

SPIKING NEURAL NETWORKS

Michele Rossi

michele.rossi@unipd.it

Dept. of Information Engineering

Dept. of Mathematics



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Bibliography

- [SNN-Book-2014] Wulfram Gerstner, “Neuronal Dynamics: from Single Neurons to Networks and Modes of Cognition,” *Cambridge University Press*, Sept. 2014.
- [SPM-2019] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke, “Surrogate Gradient Learning in Spiking Neural Networks: bringing the power of gradient-based optimization to spiking neural networks,” *IEEE Signal Processing Magazine*, 2019.
- [ProcIEEE-2023] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, Wei D. Lu, “Training Spiking Neural Networks Using Lessons From Deep Learning,” *Proceedings of the IEEE*, 2023.
- [SNNtorch] Jason K. Eshraghian, “Tutorial 5: Training Spiking Neural Networks with SNN Torch,” *online*, 2021.

https://snntorch.readthedocs.io/en/latest/tutorials/tutorial_5.html

Outline

- Introduction & motivation
- Neuron model [SNN-Book-2019]
 - Leaky Integrate and Fire (LIF) model
 - Discrete time neuron model
- Neural network model [SPM-2019, ProcIEEE-2023]
 - Continuous and discrete time
- Supervised training [SNN-SPM-2019]
 - Surrogate gradient descent
- Conclusions and open research avenues



Introduction & motivation

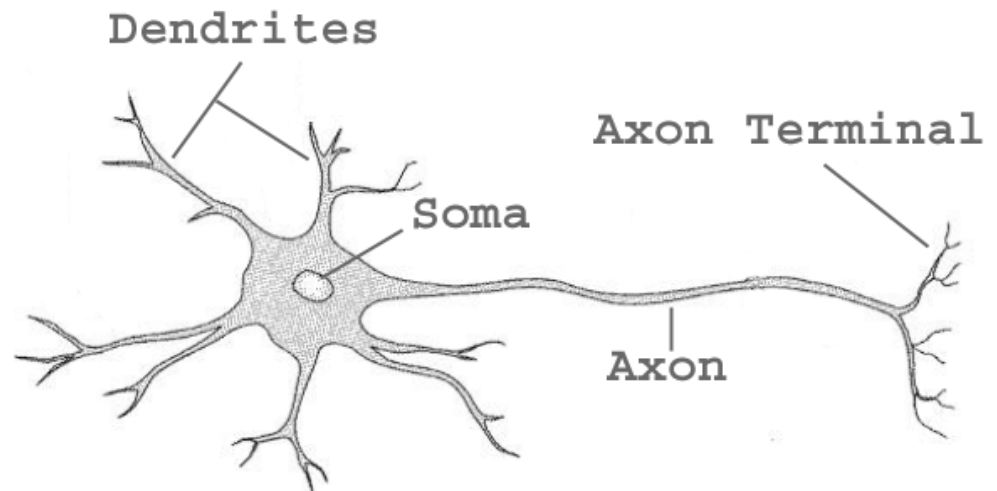
- Neural networks **are becoming huge** (billions of pars) and their training is **highly energy demanding**
- State-of-the-art solutions (e.g., transformer based) rely on convolutions and “correlations” to capture the structure and the relations underpinning data
- Current optimization approaches involve
 - **Optimizations:** weight quantization, pruning, etc.
 - **Architectural designs:** reservoir computing, extreme learning machines, etc.
- Such approaches **do not address the core of the problem:** current architectures inherently perform **dense processing** of data in both time and space

Some approaches to improve efficiency of deep learning models

- New designs + optimizations (quantization, pruning, etc.)
- Continual learning (model update)
- Unsupervised or semi-supervised learning (data efficiency)
- Reduce data transfer & communication
- Use of analog & mixed signal circuitry
- Sparsity of activation & connections

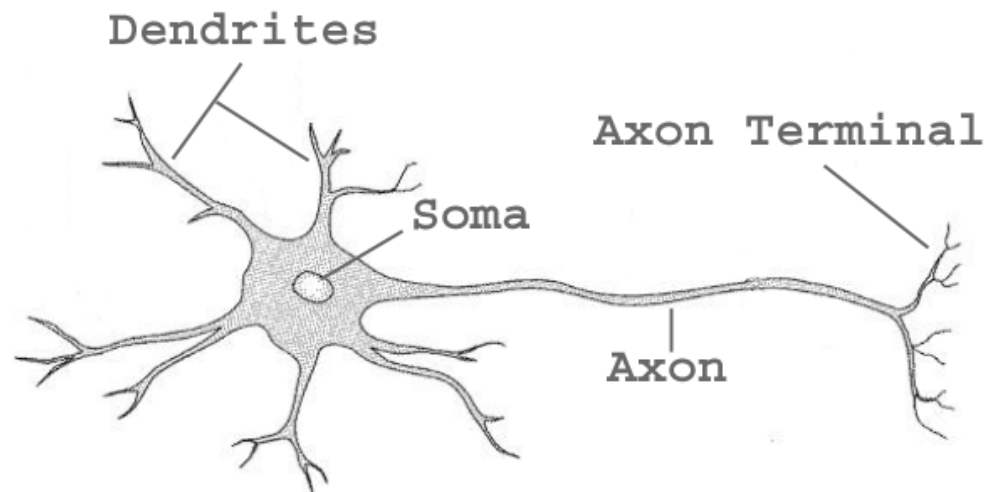
Biological neurons

- Neuron
 - basic elements in the brain
 - **soma**: “central processing unit” performs a **nonlinear** processing step. If the total input arriving at the soma exceeds a threshold → an output signal is generated



Biological neurons

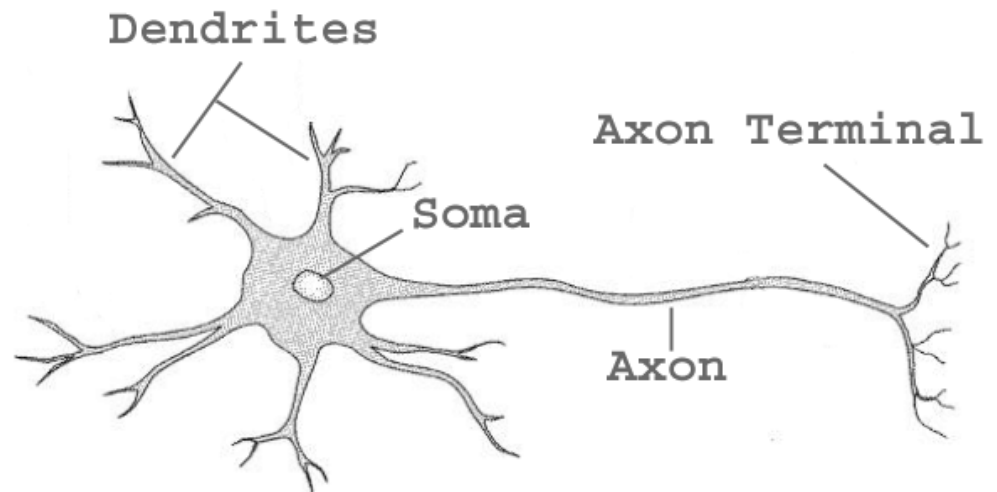
- **Neuron**
 - basic elements in the brain
 - **axon**: it is the “output device”, it takes the generated output and delivers it to other neurons (in vertebrate cortex, to more than 10^4)
 - **synapse**: it corresponds to the junction between two neurons



Biological neurons

- **Connectivity**

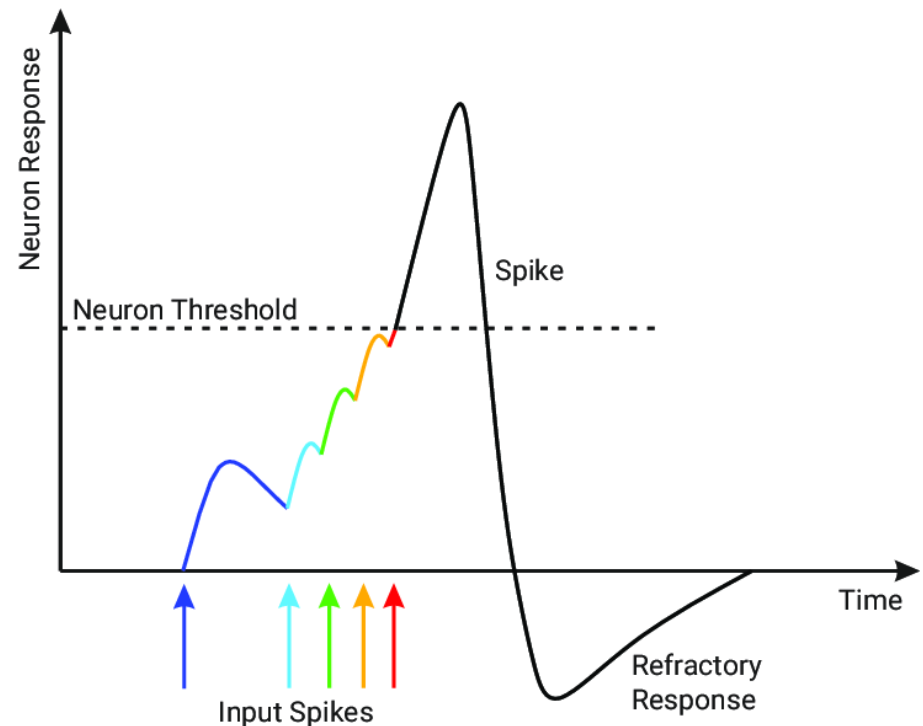
- **axons** usually is very short and ends in the neuron neighborhood, but it can occasionally stretch over several centimeters → to reach other neurons in other areas of the brain



Spikes

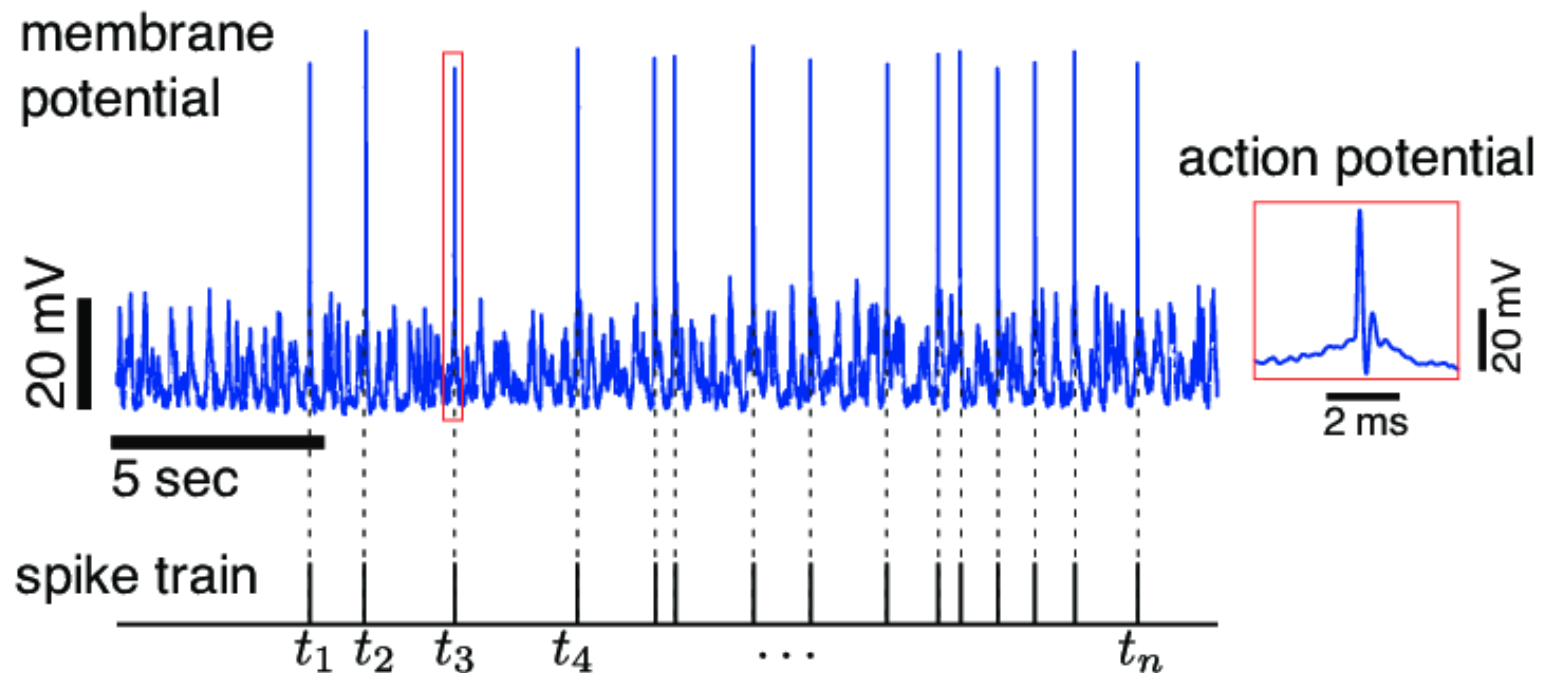
- Spikes

- Neuron output signal consists of spikes \rightarrow short pulses (1-2 ms) with a typical amplitude of 100 mV. Their form does not change as they propagate along the axon



Spike trains

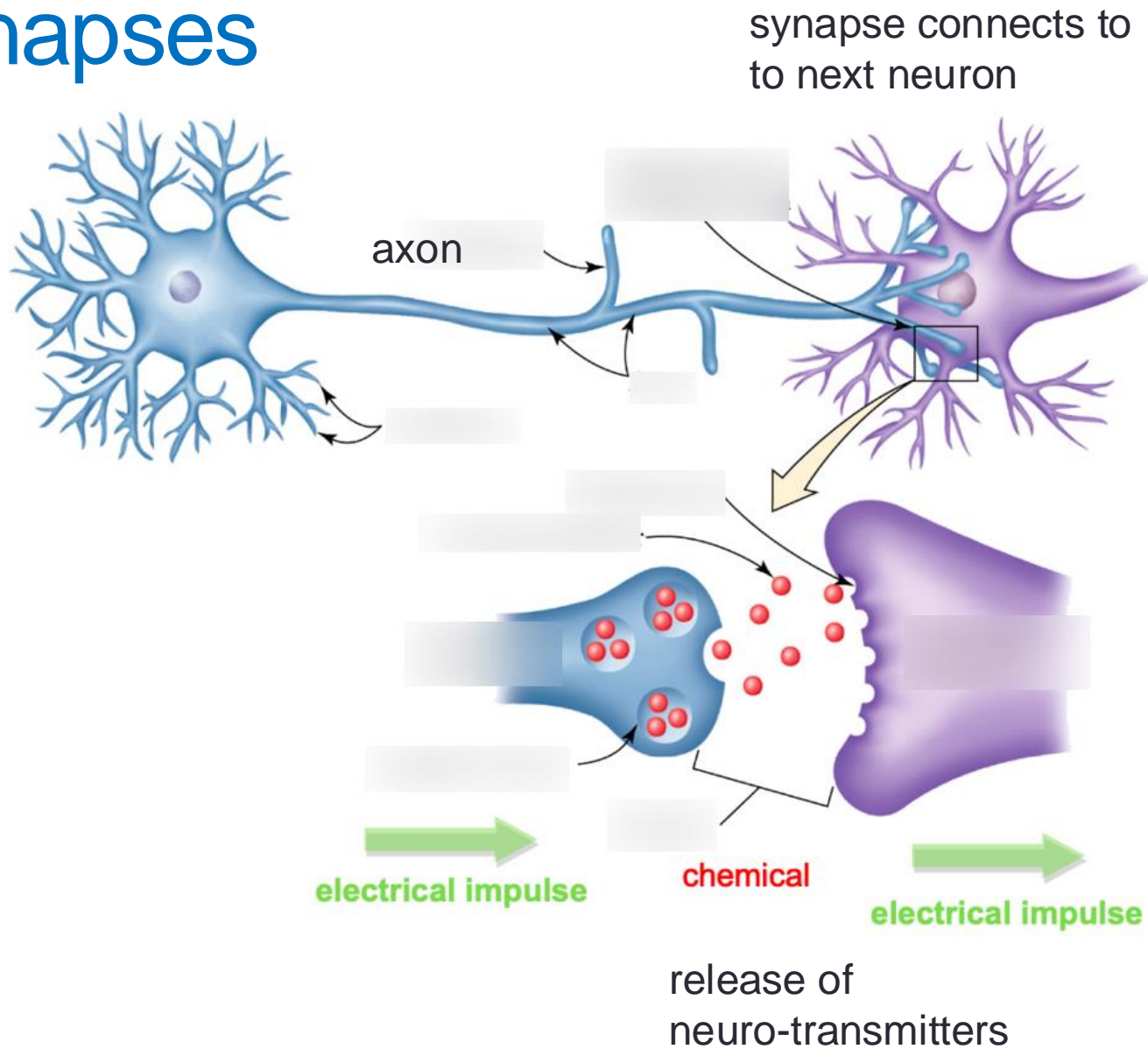
- Spikes are generated when the potential of a neuron membrane **exceeds a threshold**. Spikes in a train are well separated (there is a refractory period after the emission of a spike during which further spikes **are inhibited**)



Synapses

- When an action potential (spike) arrives at the synapse of a connected neuron → it triggers a complex chain of biochemical processing steps
- This leads to the release of neurotransmitters from the pre-synaptic site, as soon as these molecules arrive to the post-synaptic site, they will be detected by specialized receptors and lead to an opening of specific channels causing ions from the extra-cellular fluid flow into the cell
- The ion influx changes the **membrane potential (Voltage)** at the post-synaptic site so that the chemical signal is translated into an electrical response

Synapses



Membrane potential $u(t)$

- Is the potential difference between the interior of the neuronal cell and its surroundings
- Without any input, the neuron is at rest and has a rest membrane potential u_{rest} (about -65 mV)
- After the arrival of a spike, the potential changes
 - **Positive change:** excitatory synapse
 - **Negative change:** inhibitory synapse
- Our interest
 - **Time evolution of the membrane potential $u_i(t)$** of neuron i
 - In resting conditions, it holds $u_i(t) = u_{\text{rest}}$
 - If presynaptic neuron j fires its spike at $t=0$, we get a response

$$\epsilon_{ij}(t) := u_i(t) - u_{\text{rest}}$$

MODELING NEURON DYNAMICS

The Leaky Integrate and Fire (LIF) Model

Leaky Integrate and Fire Model

From [Gerstner2014]

“The model makes use of the fact that neuronal action potentials of a given neuron always have roughly the same form. If the shape of an action potential is always the same, then the shape cannot be used to transmit information: rather information is contained in the presence or absence of a spike. Therefore action potentials are reduced to ‘events’ that happen at a precise moment in time”

[Gerstner2014] Wulfram Gerstner, Werner M. Kistler, Richard Naud and Liam Paninski, “Neuronal Dynamics: from single neurons to networks and models of cognition,” *Cambridge University Press*, 2014.

Leaky Integrate and Fire Model

Objective: track the action potential $u(t)$ of a neuron

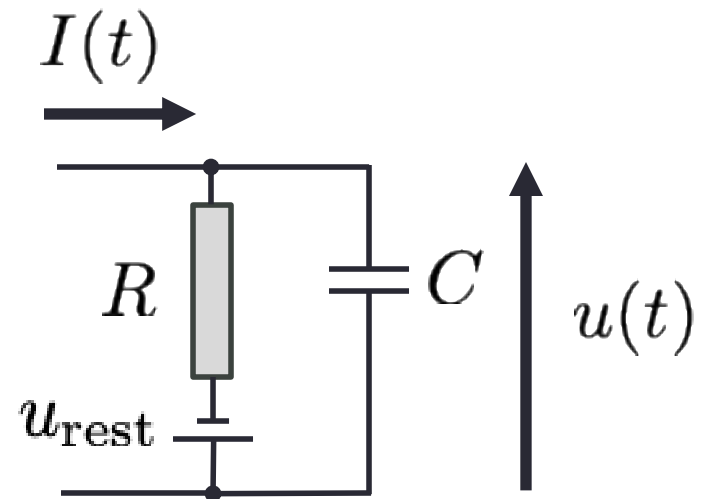
The following **RC circuit** can be used to describe the neuron behaviour

- R: input resistance
- C: neuron membrane capacitor

$I(t)$ input current (spikes)

u_{rest} membrane potential at rest

$u(t)$ membrane potential at time t



Leaky Integrate and Fire Model

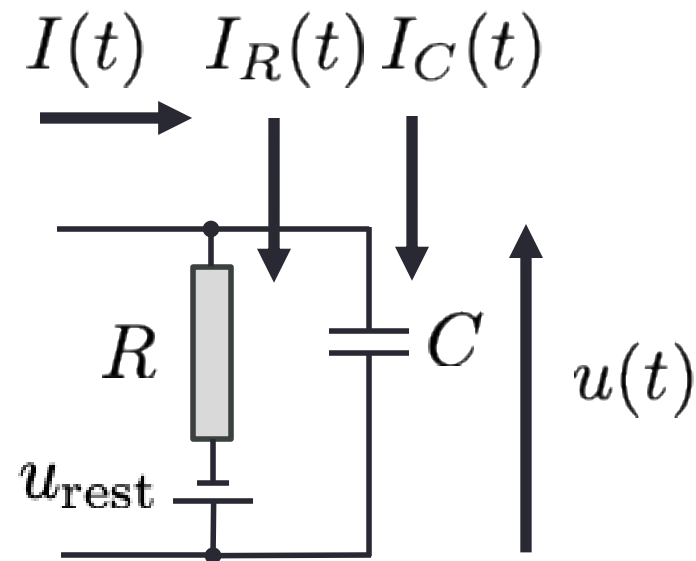
$$I(t) = I_R(t) + I_C(t)$$

$$I(t) = \frac{u(t) - u_{\text{rest}}}{R} + C \frac{du(t)}{dt}$$

$I(t)$ input (injected) current (spikes)

u_{rest} membrane potential at rest

$u(t)$ membrane potential at time t



Leaky Integrate and Fire Model

$$I(t) = \frac{u(t) - u_{\text{rest}}}{R} + C \frac{du(t)}{dt}$$

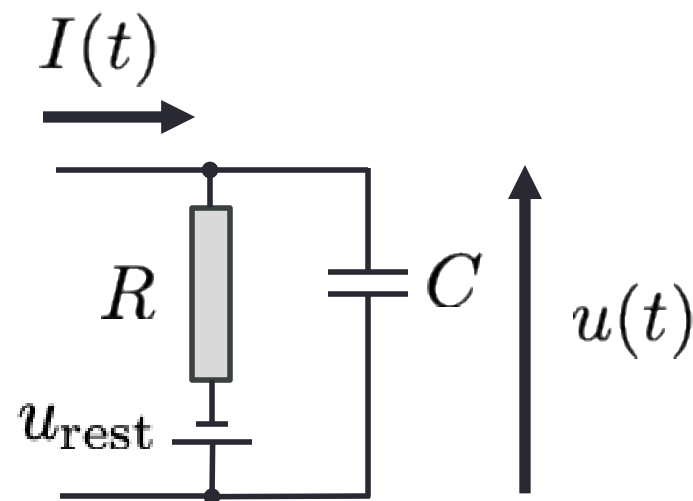
$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + RI(t)$$

$$\tau_m := RC$$

membrane **time constant**

$I(t)$

Injected current: from arriving spikes



Leaky Integrate and Fire Model

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + RI(t) \quad (1)$$

- Mathematician: this is a linear differential equation
- Engineer: the equation of a “leaky integrator”
- Neuroscientist: equation of a passive neuron membrane

Assuming that for a certain time t_0 , the injected current is $I(t)=0$ for $t \geq t_0$, and the membrane potential is $u(t_0) = u_{\text{rest}} + \Delta u$, the equation admits a closed form solution

Leaky Integrate and Fire Model

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}], \quad t \geq t_0 \quad \text{with: } I(t) = 0$$

$$u(t) - u_{\text{rest}} = \Delta u \exp\left(-\frac{t - t_0}{\tau_m}\right), \quad \text{for } t > t_0 \quad (2)$$

$$\text{with: } u(t_0) = u_{\text{rest}} + \Delta u$$

In the absence of input, $u(t)$ decays exponentially (discharging capacitor) to its resting value. The membrane time constant $\tau_m = RC$ is the characteristic time of the decay. For a typical neuron it is in the range of 10ms, which is longer than the duration of a spike (on the order of 1ms)

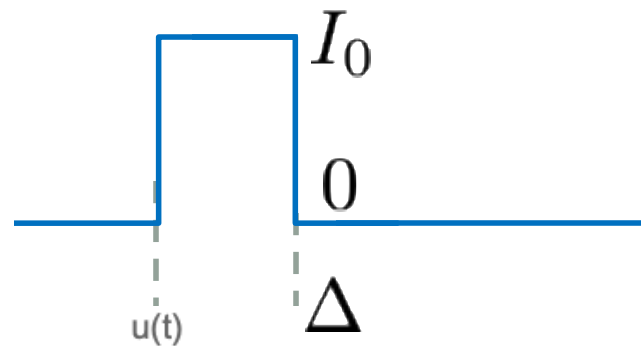
Pulse input (constant current I_0)

- At $t_0=0$ a constant current I_0 is generated, starting at t_0 and ending at $t=\Delta$
- For the sake of simplicity, we assume that $u(0)=u_{\text{rest}}$
- $u(t)$ in this case is found by integrating Eq. (1) with the initial condition $u(0)=u_{\text{rest}}$
- The solution in $0 < t < \Delta$ is – **loading capacitor**:

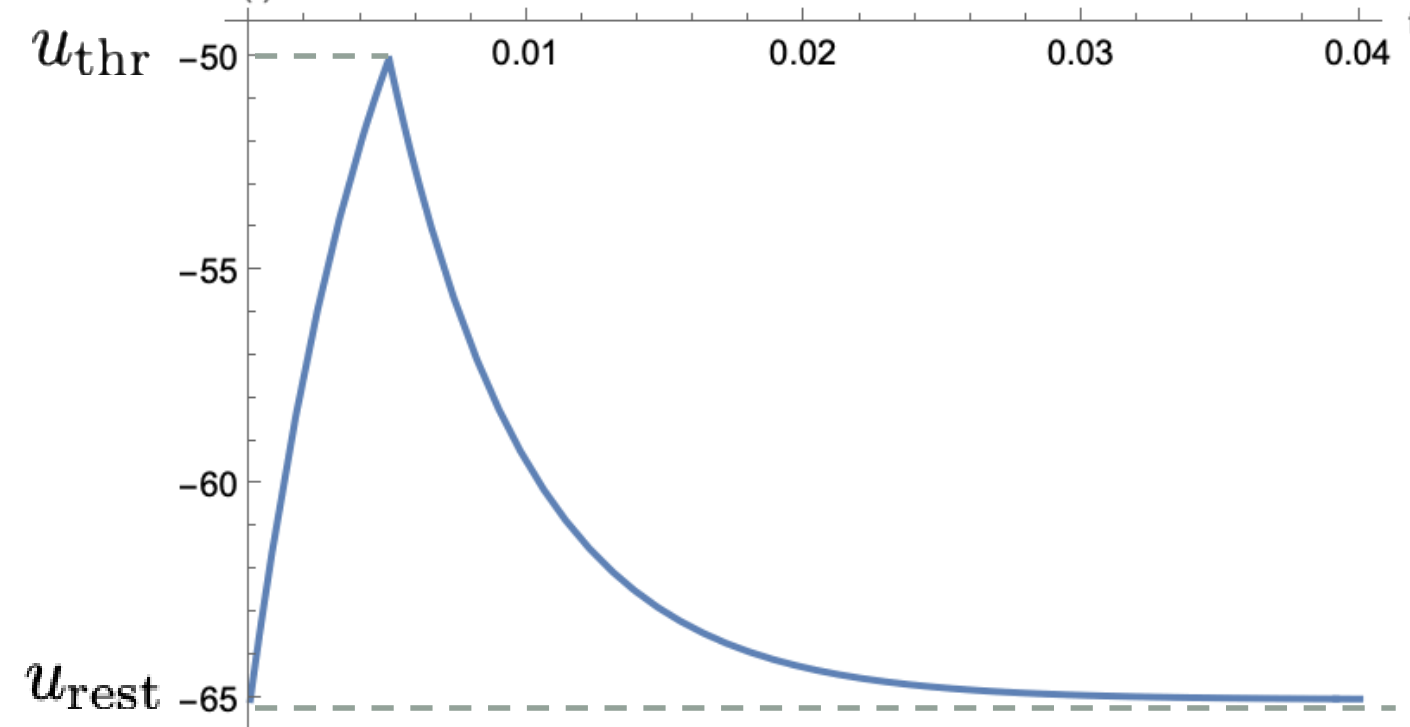
$$u(t) = u_{\text{rest}} + RI_0 \left[1 - \exp \left(-\frac{t}{t_0} \right) \right] \quad (3)$$

- After $t \geq \Delta$, the previous equation (2) applies with initial charge $u(\Delta)$ (computed using Eq. (3))

Pulse input (constant current I_0)



$$I(t) = \begin{cases} I_0 & t \in [0, \Delta] \\ 0 & \text{otherwise} \end{cases}$$



$$\Delta = 5 \text{ ms}$$

$$C = 100 \text{ pF}$$

$$R = 50 \text{ M}\Omega$$

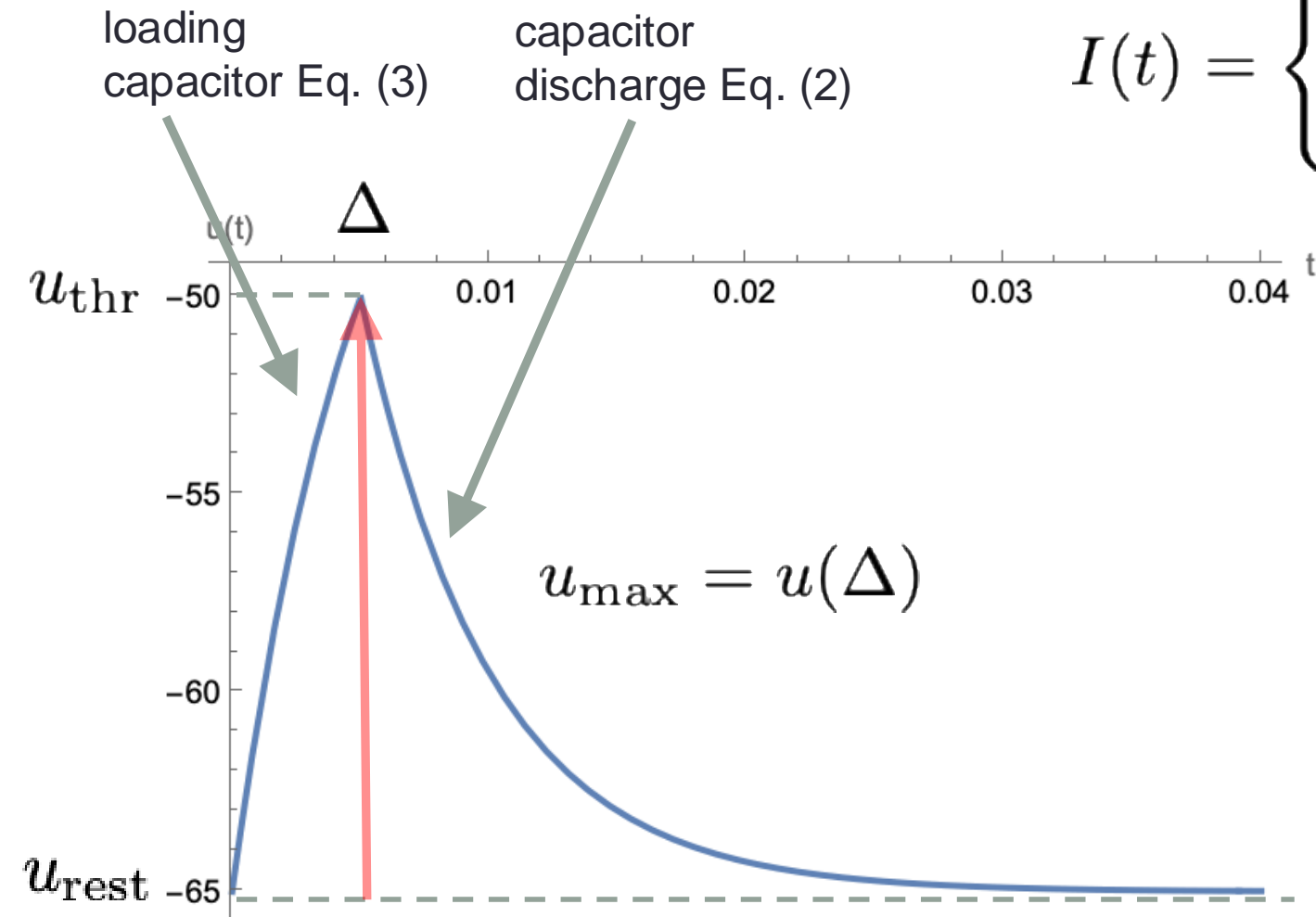
$$u_{\text{rest}} = -70 \text{ mV}$$

$$u_{\text{thr}} = -50 \text{ mV}$$

$$I_0 \simeq 0.475 \text{ }\mu\text{A}$$

Pulse input (constant current I_0)

$$I(t) = \begin{cases} I_0 & t \in [0, \Delta] \\ 0 & \text{elsewhere} \end{cases}$$



$$\Delta = 5 \text{ ms}$$

$$C = 100 \text{ pF}$$

$$R = 50 \text{ M}\Omega$$

$$u_{\text{rest}} = -70 \text{ mV}$$

$$u_{\text{thr}} = -50 \text{ mV}$$

$$I_0 \simeq 0.475 \text{ }\mu\text{A}$$

Short pulses

- For short pulses of duration $\Delta \ll \tau_m$, using Eq. (3)

$$u(\Delta) = u_{\text{rest}} + RI_0 \left[1 - \exp \left(-\frac{\Delta}{\tau_m} \right) \right] \simeq u_{\text{rest}} + RI_0 \frac{\Delta}{\tau_m} \quad (4)$$

due to Taylor expansion of $\exp()$

- We now make the current pulses (of duration Δ) shorter and shorter by maintaining the integral of the current unchanged and equal to the total charge q

$$\int_0^{\Delta} I(t) dt = I_0 \Delta = q$$

- It holds $I_0 = q/\Delta$ and, plugging into (4): $u(\Delta) = u_{\text{rest}} + R q / \tau_m$

Short pulses

- Imposing $\int_0^{\Delta} I(t)dt = I_0\Delta = q$
- As Δ gets shorter, in the limit, the final voltage $u(\Delta)$ of the cell membrane depends on the charge q and not on Δ
- Thus, we can set $I(t) = q\delta(t)$
- The **amount of charge q** transferred to the neuron remains constant, and **the spike is modeled by an impulse**, so Eq. (1) becomes (1b)

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + Rq\delta(t) \quad (1b)$$

Short pulses (spikes)

- The solution of

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + Rq\delta(t)$$

is

$$u(t) = \begin{cases} u_{\text{rest}} & t \leq 0 \\ u_{\text{rest}} + q \frac{R}{\tau_m} \exp\left(-\frac{t}{\tau_m}\right) & t > 0 \end{cases}$$

Green's function of the linear differential equation

Hence, the input spike at time $t_0=0$ is modeled using a Dirac delta

Firing times

- The firing time $t^{(f)}$, $f = 1, 2, 3, \dots$ refers to the instant in time when a neuron emits an action potential (an “output spike”). The firing time is defined by a threshold criterion

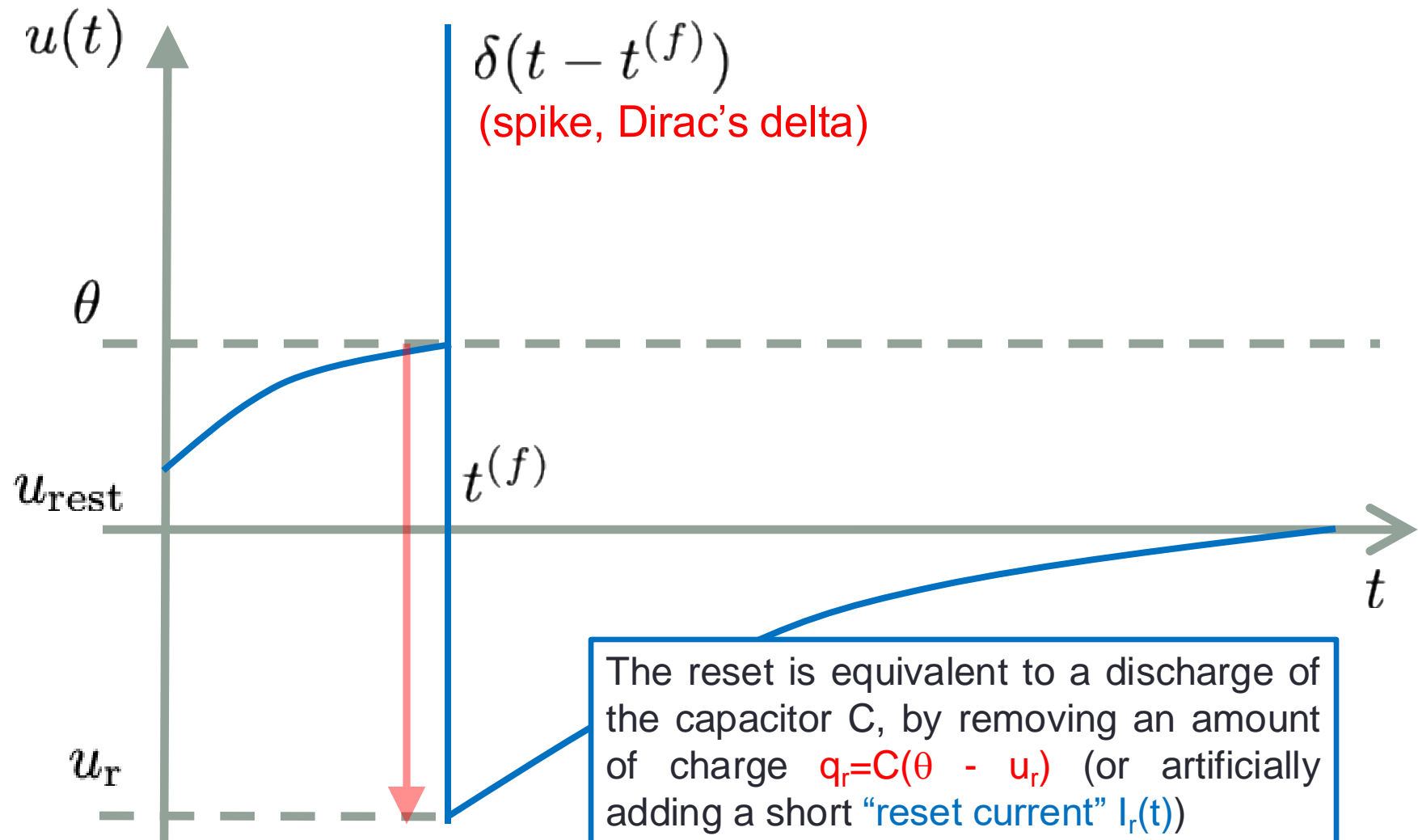
$$t^{(f)} : u(t^{(f)}) = \theta \quad (5)$$

- Immediately after t^f the potential is reset to a value

$$\lim_{\epsilon \rightarrow 0^+} u(t^{(f)} + \epsilon) = u_r$$

- From this point on the dynamics of $u(t)$ follows again Eq. (1) until the next threshold crossing occurs

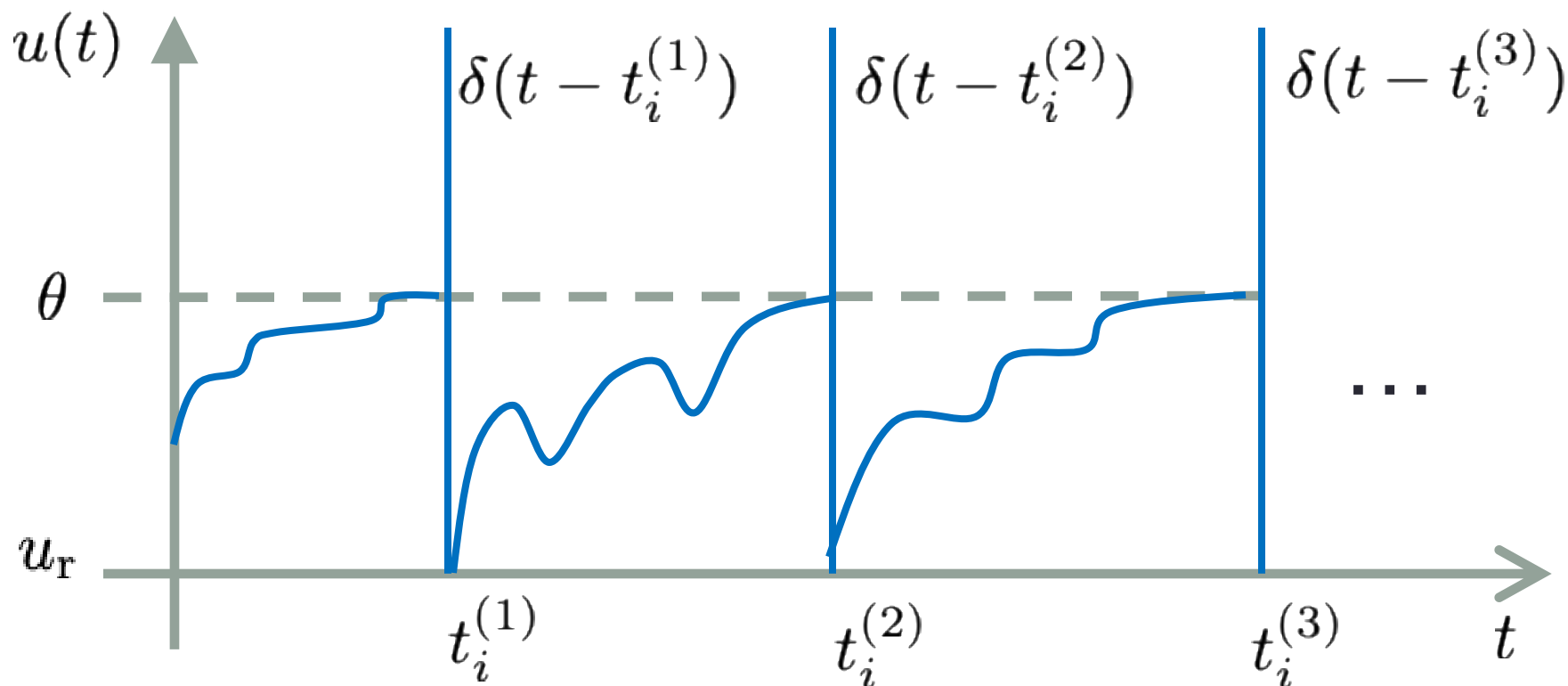
Mathematical model of a firing neuron



Firing times

$$S_i(t) = \sum_f \delta(t - t_i^{(f)}) \quad (6)$$

- We denote the spike train $S_i(t)$ from neuron i as the sequence of firing times $t^{(f)} = \{t \mid u(t) = \theta\}$



Reset current $I_r(t)$

- Each time $u(t)$ reaches the threshold θ , the membrane potential $u(t)$ is reset to u_r . In the electrical circuit, this is equivalent to removing a charge $q_r = C(\theta - u_r)$ from the capacitor, or equivalently adding a negative charge $-q_r$ to it
- Therefore, the reset is equivalent to adding a short current pulse

$$I_r = -q_r \delta(t - t_i^{(f)})$$

at the firing time $t_i^{(f)}$

- Hence, using Eq. (6), the reset current at time t is

$$I_r(t) = -q_r \sum_f \delta(t - t_i^{(f)}) = -C(\theta - u_r) S_i(t) \quad (7)$$

Arbitrary input current $I(t)$

- The **LIF model** is governed by Eq. (1), i.e.,

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + RI(t) \quad (1)$$

- In the presence of an **arbitrary time-dependent input current $I(t)$** , and without accounting for the threshold θ , it holds

$$u(t) = u_{\text{rest}} + \frac{R}{\tau_m} \int_0^{+\infty} \exp\left(-\frac{\xi}{\tau_m}\right) I(t - \xi) d\xi \quad (8)$$

- What happens if we add the threshold θ ?

LIF model revisited

arbitrary
input
current reset
current

- The input current $I(t)$ is updated as $I_{\text{tot}}(t) = I(t) + I_r(t)$
- And the differential equation governing the LIF becomes

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + R[I(t) + I_r(t)] \quad (1c)$$

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + RI(t) + RC(u_r - \theta) \sum_f \delta(t - t_i^{(f)})$$

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + \boxed{RI(t)} + \boxed{\tau_m(u_r - \theta) \sum_f \delta(t - t_i^{(f)})}$$

arbitrary input
current

$I_r(t)$ reset current

LIF model: final differential equation

- The final LIF model for the **neuron membrane evolution** is

$$\frac{du(t)}{dt} = -\frac{1}{\tau_m}[u(t) - u_{\text{rest}}] + \frac{RI(t)}{\tau_m} + (u_r - \theta) \sum_f \delta(t - t_i^{(f)}) \quad (9)$$

$$\frac{du(t)}{dt} = -\frac{1}{\tau_m}[u(t) - u_{\text{rest}}] + \frac{RI(t)}{\tau_m} + \underbrace{(u_r - \theta)S_i(t)}_{I_r(t)/\tau_m} \quad (10)$$

- This is **Eq. (7)** in reference paper [\[SPM-2019\]](#)
- [\[SPM-2019\]](#) Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke, Surrogate Gradient Learning in Spiking Neural Networks: bringing the power of gradient-based optimization to spiking neural networks, *IEEE Signal Processing Magazine*, 2019.

Continuous time solution to the LIF model

- The solution to Eq. (9) is readily found by plugging $I_r(t)$ (Eq. (7)) into Eq. (8) and exploiting the linearity of the integral and the properties of the Dirac delta (when integrated)

$$u(t) = u_{\text{rest}} + (u_r - \theta) \sum_f \exp \left(-\frac{t - t_i^{(f)}}{\tau_m} \right) + \frac{R}{\tau_m} \int_0^{+\infty} \exp \left(-\frac{\xi}{\tau_m} \right) I(t - \xi) d\xi \quad (11)$$

- Where the firing times are defined by the threshold condition

$$t^{(f)} = \{t \mid u(t) = \theta\}$$

SPIKING NEURAL NETWORK TRAINING

Moving from Continuous to Discrete Time

Forward Euler method [ProcIEEE-2023]

- To make the previous equations useful for online learning we need to move into their **discrete time representation**
- The continuous time model of a LIF neuron is

$$\tau_m \frac{du(t)}{dt} = -[u(t) - u_{\text{rest}}] + RI_{\text{tot}}(t)$$

- Without loss of generality (R can be included in the SNN weights, to be learned and u_{rest} is a rescaling), we set:
 - $R=1$, $u_{\text{rest}}=0$
- **This allows to approximate (1) as (for sufficiently small Δt), without taking the limit for $\Delta t \rightarrow 0$**

$$\tau_m \frac{u(t + \Delta t) - u(t)}{\Delta t} = -u(t) + I_{\text{tot}}(t) \quad (12)$$

Forward Euler method [ProcIEEE-2023]

- Rewriting the terms of the last Eq. (12), we get

$$u(t + \Delta t) = \left(1 - \frac{\Delta t}{\tau_m}\right) u(t) + \frac{\Delta t}{\tau_m} I_{\text{tot}}(t)$$

- By defining the **voltage decaying factor** as

$$\beta := 1 - \frac{\Delta t}{\tau_m}$$

- We obtain (still continuous time)

$$u(t + \Delta t) = \beta u(t) + (1 - \beta) I_{\text{tot}}(t)$$

Characterizing β

- Assuming that $I_{\text{tot}}(t)=0$ (no input current), we get

$$u(t + \Delta t) = \beta u(t) + 0 \Rightarrow \beta = \frac{u(t + \Delta t)}{u(t)}$$

- This is the **decay coefficient for the membrane potential**
- In the absence of the input current, we have (u_0 is the membrane potential at time $t_0=0$)

$$u(t) = u_0 \exp\left(-\frac{t}{\tau_m}\right)$$

- And we obtain:

$$\beta = \frac{u(t + \Delta t)}{u(t)} = \exp\left(-\frac{\Delta t}{\tau_m}\right) \quad (13)$$

A note on $I_{\text{tot}}(t)$

- For the total input current, it holds

$$\begin{aligned} I_{\text{tot}}(t) &= I(t) + I_r(t) = I(t) + \tau_m(u_{\text{rest}} - \theta)S_i(t) \\ &= I(t) - \tau_m\theta S_i(t) \end{aligned}$$

due to $u_{\text{rest}}=0$

- It is customary to express the output spike train in discrete time n , $S_i[n]$ as

$$S_i[n] := \Theta(u_i[n] - \theta) \quad (14)$$

Heaviside step function

- That is, $S_i[n]$ is 1 when the potential reaches the **threshold θ**

Membrane potential: discrete model

- Now, assuming that time t is discretized into subsequential time steps $\Delta t = 1$, and referring to the discrete time as n (to emphasize discrete dynamics), we get

$$u[n + 1] = \beta u[n] + (1 - \beta) I_{\text{tot}}[n]$$

- Hence, for neuron i , we get

$$\begin{aligned} u_i[n + 1] &= \beta u_i[n] + (1 - \beta) (I_i[n] - \tau_m \theta S_i[n]) = \\ &= \beta u_i[n] + \frac{\Delta t}{\tau_m} (I_i[n] - \tau_m \theta S_i[n]) = \quad (\Delta t = 1) \\ &= \beta u_i[n] + (1 - \beta) I_i[n] - \theta S_i[n] \end{aligned}$$

- $(1 - \beta)$ is absorbed into the network weights W_{ij} (see next)

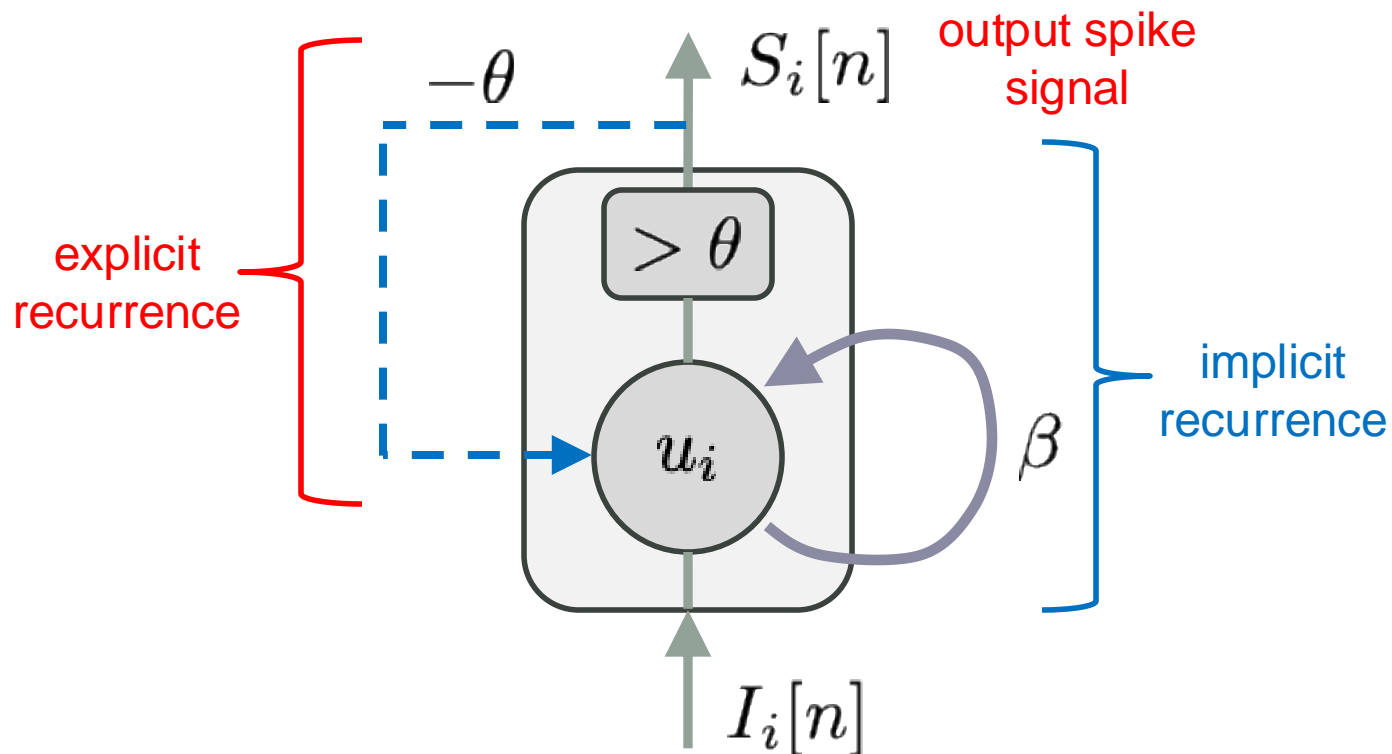
SNNs are inherently recurrent

- SNNs are **stateful**
 - Even if they do not have explicit recurrent connections
 - This is because neurons have an **internal state**

$$u_i[n + 1] = \underbrace{\beta u_i[n]}_{\text{decay}} + \underbrace{I_i[n]}_{\text{input}} - \underbrace{\theta S_i[n]}_{\text{reset}} \quad (15)$$

Recurrent nature of neuron dynamics

$$u_i[n + 1] = \underbrace{\beta u_i[n]}_{\text{decay}} + \underbrace{I_i[n]}_{\text{input}} - \underbrace{\theta S_i[n]}_{\text{reset}}$$



Discrete spiking neuron model

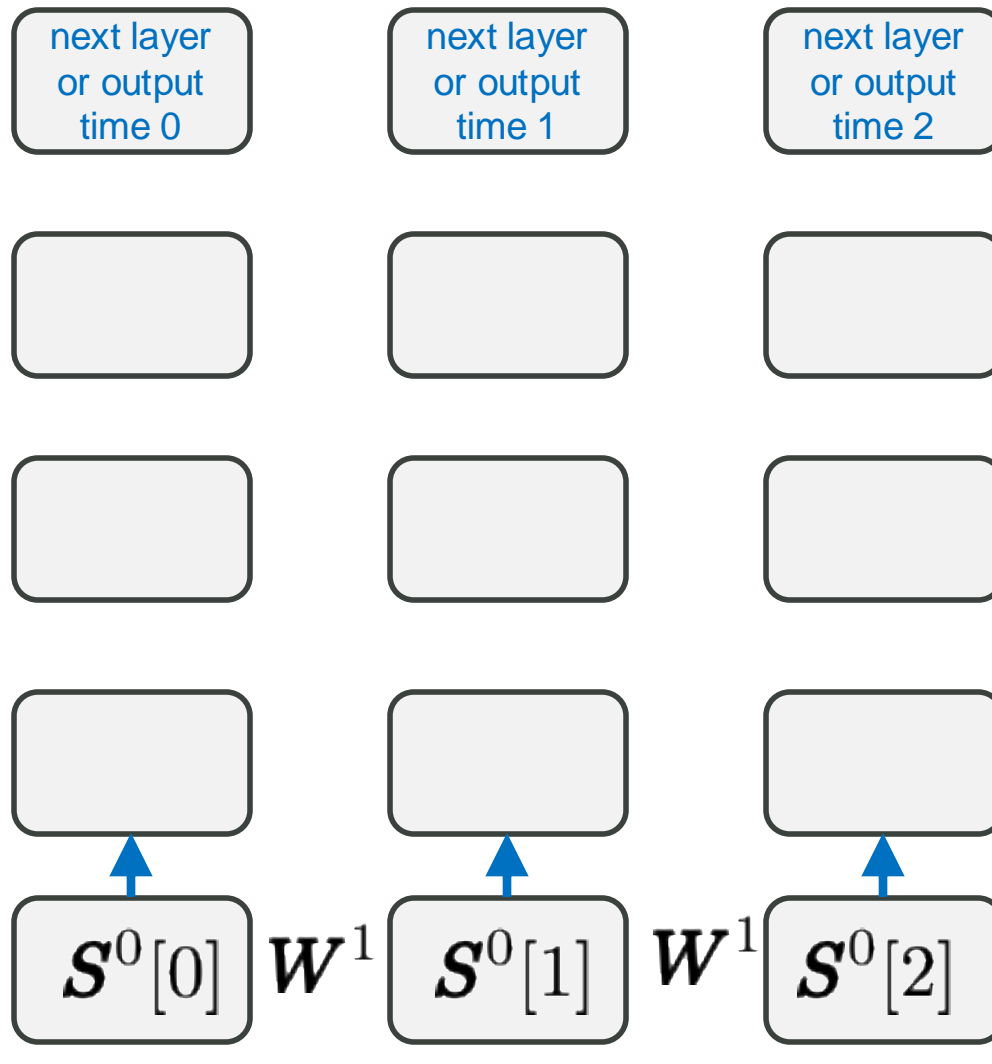
- Hence for $\begin{cases} \text{node } i \\ \text{layer } \ell \end{cases}$, we track the **state pair** $\begin{cases} u_i^{(\ell)}[n] \\ I_i^{(\ell)}[n] \end{cases}$

$$\begin{cases} u_i^{(\ell)}[n+1] = \beta u_i^{(\ell)}[n] + I_i^{(\ell)}[n] - \theta S_i^{(\ell)}[n] \\ I_i^{(\ell)}[n] = \sum_j W_{ij}^{(\ell)} S_j^{(\ell-1)}[n] + \sum_j V_{ij}^{(\ell)} S_j^{(\ell)}[n] \end{cases} \quad (16)$$

spike signal from
neurons in
previous layer
(**feedforward part**)

spikes from neurons in
same layer
(**only for recurrent SNNs**)

SNN computational graph (unrolled)



Input spikes (layer 0) are fed into the network from the bottom and propagate towards the higher layers

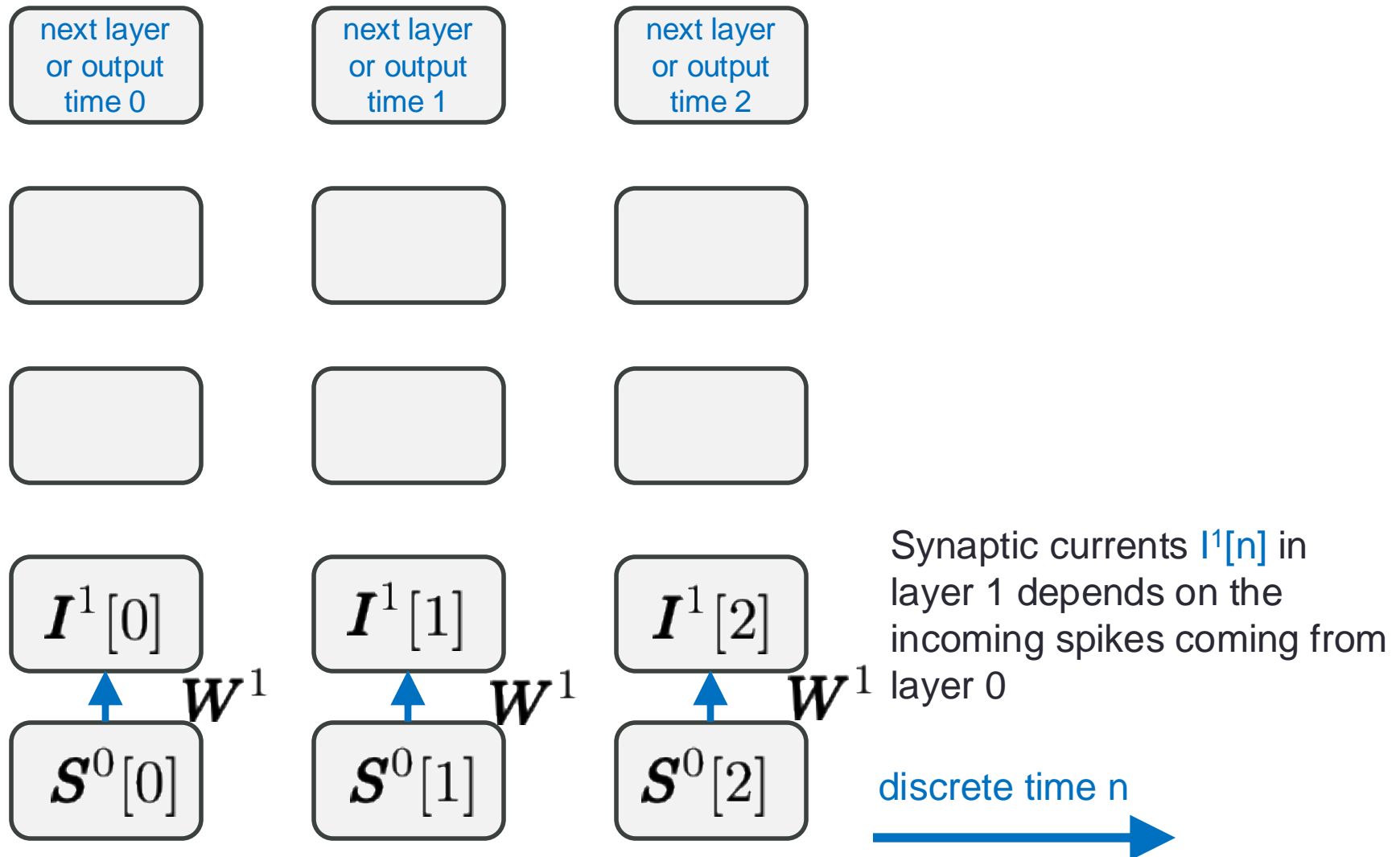
Spikes generated at time n entering layer ℓ

$s^\ell[n]$: ℓ layer , n time

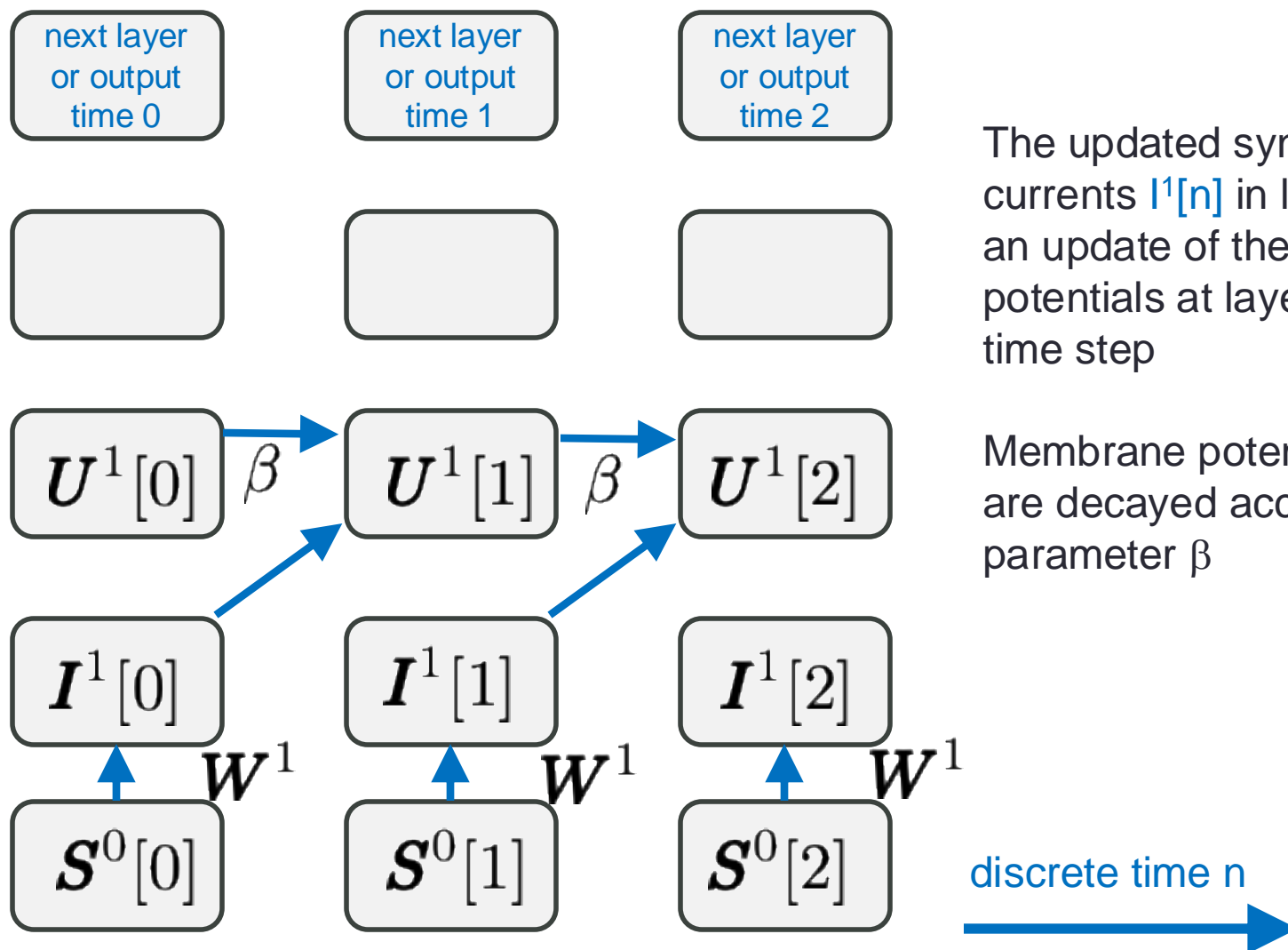
discrete time n



SNN computational graph (unrolled)



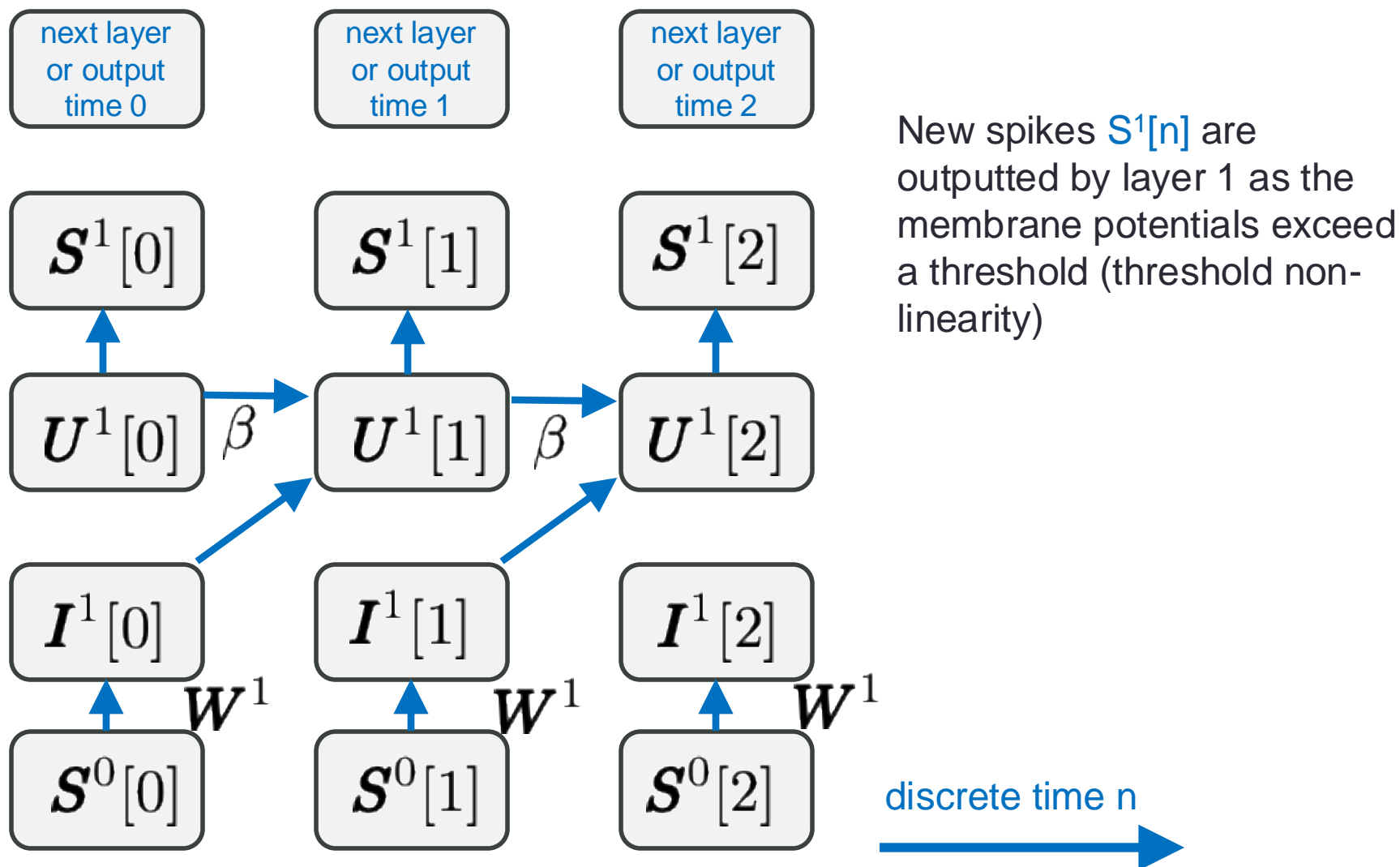
SNN computational graph (unrolled)



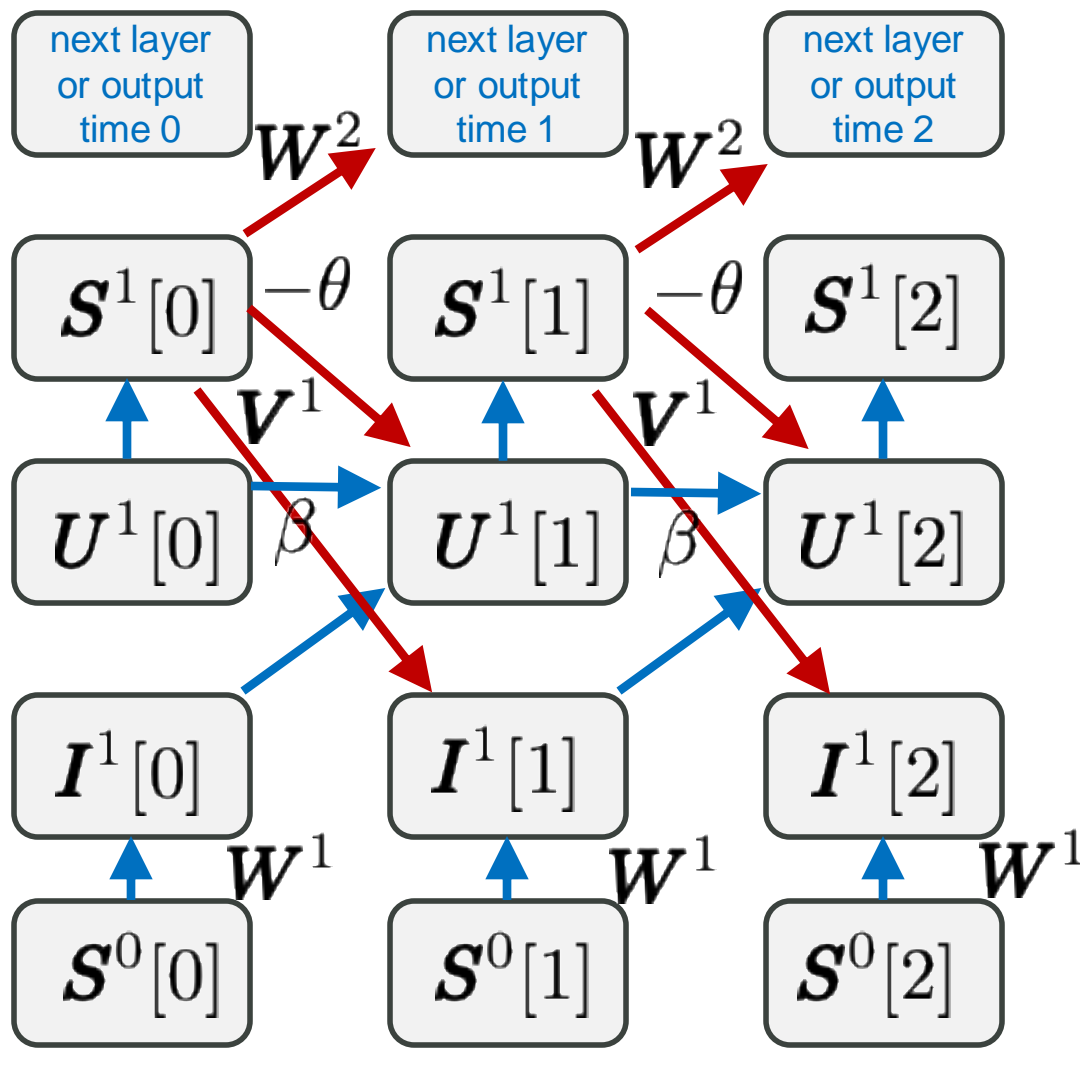
The updated synaptic currents $I^1[n]$ in layer 1 cause an update of the membrane potentials at layer 1 at each time step

Membrane potentials $U^1[n]$ are decayed according to parameter β

SNN computational graph (unrolled)



SNN computational graph (unrolled)



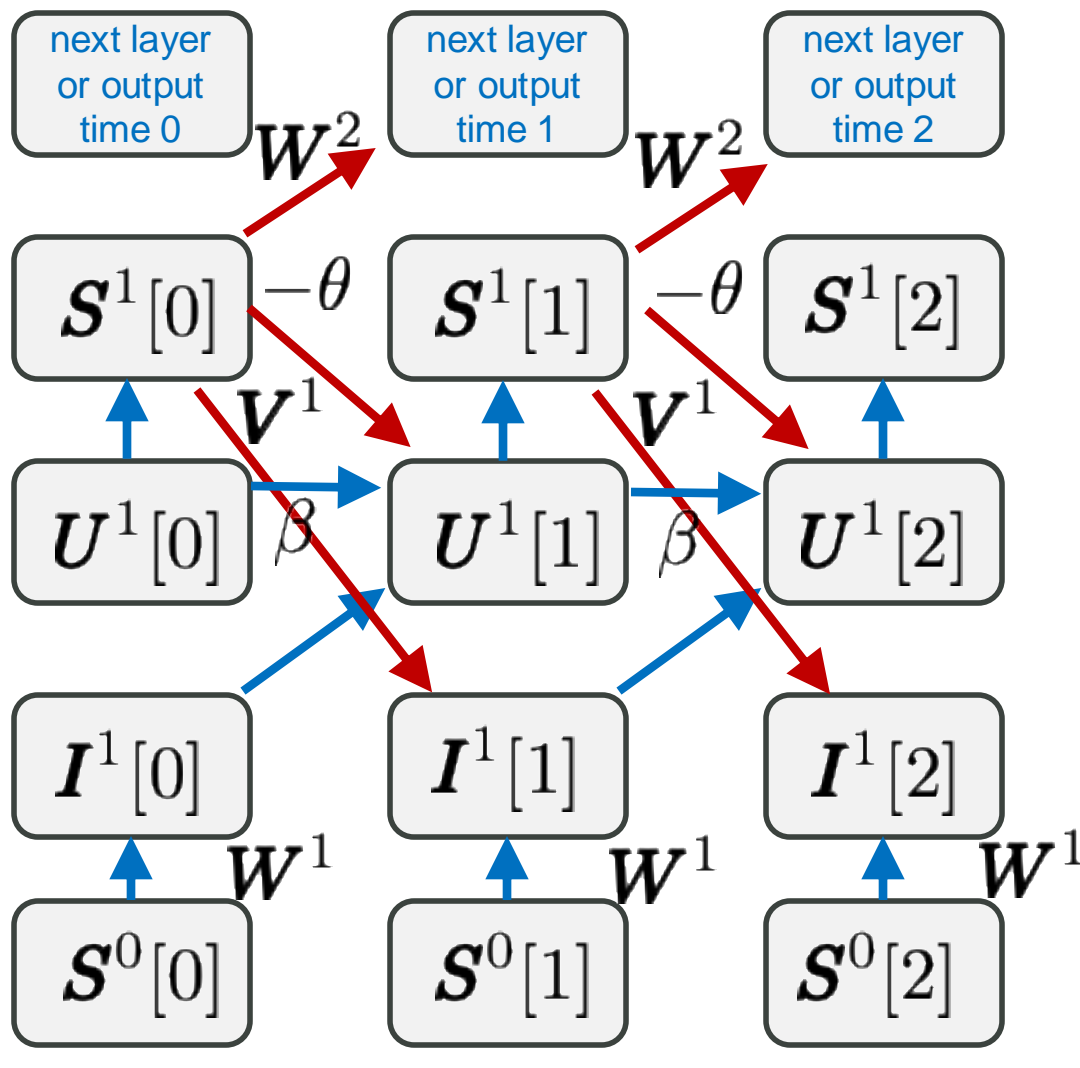
New spikes $S^1[n]$ casually affect the network state (red arrows)

-1: the spike causes the membrane potential of the neuron that emits it to be reset

V^1 : can be fed into the membrane potentials of neurons at the same layer 1 (internal recurrence)

W^2 : can be communicated to the next (downstream) layer or to a readout function on which a cost function is defined

SNN computational graph (unrolled)



An SNN is inherently a
Recurrent Neural Network

SPIKING NEURAL NETWORK TRAINING

Backpropagation – Surrogate Gradient Descent

For a feed-forward SNN

- Equations are:

$$\left\{ \begin{array}{l} S_i[n] := \Theta(u_i[n] - \theta) \\ u_i^{(\ell)}[n+1] = \beta u_i^{(\ell)}[n] + I_i^{(\ell)}[n] - \theta S_i^{(\ell)}[n] \\ I_i^{(\ell)}[n] = \sum_j W_{ij}^{(\ell)} S_j^{(\ell-1)}[n] \end{array} \right.$$

For a feed-forward SNN

- Equations are:

$$\left\{ \begin{array}{l} S_i[n] := \Theta(u_i[n] - \theta) \\ u_i^{(\ell)}[n] = \beta u_i^{(\ell)}[n-1] + I_i^{(\ell)}[n-1] - \theta S_i^{(\ell)}[n-1] \\ I_i^{(\ell)}[n] = \sum_j W_{ij}^{(\ell)} S_j^{(\ell-1)}[n] \end{array} \right.$$

- Setting $n+1 \rightarrow n$

For a feed-forward SNN

- Equations are:

$$\left\{ \begin{array}{l} S_i[n] := \Theta(u_i[n] - \theta) \\ u_i^{(\ell)}[n] = \beta u_i^{(\ell)}[n-1] + I_i^{(\ell)}[n] - \theta S_i^{(\ell)}[n-1] \\ I_i^{(\ell)}[n] = \sum_j W_{ij}^{(\ell)} S_j^{(\ell-1)}[n] \end{array} \right.$$

often

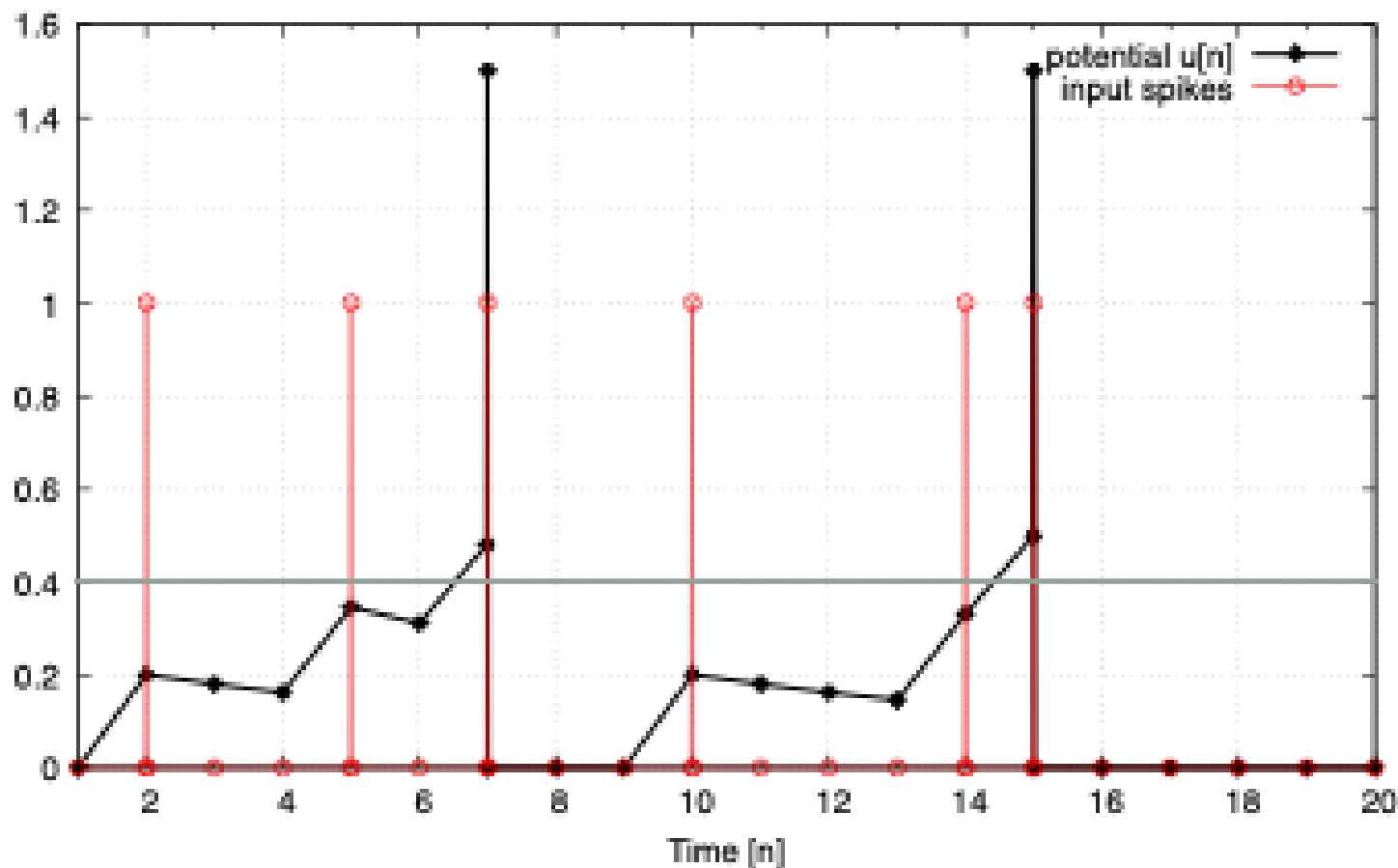
- Typically:** the time index of the input current is **time-shifted by one unit** so that it can instantaneously contribute to the membrane potential. This is not physiologically precise, but it is preferred to cast the SNN model into a standard RNN model

Spiking neuron – example 1

$$\beta = 0.9$$

$$\theta = 0.4$$

$$W = 0.2$$



$$\theta = 0.4$$

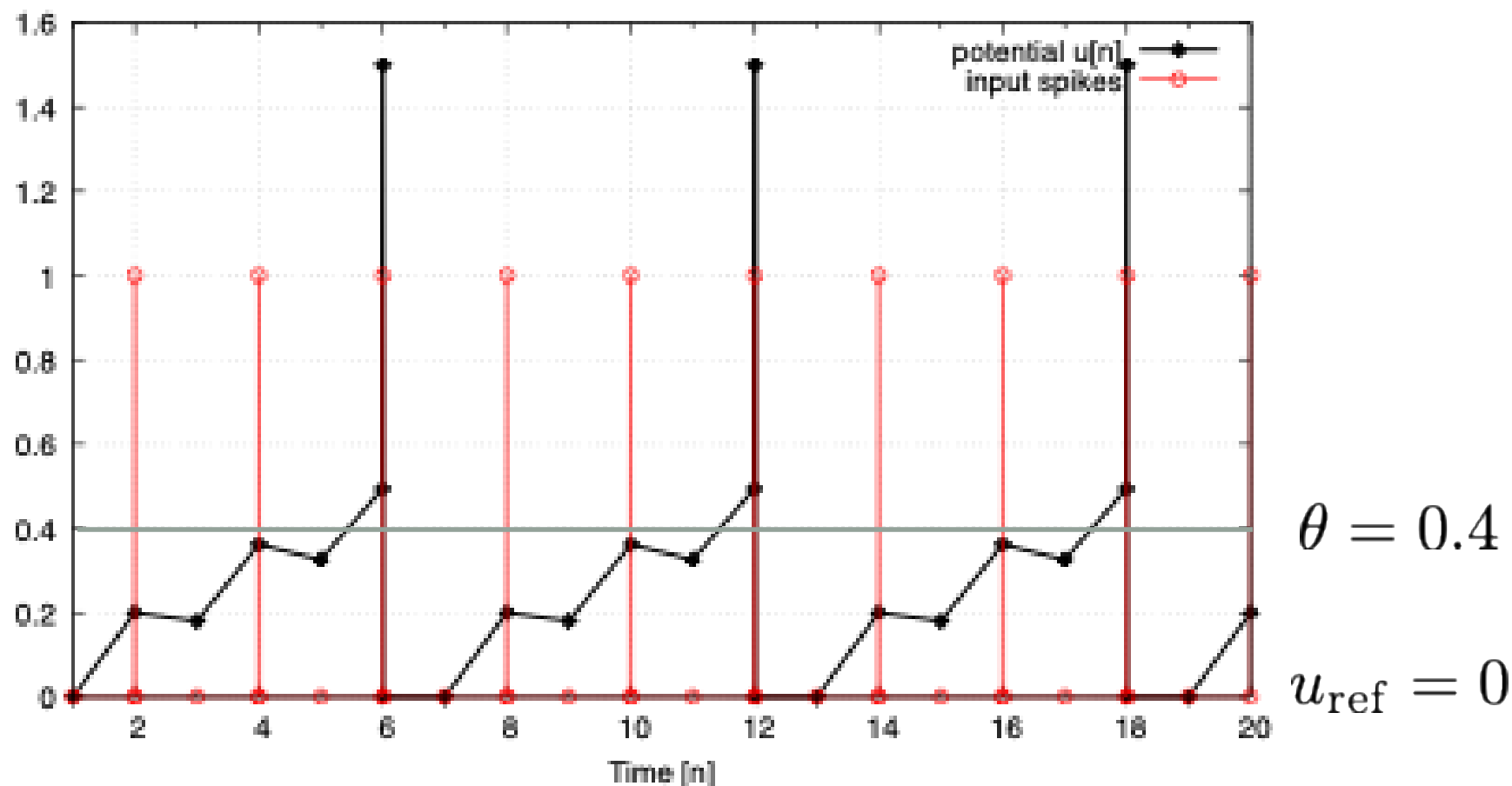
$$u_{\text{ref}} = 0$$

Spiking neuron – example 2

$$\beta = 0.9$$

$$\theta = 0.4$$

$$W = 0.2$$



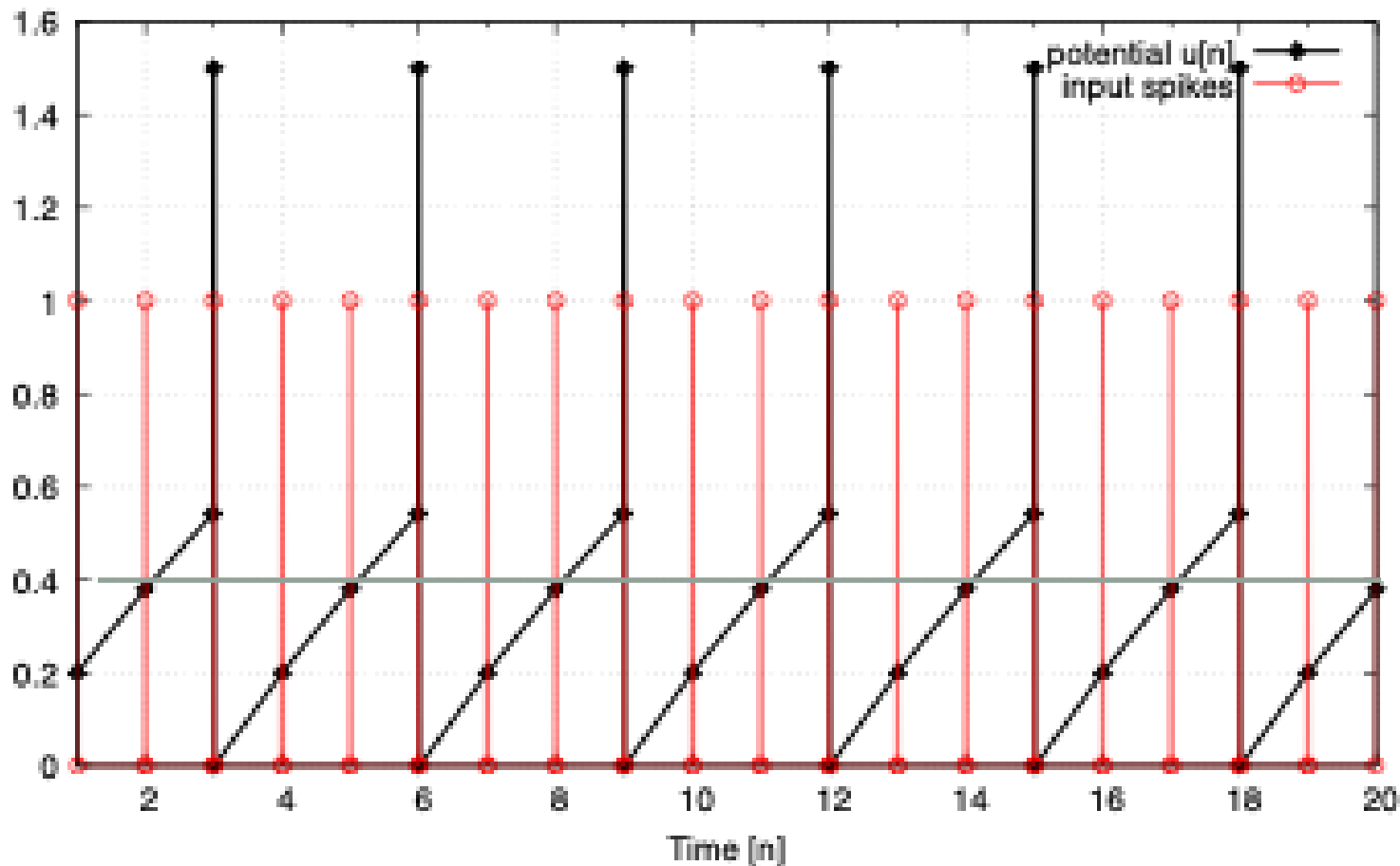
Spiking neuron – example 3

$$\beta = 0.9$$

$$\theta = 0.4$$

$$W = 0.2$$

constant input current



$$\theta = 0.4$$

$$u_{\text{ref}} = 0$$

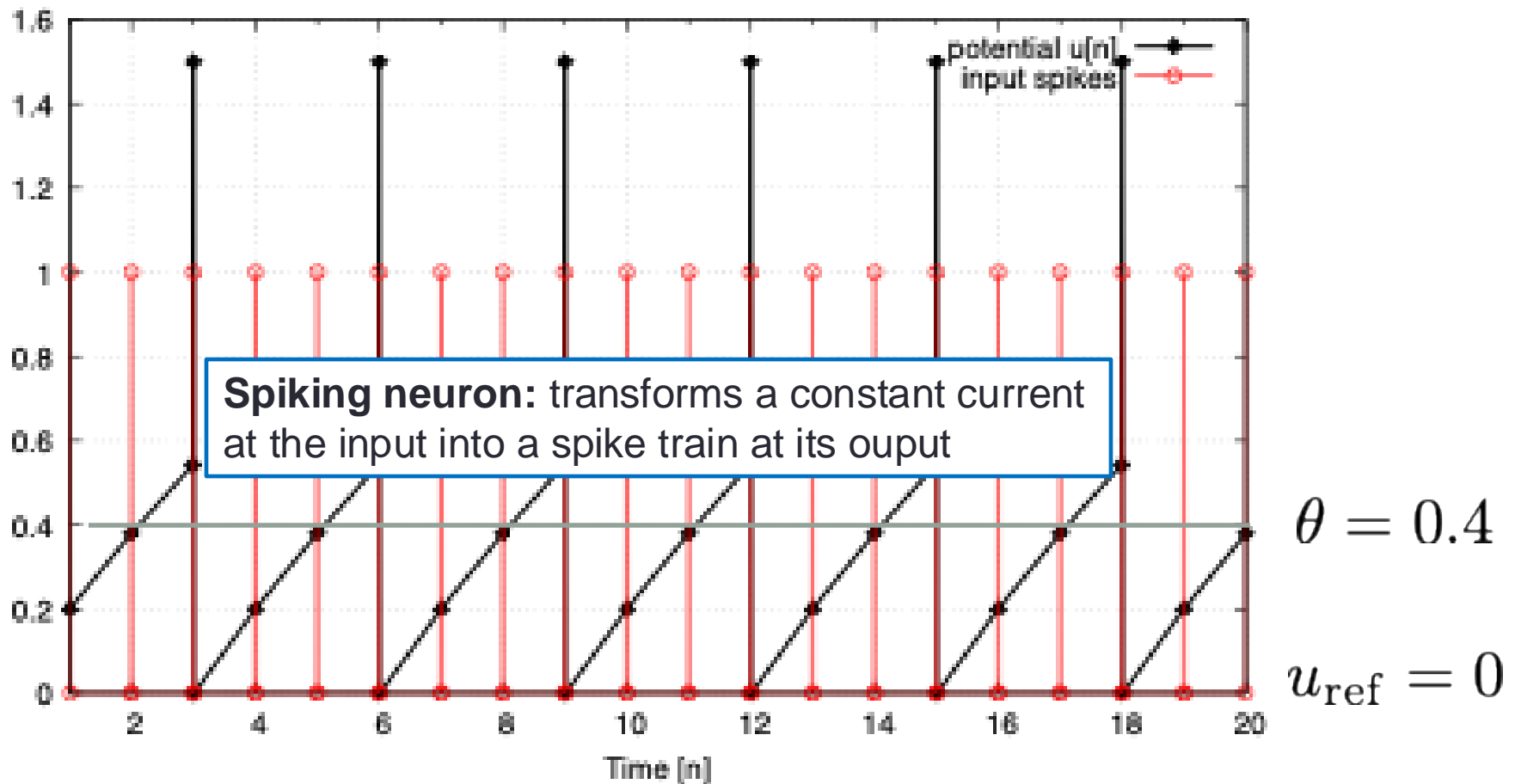
Spiking neuron – example 3

$$\beta = 0.9$$

$$\theta = 0.4$$

$$W = 0.2$$

constant input current



For a feed-forward SNN

- Equations are:

$$\left\{ \begin{array}{l} S_i[n] := \Theta(u_i[n] - \theta) \\ u_i^{(\ell)}[n] = \beta u_i^{(\ell)}[n-1] + I_i^{(\ell)}[n] - \theta S_i^{(\ell)}[n-1] \\ I_i^{(\ell)}[n] = \sum_j W_{ij}^{(\ell)} S_j^{(\ell-1)}[n] \end{array} \right.$$

- Problem:** Heaviside step function is **non-differentiable**
 - The gradient is zero everywhere except when $u_i[n]$ equals the threshold θ , for which it is a Dirac delta
 - Gradient backpropagation cannot be used directly

Surrogate gradient for SNN training

1) Use the **Heaviside function** for the forward pass

- Propagate DATA from input → output

2) **Unfold the SNN** through time – obtaining a DAG

- As you would do for a standard RNN

3) **Approximate the Heaviside** function during the gradient backpropagation using a **surrogate function**

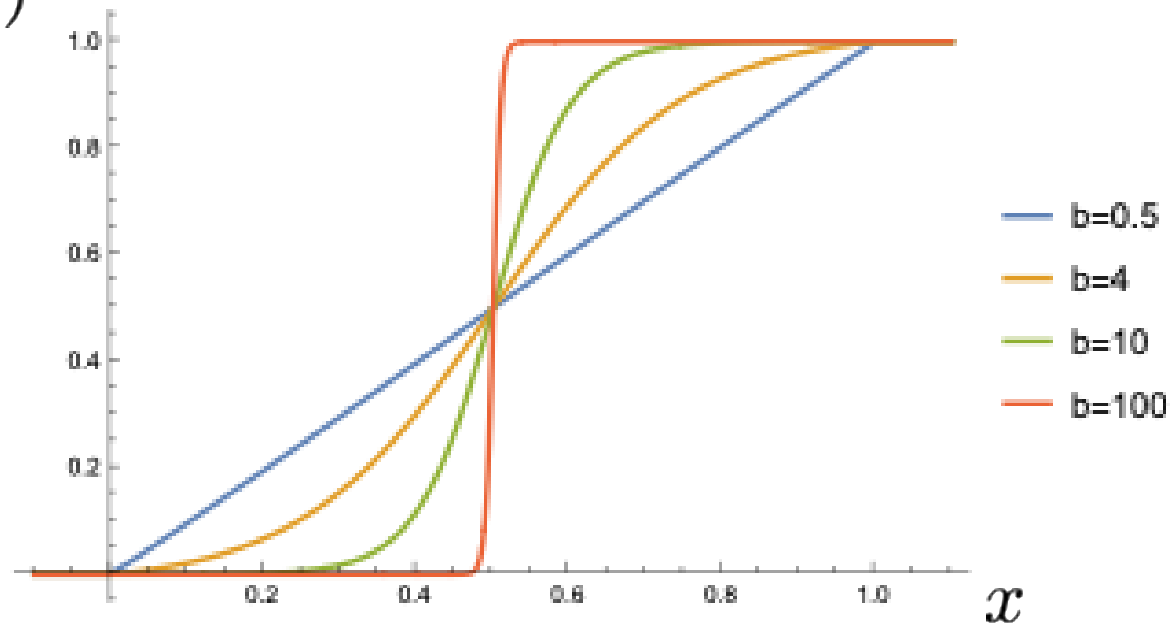
- The surrogate function has a proper derivative everywhere

4) Several surrogate functions can be used

- Learning is reasonably robust to such approximations
- In this way, **standard backpropagation** can be used (usual chain rule of derivatives for functions defined on DAGs)

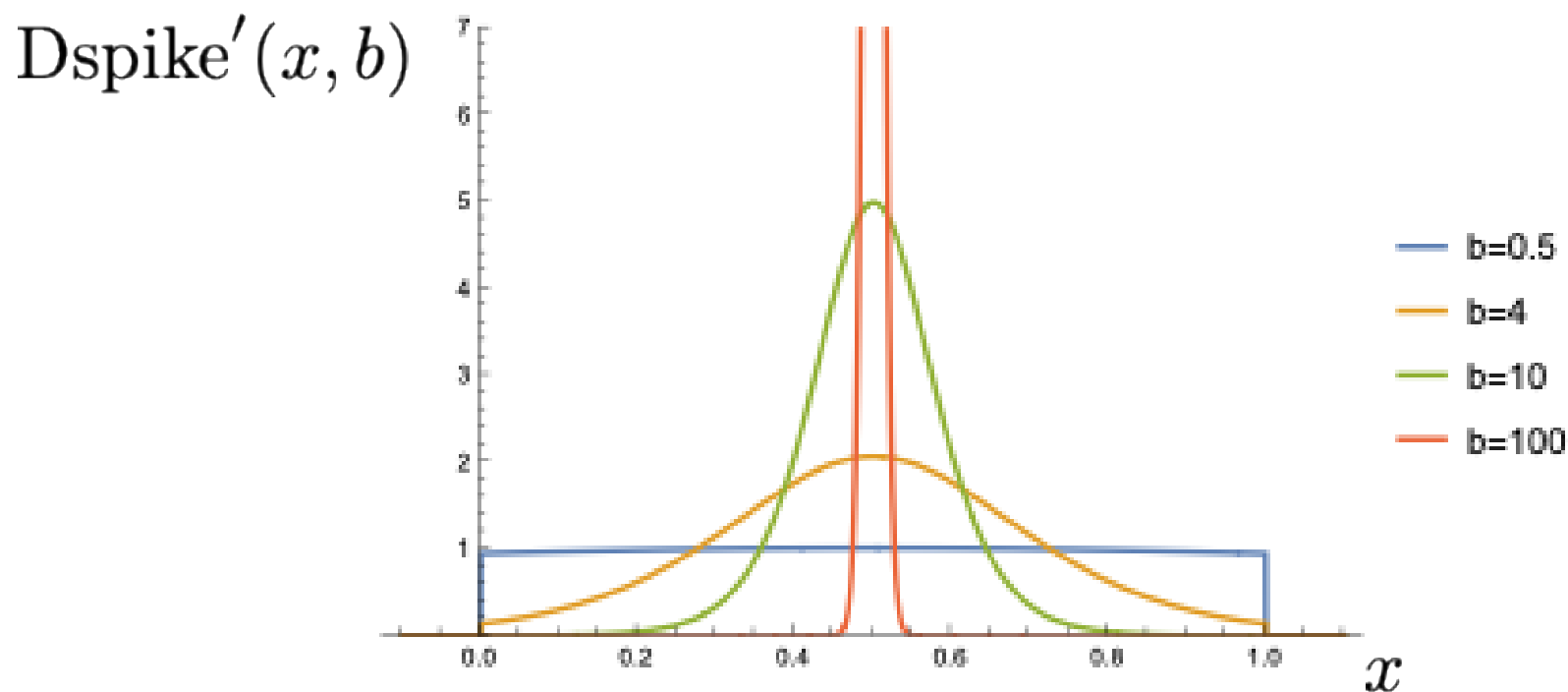
Parameterized smoothing function

$\text{Dspike}(x, b)$



$$\text{Dspike}(x, b) = \begin{cases} 0 & x < 0 \\ \frac{\tanh(b(x-0.5)) + \tanh(b/2)}{2 \tanh(b/2)} & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$$

Smoothing function derivative

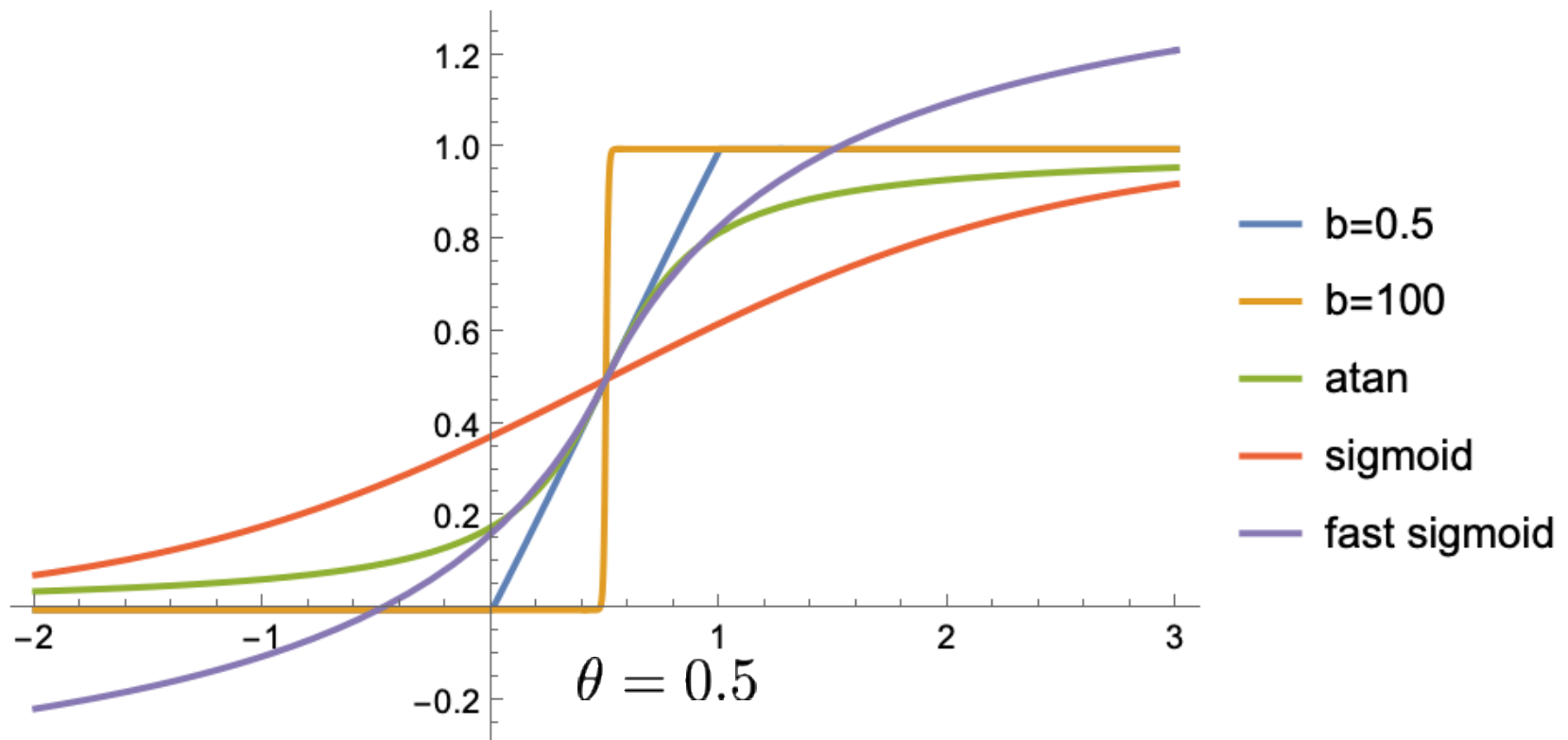


$$D\text{spike}'(x, b) = \frac{a}{2} \coth\left(\frac{a}{2}\right) \text{sech}\left[a(y - 0.5)\right]^2$$

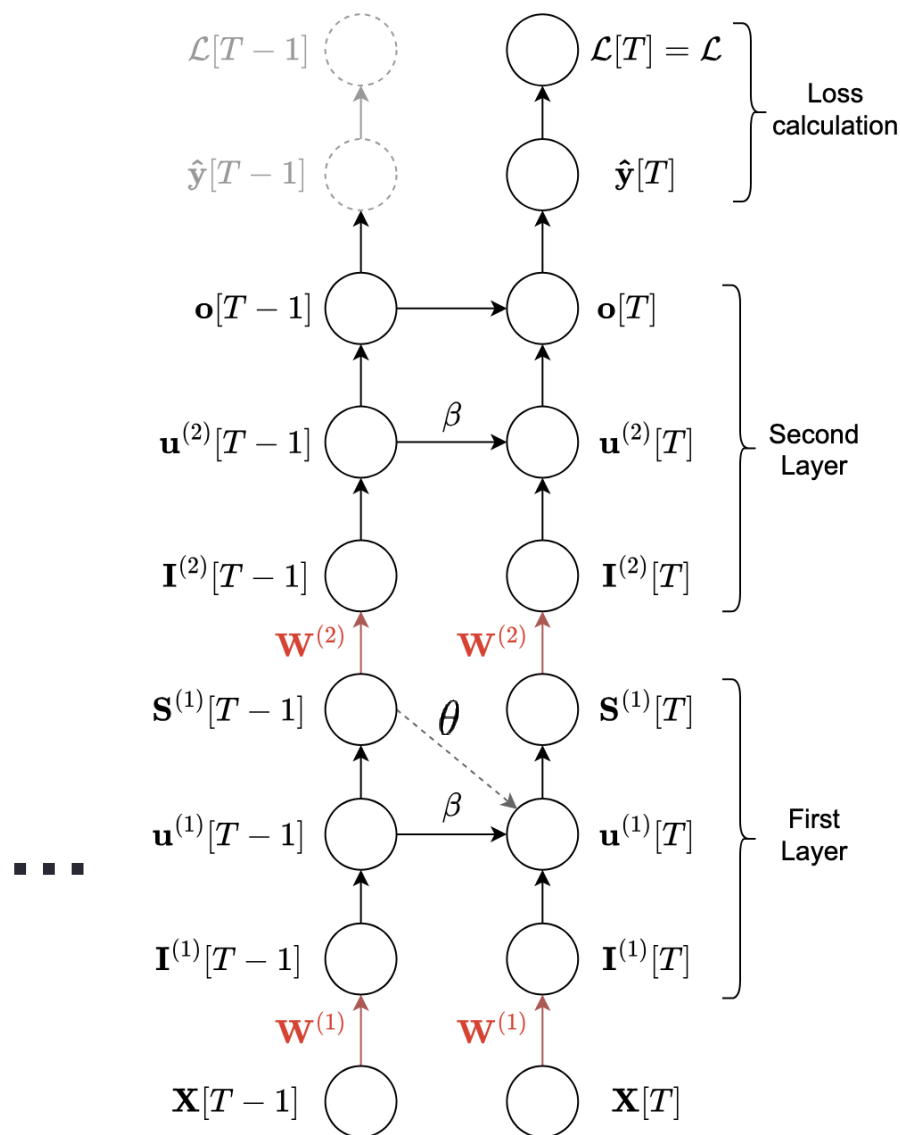
Typical surrogate gradient functions

$\frac{1}{\pi} \arctan(\pi x)$	arctan
$\frac{1}{1 + \exp(-x)}$	sigmoid
$\frac{x}{1 + x }$	fast sigmoid
$\text{Dspike}(b, x)$	dspike

Typical surrogate gradient functions



Two-layer SNN (unfolded)



$$\mathbf{I}^{(1)}[t] = \mathbf{W}^{(1)} \mathbf{X}[t] \in \mathbb{R}^H$$

$$\mathbf{u}^{(1)}[t] = \beta \mathbf{u}^{(1)}[t-1] + \mathbf{I}^{(1)}[t] - \text{reset} \theta \mathbf{S}^{(1)}[t-1] \in \mathbb{R}^H$$

$$\mathbf{S}^{(1)}[t] = \Theta(\mathbf{u}^{(1)}[t] - \theta) \in \mathbb{R}^H$$

$$\mathbf{I}^{(2)}[t] = \mathbf{W}^{(2)} \mathbf{S}^{(1)}[t] \in \mathbb{R}^K$$

$$\mathbf{u}^{(2)}[t] = \beta \mathbf{u}^{(2)}[t-1] + \mathbf{I}^{(2)}[t] \in \mathbb{R}^K$$

$$\mathbf{o}[t] = \sum_{j \leq t} \mathbf{u}^{(2)}[j] \in \mathbb{R}^K$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{o}[t]) \in \mathbb{R}^K$$

$$\mathcal{L} = \mathcal{L}[T] = - \sum_{k=1}^K y_k \log(\hat{y}_k[T])$$

$$= - \langle \mathbf{y}, \log(\mathbf{y}[T]) \rangle$$

- Loss is computed in the last time step T
- All vars at time $t=0$ are 0

Gradient descent – layer 2

- Using the chain rule of derivatives for the output Layer 2, we have:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(2)}} = \frac{\partial \mathcal{L}}{\partial o_i[T]} \frac{\partial o_i[T]}{\partial W_{ij}^{(2)}}$$

- Since

$$o_i[T] = \sum_{t=1}^T u_i^{(2)}[t] \quad \Rightarrow \quad \frac{\partial o_i[T]}{\partial W_{ij}^{(2)}} = \sum_{t=1}^T \frac{\partial u_i^{(2)}[t]}{\partial W_{ij}^{(2)}}$$

Gradient descent – layer 2

- Starting from

$$\mathbf{u}^{(2)}[t] = \beta \mathbf{u}^{(2)}[t-1] + \mathbf{I}^{(2)}[t]$$

- We can develop the recursion:


$$\begin{aligned} u_i^{(2)}[t] &= \beta u_i^{(2)}[t-1] + I_i^{(2)}[t] = \beta^2 u_i^{(2)}[t-2] + \beta u_i^{(2)}[t-1] + I_i^{(2)}[t] = \\ &= \beta^2 u_i^{(2)}[t-2] + \beta I_i^{(2)}[t-1] + I_i^{(2)}[t] = \\ &= \beta^3 u_i^{(2)}[t-3] + \beta^2 I_i^{(2)}[t-2] + \beta I_i^{(2)}[t-1] + I_i^{(2)}[t] = \dots \\ &= \beta^{t-1} u_i^{(2)}[1] + \beta^{t-2} I_i^{(2)}[2] + \beta^{t-3} I_i^{(2)}[3] + \dots + \beta I_i^{(2)}[t-1] + I_i^{(2)}[t] \end{aligned}$$

$$\Rightarrow u_i^{(2)}[t] = \sum_{k=1}^t \beta^{t-k} I_i^{(2)}[k] \Rightarrow \frac{\partial u_i^{(2)}[t]}{\partial W_{ij}^{(2)}} = \sum_{k=1}^t \beta^{t-k} \frac{\partial I_i^{(2)}[k]}{\partial W_{ij}^{(2)}}$$

Gradient descent – layer 2


- Layer 2

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(2)}} = \frac{\partial \mathcal{L}}{\partial o_i[T]} \frac{\partial o_i[T]}{\partial W_{ij}^{(2)}} = \frac{\partial \mathcal{L}}{\partial o_i[T]} \sum_{t=1}^T \frac{\partial u_i^{(2)}[t]}{\partial W_{ij}^{(2)}}$$



$$\frac{\partial \mathcal{L}}{\partial o_i[T]} = \hat{y}_i[T] - y_i$$

softmax



$$\frac{\partial u_i^{(2)}[t]}{\partial W_{ij}^{(2)}} = \sum_{k=1}^t \beta^{t-k} \frac{\partial I_i^{(2)}[k]}{\partial W_{ij}^{(2)}} = \sum_{k=1}^t \beta^{t-k} S_j^{(1)}[k]$$

Gradient descent – layer 2

- Hence, we get

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(2)}} = (\hat{y}_i[T] - y_i) \sum_{t=1}^T \sum_{k=1}^t \beta^{t-k} S_j^{(1)}[k]$$

Gradient descent – Layer 1

- For layer 1, it holds

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial W_{ij}^{(1)}} &= \frac{\partial \mathcal{L}}{\partial o_i[T]} \frac{\partial o_i[T]}{\partial W_{ij}^{(1)}} = \\
 &= \frac{\partial \mathcal{L}}{\partial o_i[T]} \sum_{t=1}^T \frac{\partial u_i^{(2)}[t]}{\partial W_{ij}^{(1)}} = \text{similar recursion as for layer 2} \\
 &= \frac{\partial \mathcal{L}}{\partial o_i[T]} \sum_{t=1}^T \sum_{k=1}^t \beta^{t-k} \frac{\partial I_i^{(2)}[k]}{\partial W_{ij}^{(1)}}
 \end{aligned}$$

?

Gradient descent – layer 1

- For term $\frac{\partial I_i^{(2)}[k]}{\partial W_{ij}^{(1)}}$

- The following holds

$$\frac{\partial I_i^{(2)}[k]}{\partial W_{ij}^{(1)}} = W_{ii}^{(2)} \frac{\partial S_i^{(1)}[k]}{\partial W_{ij}^{(1)}}$$

surrogate function (e.g., arctan)

$$\frac{\partial S_i^{(1)}[k]}{\partial W_{ij}^{(1)}} = \frac{\partial \tilde{S}_i^{(1)}[k]}{\partial u_i^{(1)}[k]} \frac{\partial u_i^{(1)}[k]}{\partial W_{ij}^{(1)}} = \frac{1}{1 + (\pi u_i^{(1)}[k])^2} \sum_{s=1}^k \beta^{k-s} \frac{\partial I_i^{(1)}[s]}{\partial W_{ij}^{(1)}}$$

Gradient descent – layer 1

- Putting it all together

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial W_{ij}^{(1)}} &= \frac{\partial \mathcal{L}}{\partial o_i[T]} \sum_{t=1}^T \sum_{k=1}^t \beta^{t-k} W_{ii}^{(2)} \frac{\partial \tilde{S}_i[k]}{\partial u_i^{(1)}[k]} \frac{\partial u_i^{(1)}[k]}{\partial W_{ij}^{(1)}} = \\
 &= (\hat{y}_i[t] - y_i) \sum_{t=1}^T \sum_{k=1}^t \beta^{t-k} \frac{W_{ii}^{(2)}}{1 + (\pi u_i^{(1)}[k])^2} \sum_{s=1}^k \beta^{k-s} X_j[s]
 \end{aligned}$$

Dataset

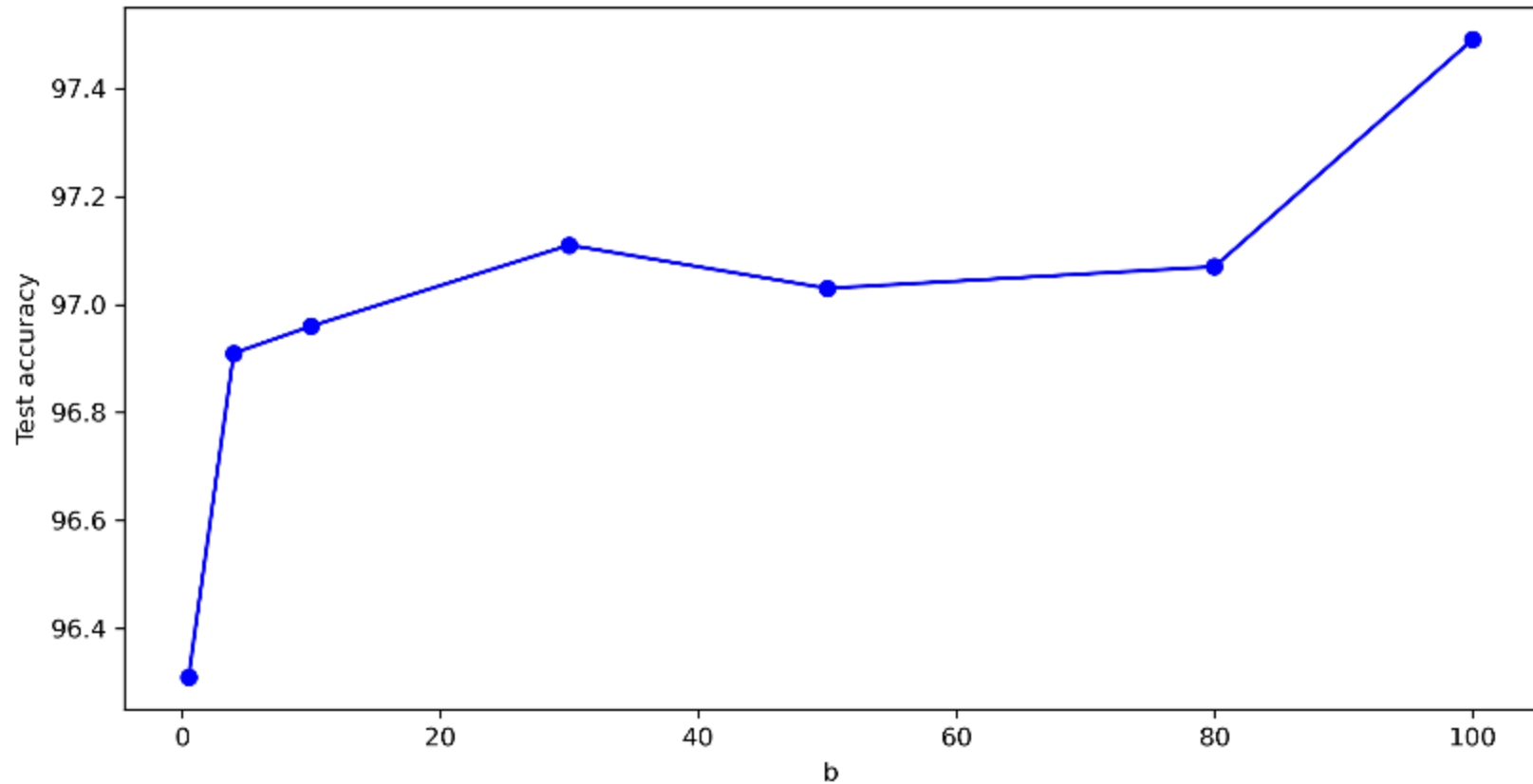
MNIST

The MNIST database of handwritten digits is a well-known dataset in the field of image recognition. It consists of 60,000 training samples and 10,000 test samples of 28x28 resolution.

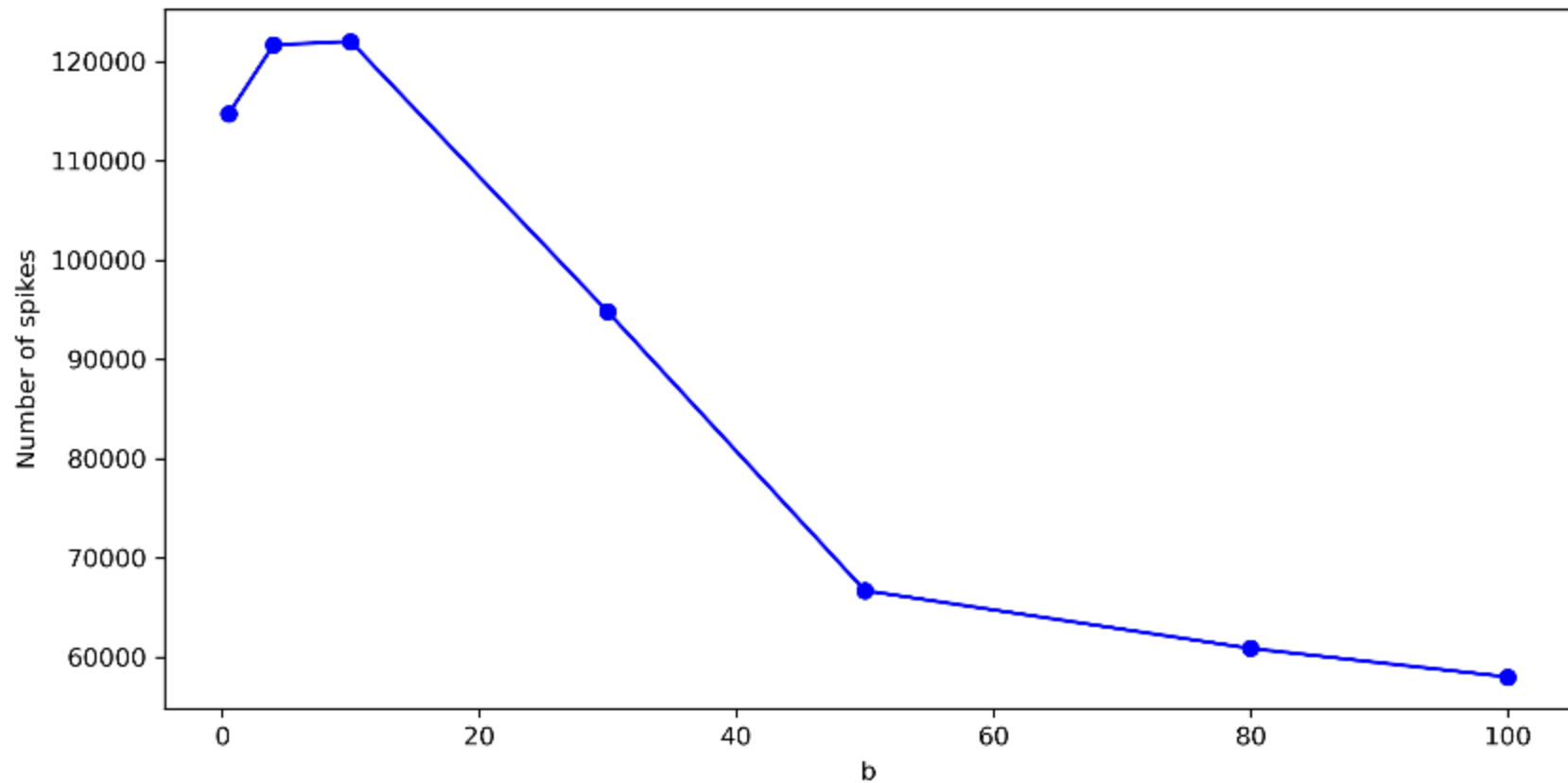


Download website: <https://github.com/cvdfoundation/mnist>

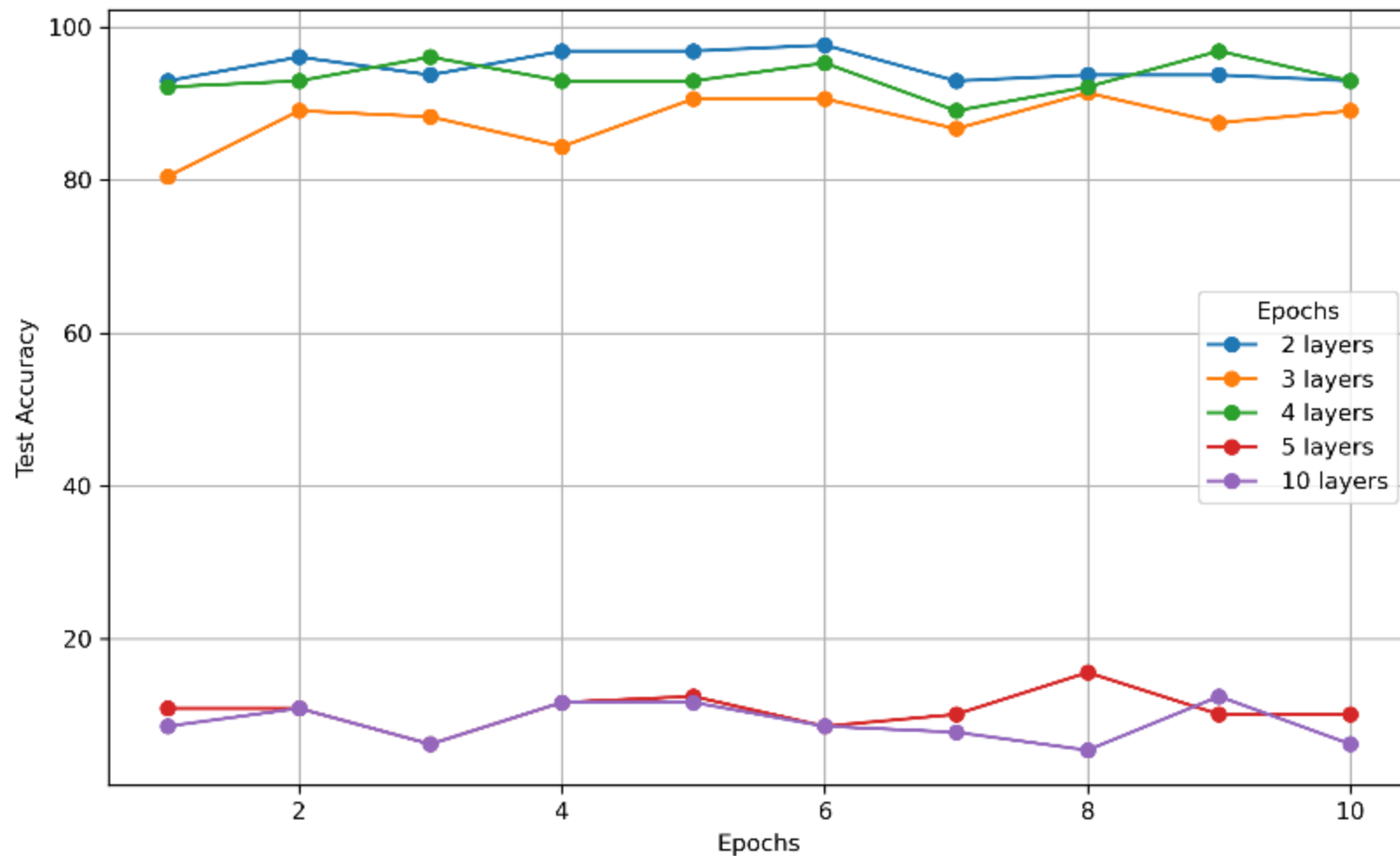
Dspike - accuracy vs b



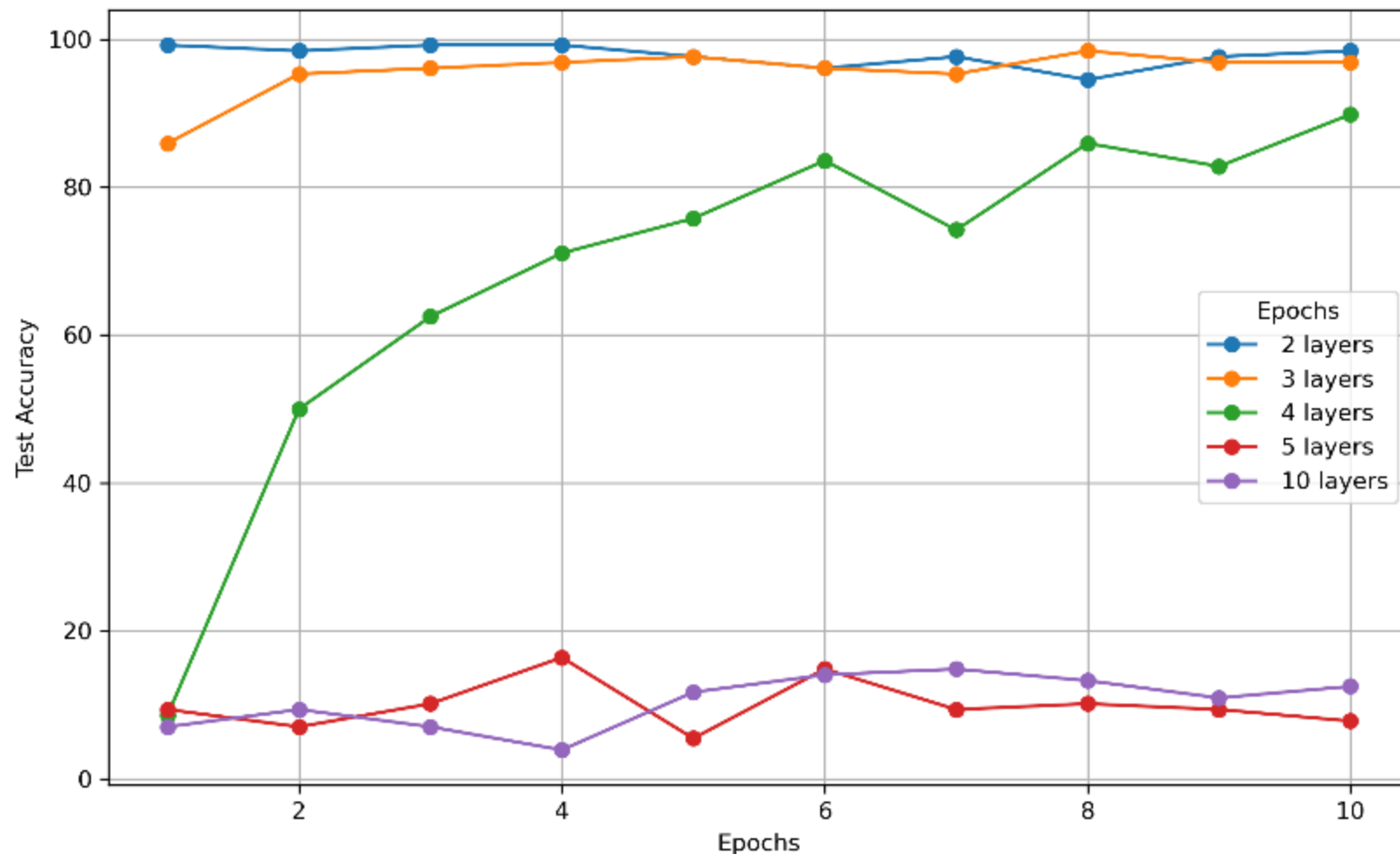
Dspike - no. of spikes vs b



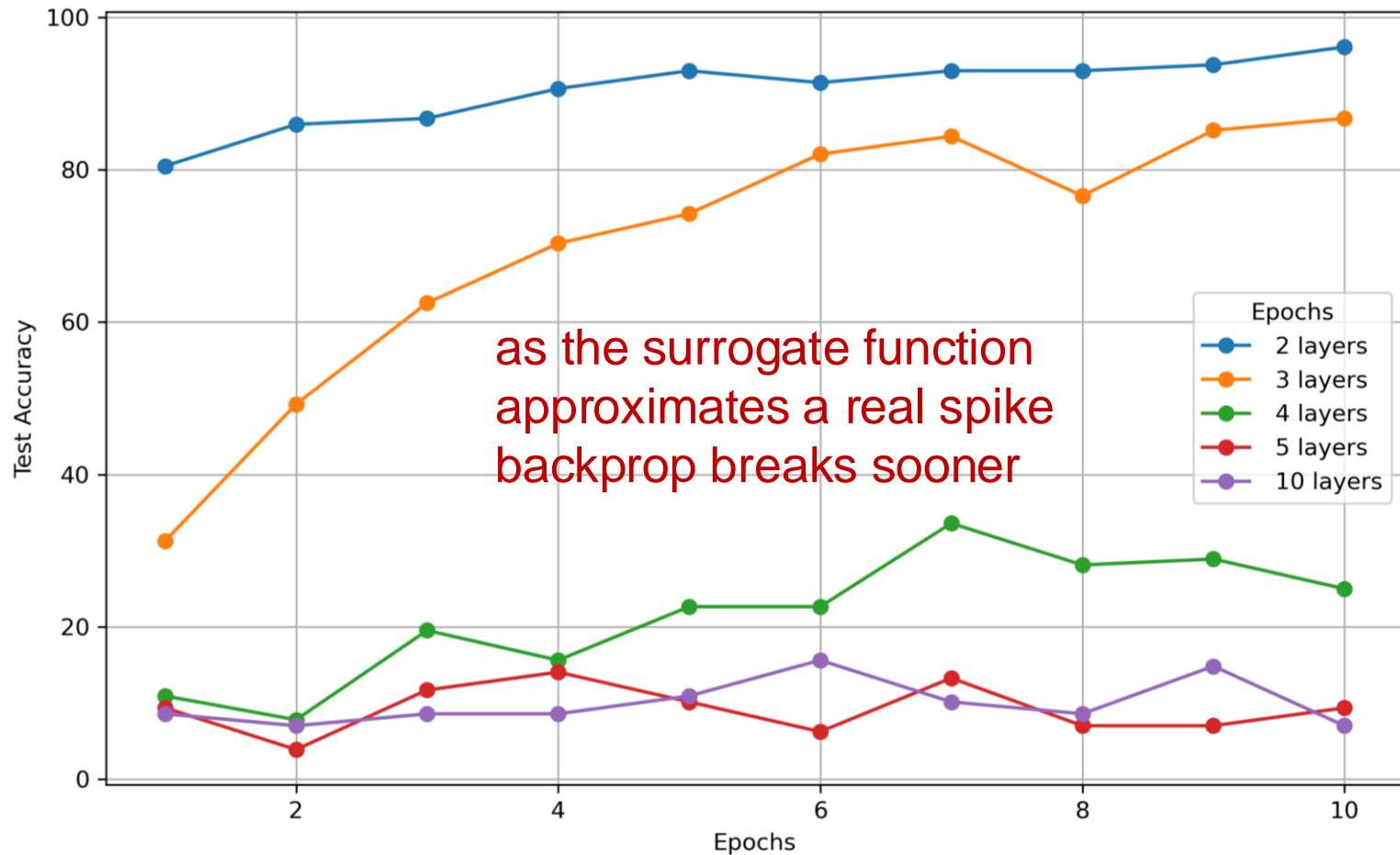
Test accuracy vs training time ($b=0.5$)



Test accuracy vs training time (b=100)



Test accuracy vs training time (b=1000)



Open problems

- **Objective: increase no. of layers**
 - Defining a GRU cell for SNN
 - Use Batchnorm+GRU+residual connections
- **A new training approach?**
 - Departing from backpropagation?

SPIKING NEURAL NETWORKS

Michele Rossi

michele.rossi@unipd.it

Dept. of Information Engineering

Dept. of Mathematics



UNIVERSITÀ
DEGLI STUDI
DI PADOVA