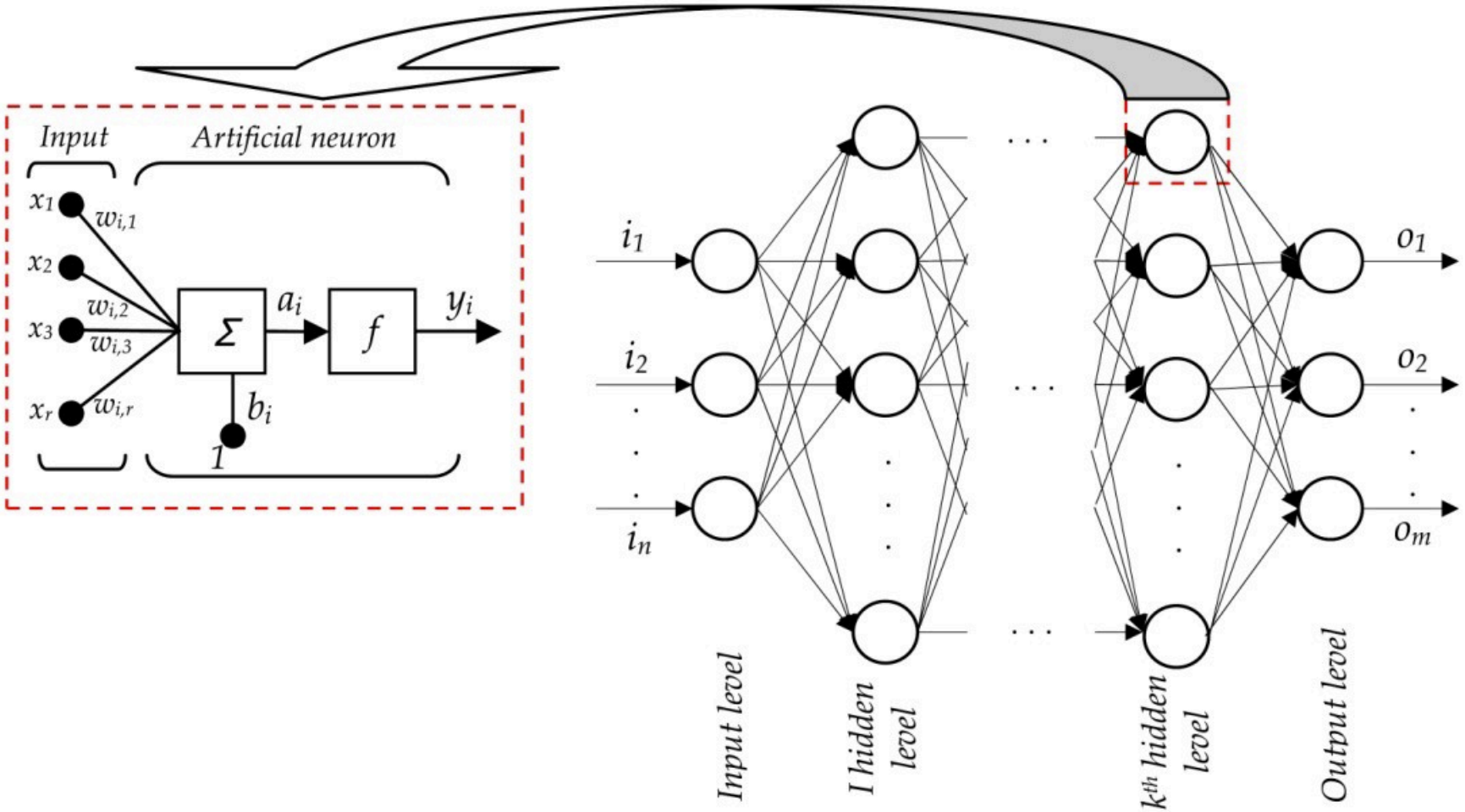
$$C\frac{dV}{dt} = -g_l(E_l - V) + I$$

In praise of neuromorphic engineering

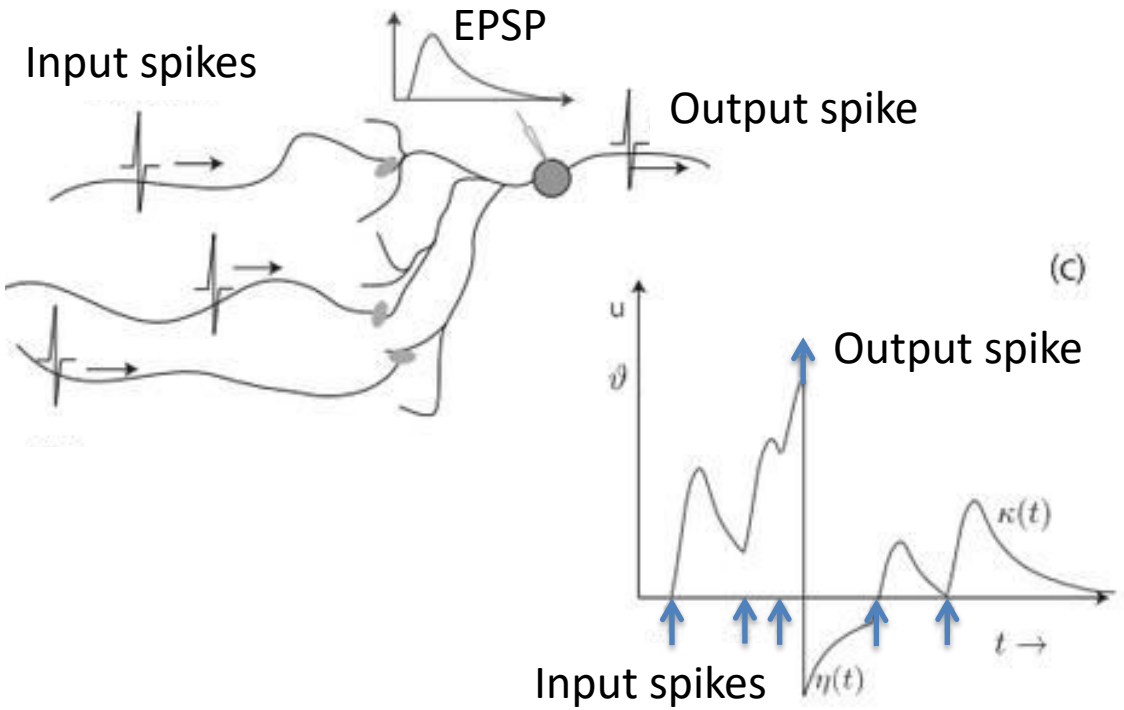
- Biological nervous systems are optimized by natural selection in terms of: speed, robustness, energy consumption, learning (speed, flexibility, generalization)
- High performances despite:
  - Slow hardware (firing rates ≤ 100Hz, conduction ~1-2m/s)
  - Unreliable hardware
  - “Only” ~ 4.10^9 neurons in the human visual system

$\Rightarrow$ much room for improvements!

Artificial Neural Networks (ANN) (feedforward)



Biological neurons emit “spikes”



- Spikes are stereotyped electrical impulses (“all-or-none”) that a neuron emits when sufficiently stimulated
- They propagate along the axons without attenuation (“active propagation”).
- Neurons can only exchange information via spikes.

A biophysical model: the Hodgkin–Huxley model

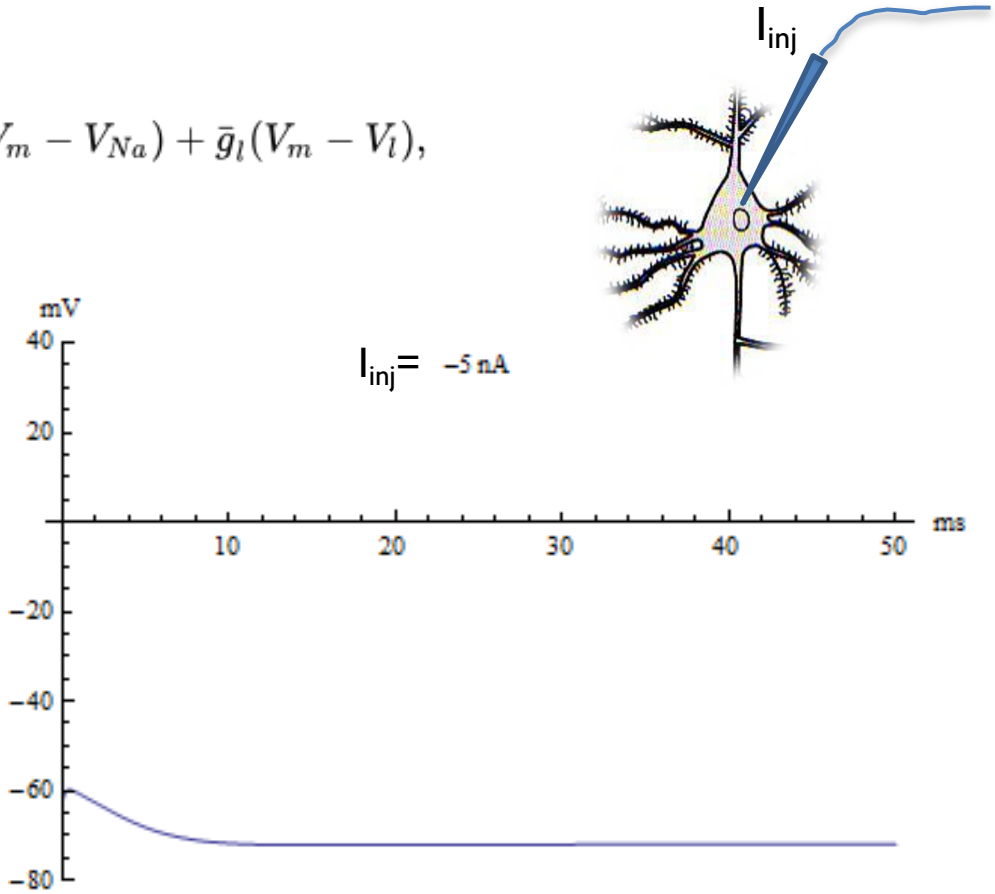
Ionic current dynamics:

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + \bar{g}_{\mathrm{K}} n^4 (V_m - V_K) + \bar{g}_{\mathrm{Na}} m^3 h (V_m - V_{Na}) + \bar{g}_l (V_m - V_l),$$

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n$$

$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m$$

$$\frac{dh}{dt} = \alpha_h(V_m)(1-h) - \beta_h(V_m)h$$

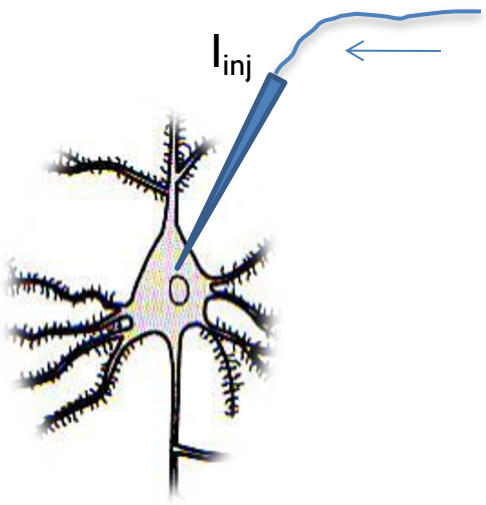


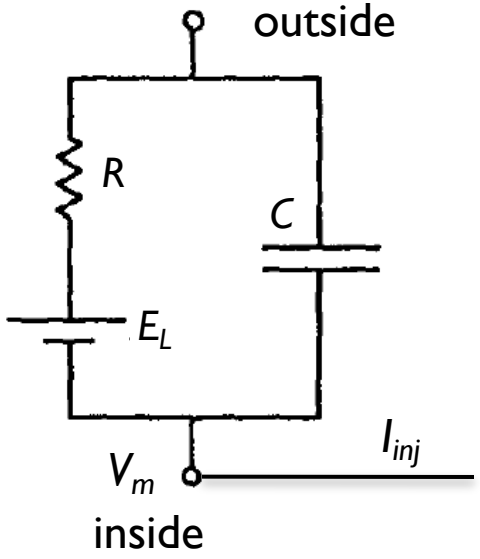
Subthreshold + spike dynamics are modelled

Hodgkin & Huxley 1952

A phenomenological model: the Leaky Integrate and Fire neuron

Only the subthreshold dynamics are modelled :





$$C\frac{dV_m}{dt} + \frac{(V_m - E_L)}{R} = I_{inj}$$

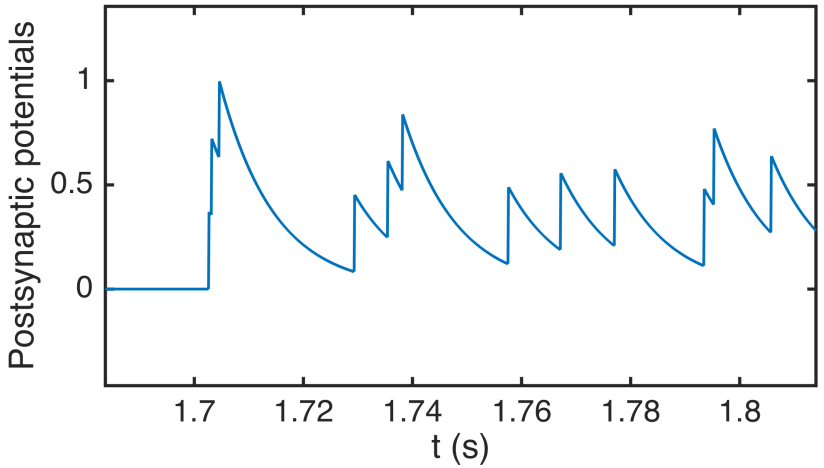
$$\tau\frac{dV_m}{dt} = E_L - V_m + RI_{inj} \qquad \text{with} \qquad \tau = RC$$

Membrane time constant (typically 3-100 ms)

If $V = V_t$ (threshold) then: neuron spikes and $V \rightarrow V_r$ (reset)

The LIF's response to input spikes

$$\begin{cases} \tau \dfrac{dV_m}{dt} = E_L - V_m + RI \\ RI = \tau \sum_{i,j} w_i(t_{i,j})\delta(t - t_{i,j}) \end{cases}$$

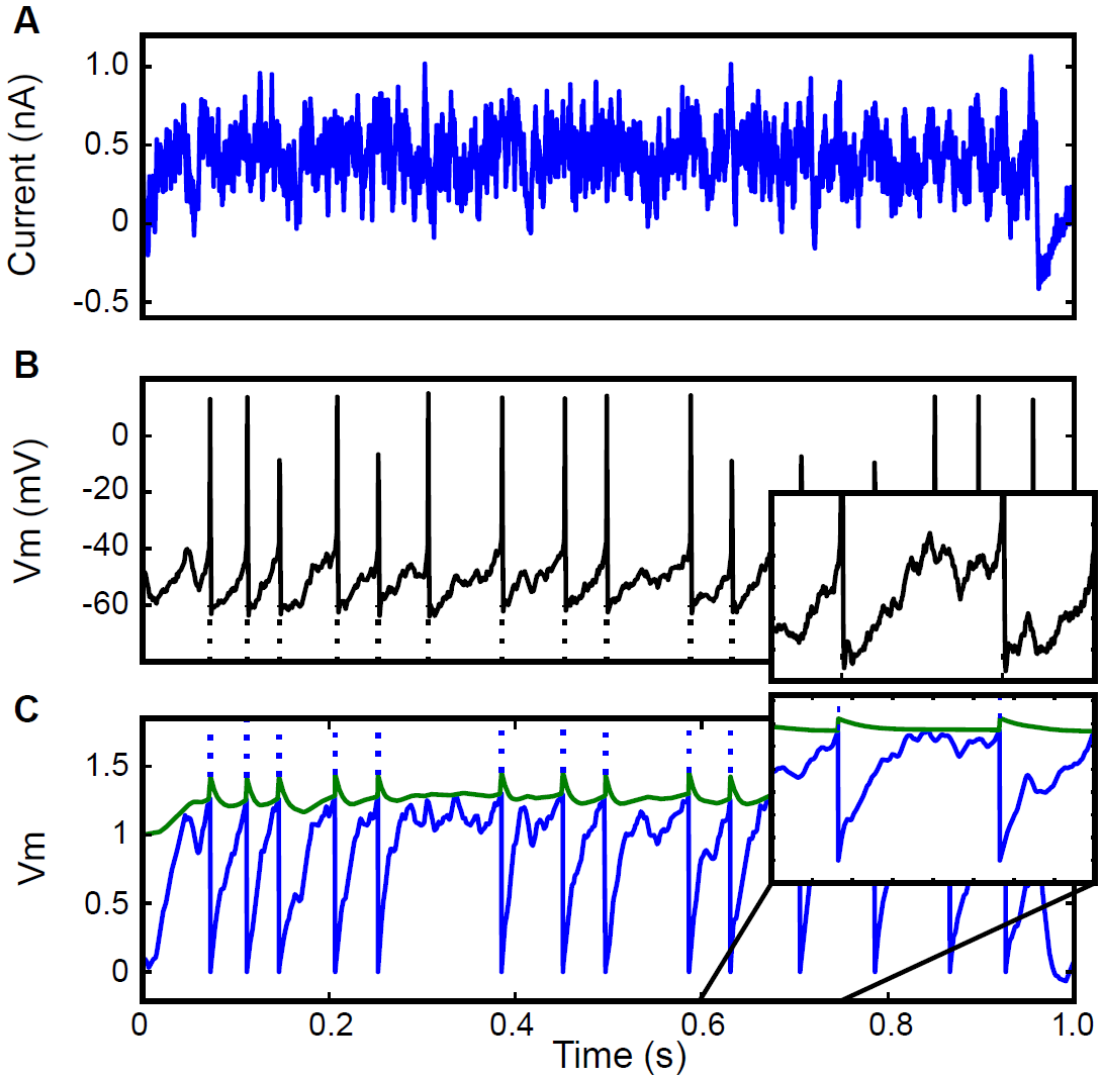


Or, equivalently:

$$\begin{cases} \text{Input spike through synapse } i\text{: } V_m \rightarrow V_m + w_i \\ \text{Otherwise: } \tau \dfrac{dV_m}{dt} = E_L - V_m \end{cases}$$

If $V = V_t$ (threshold) then: neuron spikes and $V \rightarrow V_r$ (reset)

By the way: the LIF model is not a bad model of cortical neurons



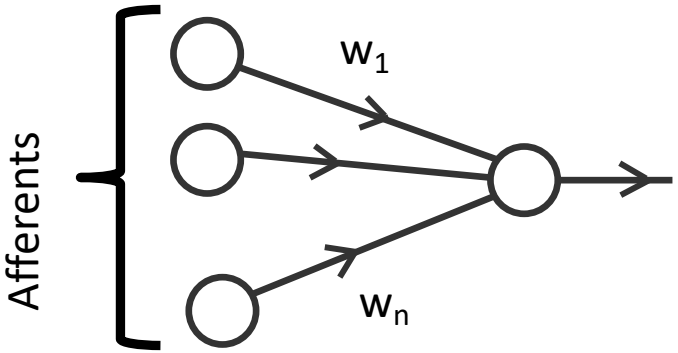
Injected current (slice)

Fast spiking cortical cell

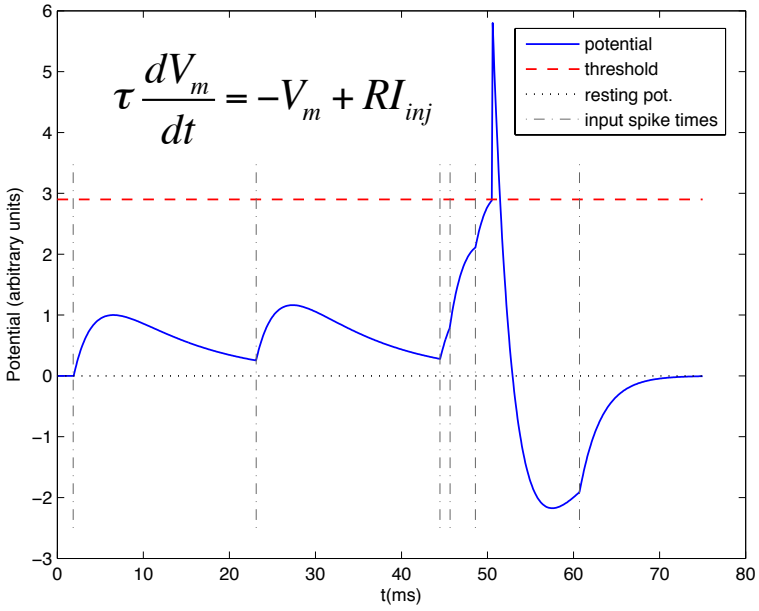
LIF model with adaptive threshold

Gerstner & Naud. Science. 2009

2 levels of description

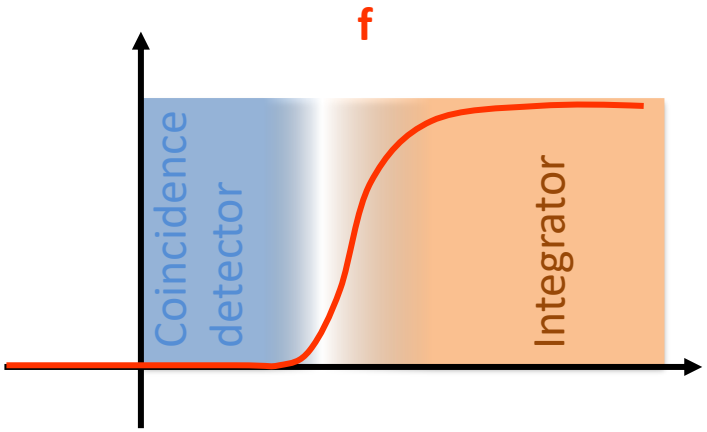


Spike-based description



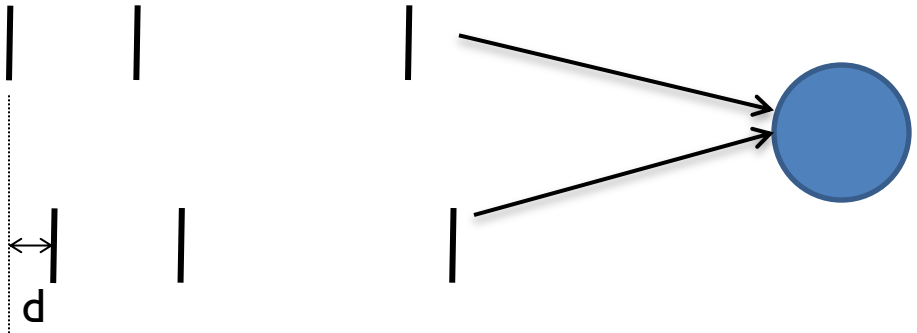
Rate-based description

Steady state



output = f(Σ w_i . $input_i$)

Neurons as coincidence detectors

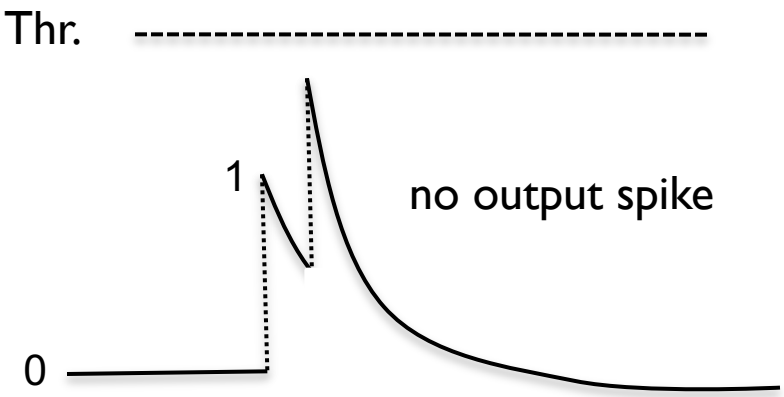


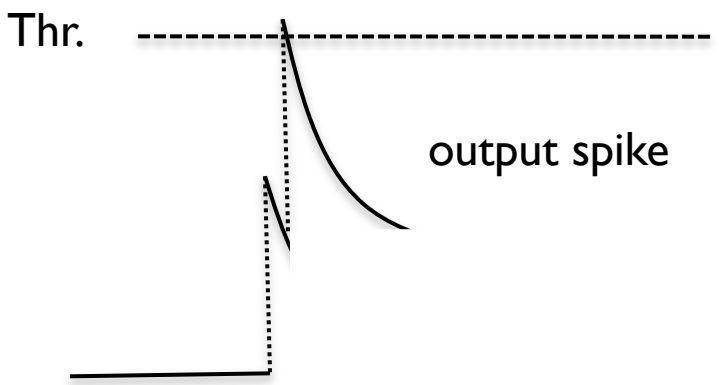
Same spike train
Time lag d

$$\tau \frac{dV}{dt} = -V$$

Input spike: $V \rightarrow V + 1$

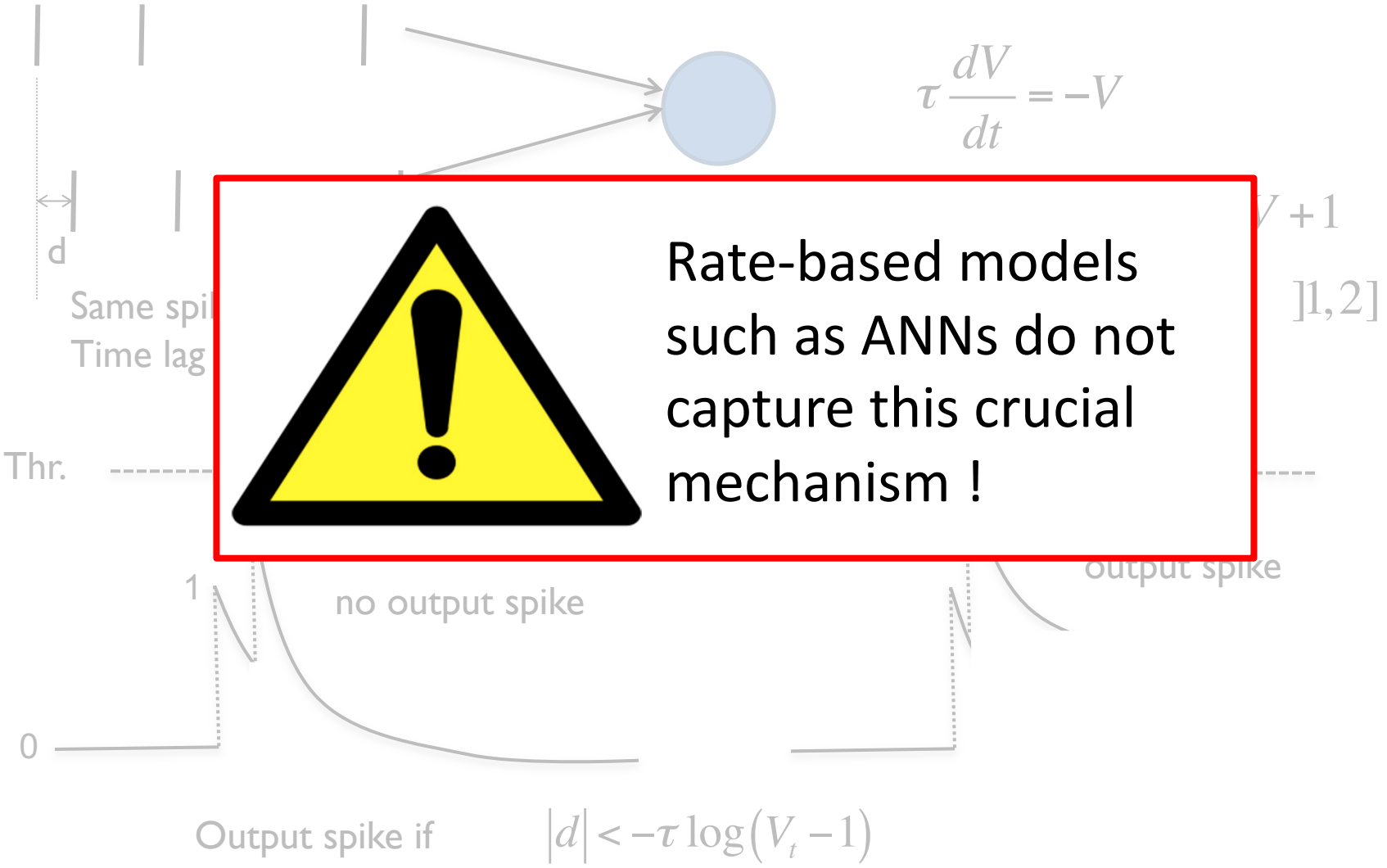
Threshold = $V_t \in]1, 2]$





Output spike if $|d| < -\tau \log(V_t - 1)$

Neurons as coincidence detectors



Rate-based models such as ANNs do not capture this crucial mechanism !

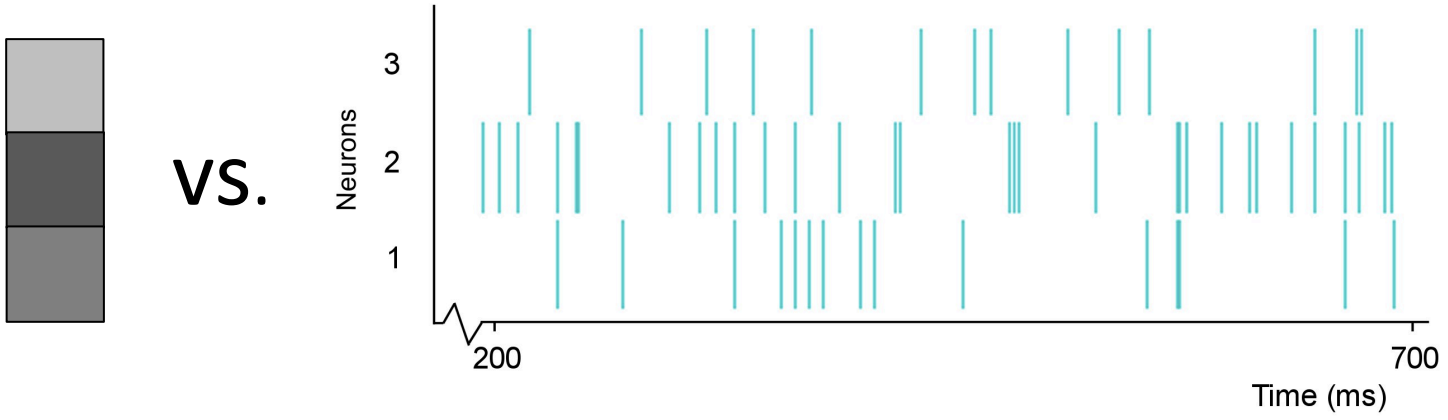
SNNs for neuroscience

Unlike Artificial Neural Networks (ANNs), Spiking Neural Networks (SNNs) can capture:

- An important neuronal function: coincidence detection
- Dynamics: transients, oscillations, etc.
- Fast processing, with few (1?) spikes per neuron

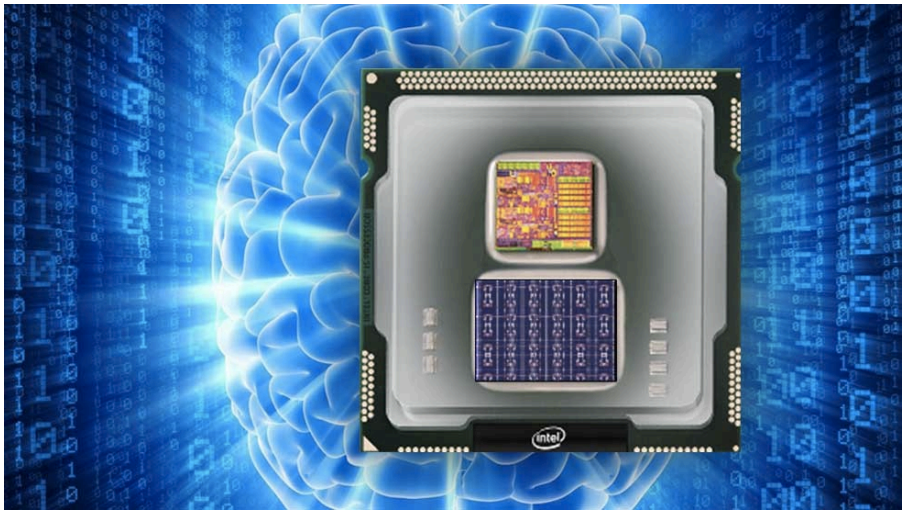
SNNs are appealing for AI

- More computational & representational power than ANNs, due to an extra dimension: continuous time.



SNNs are appealing for AI

- More computational & representational power than ANNs, due to an extra dimension: continuous time.
- Particularly appealing for dynamic inputs (sounds, videos...)
- Run best on dedicated “neuromorphic” chips (IBM True North, Intel Loihi, BrainChip Akida, etc.)



Intel Loihi (2018)

SNNs are appealing for AI

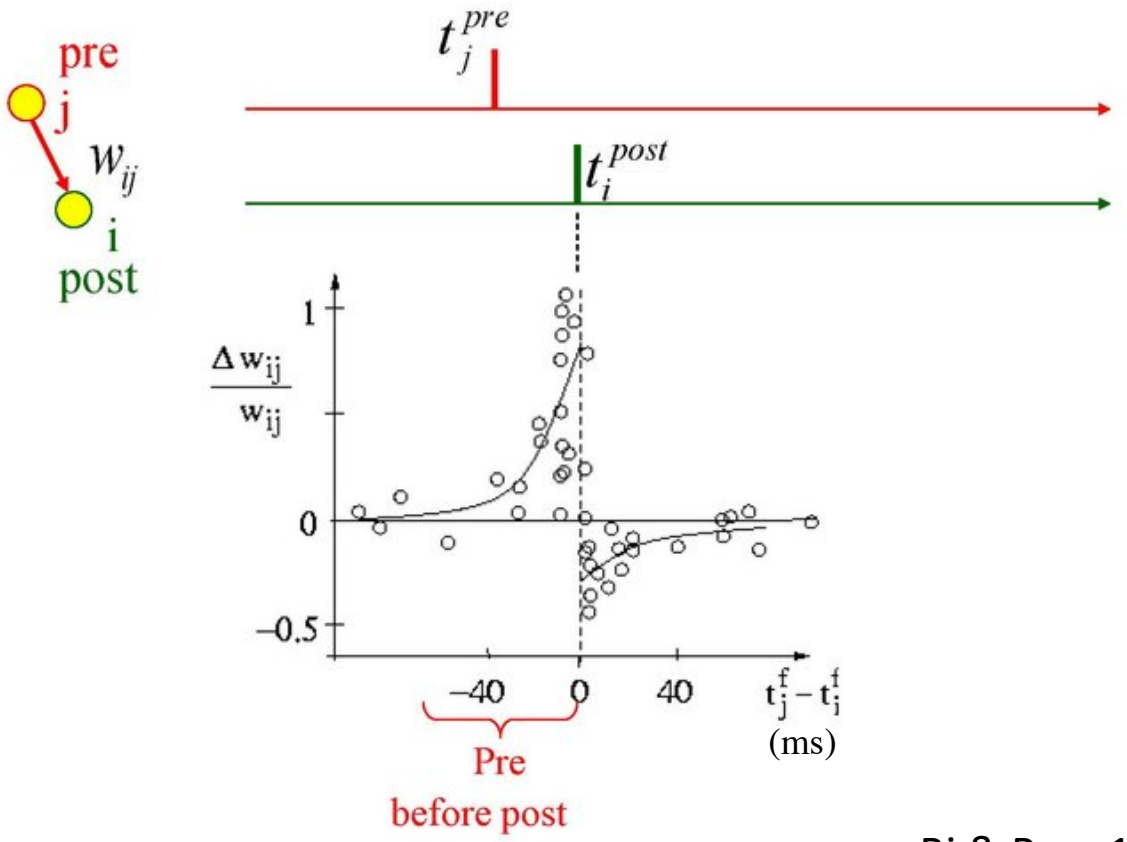
- More computational & representational power than ANNs, due to an extra dimension: continuous time.
- Particularly appealing for dynamic inputs (sounds, videos...)
- Run best on dedicated “neuromorphic” chips (IBM True North, Intel Loihi, BrainChip Akida, etc.)
- Efficient distributed event-driven & low power computation, especially if:
  - Spikes are rare (as in the brain!)
  - Connectivity is sparse (as in the brain!)

  Different from GPUs, which perform synchronous dense tensor products (required for ANNs)

Training SNNs - Outline

- Unsupervised learning
  - Spike timing dependent plasticity (STDP)
  - Application: visual feature extraction
- Supervised learning:
  - Back-propagation & surrogate gradient

Spike Timing-Dependent Plasticity



Bi & Poo, 1998

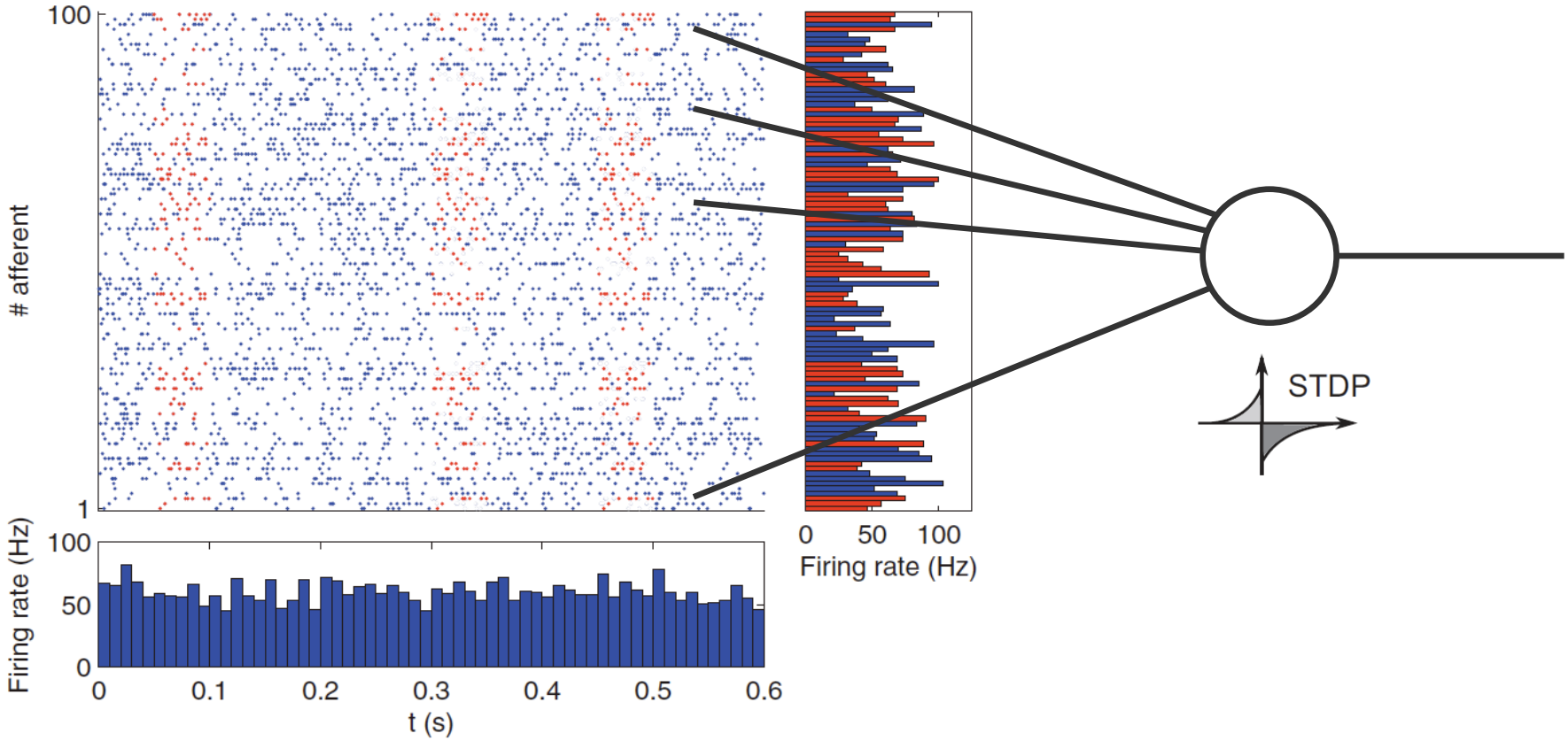
OPEN ACCESS Freely available online

PLoS one

Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains

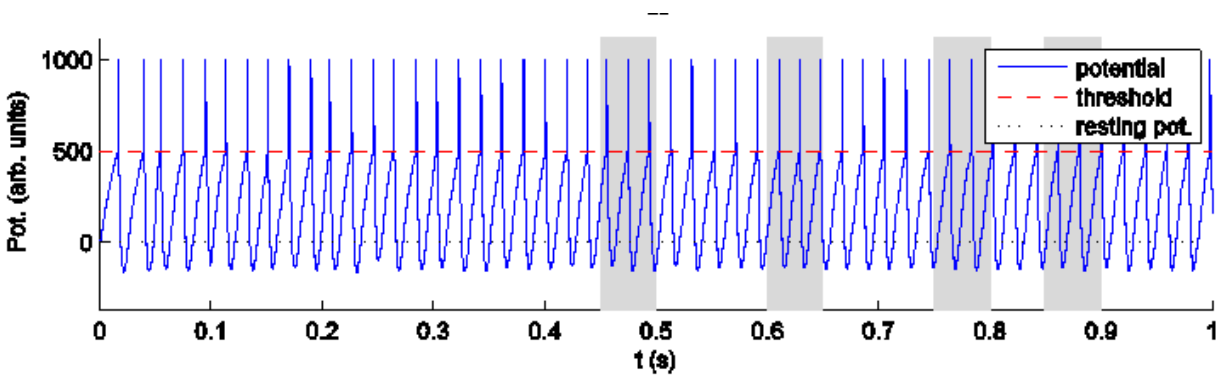
Timothée Masquelier[1,2*], Rudy Guyonneau[1,2], Simon J. Thorpe[1,2]

January 2008 | Issue 1 | e1377

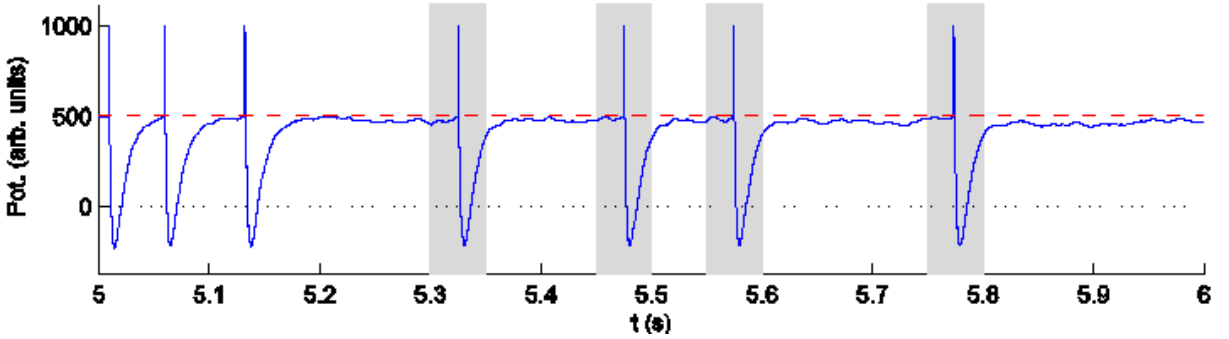


Results

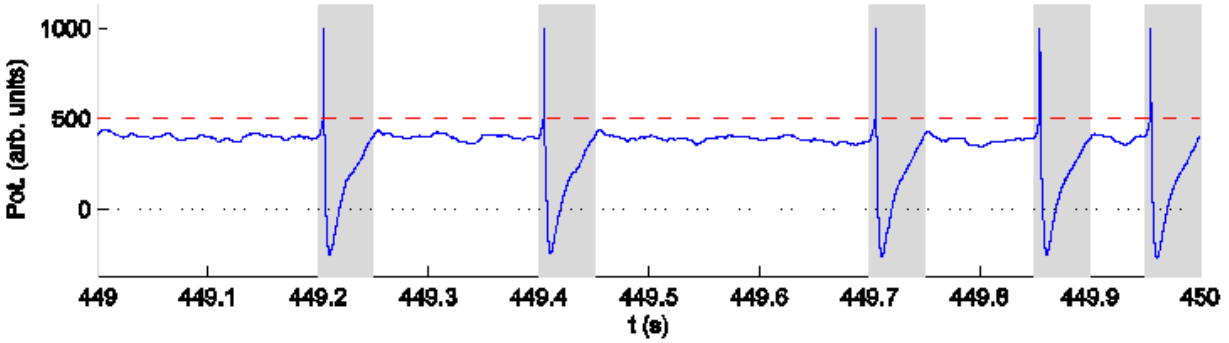
- Initial State



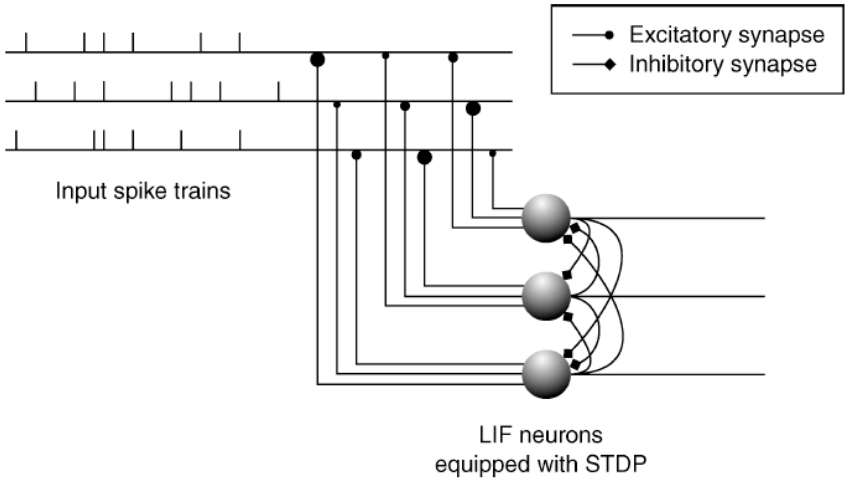
- During Learning



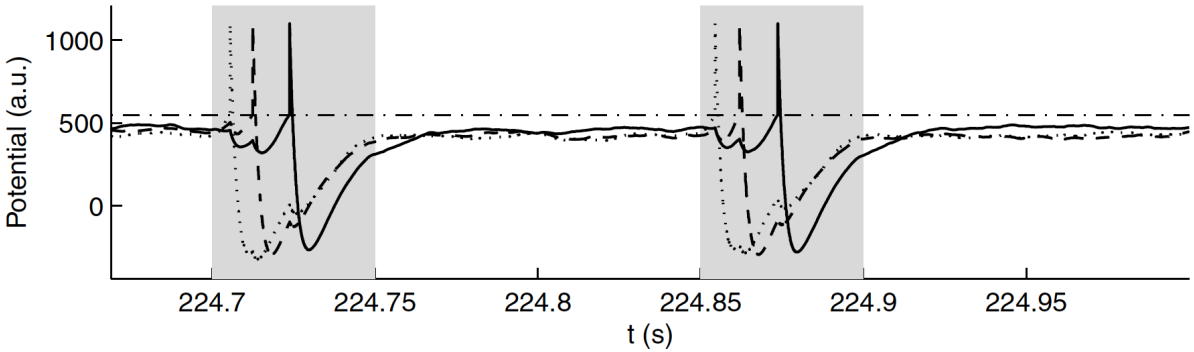
- After Learning



Learning the full pattern



Competitive learning



Neurons « stack »

Masquelier, Guyonneau & Thorpe 2009. Neural Comp

Application: visual processing

Repeating spike patterns

= repeating visual features

Speed of processing in the human visual system

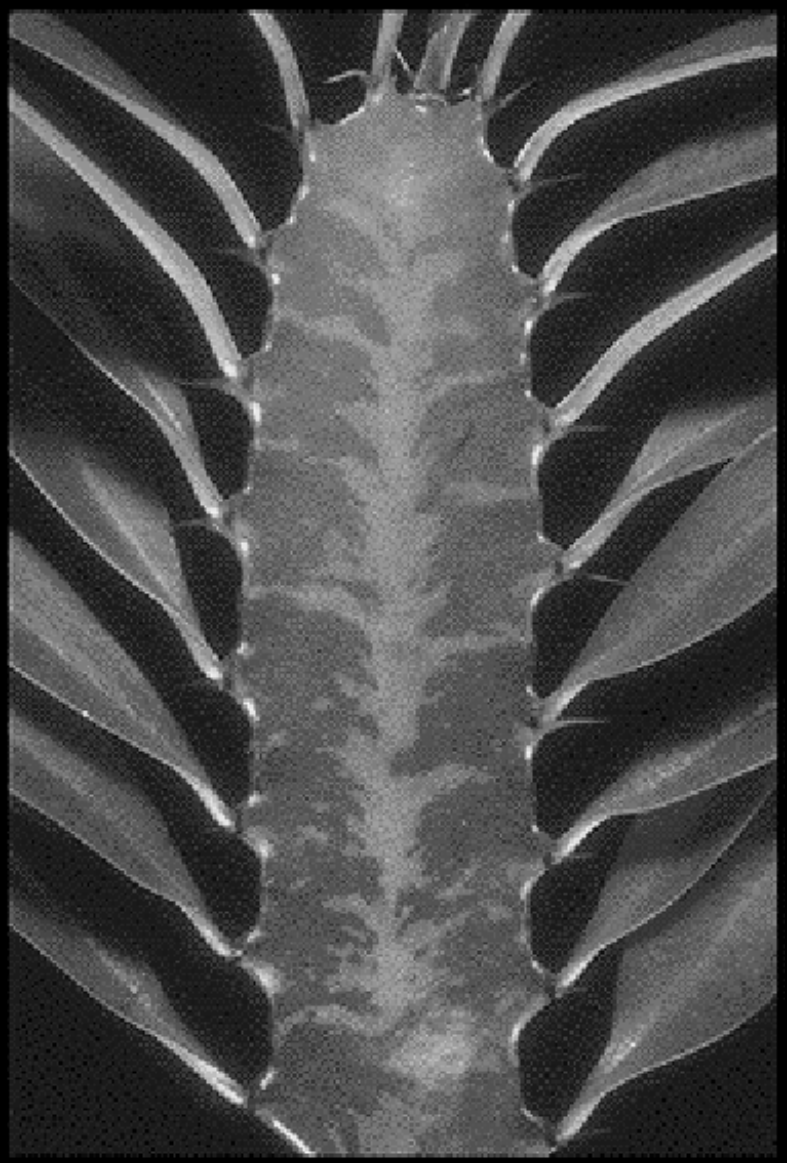
E.g. saccadic forced-choice

In which of the two images (left or right) is the animal?

+

+

+



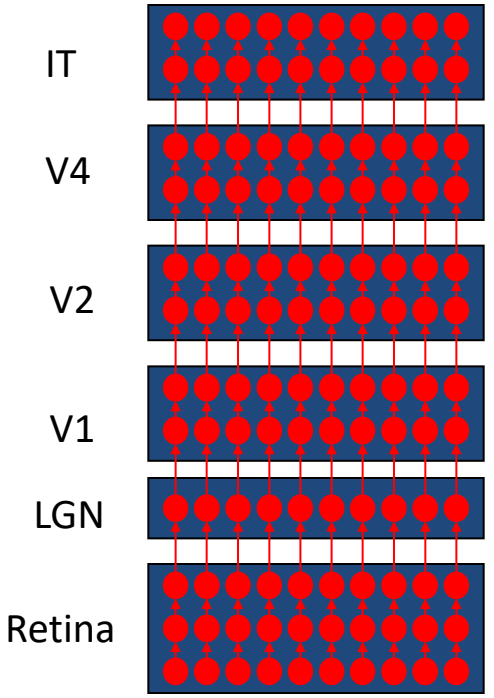
+

Visual system: speed
Psychophysics

- The eyes can move towards animals in as little as 120-30 ms!!! (Kirchner&Thorpe 2006)
- It is even faster for faces: 100ms! (Crouzet, Kirchner & Thorpe 2010)
- 20-30ms for motor delays => visual processing must occur in 100 ms or less, and even less for faces

Visual system: speed
Electrophysiology

IT responses are rapid (100 ms) & selective to complex stimuli (e.g. faces) (Oram&Perrett, 1992; Keysers et al., 2001; Hung et al., 2005)



Thorpe et al's reasoning:

Arguments

1) Roughly 10 layers
2) 10 ms per layer
3) Firing rates 0-100 Hz

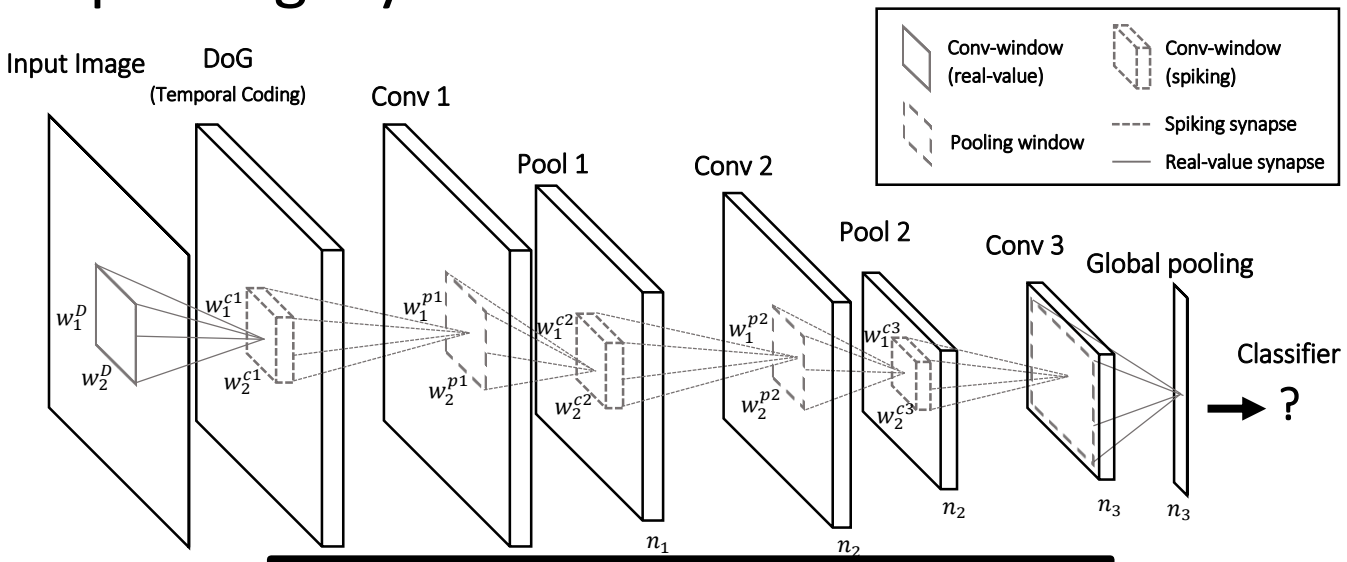
Therefore:

1) At most one spike per neuron
2) Mainly feedforward
3) The wave of first spikes does a lot!

Models of the ventral stream of the visual cortex

- Feedforward
- Convolutional (weight sharing) layers
- Max pooling layers

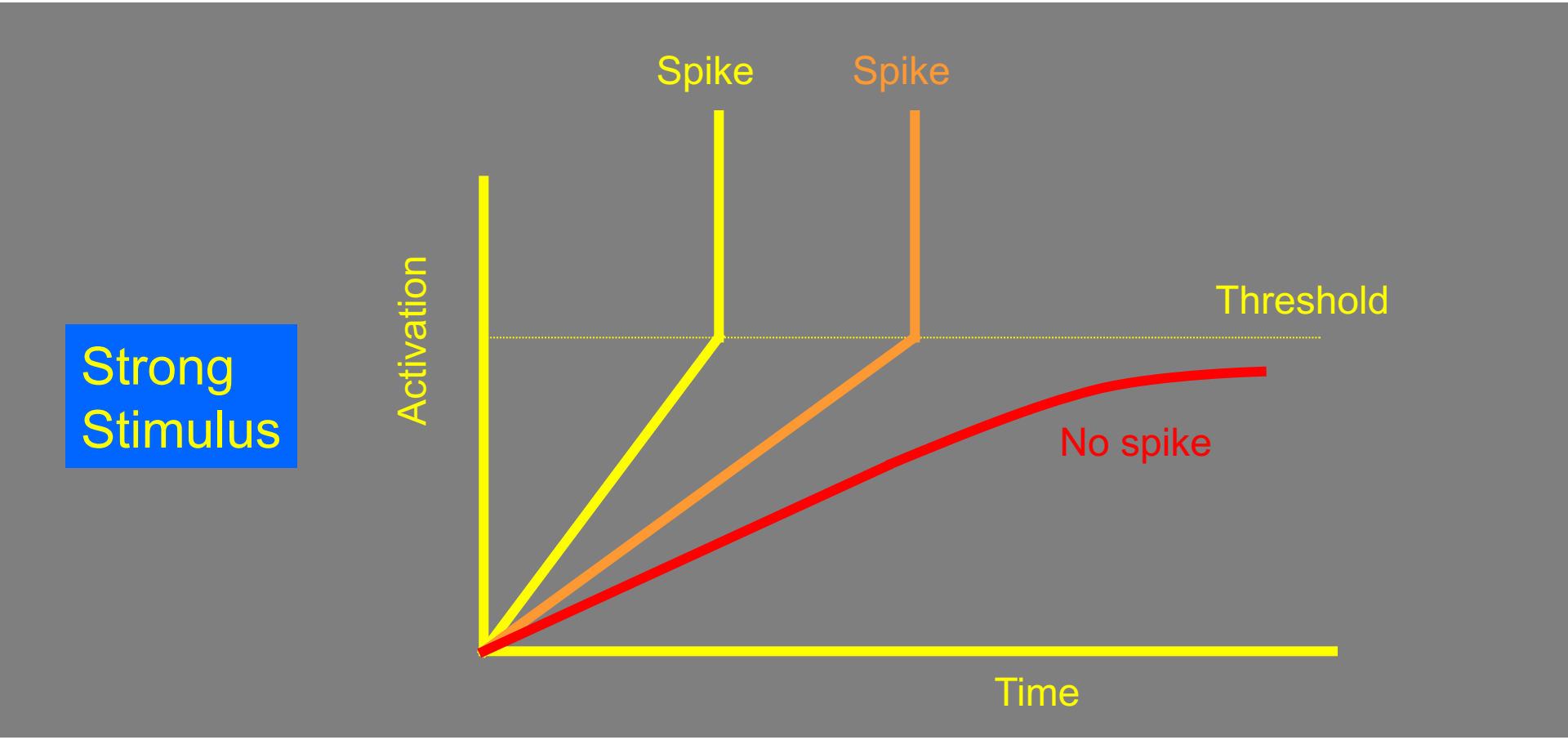
- Along the hierarchy
  - Selectivity increases
  - Invariance increases



At most one spike per neuron!

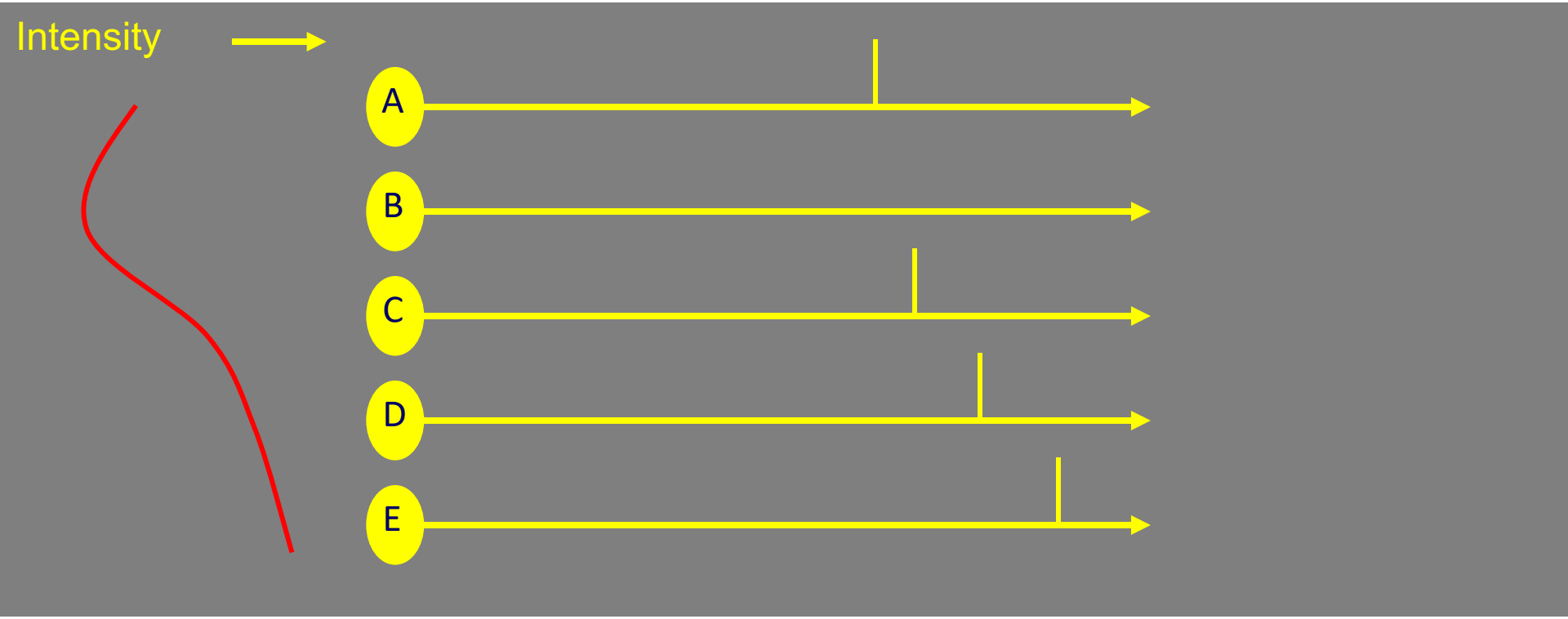
Fukushima, 1980; LeCun and Bengio, 1998; Riesenhuber and Poggio, 1999; Wallis and Rolls 1997; Rolls and Milward, 2000; Stringer and Rolls, 2000; Serre et al., 2007 + deep learning

First layer: intensity-to-latency conversion



≠ intensity-to-rate conversion (conventional view)

First layer: intensity-to-latency conversion

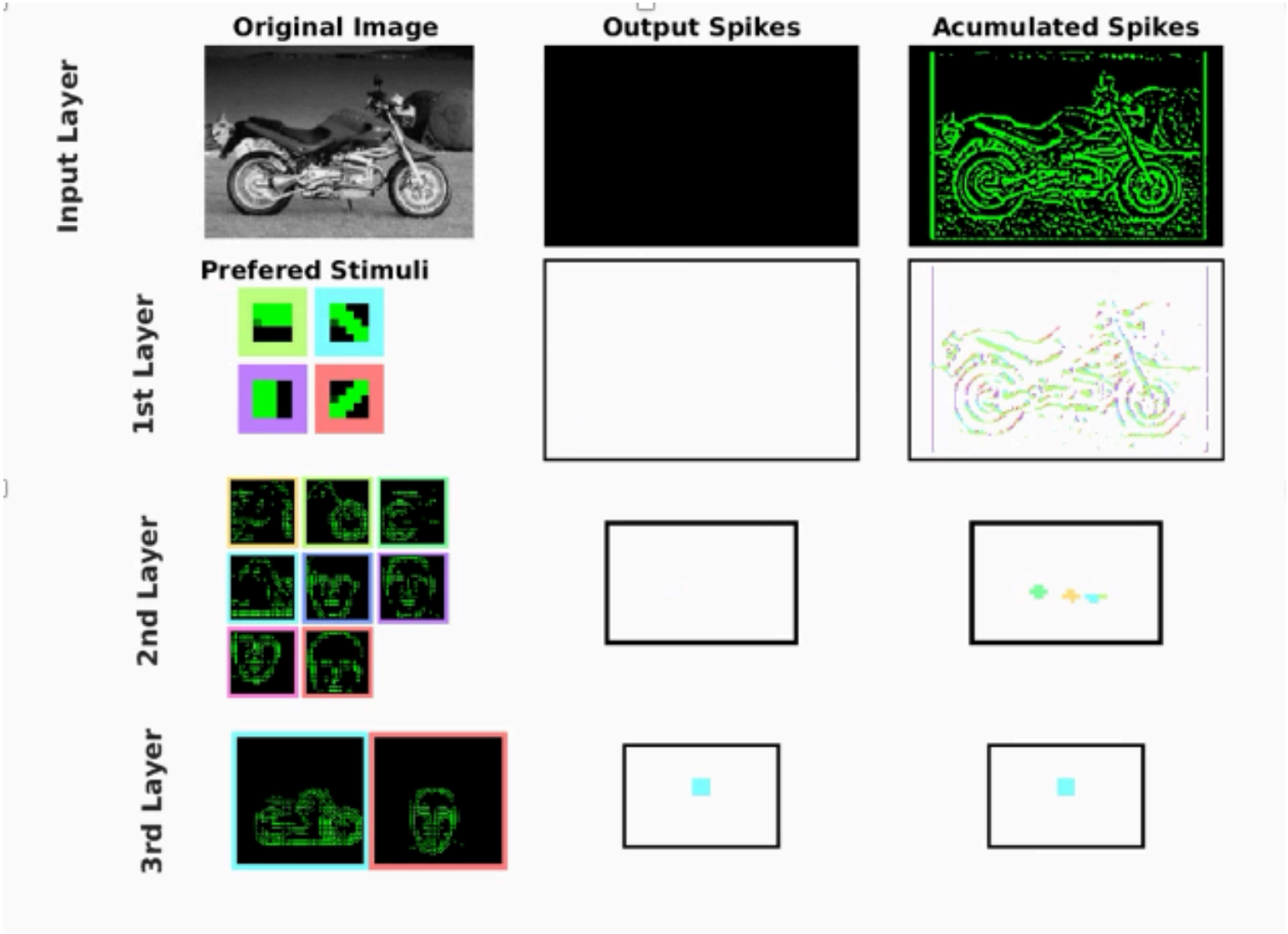


Spike waves

STDP-based deep feature extraction

Kheradpisheh, Ganjtabesh, Thorpe & Masquelier 2018. Neural Networks

STDP-based deep feature extraction



Kheradpisheh, Ganjtabesh, Thorpe & Masquelier 2018. Neural Networks

Discussion

AI:

Accuracy does not match (yet?) deep learning but:

- Energy efficient (sparse coding)
- Hardware friendly
- STDP is a local rule
- Online, on-chip, learning
- (Mostly) unsupervised learning
- Only a few tens of labeled examples needed per category

Neuroscience:

Our proposal is compatible with

- The temporal constraints (object recognition is fast in primates)
- The fact that we learn mostly by observing the world, in an unsupervised way

Training SNNs - Outline

- Unsupervised learning
  - Spike timing dependent plasticity (STDP)
  - Application: visual feature extraction
- Supervised learning:
  - Back-propagation & surrogate gradient

What is backprop?

- Supervised learning in feedforward ANNs
- Fitting algorithm: the cost function L is the distance between actual and desired activations in the last layer (e.g. MSE, cross-entropy).
- Optimal weights are found through gradient descent:

$$\Delta w_{ji} = -\eta \frac{\partial L}{\partial w_{ji}}$$

- All gradients can be computed recursively (backward) using automatic differentiation (backprop = short for "backward propagation of errors")

Backprop's tour de force

- Solves the credit assignment problem (i.e. what should the hidden layers do?)
- Optimize features & classifier jointly
- Nb of layers is arbitrary: opens the door for very deep nets

$\Rightarrow$This motivated us (and others!) to adapt backprop to SNNs

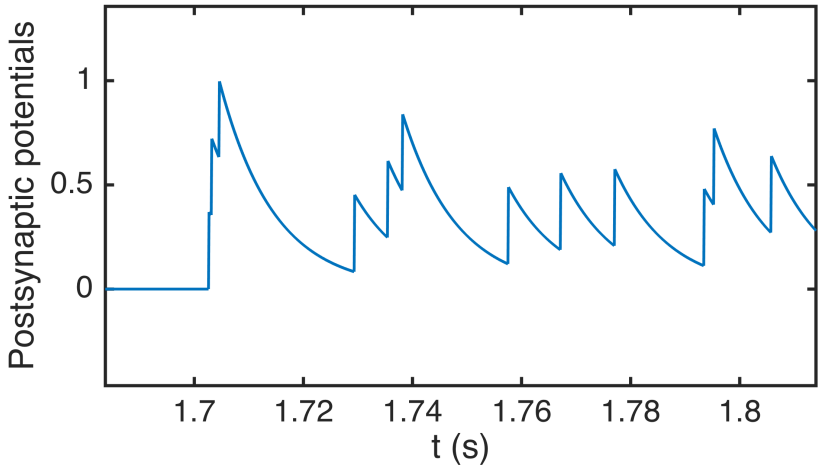
Surrogate Gradient Learning (SGL)

In a nutshell:

- Train SNNs with backprop through time
- Time is discretized
- The firing threshold causes optimization issues => use a “surrogate gradient”
- In practice: use automatic differentiation (e.g. PyTorch, TensorFlow)

Leaky integrate-and-fire neuron
Continuous time

$$\begin{cases} \tau \dfrac{dV}{dt} = -V + I \\ I = \tau \sum_j w_j \sum_k \delta(t - t_{j,k}) \end{cases}$$



Or, equivalently:

$$\begin{cases} \text{Input spike through synapse } j\text{: } V \to V + w_j \\ \text{Otherwise: } \tau \dfrac{dV}{dt} = -V \end{cases}$$

If $V = Threshold$ then an output spike is fired and $V \to 0$

Leaky integrate-and-fire neuron
Discrete time

$$V[n] = V(n\Delta t)$$

Input spikes (in the time bin)

$$V[n+1] = \beta V[n] + \sum_j w_j S_j^{\text{in}}[n] \qquad \text{(leaky integration)}$$

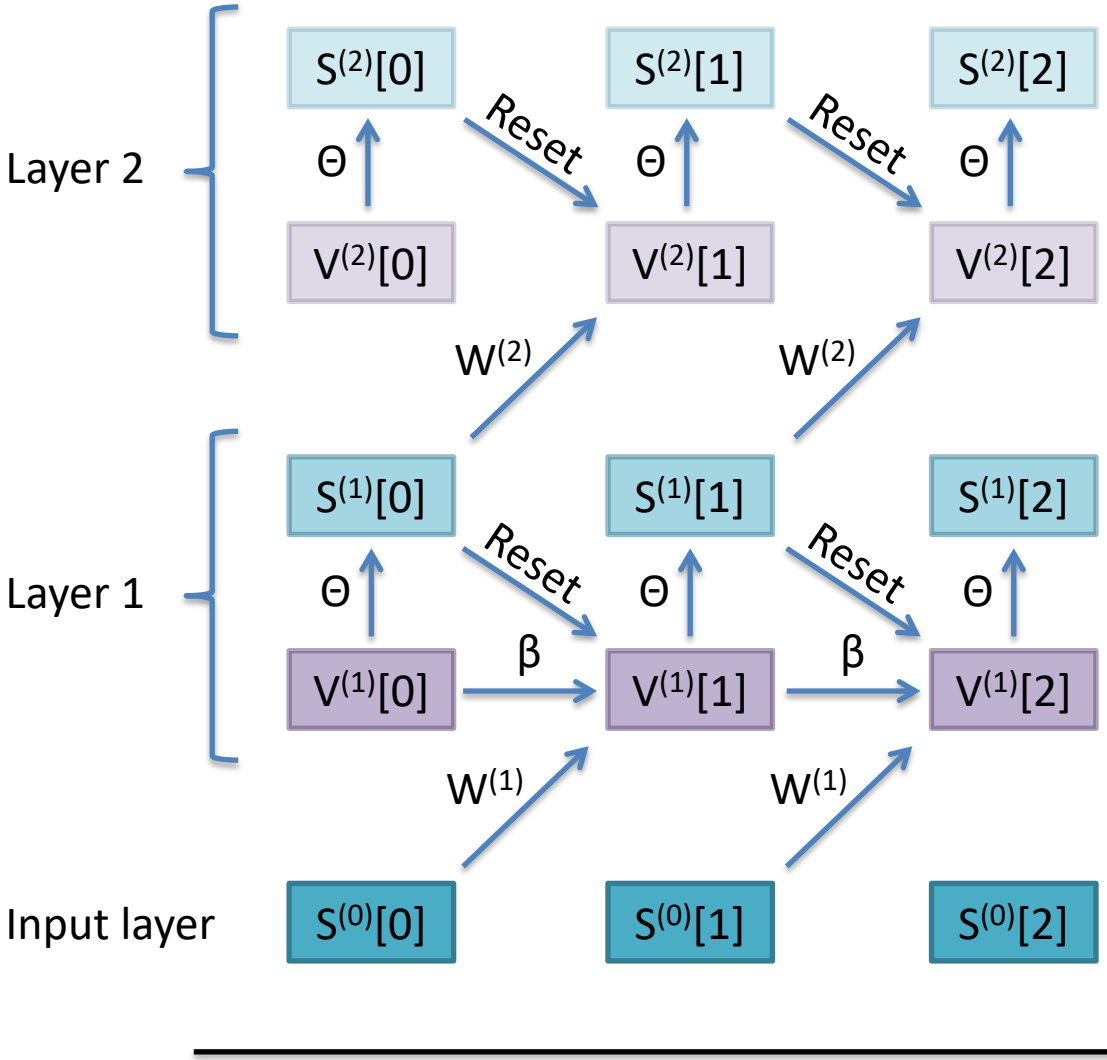
Leak

$$\beta = \exp(-\Delta t / \tau)$$

$$S^{\text{out}}[n+1] = \theta(V[n+1] - Threshold) \qquad \text{(output spikes)}$$

$$V[n+1] = (1 - S^{\text{out}}[n+1])\ V[n+1] \qquad \text{(reset)}$$

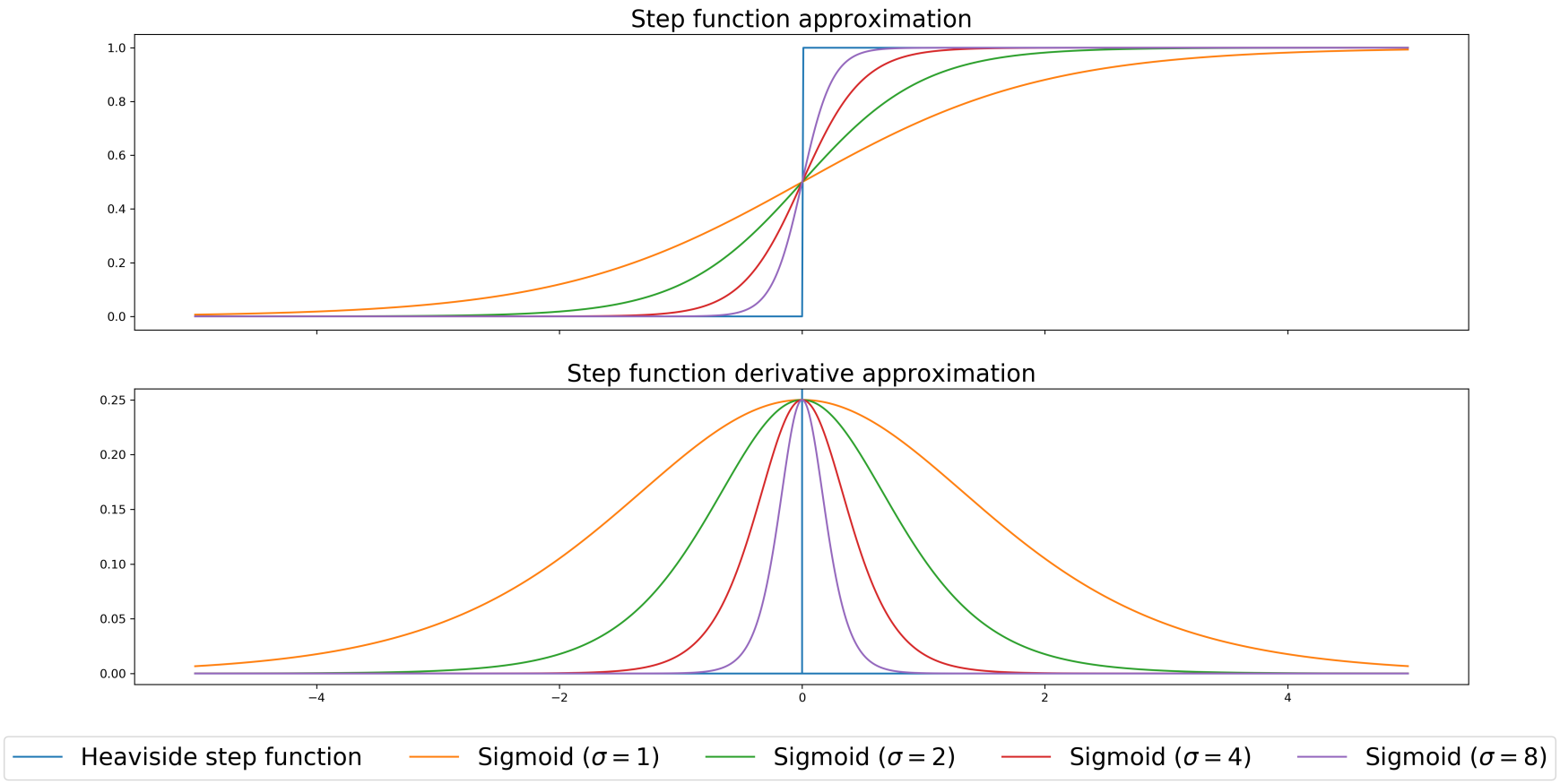
Backprop through time (BPTT)



1) Unroll the network to get a feedforward network

2) Train it with backprop

Surrogate gradient for Θ



Neftci, Mostafa & Zenke (2019) Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Mag*

Training SNNs with BPTT - Pros

- Solves the spatial & temporal credit assignment pb
- Handles:
  - Deep networks
  - Dense & conv. layers
  - Delays
  - Trainable τ
- Agnostic about rate vs. temporal coding (≠S4NN, conversion)
- Regularization can encourage sparse activity, fast processing, etc.
- In practice: SNNs can be trained using PyTorch/TensorFlow
  - Automatic-differentiation
  - SOTA gradient descent algorithms (e.g. Adam)
  - CPU(s) / GPU(s)
  - Large community

Training SNN with BPTT - Cons

- Memory hungry (linear with nb of time steps).
- Does not take advantage of highly sparse tensors.
- Not hardware friendly (but see Neftci et al.)
- Not biologically plausible (but see E-prop)