Training spiking neural networks via adjoint sensitivity analysis

Quique Toloza

Supervisor: Mark Harnett

subMIT IAP Workshop 2024

February 2nd, 2024

Big idea: The brain is extraordinarily effective on a modest energy budget

Luccioni et al. 2022

Big idea: The brain is extraordinarily effective on a modest energy budget

$1e^{14}$ synapses

$4e^{3}$ MJ/year (human)

Luccioni et al. 2022

Big idea: The brain is extraordinarily effective on a modest energy budget

GPT-3

$1e^{14}$ synapses

$2e^{11}$ parameters

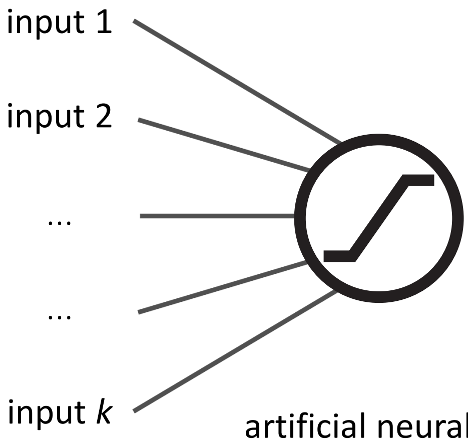
$4e^{3}$ MJ/year (human)

$4e^{6}$ MJ (training)

Luccioni et al. 2022

How are biological neurons different from artificial neurons?

How are biological neurons different from artificial neurons?

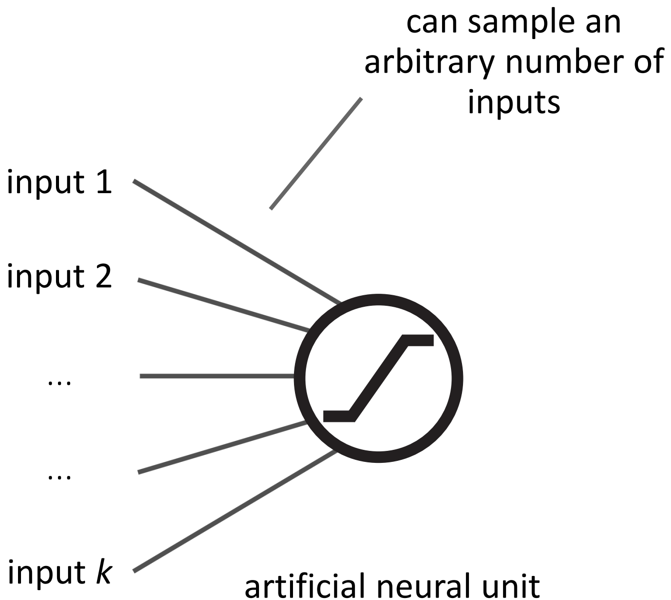


artificial neural unit

aka

“point” neuron

How are biological neurons different from artificial neurons?

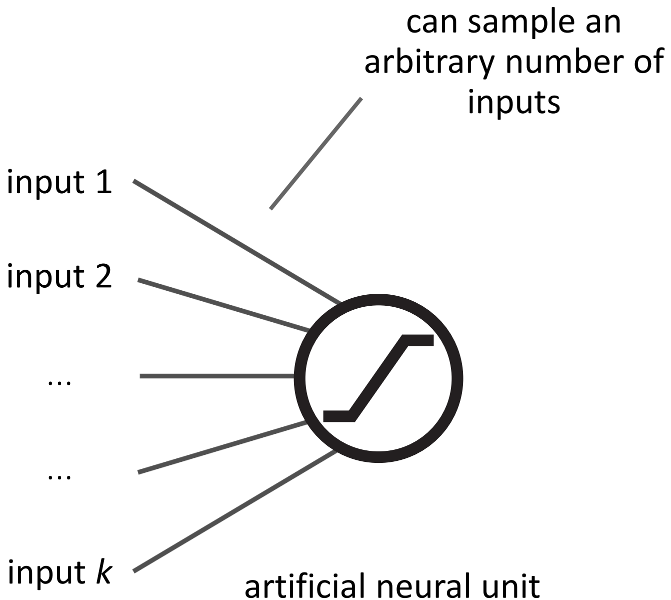


artificial neural unit

aka

“point” neuron

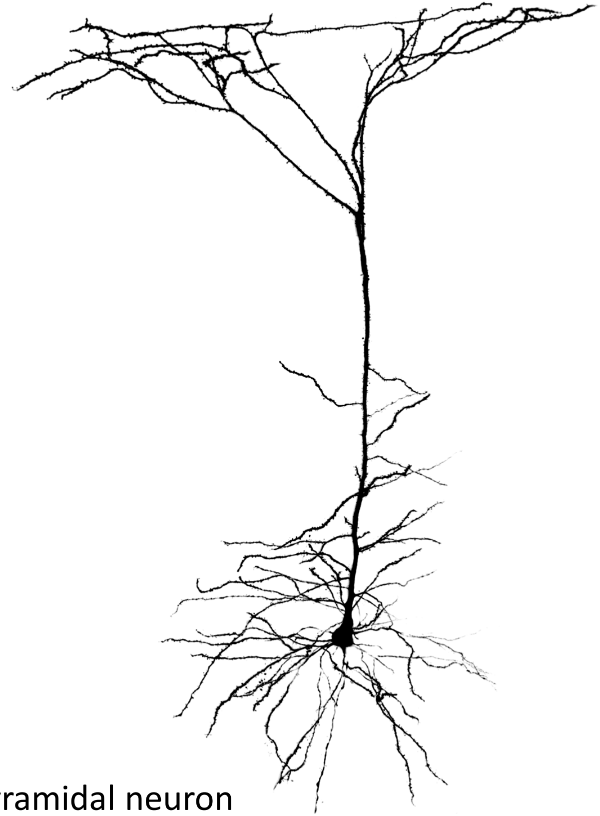
How are biological neurons different from artificial neurons?



artificial neural unit

aka

“point” neuron

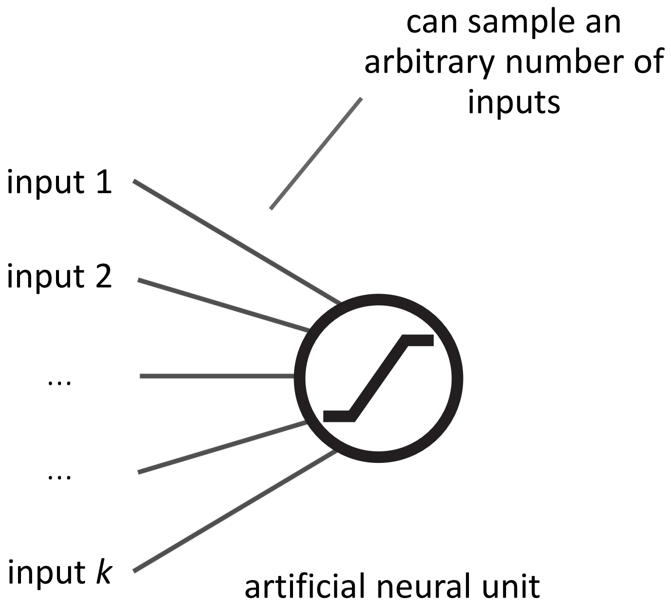


layer V pyramidal neuron

Mark Harnett

Vardalaki et al. 2022

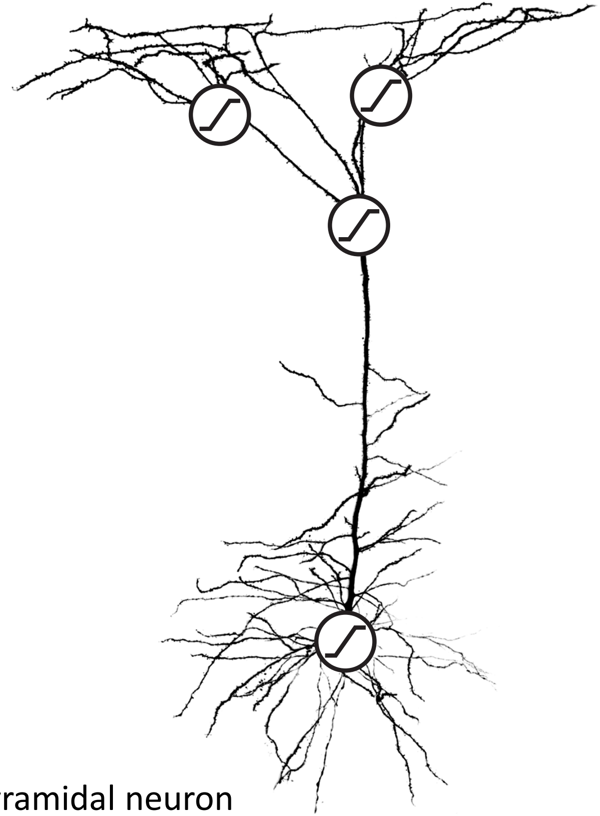
How are biological neurons different from artificial neurons?



artificial neural unit

aka

“point” neuron



layer V pyramidal neuron

Mark Harnett

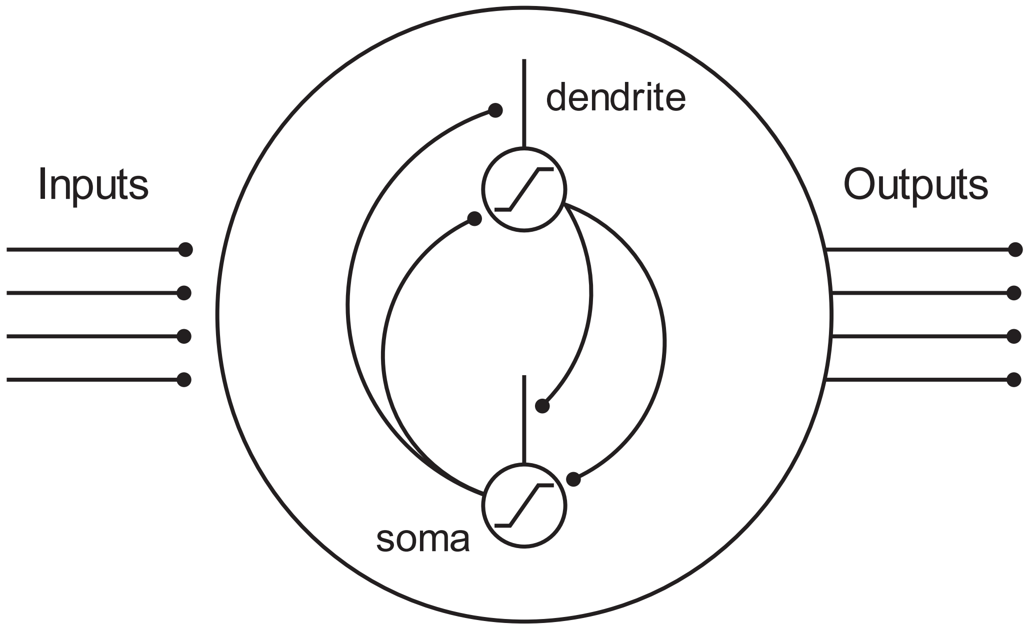
Vardalaki et al. 2022

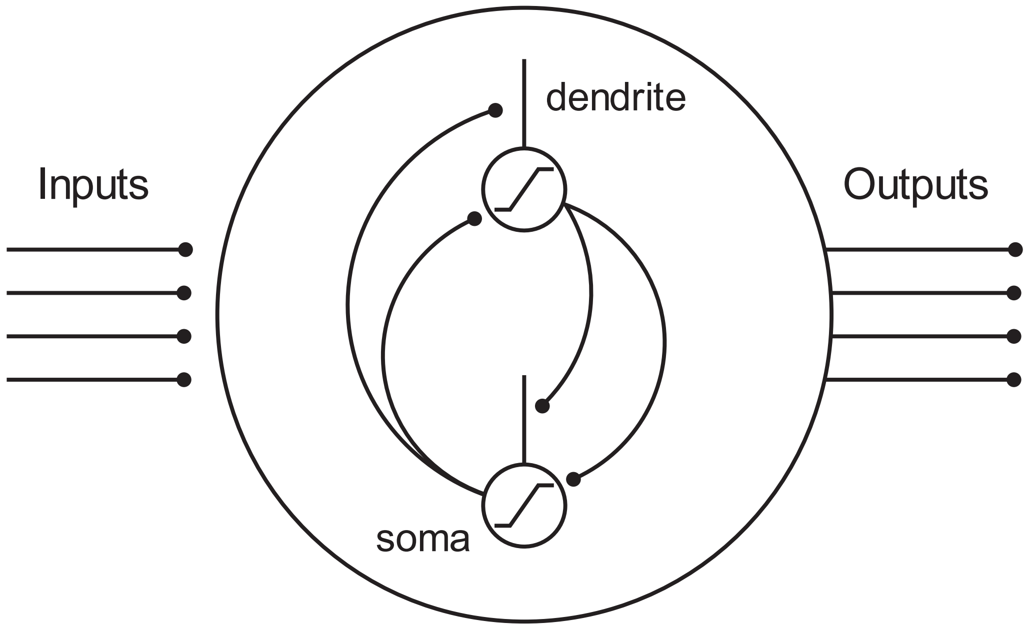
Why do we have dendrites?

Why do we have dendrites?

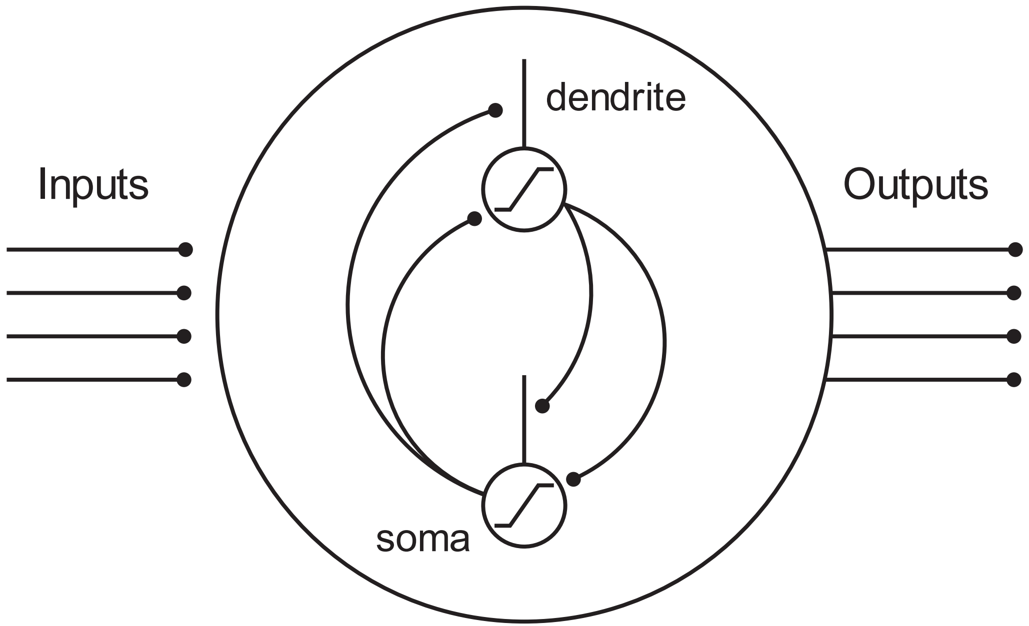
↓

What computational role do dendrites serve?


Inputs
dendrite
Outputs
soma

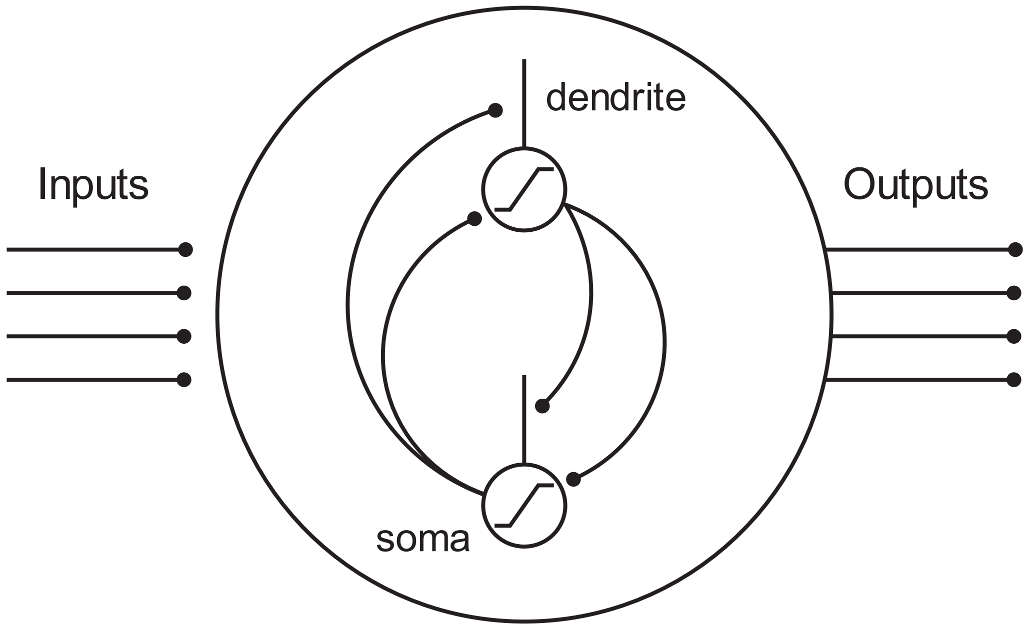


What do dendrites get us?



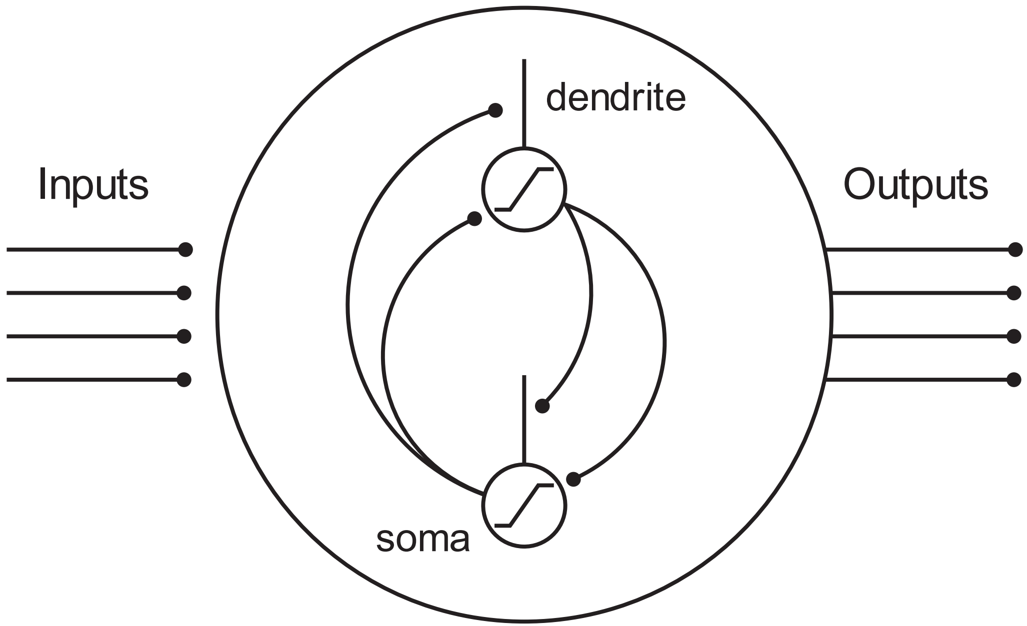
What do dendrites get us?

- without assumptions on connectivity



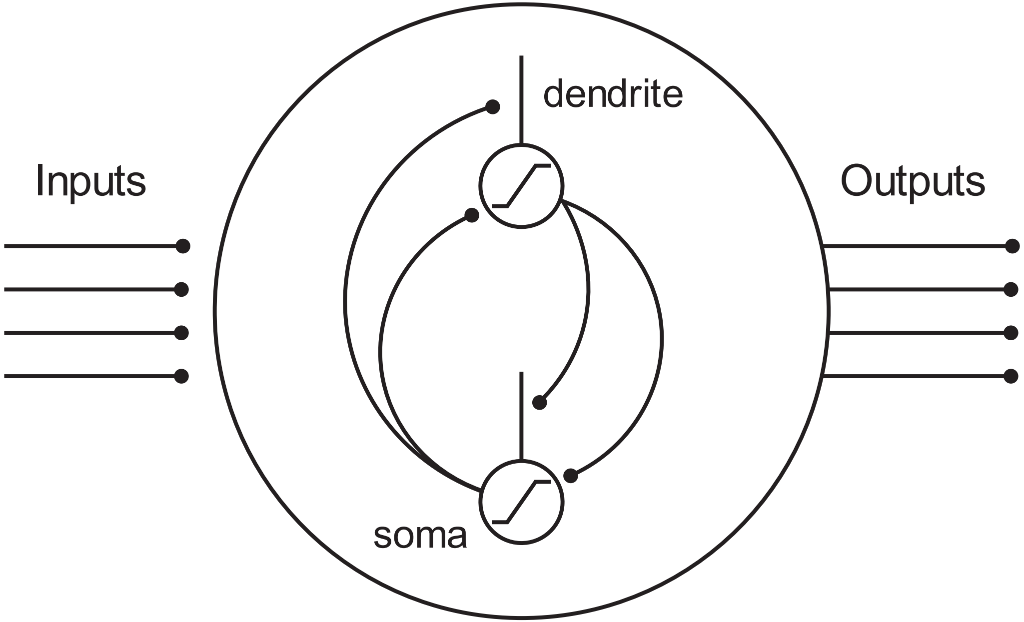
What do dendrites get us?

- without assumptions on connectivity
- does it change from task to task?



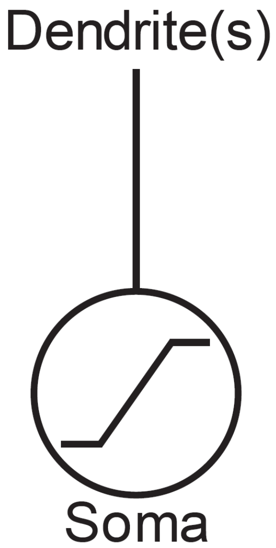
What do dendrites get us?

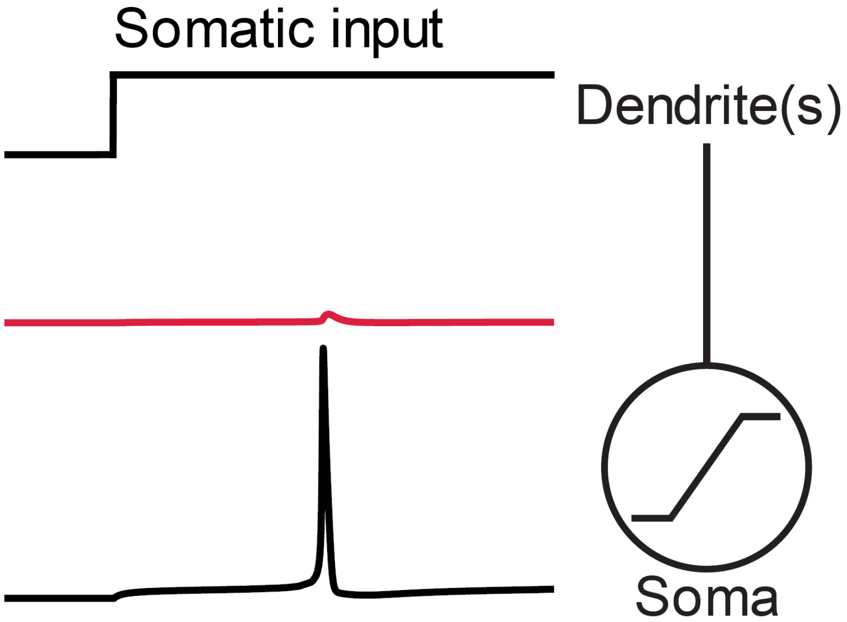
- without assumptions on connectivity
- does it change from task to task?
- does it change from model to model?

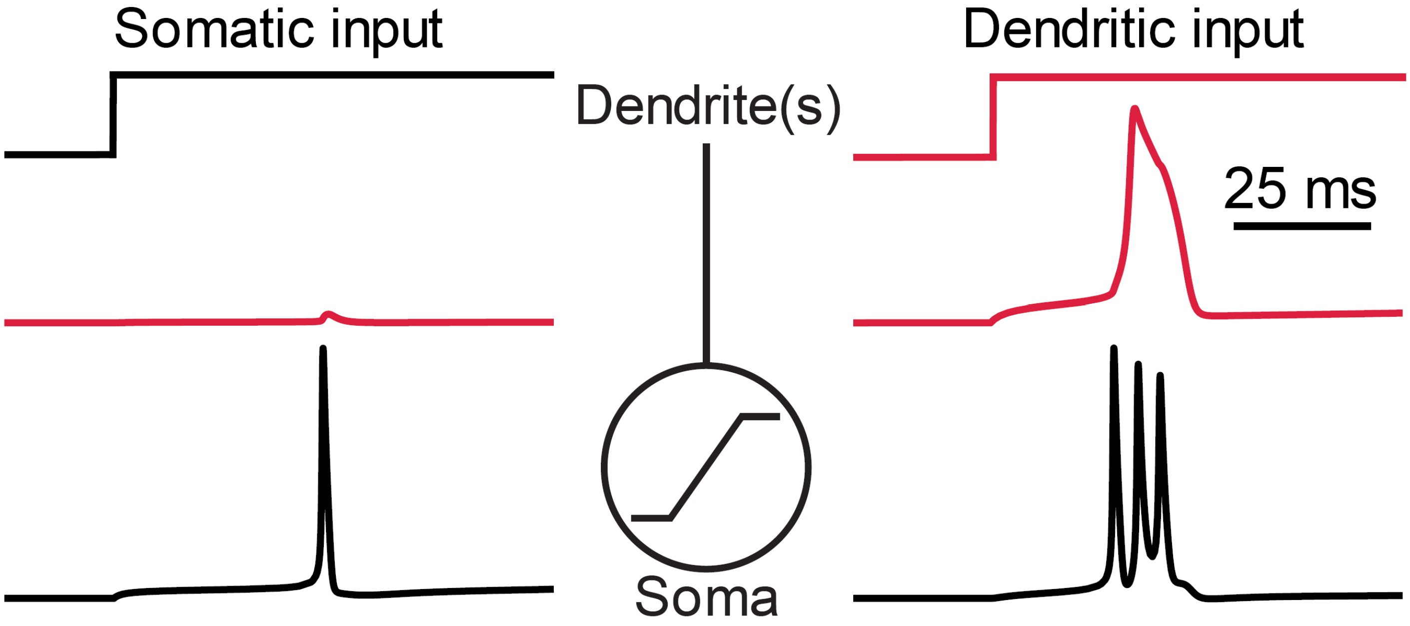


What do dendrites get us?

- without assumptions on connectivity
- does it change from task to task?
- does it change from model to model?

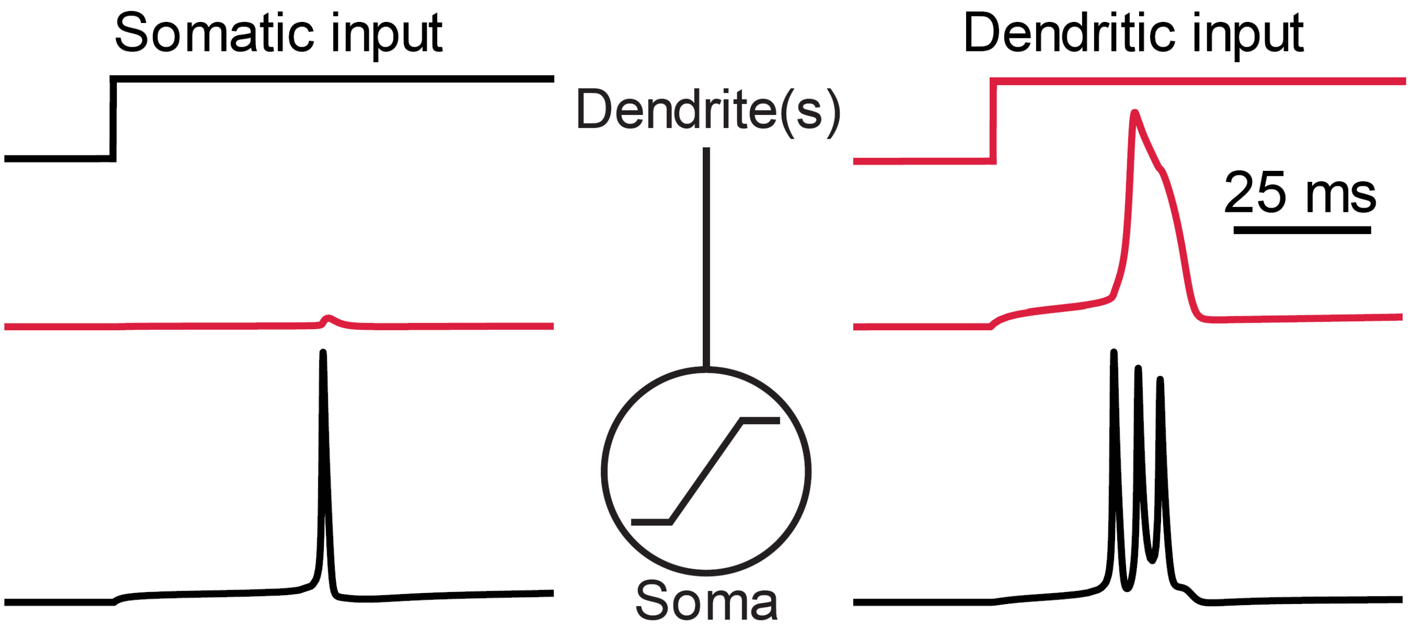

Dendrite(s)
Soma


Somatic input
Dendrite(s)
Soma

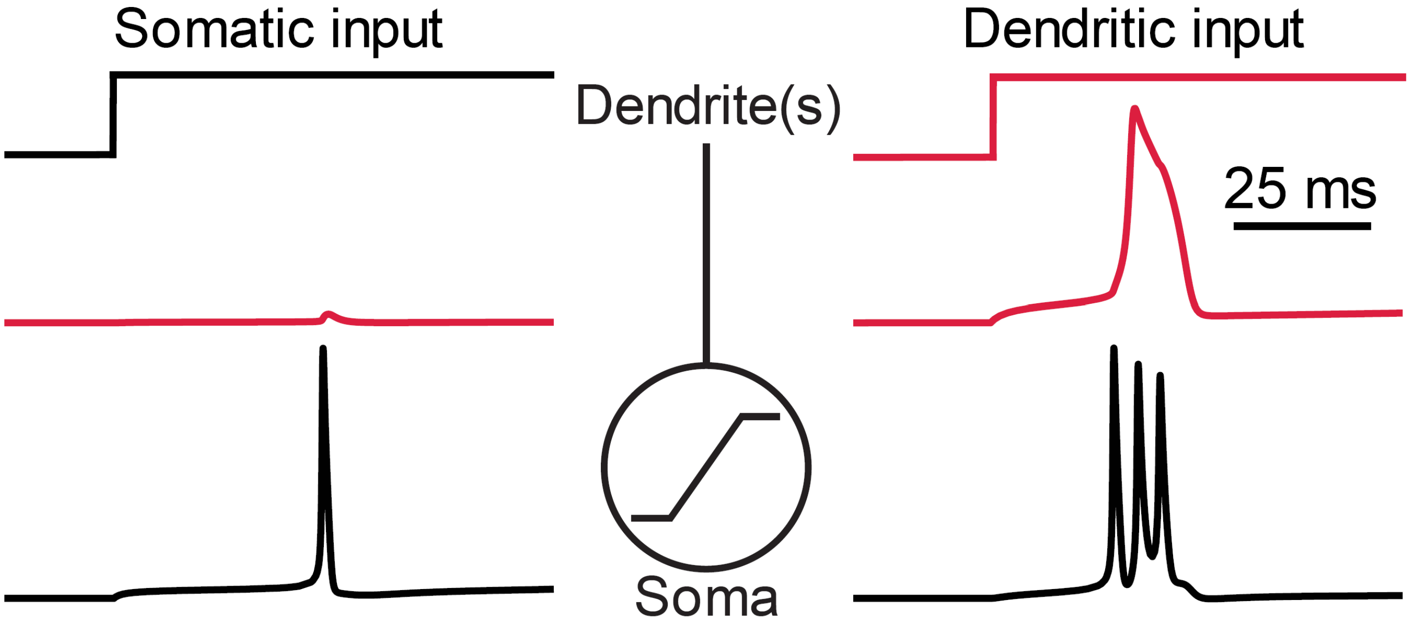


Somatic input
Dendrite(s)
Dendritic input
25 ms
Soma

A minimal but biologically-realistic model of neural spiking

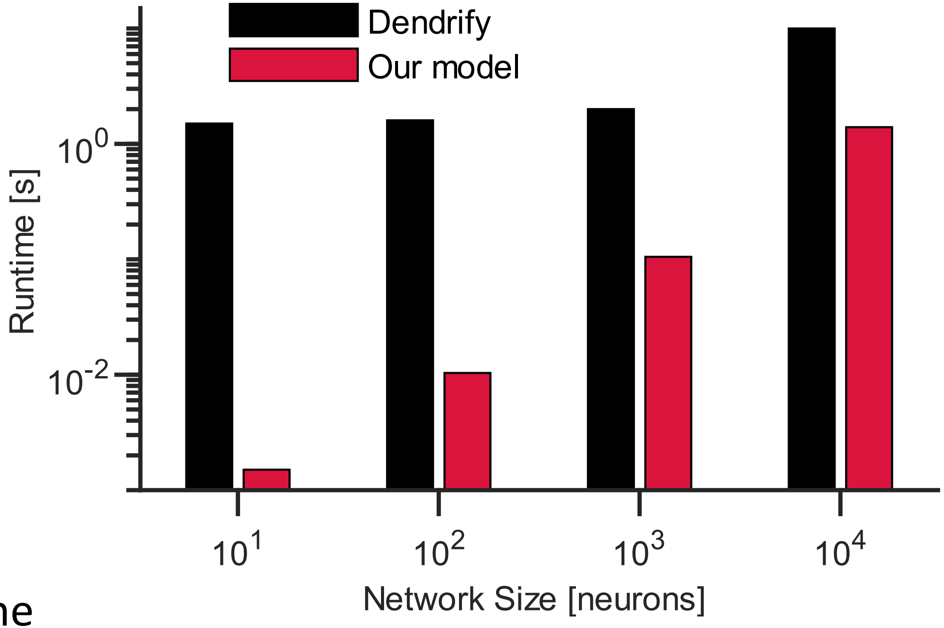


A minimal but biologically-realistic model of neural spiking



Implemented in Julia!

A minimal but biologically-realistic model of neural spiking

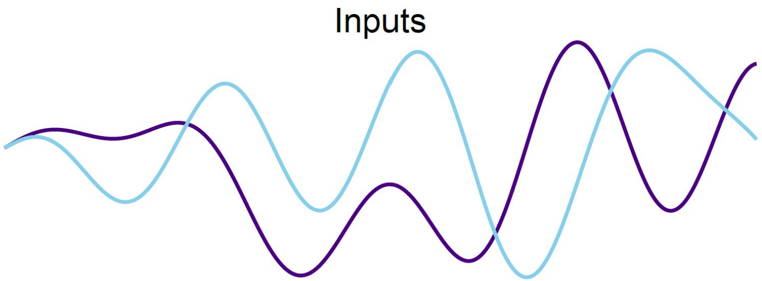


- 1.0 s simulation time
- Euler with 0.1 ms time step
- Neglecting recurrence for fair comparison
- Dendrify timings include “build time”

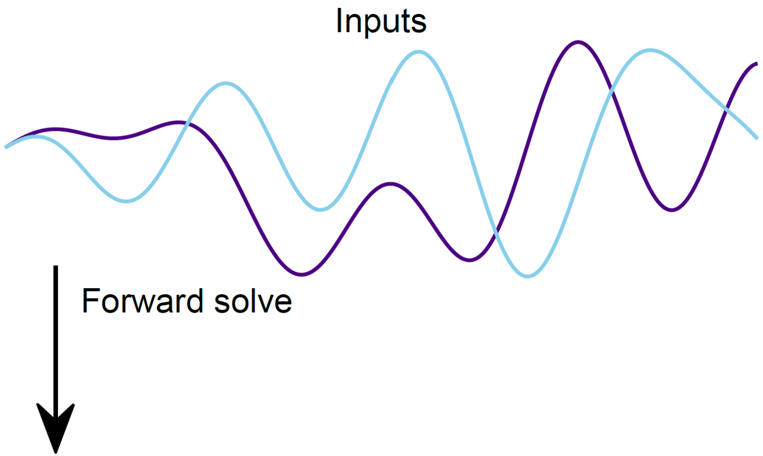
Pagkalos et al., 2023

Training via adjoint sensitivity analysis

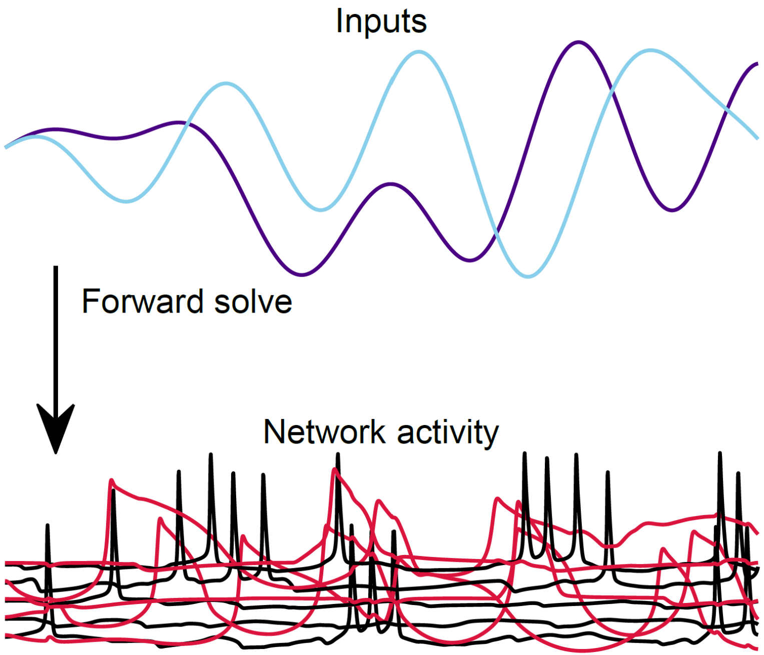
Training via adjoint sensitivity analysis



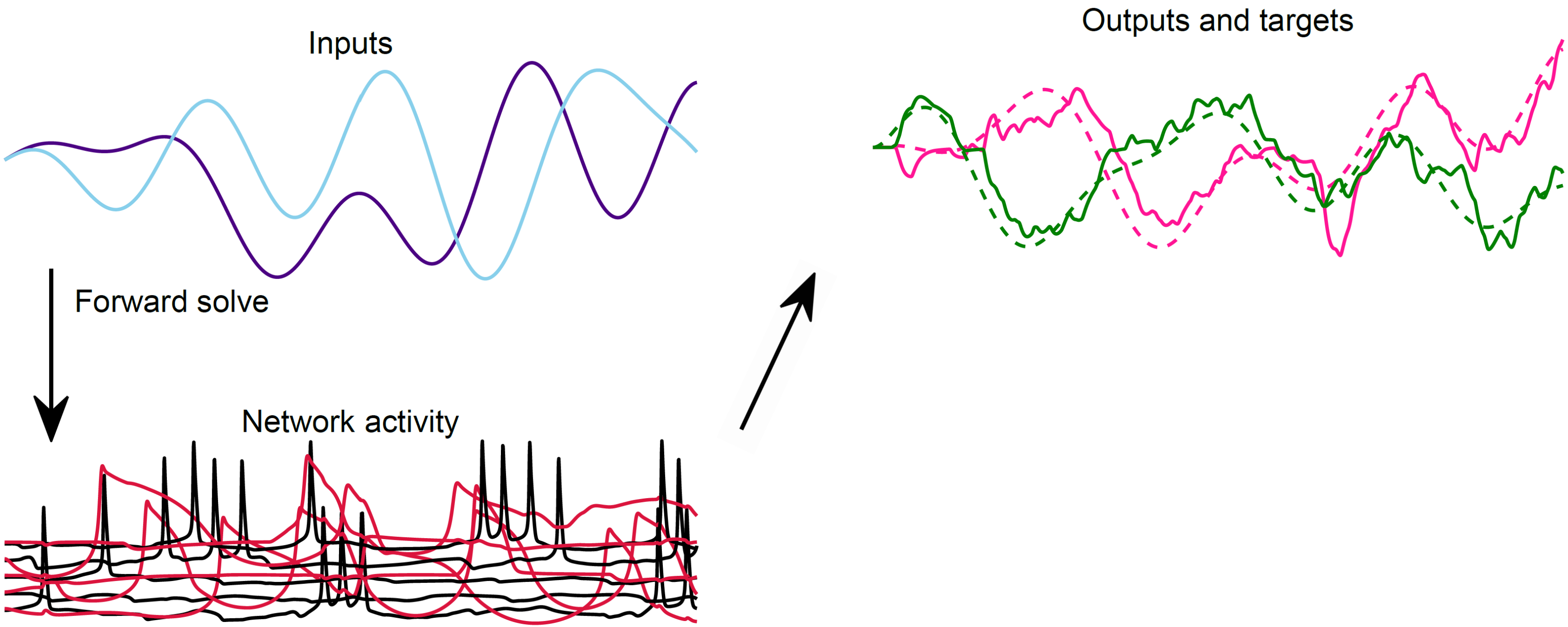
Training via adjoint sensitivity analysis



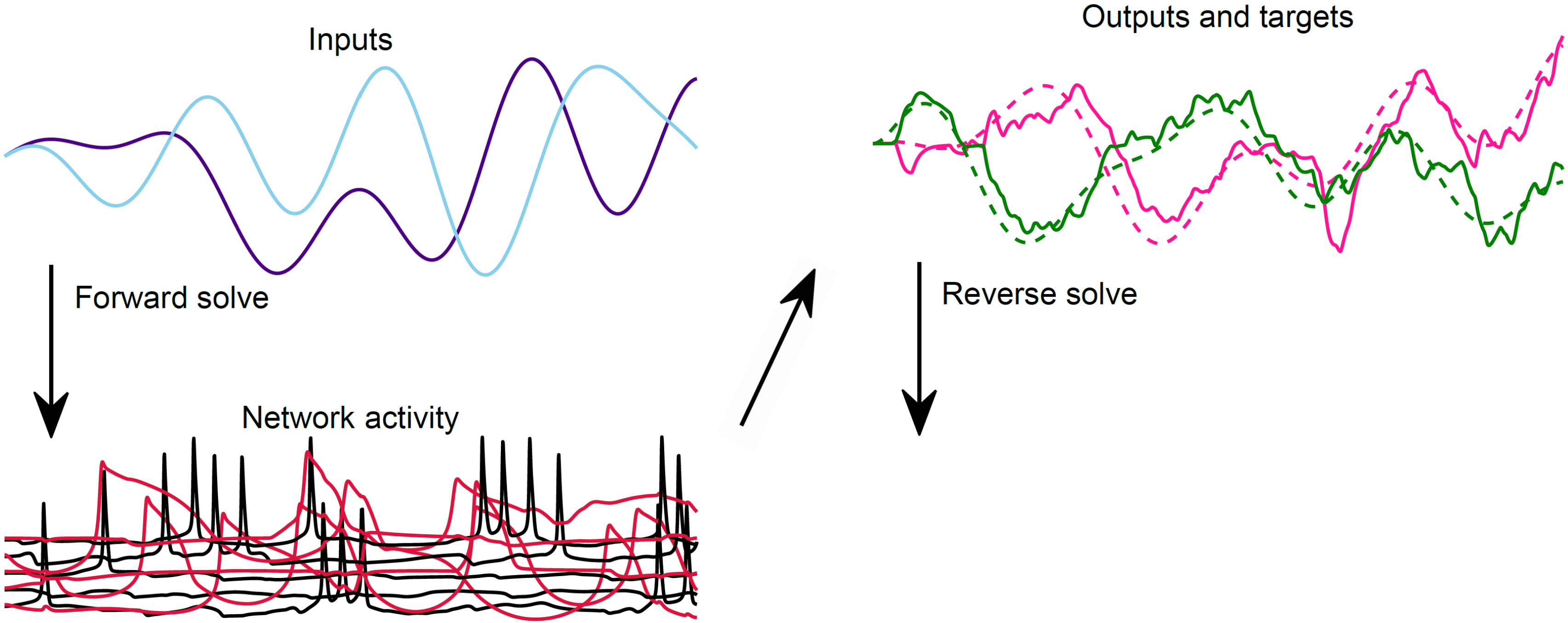
Training via adjoint sensitivity analysis



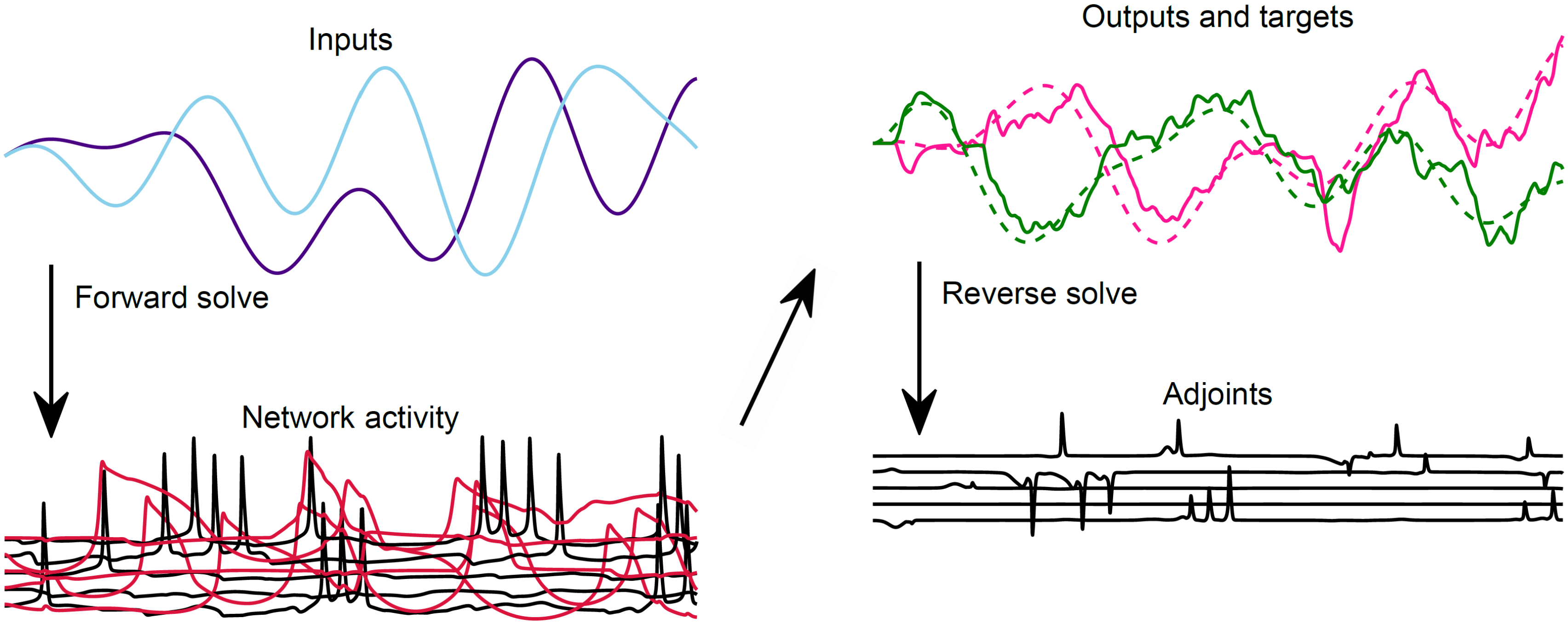
Training via adjoint sensitivity analysis



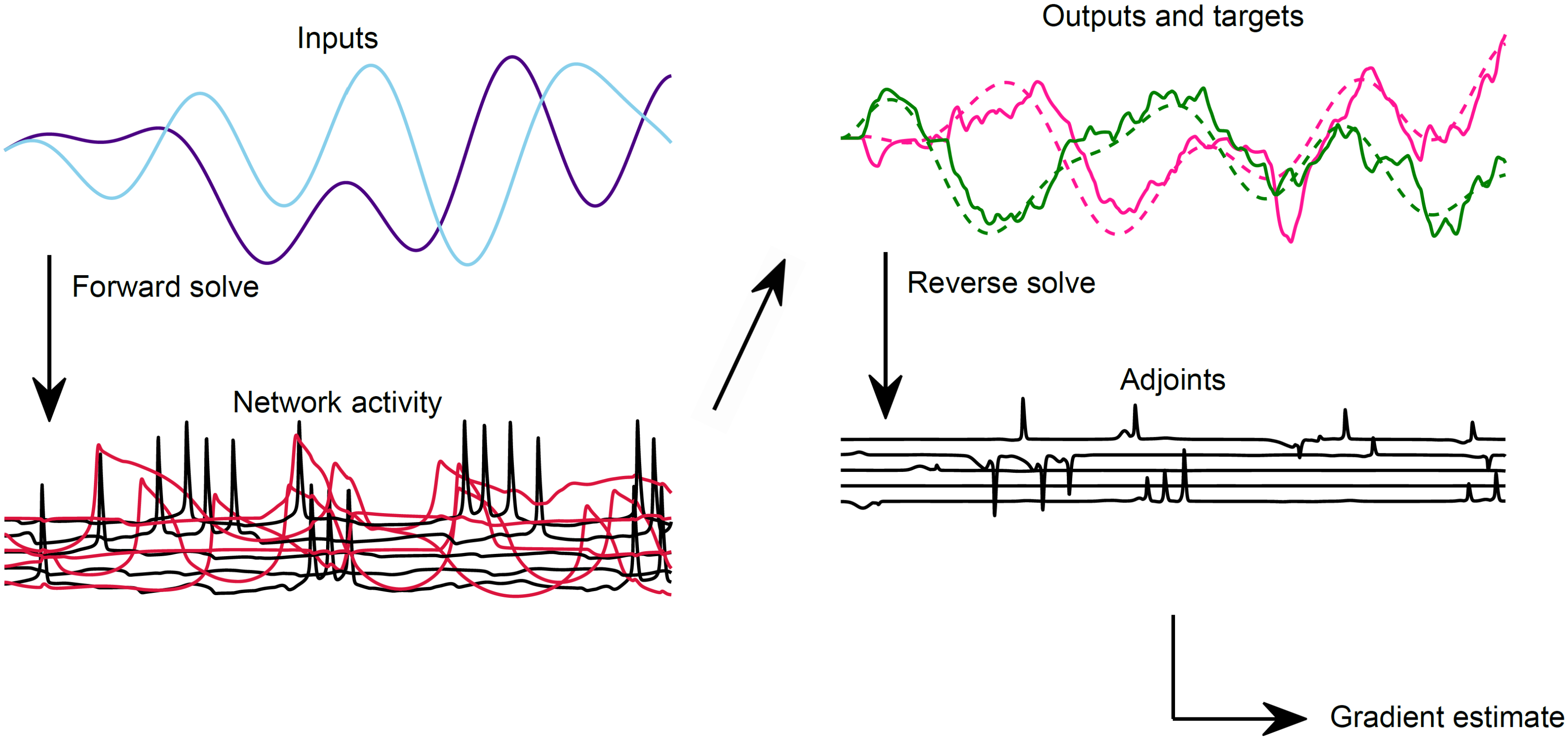
Training via adjoint sensitivity analysis



Training via adjoint sensitivity analysis



Training via adjoint sensitivity analysis



Recap

Recap

1. Each compartment is comprised of 3 coupled ODEs

Recap

1. Each compartment is comprised of 3 coupled ODEs

2. A network is made up of *N* of these compartments connected by synaptic weight matrices

Recap

1. Each compartment is comprised of 3 coupled ODEs

2. A network is made up of N of these compartments connected by synaptic weight matrices

3. We simulate network activity for 100 – 1000 ms in the forward solve

Recap

1. Each compartment is comprised of 3 coupled ODEs

2. A network is made up of *N* of these compartments connected by synaptic weight matrices

3. We simulate network activity for 100 – 1000 ms in the forward solve

4. If training, we perform a reverse solve by integrating backwards in time and interpolating in our forward solve variables

Recap

1. Each compartment is comprised of 3 coupled ODEs
2. A network is made up of *N* of these compartments connected by synaptic weight matrices
3. We simulate network activity for 100 – 1000 ms in the forward solve
4. If training, we perform a reverse solve by integrating backwards in time and interpolating in our forward solve variables
5. Compute the gradients

Recap

1. Each compartment is comprised of 3 coupled ODEs

2. A network is made up of *N* of these compartments connected by synaptic weight matrices

3. We simulate network activity for 100 – 1000 ms in the forward solve

4. If training, we perform a reverse solve by integrating backwards in time and interpolating in our forward solve variables

5. Compute the gradients

6. Repeat >10,000 times

Computational needs

Computational needs

- Training is bottlenecked by serial steps with limited opportunities for parallelization within ODE solves

Computational needs

- Training is bottlenecked by serial steps with limited opportunities for parallelization within ODE solves

- Individual ODE solves are CPU-limited, *but not that expensive*

Computational needs

- Training is bottlenecked by serial steps with limited opportunities for parallelization within ODE solves

- Individual ODE solves are CPU-limited, *but not that expensive*

- Approach: train many networks simultaneously

subMIT usage

- Will typically submit 100 – 300 batch requests at a time
- 1 – 3 cores per job
- 9 – 12 hour runtime (with checkpointing)

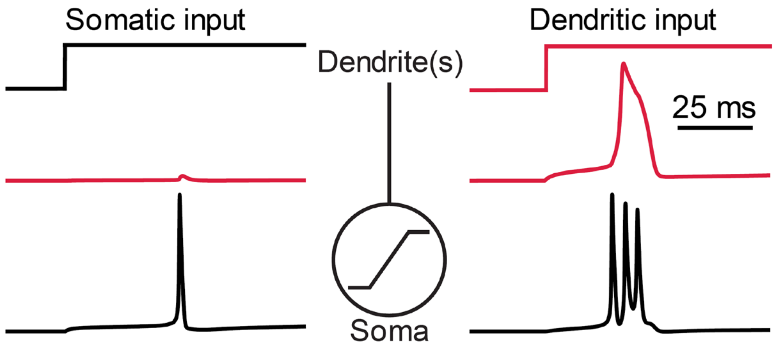
subMIT usage

- Will typically submit 100 – 300 batch requests at a time
- 1 – 3 cores per job
- 9 – 12 hour runtime (with checkpointing)

- subMIT’s cores are relatively fast, so I use it on my most expensive networks

Questions? Feedback?

A minimal but biologically-realistic model of neural spiking



$$\dot{v}_i = (a_i(v_i - b_i)^2 + c_i)(v_i - 1) - j_i u_i(v_i - l_i) - k_i w_i(v_i - m_i) + \sigma_{ji}(v_j - \mu_j) + J_i$$

$$\tau_{u_i}\dot{u}_i = d_i(v_i - e_i)^4 + f_i - u_i$$

$$\tau_{w_i}\dot{w}_i = g_i(v_i - h_i)^2 + i_i - w_i$$

$$[i, j \neq i] \in [\text{soma}, \text{dendrite}]$$

Training via adjoint sensitivity analysis

Given an instantaneous loss *L*, we seek to minimize a cost function *C* such that

$$\delta C = \delta \int_{t_1}^{t_2} dt\, L(t, \boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\theta}) = 0$$

We can perform a Legendre transformation on *L* to obtain a “Hamiltonian” that introduces new adjoint variables ***p***

$$H(t, \boldsymbol{x}, \boldsymbol{p}, \boldsymbol{\theta}) = \dot{\boldsymbol{x}} \cdot \boldsymbol{p} + L(t, \boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\theta})$$

where ***p*** are defined by

$$\dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{x}}$$

The parameter gradients of the initial cost function can then be computed as

$$\frac{\partial C}{\partial \boldsymbol{\theta}} = \int_{t_1}^{t_2} dt\, \frac{\partial H}{\partial \boldsymbol{\theta}}$$