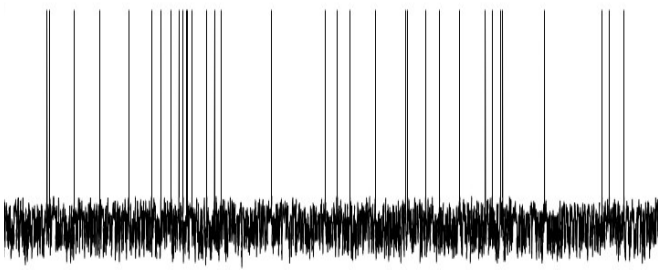
Credit assignment in space and time

Training spiking neural networks with surrogate gradients

Friedemann Zenke
https://fzenke.net

EPFL Neuro Symposium 2019 “Neuroscience meets deep learning”
Organized by the Brain Mind Institute at the School of Life Sciences.

FMI
Friedrich Miescher Institute
for Biomedical Research



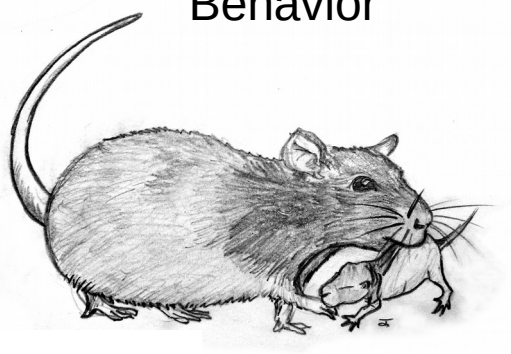
wellcome

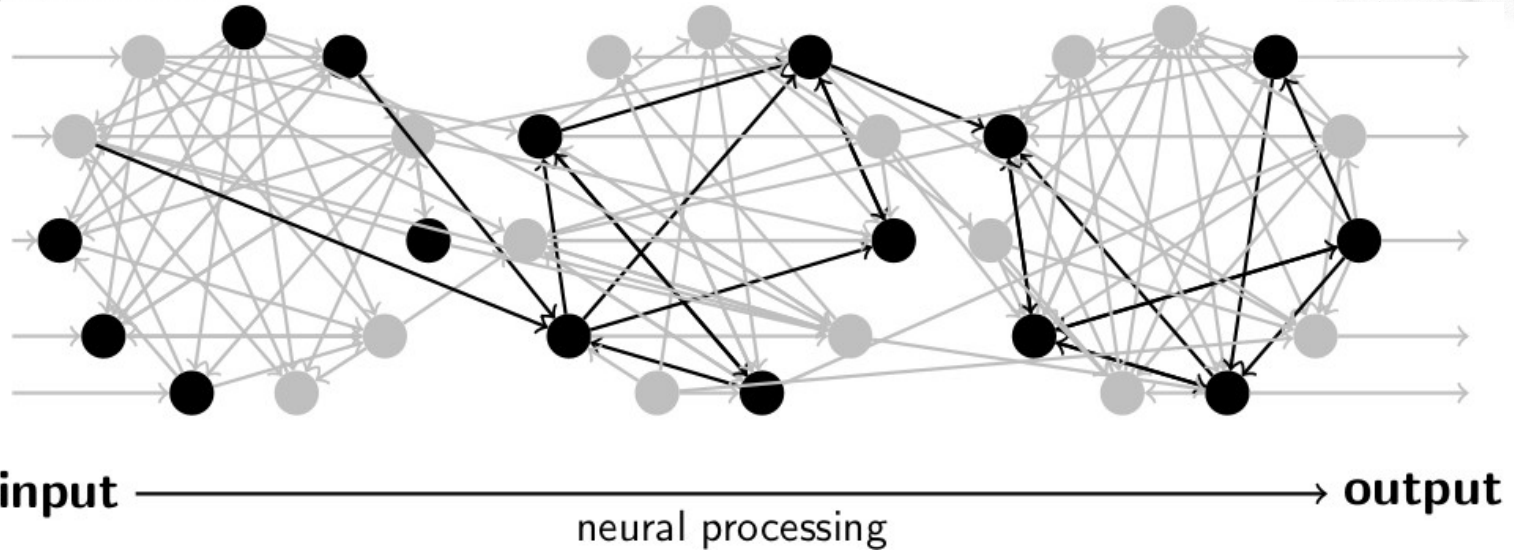
UNIVERSITY OF OXFORD

Animals process information using neural networks

Sensory inputs

Behavior



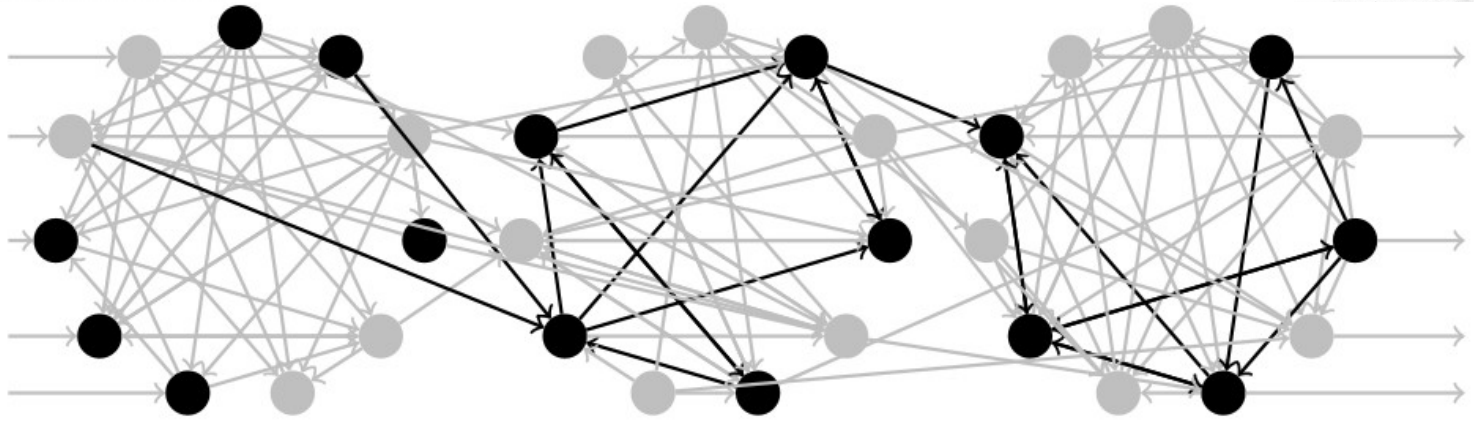


Animals process information using neural networks

Sensory inputs

Function

Behavior

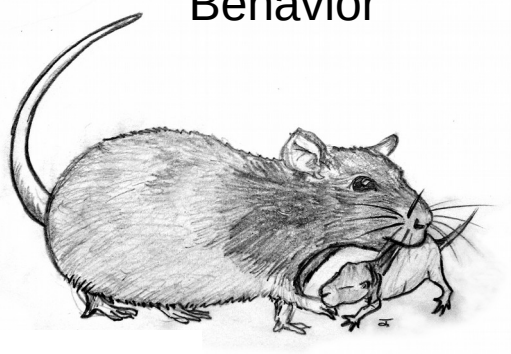


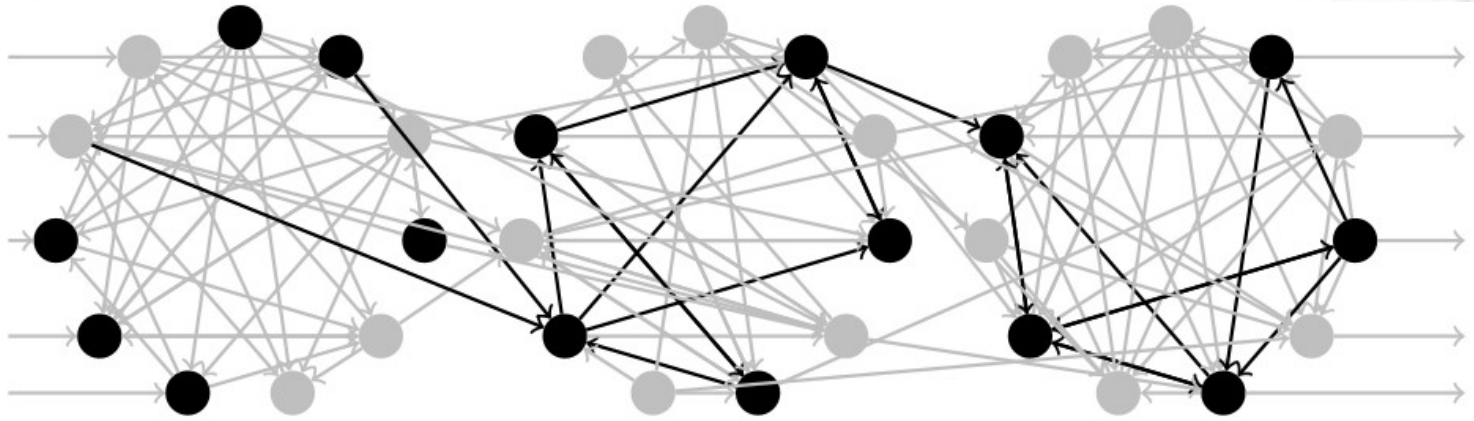
Animals process information using neural networks

Sensory inputs

Function

Behavior





Hidden units

input → output

neural processing

Animals process information using neural networks

Sensory inputs

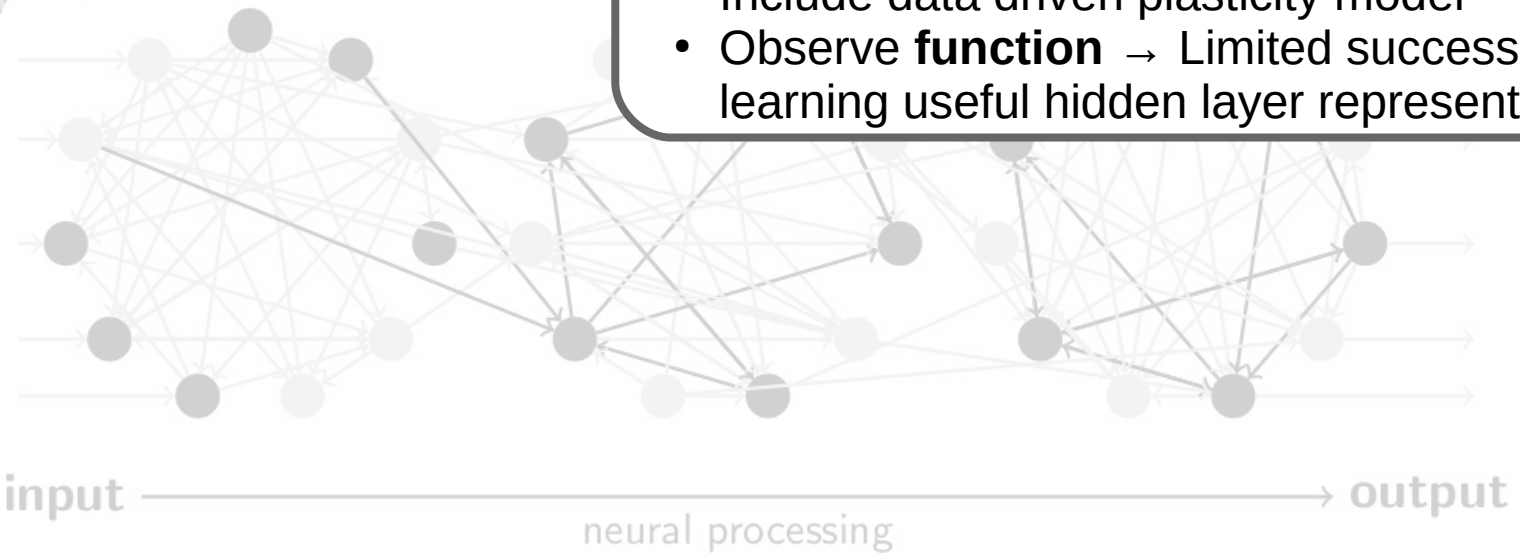
Key question: How do hidden units learn?

Animals process information using neural networks

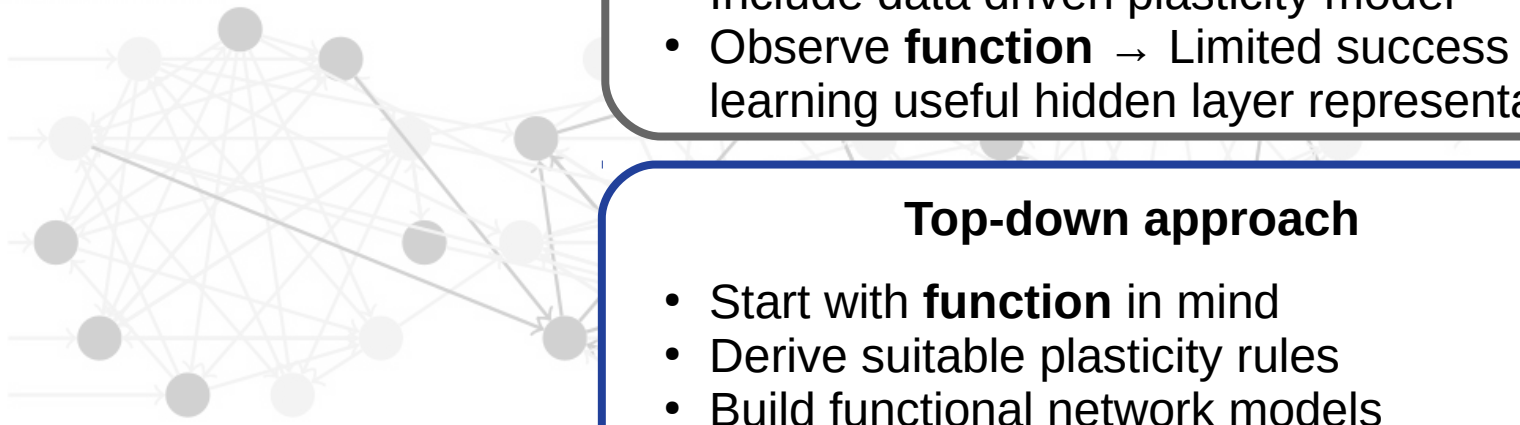
Key question: How do hidden units learn?

Bottom-up approach

- Start with a random network model
- Include data driven plasticity model
- Observe **function** → Limited success in learning useful hidden layer representations



Animals process information using neural networks



input → output

neural processing

Key question: How do hidden units learn?

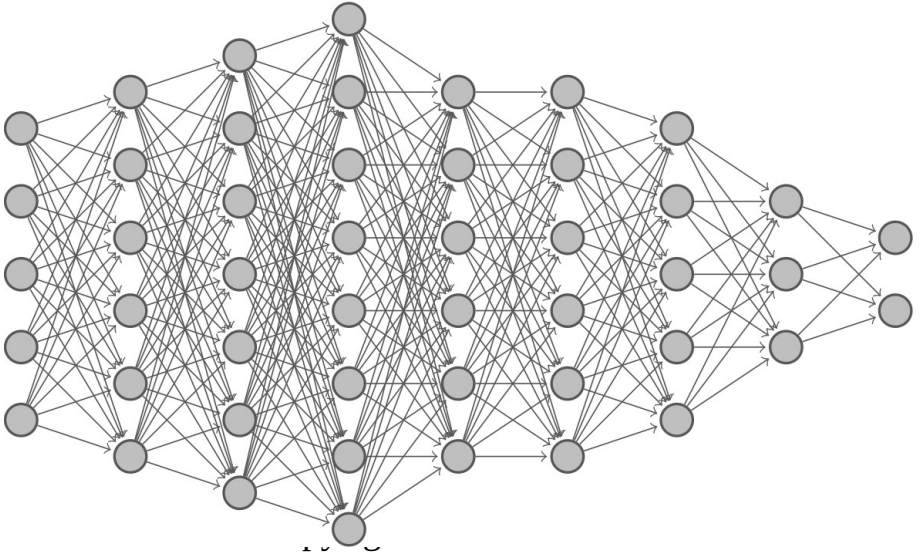
Bottom-up approach

- Start with a random network model
- Include data driven plasticity model
- Observe **function** → Limited success in learning useful hidden layer representations

Top-down approach

- Start with **function** in mind
- Derive suitable plasticity rules
- Build functional network models

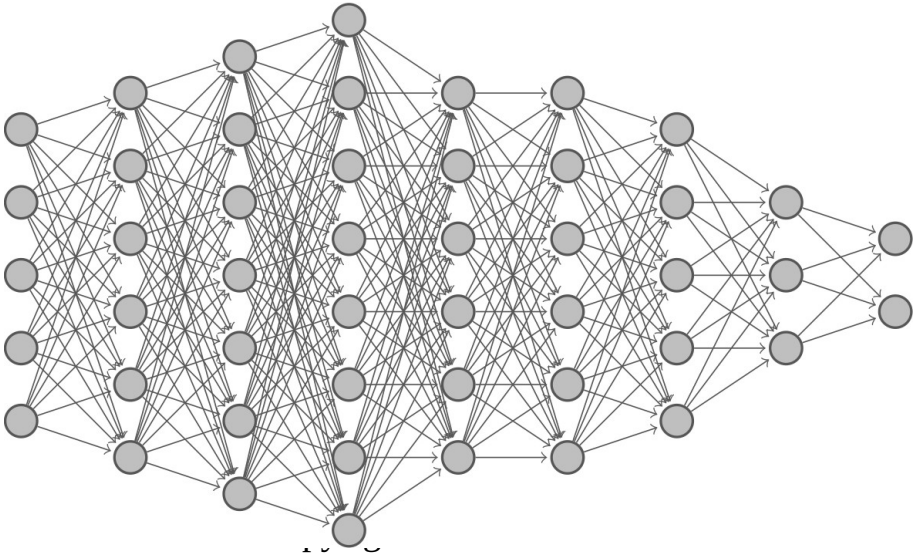
Deep learning provides a useful framework



Deep learning provides a useful framework

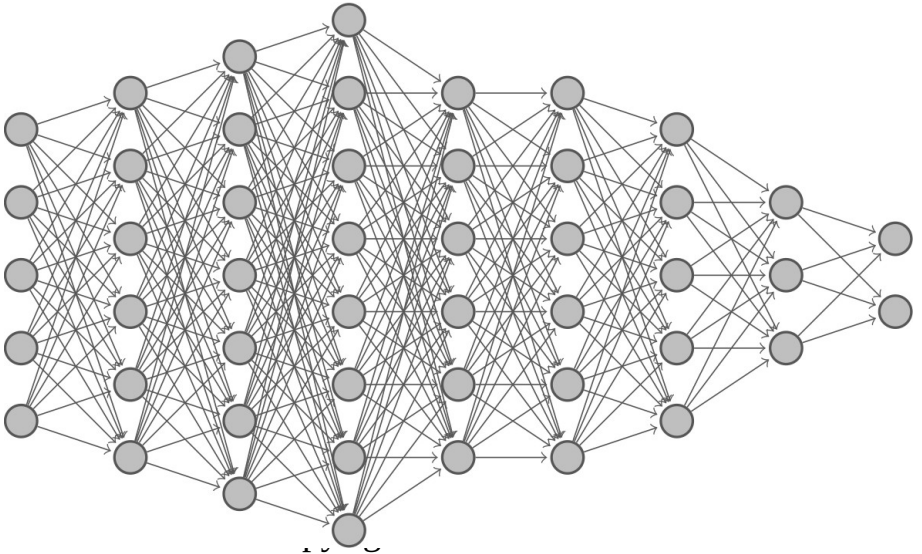
1) Input data

X



Deep learning provides a useful framework

1) Input data X



$\mathcal{L}$ **2) Loss function**

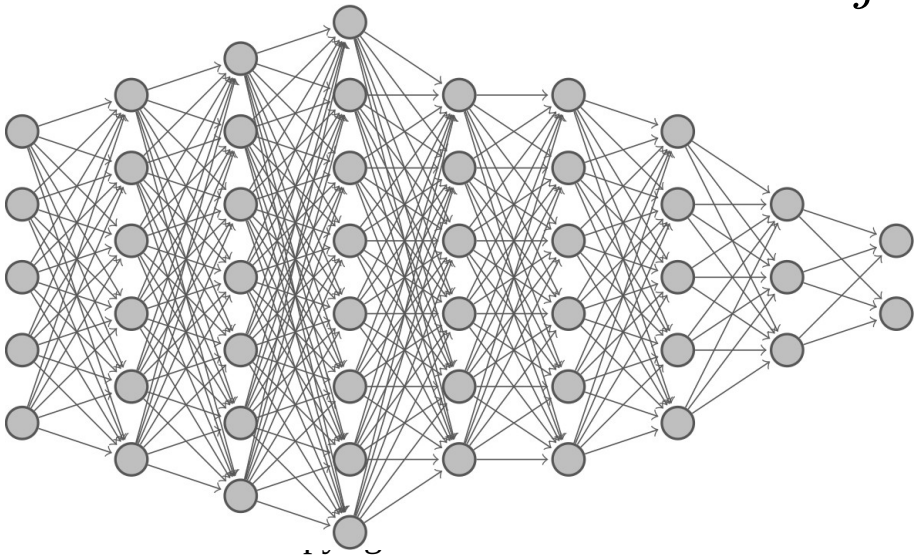
Deep learning provides a useful framework

3) Adjust weights
Gradient descent

$$\Delta W_{ij} \propto -\frac{\partial \mathcal{L}}{\partial W_{ij}}$$

1) Input data

X



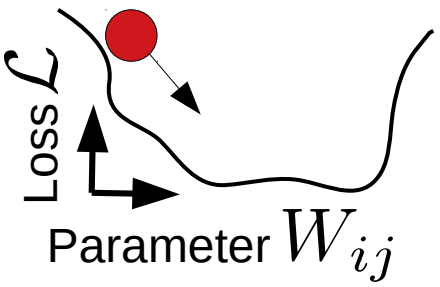
$\mathcal{L}$

2) Loss function

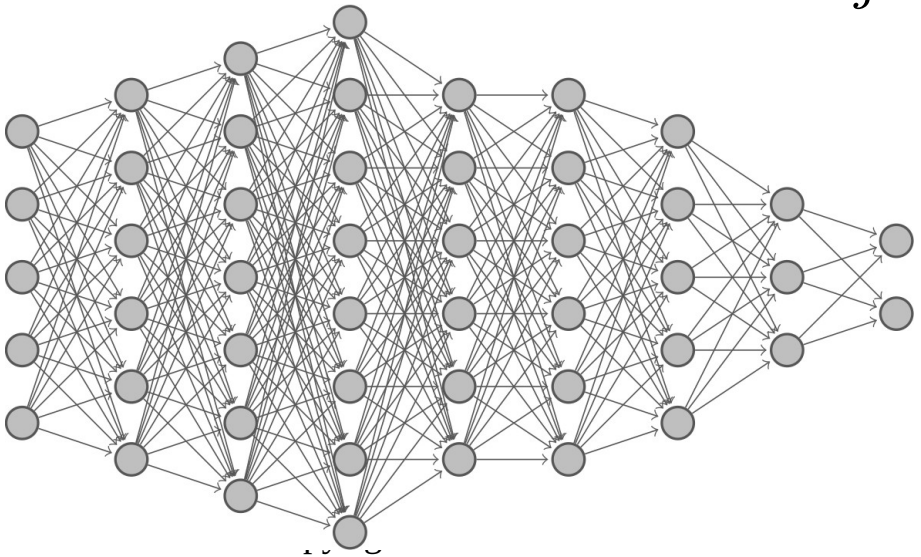
Deep learning provides a useful framework

3) Adjust weights
Gradient descent

$$\Delta W_{ij} \propto -\frac{\partial \mathcal{L}}{\partial W_{ij}}$$



1) Input data X

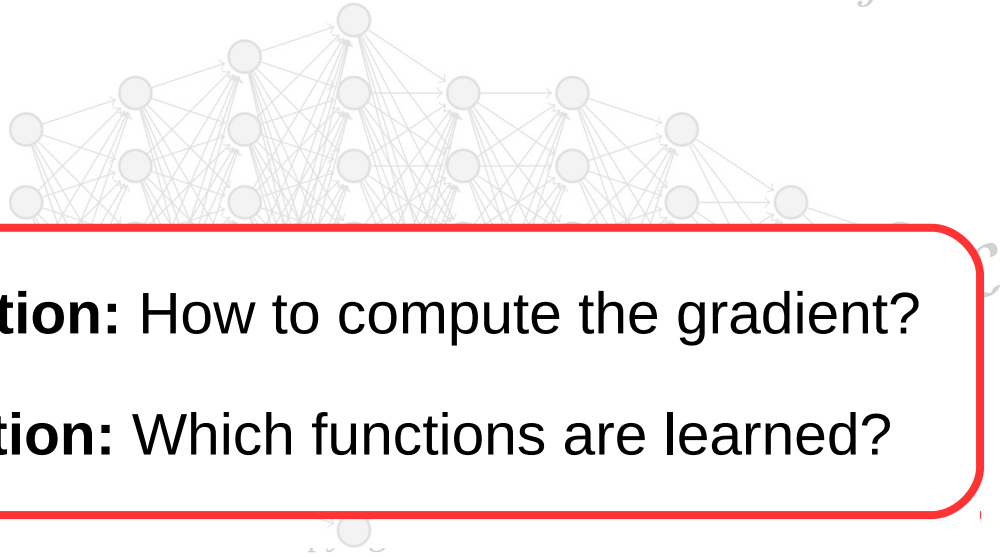


$\mathcal{L}$ **2) Loss function**

Deep neural networks implement functions
They ”learn”, but they don’t spike

3) Adjust weights
Gradient descent

$$\Delta W_{ij} \propto -\frac{\partial \mathcal{L}}{\partial W_{ij}}$$



2) Loss function

Algorithmic question: How to compute the gradient?

Conceptual question: Which functions are learned?

The recent excitement about neural networks

Francis Crick

The remarkable properties of some recent computer algorithms for neural networks seemed to promise a fresh approach to understanding the computational properties of the brain. Unfortunately most of these neural nets are unrealistic in important respects.

1989

The recent excitement about neural networks

Francis Crick

The remarkable properties of some recent computer algorithms for neural networks seemed to promise a fresh approach to understanding the computational properties of the brain. Unfortunately most of these neural nets are unrealistic in important respects.

1989

The recent excitement about neural networks

Francis Crick

The remarkable properties of some recent computer algorithms for neural networks seemed to promise a fresh approach to understanding the computational properties of the brain. Unfortunately most of these neural nets are unrealistic in important respects.

NATURE VOL. 337 12 JANUARY 1989

COMMENTARY

1989

The recent excitement about neural networks

Francis Crick

The remarkable properties of some recent computer algorithms for neural networks seemed to promise a fresh approach to understanding the computational properties of the brain. Unfortunately most of these neural nets are unrealistic in important respects.

“Unrealistic in important respects”

- Non-locality of learning rules
  (a.k.a. the weight transport problem)

NATURE VOL. 337 12 JANUARY 1989

COMMENTARY

1989

The recent excitement about neural networks

Francis Crick

The remarkable properties of some recent computer algorithms for neural networks seemed to promise a fresh approach to understanding the computational properties of the brain. Unfortunately most of these neural nets are unrealistic in important respects.

“Unrealistic in important respects”

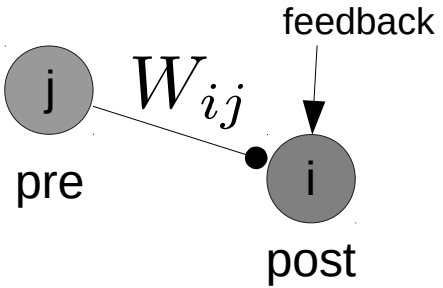
- Non-locality of learning rules
  (a.k.a. the weight transport problem)
- Graded activation functions vs spikes

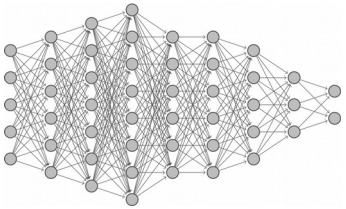
The *more recent* excitement about (deep) neural networks

“Unrealistic in important respects”

- Non-locality of learning rules
  (a.k.a. the weight transport problem)
- Graded activation functions vs spikes

The *more recent* excitement about (deep) neural networks



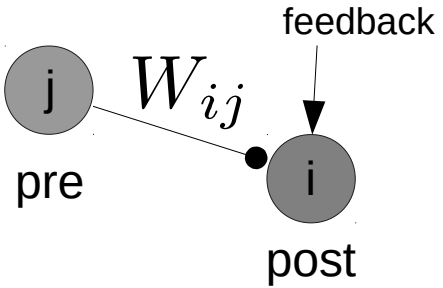


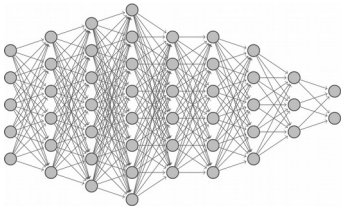
$$\Delta W_{ij} \propto (\mathrm{pre}_j) f(\mathrm{post}_i)(\mathrm{feedback}_i)$$

“Unrealistic in important respects”

- Non-locality of learning rules
  (a.k.a. the weight transport problem)
- Graded activation functions vs spikes

The *more recent* excitement about (deep) neural networks



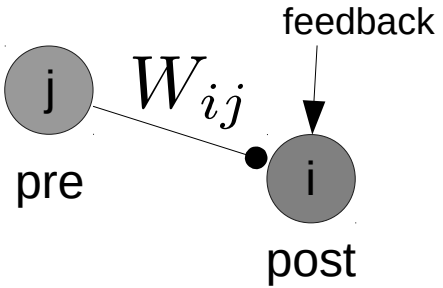


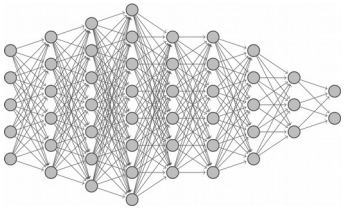
$$\Delta W_{ij} \propto (\mathrm{pre}_j) f(\mathrm{post}_i) (\mathrm{feedback}_i)$$

“Unrealistic in important respects”

- Non-locality of learning rules (a.k.a. the weight transport problem)
- Graded activation functions vs spikes

The *more recent* excitement about (deep) neural networks



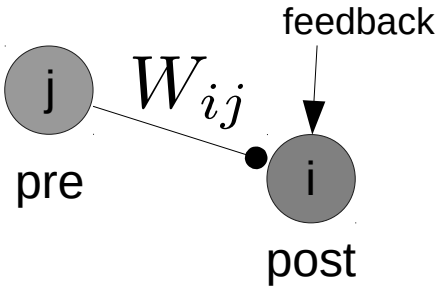


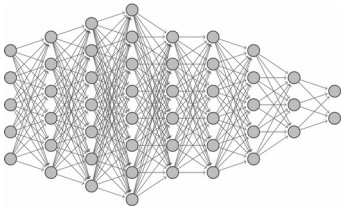
$$\Delta W_{ij} \propto (\mathrm{pre}_j) f(\mathrm{post}_i) (\mathrm{feedback}_i)$$

“Unrealistic in important respects”

- Non-locality of learning rules (a.k.a. the weight transport problem)
- Graded activation functions vs spikes

The *more recent* excitement about (deep) neural networks





$$\Delta W_{ij} \propto (\mathrm{pre}_j) f(\mathrm{post}_i) (\mathrm{feedback}_i)$$

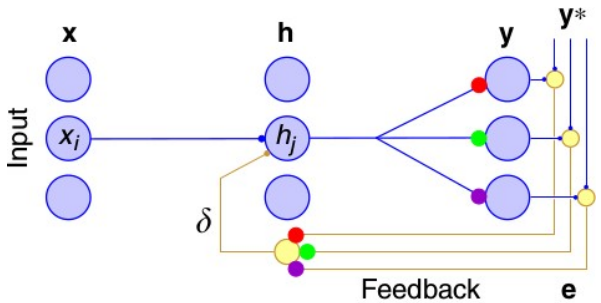
“Unrealistic in important respects”

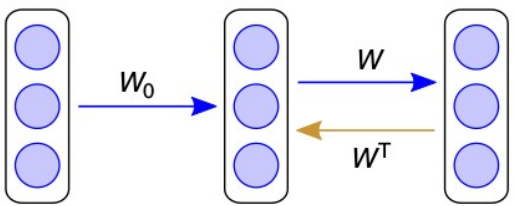
- Non-locality of learning rules (a.k.a. the weight transport problem)
- Graded activation functions vs spikes

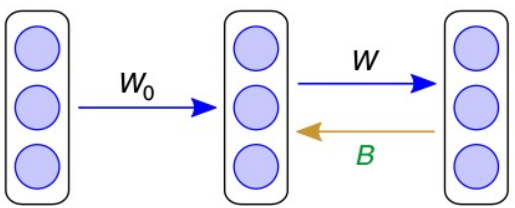
Plausible vector-valued feedback!

- Lillicrap et al. (2016)
- Nøkland (2016)
- Guerguiev et al. (2017)
- Scellier & Bengio (2017)
- Whittington & Bogacz (2017)
- Sacramento et al. (2018)
- Pozzi et al. (2018)

Random feedback / feedback alignment

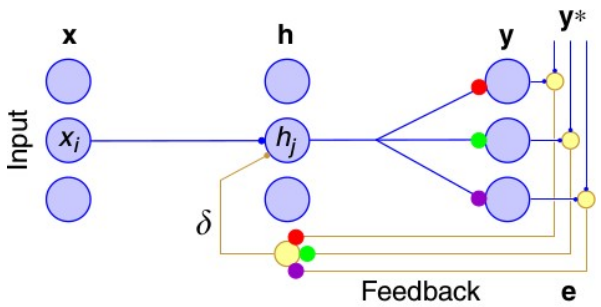


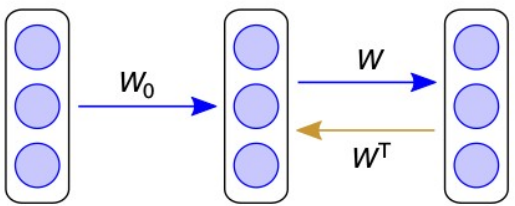


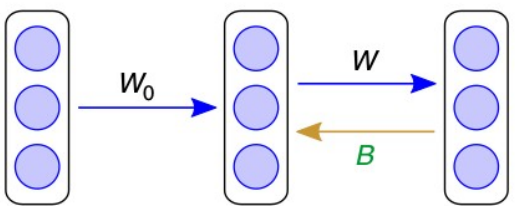


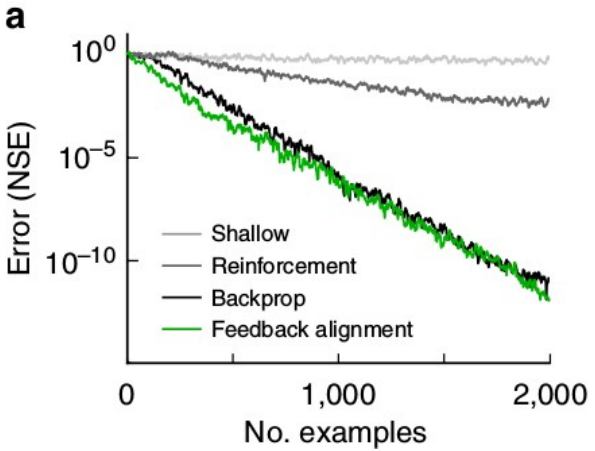
Lillicrap, T.P., Cownden, D., Tweed, D.B., and Akerman, C.J. (2016)

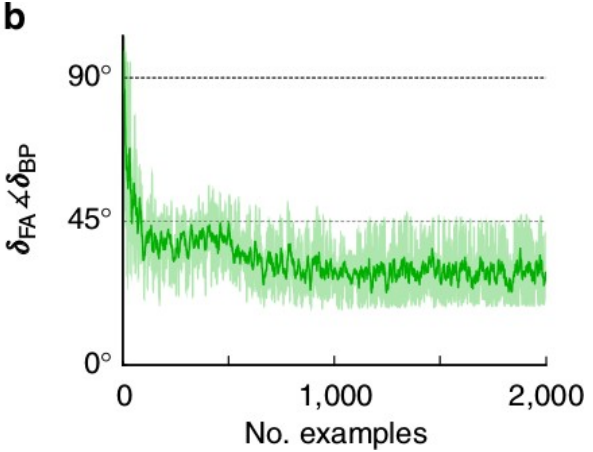
Random feedback / feedback alignment





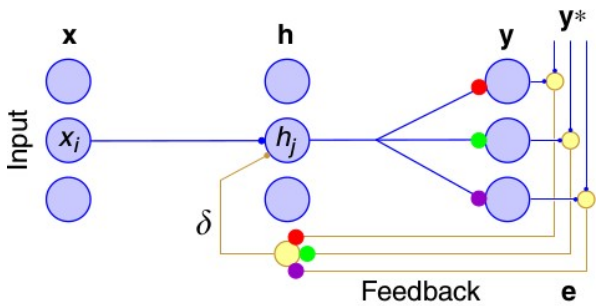


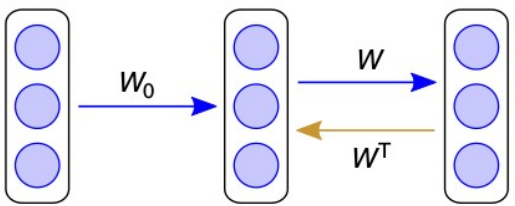


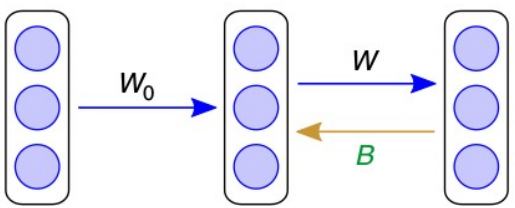


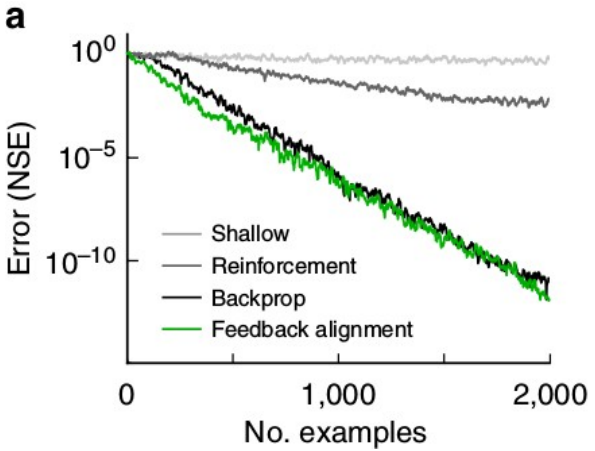
Lillicrap, T.P., Cownden, D., Tweed, D.B., and Akerman, C.J. (2016)

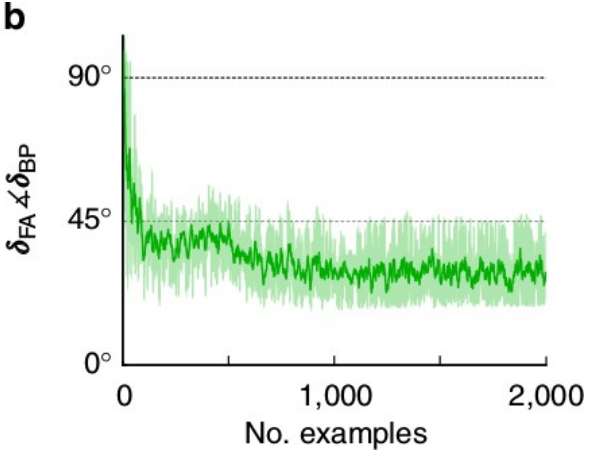
Random feedback / feedback alignment

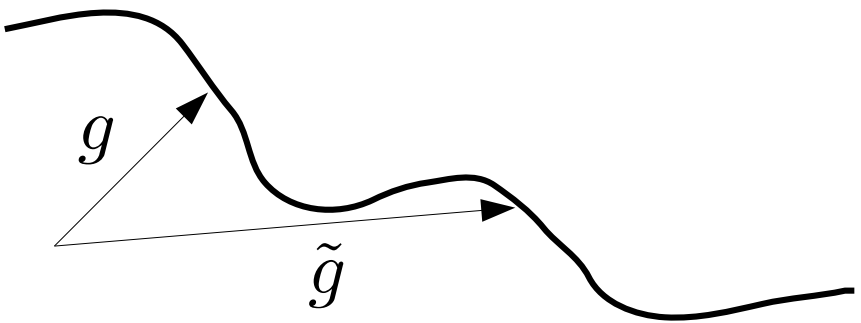






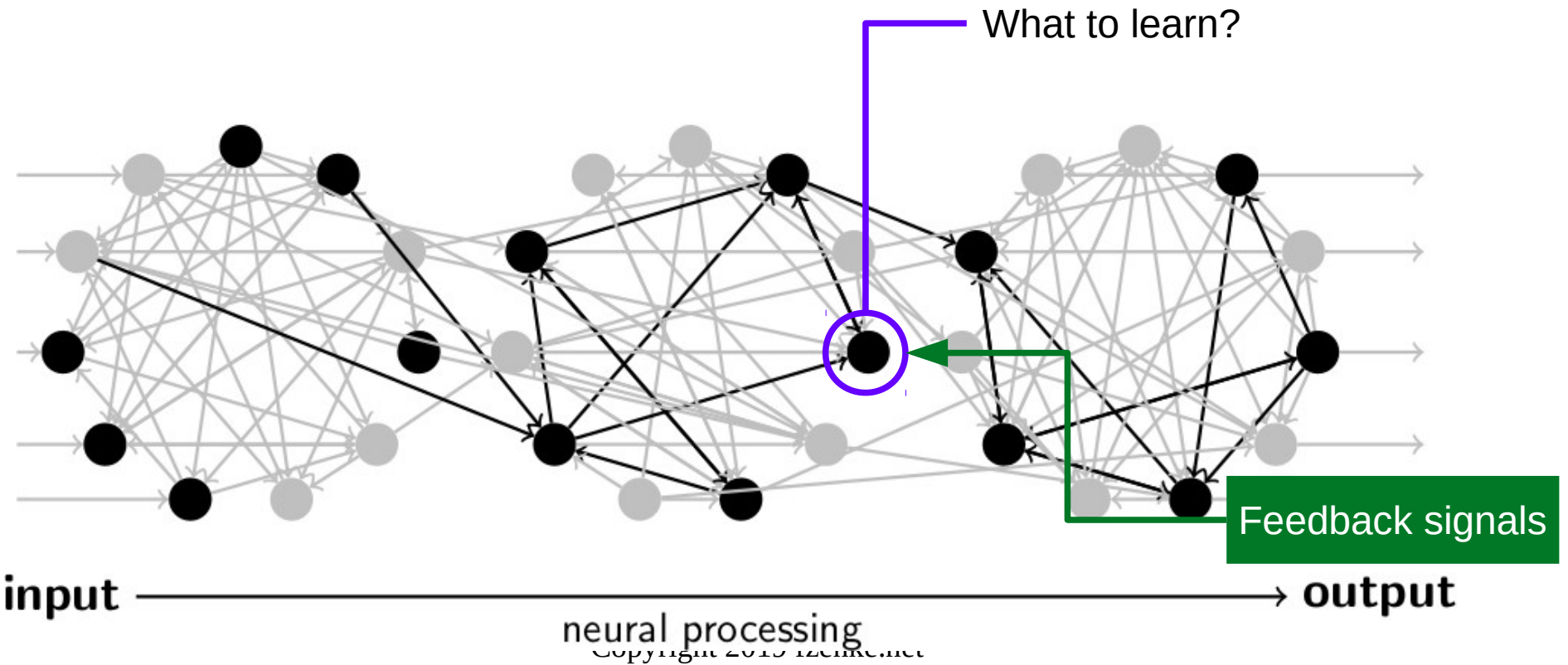




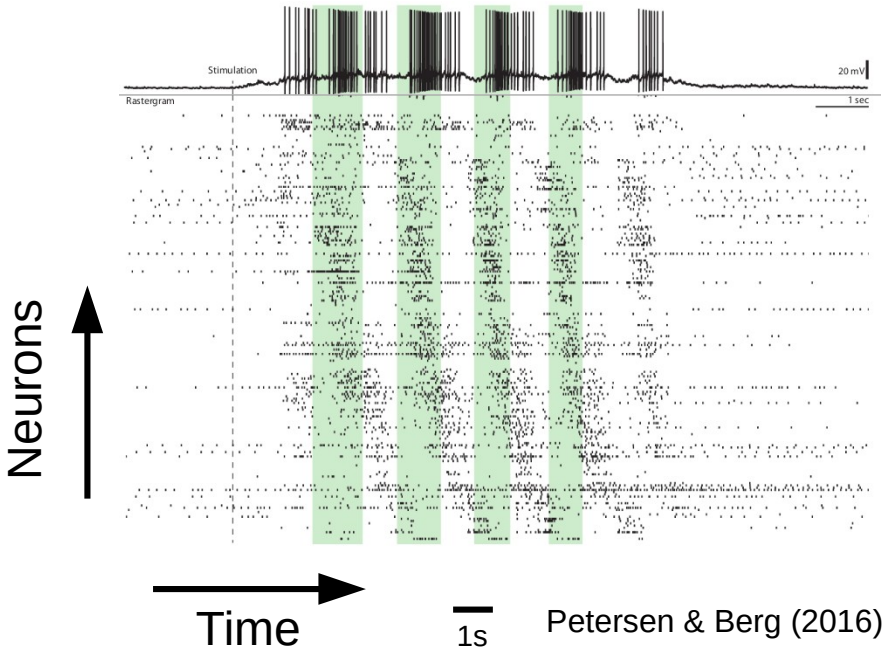


Lillicrap, T.P., Cownden, D., Tweed, D.B., and Akerman, C.J. (2016)

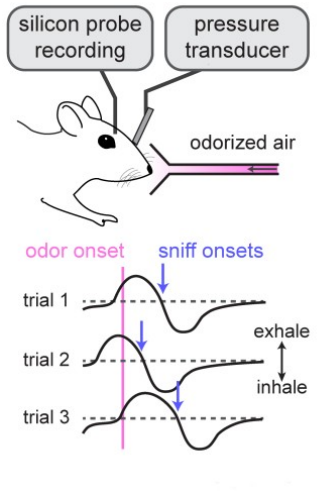
Spatial credit assignment

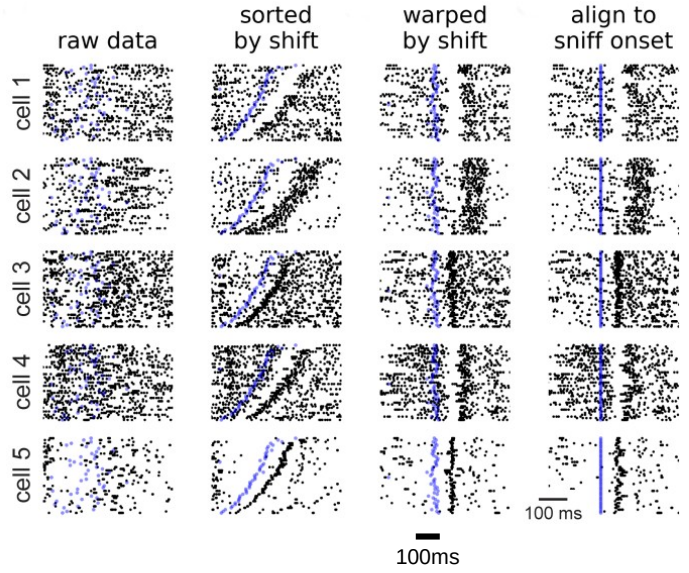


Neural networks use spikes to process temporal information



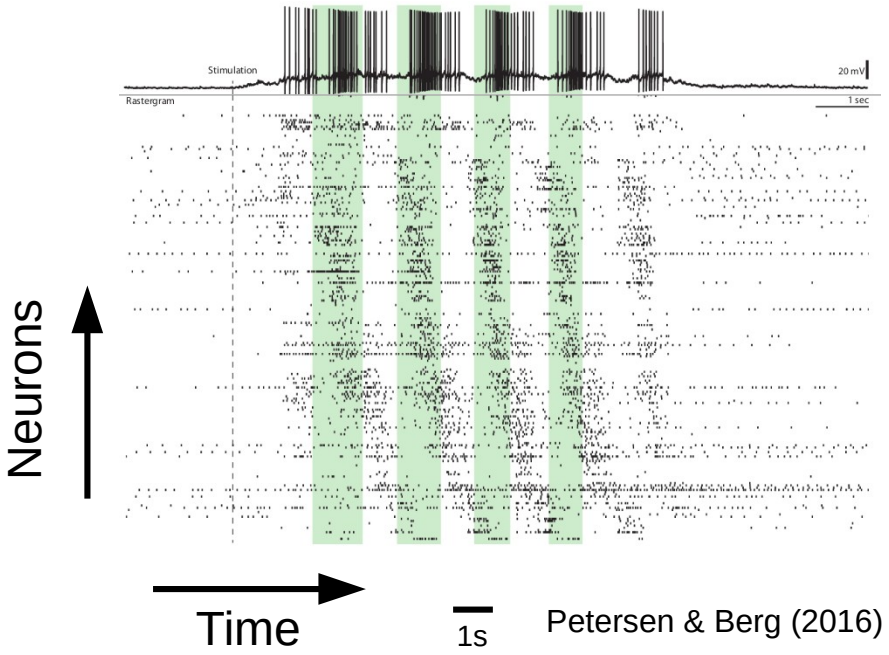
Petersen & Berg (2016)



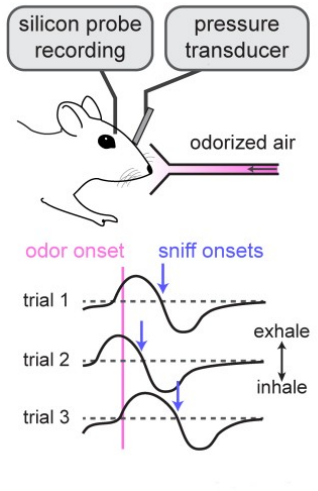


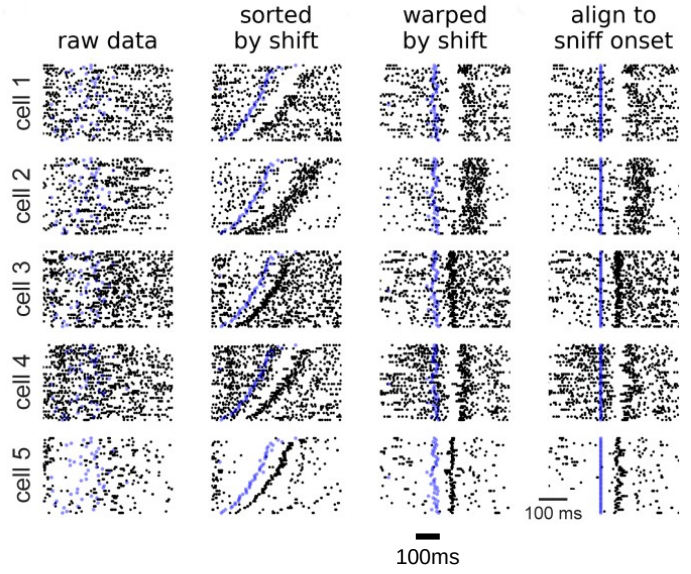
Williams et al. (2019)

Neural networks use spikes to process temporal information



Petersen & Berg (2016)





Williams et al. (2019)

$$\Delta W_{ij} \propto (\mathrm{pre}_j(t)) f(\mathrm{post}_i(t)) (\mathrm{feedback}_i(t))$$

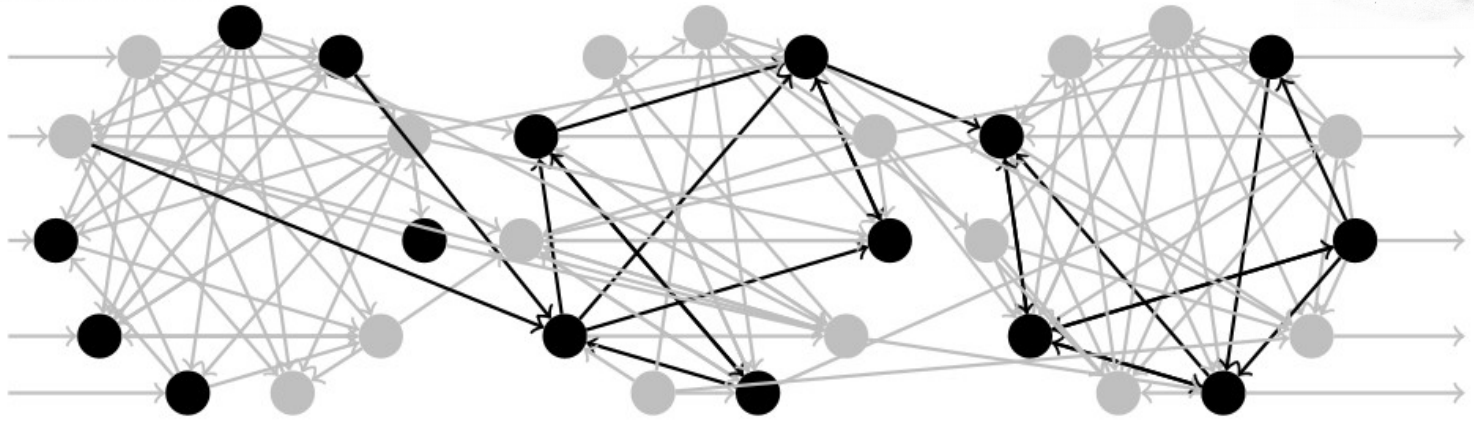
Outline

- **Aim:** Solve temporal tasks with spiking networks
- **Problems:**
  - Spikes → ill defined derivative
  - Temporal credit assignment
- **Solution:** Surrogate gradients & Eligibility traces
- **A look at:** Plausibility, robustness, performance

Towards functional neural network models

Sensory inputs

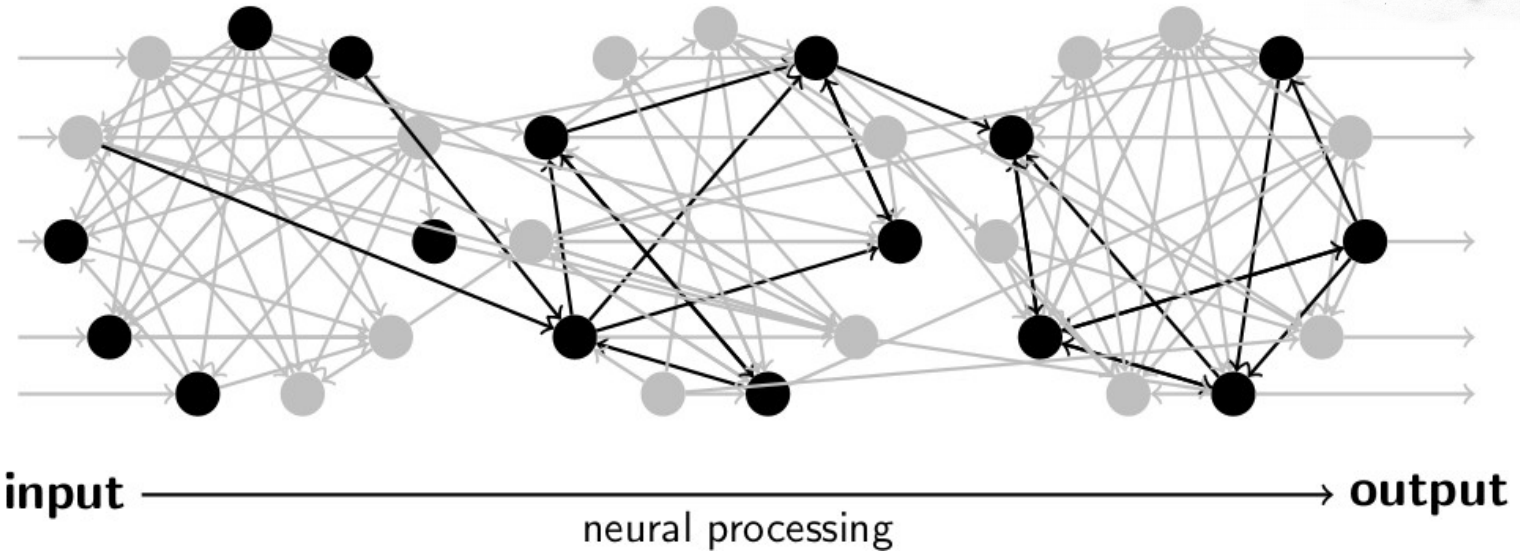
Behavior



Towards functional neural network models

Behavior

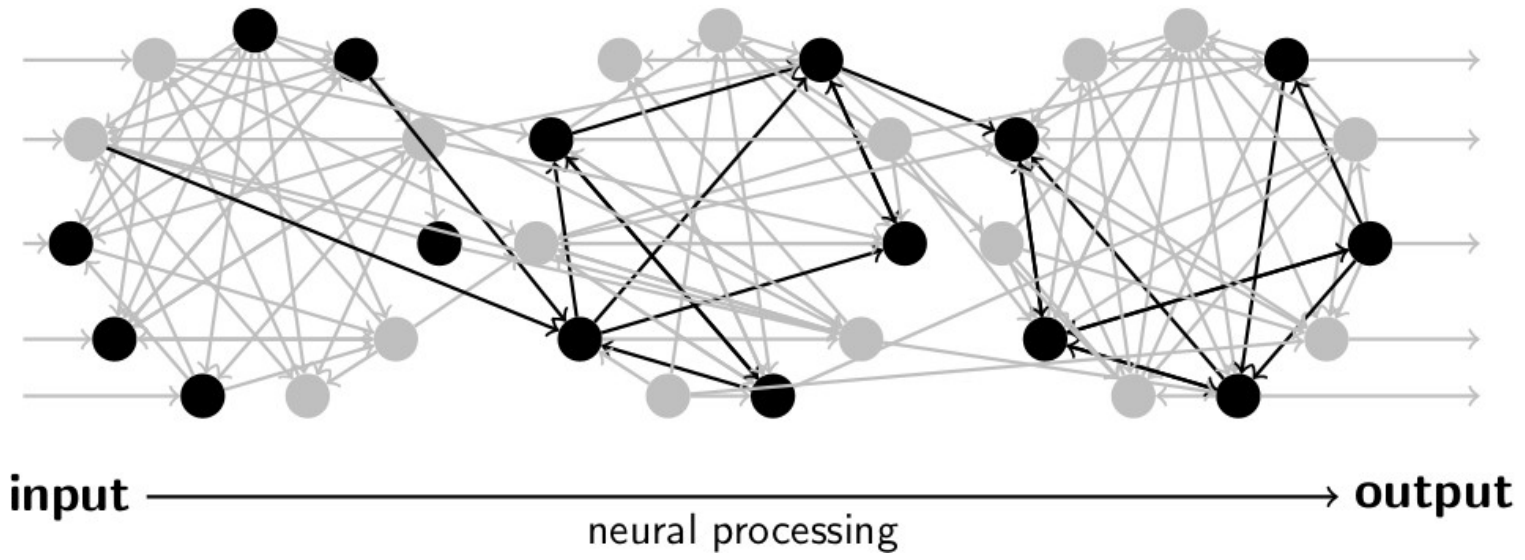
1) Input



Towards functional neural network models

1) Input

2) Output

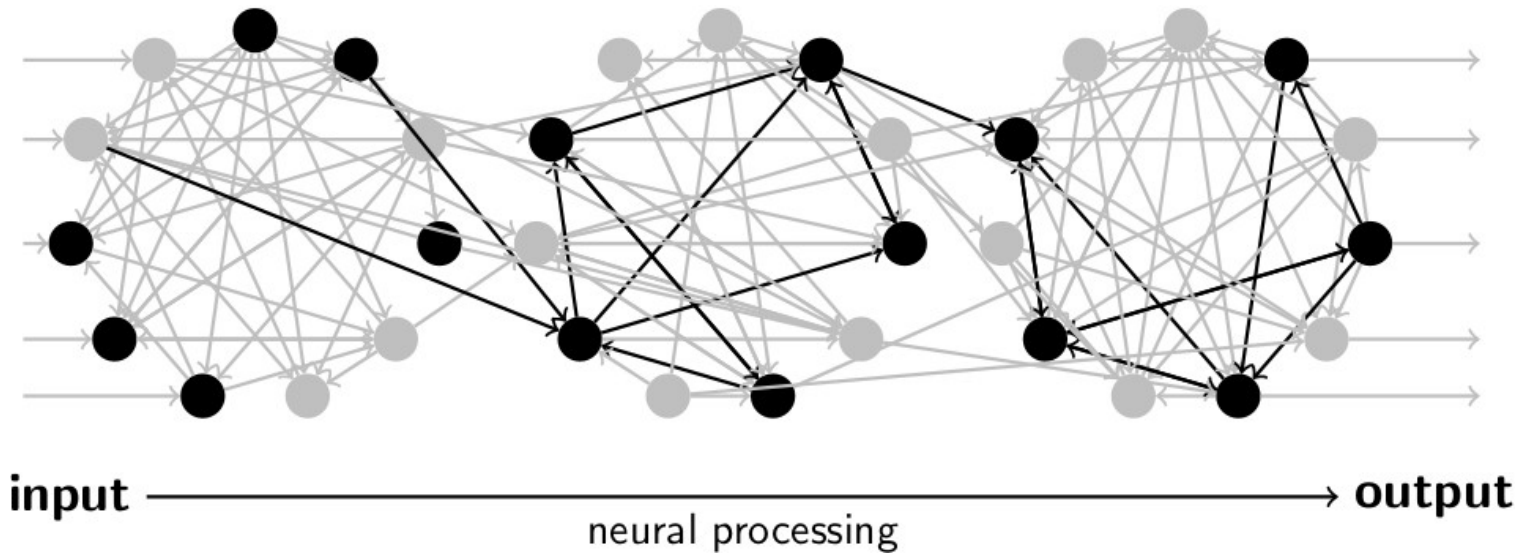


Towards functional neural network models

1) Input

3) Adjust weights

2) Output

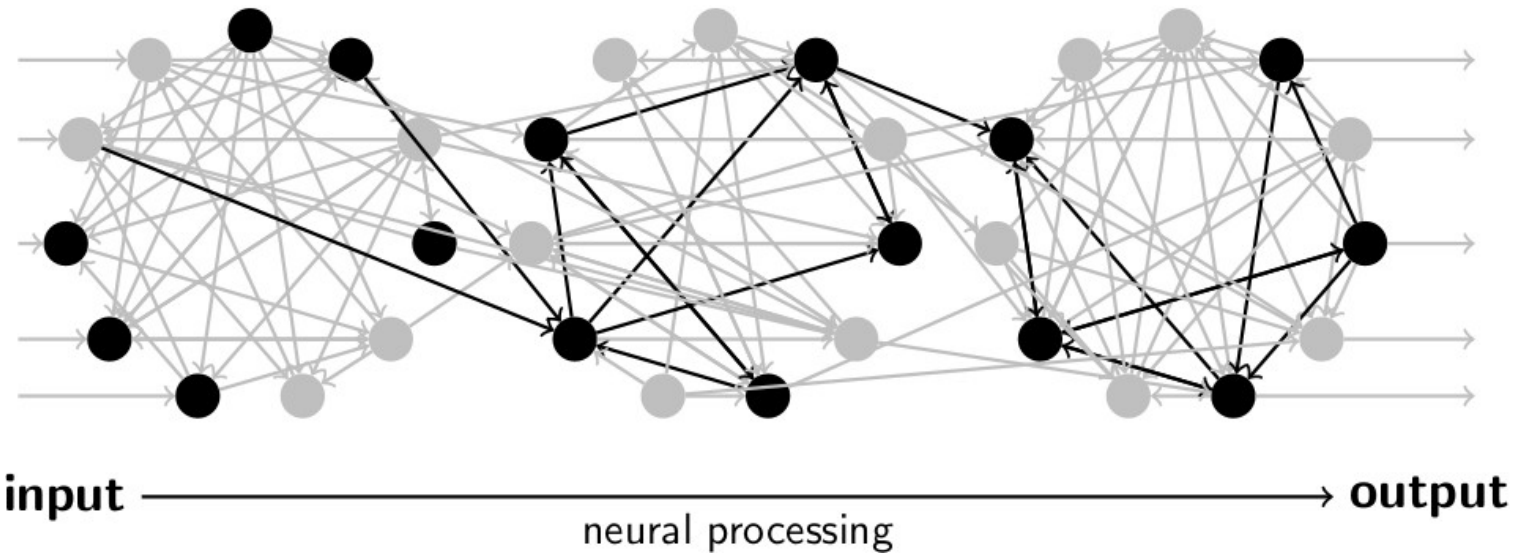


Towards functional neural network models

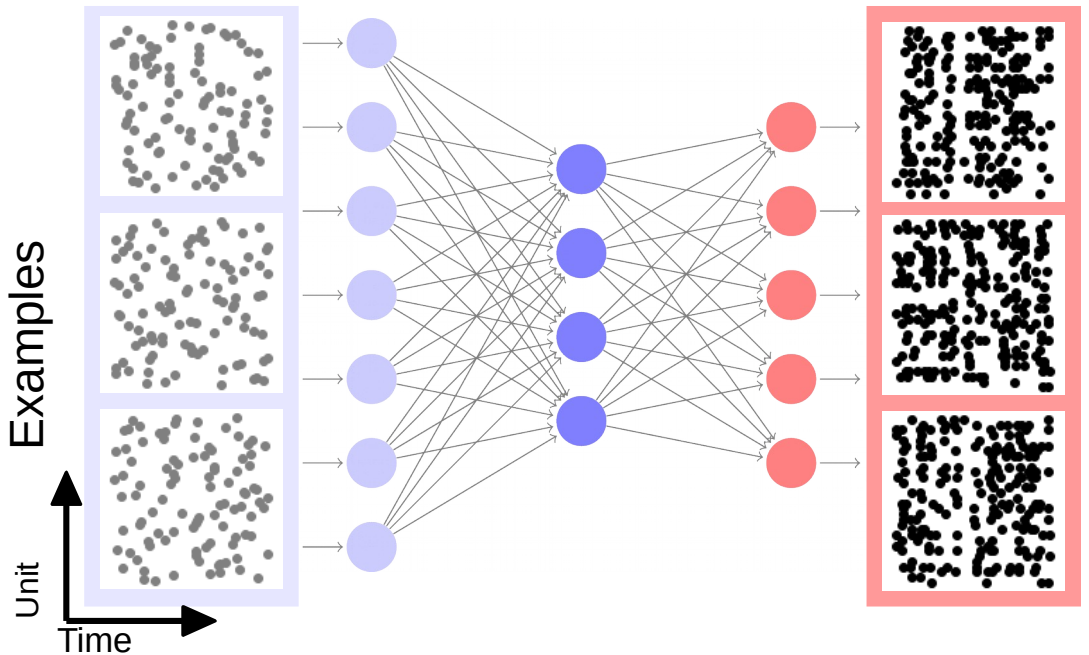
1) Input
(spatiotemporal)

3) Adjust weights
(surrogate gradients)

2) Output
(target spike train)

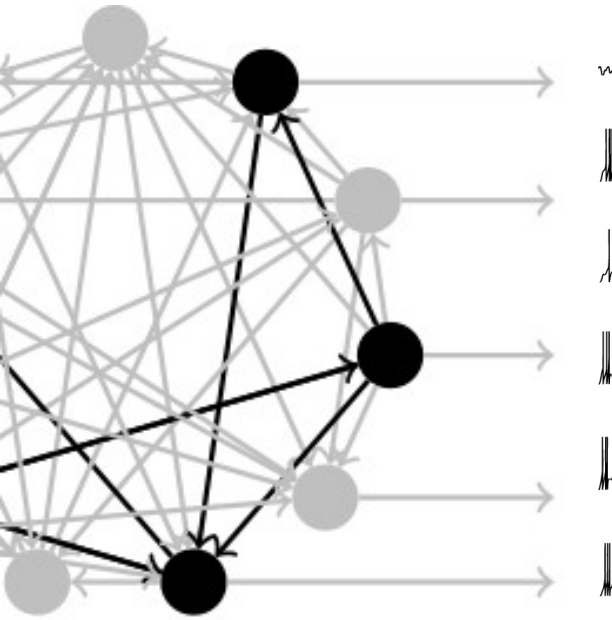


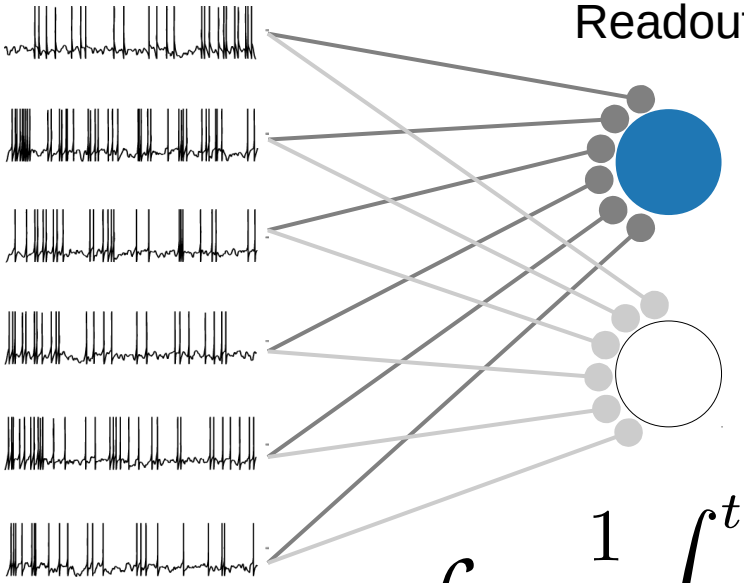
Starting point: Can we do supervised learning in spiking multi-layer networks with a local online learning rule?



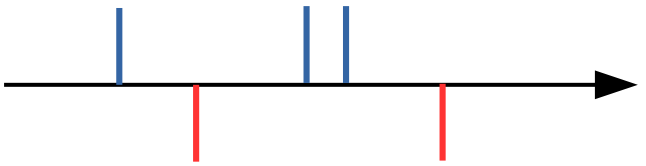
Zenke & Ganguli (2018)

Output: Van Rossum distance between output and target spike trains





Readout



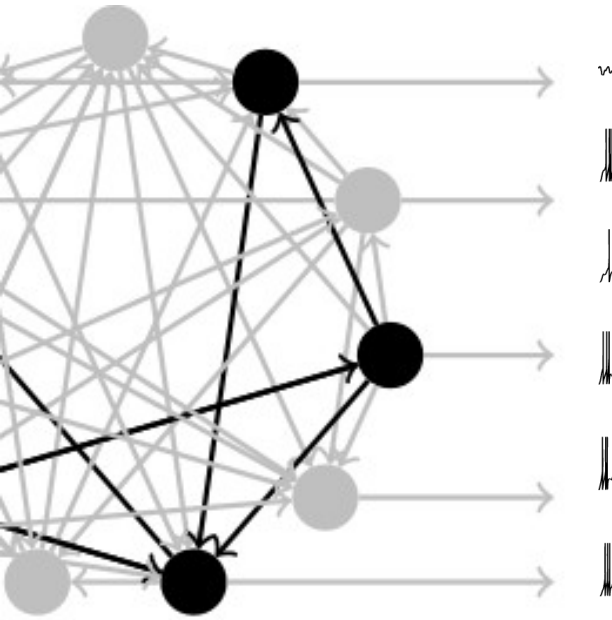
$$\mathcal{L} = \frac{1}{2}\int_{-\infty}^{t} \lambda * \left(\hat{S}(s) - S(s)\right)^2 ds$$

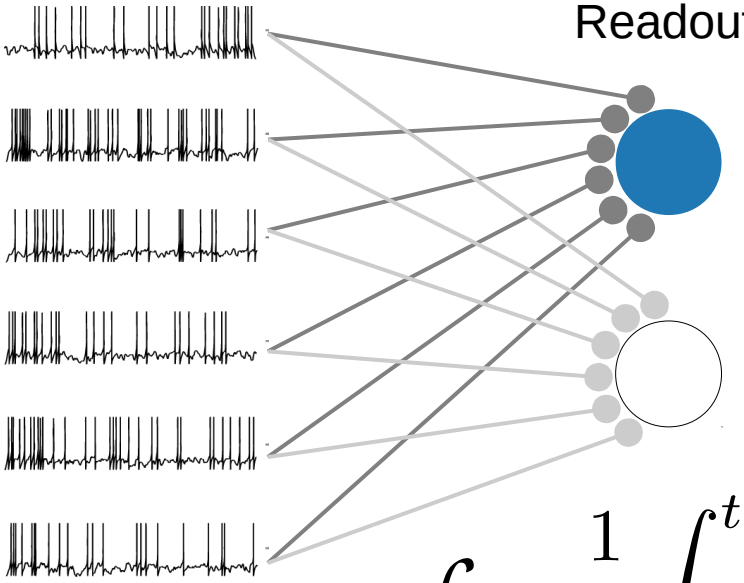
output

Zenke & Ganguli (2018)

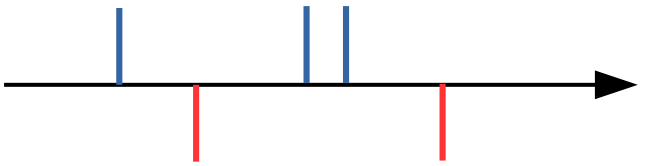
$$S(t) = \sum_l \delta(t - t^l)$$

Output: Van Rossum distance between output and target spike trains





Readout



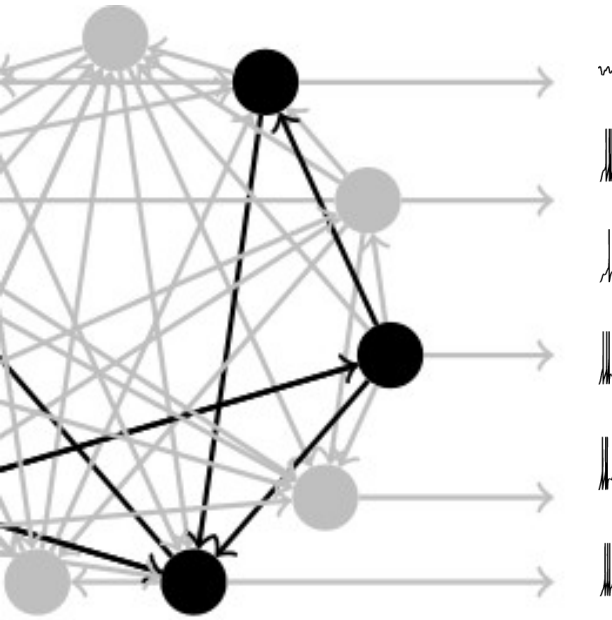
$$\mathcal{L} = \frac{1}{2}\int_{-\infty}^{t} \lambda * \left(\hat{S}(s) - S(s)\right)^2 ds$$

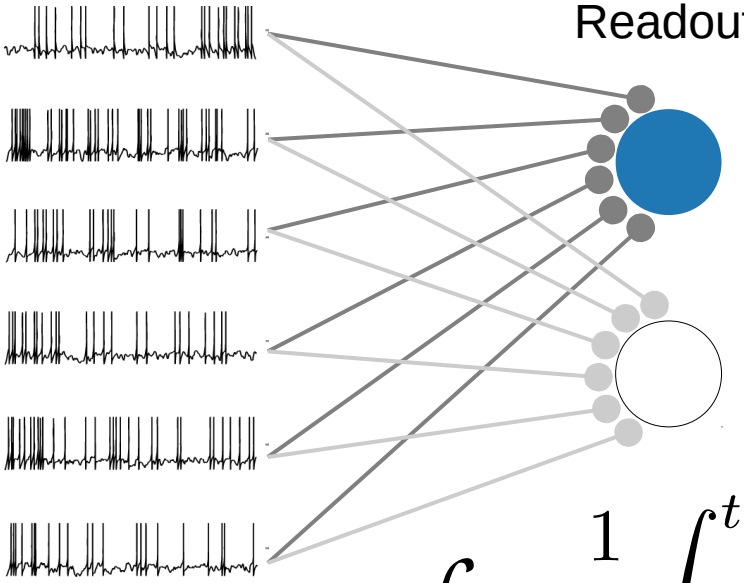
output

Zenke & Ganguli (2018)

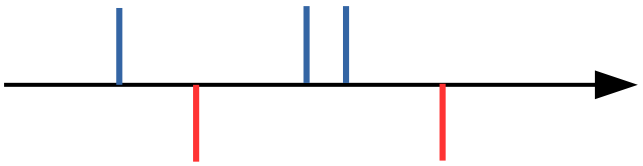
$$S(t) = \sum_l \delta(t - t^l)$$

Output: Van Rossum distance between output and target spike trains





Readout



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

output

Zenke & Ganguli (2018)

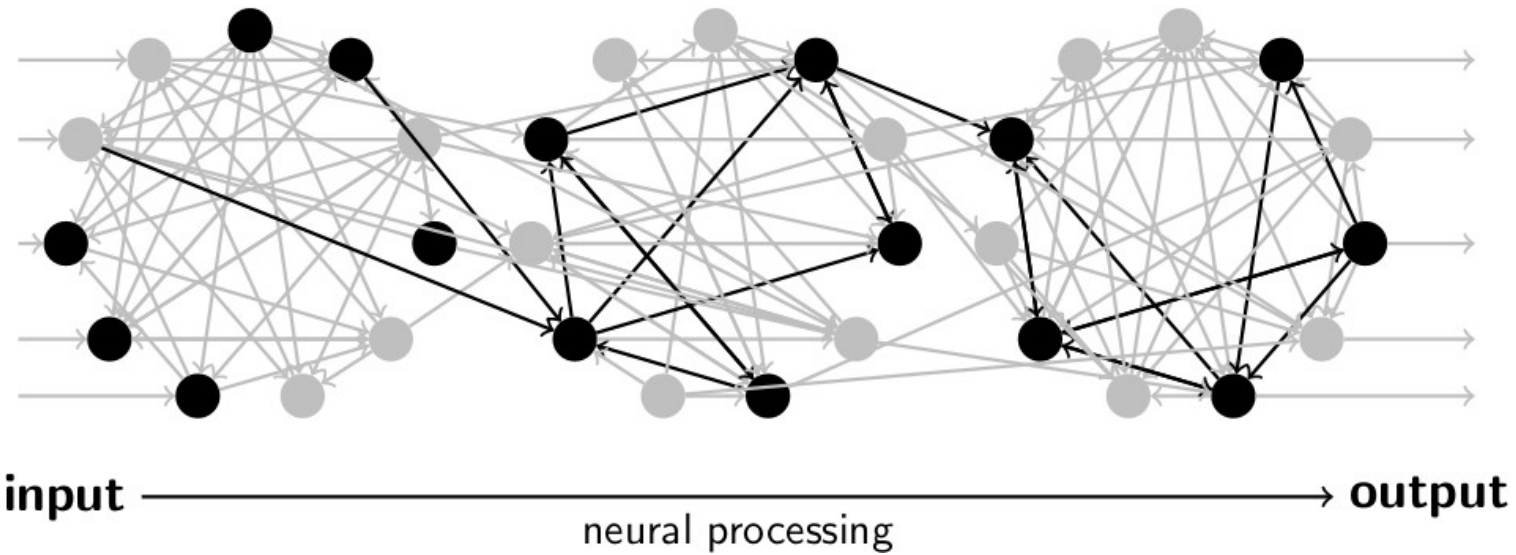
$$S(t) = \sum_l \delta(t - t^l)$$

Towards spiking network models which compute

1) Input
(spatiotemporal)

3) Adjust weights
(gradient descent)

2) Output
(classification)



Problem: The derivative of a spike train vanishes almost everywhere

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

Zenke & Ganguli (2018)

$$S(t) = \sum_l \delta(t - t^l)$$

Problem: The derivative of a spike train vanishes almost everywhere

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

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

Zenke & Ganguli (2018)

$$S(t) = \sum_l \delta(t - t^l)$$

Problem: The derivative of a spike train vanishes almost everywhere

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

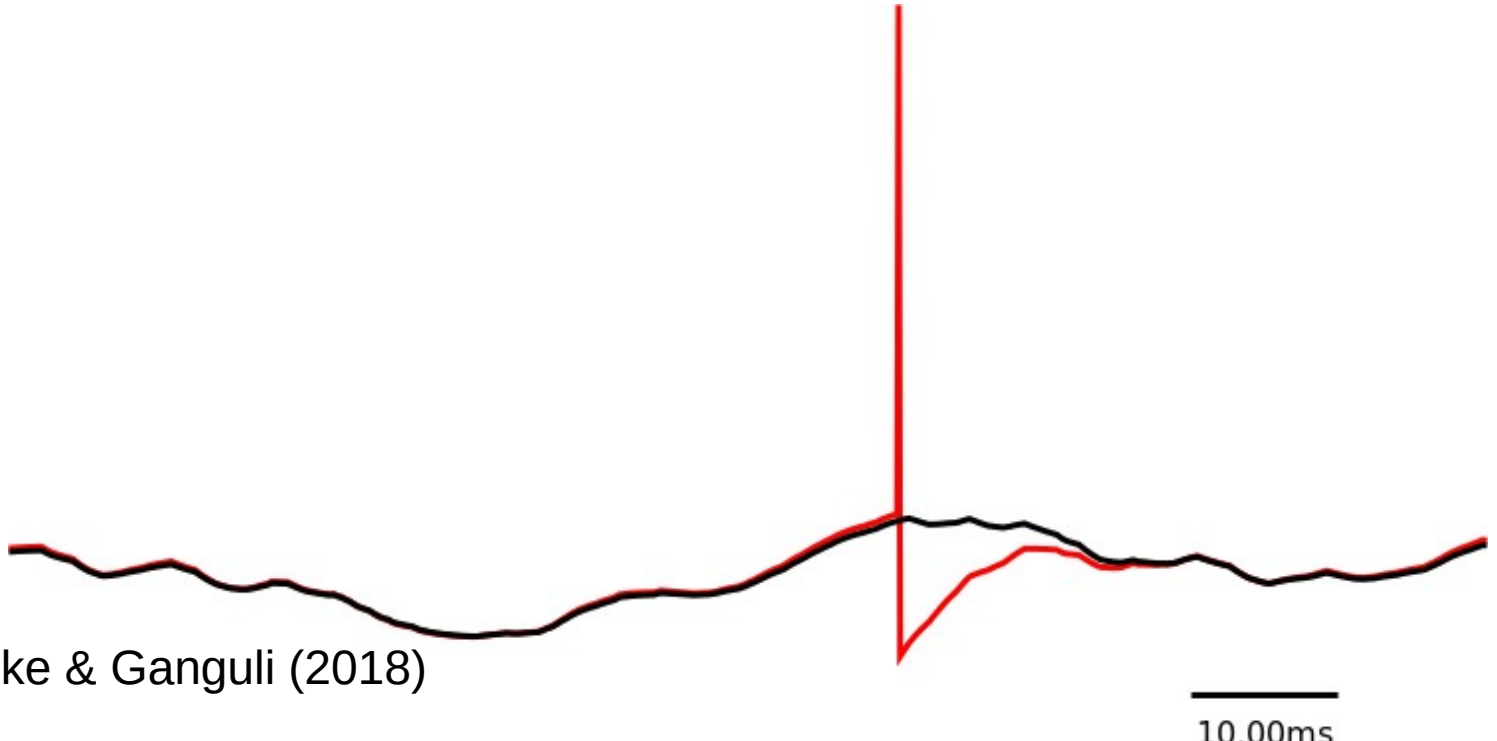
D'Oh!

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

Zenke & Ganguli (2018)

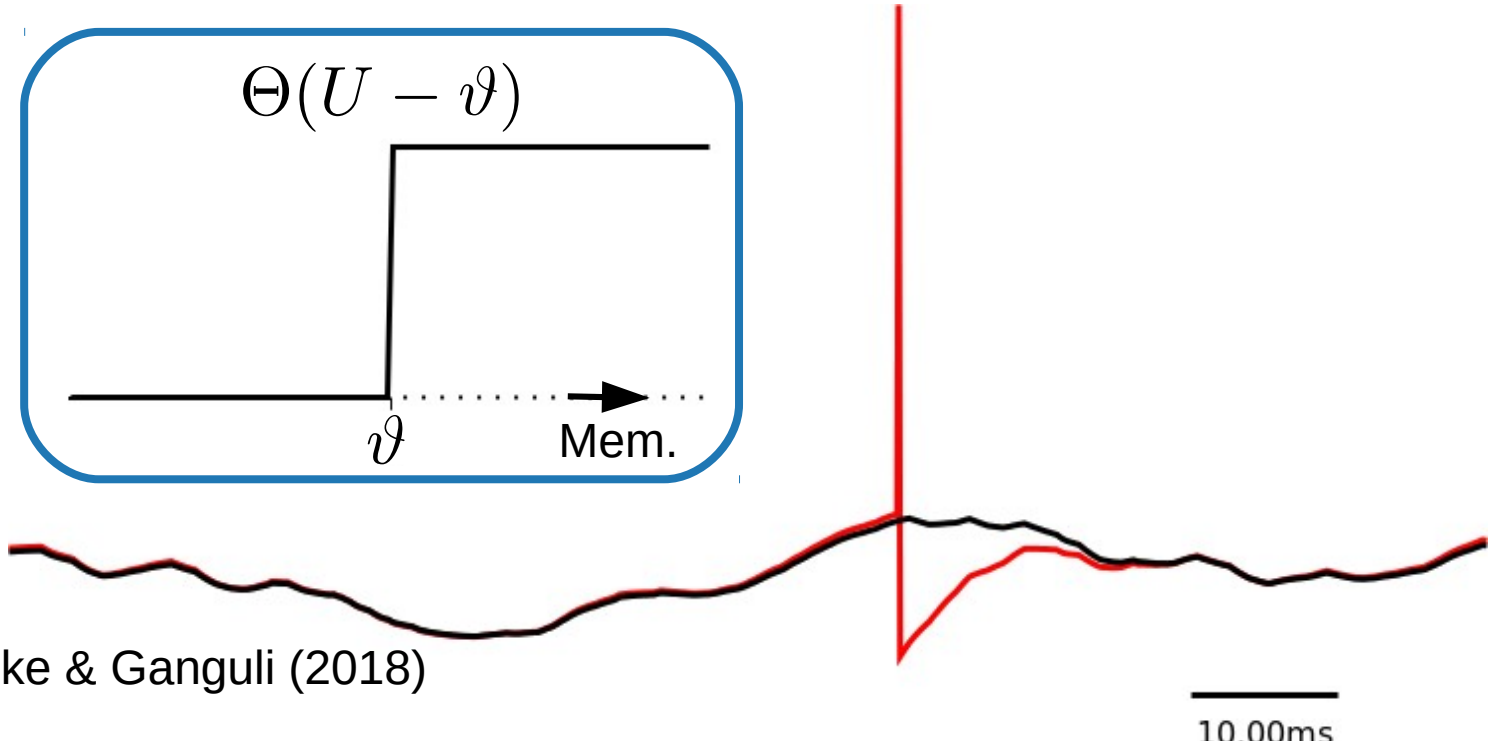
$$S(t) = \sum_l \delta(t - t^l)$$

Problem: The derivative of a spike train vanishes almost everywhere



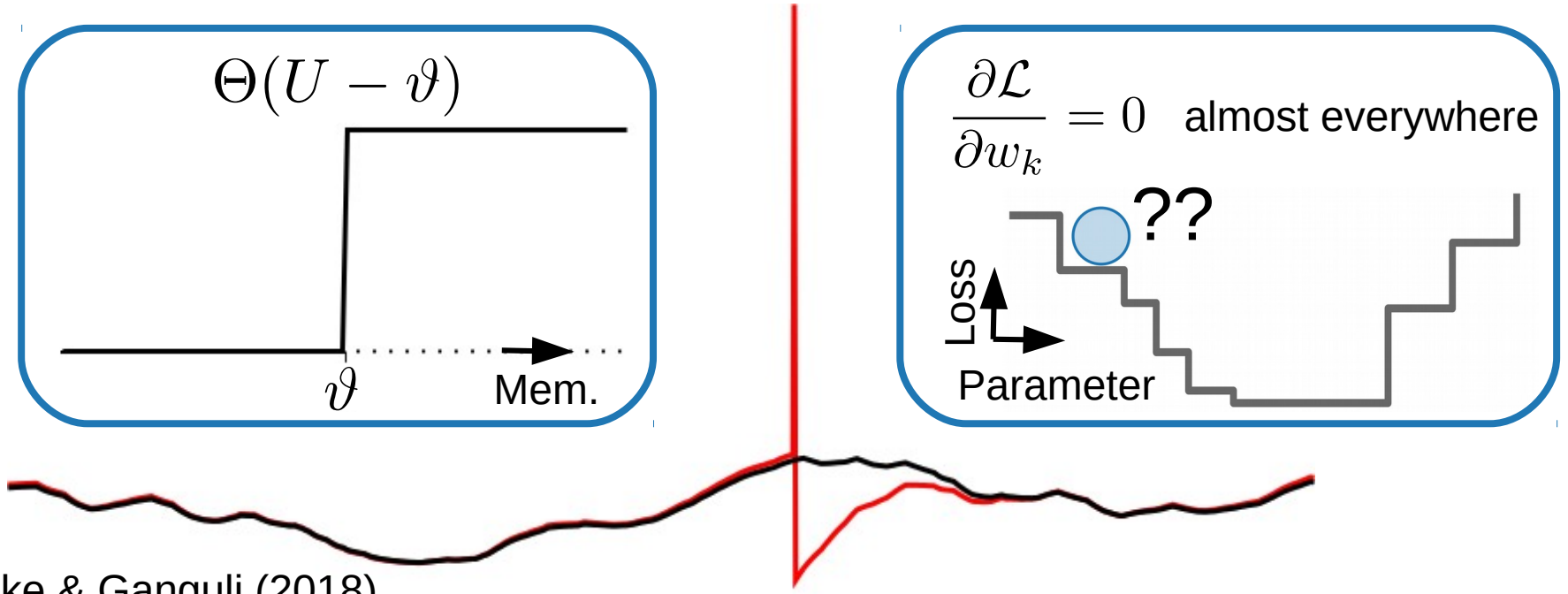
Zenke & Ganguli (2018)

Problem: The derivative of a spike train vanishes almost everywhere



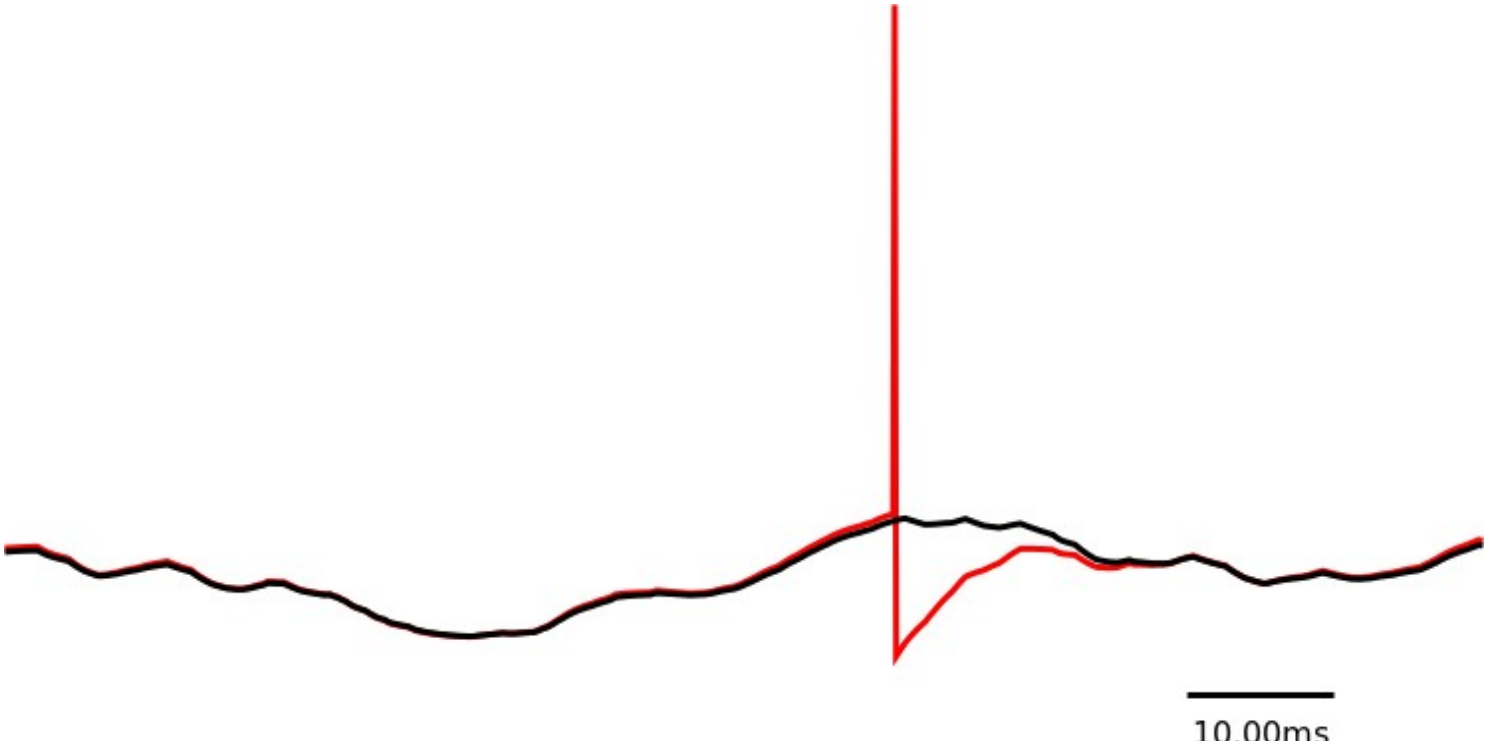
Zenke & Ganguli (2018)

Problem: The derivative of a spike train vanishes almost everywhere



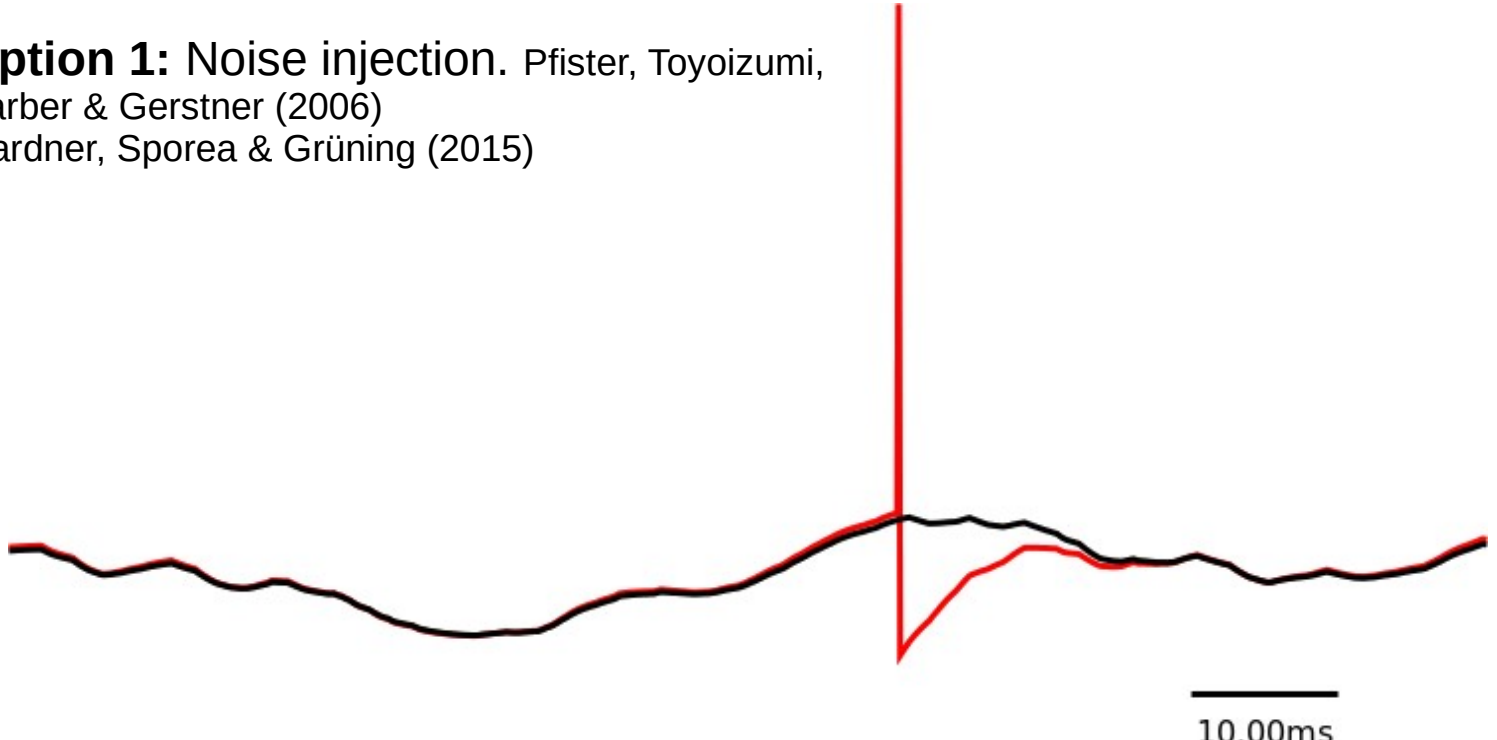
Zenke & Ganguli (2018)

An **awesome problem**
& a history of struggle



An **awesome problem** & a history of struggle

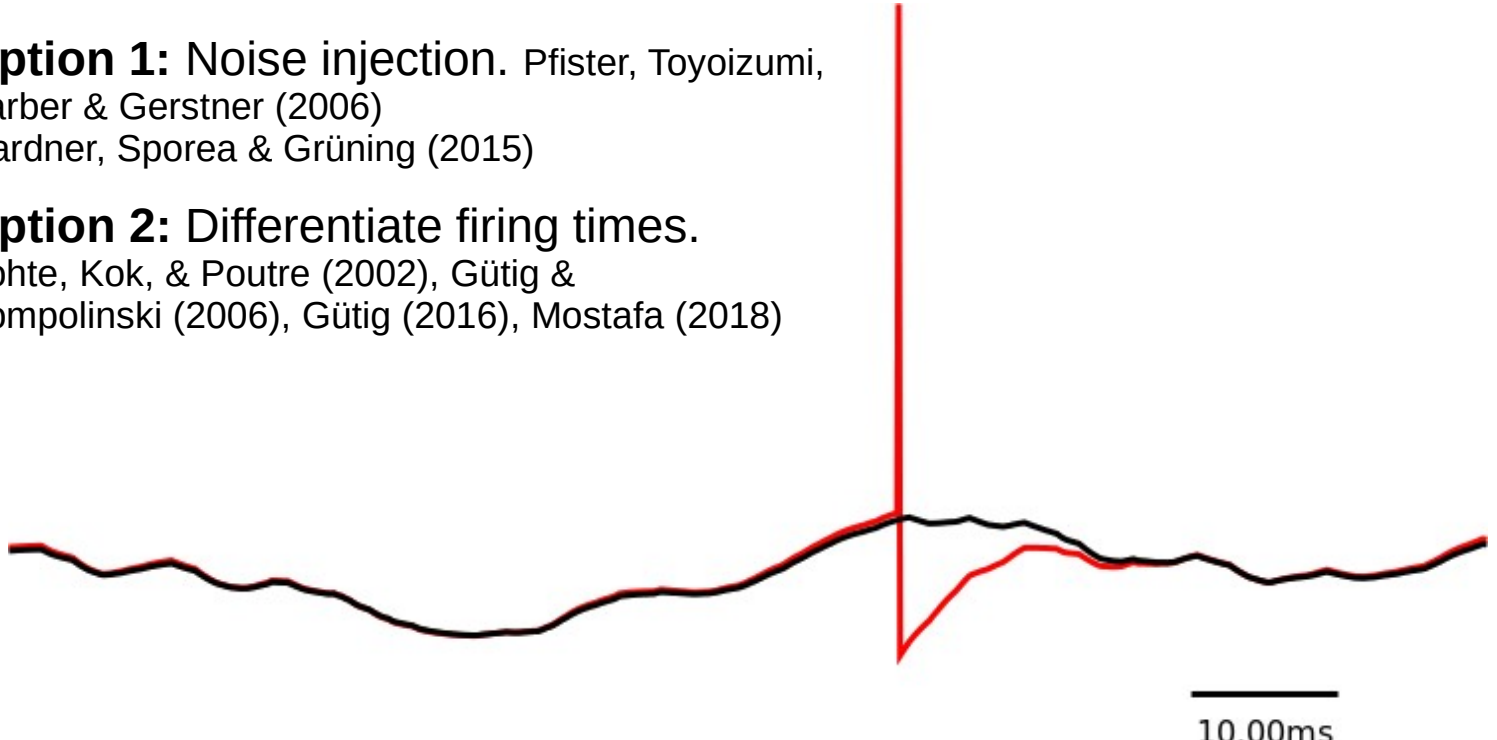
Option 1: Noise injection. Pfister, Toyoizumi, Barber & Gerstner (2006)
Gardner, Sporea & Grüning (2015)



An **awesome problem** & a history of struggle

Option 1: Noise injection. Pfister, Toyoizumi, Barber & Gerstner (2006)
Gardner, Sporea & Grüning (2015)

Option 2: Differentiate firing times.
Bohte, Kok, & Poutre (2002), Gütig & Sompolinski (2006), Gütig (2016), Mostafa (2018)

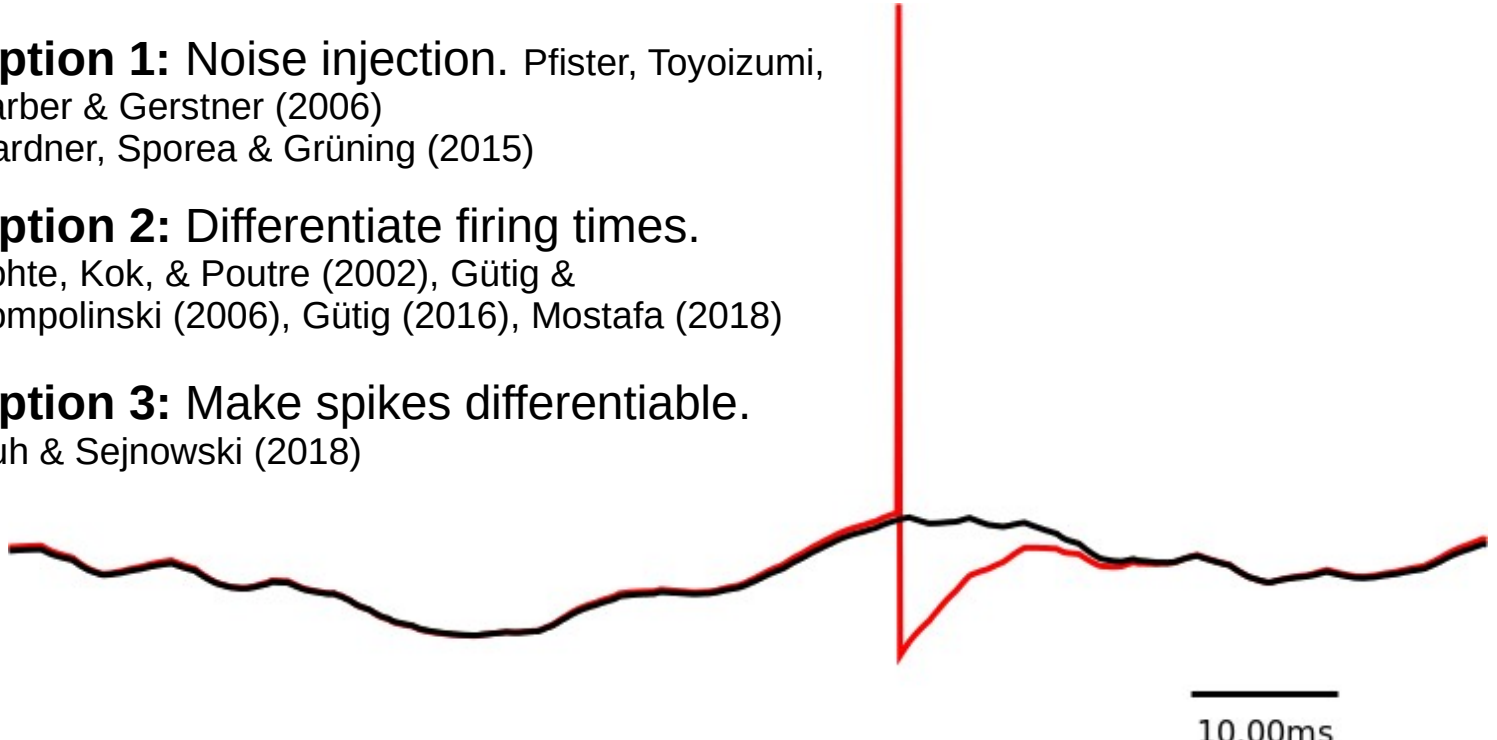


An **awesome problem** & a history of struggle

Option 1: Noise injection. Pfister, Toyoizumi, Barber & Gerstner (2006)
Gardner, Sporea & Grüning (2015)

Option 2: Differentiate firing times.
Bohte, Kok, & Poutre (2002), Gütig & Sompolinski (2006), Gütig (2016), Mostafa (2018)

Option 3: Make spikes differentiable.
Huh & Sejnowski (2018)



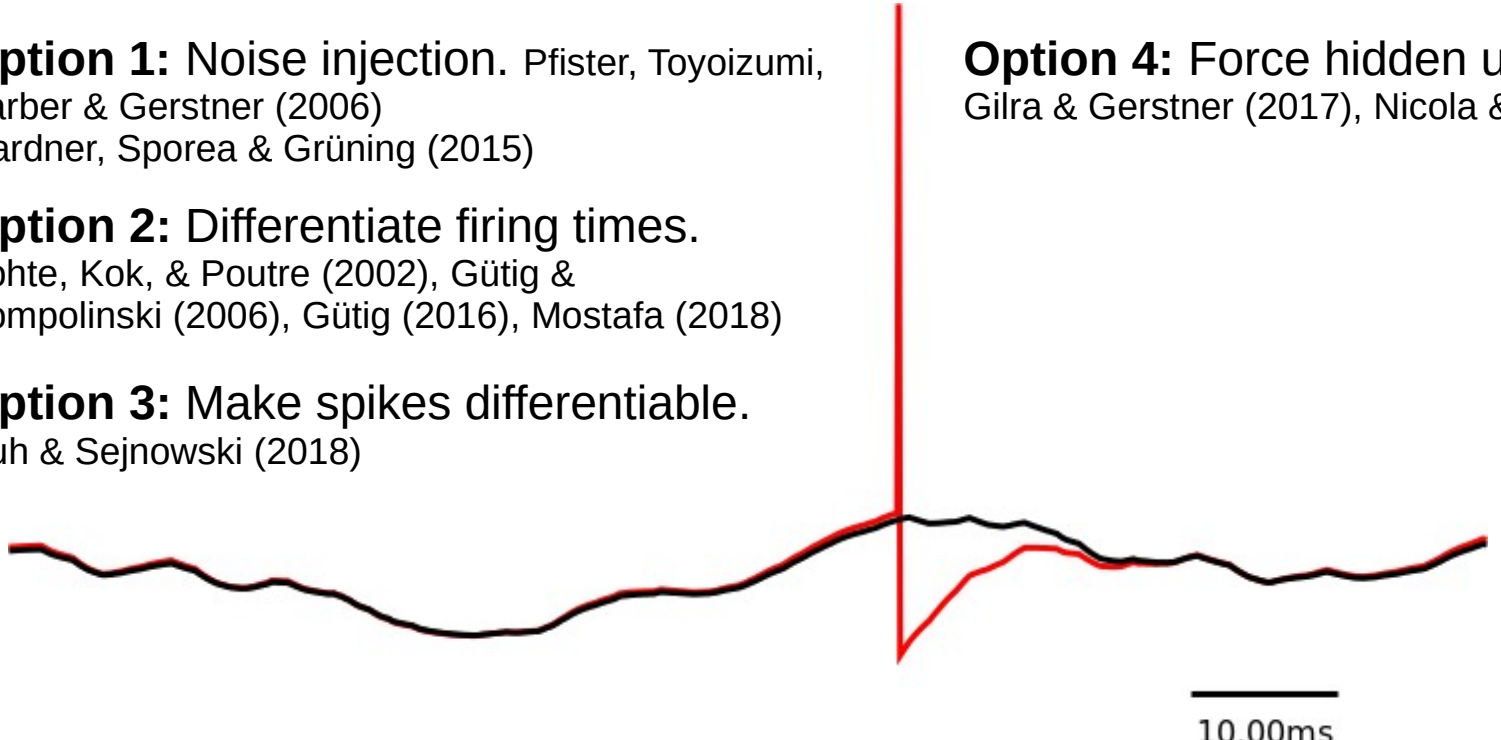
An **awesome problem** & a history of struggle

Option 1: Noise injection. Pfister, Toyoizumi, Barber & Gerstner (2006)
Gardner, Sporea & Grüning (2015)

Option 2: Differentiate firing times.
Bohte, Kok, & Poutre (2002), Gütig & Sompolinski (2006), Gütig (2016), Mostafa (2018)

Option 3: Make spikes differentiable.
Huh & Sejnowski (2018)

Option 4: Force hidden units “on target”.
Gilra & Gerstner (2017), Nicola & Clopath (2017)



An **awesome problem** & a history of struggle

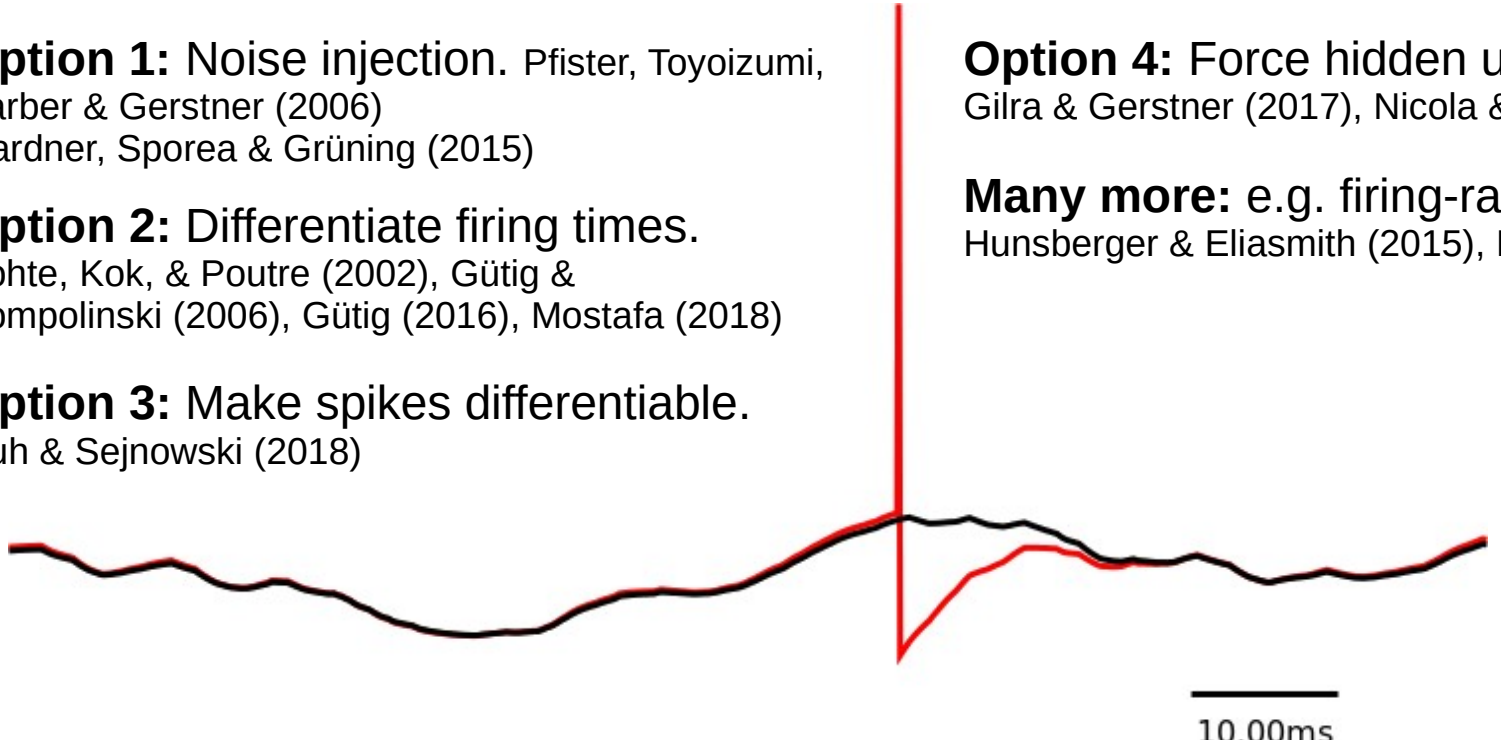
Option 1: Noise injection. Pfister, Toyoizumi, Barber & Gerstner (2006)
Gardner, Sporea & Grüning (2015)

Option 2: Differentiate firing times.
Bohte, Kok, & Poutre (2002), Gütig & Sompolinski (2006), Gütig (2016), Mostafa (2018)

Option 3: Make spikes differentiable.
Huh & Sejnowski (2018)

Option 4: Force hidden units “on target”.
Gilra & Gerstner (2017), Nicola & Clopath (2017)

Many more: e.g. firing-rate approaches
Hunsberger & Eliasmith (2015), Lee et al. (2016), ...



An **awesome problem** & a history of struggle

Today: Surrogate gradients.
Bohte (2011), Zenke & Ganguli (2018),
Shrestha & Orchard (2018),
Bellec, Salaj, Subramoney, Legenstein, and Maass (2018)
Neftci, Mostafa, & Zenke (2019)

In ML: “Straight-through estimators” Bengio et al. (2013)

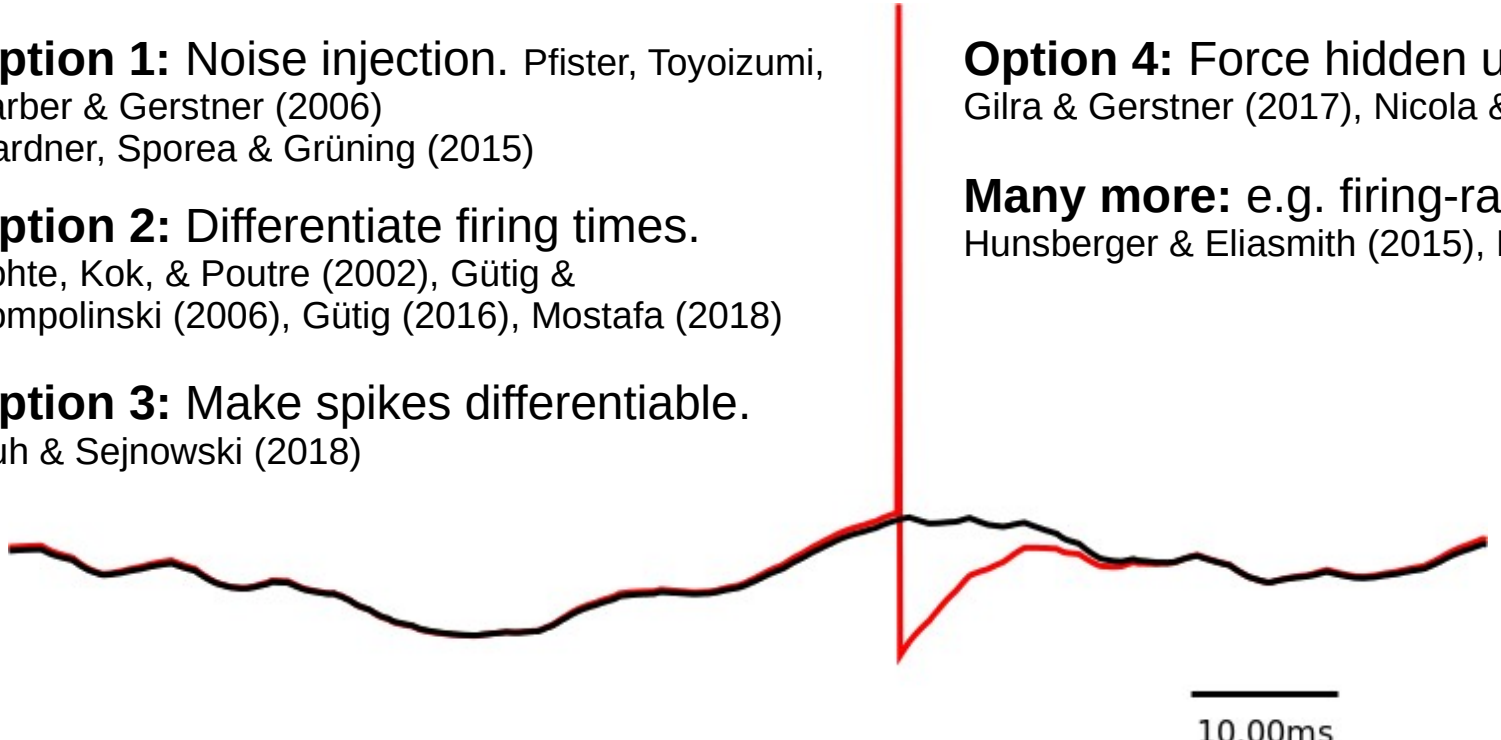
Option 1: Noise injection. Pfister, Toyoizumi, Barber & Gerstner (2006)
Gardner, Sporea & Grüning (2015)

Option 2: Differentiate firing times.
Bohte, Kok, & Poutre (2002), Gütig & Sompolinski (2006), Gütig (2016), Mostafa (2018)

Option 3: Make spikes differentiable.
Huh & Sejnowski (2018)

Option 4: Force hidden units “on target”.
Gilra & Gerstner (2017), Nicola & Clopath (2017)

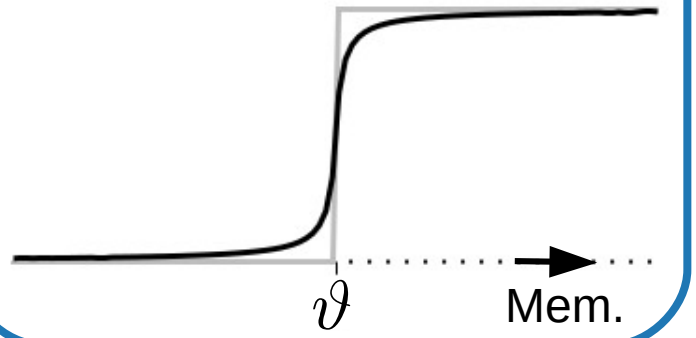
Many more: e.g. firing-rate approaches
Hunsberger & Eliasmith (2015), Lee et al. (2016), ...



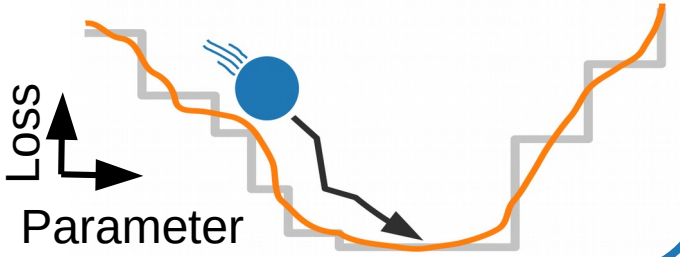
Option 5: Surrogate gradients.
Bohte (2011), Zenke & Ganguli (2018),
Shrestha & Orchard (2018),
Bellec, Salaj, Subramoney, Legenstein, and Maass (2018)
Neftci, Mostafa, & Zenke (2019)

In ML: “Straight-through estimators” Bengio et al. (2013)

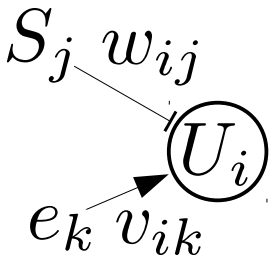
$$\Theta(U(t) - \vartheta) \approx \sigma(U(t))$$

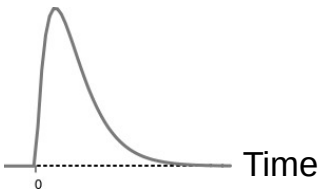


$$\frac{\partial \mathcal{L}}{\partial w_k} \neq 0$$



SuperSpike: An online rule for surrogate gradient learning in spiking neural networks

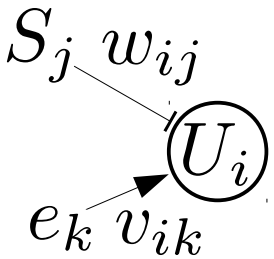


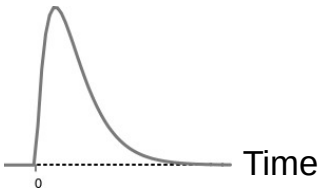


$$\frac{\partial w_{ij}}{\partial t} \equiv \lambda * (\underbrace{\epsilon * S_j(t)}_{\text{pre}} \underbrace{f(U_i)}_{\text{post}}) \quad \underbrace{e_i(t)}_{\text{error signal}}$$

Zenke & Ganguli (2018)

SuperSpike: An online rule for surrogate gradient learning in spiking neural networks



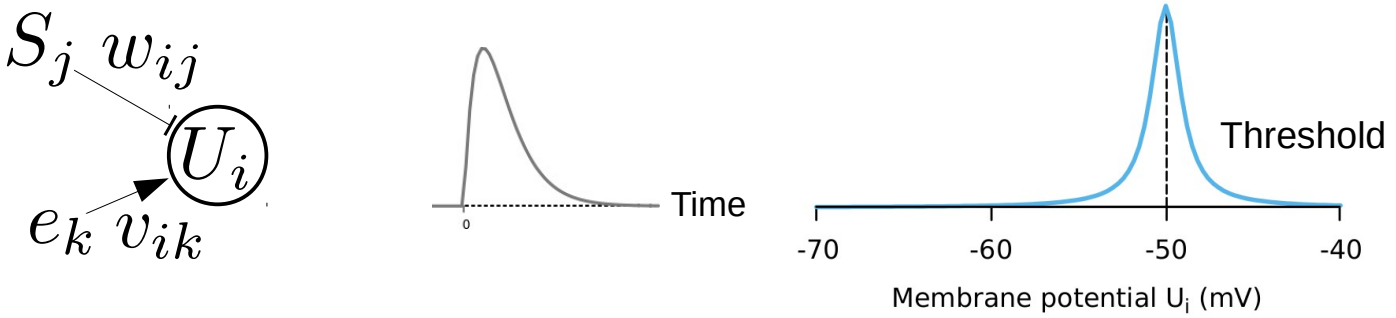


$$\frac{\partial w_{ij}}{\partial t} \equiv \lambda * \underbrace{(\epsilon * S_j(t)}_{\text{pre}} \underbrace{f(U_i))}_{\text{post}} \underbrace{e_i(t)}_{\text{error signal}}$$

- Hebbian part “STDP-like”

Zenke & Ganguli (2018)

SuperSpike: An online rule for surrogate gradient learning in spiking neural networks

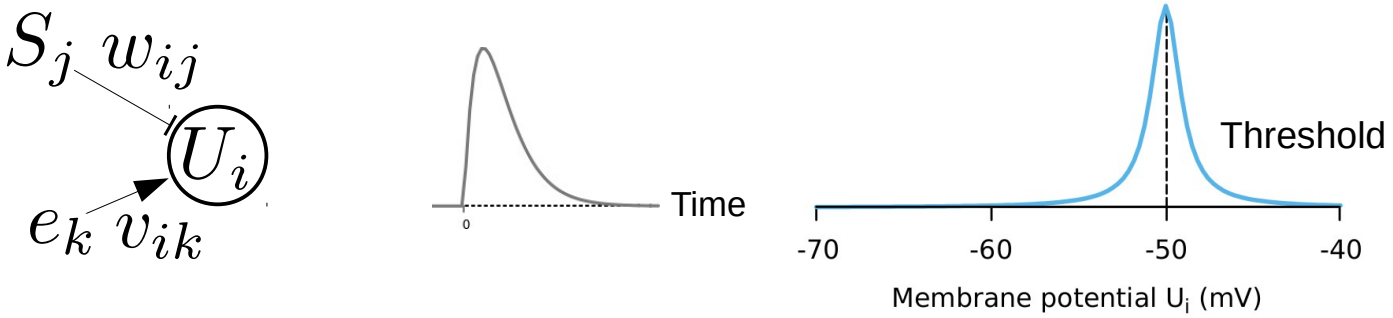


$$\frac{\partial w_{ij}}{\partial t} \equiv \lambda * \underbrace{(\epsilon * S_j(t)}_{\text{pre}} \underbrace{f(U_i))}_{\text{post}} \underbrace{e_i(t)}_{\text{error signal}}$$

- Hebbian part “STDP-like”
- Voltage nonlinearity

Zenke & Ganguli (2018)

SuperSpike: An online rule for surrogate gradient learning in spiking neural networks

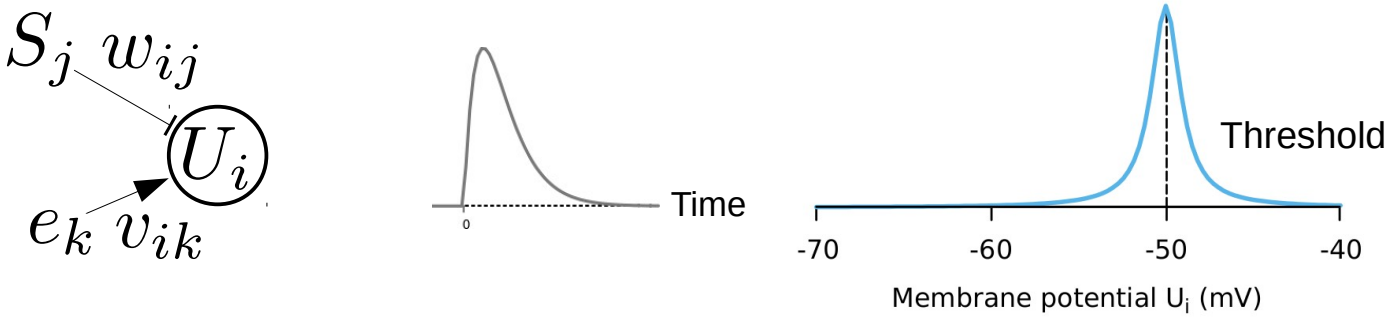


$$\frac{\partial w_{ij}}{\partial t} \equiv \lambda * \underbrace{(\epsilon * S_j(t)}_{\text{pre}} \underbrace{f(U_i))}_{\text{post}} \underbrace{e_i(t)}_{\text{error signal}}$$

- Hebbian part “STDP-like”
- Voltage nonlinearity
- Third factor ”specific feedback”

Zenke & Ganguli (2018)

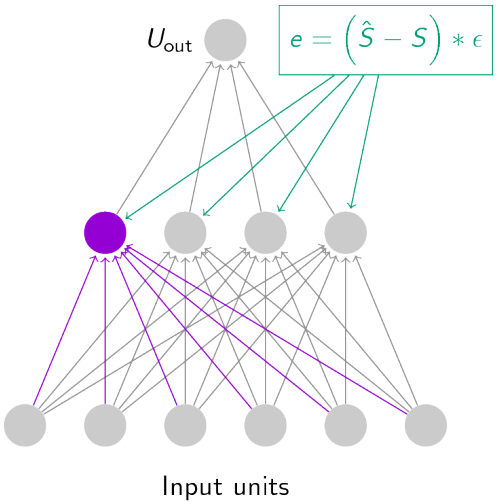
SuperSpike: An online rule for surrogate gradient learning in spiking neural networks

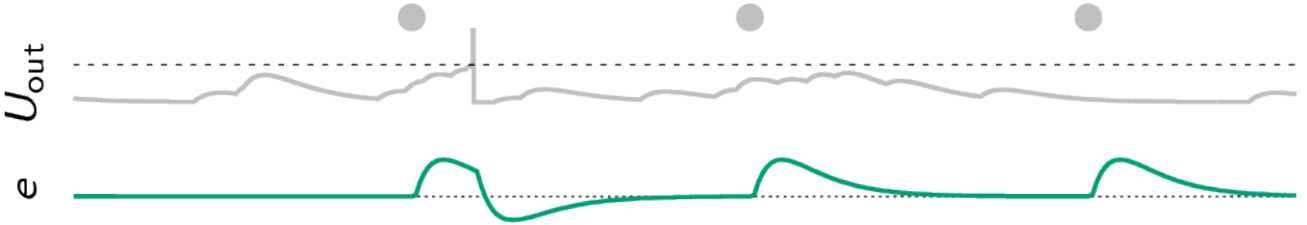


$$\frac{\partial w_{ij}}{\partial t} \equiv \lambda * \underbrace{(\epsilon * S_j(t)}_{\text{pre}} \underbrace{f(U_i))}_{\text{post}} \underbrace{e_i(t)}_{\text{error signal}}$$

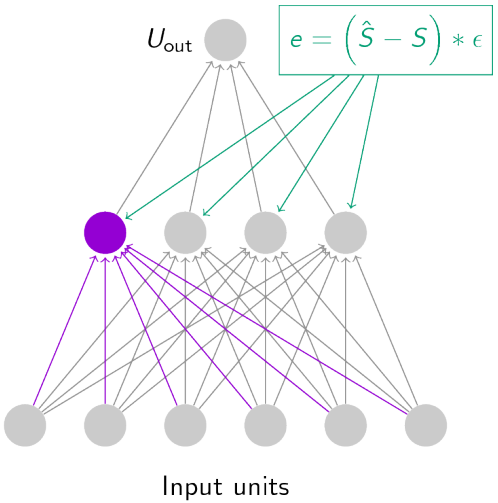
- Hebbian part “STDP-like”
- Voltage nonlinearity
- Third factor ”specific feedback”
- Eligibility trace “Ca transient”

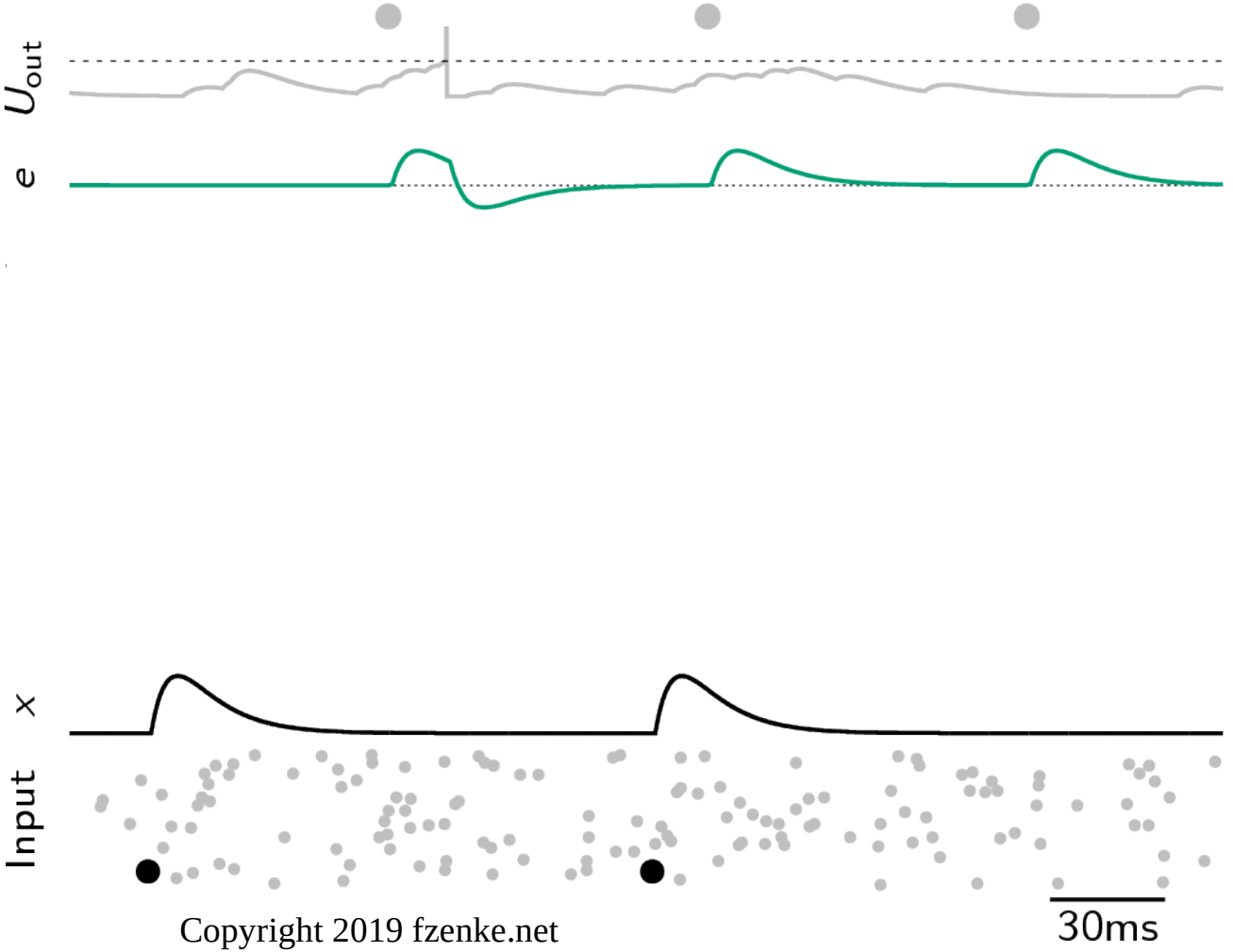
Zenke & Ganguli (2018)



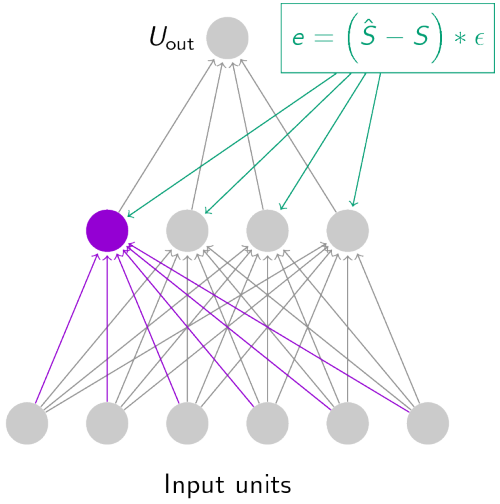


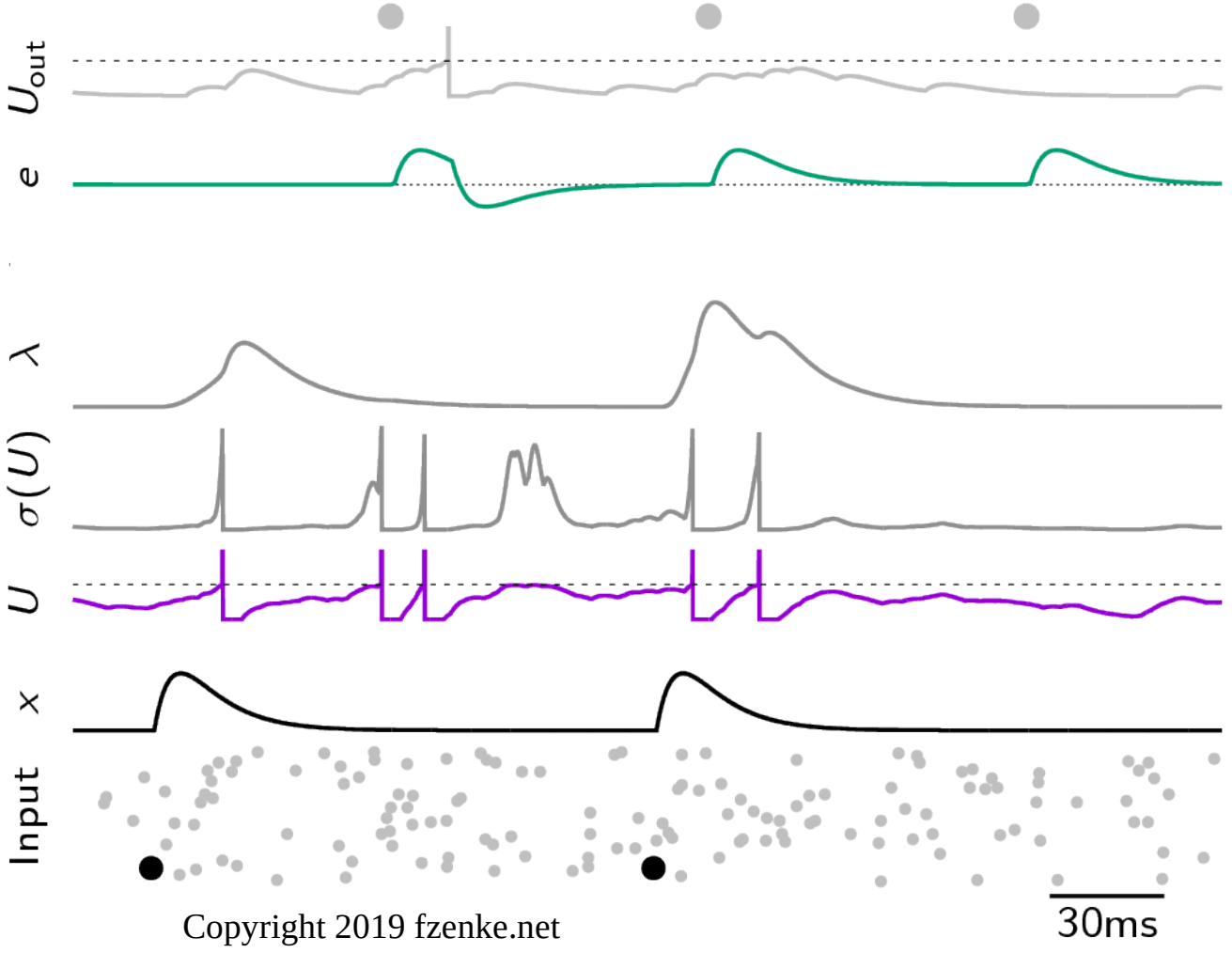
Zenke & Ganguli (2018)



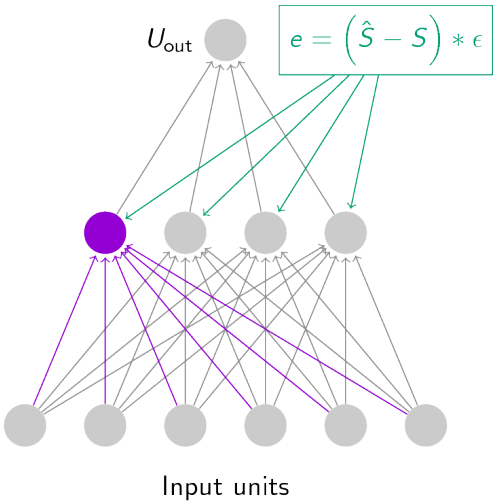


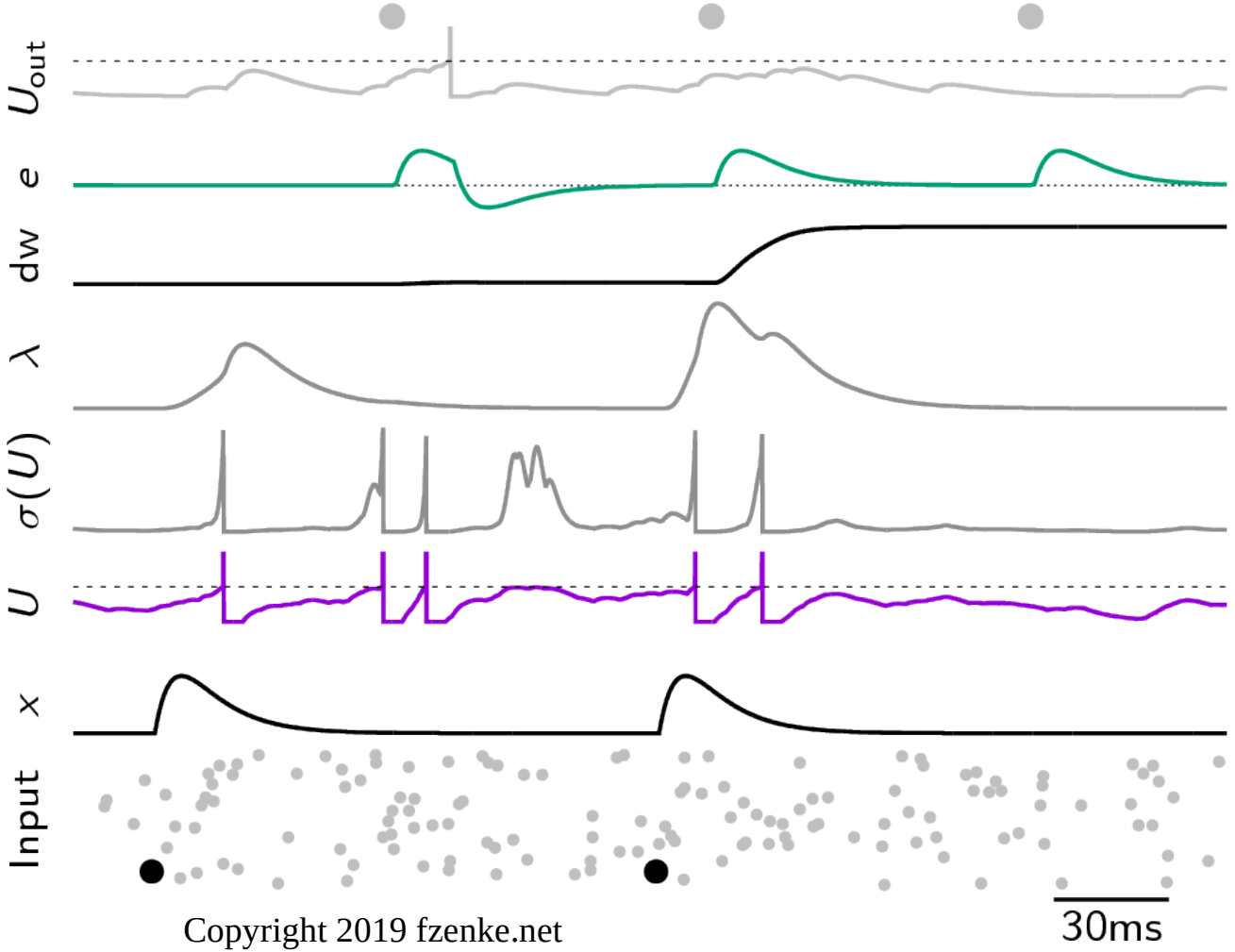
Zenke & Ganguli (2018)



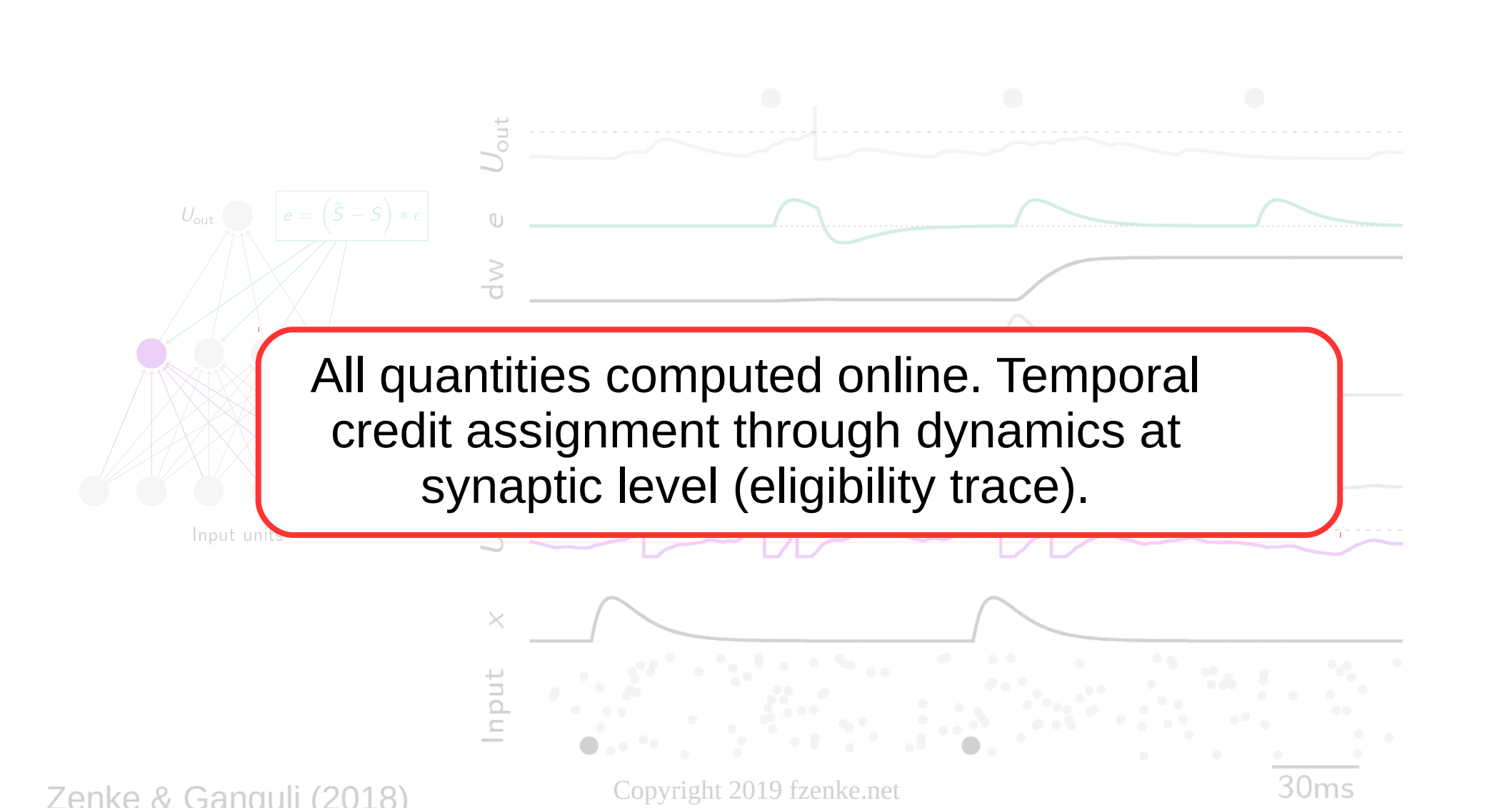


Zenke & Ganguli (2018)





Zenke & Ganguli (2018)



All quantities computed online. Temporal credit assignment through dynamics at synaptic level (eligibility trace).

Zenke & Ganguli (2018)

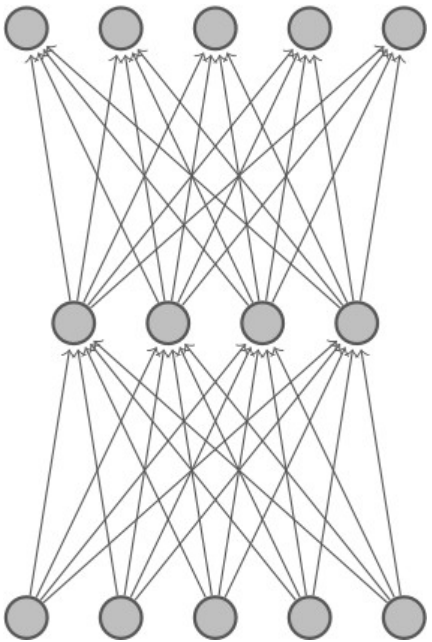
Output

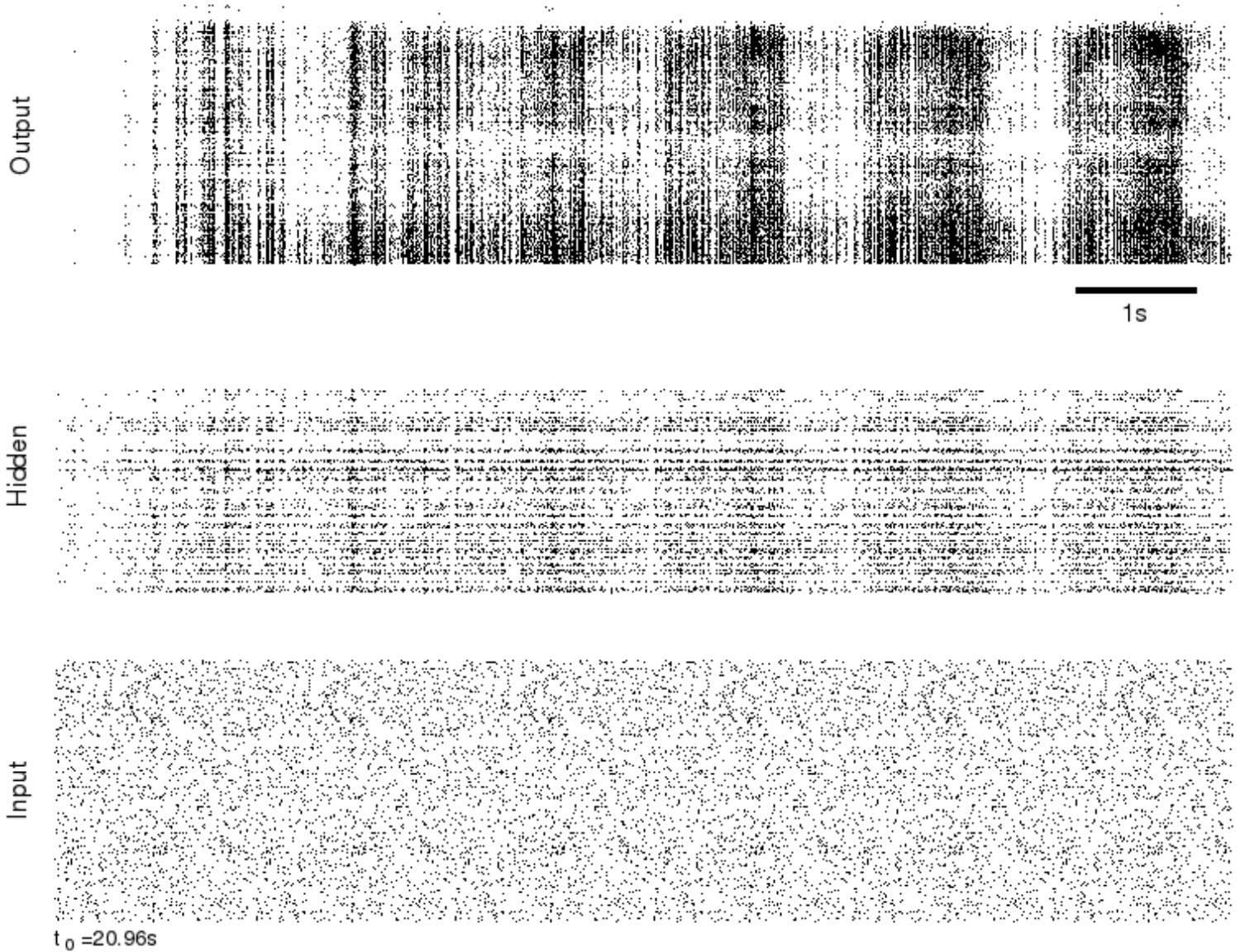
1s

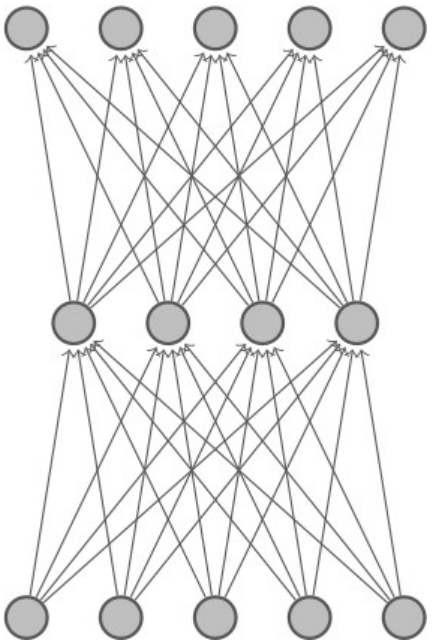
Hidden

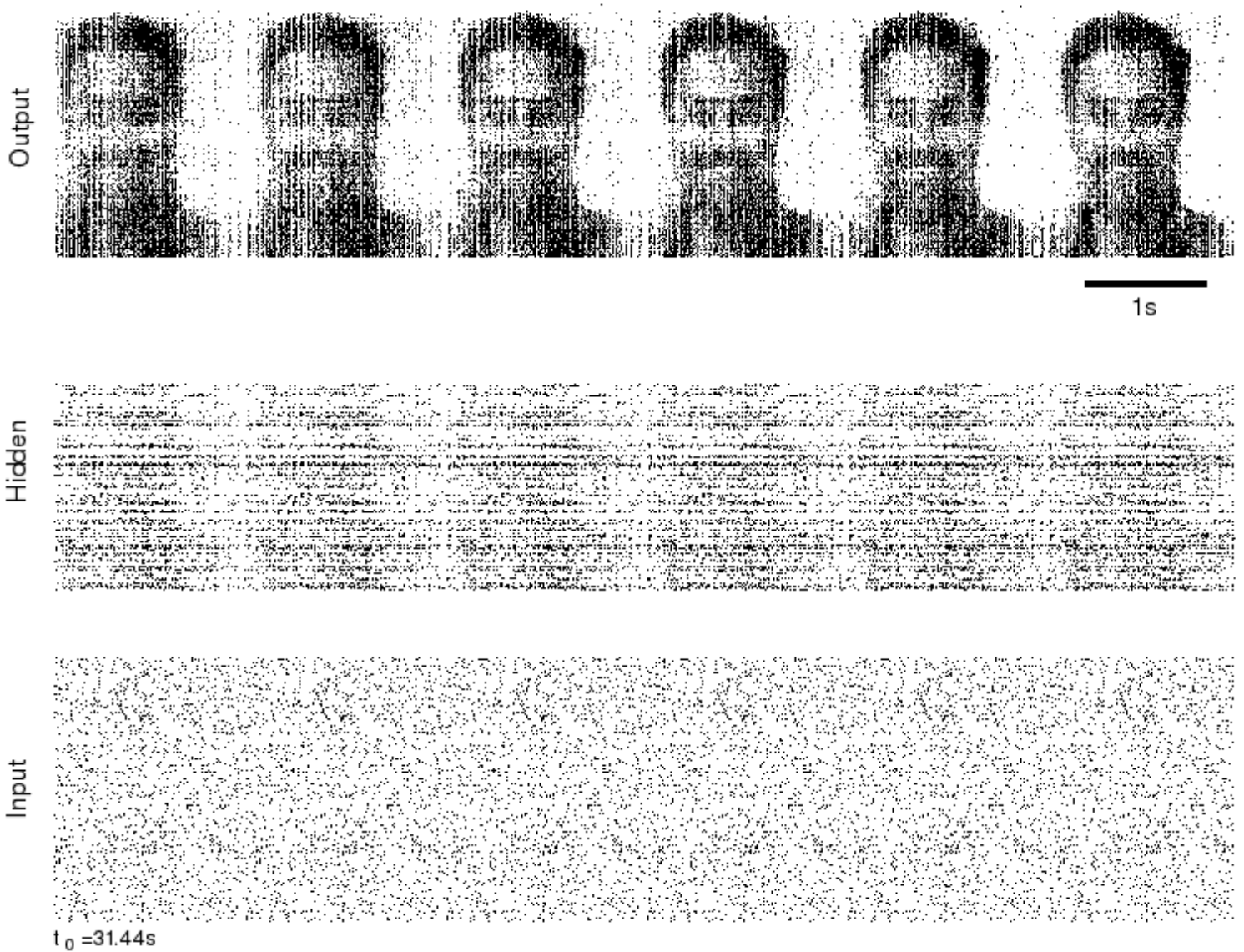
Input

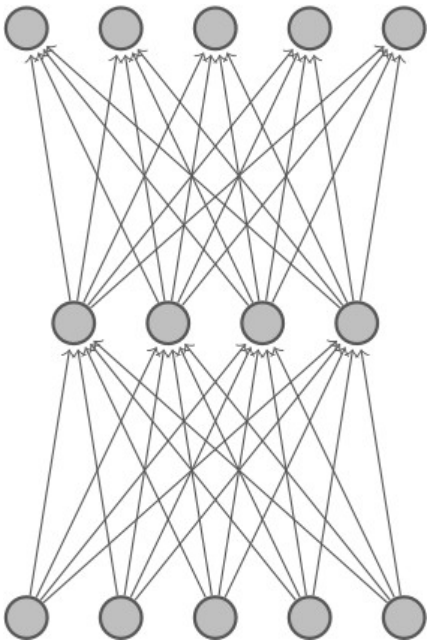
t_0 =0.00s

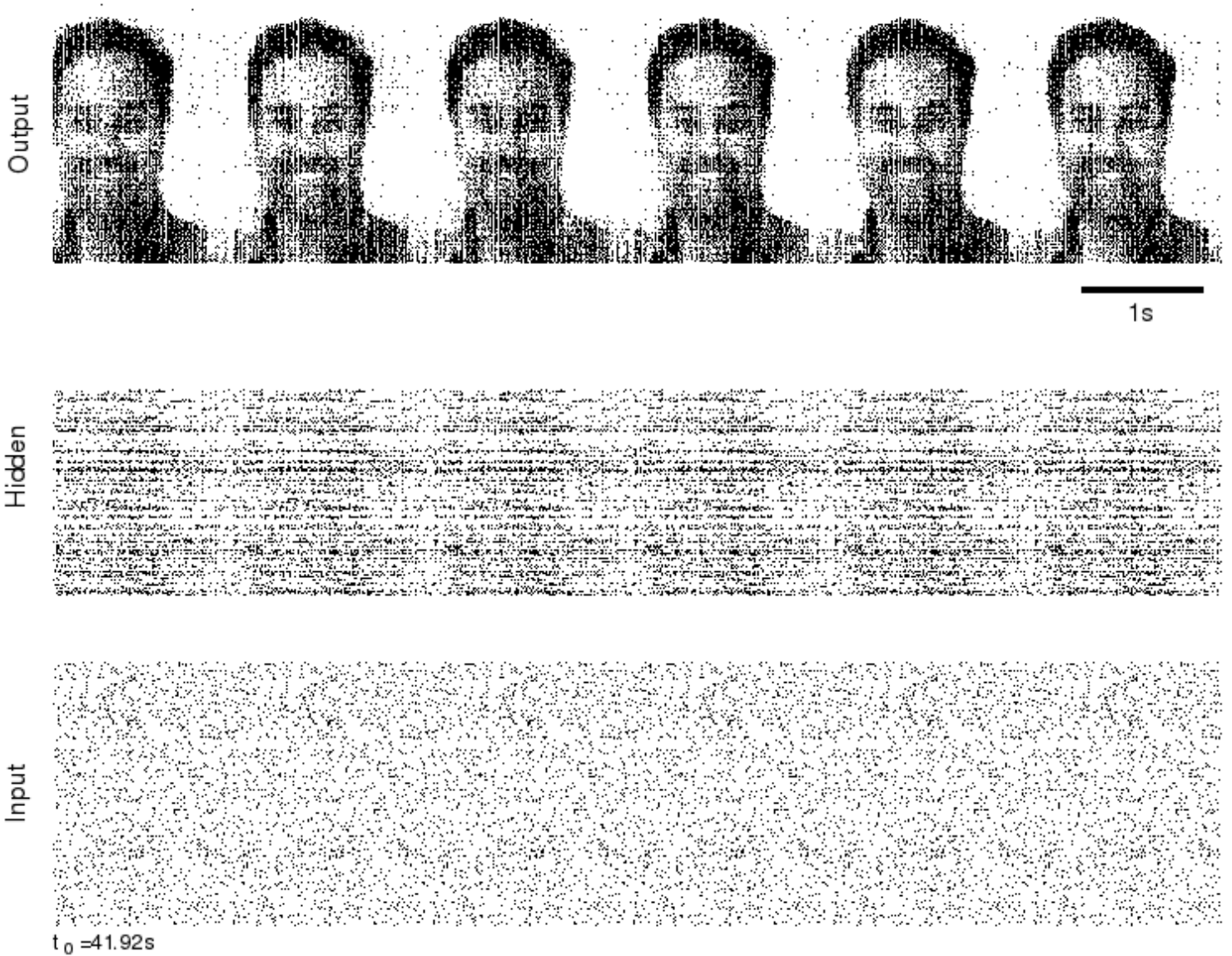


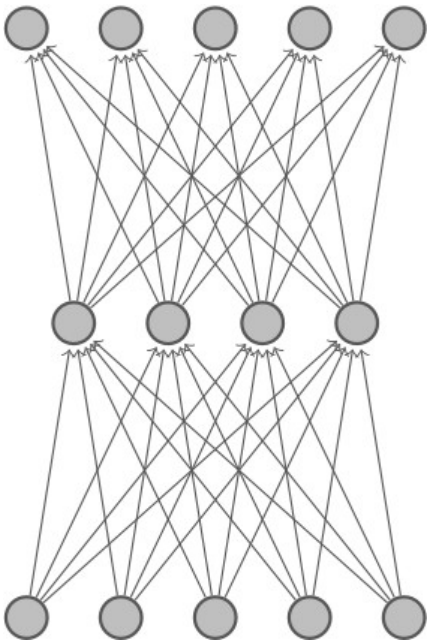

Output
1s
Hidden
Input
t 0 =20.96s




Output
1s
Hidden
Input
t 0 =31.44s

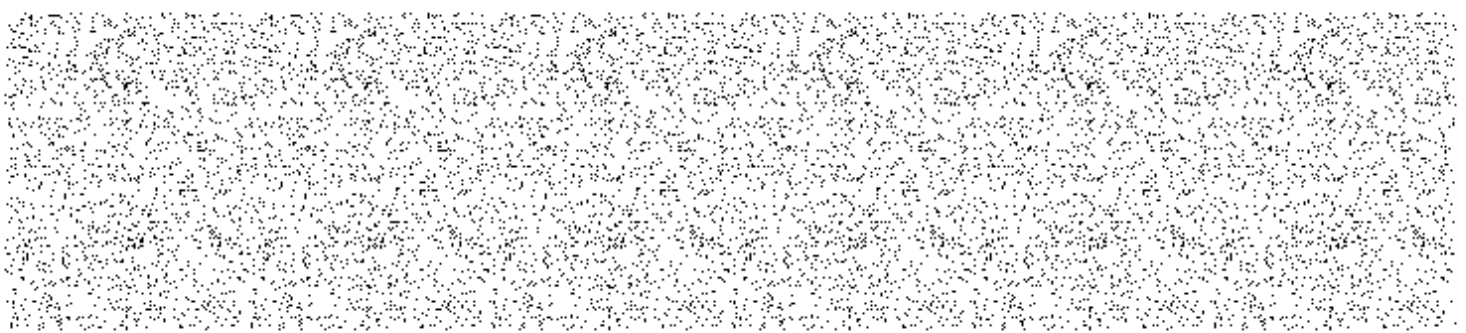



Output
1s
Hidden
Input
t_0 =41.92s

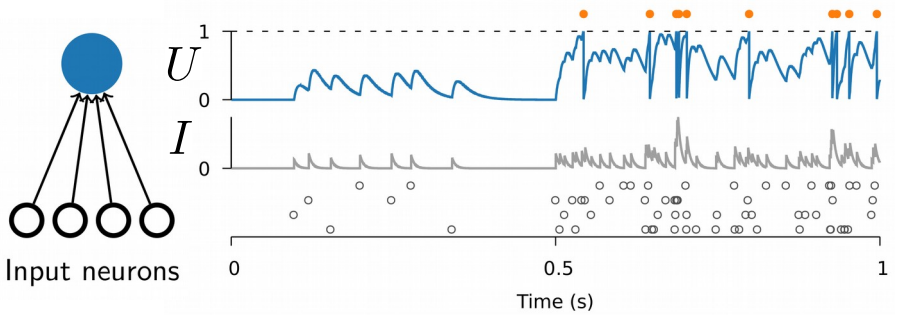



Output
1s


Hidden

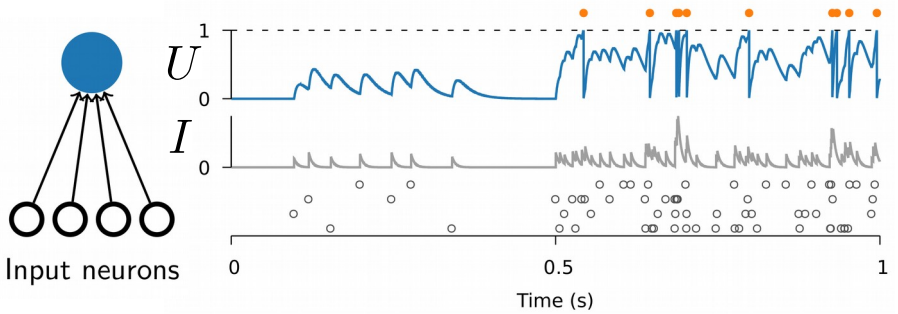

Input
t_0 =744.08s

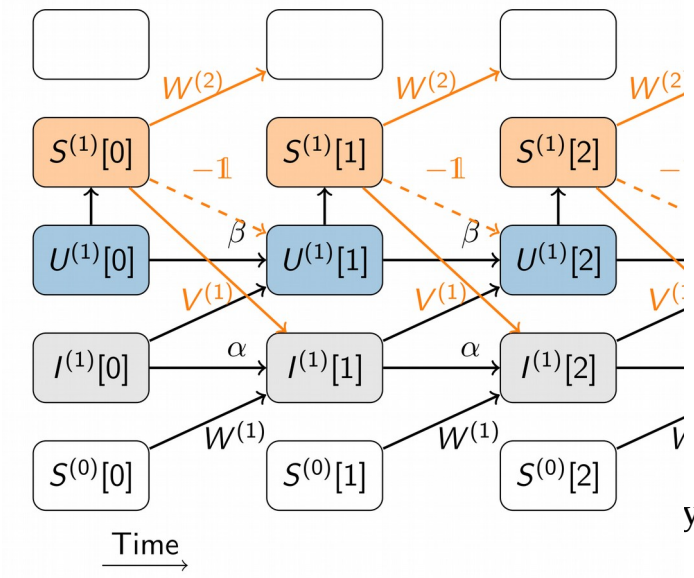
Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence



Neftci, Mostafa, & Zenke (in rev.)

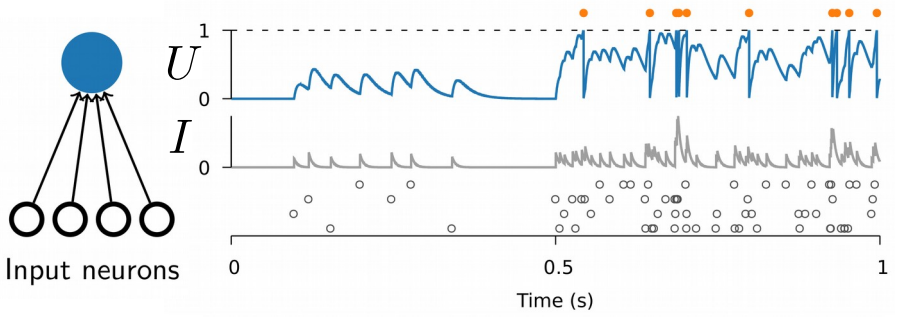
Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence

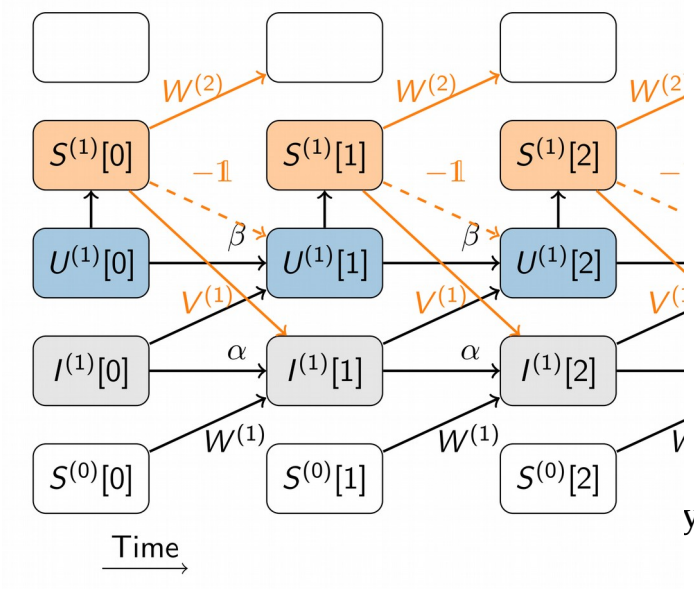




Neftci, Mostafa, & Zenke (in rev.)

Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence

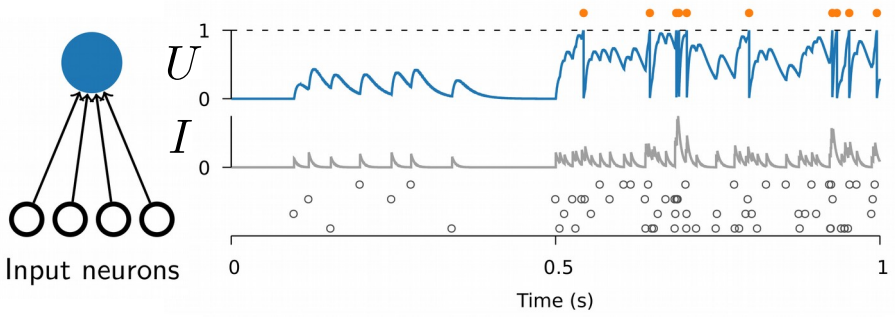


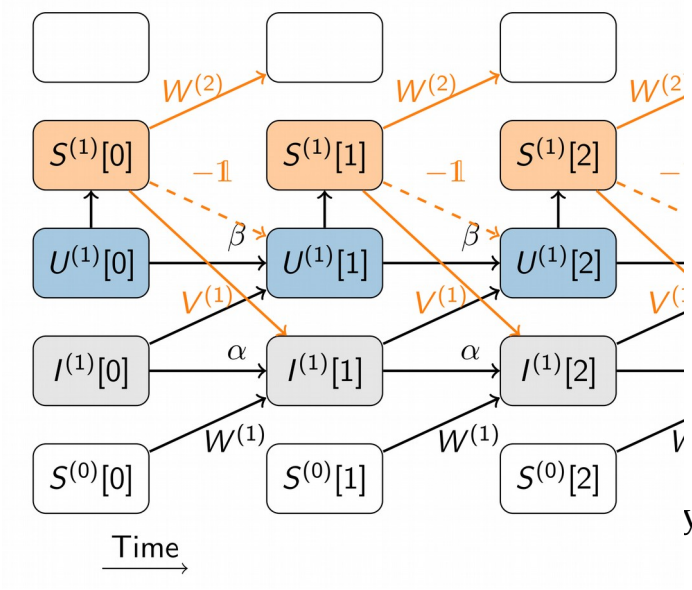


$$I_i^{(1)}[n+1] = \underbrace{\alpha I_i^{(1)}[n]}_{\text{exp. current decay}} + \underbrace{\sum_j W_{ij} S_j^{(0)}[n]}_{\text{feed-forward input}}$$

Neftci, Mostafa, & Zenke (in rev.)

Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence

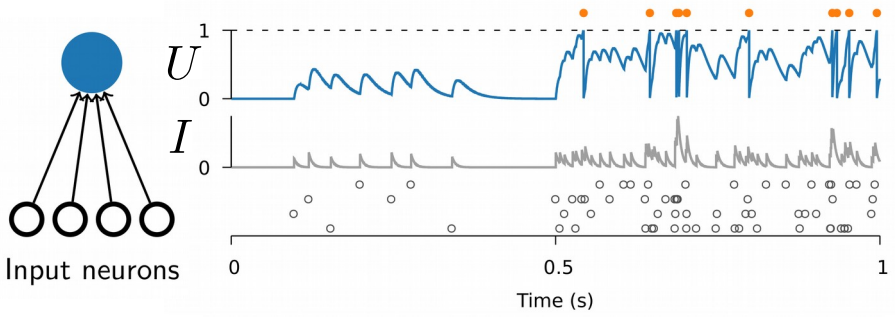


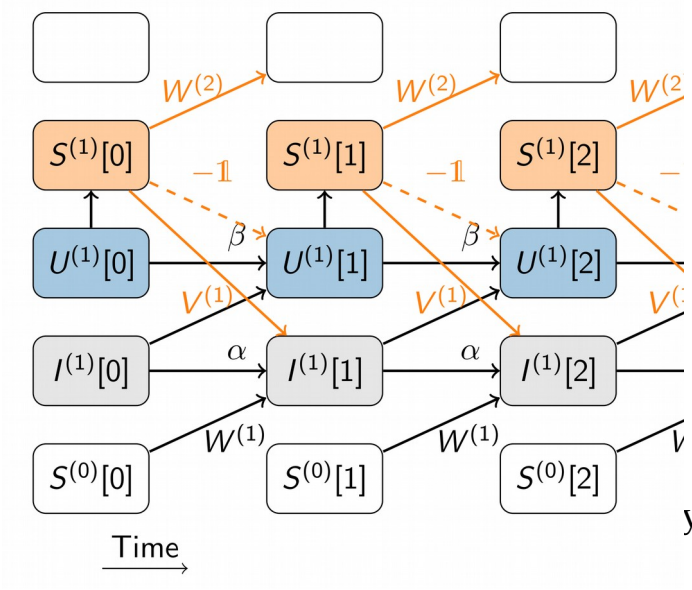


$$I_i^{(1)}[n+1] = \underbrace{\alpha I_i^{(1)}[n]}_{\text{exp. current decay}} + \underbrace{\sum_j W_{ij} S_j^{(0)}[n]}_{\text{feed-forward input}} + \underbrace{\sum_k V_{ik} S_k^{(1)}[n]}_{\text{recurrent input}}$$

Neftci, Mostafa, & Zenke (in rev.)

Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence



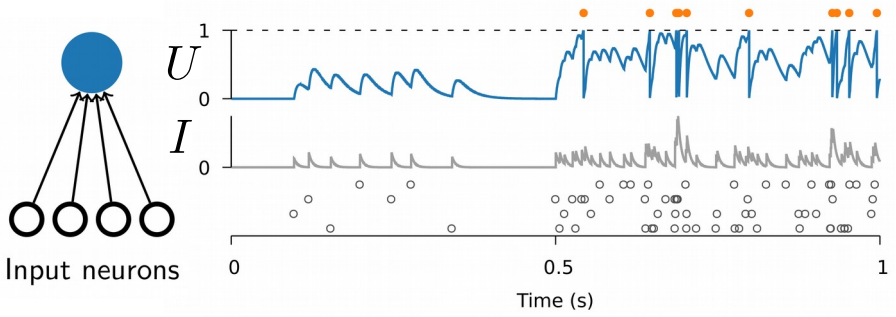


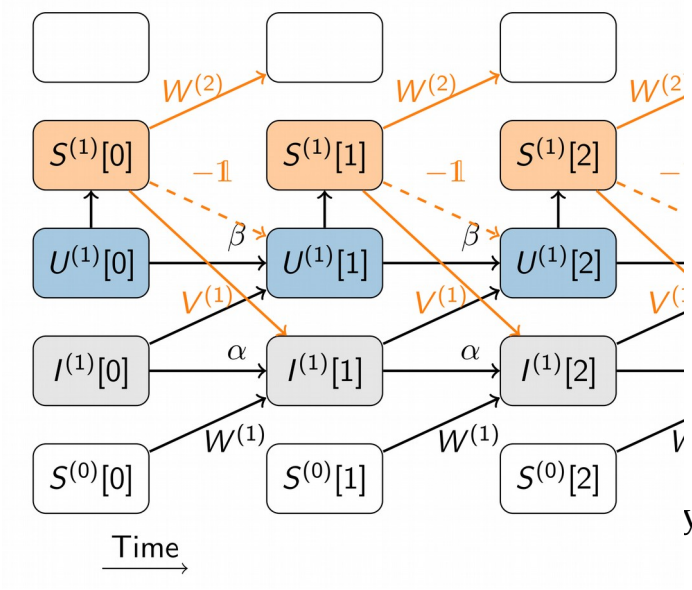
$$U_i^{(1)}[n+1] = \beta U_i^{(1)}[n] + I_i^{(1)}[n] - S_i[n]$$

$$I_i^{(1)}[n+1] = \underbrace{\alpha I_i^{(1)}[n]}_{\text{exp. current decay}} + \underbrace{\sum_j W_{ij} S_j^{(0)}[n]}_{\text{feed-forward input}} + \underbrace{\sum_k V_{ik} S_k^{(1)}[n]}_{\text{recurrent input}}$$

Neftci, Mostafa, & Zenke (in rev.)

Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence





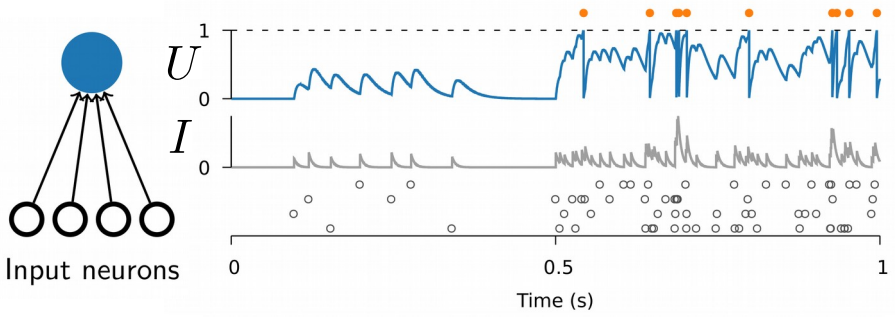
$$S_i^{(1)}[n] = \Theta\left(U_i^{(1)}[n] - \vartheta\right)$$

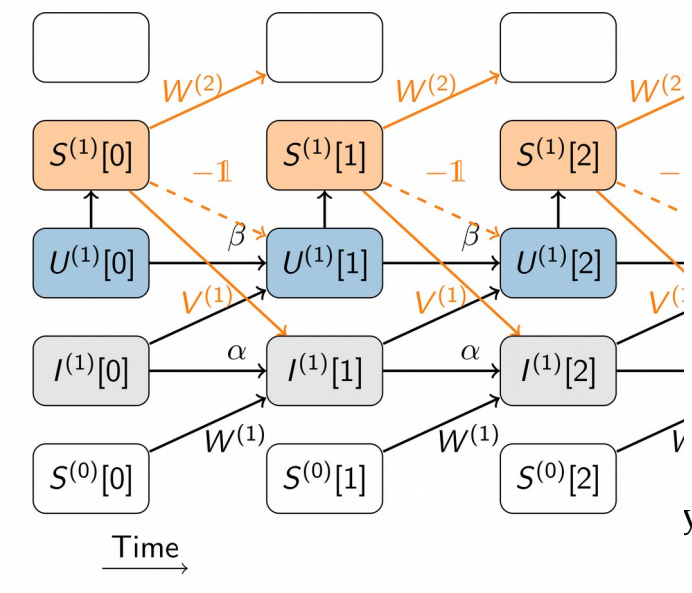
$$U_i^{(1)}[n+1] = \beta U_i^{(1)}[n] + I_i^{(1)}[n] - S_i[n]$$

$$I_i^{(1)}[n+1] = \underbrace{\alpha I_i^{(1)}[n]}_{\text{exp. current decay}} + \underbrace{\sum_j W_{ij} S_j^{(0)}[n]}_{\text{feed-forward input}} + \underbrace{\sum_k V_{ik} S_k^{(1)}[n]}_{\text{recurrent input}}$$

Neftci, Mostafa, & Zenke (in rev.)

Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence





$$S_i^{(1)}[n] = \Theta\left(U_i^{(1)}[n] - \vartheta\right)$$

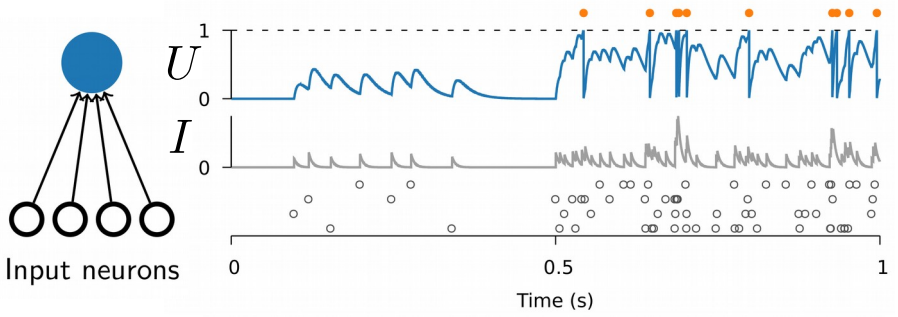
Surrogate gradient

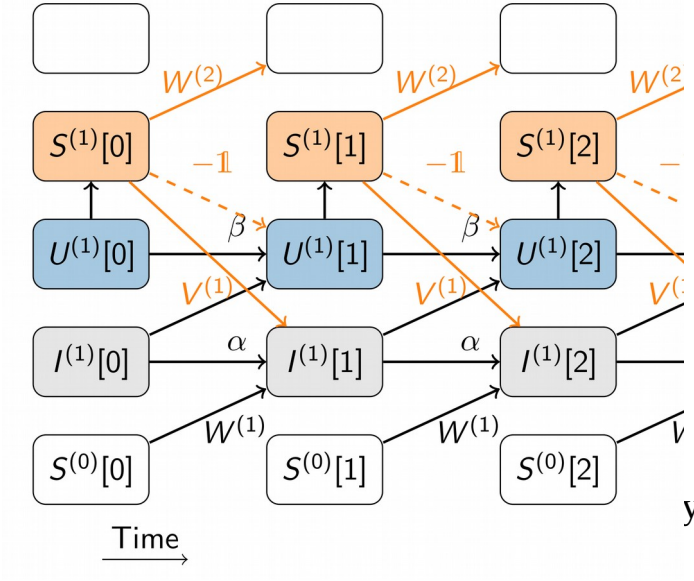
$$U_i^{(1)}[n+1] = \beta U_i^{(1)}[n] + I_i^{(1)}[n] - S_i[n]$$

$$I_i^{(1)}[n+1] = \underbrace{\alpha I_i^{(1)}[n]}_{\text{exp. current decay}} + \underbrace{\sum_j W_{ij} S_j^{(0)}[n]}_{\text{feed-forward input}} + \underbrace{\sum_k V_{ik} S_k^{(1)}[n]}_{\text{recurrent input}}$$

Neftci, Mostafa, & Zenke (in rev.)

Important insight: Spiking neural networks are binary RNNs with specific intrinsic recurrence





- Can be trained using BPTT or RTRL
- Several groups have realized this:
  - Esser, Merolla, Arthur, Cassidy, Appuswamy, Andreopoulos, Berg, McKinstry, Melano, Barch, et al. (2016)
  - Zenke & Ganguli (2018)
  - Huh & Sejnowski (2018)
  - Shrestha & Orchard (2018)
  - Bellec, Salaj, Subramoney, Legenstein, and Maass (arXiv)
  - Neftci, Mostafa, & Zenke (arXiv)

$$S_i^{(1)}[n] = \Theta\left(U_i^{(1)}[n] - \vartheta\right)$$

Surrogate gradient

$$U_i^{(1)}[n+1] = \beta U_i^{(1)}[n] + I_i^{(1)}[n] - S_i[n]$$

$$I_i^{(1)}[n+1] = \underbrace{\alpha I_i^{(1)}[n]}_{\text{exp. current decay}} + \underbrace{\sum_j W_{ij} S_j^{(0)}[n]}_{\text{feed-forward input}} + \underbrace{\sum_k V_{ik} S_k^{(1)}[n]}_{\text{recurrent input}}$$

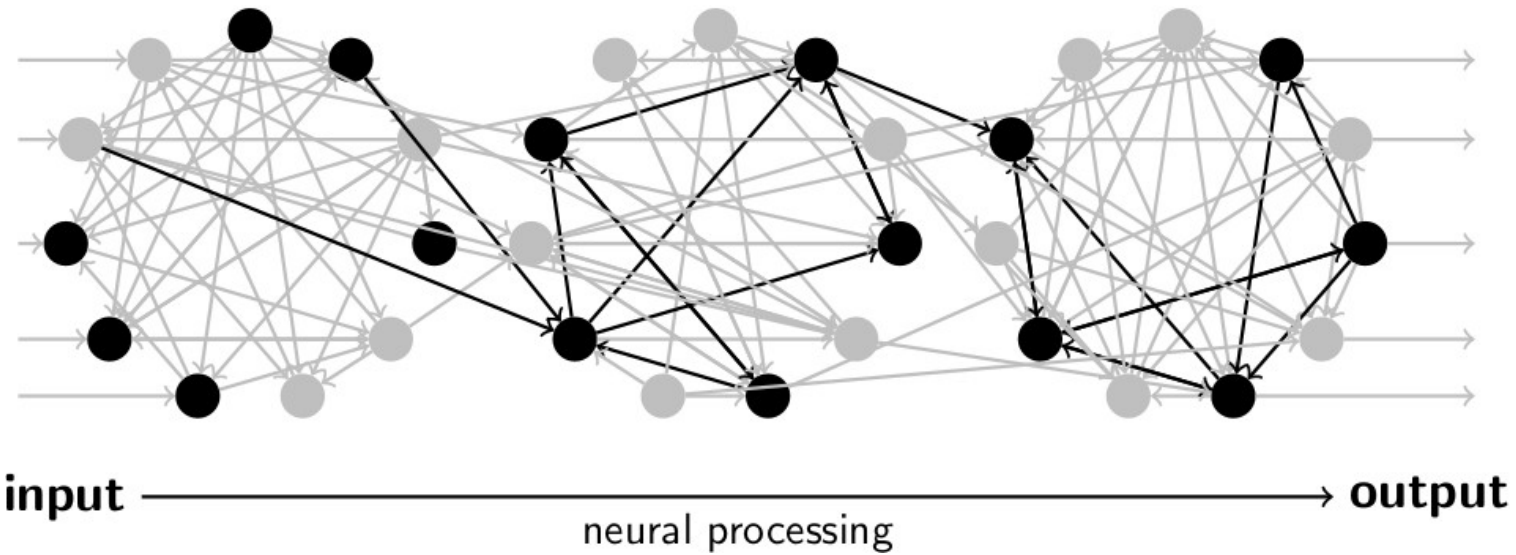
Neftci, Mostafa, & Zenke (in rev.)

Towards functional neural network models

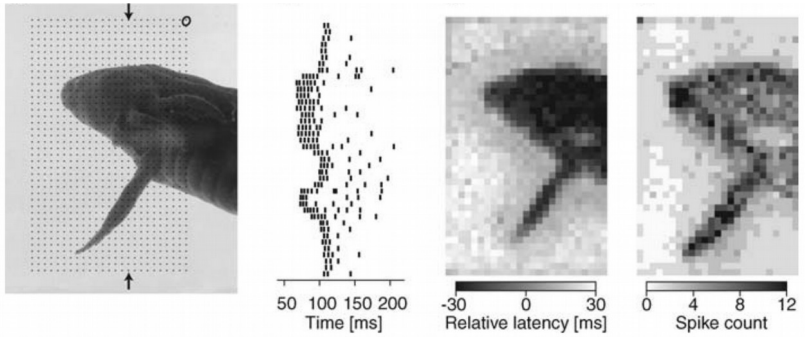
1) Input
(spatiotemporal)

3) Adjust weights
(surrogate gradients)

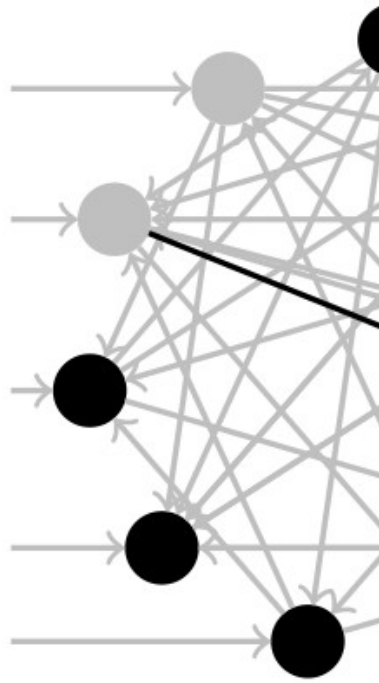
2) Output
(classification)



Input: Spatiotemporal spike patterns

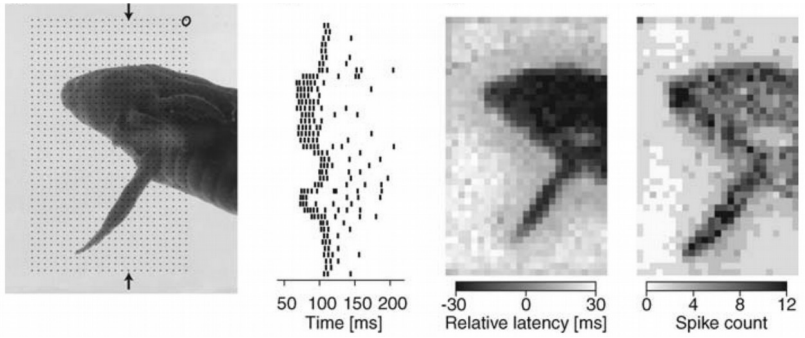


Gollisch & Meister (2008)

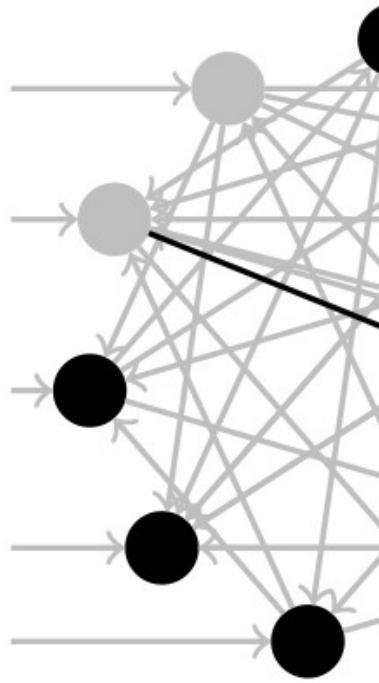


input

Input: Spatiotemporal spike patterns

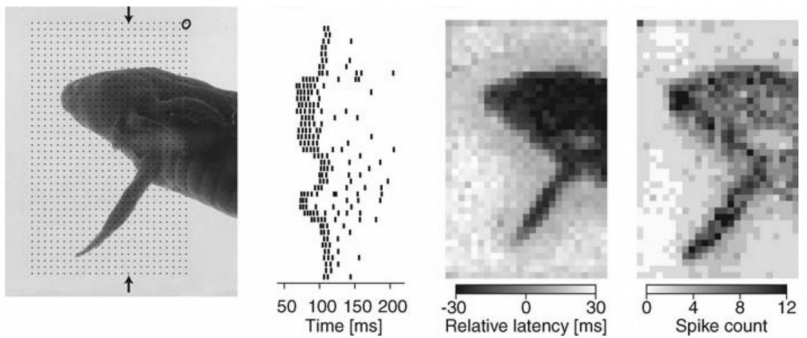


Gollisch & Meister (2008)

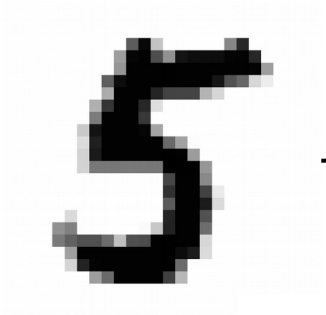


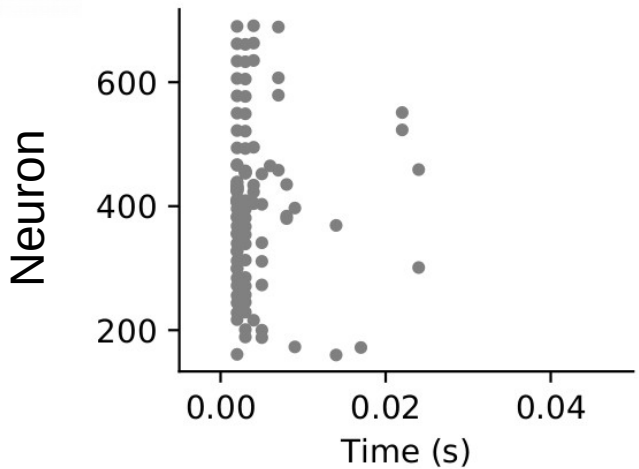
input

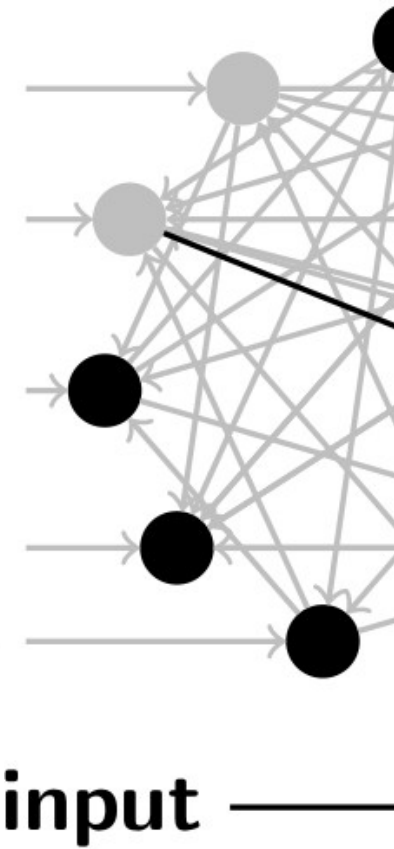
Input: Spatiotemporal spike patterns



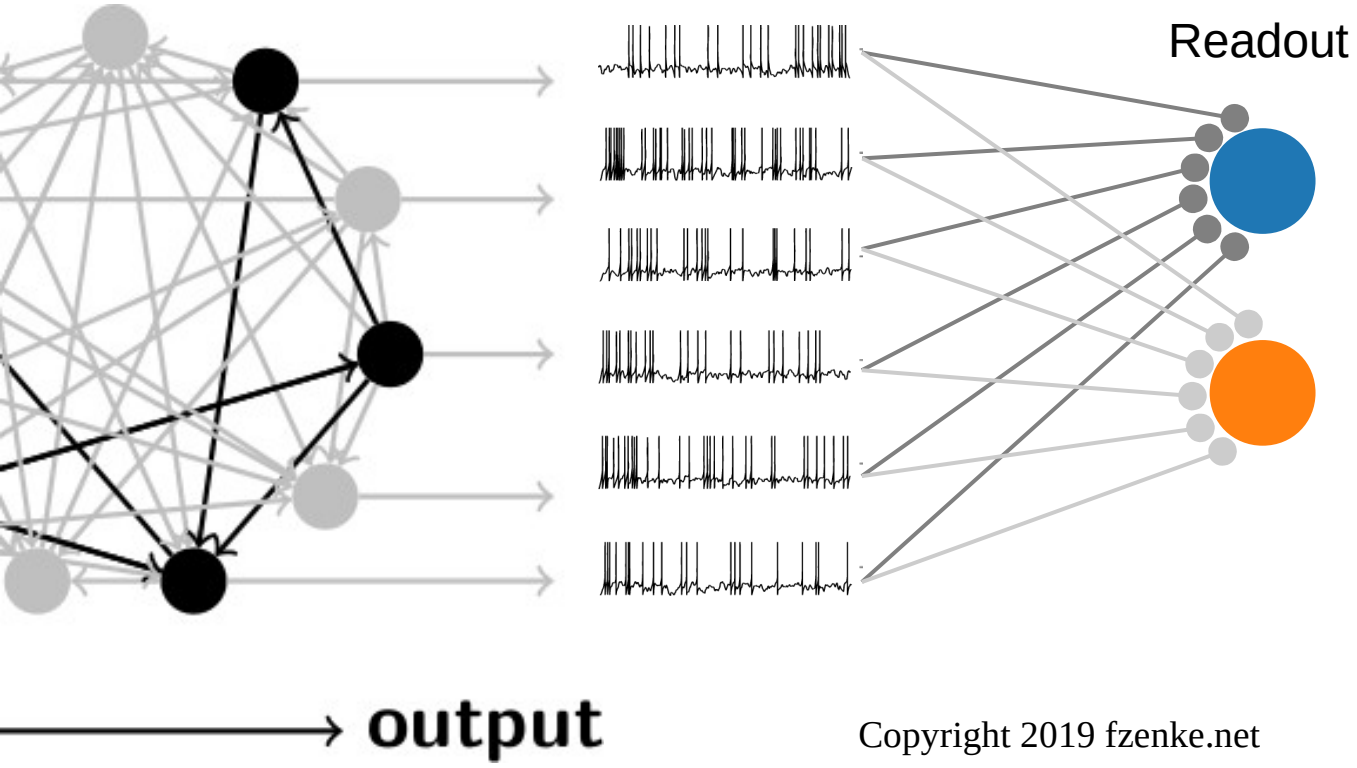
Gollisch & Meister (2008)



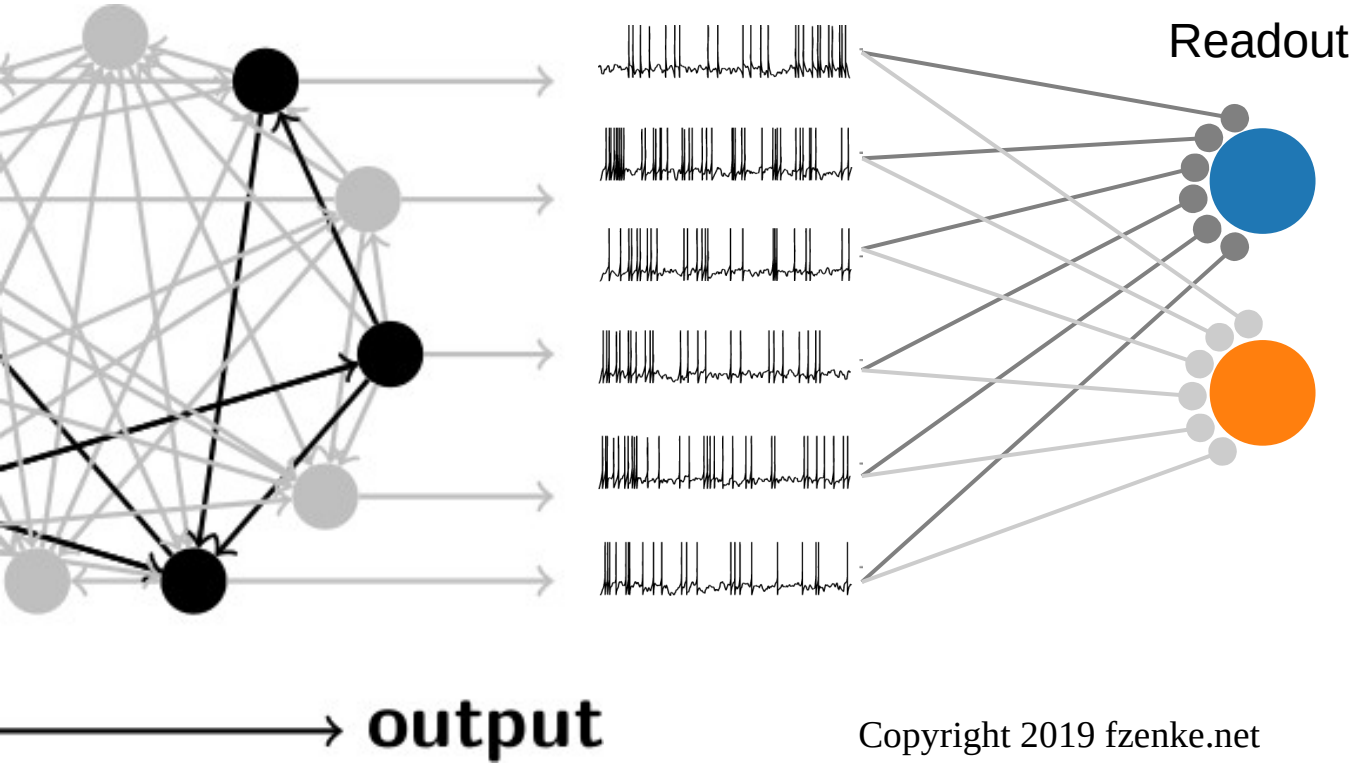




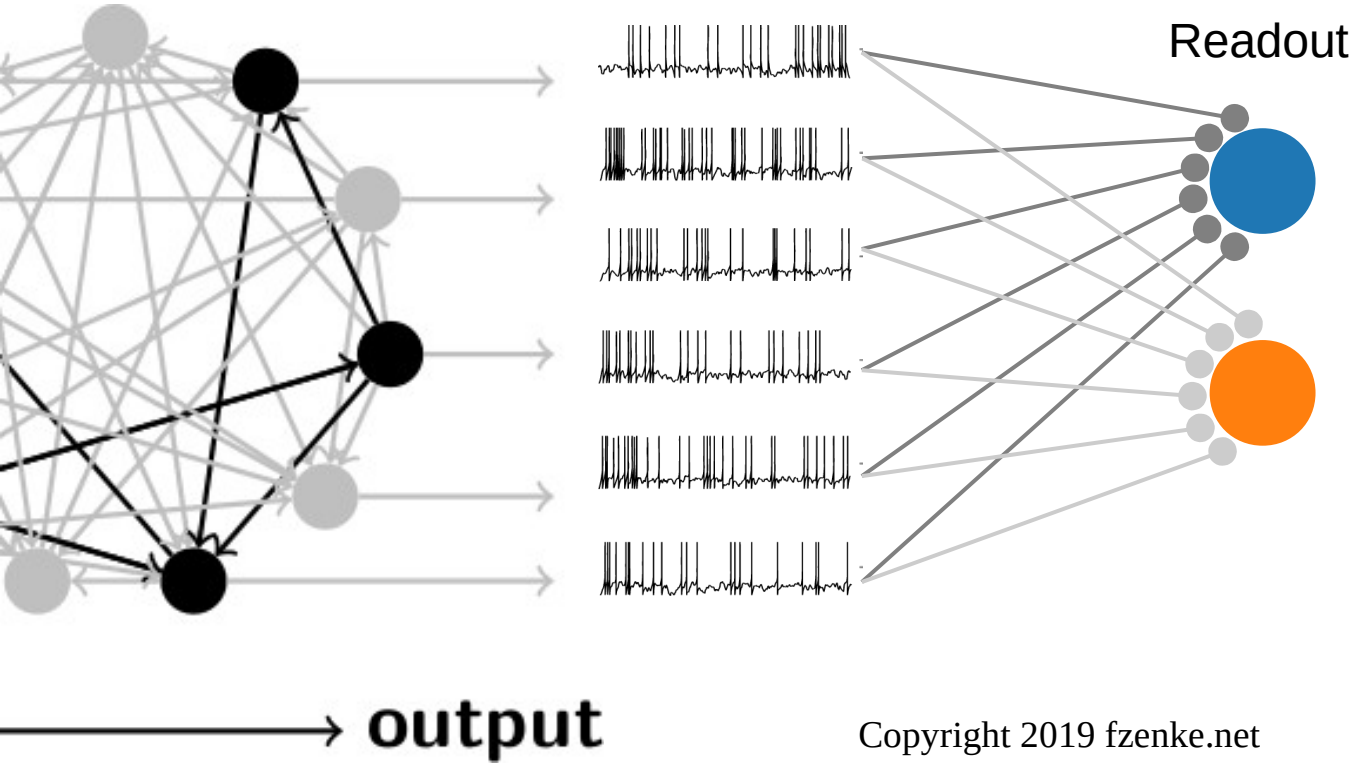
Output: Linear combination of filtered output spike trains

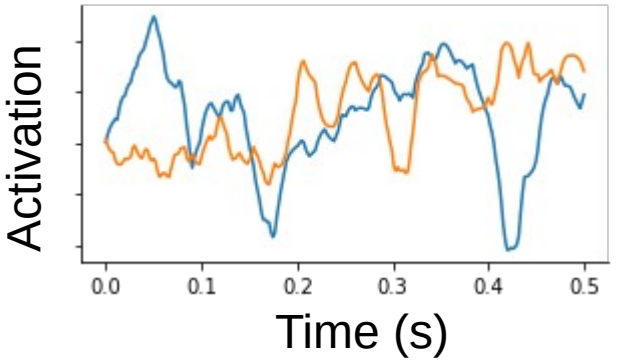


Output: Linear combination of filtered output spike trains

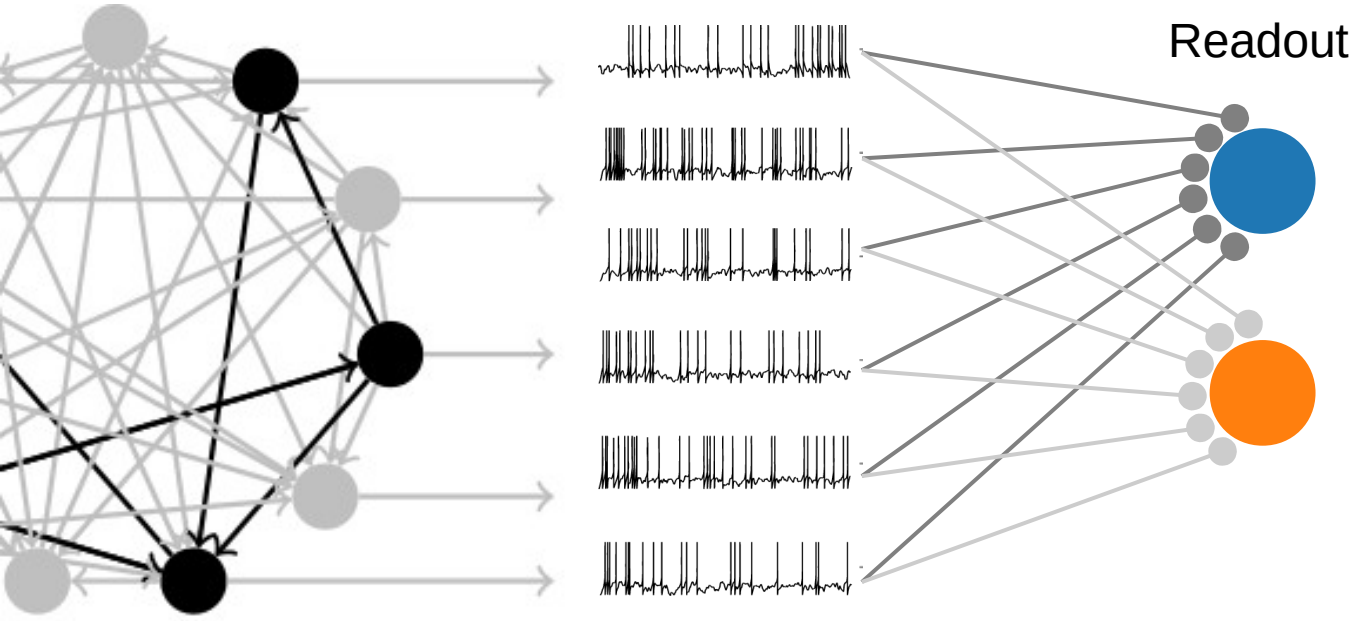


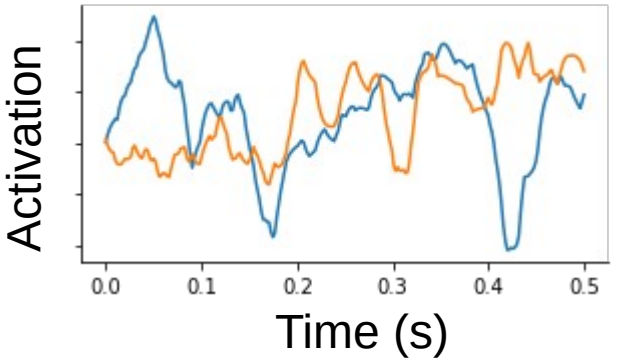
Output: Linear combination of filtered output spike trains





Output: Linear combination of filtered output spike trains

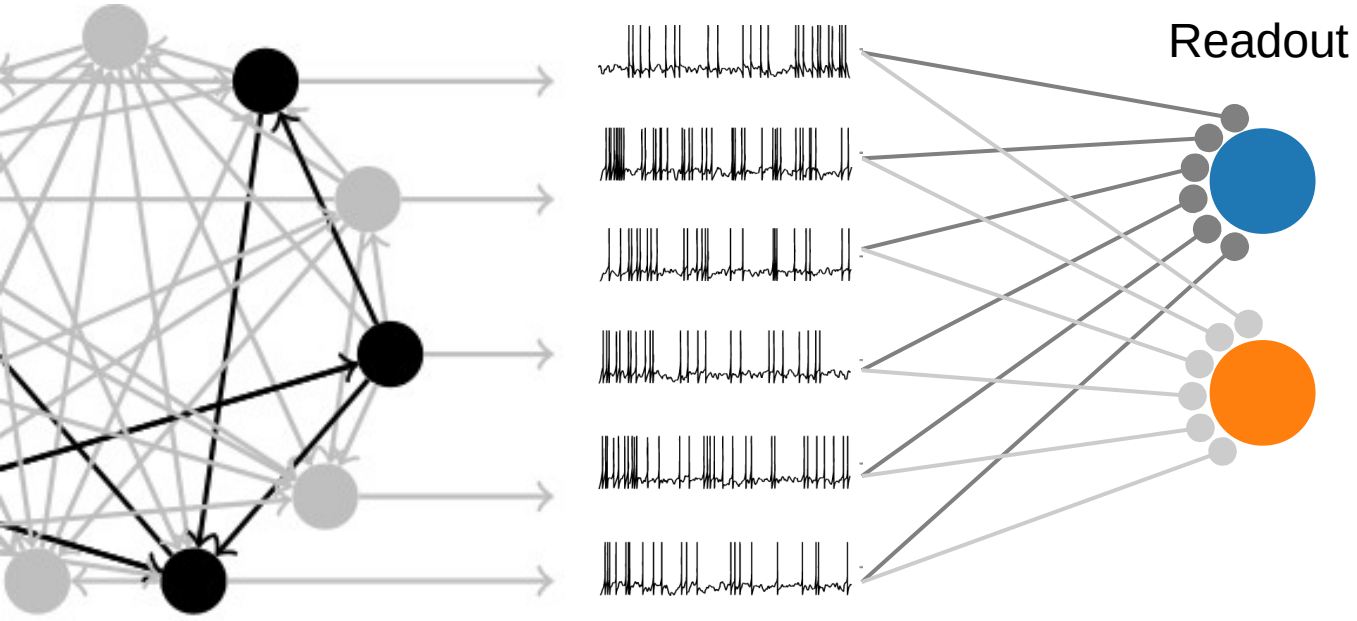


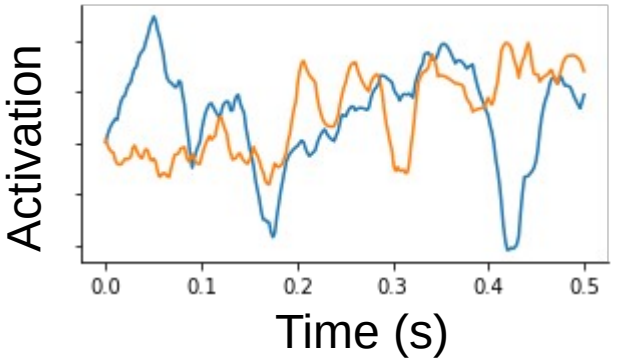


$$p_i = \underset{i}{\text{softmax}} \left(\max_t U_i(t) \right)$$

output

Output: Linear combination of filtered output spike trains



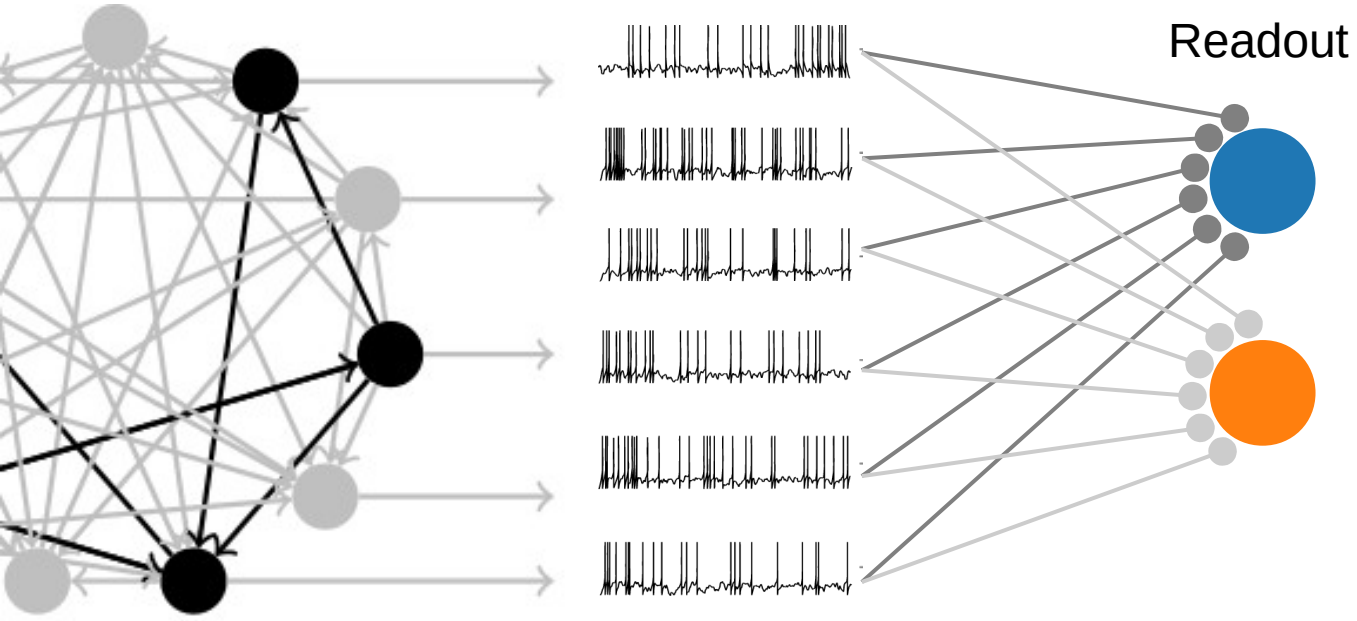


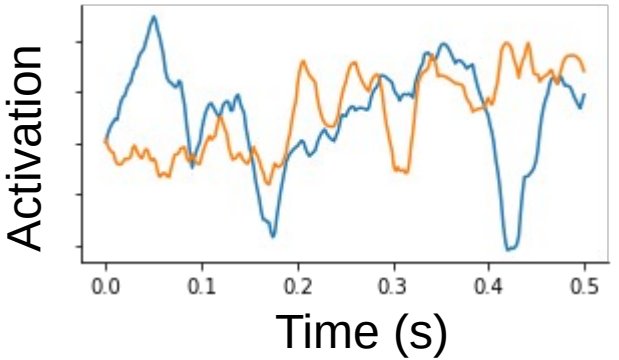
$$p_i = \underset{i}{\text{softmax}}\left(\max_t U_i(t)\right)$$

output

Max over time idea from Tempotron
Gütig & Sompolinsky (2006); Gütig (2016)

Output: Linear combination of filtered output spike trains



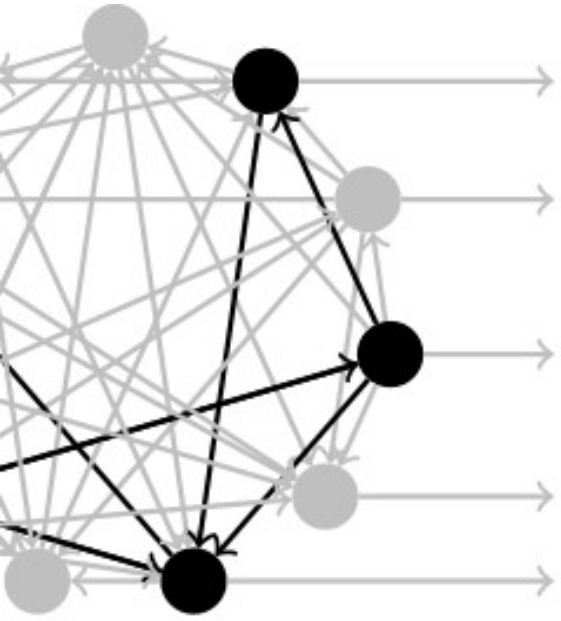


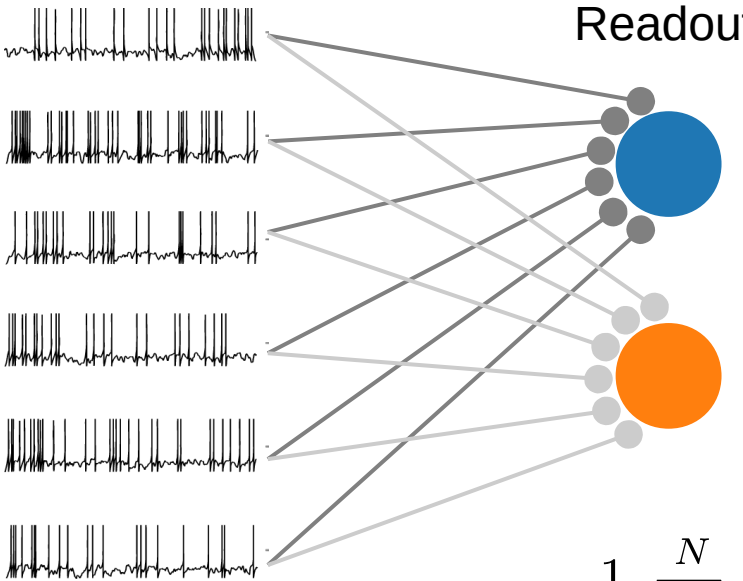
$$p_i = \underset{i}{\text{softmax}}\left(\max_t U_i(t)\right)$$

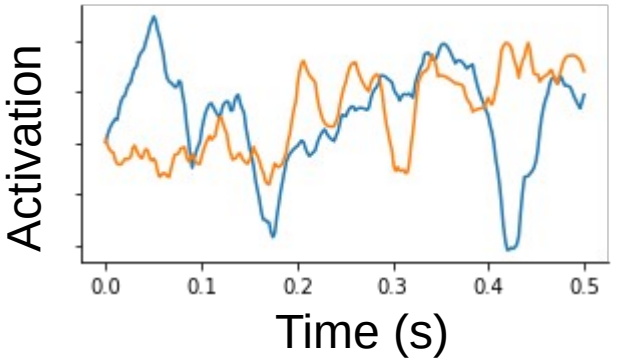
output

Max over time idea from Tempotron
Gütig & Sompolinsky (2006); Gütig (2016)

Output: Linear combination of filtered output spike trains







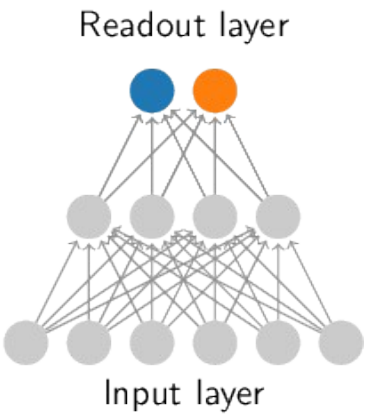
$$p_i = \underset{i}{\text{softmax}}\left(\max_t U_i(t)\right)$$

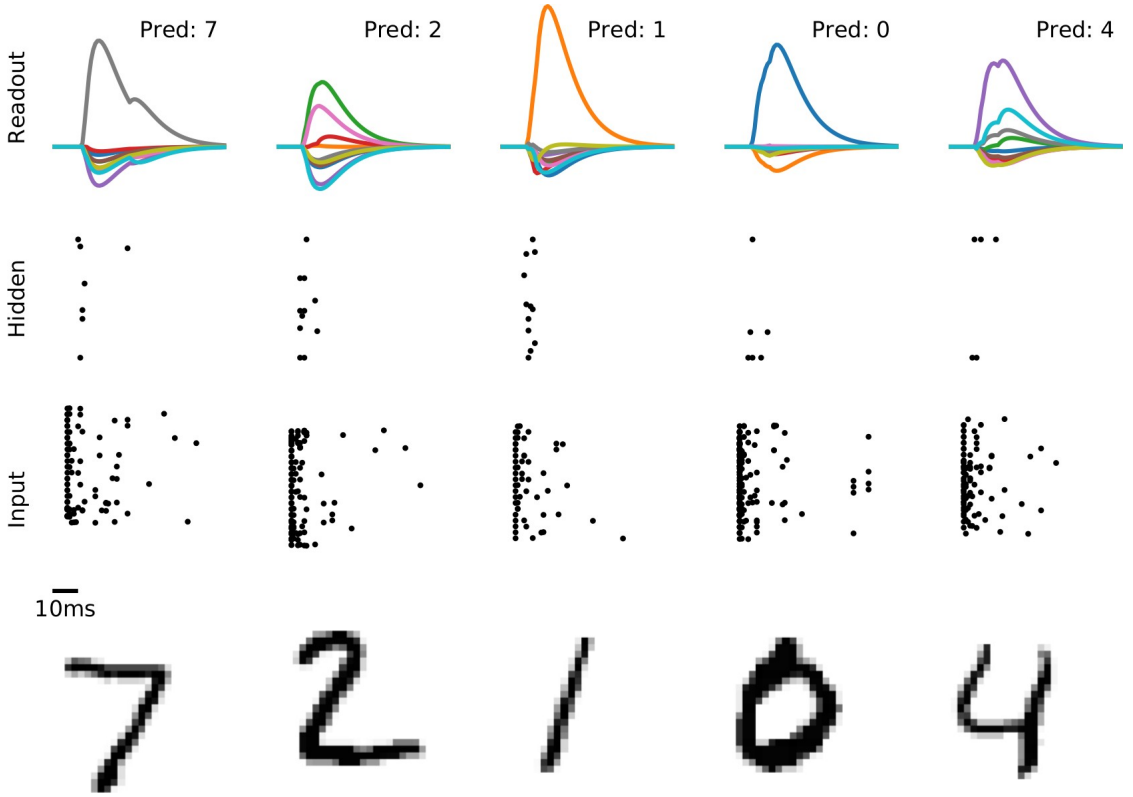
$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log p_n + (1-y_n)\log(1-p_n)\right]$$

→ **output**

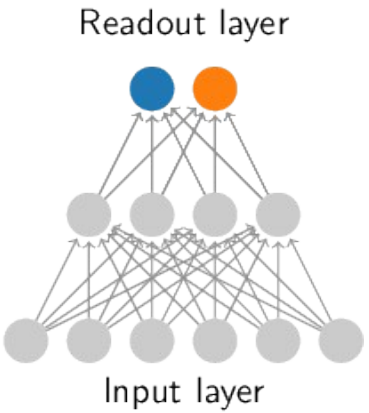
Max over time idea from Tempotron
Gütig & Sompolinsky (2006); Gütig (2016)

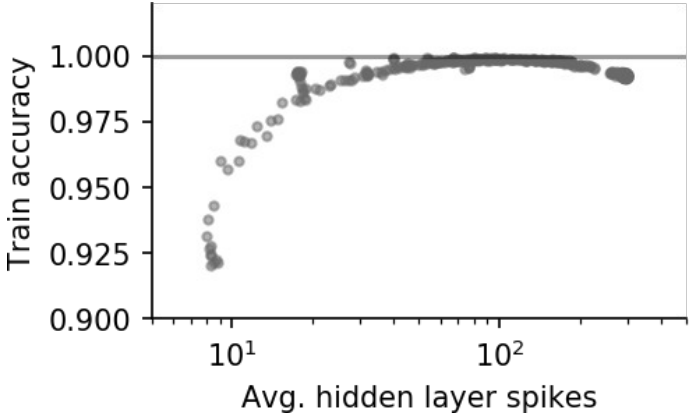
MNIST is solved with a handful of spikes

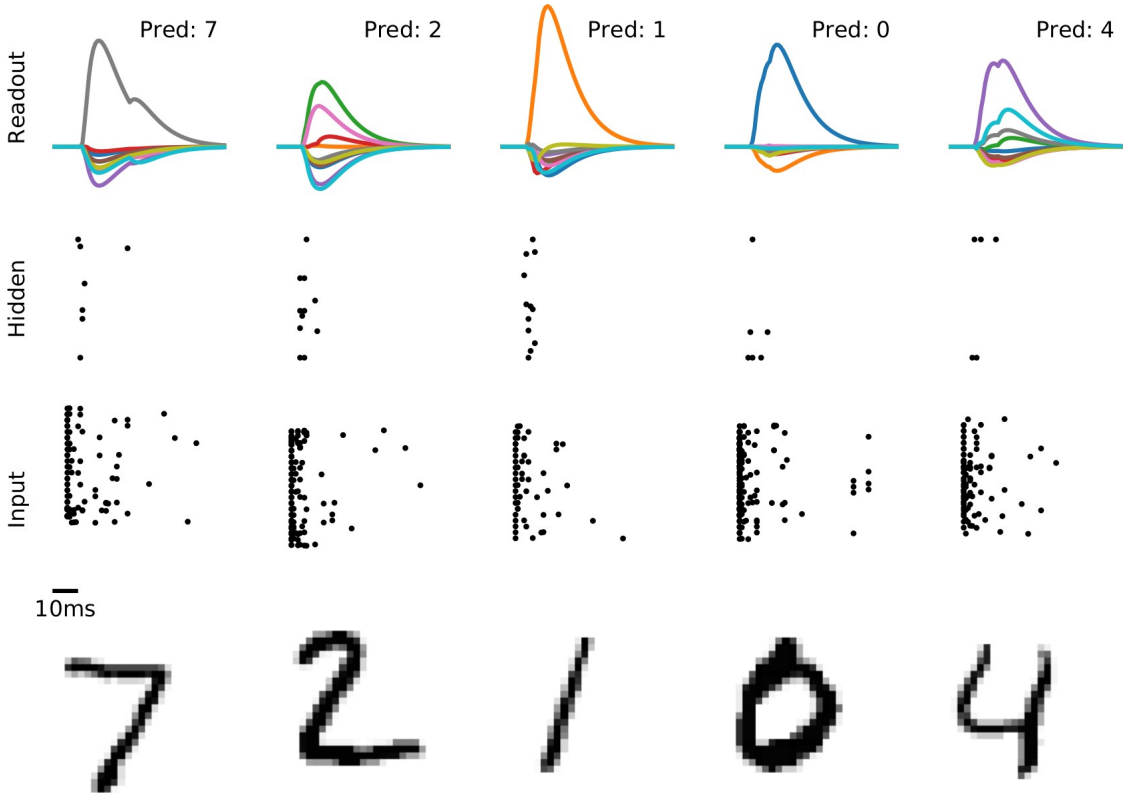




MNIST is solved with a handful of spikes

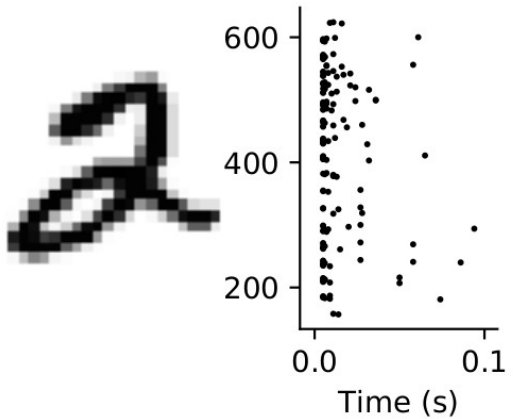


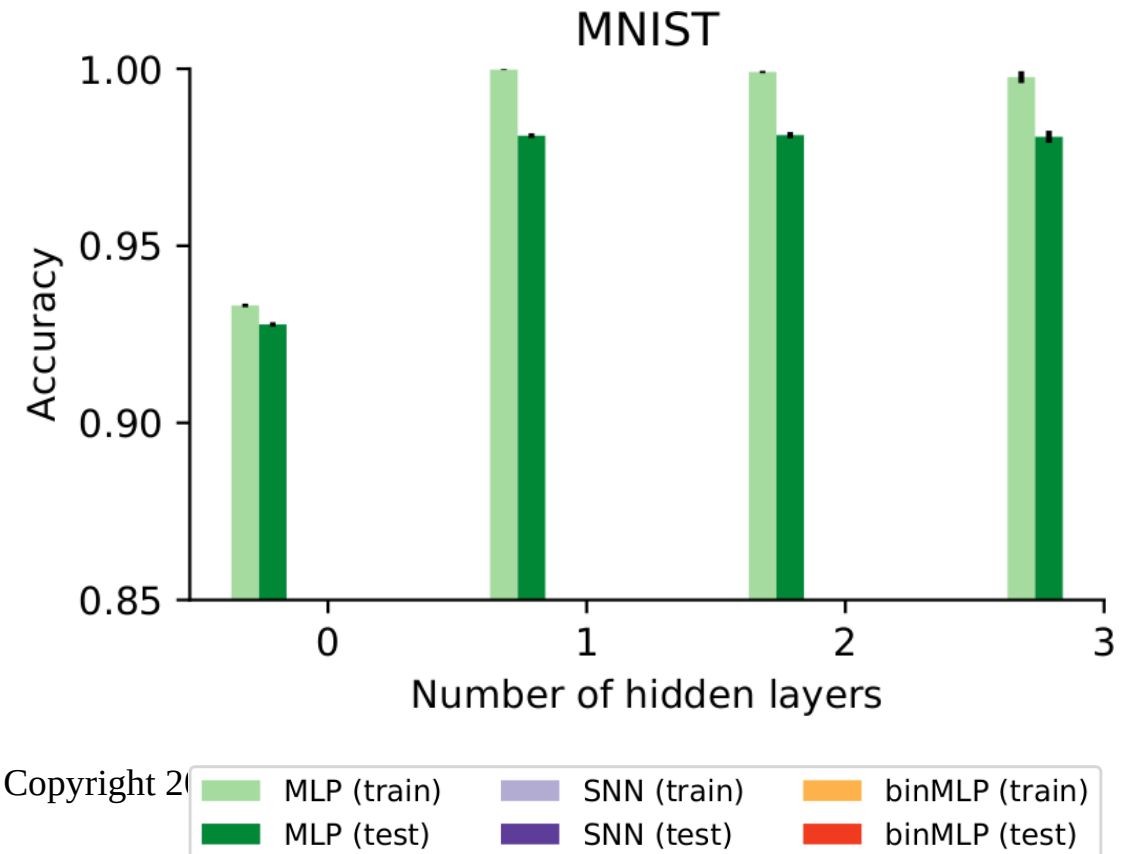




Benchmarks

MNIST
LeCun, Cortes & Burges (1998)

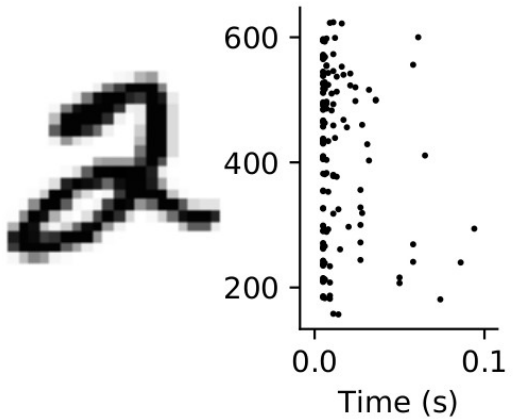


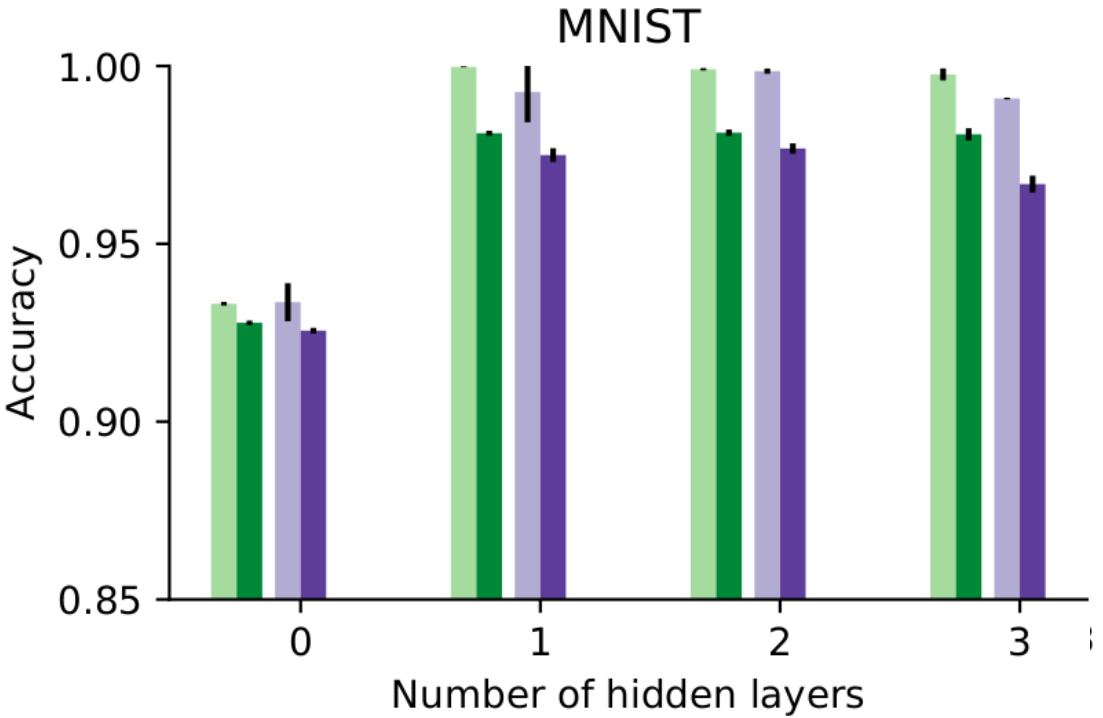


Zenke & Vogels (in prep.)

Benchmarks

MNIST
LeCun, Cortes & Burges (1998)

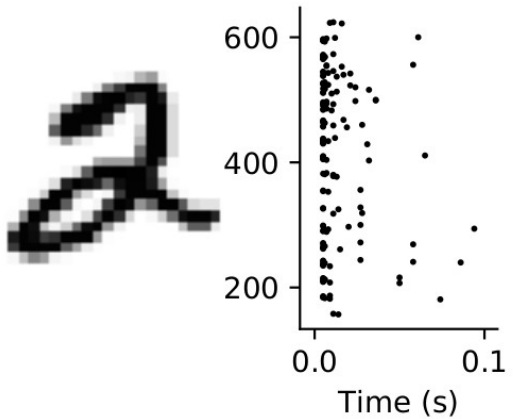


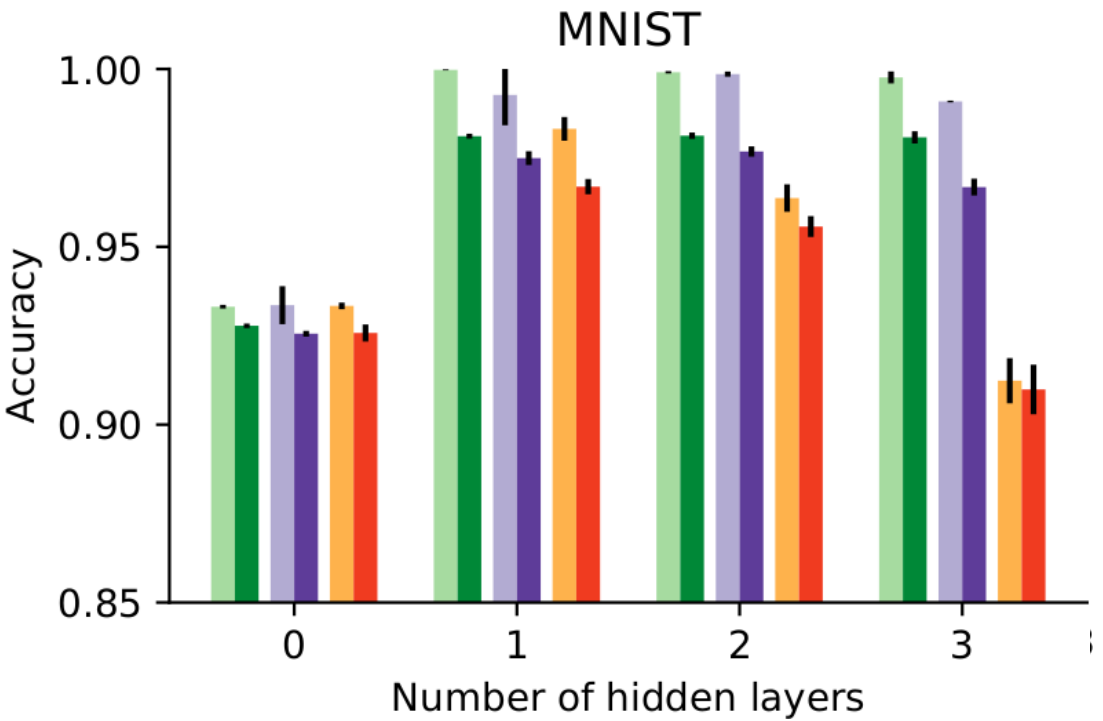


Zenke & Vogels (in prep.)

Benchmarks

MNIST
LeCun, Cortes & Burges (1998)

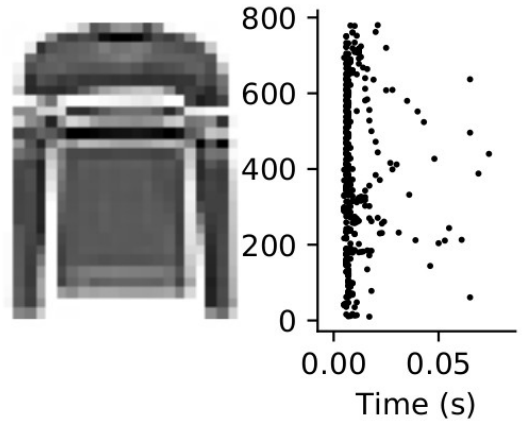


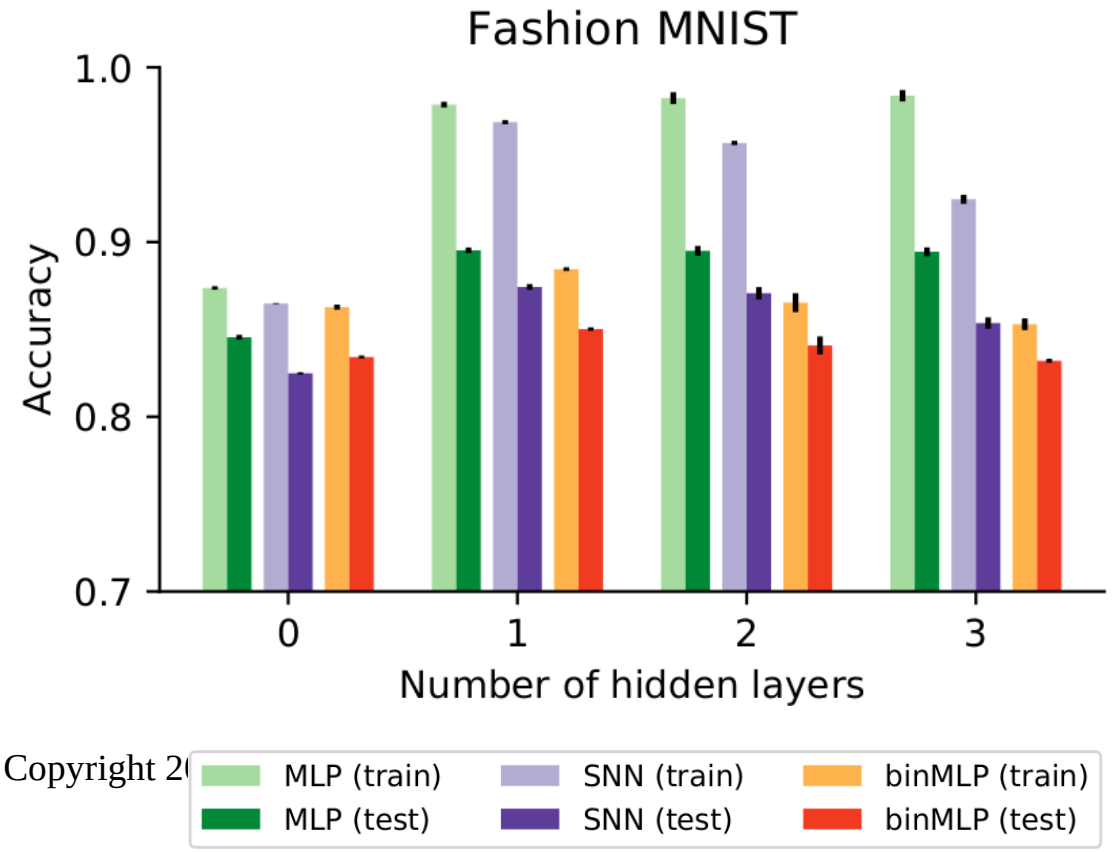


Zenke & Vogels (in prep.)

Benchmarks (2)

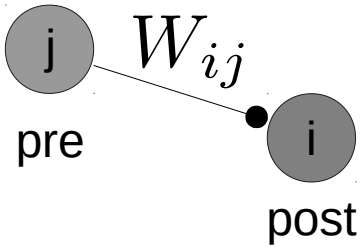
Fashion MNIST
Xiao, Rasul & Vollgraf (2017)





Zenke & Vogels (in prep.)

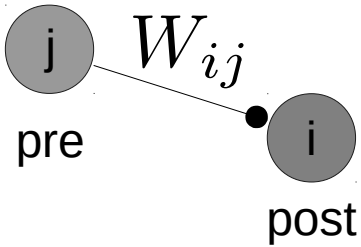
Surrogate gradient learning is robust to the choice of voltage nonlinearity



$$\Delta W_{ij} \propto (\mathrm{pre}_j) f(U_i^{\mathrm{post}}) (\mathrm{feedback}_i)$$

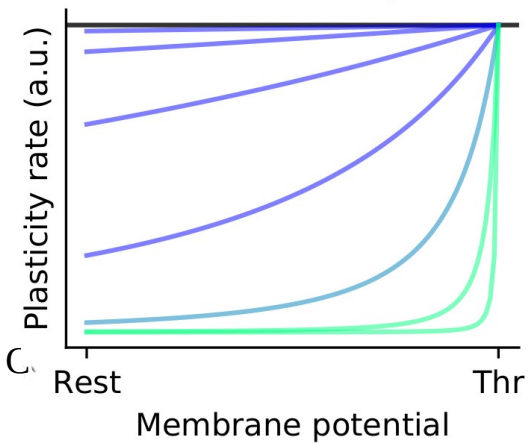
Zenke & Vogels (in prep.)

Surrogate gradient learning is robust to the choice of voltage nonlinearity

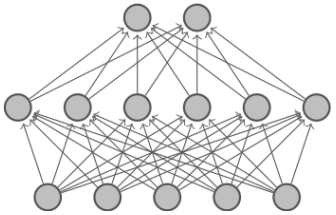


$$\Delta W_{ij} \propto (\mathrm{pre}_j) f(U_i^{\mathrm{post}}) (\mathrm{feedback}_i)$$

Nonlinearity

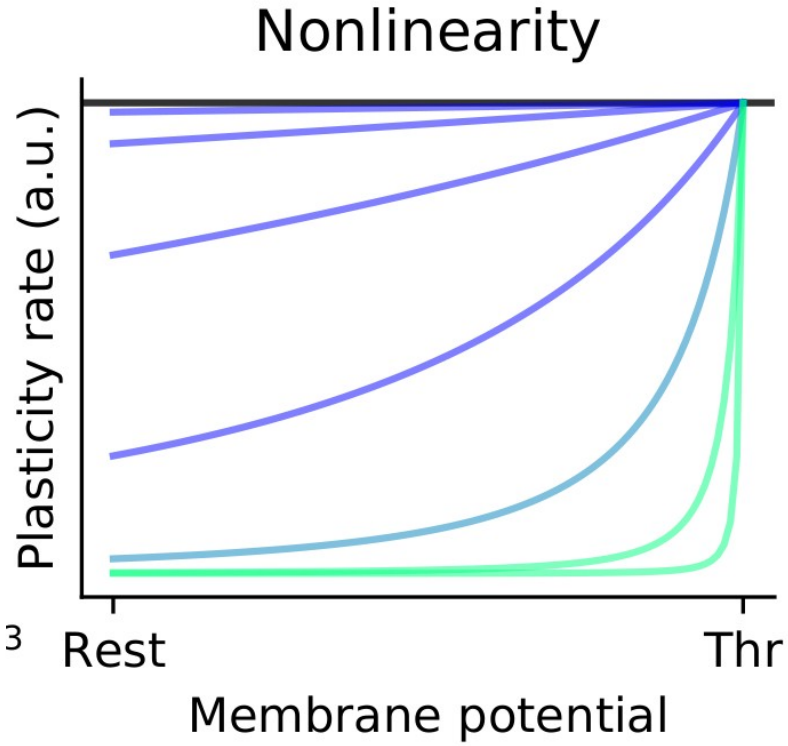


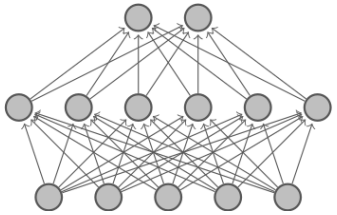
Zenke & Vogels (in prep.)



$$\Delta W_{ij} \propto (\mathrm{pre}_j) f(\mathrm{post}_i) (\mathrm{feedback}_i)$$

Zenke & Vogels (in prep.)

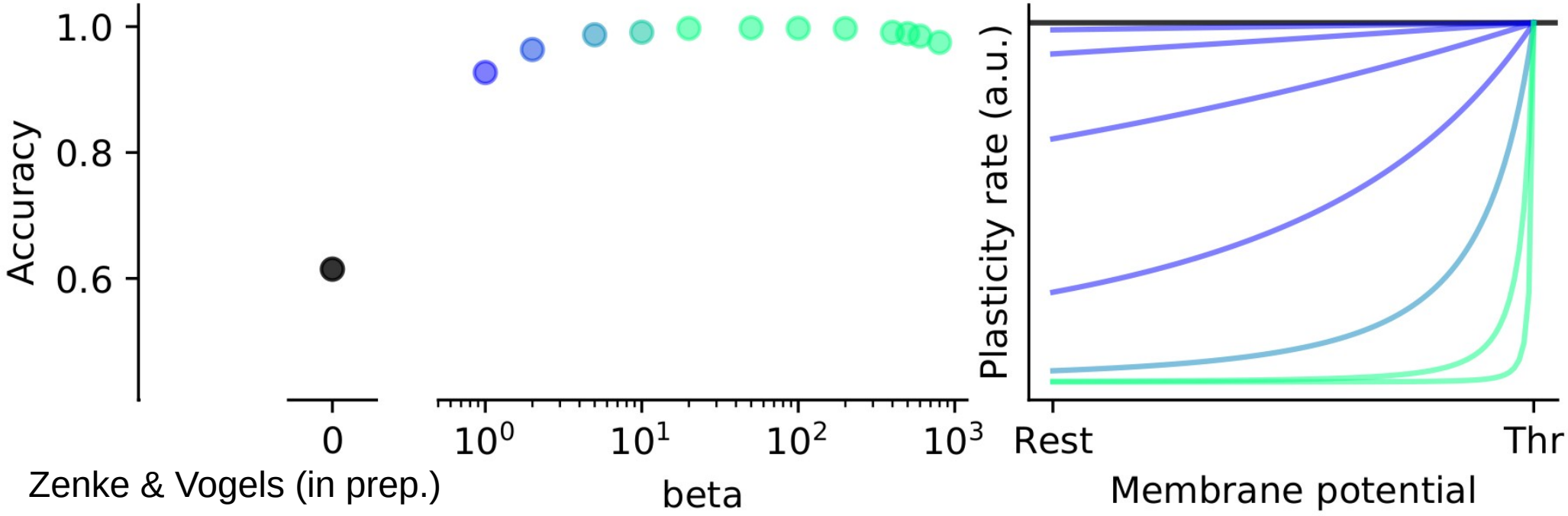




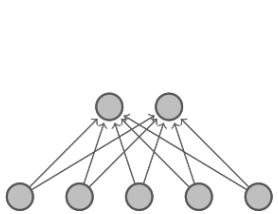
$$\Delta W_{ij} \propto (\mathrm{pre}_j)\boxed{f(\mathrm{post}_i)}(\mathrm{feedback}_i)$$

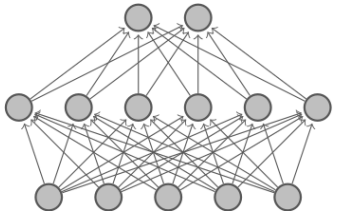
With hidden layer

Nonlinearity



Zenke & Vogels (in prep.)



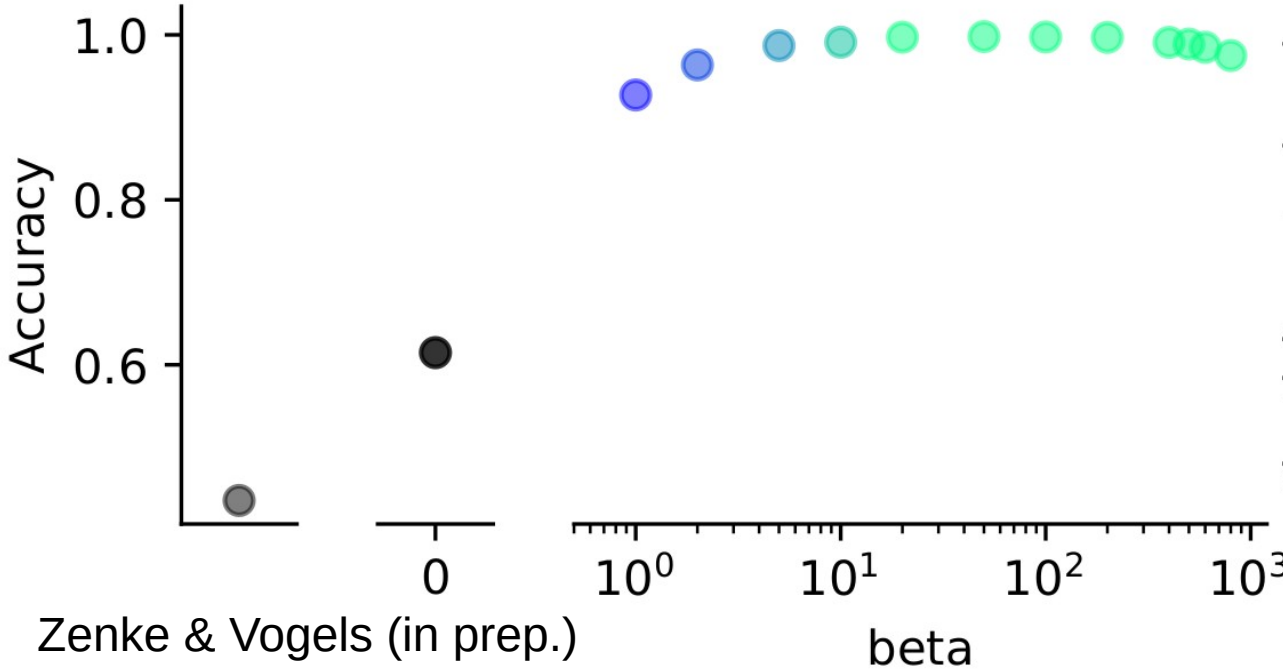


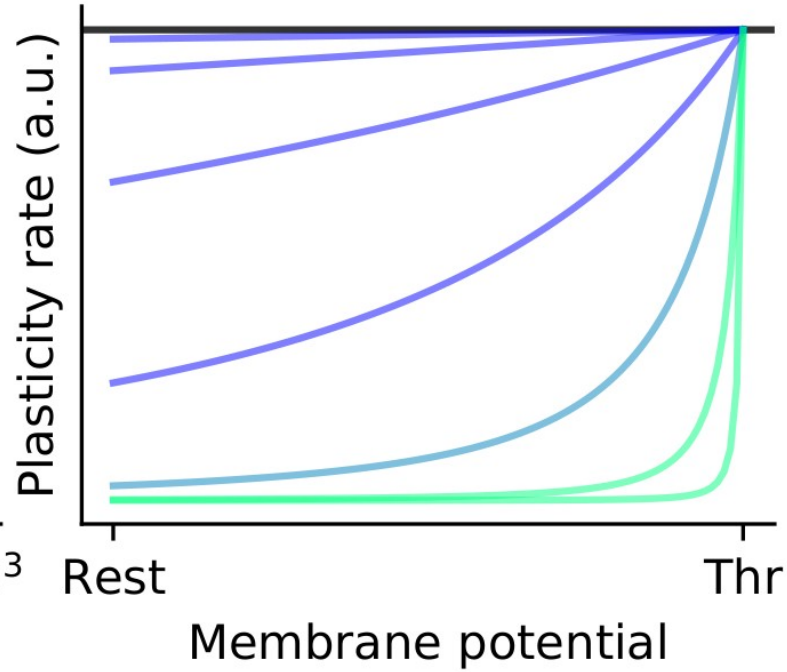
$$\Delta W_{ij} \propto (\mathrm{pre}_j)\boxed{f(\mathrm{post}_i)}(\mathrm{feedback}_i)$$

Shallow

With hidden layer

Nonlinearity

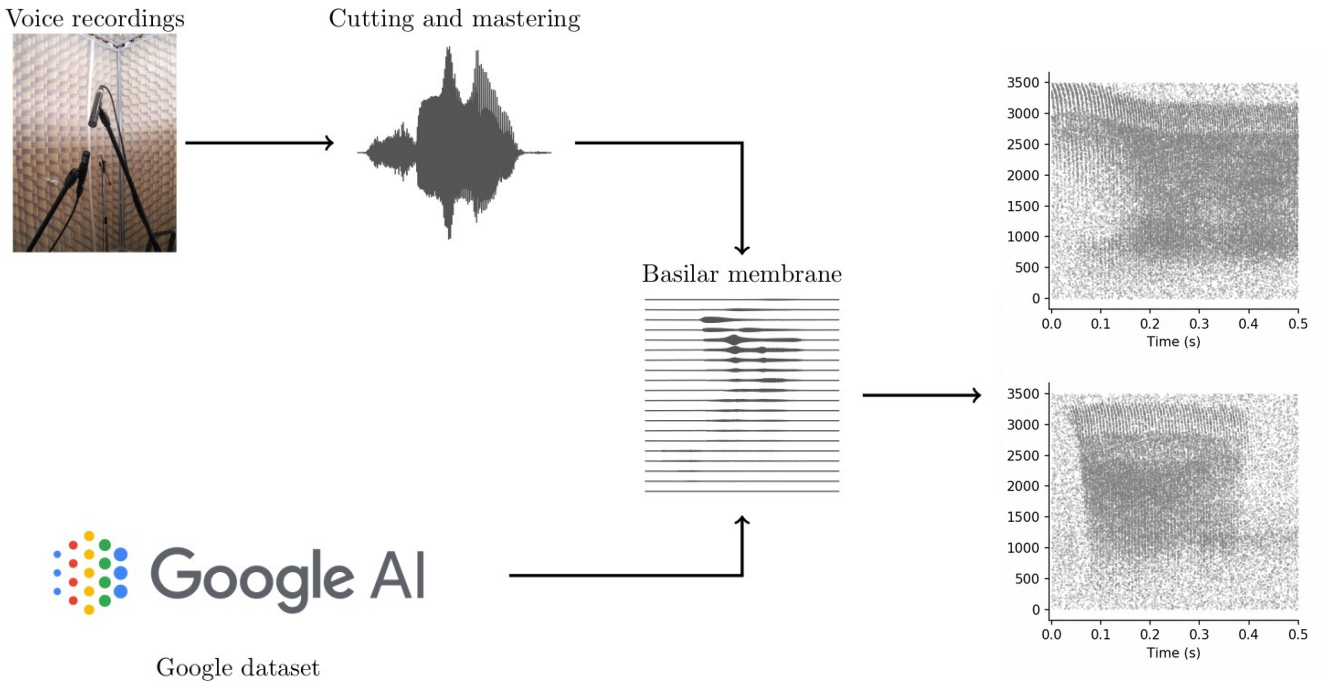




Zenke & Vogels (in prep.)

Benchmarks: The need for objective comparison of spiking networks

In collaboration with
Benjamin Cramer
Kirchhoff Institute of Physics
Uni Heidelberg



Spiking benchmark data sets

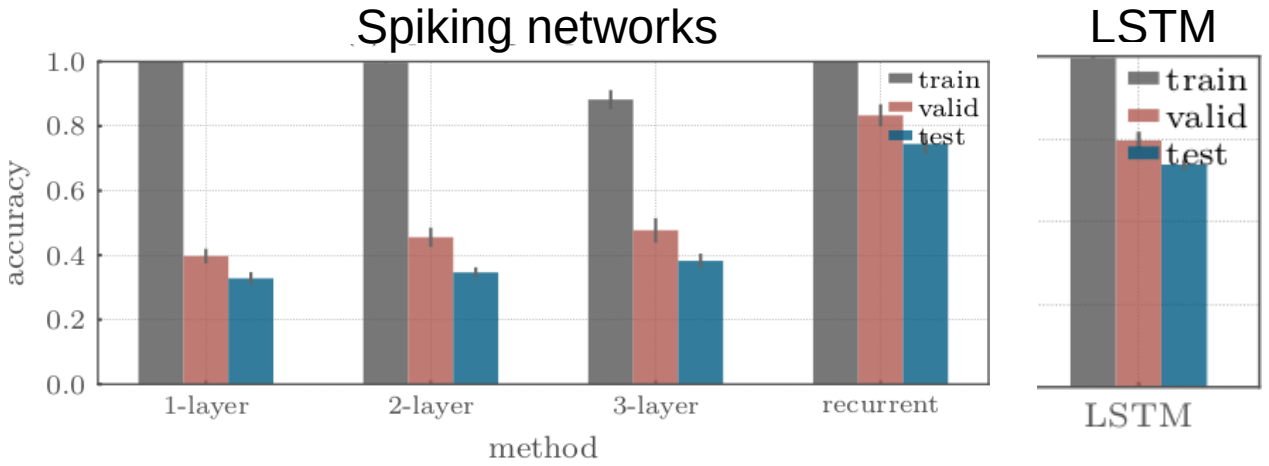
- Spoken digits & commands German/English
- More than 100k examples
- Spikes from cochlea model (3.5k channels)

Cramer, Stradmann, Schemmel & Zenke (in prep.)

Benchmark results

In collaboration with
Benjamin Cramer
Kirchhoff Institute of Physics
Uni Heidelberg

Preliminary



Cramer, Stradmann, Schemmel & Zenke (in prep.)

Summary & Outlook

Summary & Outlook

- Surrogate gradient learning in spiking neural networks
- A nonlinear voltage-dependent learning rule is crucial for learning with *hidden units*
- Temporal credit assignment through eligibility traces

Summary & Outlook

- Surrogate gradient learning in spiking neural networks
- A nonlinear voltage-dependent learning rule is crucial for learning with *hidden units*
- Temporal credit assignment through eligibility traces
- What next …?
  - Elucidate feedback channels
    (e.g. inhibitory microcircuits, neuromodulators)
  - Study unsupervised cost functions (e.g. prediction)

Thanks

Post-doc advisors

Stanford University

Surya Ganguli and the Gang

UNIVERSITY OF OXFORD

Tim Vogels and Group

Review/Tutorial : Neftci, Mostafa, & Zenke (2019). ArXiv

Emre Neftci, UC Irvine

Funding:

wellcome

Code & Tutorials:
fzenke.net

Artwork:
K. Yadava (kyadava.net)