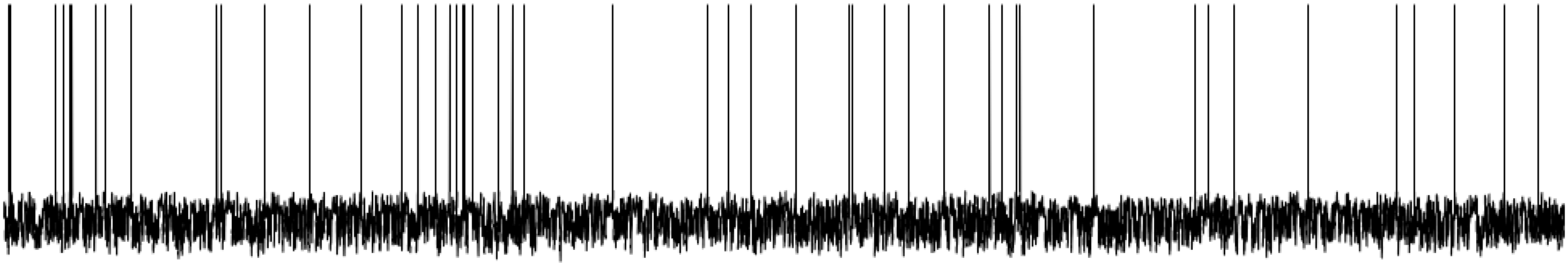
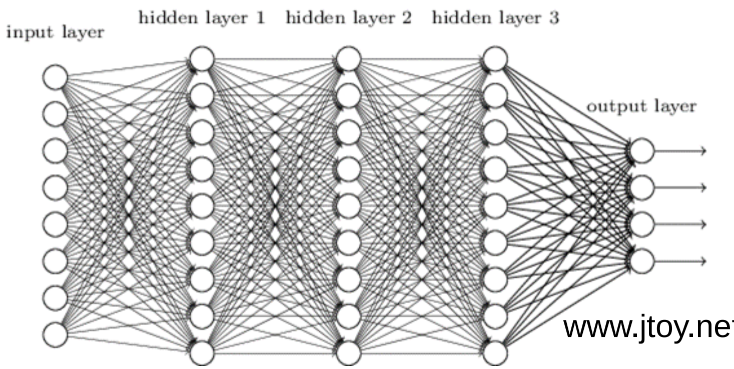
Learning in spiking neural networks

Friedemann Zenke
Surya Ganguli
@
Stanford University

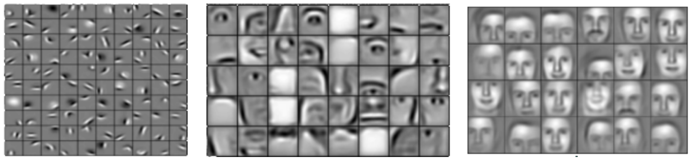
https://fzenke.net



Opposite motivations & approaches to understanding neural networks

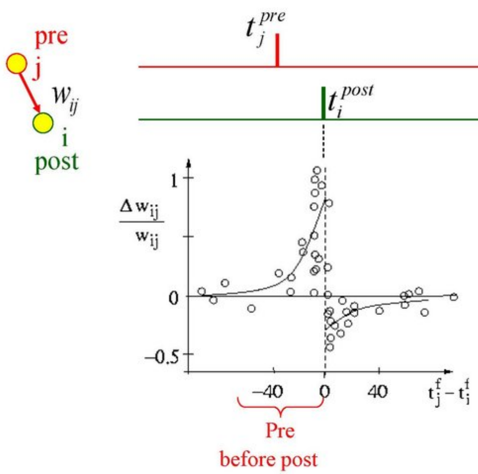


www.jtoy.net

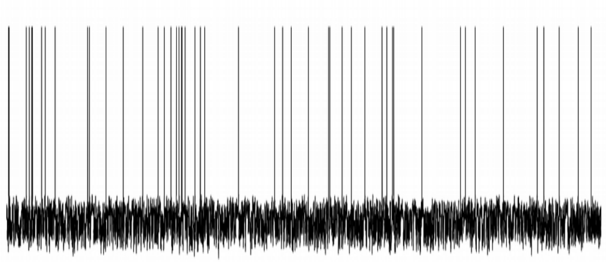


- Spikes
- STDP
- Biological plausibility
- Unsupervised learning

- MNIST, ImageNet, ...
- Classification performance
- Impressive stuff



Sjöström and Gerstner (2010)



Comp. Neuroscience vs Machine Learning

Goals
- Capture experimental data
- Understand computation in the brain

- Maximize performance on given task

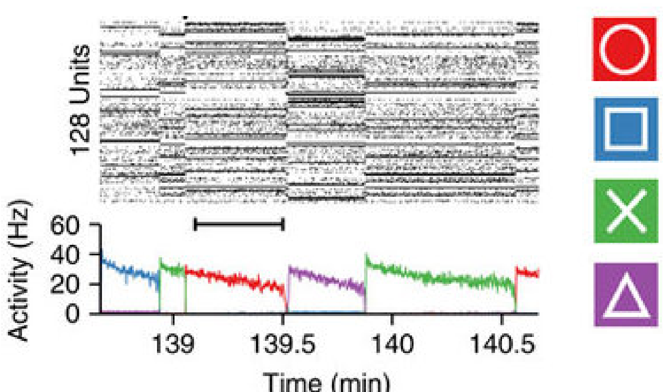
Architecture
- Spiking neurons (sparse connect.)
- Continuous time
- Non-differentiable

- Rate-based neurons (dense)
- Discrete time
- Differentiable

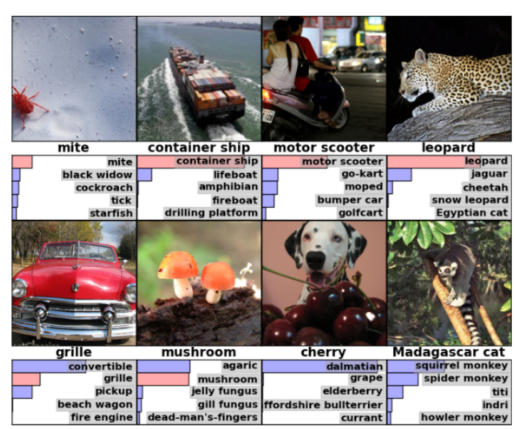
Learning
- Local learning rules
- Unsupervised/reinforcement learning
- Neuromodulators
- Non-stationary data
- Continual learning
  (learning and recall at the same time)

Training
- Global objective
- Stochastic gradient descent
- Stationary training data
- Separation training/recall

Performance



Zenke et al (2015)

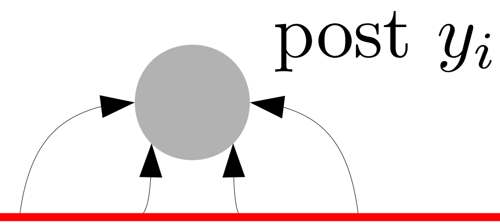


Krizhevsky et al. (2012)

Outline

- Part 1: Review of work on rapid compensatory processes
- Part 2: Supervised learning in deterministic multi-layer spiking neural networks starting from a cost function

Hebbian plasticity needs compensatory processes

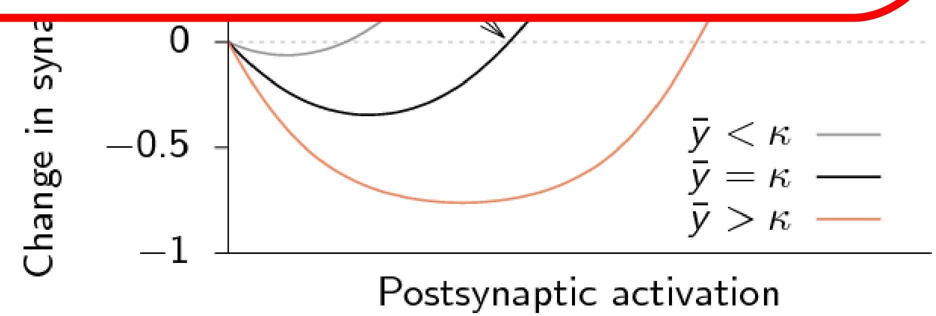


$$\frac{dw_{ij}}{dt} = \eta \left(x_j y_i - w_{ij} y_i^2 \right)$$

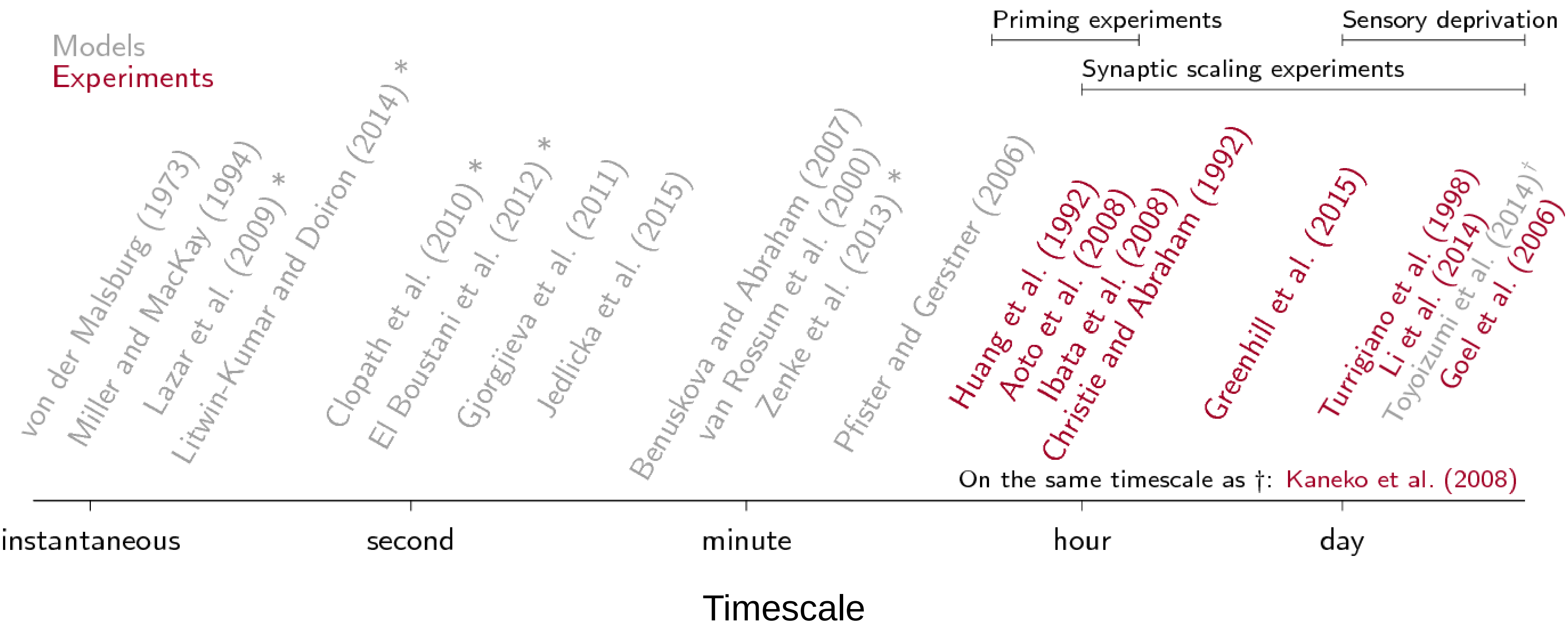
Compensatory processes are often thought to be the same as homeostatic plasticity.

$$\tau \frac{d\theta_i(t)}{dt} = -\theta_i + \kappa^{-1} y_i^2$$

BCM, Bienenstock et al. (1982)

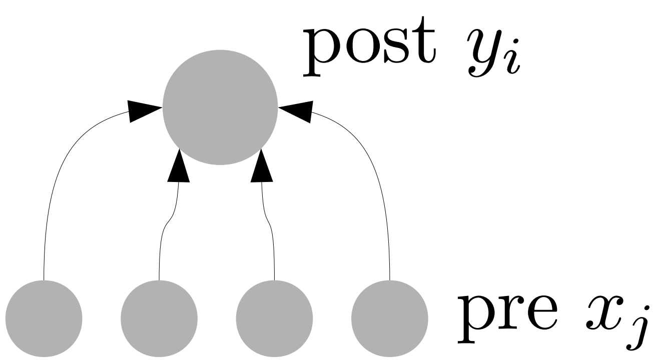


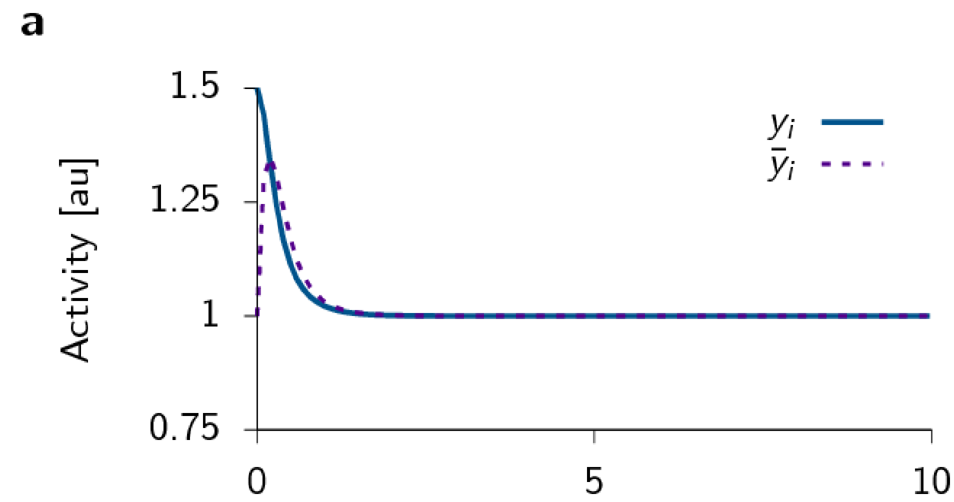
The temporal paradox of Hebbian and homeostatic plasticity



Zenke & Gerstner (2017) Phil. Trans. R. Soc. B

Why Hebbian plasticity needs **rapid** compensatory processes





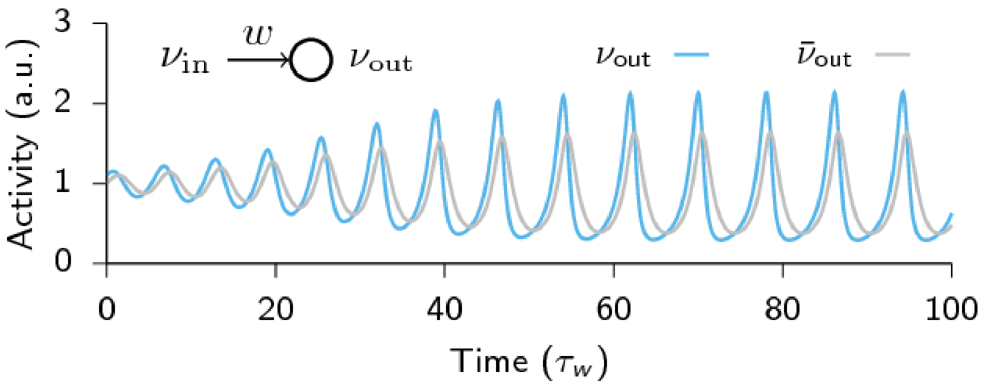
$$\frac{dw_{ij}}{dt} = \eta \left(x_j y_i - w_{ij} \bar{y}_i^2 \right)$$

Zenke & Gerstner (2017) Phil. Trans. R. Soc. B

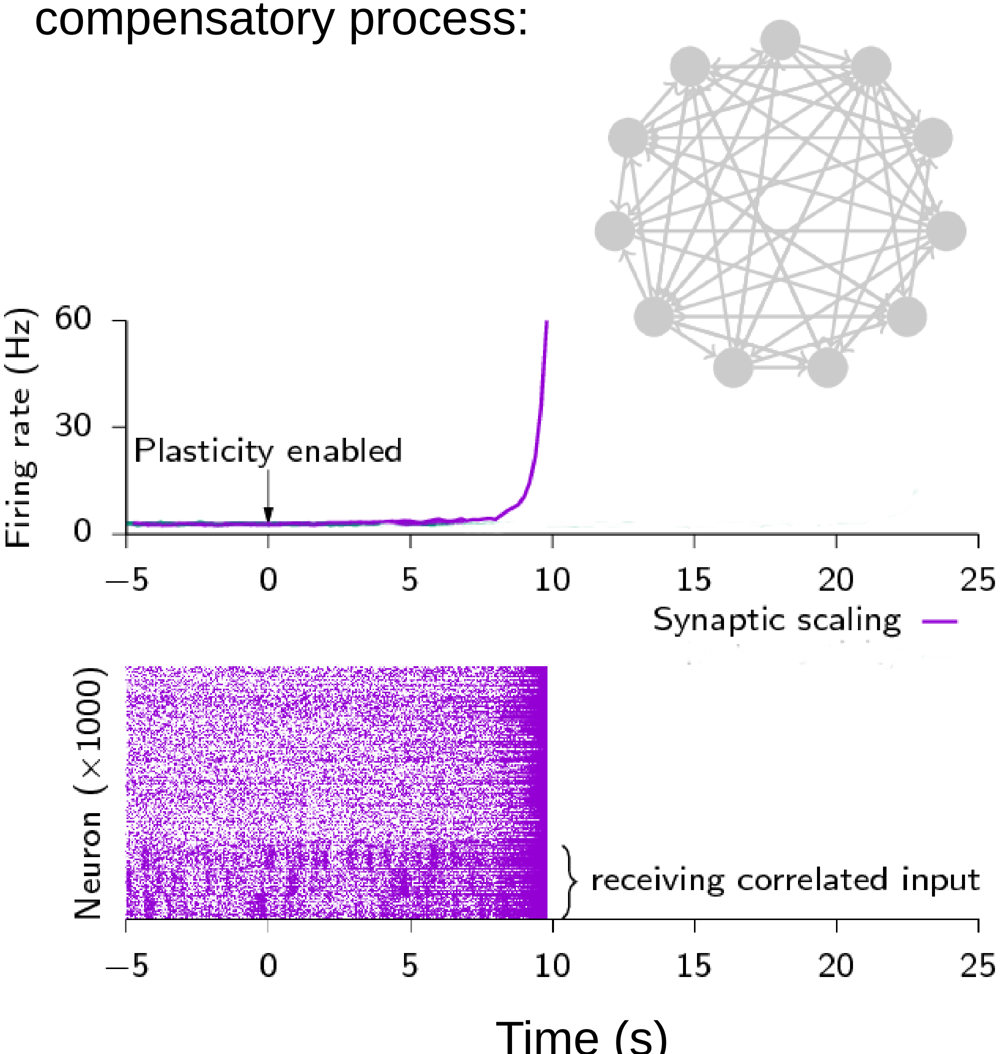
Bio inspired plasticity needs **rapid** compensatory processes

BCM with slow compensatory process:

$$\frac{dw_{ij}}{dt} = \eta x_j \phi\left(y_i - \theta(t)\right)$$



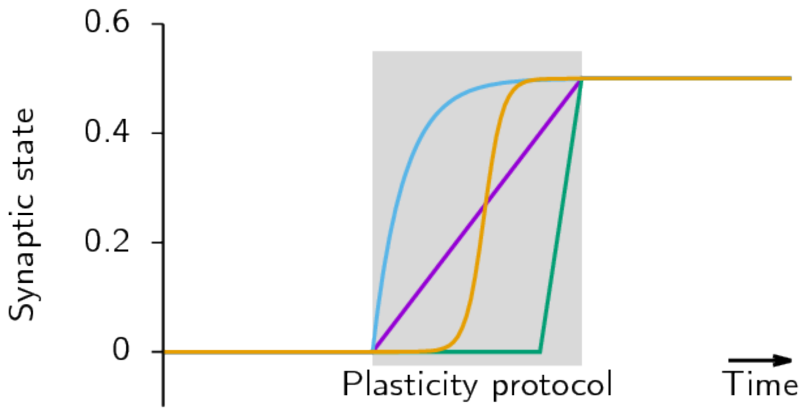
Spiking network with STDP with slow compensatory process:



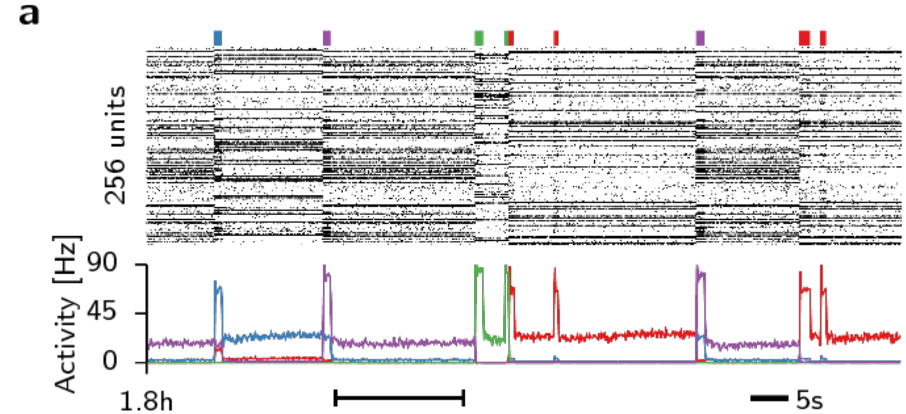
Zenke, Gerstner & Ganguli (in revision)

Bio inspired learning rules often lack rapid compensatory processes

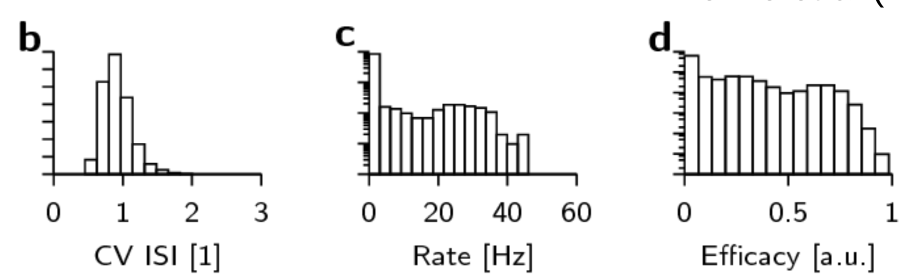
- Bio inspired learning rules (bottom up)
  - Often unstable
  - Under constrained
  - Add rapid compensatory processes for stability
- Functionally motivated rules (top down)
  - Derived from objective function
  - Stability/rapid comp. processes built in



Stable latching dynamics of cell assemblies in plastic spiking net



Zenke et al (2015)



- Attractor states = fixed points of plasticity
- Negative feedback terms in the learning rule

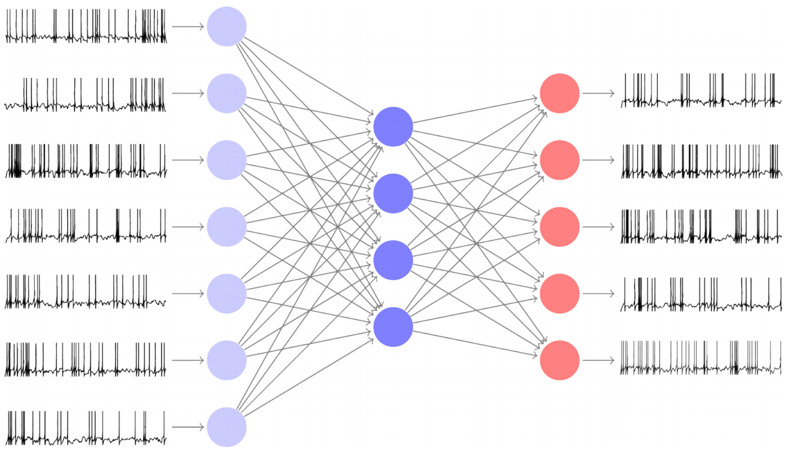
More on this: Zenke & Gerstner (2017) Phil. Trans. R. Soc. B
Zenke, Gerstner & Ganguli (in revision)
preprints on fzenke.net

Outline

- Part 1: Review of work on rapid compensatory processes
- Part 2: Supervised learning in deterministic multi-layer spiking neural networks starting from a cost function

Desiderata

- Spiking network which solves complex task
- Use spike timing
- Algorithm which could be implemented by a biological synapse

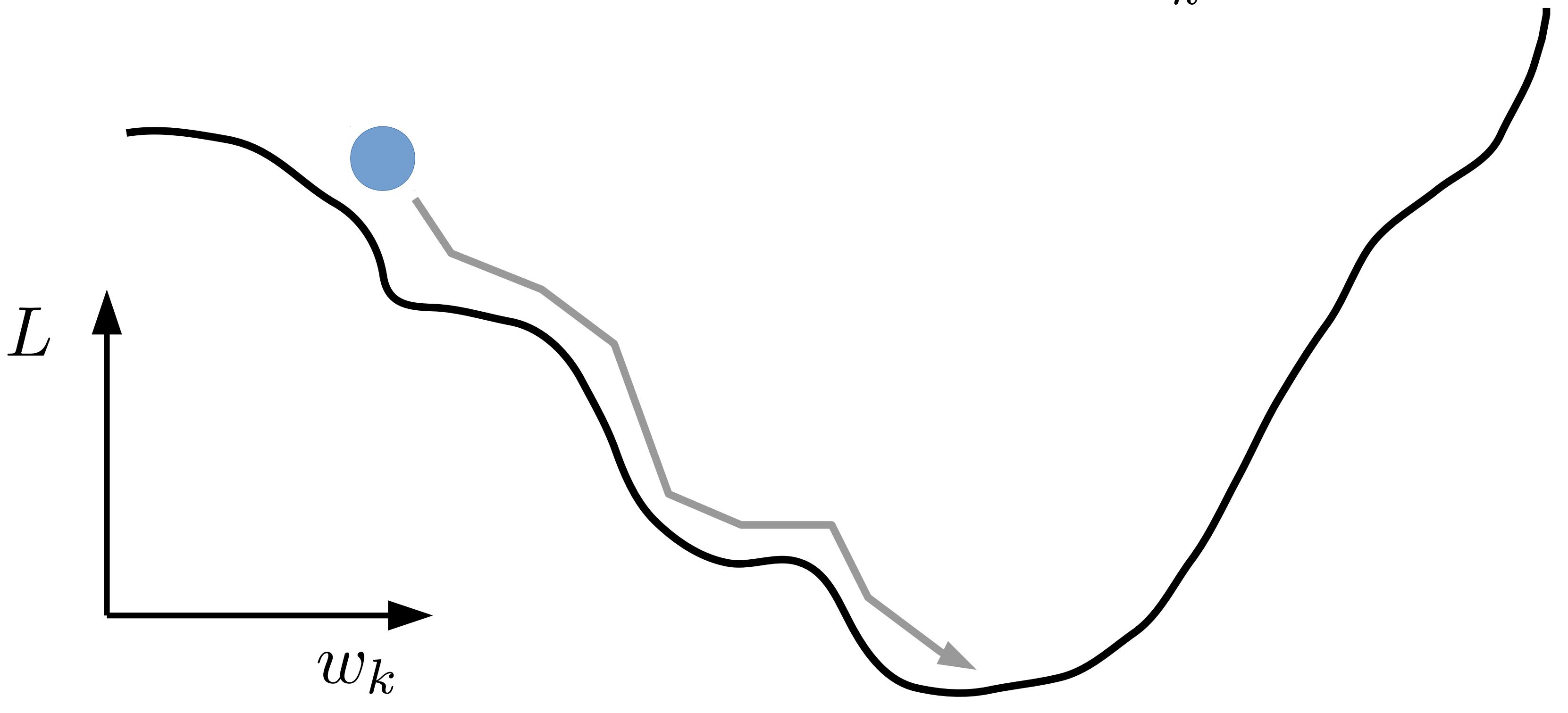


Aim

Get spiking networks to do something interesting, by starting from an objective function approach.

"Smooth" machine learning approach

- Start with suitable cost function L
- Take the derivative w.r.t. to parameters
- Do gradient descent $\frac{\partial w_k}{\partial t} \propto -\frac{\partial L}{\partial w_k}$



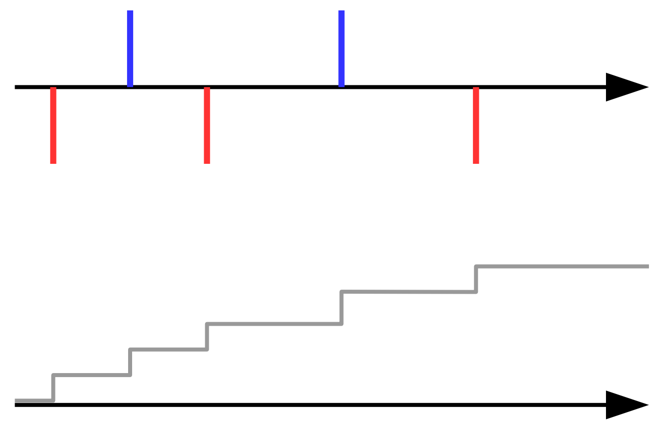
Problems with ML approach for spiking neural networks

- Suitable cost function for spike trains
- Spikes are inherently non-differentiable
- Spiking neurons have history dependence due to spike reset
- Credit assignment in hidden layers

Which spike train metric to use

Spike train:

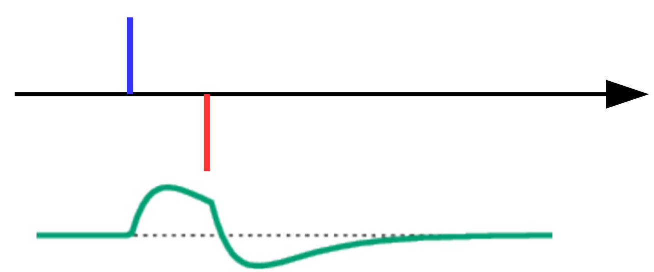
$$S_i(t) = \sum_k \delta(t - t_i^k)$$



$$L_{\mathrm{naive}} = \int_{-\infty}^{t} \left| \hat{S}_i(s) - S_i(s) \right| ds$$

van Rossum distance:

$$L = \int_{-\infty}^{t} \left(\epsilon * \hat{S}_i(s) - \epsilon * S_i(s) \right)^2 ds$$



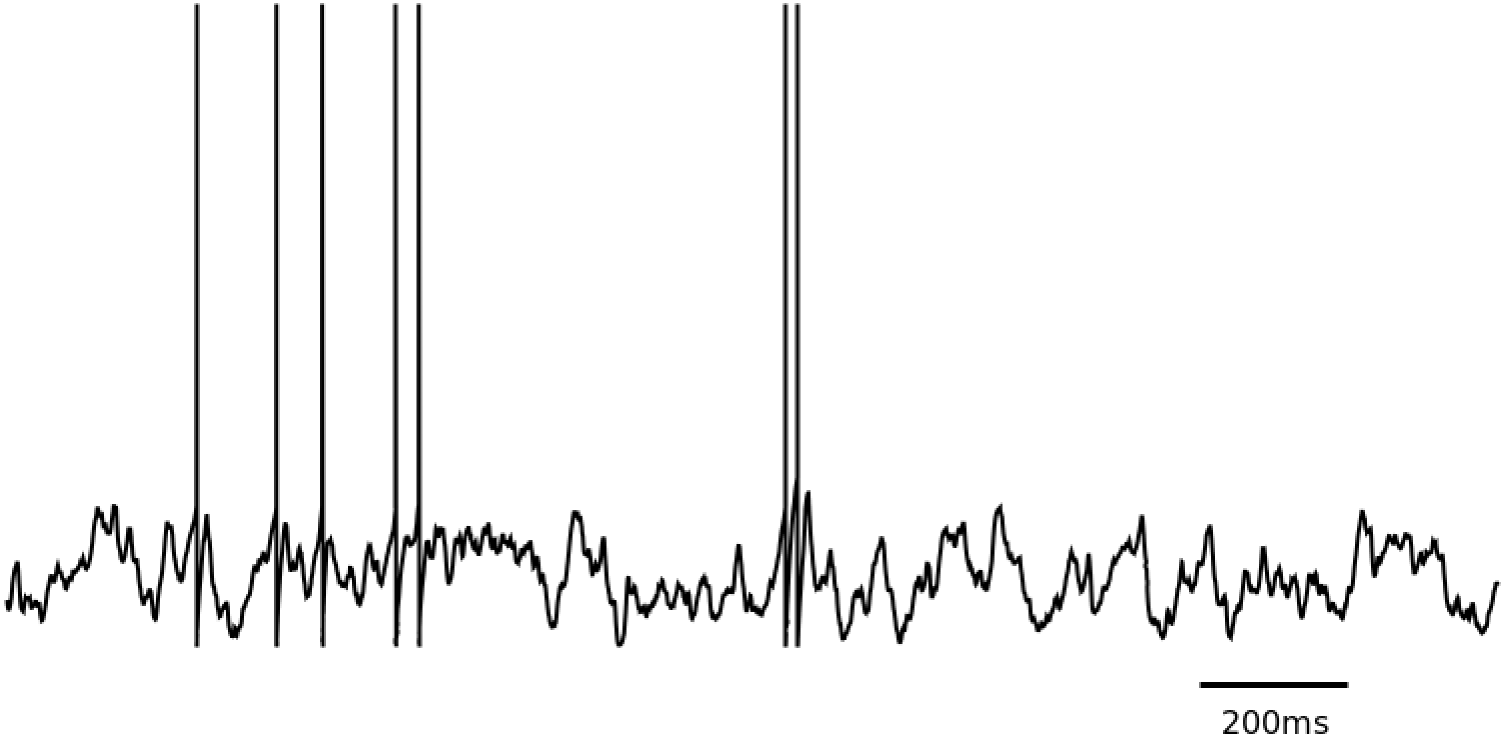
Gardner & Grunig 2015

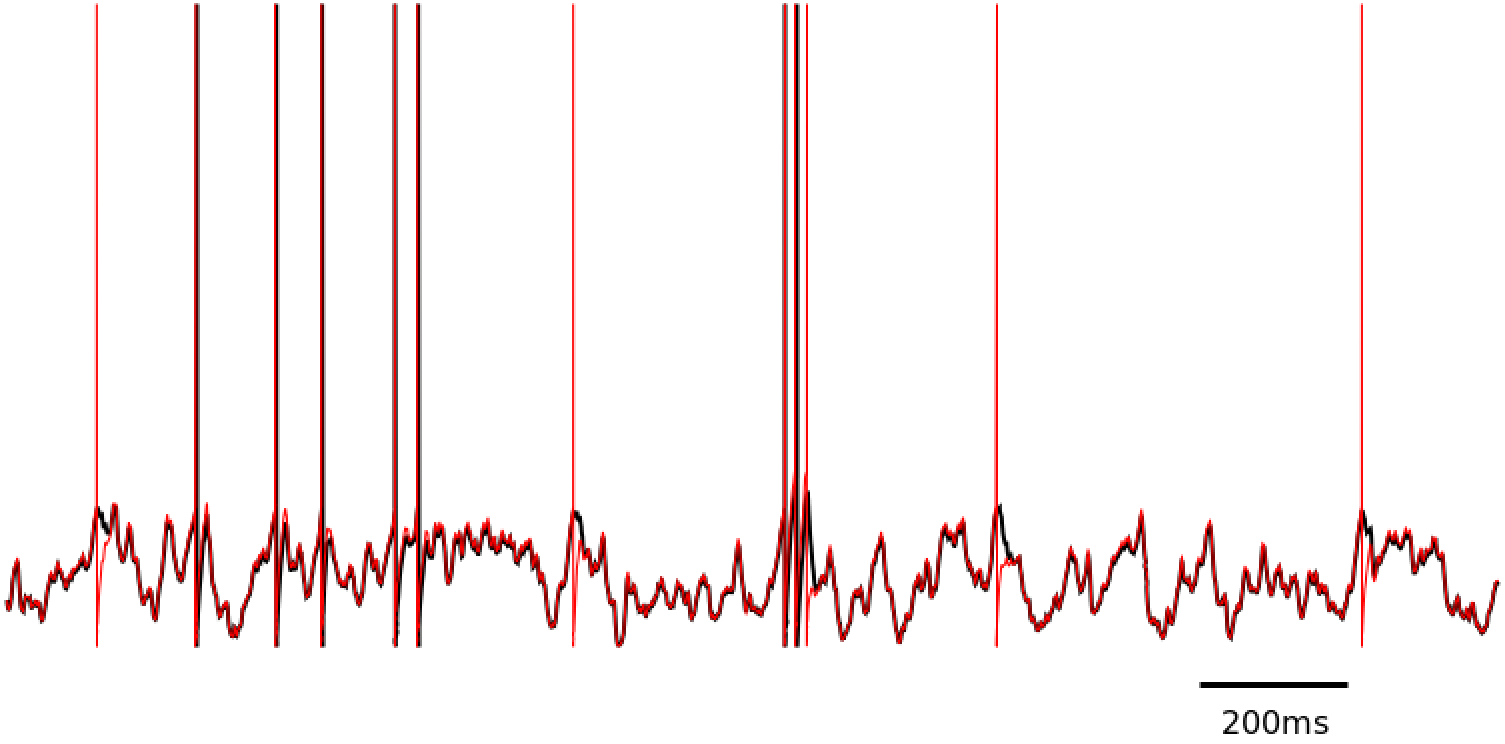
Derivative of a spike train is zero almost everywhere

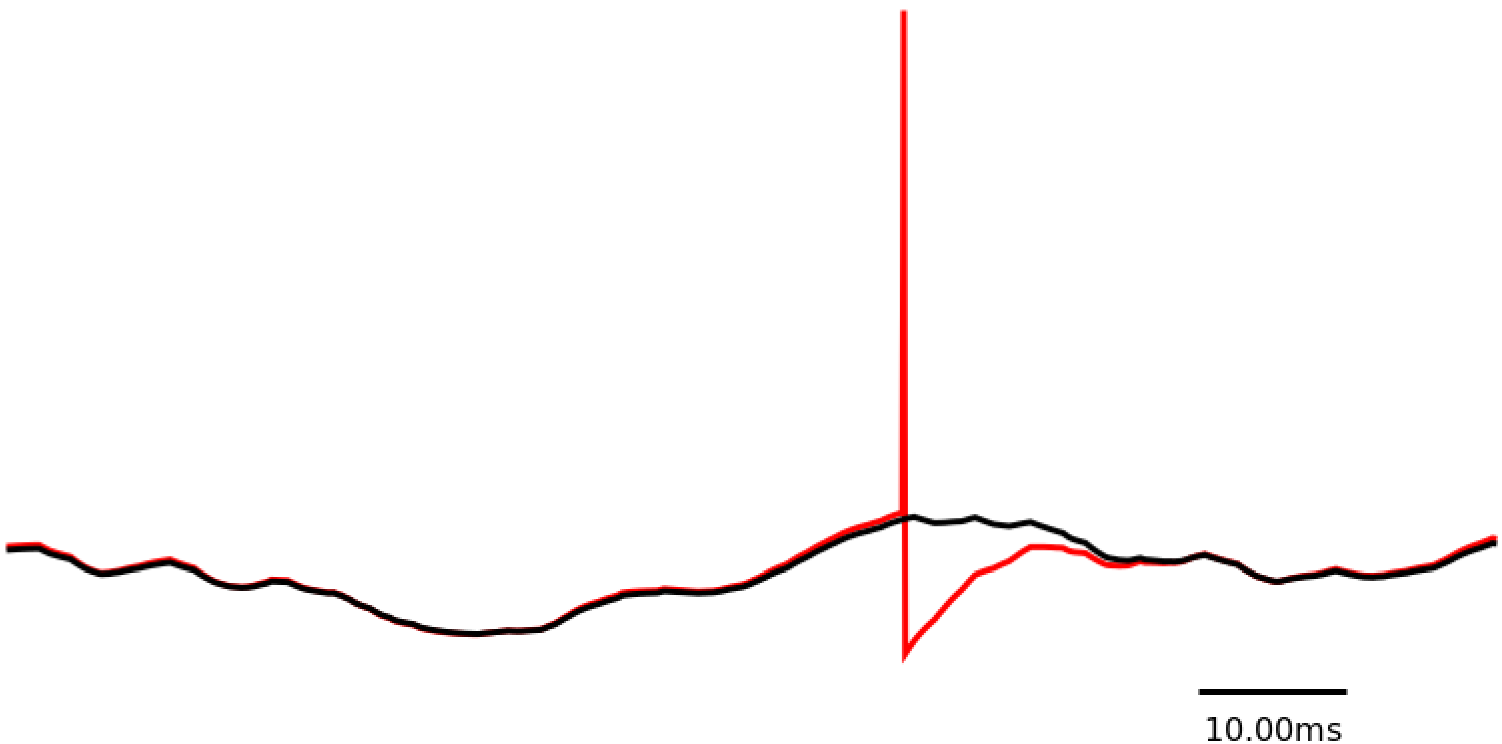
$$L = \frac{1}{2} \int_{-\infty}^{t} \left(\epsilon * \left(\hat{S}_i(s) - S_i(s) \right) \right)^2 ds$$

D'Oh!

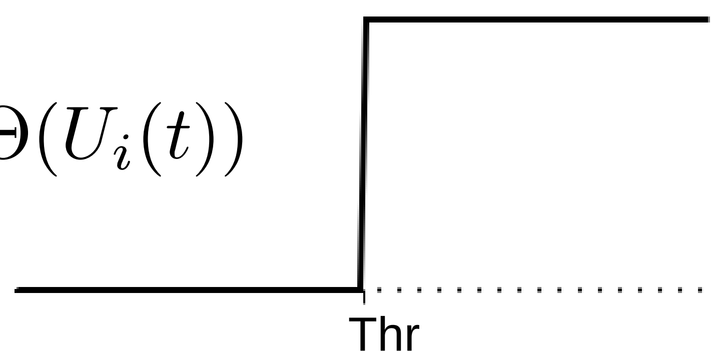
$$-\frac{\partial L}{\partial w_k} = \int_{-\infty}^{t} \epsilon * \left(\hat{S}_i(s) - S_i(s) \right) \epsilon * \frac{\partial S_i(t)}{\partial w_k} ds$$


200ms

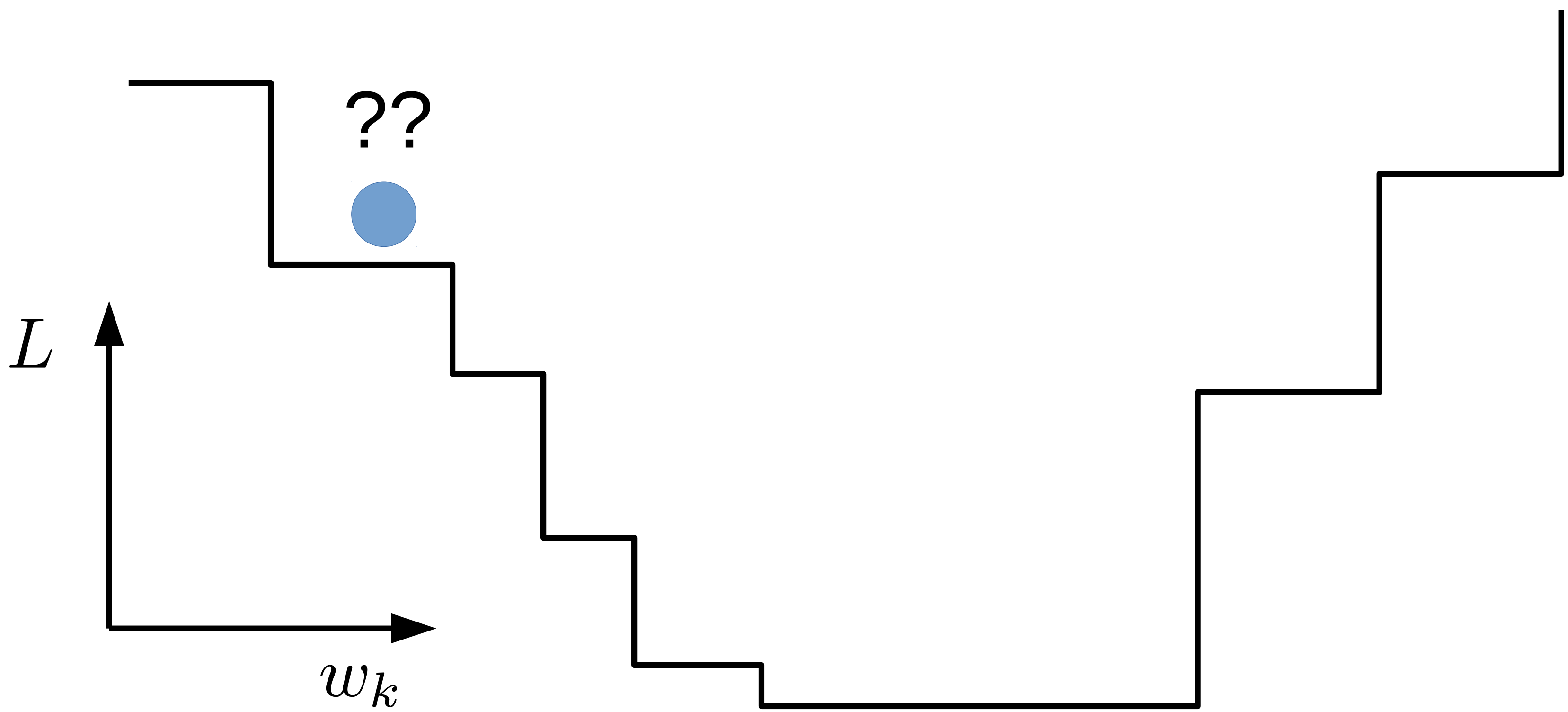

200ms



$$S(U_i(t)) \propto \Theta(U_i(t))$$

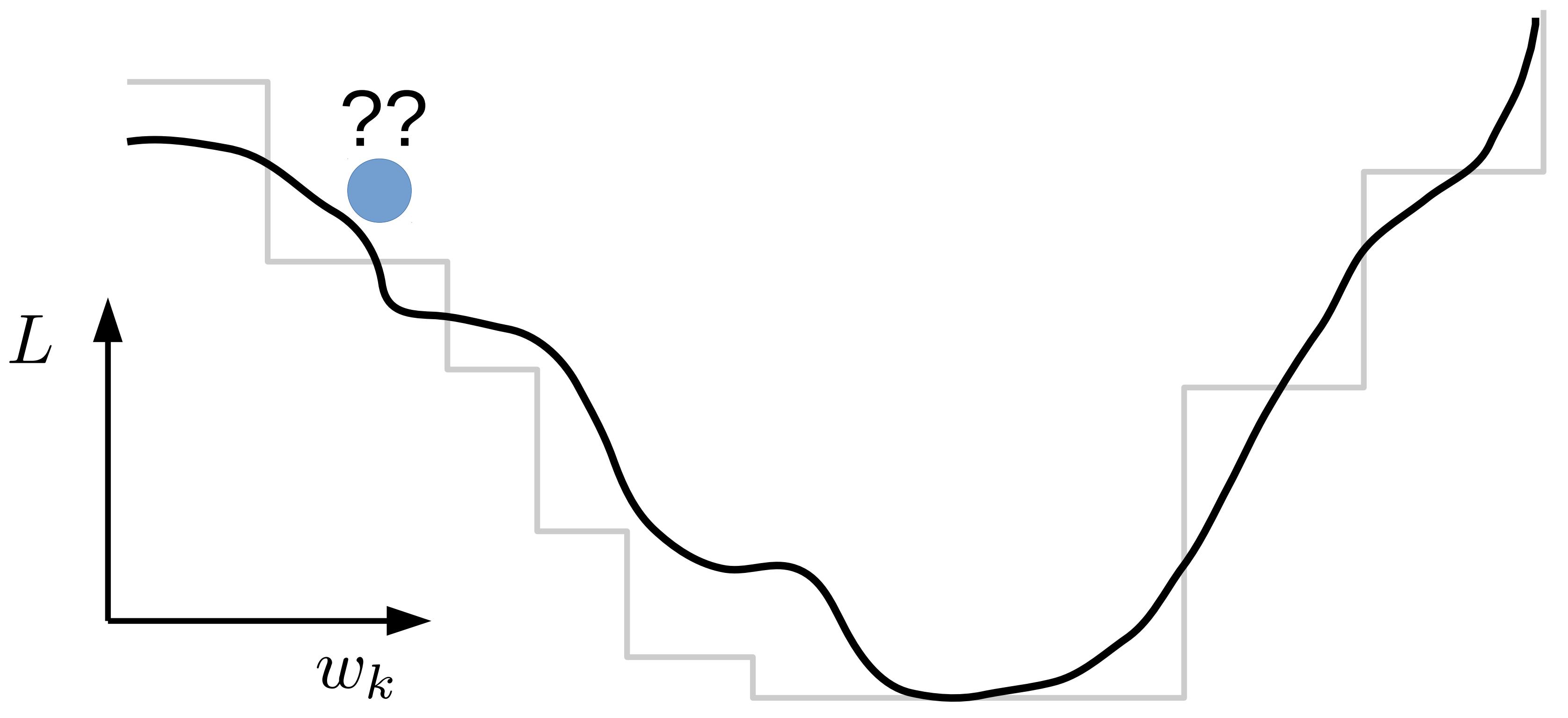


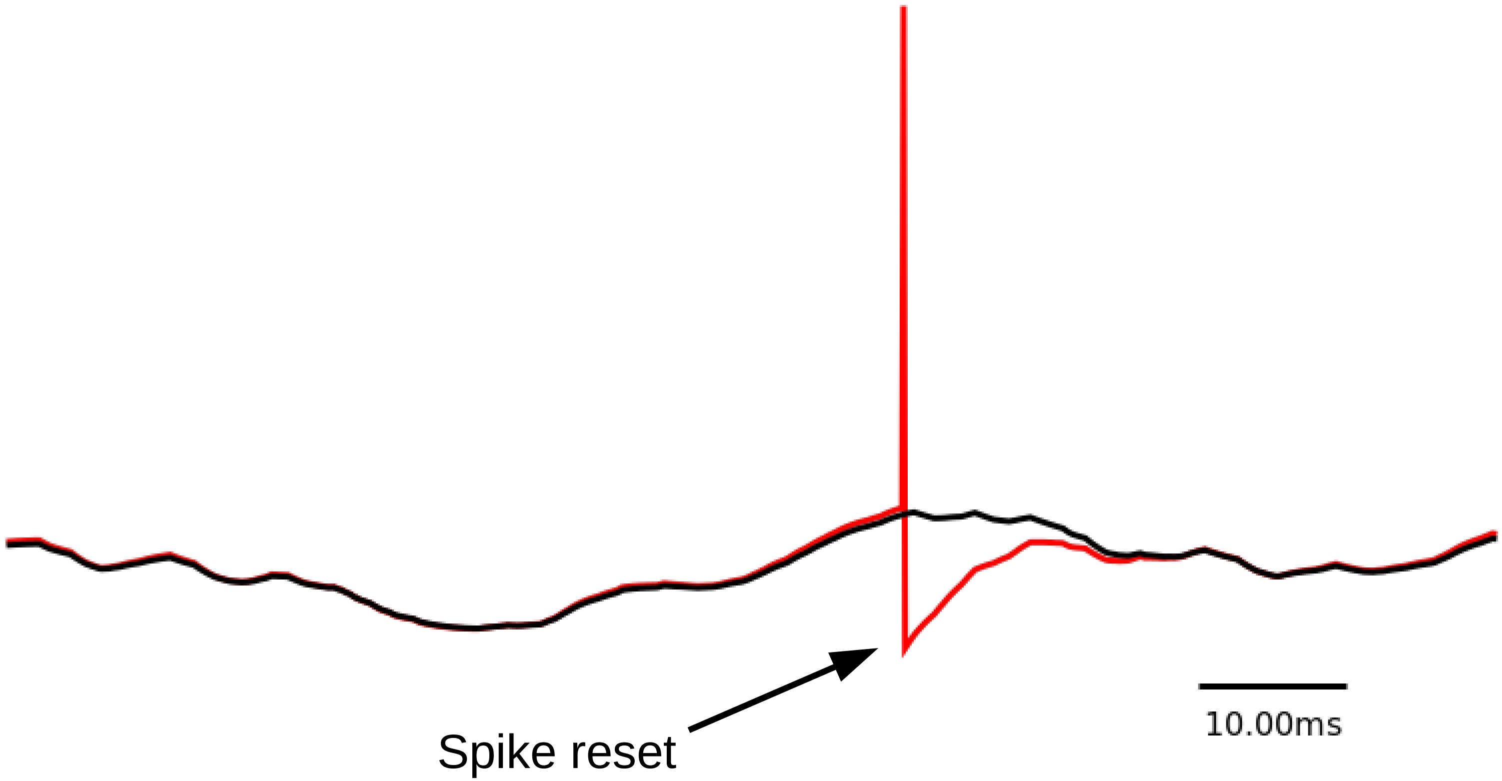
Optimization landscape in spiking neural networks flat everywhere



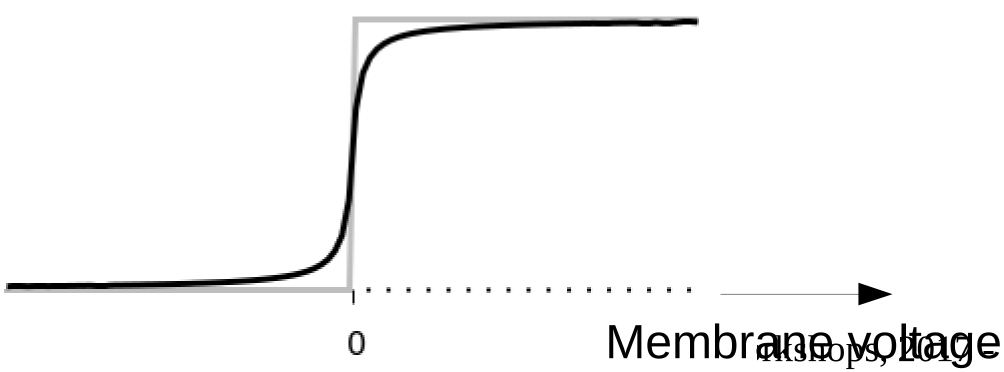
Common approach: Smooth out the optimization landscape

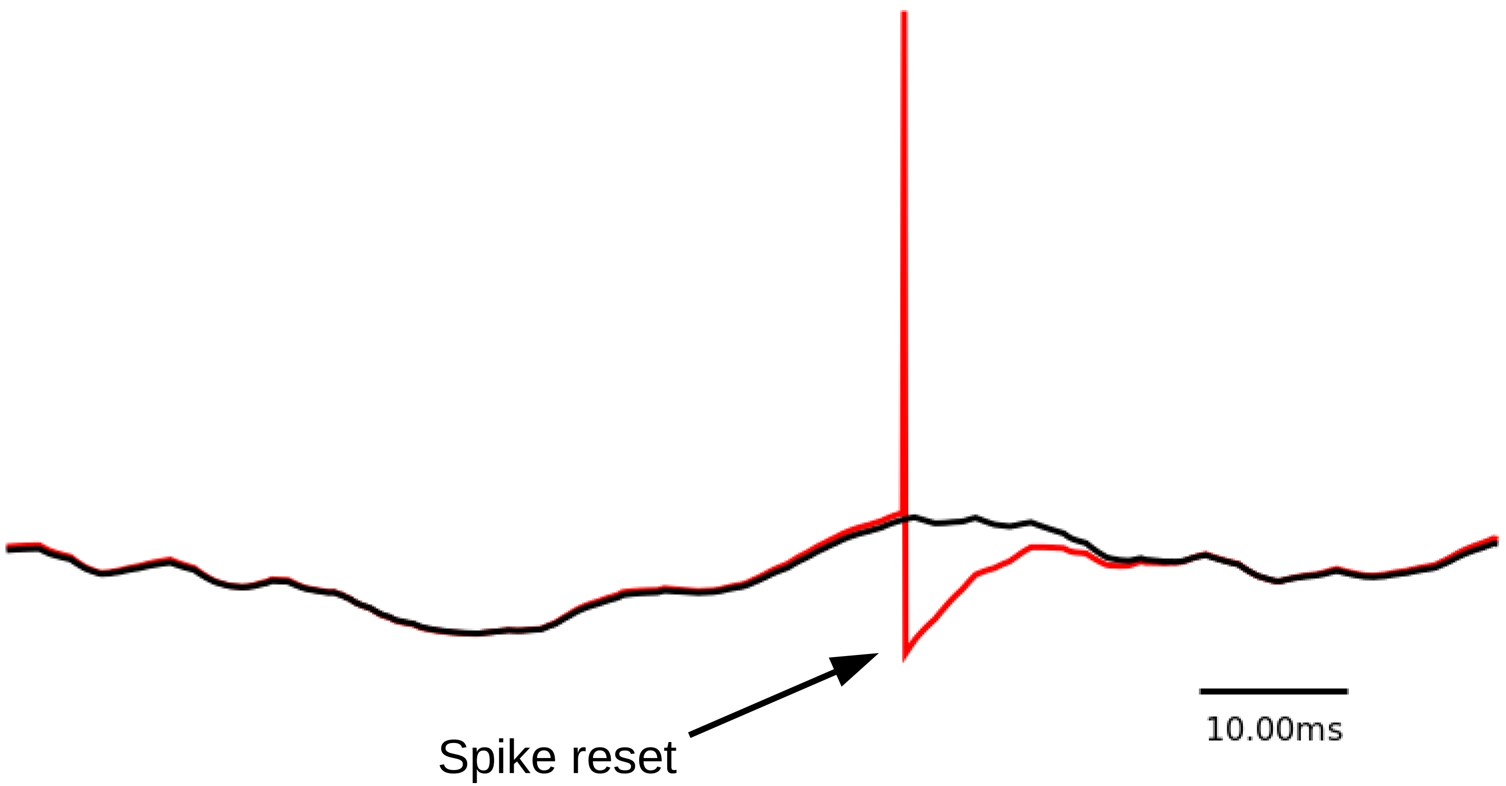
- Probabilistic model / add noise
- Non-zero approximations to the gradient



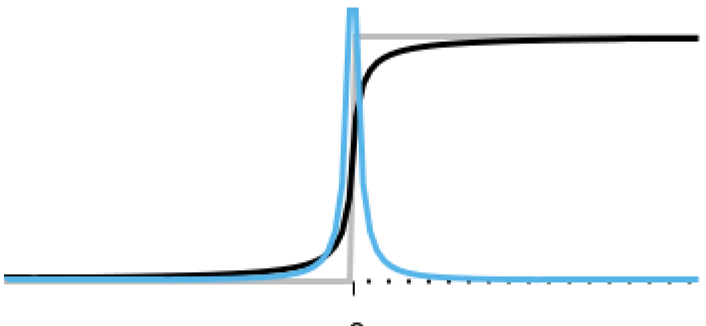


$$S(U_i(t)) \propto \Theta(U_i(t)) \approx \sigma(U_i(t))$$





$$S(U_i(t)) \propto \Theta(U_i(t)) \approx \sigma(U_i(t))$$

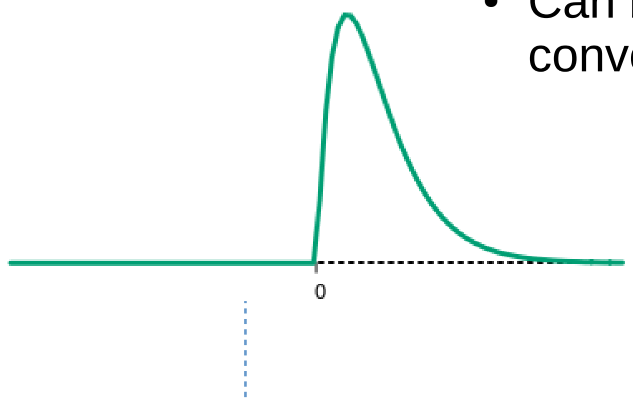


$$\frac{\partial S_i}{\partial w_{ij}} \approx \sigma'(U_i)\frac{\partial U_i}{\partial w_{ij}}$$

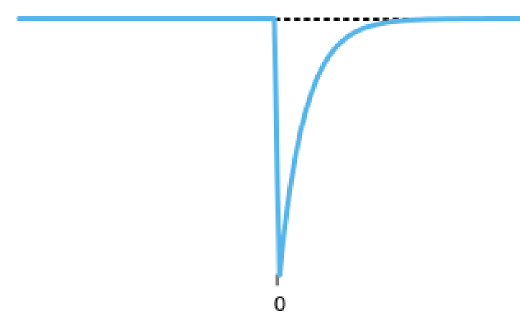
need $\frac{\partial U_i}{\partial w_{ij}}$

- Leaky integrate-and-fire neuron model
- Dynamics defined by ODE
- Equivalent to the spike-response-model (SRM0)
- Can be written with temporal convolutions

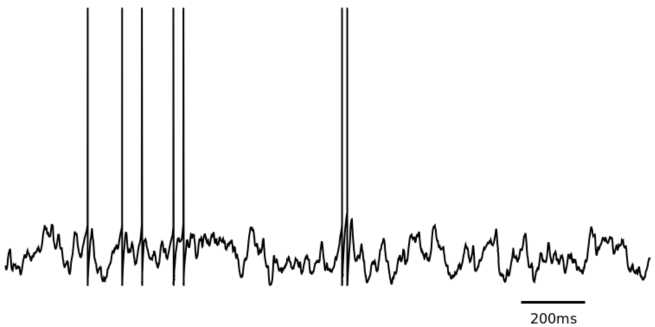
PSP kernel ϵ



Reset kernel η



Membrane voltage



$$U_i(t) = \sum_j w_{ij} \epsilon * S_j(t) + \cancel{\eta * S_i(t)}$$

$$\frac{\partial S_i}{\partial w_{ij}} \approx \underbrace{\sigma'(U_i)}_{\text{post}} \underbrace{\epsilon * S_j(t)}_{\text{pre}}$$

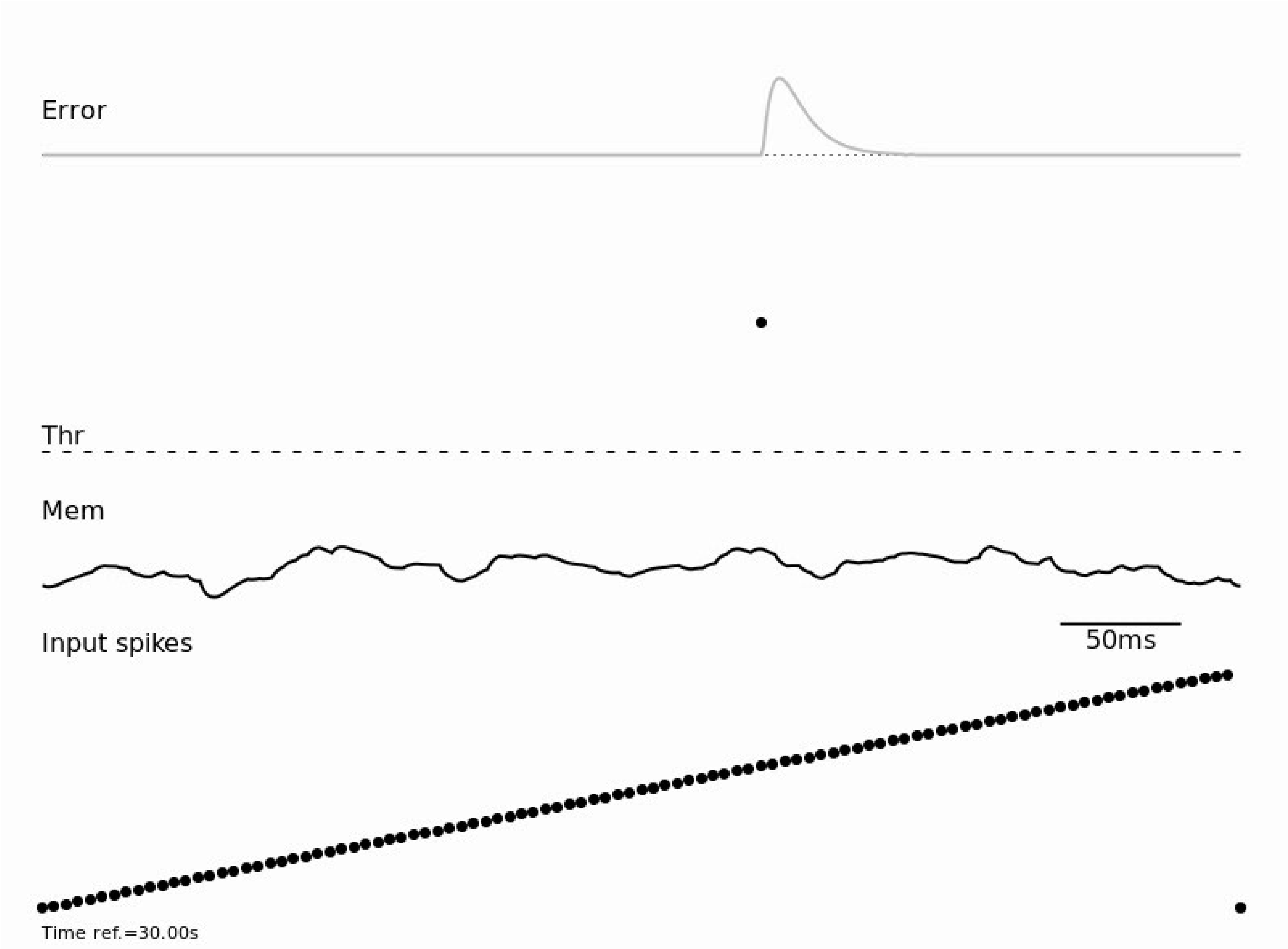
The update equation can be interpreted as a three factor rule

$$-\frac{\partial L}{\partial w_k} = \int_{-\infty}^{t} \underbrace{\epsilon * \left(\hat{S}_i(s) - S_i(s)\right)}_{\equiv \mathrm{e_i(t)}} \epsilon * \frac{\partial S_i(t)}{\partial w_k} ds$$

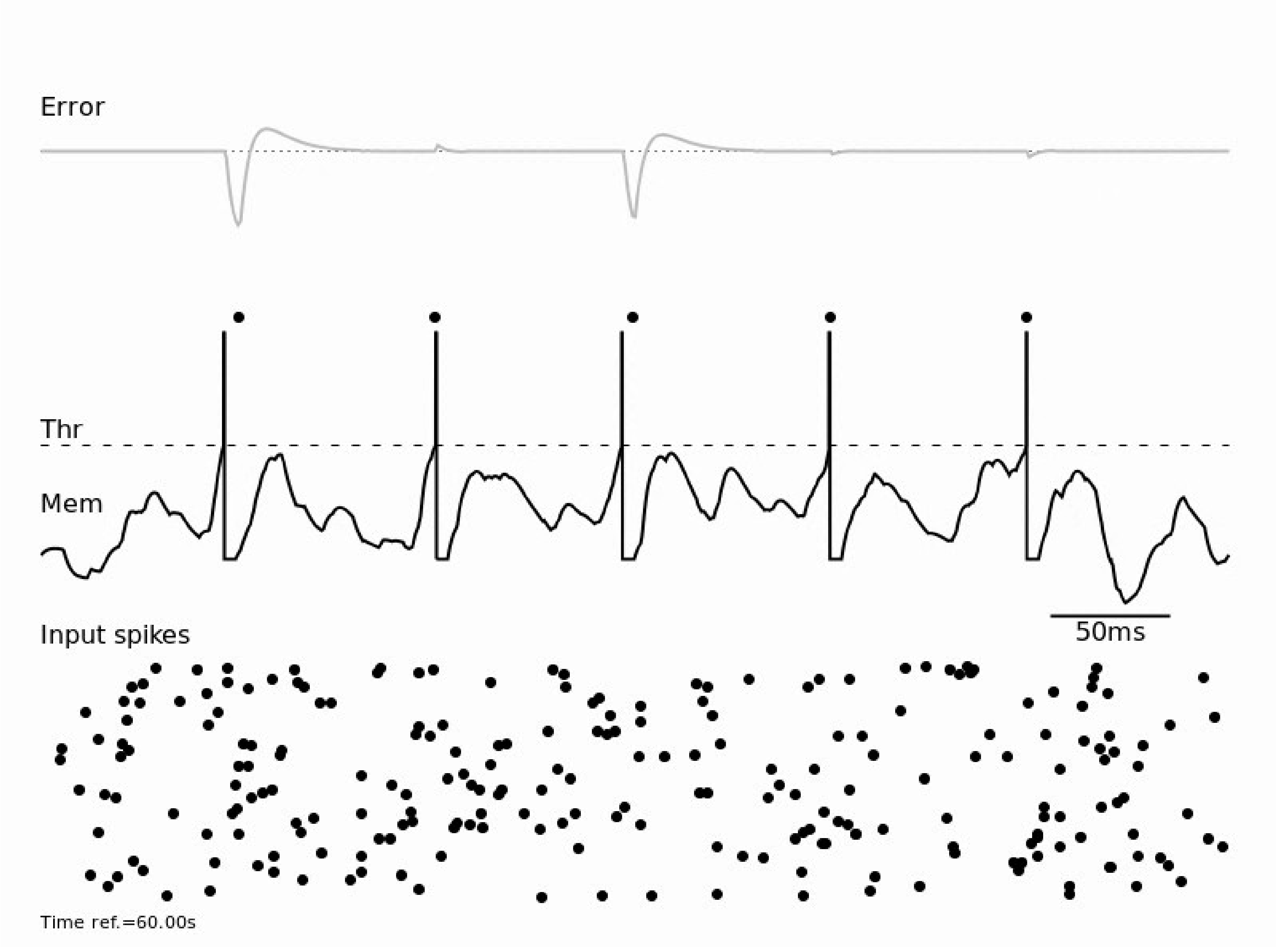
$$\frac{\partial w_{ij}}{\partial t} \equiv \underbrace{e_i(t)}_{\text{error signal}} \epsilon * (\underbrace{\epsilon * S_j(t)}_{\text{pre}} \underbrace{\sigma'(U_i)}_{\text{post}})$$

- Three factor rule
- Non-linear Hebbian
- Voltage-based
- Think of outer convolution as eligibility trace

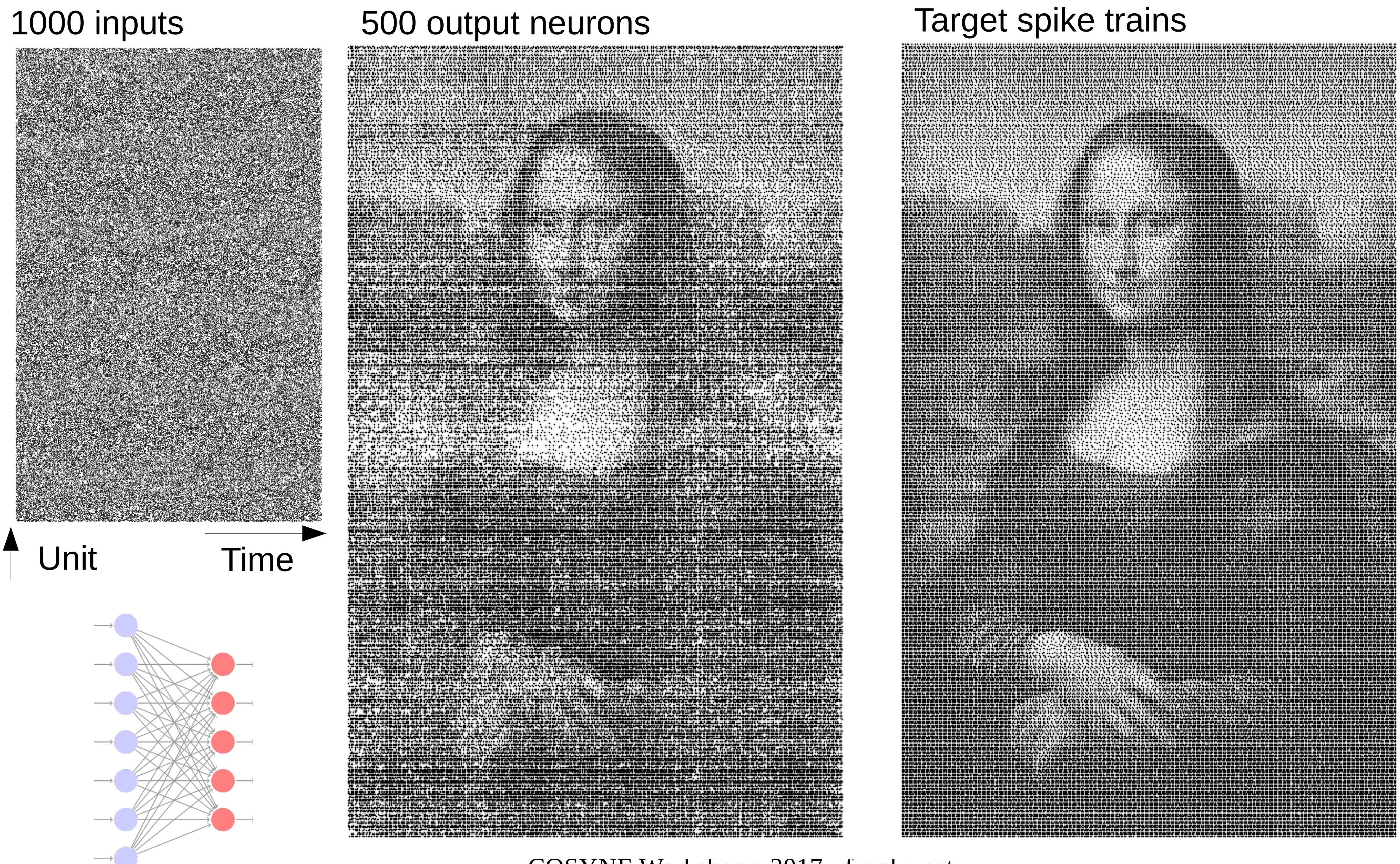
Sequential inputs → single spike



Learning of output spike train



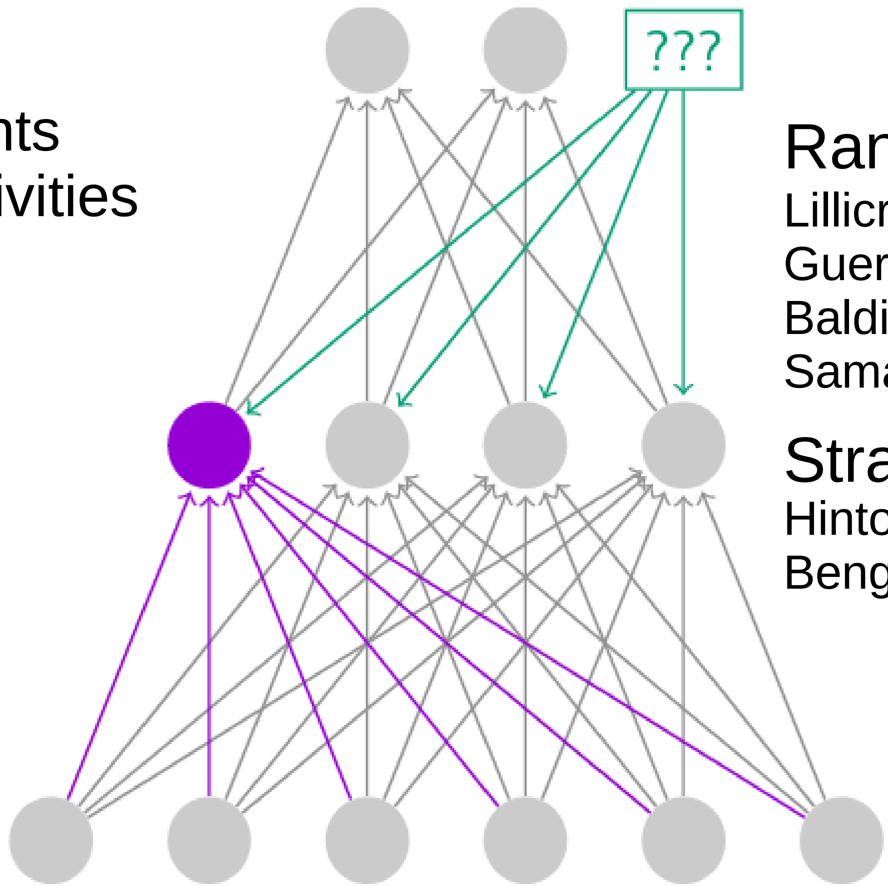
Training many outputs in parallel



Can we take this to multiple layers?

$$\frac{\partial w_{ij}}{\partial t} \equiv \sum_k e_k(t)\epsilon * \left[w_{ki}\epsilon * \left(\epsilon * S_j(t)\sigma'(U_i)\right)\sigma'(U_k)\right]$$

Problems:
- Symmetric weights
- Downstream activities



Random feedback

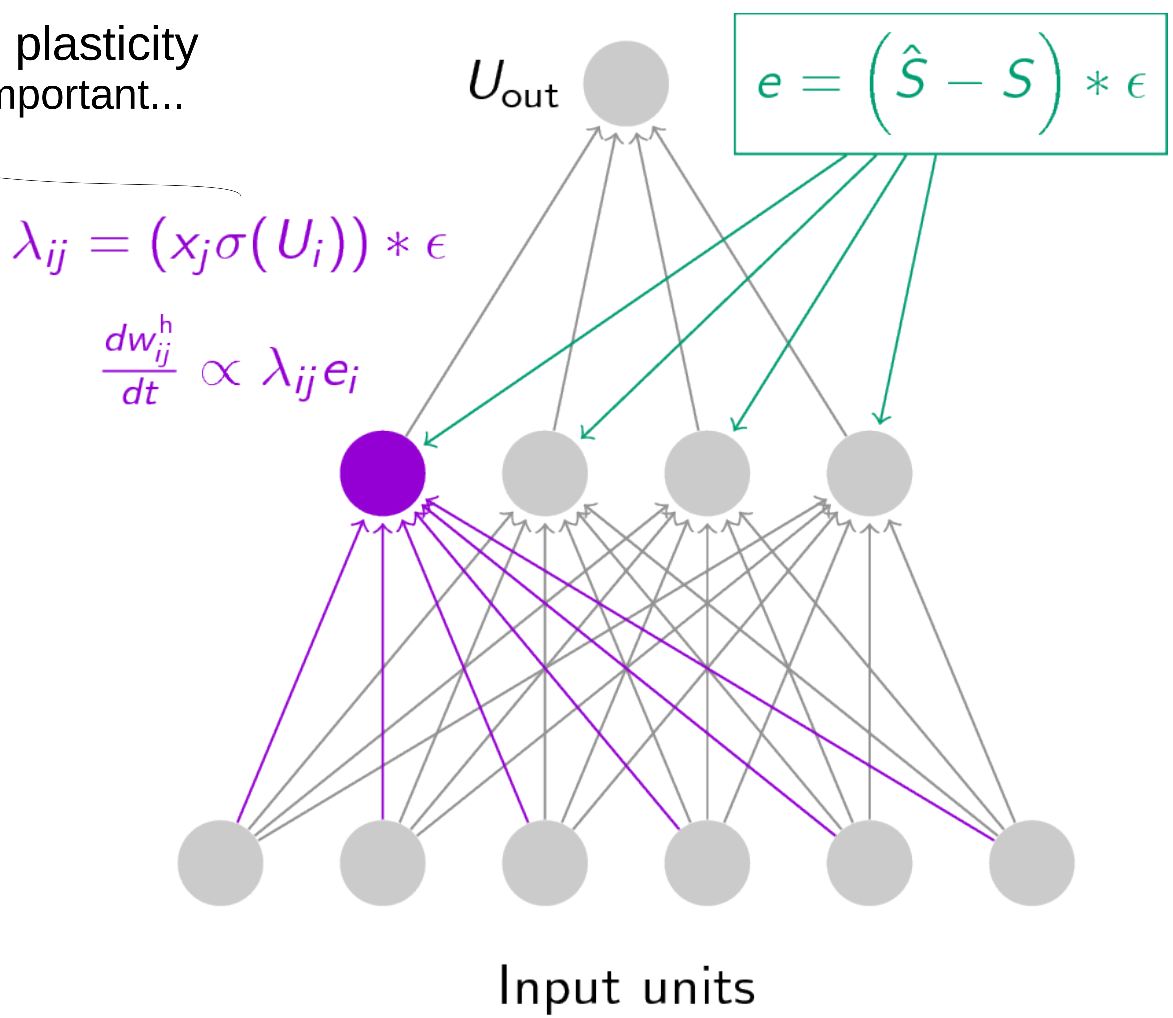
Lillicrap et al. (2014, 2016)
Guergiuev et al. (2016)
Baldi et al. (2016)
Samadi et al. (2017)

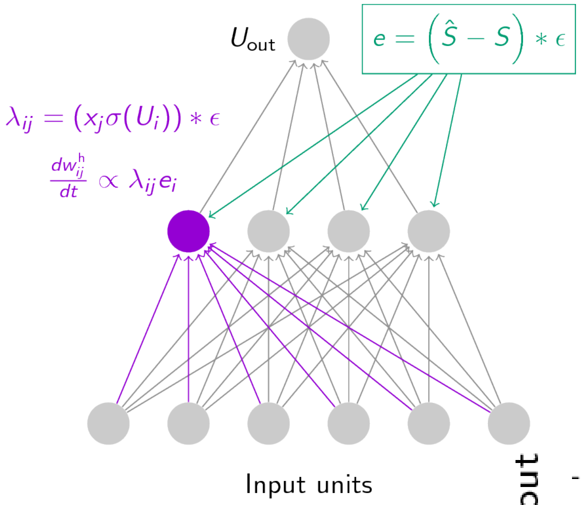
Straight-through estim.

Hinton (2012)
Bengio et al. (2013)

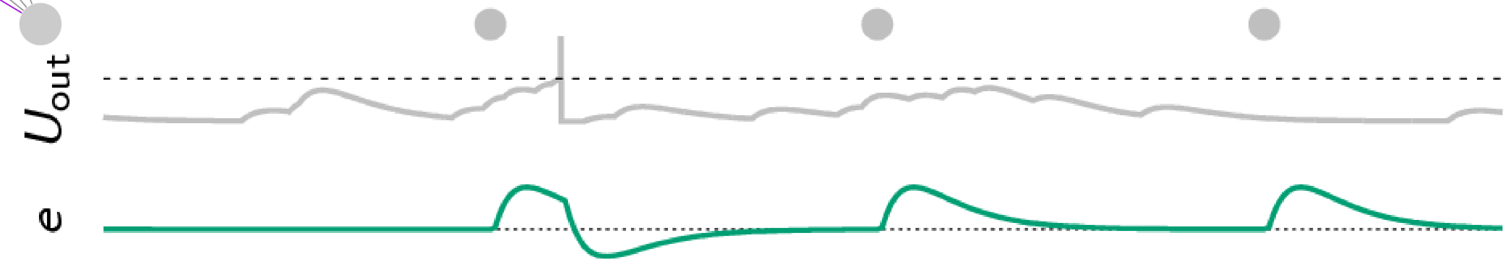
Can we use random feedback to take this to multiple layers?

Synapto-centric form plasticity
Dendrites and spines important...



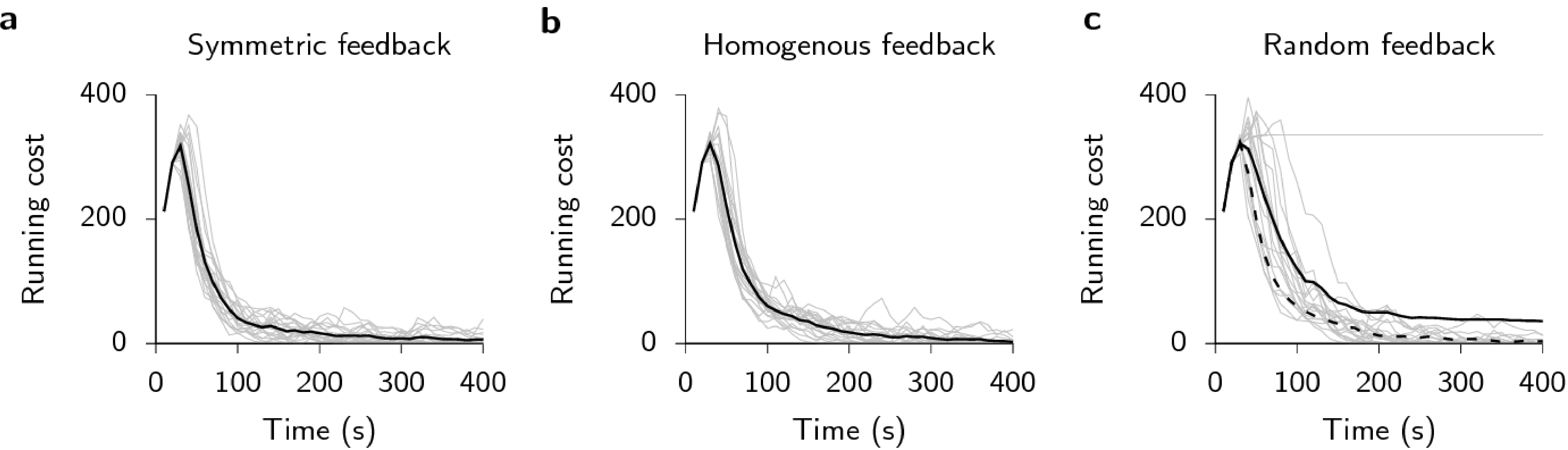


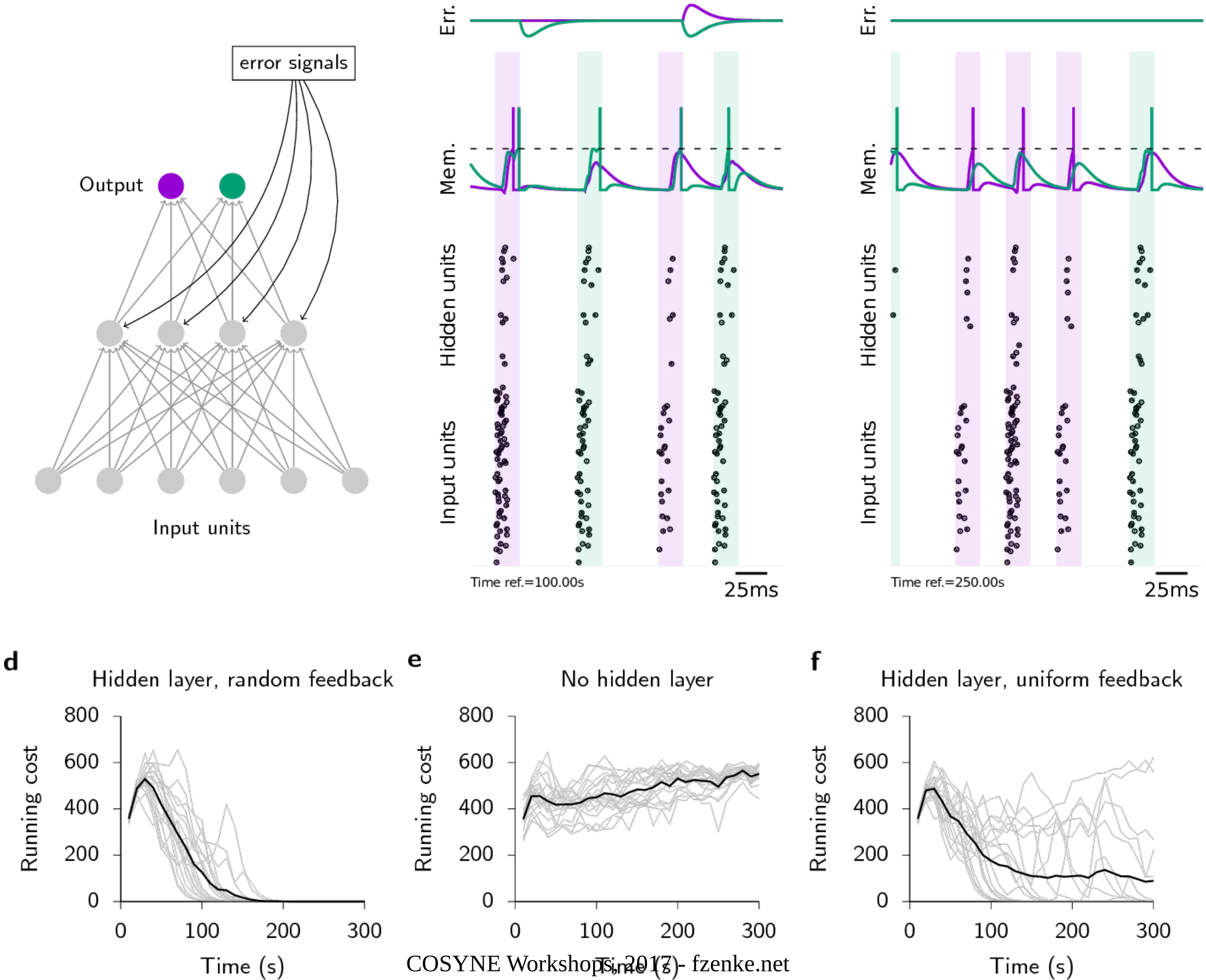
Hidden layer synapse walk-through

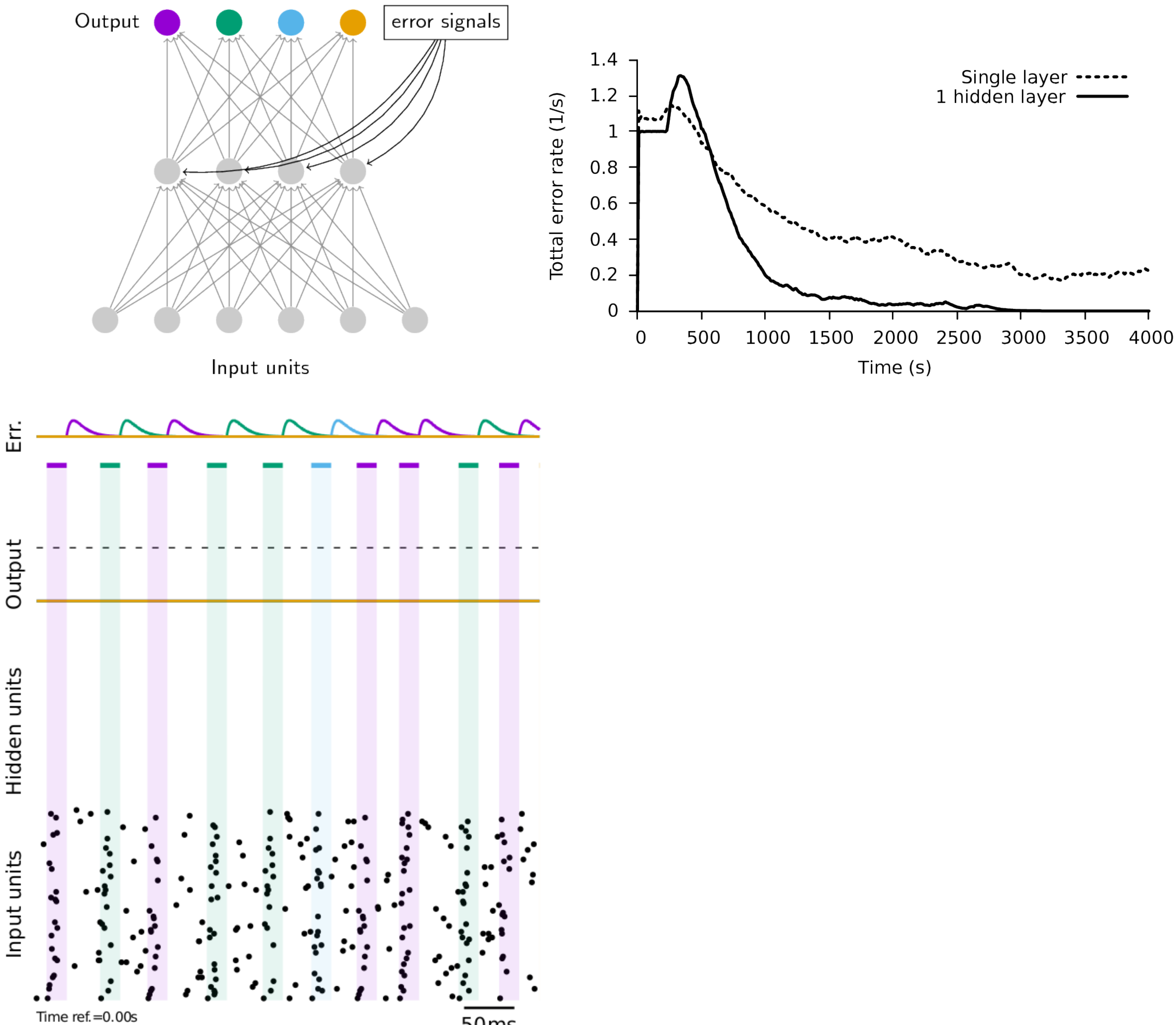


Random feedback, one hidden layer

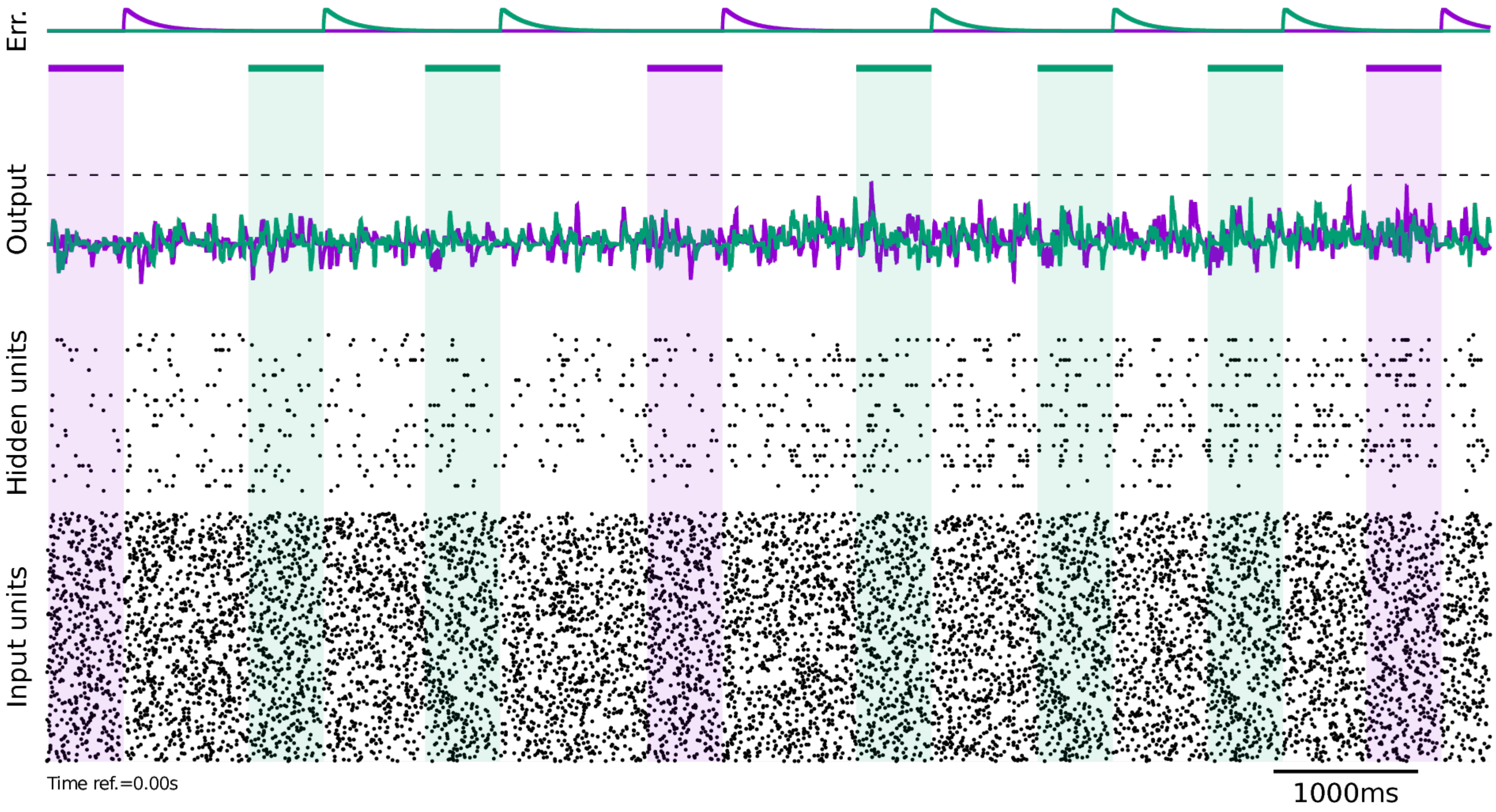
Random feedback does fine



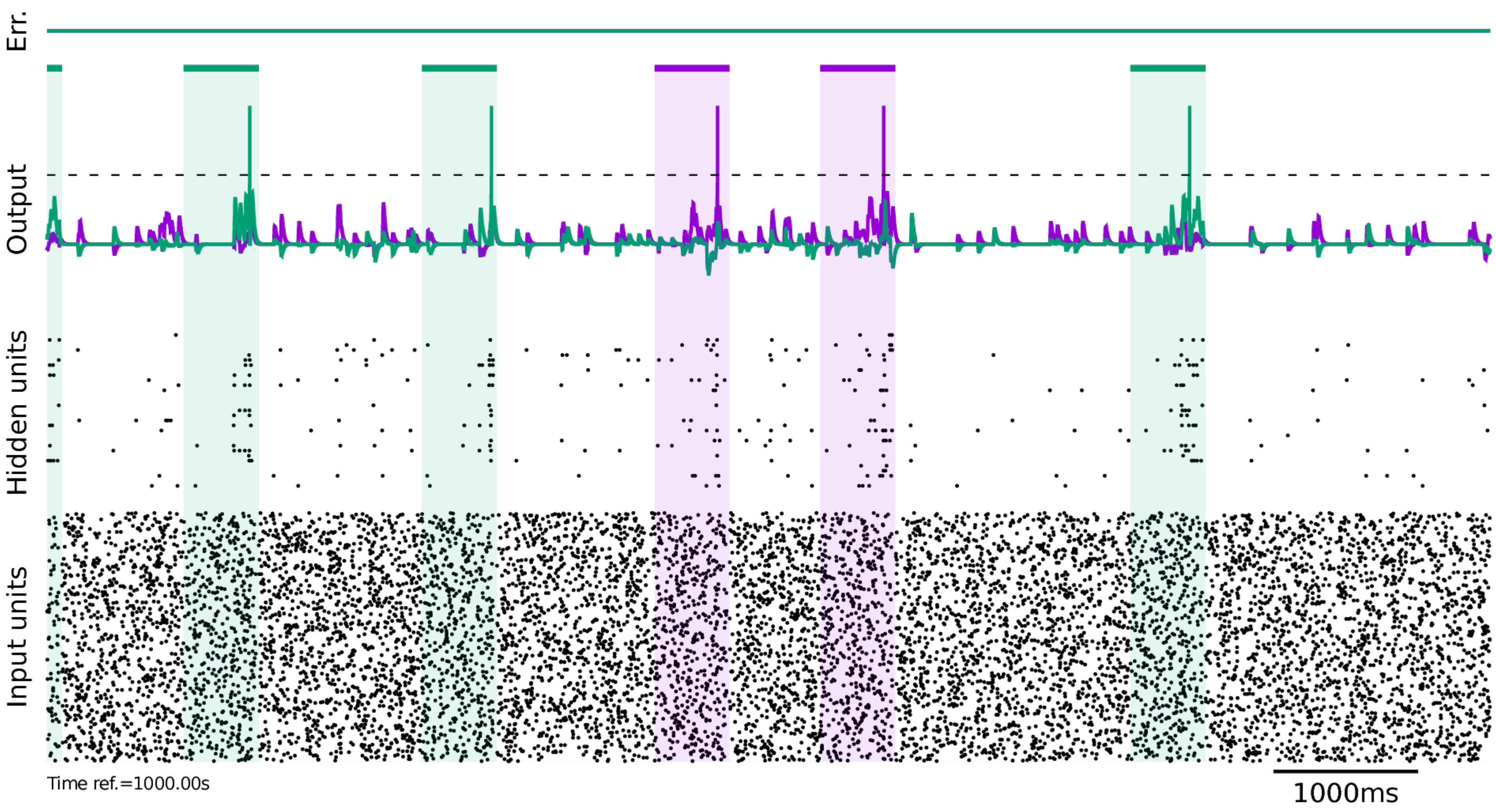

error signals
Output
Input units
Err.
Mem.
Hidden units
Input units
Time ref.=100.00s
25ms
Time ref.=250.00s
25ms
d
Hidden layer, random feedback
e
No hidden layer
f
Hidden layer, uniform feedback
Running cost
Time (s)
0
100
200
300
400
600
800



Output
error signals
Input units
Single layer
1 hidden layer
Tottal error rate (1/s)
Time (s)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
3000
3500
4000
Err.
Output
Hidden units
Input units
Time ref.=0.00s
50ms



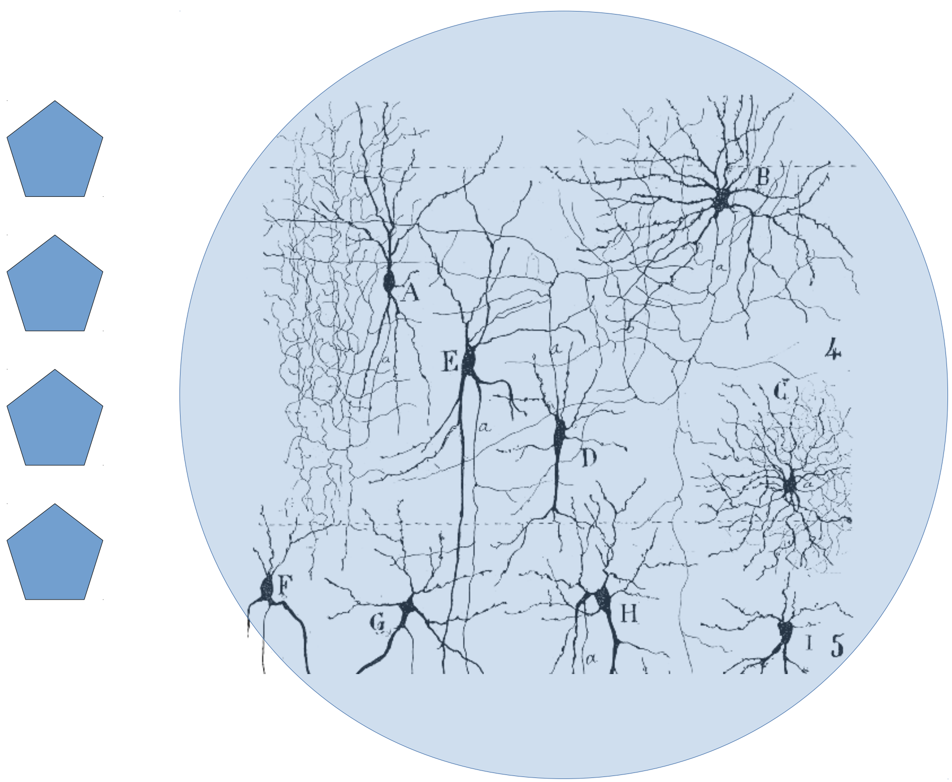
Err.
Output
Hidden units
Input units
Time ref.=0.00s
1000ms


Err.
Output
Hidden units
Input units
Time ref.=1000.00s
1000ms

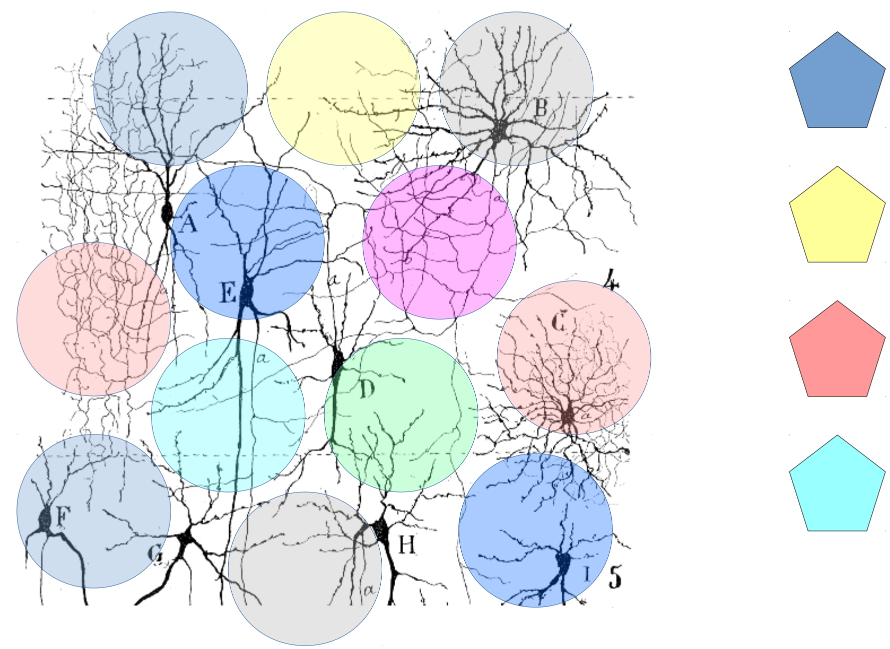
Think of heterogeneity of third factor

$$\frac{\partial w_{ij}}{\partial t} \equiv \sum_k e_k(t)\epsilon * \left[b_{ki}\epsilon * (\epsilon * S_j(t)\sigma'(U_i))\right]$$

Third factor



Global third factor



Neurons by Ramon y Cajal

Summary

- Started from cost function approach
- Method to teach spiking nets to solve non-trivial temporally coded problems
- Learning rule has a simple interpretation in a biological context

Thanks

Surya Ganguli and the gang

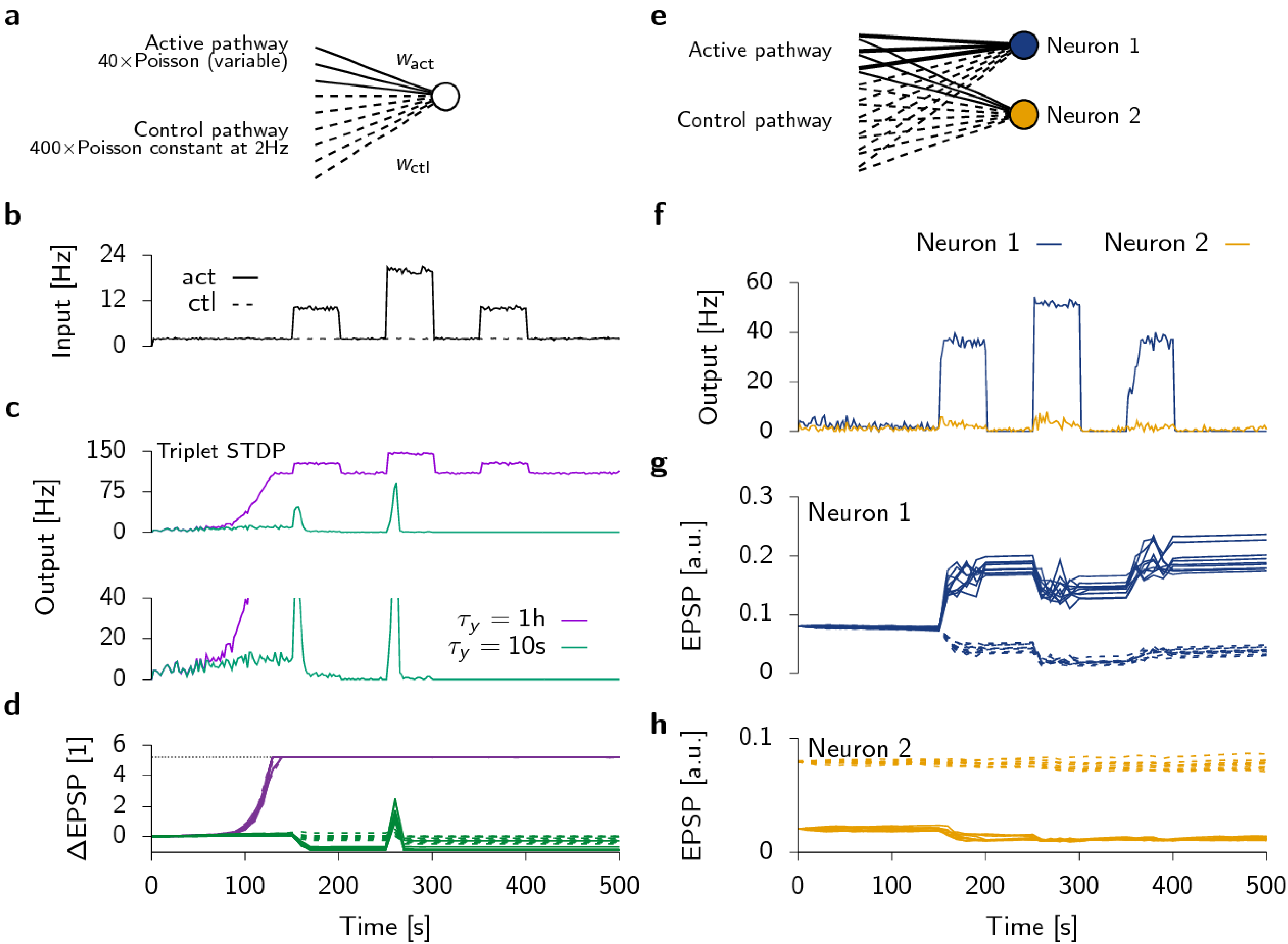
- Kiah Hardcastle
- Sarah Harvey
- Subhy Lahiri
- Niru Maheswaranathan
- Lane McIntosh
- Sam Ocko
- Ben Poole
- Chris Stock
- Alex Williams

Funding:

FNSNF

SWISS NATIONAL SCIENCE FOUNDATION

wellcome



a
Active pathway
40×Poisson (variable)
w_{act}
Control pathway
400×Poisson constant at 2Hz
w_{ctl}
b
Input [Hz]
act
ctl
c
Output [Hz]
Triplet STDP
$\tau_y = 1h$
$\tau_y = 10s$
d
ΔEPSP [1]
Time [s]
e
Active pathway
Control pathway
Neuron 1
Neuron 2
f
Output [Hz]
g
EPSP [a.u.]
Neuron 1
h
EPSP [a.u.]
Neuron 2
Time [s]

Random feedback, one hidden layer