Deep learning in the brain

Deep learning summer school
Toronto 2018

Blake Aaron Richards
Learning in Neural Circuits Laboratory
Department of Biological Sciences
University of Toronto Scarborough

LiNC LAB
linclab.org

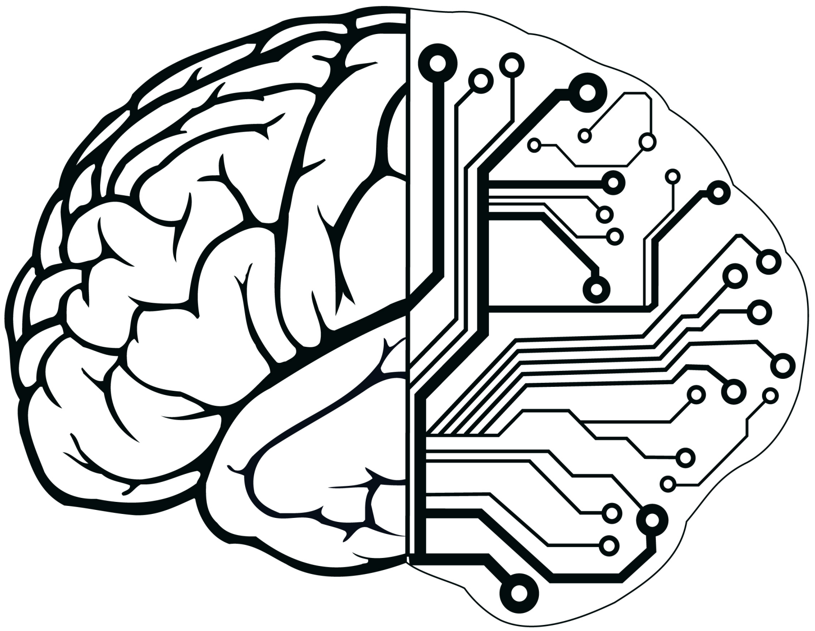
UNIVERSITY OF TORONTO SCARBOROUGH

1. Why deep learning is not just for AI

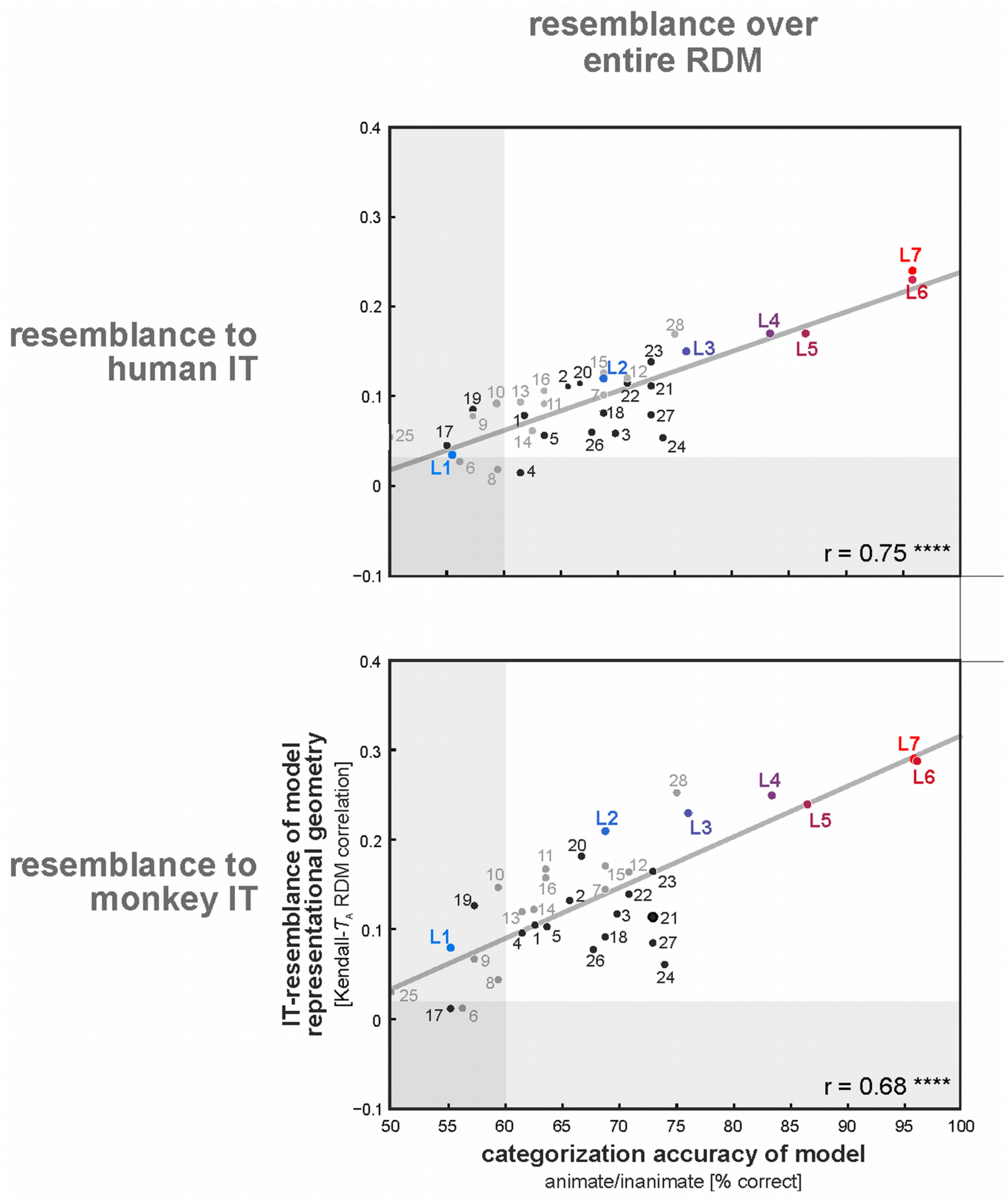
The recent success of deep learning in artificial intelligence (AI) means that most people associate it exclusively with AI

But, one of the goals of (some) deep learning research has always been to understand how our own brains work

In this session, I'm going to give you a brief overview on current research into the challenges associated with connecting deep learning and neuroscience, and some of the work in this area



1. Why deep learning is not just for AI



A little more motivation:

Deep learning models are a better fit to neocortical representations than models developed by neuroscientists!

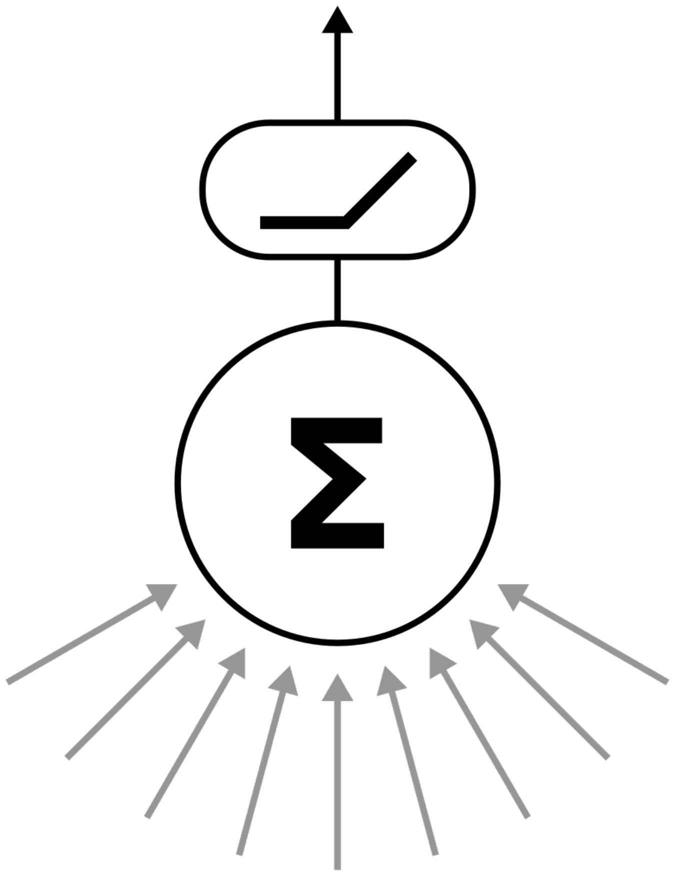
This would suggest that our neocortex is doing something akin to deep learning

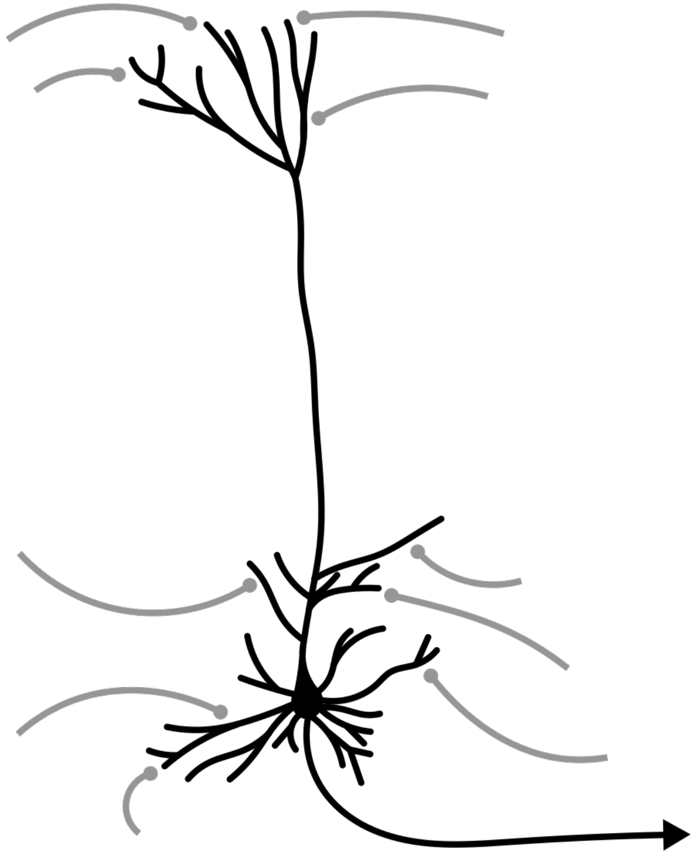
Khaligh-Razavi and Kriegeskorte (2014) PloS Comp. Biol. 10(11): e1003915

1. Why deep learning is not just for AI

UNIVERSITY OF TORONTO SCARBOROUGH

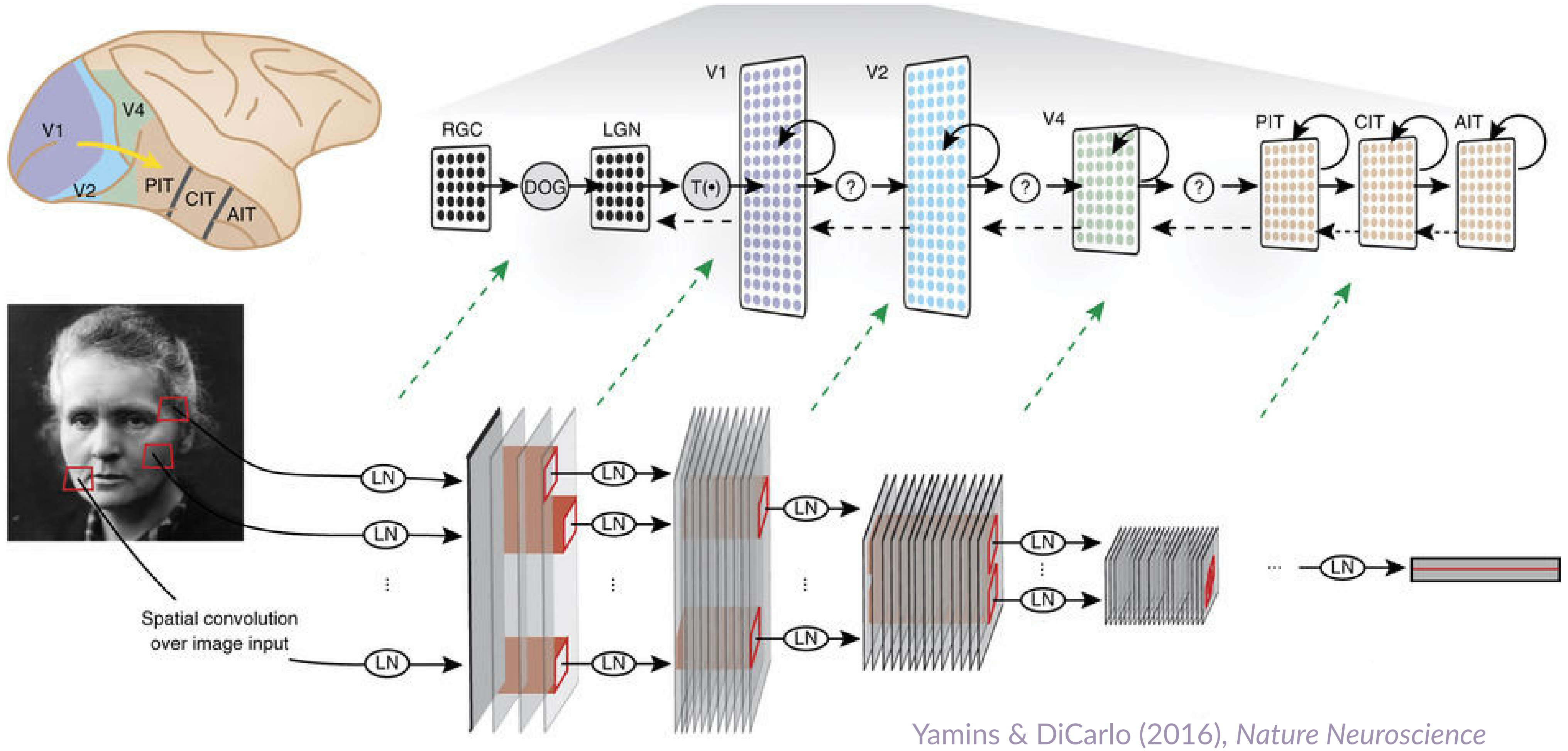
At an abstract level, deep neural networks operate with some similar principals to the real brain (though there are some important differences!)





1. Why deep learning is not just for AI

At an abstract level, deep neural networks operate with some similar principals to the real brain (though there are some important differences!)

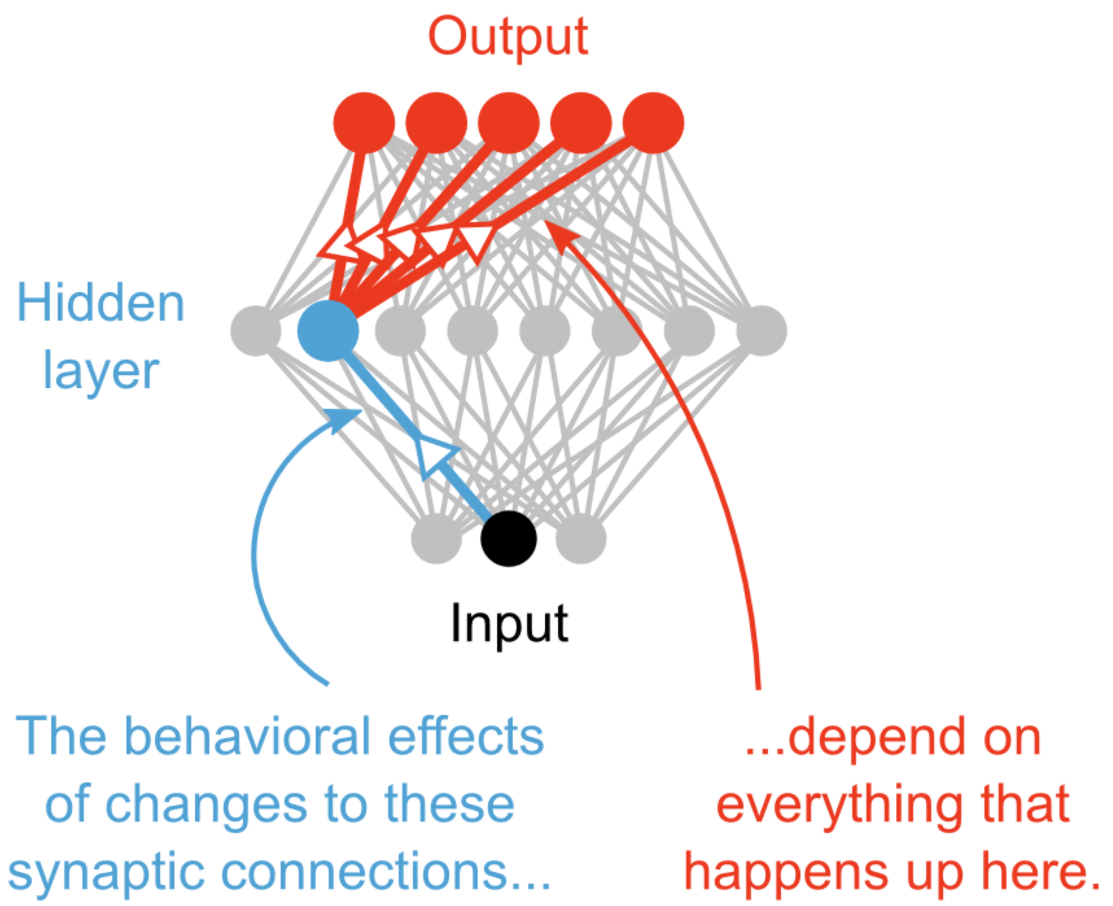


Yamins & DiCarlo (2016), *Nature Neuroscience*

2. Foundations: what is required for deep learning?

The key feature of deep learning is the ability to engage in **end-to-end optimization** in a many-layered network

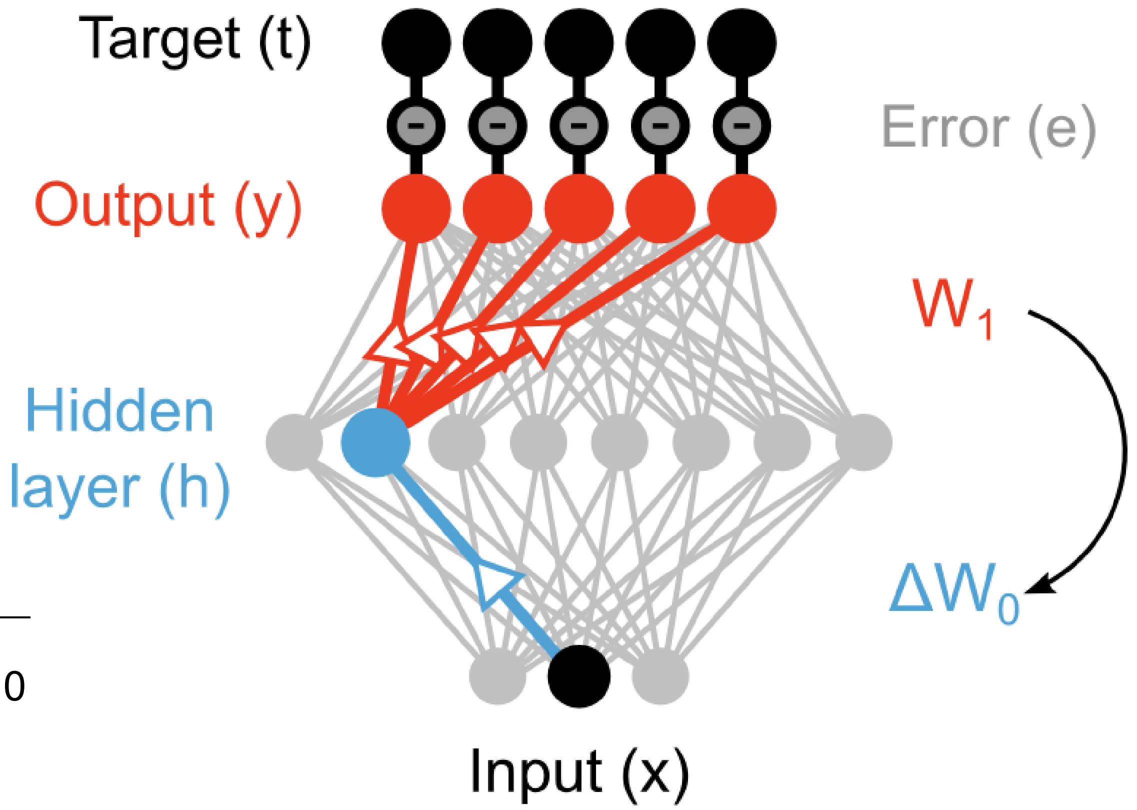
To do so, you must be able to assign “credit” (or “blame”) to synapses in the hidden layers for their contribution to output of the network


Output
Hidden layer
Input
The behavioral effects of changes to these synaptic connections...
...depend on everything that happens up here.

3. The obvious solution: backpropagation

The most obvious solution to credit assignment is to explicitly calculate the partial derivative of your cost function with respect to your synaptic weights in the hidden layers (AKA backpropagation)

$$u = W_0 x,\; v = W_1 h$$
$$h = \sigma(u),\; y = \sigma(v)$$
$$e = (y - t),\; L = \frac{1}{2} e^2$$
$$\Delta W_0 \propto \frac{\partial L}{\partial W_0}$$
$$= \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h} \cdot \frac{\partial h}{\partial u} \cdot \frac{\partial u}{\partial W_0}$$
$$= e \cdot W_1^T \cdot \sigma'(u) \cdot x$$



3. The obvious solution: backpropagation

The most obvious solution to credit assignment is to explicitly calculate the partial derivative of your cost function with respect to your synaptic weights in the hidden layers (AKA backpropagation)

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

Here's what we need to do backprop updates in the hidden layer

3. The obvious solution: backpropagation

The most obvious solution to credit assignment is to explicitly calculate the partial derivative of your cost function with respect to your synaptic weights in the hidden layers (AKA backpropagation)

$$\Delta W_0 \propto \boxed{e} \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need the error (difference between output generated by a forward pass and the target)

3. The obvious solution: backpropagation

The most obvious solution to credit assignment is to explicitly calculate the partial derivative of your cost function with respect to your synaptic weights in the hidden layers (AKA backpropagation)

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need to multiply that error by the transpose of W_1

3. The obvious solution: backpropagation

The most obvious solution to credit assignment is to explicitly calculate the partial derivative of your cost function with respect to your synaptic weights in the hidden layers (AKA backpropagation)

$$\Delta W_0 \propto e \cdot W_1^T \cdot \boxed{\sigma'}(u) \cdot x$$

We need the derivative of the hidden unit activation function

3. The obvious solution: backpropagation

The most obvious solution to credit assignment is to explicitly calculate the partial derivative of your cost function with respect to your synaptic weights in the hidden layers (AKA backpropagation)

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need a forward pass without backwards flow of activity

4. Biological problems with vanilla backprop

Unfortunately, all four of those things we need are biologically problematic...

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need:
(1) Error term
(2) Transpose of downstream weights
(3) Derivative of activation function
(4) Separate forward and backward passes

4. Biological problems with vanilla backprop

Unfortunately, all four of those things we need are biologically problematic...

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need:
(1) Error term
(2) Transpose of downstream weights
(3) Derivative of activation function
(4) Separate forward and backward passes

Moreover, it's not immediately clear that this even fits with what we do know about synaptic changes in the brain...

4. Biological problems with vanilla backprop

Unfortunately, all four of those things we need are biologically problematic...

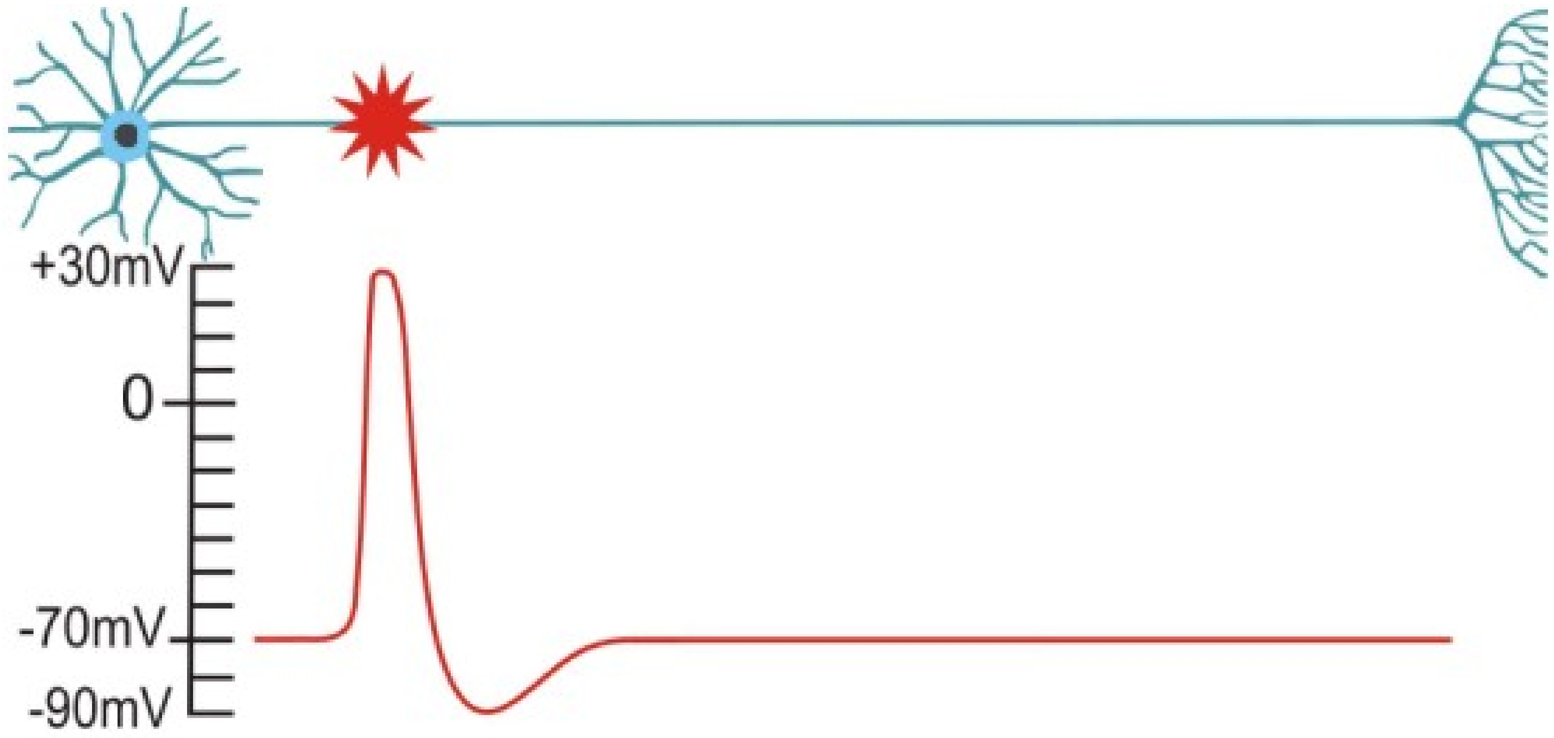
$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need:
(1) Error term
(2) Transpose of downstream weights
(3) Derivative of activation function
(4) Separate forward and backward passes

The last couple of years have seen much progress in addressing all four of these issues – I'm gonna bring you up-to-date and maybe just convince you that the brain might also do end-to-end optimization!

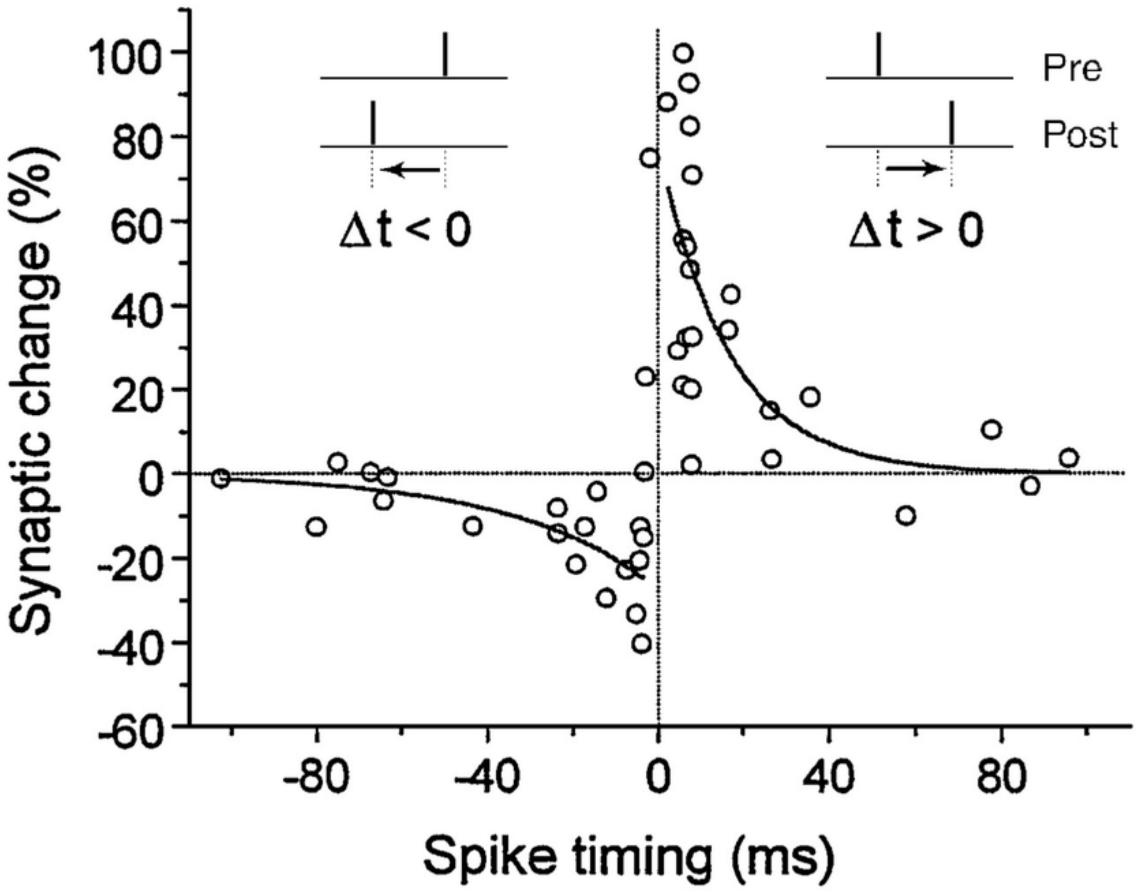
5. Issue one: error term

The brain can definitely calculate errors, but there's no evidence that synapses change based on error terms, instead it seems to use the timing of ***spikes:*** a phenomenon known as ***spike-timing-dependent plasticity***



5. Issue one: error term

The brain can definitely calculate errors, but there's no evidence that synapses change based on error terms, instead it seems to use the timing of ***spikes:*** a phenomenon known as ***spike-timing-dependent plasticity***



5. Issue one: error term

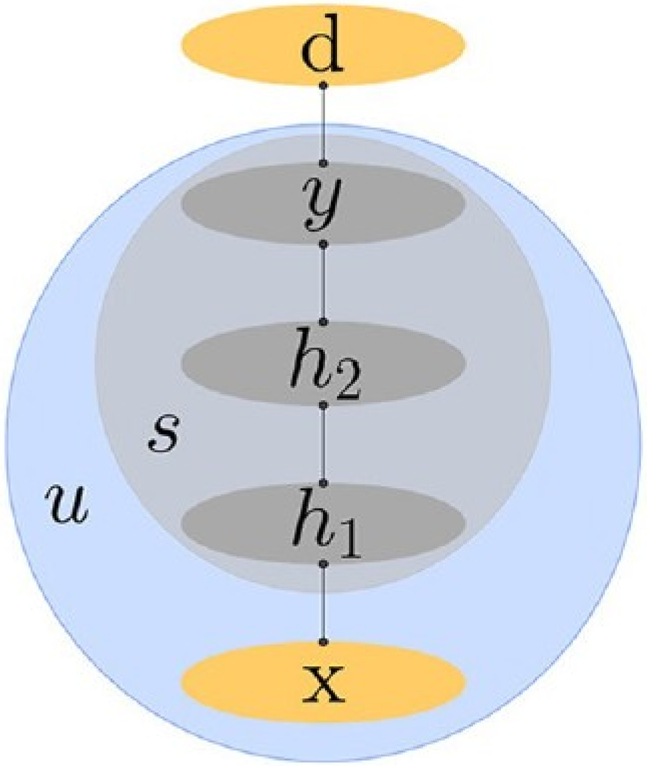
UNIVERSITY OF TORONTO SCARBOROUGH

Scellier & Bengio (2017) proposed Equilibrium Propagation, which uses a “free phase” (with no external feedback) and a “weakly clamped phase” (where the external environment nudges the network towards the correct answer)

$u = \{x, h, y\}$ w/o sigmoid

$\beta = 0$ free phase

$\beta > 0$ weakly clamped phase



If we assume symmetric weights between units then in the limit $\beta \rightarrow 0$

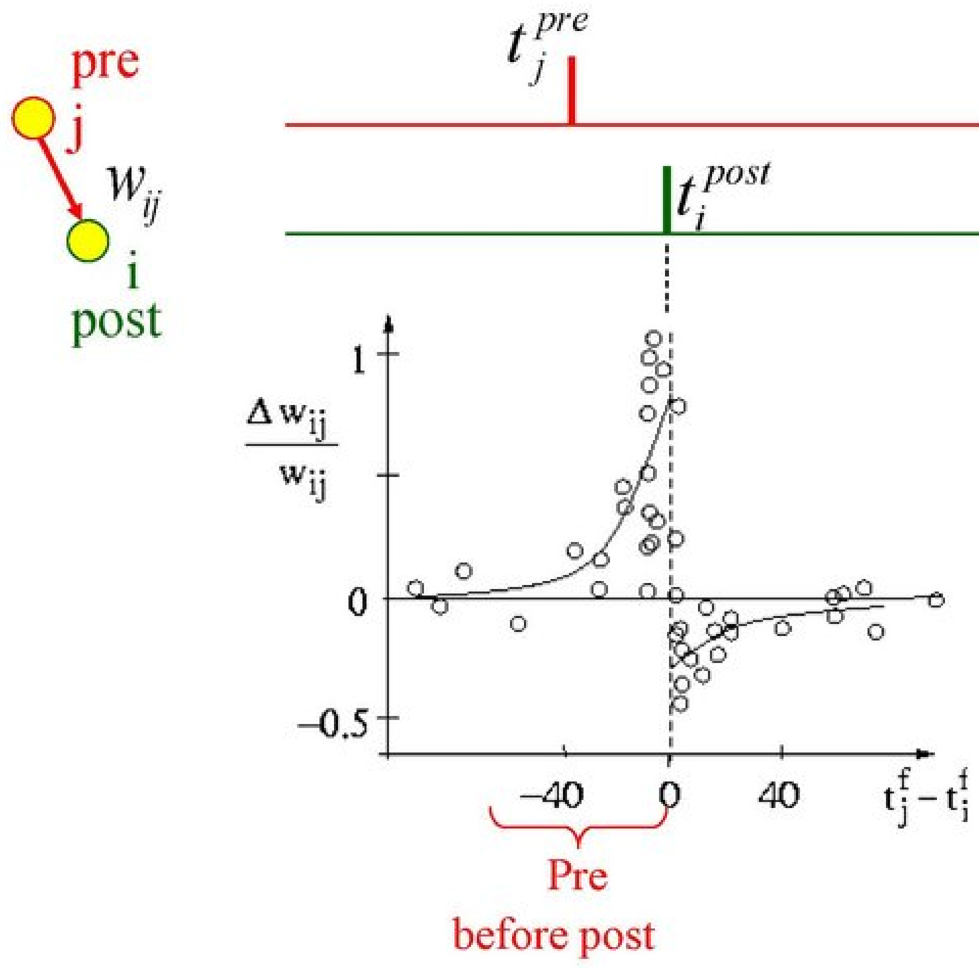
$$\Delta W_{ij} = \frac{1}{\beta}\left(\sigma(u_i^{\beta})\sigma(u_j^{\beta}) - \sigma(u_i^{0})\sigma(u_j^{0})\right)$$

This can be shown to implement backprop!

Scellier & Bengio (2017), Front. Comp. Neurosci. 11(24)

5. Issue one: error term

And fascinatingly, it predicts **spike-timing-dependent plasticity (STDP)**



If this relationship holds:

$$\frac{dW_{ij}}{dt} = \sigma(u_j)\frac{du_i}{dt}$$

We can get the same STDP result experimentalists see

And this relationship **does** hold (approx.) for equilibrium propagation

5. Issue one: error term

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need:

~~(1) Error term~~

(2) Transpose of downstream weights

(3) Derivative of activation function

(4) Separate forward and backward passes

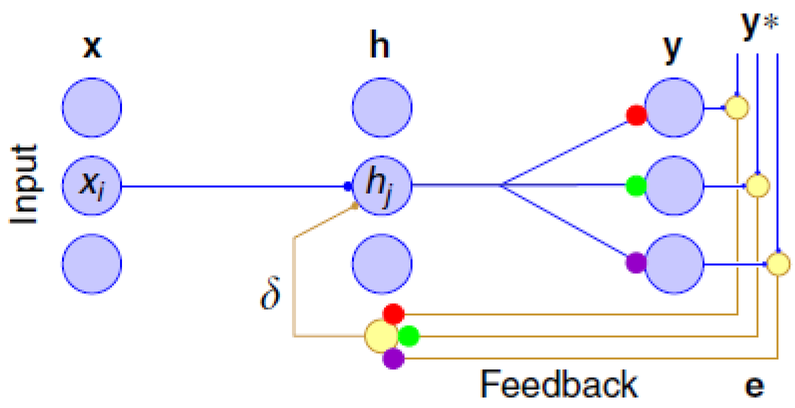
One item down:

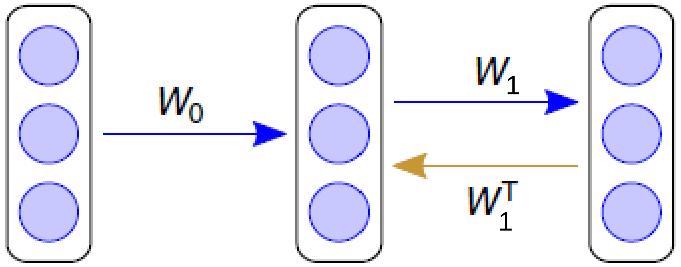
We can do gradient descent without explicit error terms being used to update synapses (and it seems to match experimental data on STDP)

6. Issue two: downstream weights

As noted, the backprop update rule assumes that the hidden layer neurons have access to the error term multiplied by the transpose of the downstream weights, W_1

That's not a realistic proposal for the brain, and it has led many neuroscientists to dismiss backprop



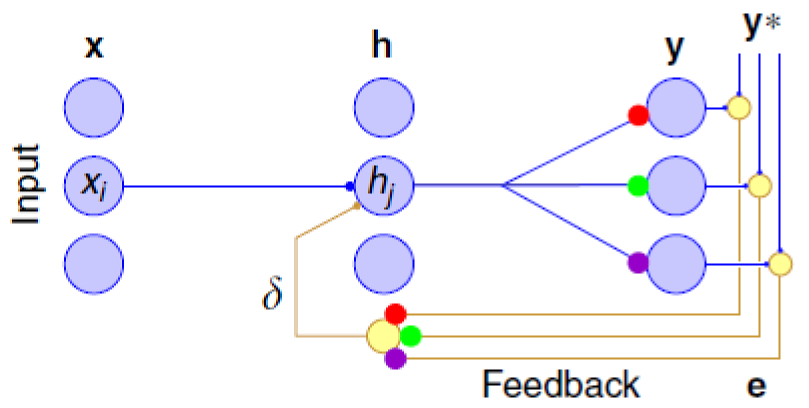


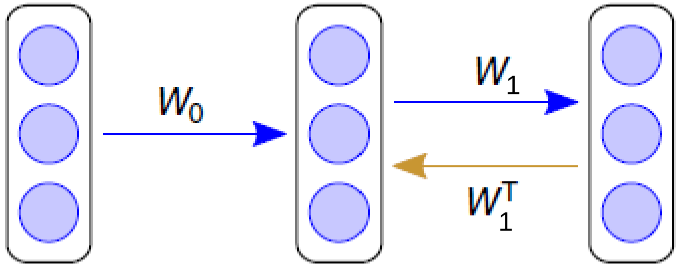
Lillicrap et al. (2016), Nat. Commi. 7(13276)

6. Issue two: downstream weights

As noted, the backprop update rule assumes that the hidden layer neurons have access to the error term multiplied by the transpose of the downstream weights, W_1

Timothy Lillicrap had an idea, maybe we could train the network to develop symmetric weights?





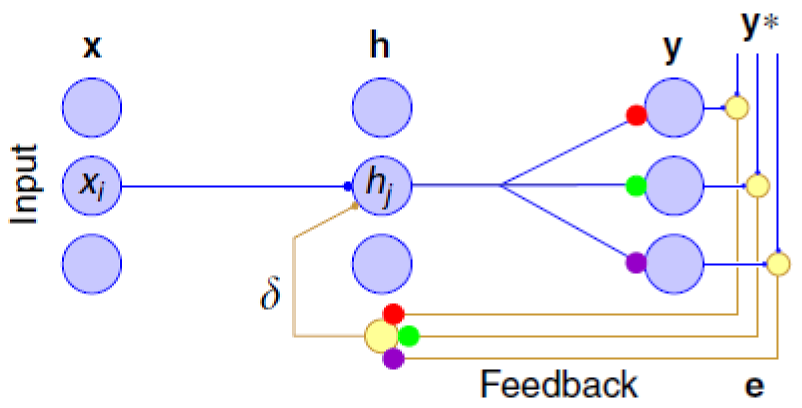
Lillicrap et al. (2016), Nat. Commi. 7(13276)

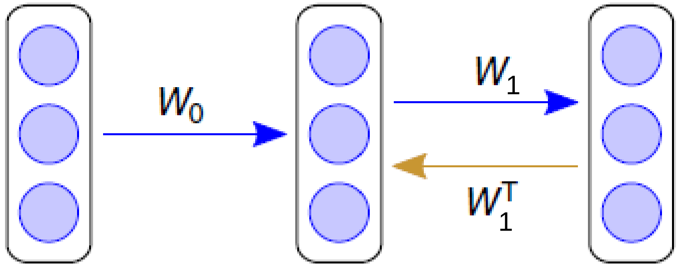
6. Issue two: downstream weights

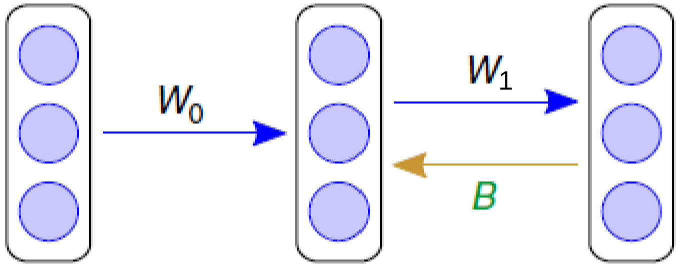
As noted, the backprop update rule assumes that the hidden layer neurons have access to the error term multiplied by the transpose of the downstream weights, W_1

Timothy Lillicrap had an idea, maybe we could train the network to develop symmetric weights?

To test his first attempt at such an algorithm, he used a control condition wherein the error was sent back through a random matrix B





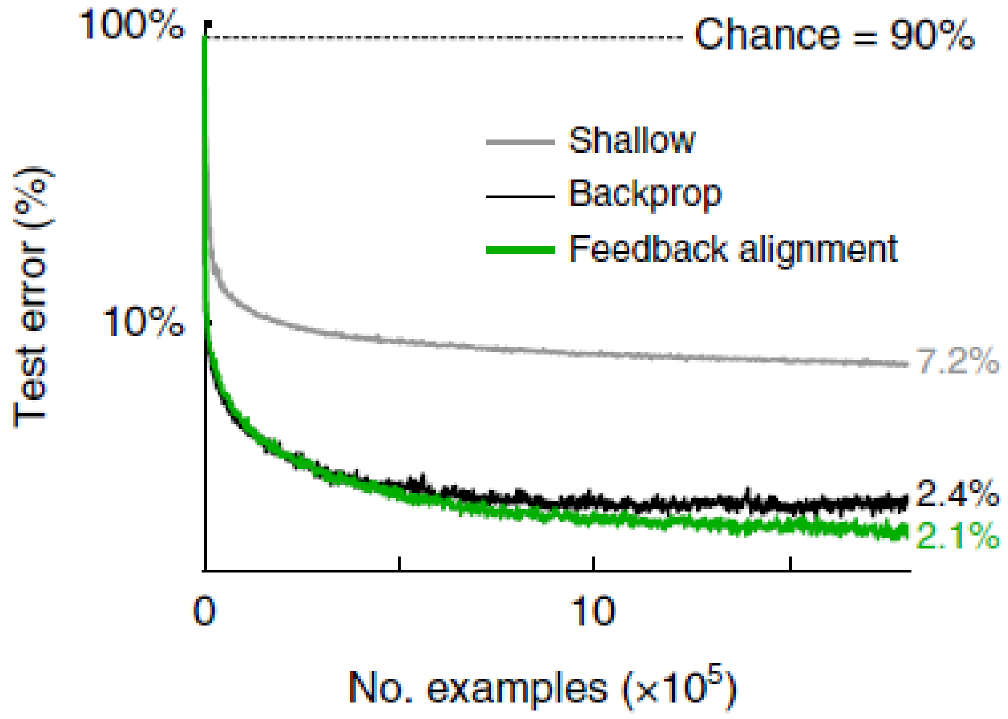


Lillicrap et al. (2016), Nat. Commi. 7(13276)

6. Issue two: downstream weights

Weirdly, the control network learned quite well!!!

Results on MNIST:



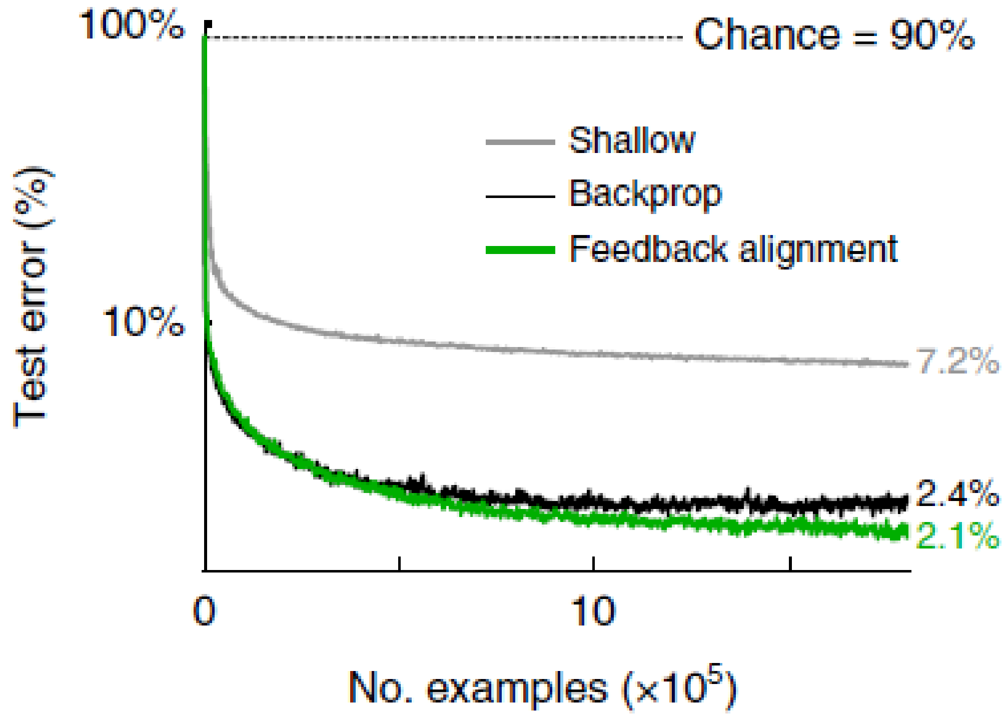
Lillicrap et al. (2016), Nat. Commi. 7(13276)

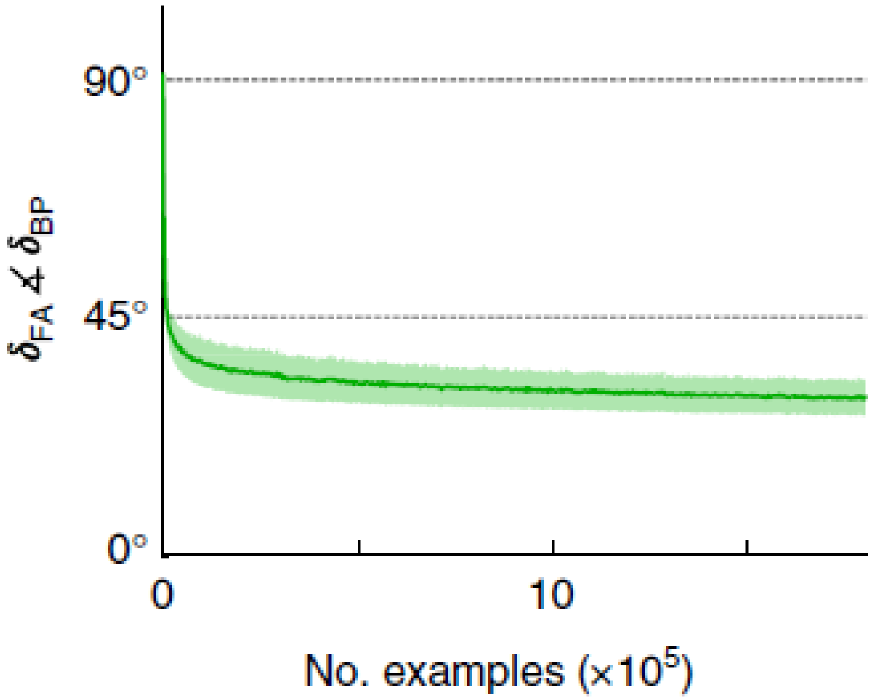
6. Issue two: downstream weights

Weirdly, the control network learned quite well!!!

Results on MNIST:

The reason was that the forward weights "aligned" themselves with the backwards weights

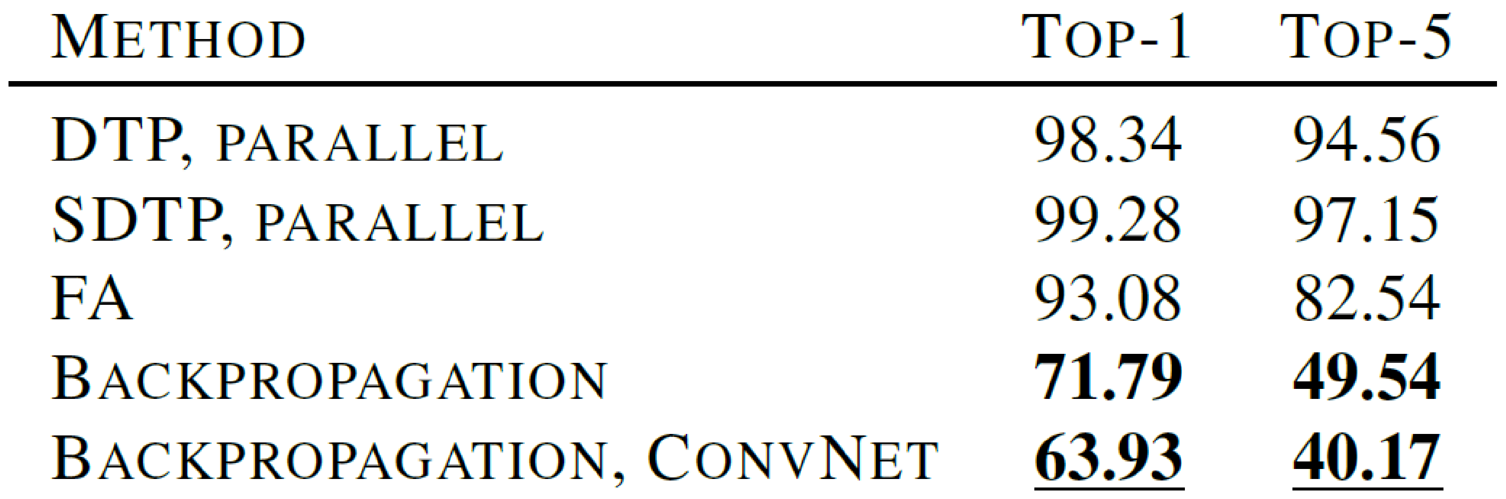




Lillicrap et al. (2016), Nat. Commi. 7(13276)

6. Issue two: downstream weights

There's still some work to do though...



Method	Top-1	Top-5
DTP, parallel	98.34	94.56
SDTP, parallel	99.28	97.15
FA	93.08	82.54
Backpropagation	**71.79**	**49.54**
Backpropagation, ConvNet	**63.93**	**40.17**

Table 2. Test errors on ImageNet.

Bartunov et al. (2018) – ICML

6. Issue two: downstream weights

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need:
~~(1) Error term~~
~~(2) Transpose of downstream weights~~
(3) Derivative of activation function
(4) Separate forward and backward passes

Two items down:
We can do gradient descent without the transpose of downstream weights

7. Issue three: derivatives of spikes

Neurons don't communicate with analog signals. Spikes are all-or-none events

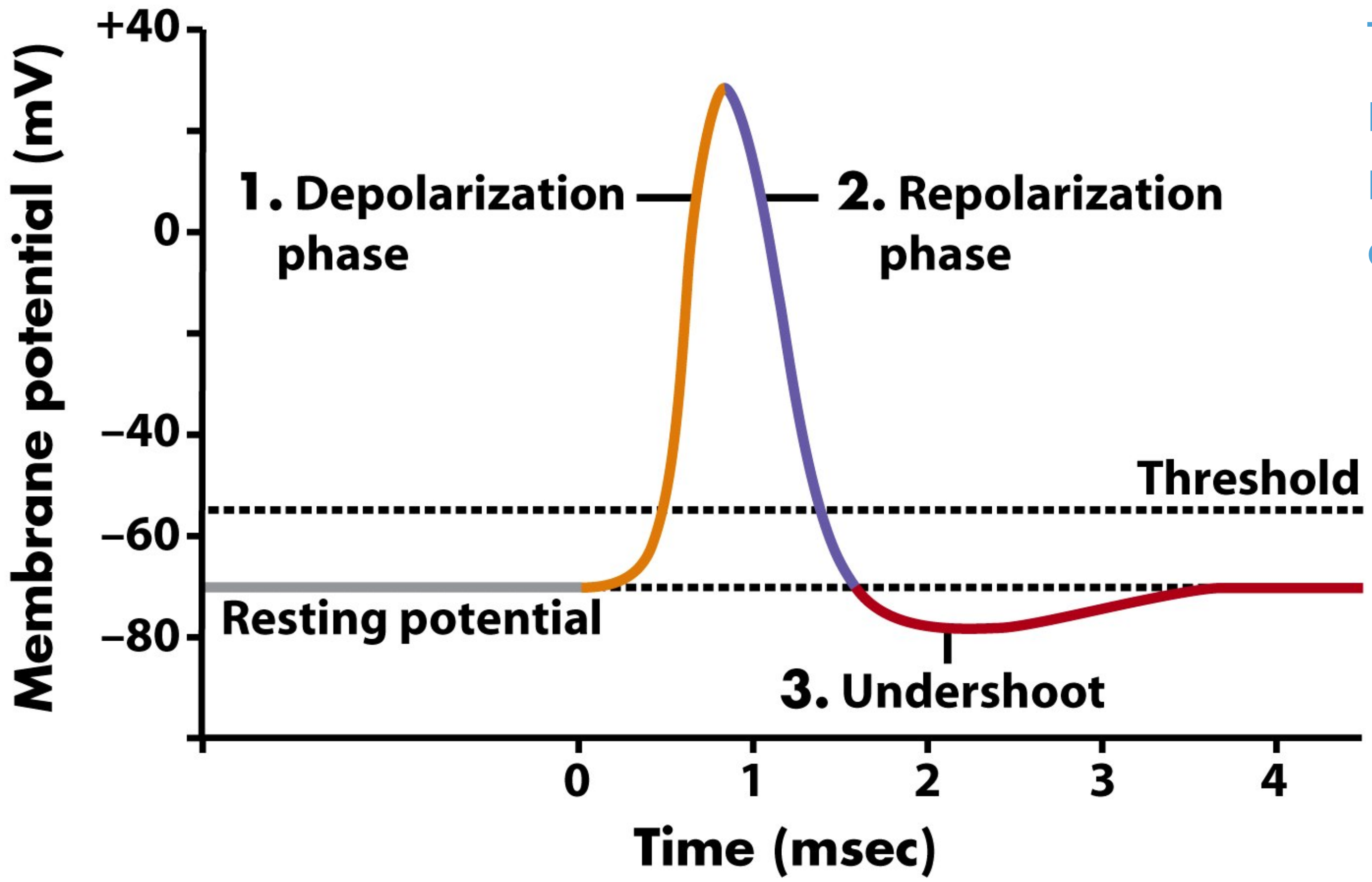


Figure 45-5 Biological Science, 2/e
© 2005 Pearson Prentice Hall, Inc.

This usually gets represented with a non-differentiable δ function:

$$S_i(t) = \sum_k \delta(t - t_k)$$

$$\delta(x) = \begin{cases} 0, \text{ if } x \neq 0 \\ \frac{1}{dt}, \text{ o/w} \end{cases}$$

7. Issue three: derivatives of spikes

Neurons don't communicate with analog signals. Spikes are all-or-none events

But, remember, we need to take the derivative of the activation function, which is supposed to represent the spiking activity...

We could treat the activation function as the spike rate (which is the typical interpretation), but that's problematic, since there's good evidence that the specific timing of spikes can carry a fair bit of information

This usually gets represented with a non-differentiable δ function:

$$S_i(t) = \sum_k \delta(t - t_k)$$

$$\delta(x) = \begin{cases} 0, \text{ if } x \neq 0 \\ \frac{1}{dt}, \text{ o/w} \end{cases}$$

7. Issue three: derivatives of spikes

Zenke & Ganguli (2018) approach this by first modifying the loss function to minimize the van Rossum distance between a target spike train and the actual spike train:

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

Where $\hat{S}_i$ is the target spike train, and α is a temporal convolution kernel

Taking the gradient, we get:

$$\frac{\partial L}{\partial W_{ij}} = - \int_{-\infty}^{t} ds \left[\left(\alpha * \hat{S}_i - \alpha * S_i \right)(s) \right] \left(\alpha * \frac{\partial S_i}{\partial W_{ij}} \right)(s)$$

Zenke & Ganguli (2018) Neural Computation, 30(6): 1514-1541

7. Issue three: derivatives of spikes

Zenke & Ganguli (2018) approach this by first modifying the loss function to minimize the van Rossum distance between a target spike train and the actual spike train:

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

Where $\hat{S}_i$ is the target spike train, and α is a temporal convolution kernel

Taking the gradient, we get:

Ah, but here's this bugger...

$$\frac{\partial L}{\partial W_{ij}} = -\int_{-\infty}^{t} ds \left[\left(\alpha * \hat{S}_i - \alpha * S_i \right)(s) \right] \left(\alpha * \frac{\partial S_i}{\partial W_{ij}} \right)(s)$$

Zenke & Ganguli (2018) Neural Computation, 30(6): 1514-1541

7. Issue three: derivatives of spikes

Zenke & Ganguli (2018) deal with the spike train derivative by replacing the spike train, S_i, in the gradient equation with an auxilliary function of the membrane potential, $\sigma(U_i(t))$, where:

$$U_i(t) \approx \sum_j W_{ij} (\epsilon * S_j(t))$$

Where ε is the shape of the postsynaptic response to a spike.

We can then estimate the gradient with:

$$\frac{\partial L}{\partial W_{ij}} \approx - \int_{-\infty}^{t} ds [(\alpha * \hat{S}_i - \alpha * S_i)(s)] \alpha (\sigma'(U_i(s))) (\epsilon * Sj)(s)$$

Zenke & Ganguli (2018) Neural Computation, 30(6): 1514-1541

7. Issue three: derivatives of spikes

Zenke & Ganguli (2018) deal with the spike train derivative by replacing the spike train, S_i, in the gradient equation with an auxilliary function of the membrane potential, $\sigma(U_i(t))$, where:

$$U_i(t) \approx \sum_j W_{ij}(\epsilon * S_j(t))$$

Where ε is the shape of the postsynaptic response to a spike.

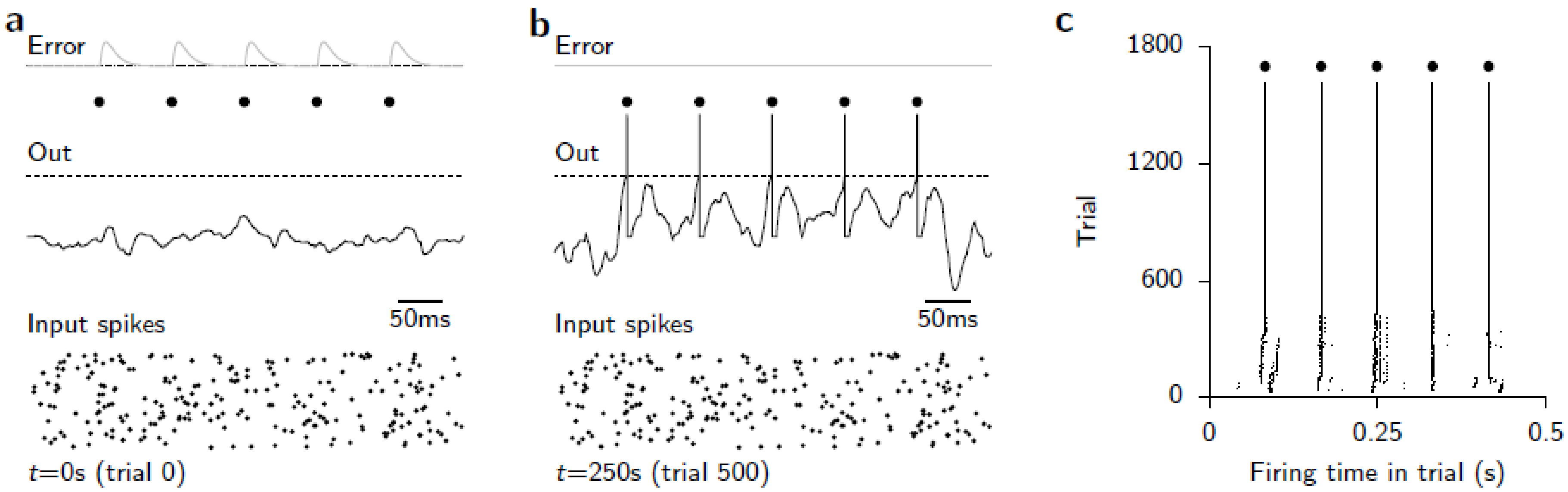
We can then estimate the gradient with:

$$\frac{\partial L}{\partial W_{ij}} \approx -\int_{-\infty}^{t} ds \underbrace{\left[(\alpha * \hat{S}_i - \alpha * S_i)(s)\right]}_{\text{Error term}} \underbrace{\alpha(\sigma'(U_i(s)))(\epsilon * Sj)(s)}_{\text{Eligibility trace}}$$

Zenke & Ganguli (2018) Neural Computation, 30(6): 1514-1541

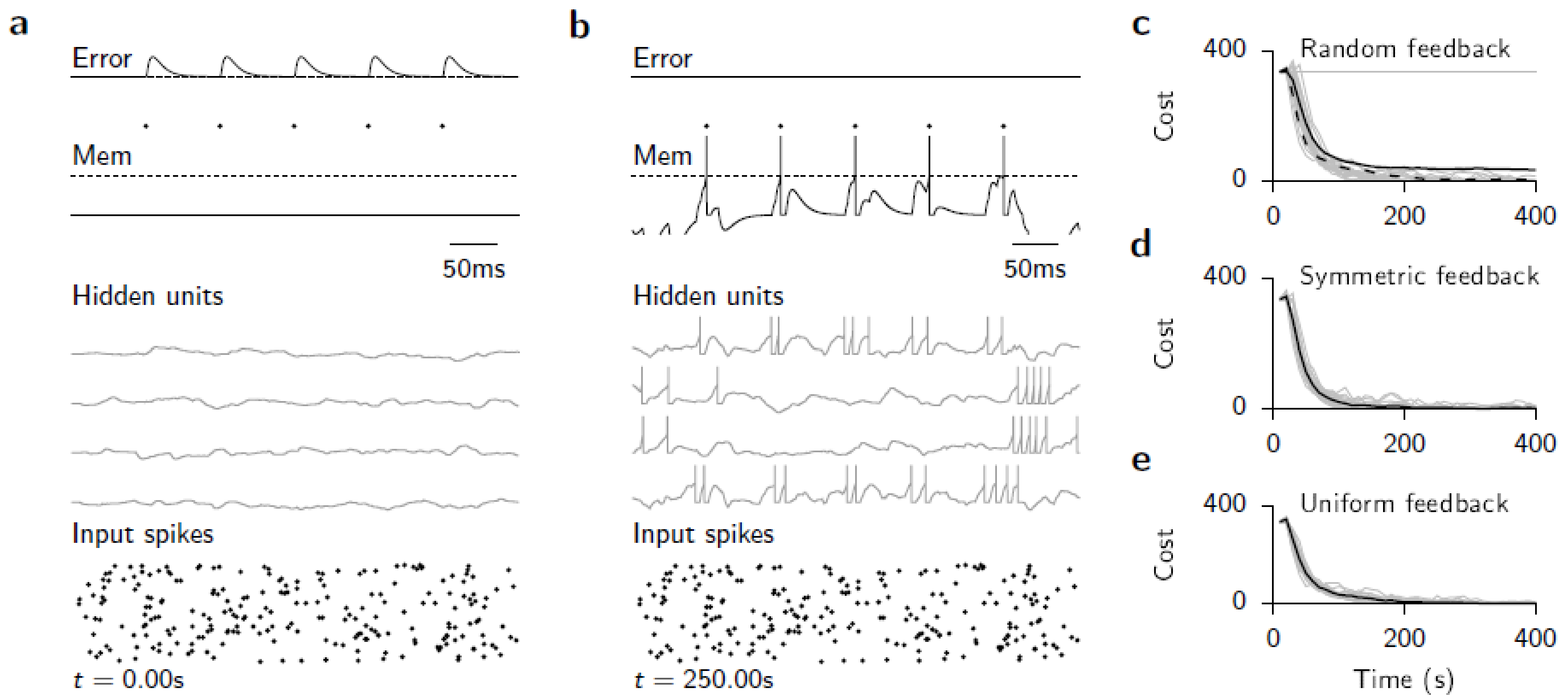
7. Issue three: derivatives of spikes

The network can now be trained to generate specific spike sequences:



Zenke & Ganguli (2018) Neural Computation, 30(6): 1514-1541

Training in networks with hidden layers is a straightforward extension:



Zenke & Ganguli (2018) Neural Computation, 30(6): 1514-1541

7. Issue three: derivatives of spikes

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need:
~~(1) Error term~~
~~(2) Transpose of downstream weights~~
~~(3) Derivative of activation function~~
(4) Separate forward and backward passes

Three items down:
We approximate gradient descent with precise spike trains

8. Issue four: forward/backward passes

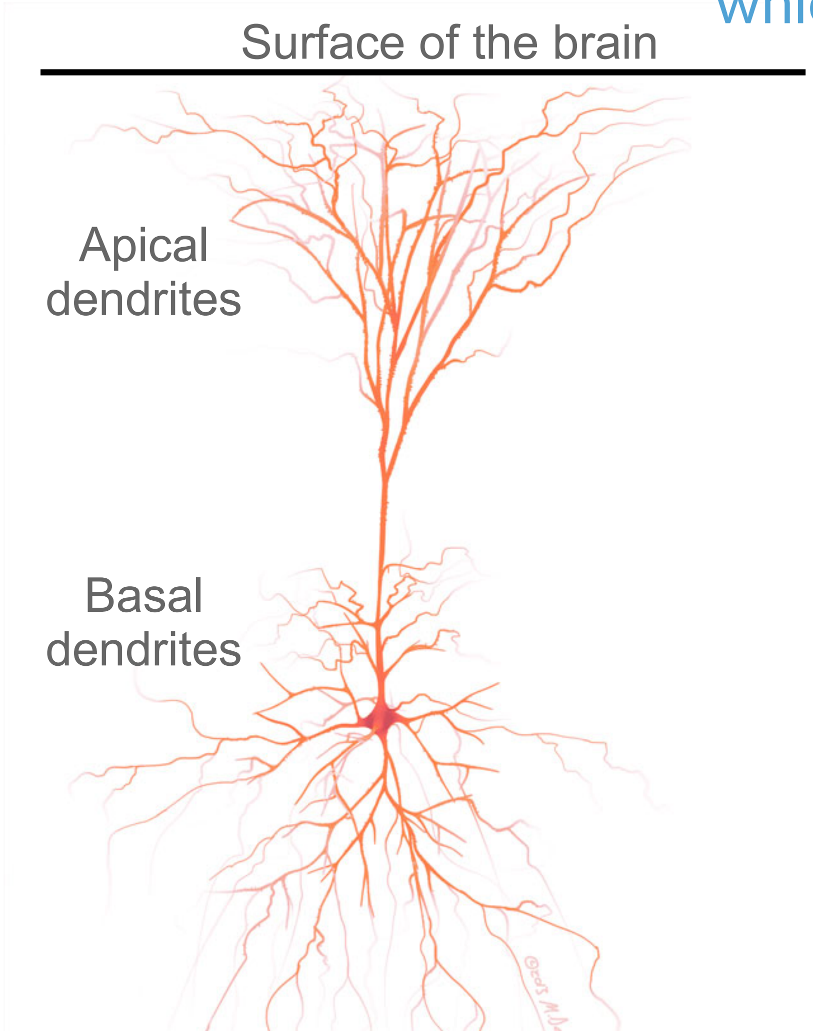
Our brains are constantly active (don't listen to the media), and there are massive backwards projections everywhere you look

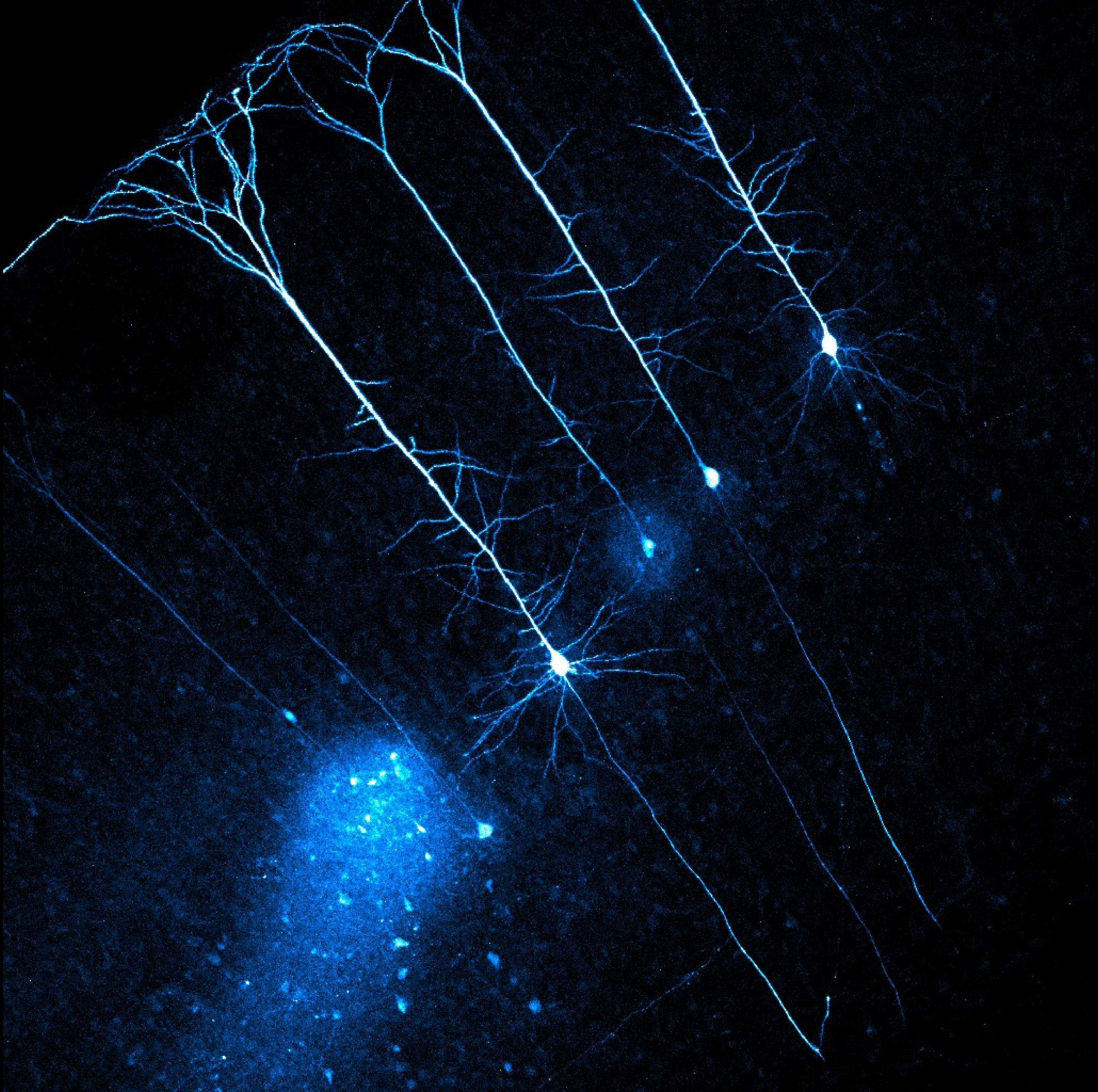
At face value that would suggest that there probably isn't a forward pass followed by a backward pass...

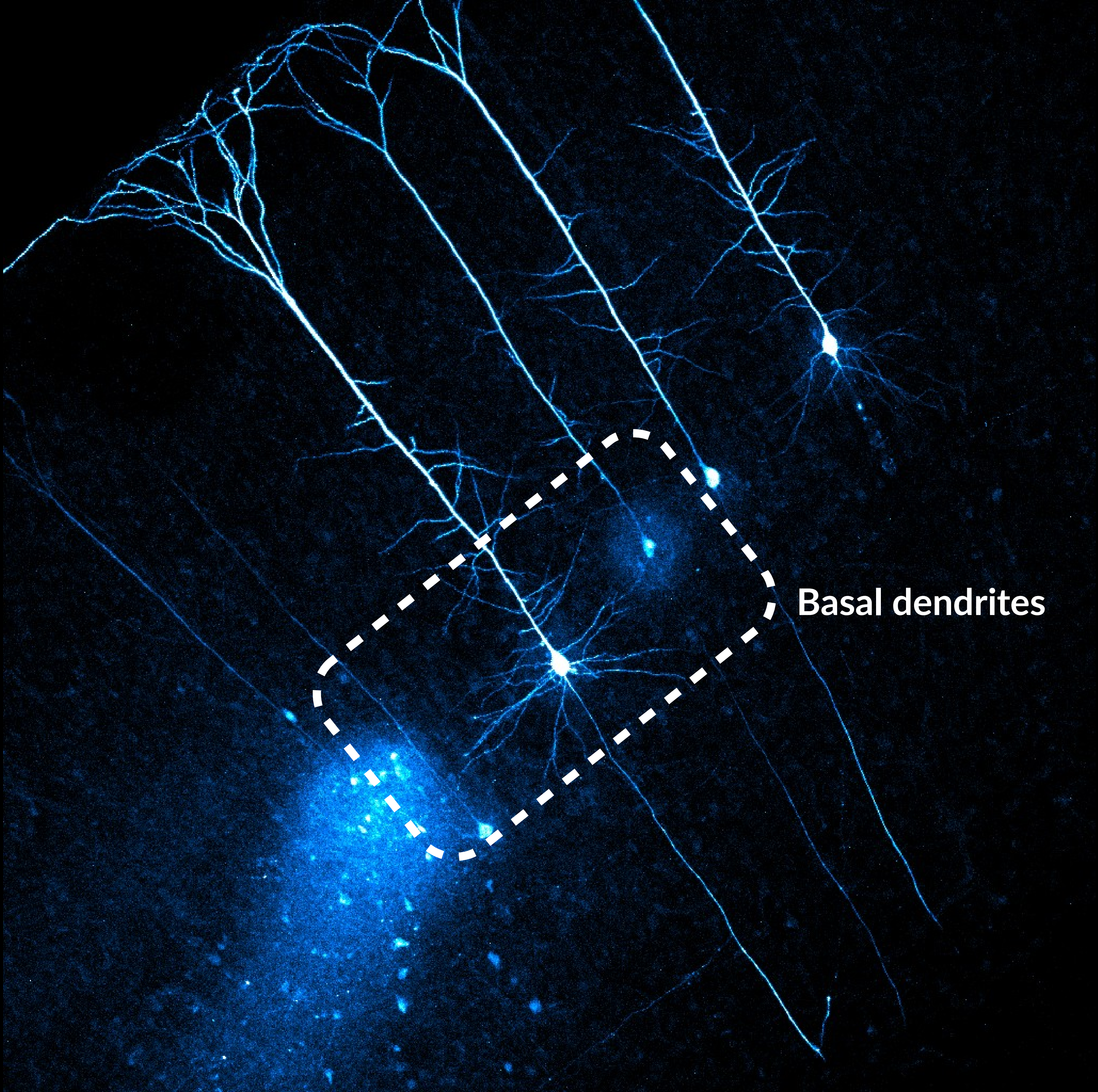
8. Issue four: forward/backward passes

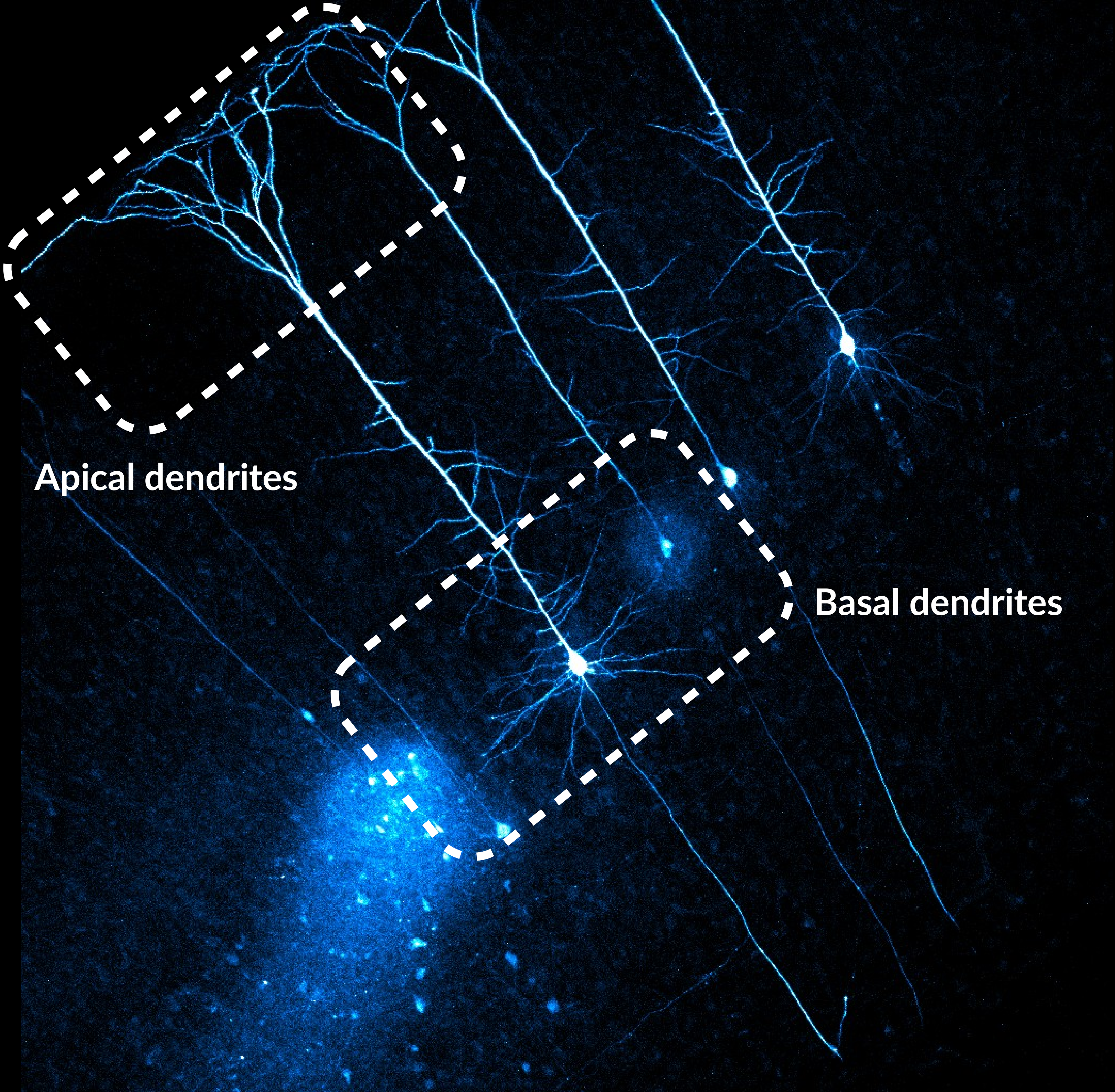
However, real neurons in the neocortex are far more complicated than the linear-non-linear points we typically use in neural networks

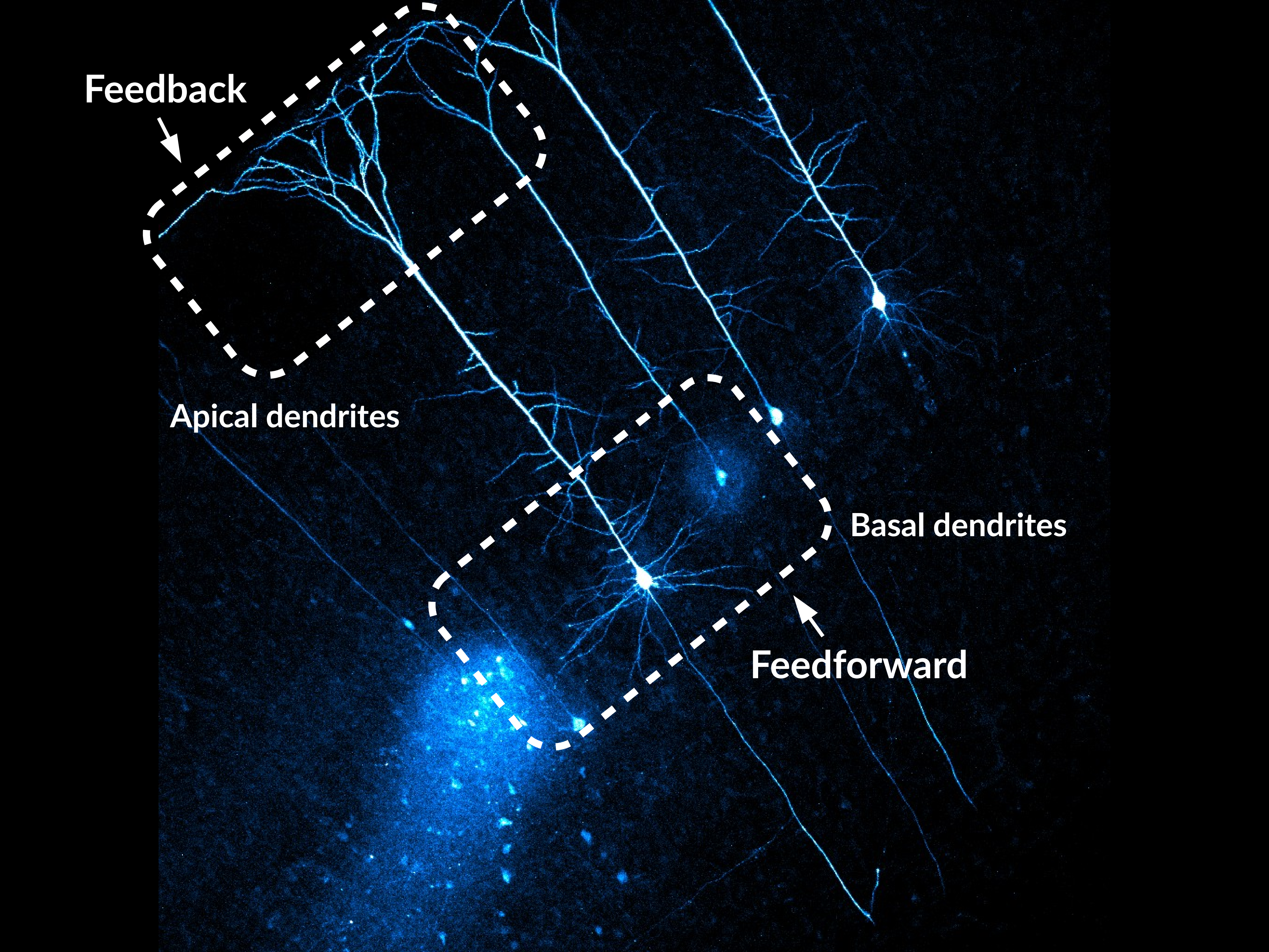
The majority of neurons in the neocortex are pyramidal neurons, which are shaped kind of like a big tree





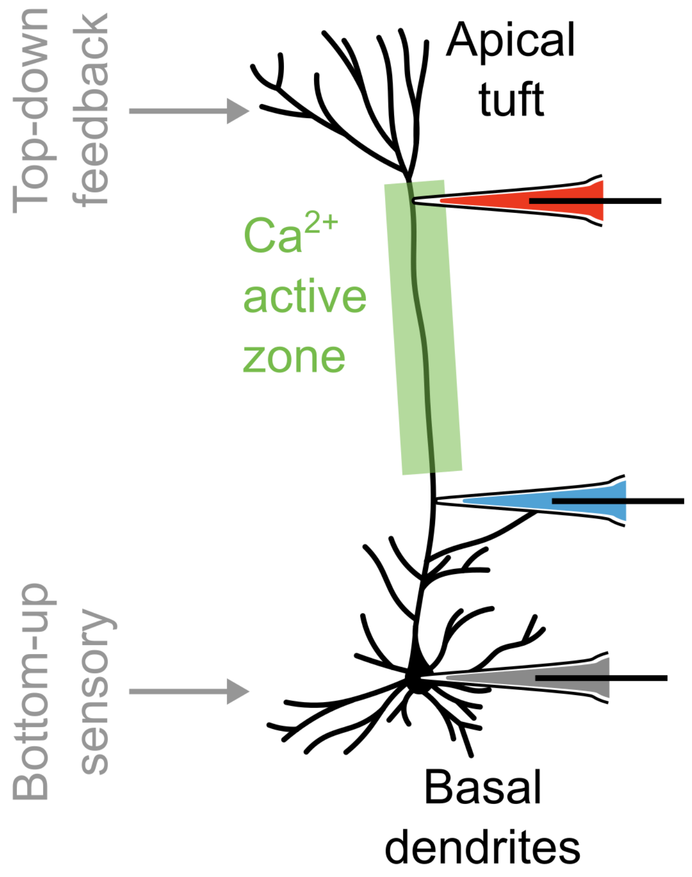

Basal dendrites

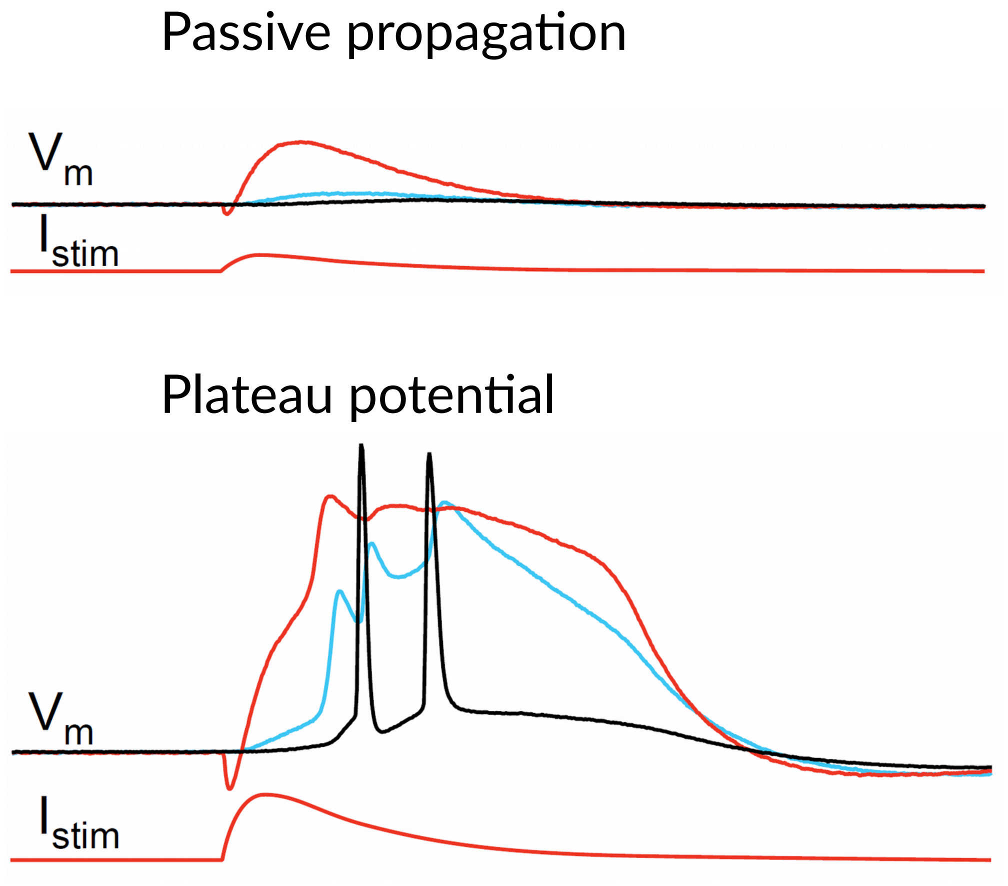

Apical dendrites
Basal dendrites


Feedback
Apical dendrites
Basal dendrites
Feedforward

8. Issue four: forward/backward passes

The apical dendrites are electrotonically segregated from the cell body, they communicate via **non-linear plateau potentials**

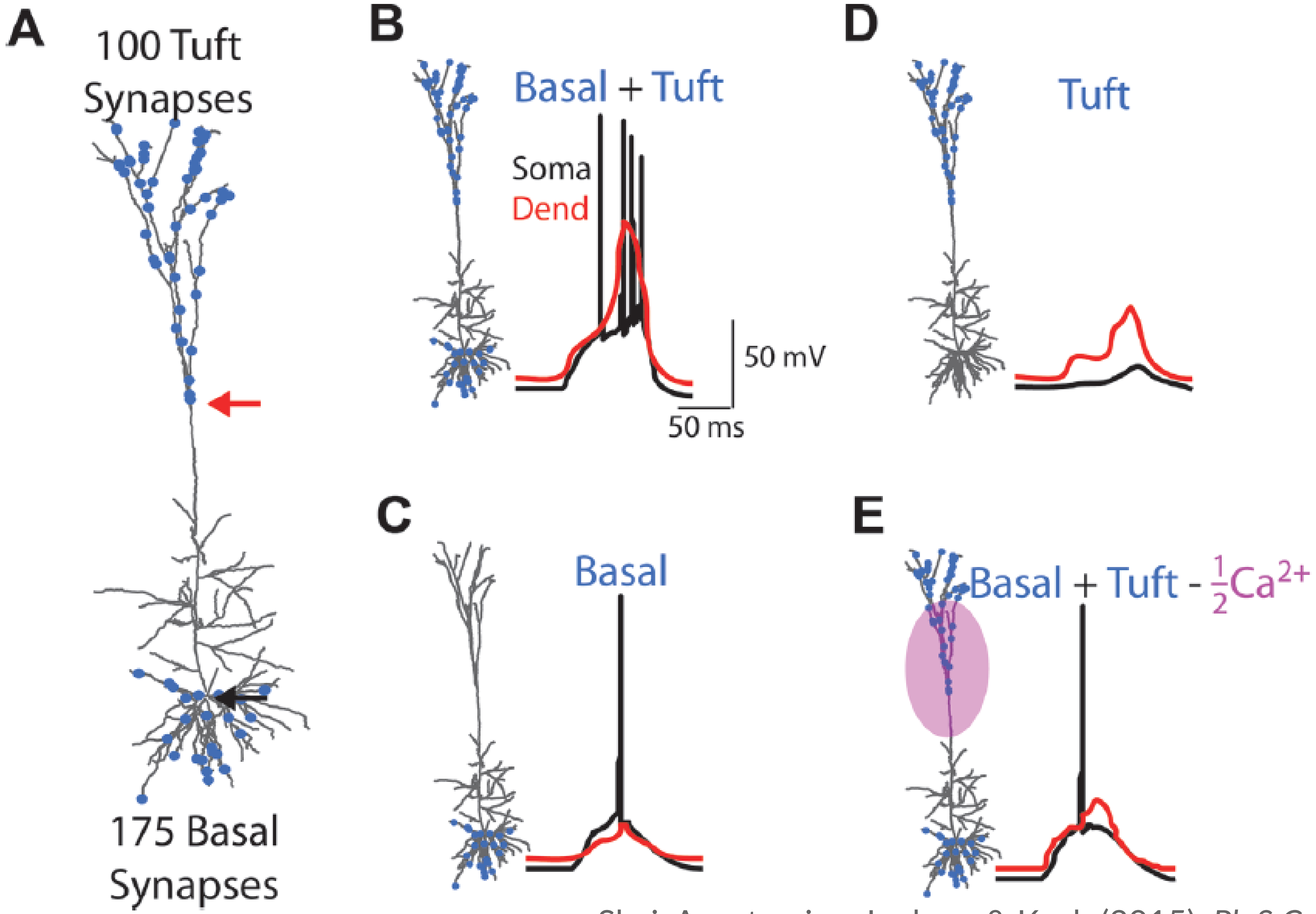




Larkum, Zhu & Sakmann. (1999), *Nature*

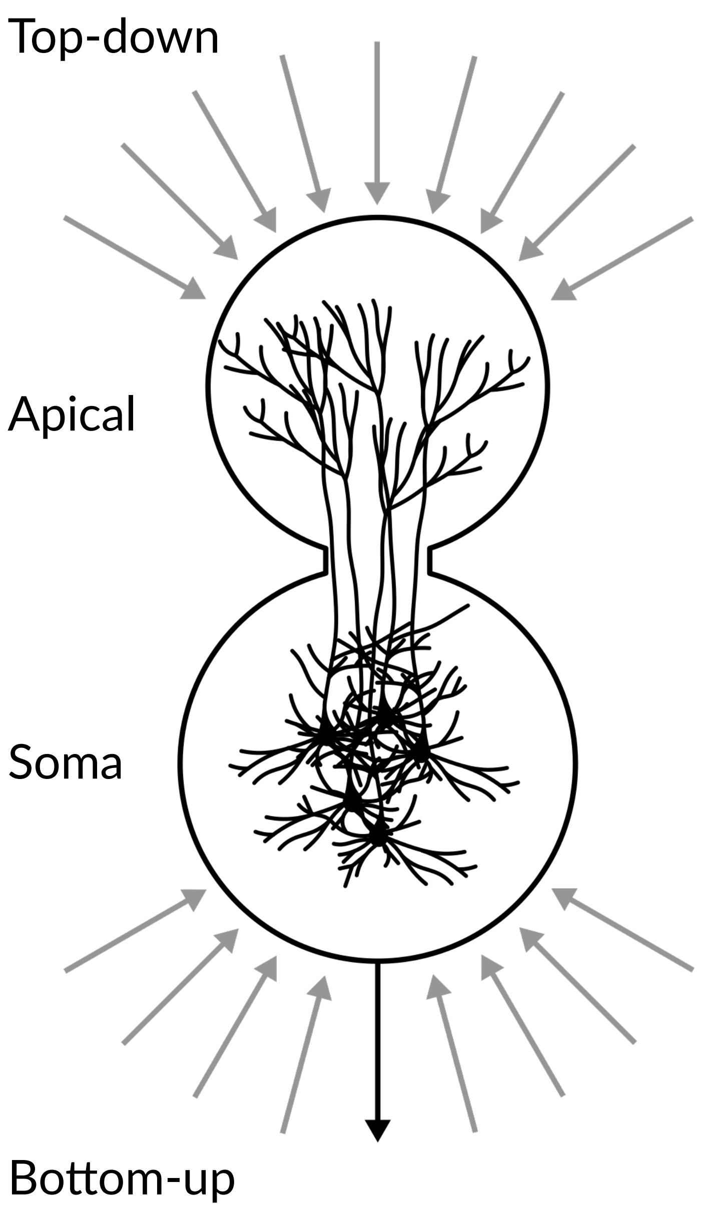
8. Issue four: forward/backward passes

Simultaneous basal and apical inputs can trigger plateau potentials, so burst-firing functions as a **coincidence detector**



Shai, Anastassiou, Larkum & Koch (2015), *PloS Comp. Bio.*

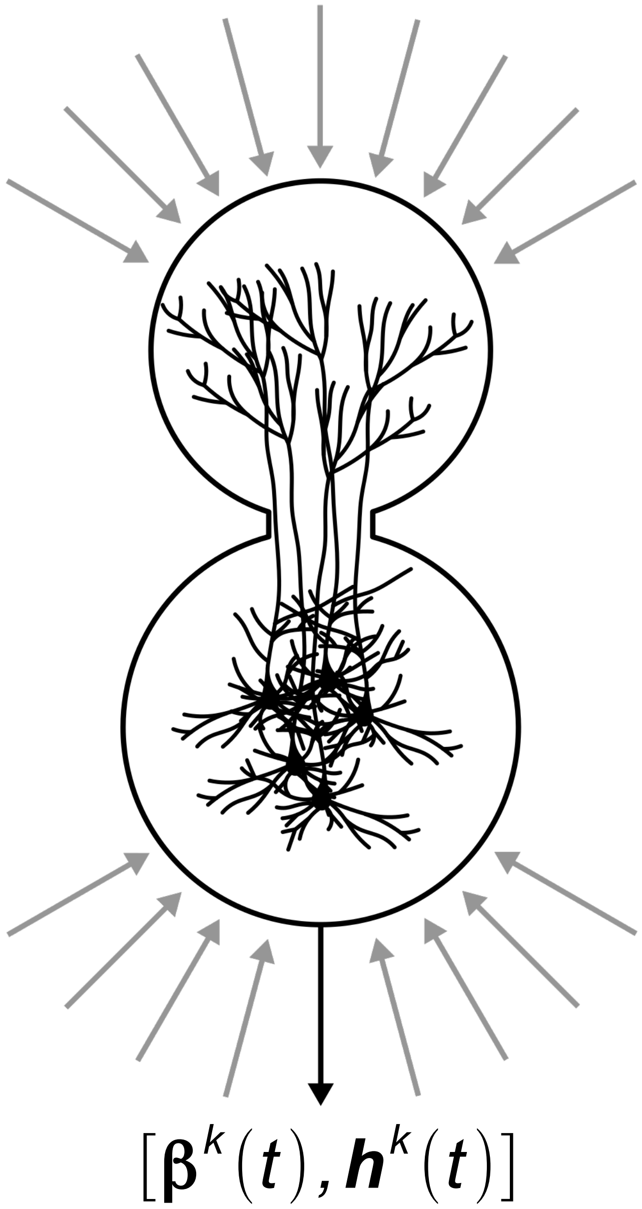
My laboratory has been working on neural network models that incorporate pyramidal neuron multiplexing to solve credit assignment



Each unit in the hidden layers of the networks are assumed to be an *ensemble* of pyramidal neurons with two distinct compartments (apical and soma)

Bottom-up, feedforward inputs arrive at the soma compartments, while top-down, feedback inputs arrive at the apical compartments

We now assume that each unit in the network is an ensemble of neurons, which can multiplex bottom-up and top-down signals



The bottom-up signals determine an **event rate**:

$\boldsymbol{h}^k(t)$ ***k* refers to the layer**

The top-down signals determine the **burst probability**:

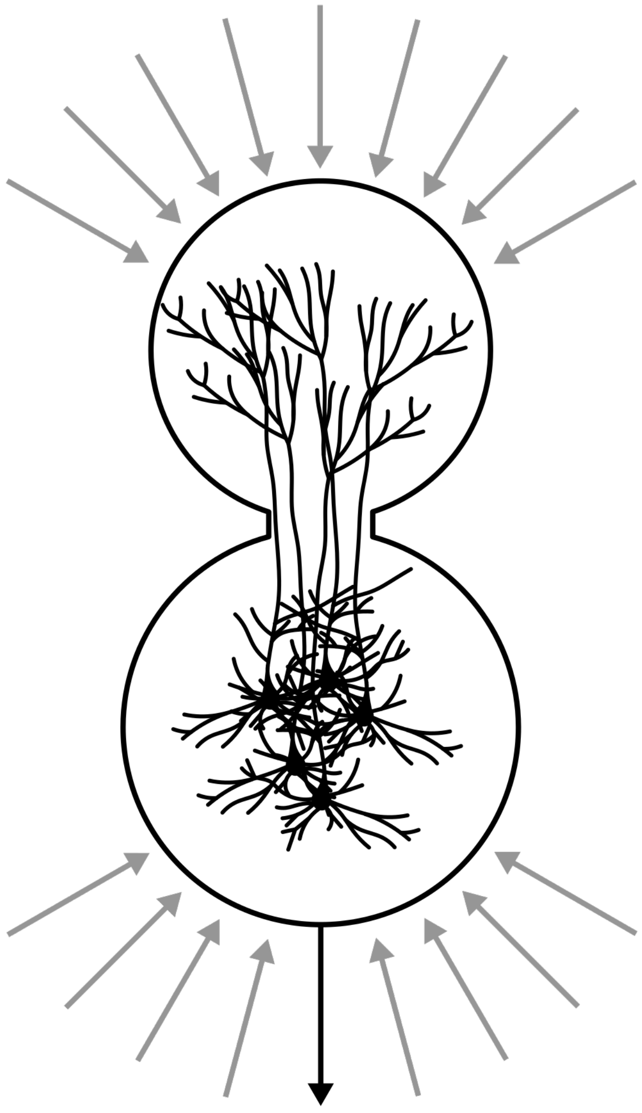
$\boldsymbol{p}^k(t)$

The product of the event-rate and burst probability determines the **burst rate**:

$$\beta^k(t) = \boldsymbol{h}^k(t) \odot \boldsymbol{p}^k(t)$$

See also:
Körding & König (2001), J. Comp. Neurosci., 11(3): 207

We now assume that each unit in the network is an ensemble of neurons, which can multiplex bottom-up and top-down signals



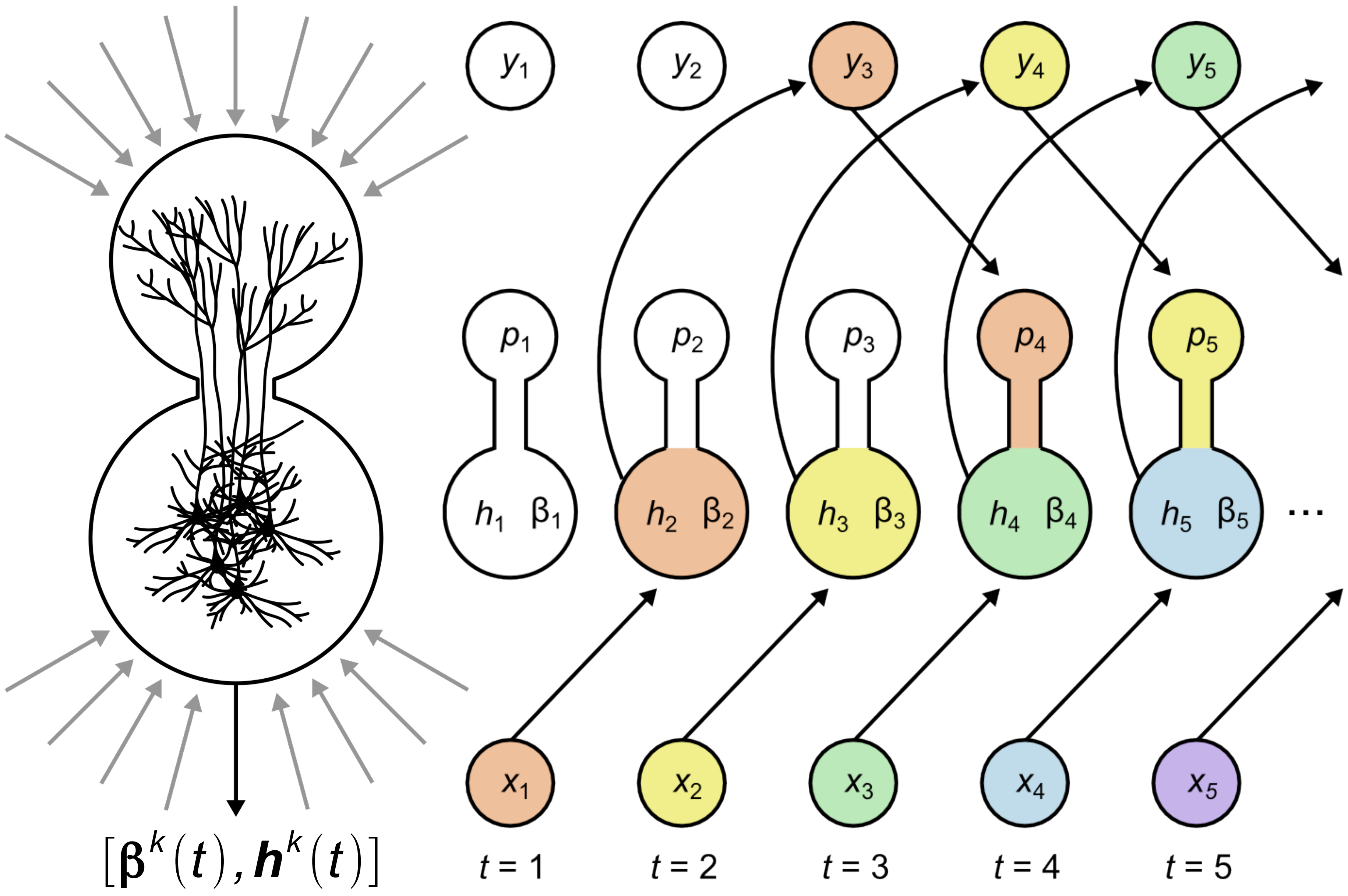
$$[\boldsymbol{\beta}^k(t), \boldsymbol{h}^k(t)]$$

We use a local loss function based on the temporal difference in bursting. If the impact of an instructive "nudge" arrives at time t^*:

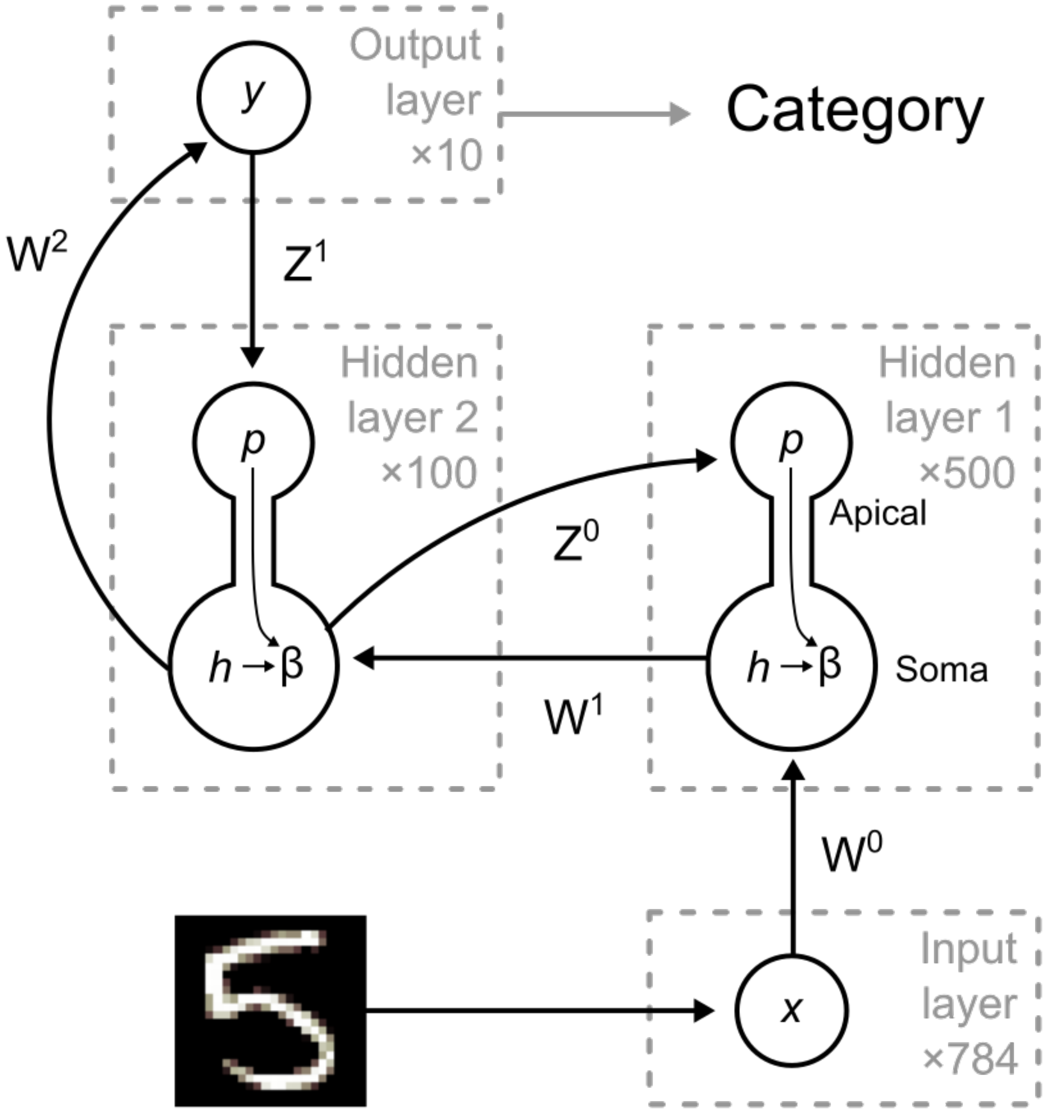
$$L^{W^k}(t^*) = \|\boldsymbol{\beta}^k(t^*) - \boldsymbol{\beta}^k(t^* - 1)\|_2^2$$

$$\Delta W^k(t^*) = \alpha(t^*) \frac{\partial L^{W^k}(t^*)}{\partial W^k}$$

The multiplexing allows us to run the network in time without **separate forward/backward passes**



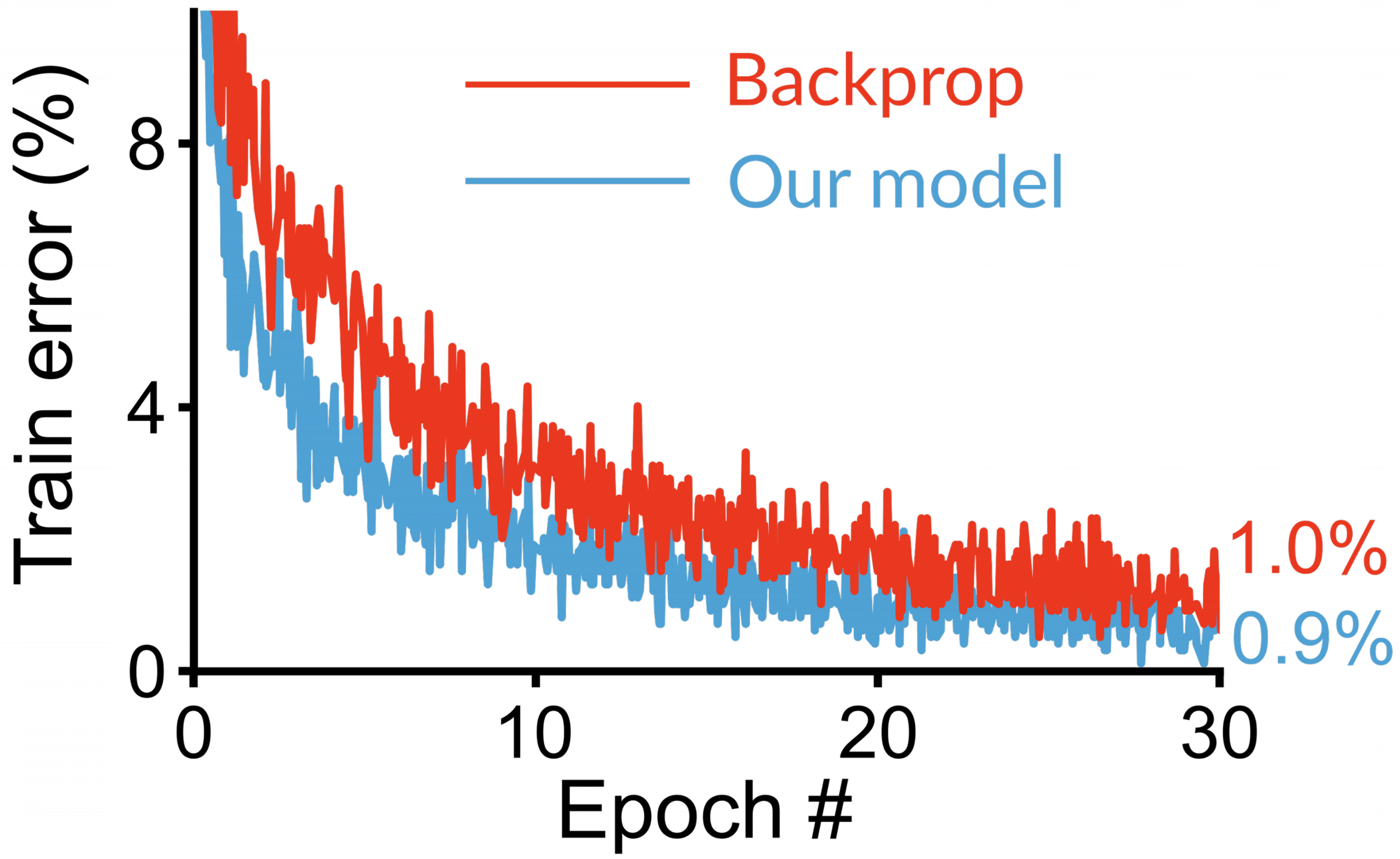
The multiplexing allows us to run the network in time without **separate forward/backward passes**



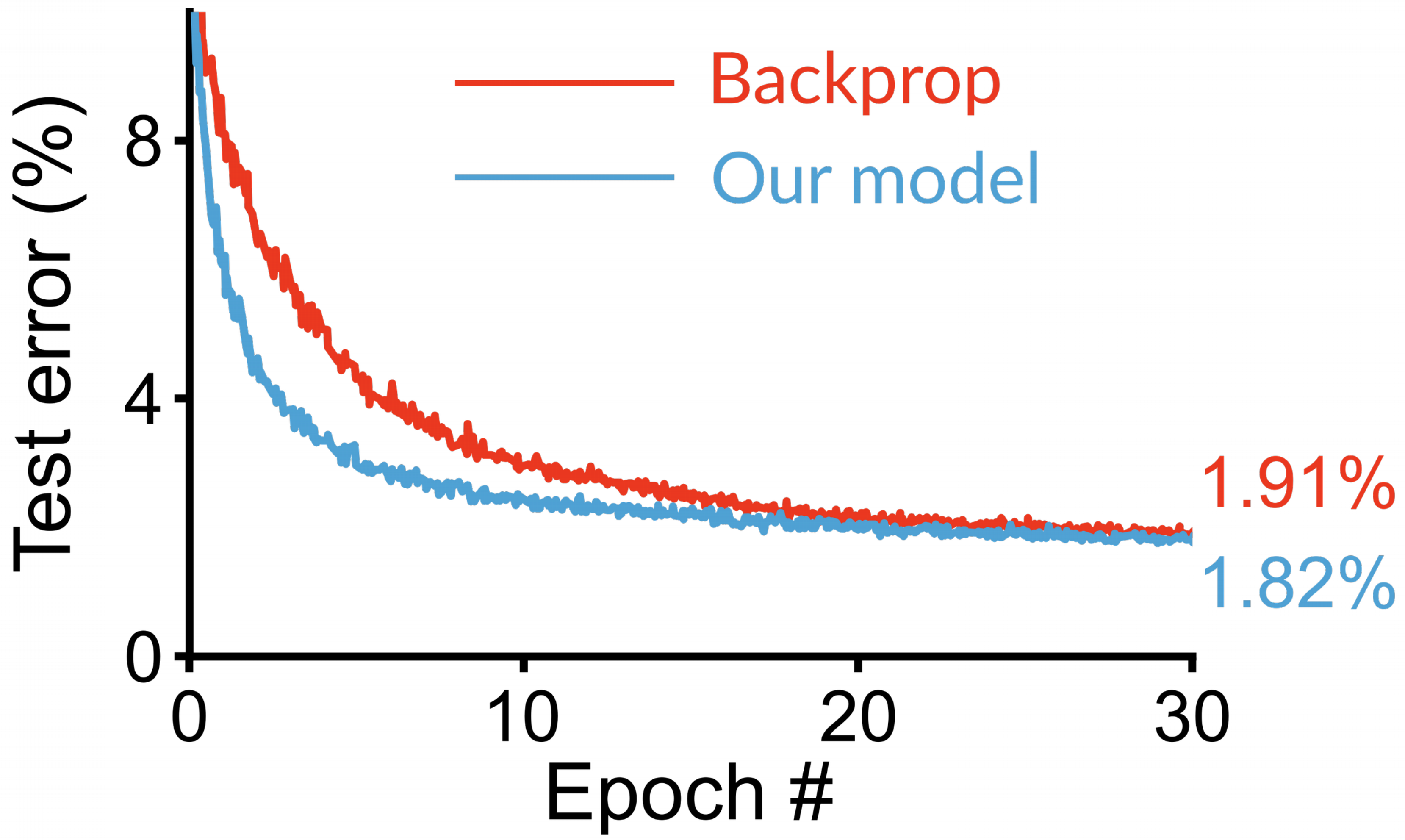
Each image in the training set is presented for 100 ms of simulated time, and one nudge is received

We test the network's error rates after each run through the 50,000 training images (referred to as an ***epoch***), 10,000 images from the training set are reserved for hyperparameter optimization

The multiplexing allows us to run the network in time without **separate forward/backward passes**



The multiplexing allows us to run the network in time without **separate forward/backward passes**



8. Issue four: forward/backward passes

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

We need:

~~(1) Error term~~

~~(2) Transpose of downstream weights~~

~~(3) Derivative of activation function~~

~~(4) Separate forward and backward passes~~

Four items down:
The neocortex is effectively designed to multiplex forward passes and backward passes simultaneously!

9. Summary

I began this talk by identifying four major issues with the biological plausibility of the backpropagation weight update rule for hidden layers:

$$\Delta W_0 \propto e \cdot W_1^T \cdot \sigma'(u) \cdot x$$

(1) Error term
(2) Transpose of downstream weights
(3) Derivative of activation function
(4) Separate forward and backward passes

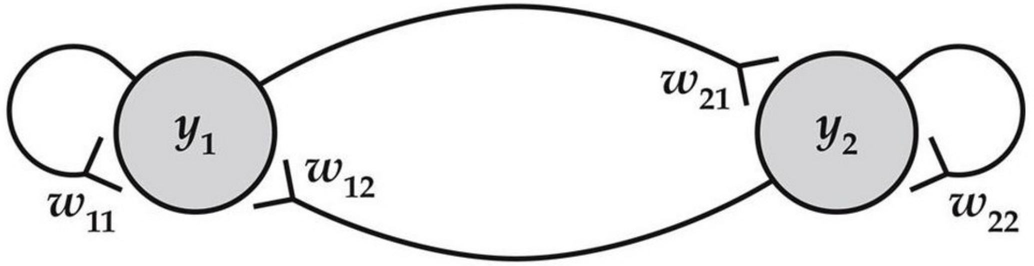
For decades, neuroscientists have dismissed the possibility of deep learning in the brain because of these issues, but over the last two years every one of these problems have been demonstrated as being very surmountable!!!

10. Major unresolved issue: backprop through time

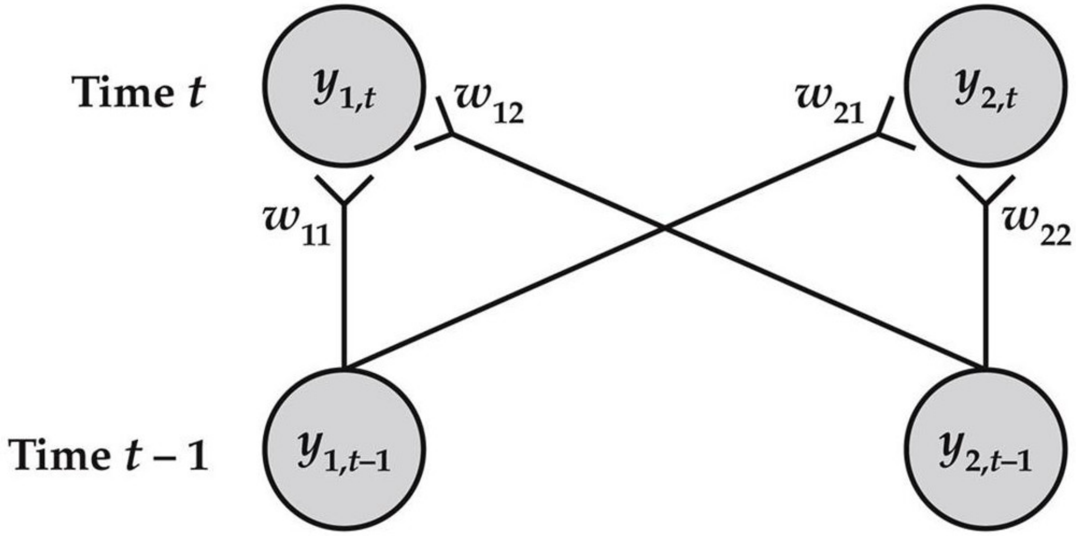
This is all very exciting, but there's still an elephant in the room: ***backprop through time***

10. Major unresolved issue: backprop through time

Backprop through time is critical for training recurrent nets, but it is very hard to see how it could be done in a biologically realistic manner

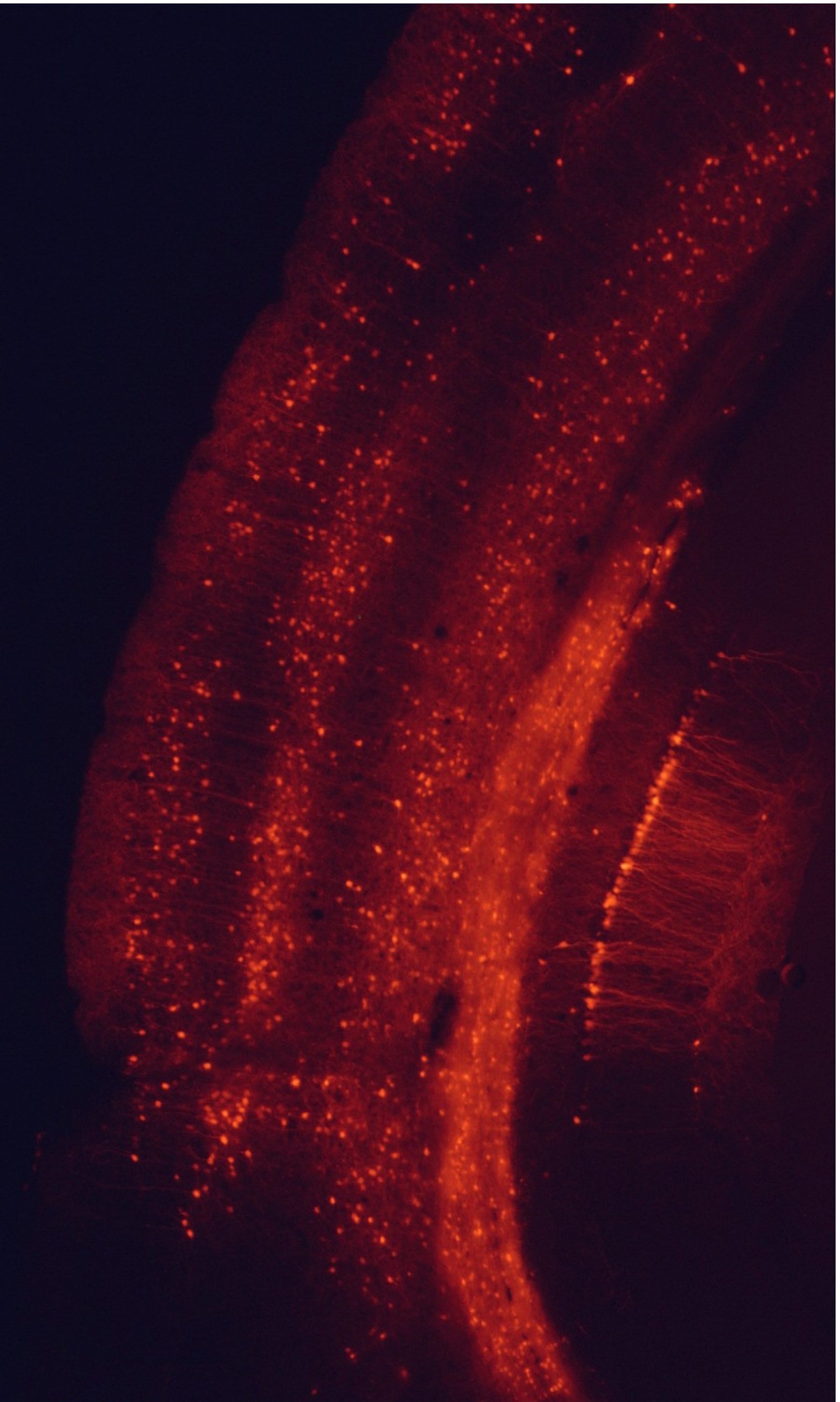


As it is, backprop through time requires time-stamped records of activity patterns and inputs – not an easy ask for a group of real neurons



There might be some ways to address this, but I'm gonna leave this as an unresolved problem for now...

11. Conclusion

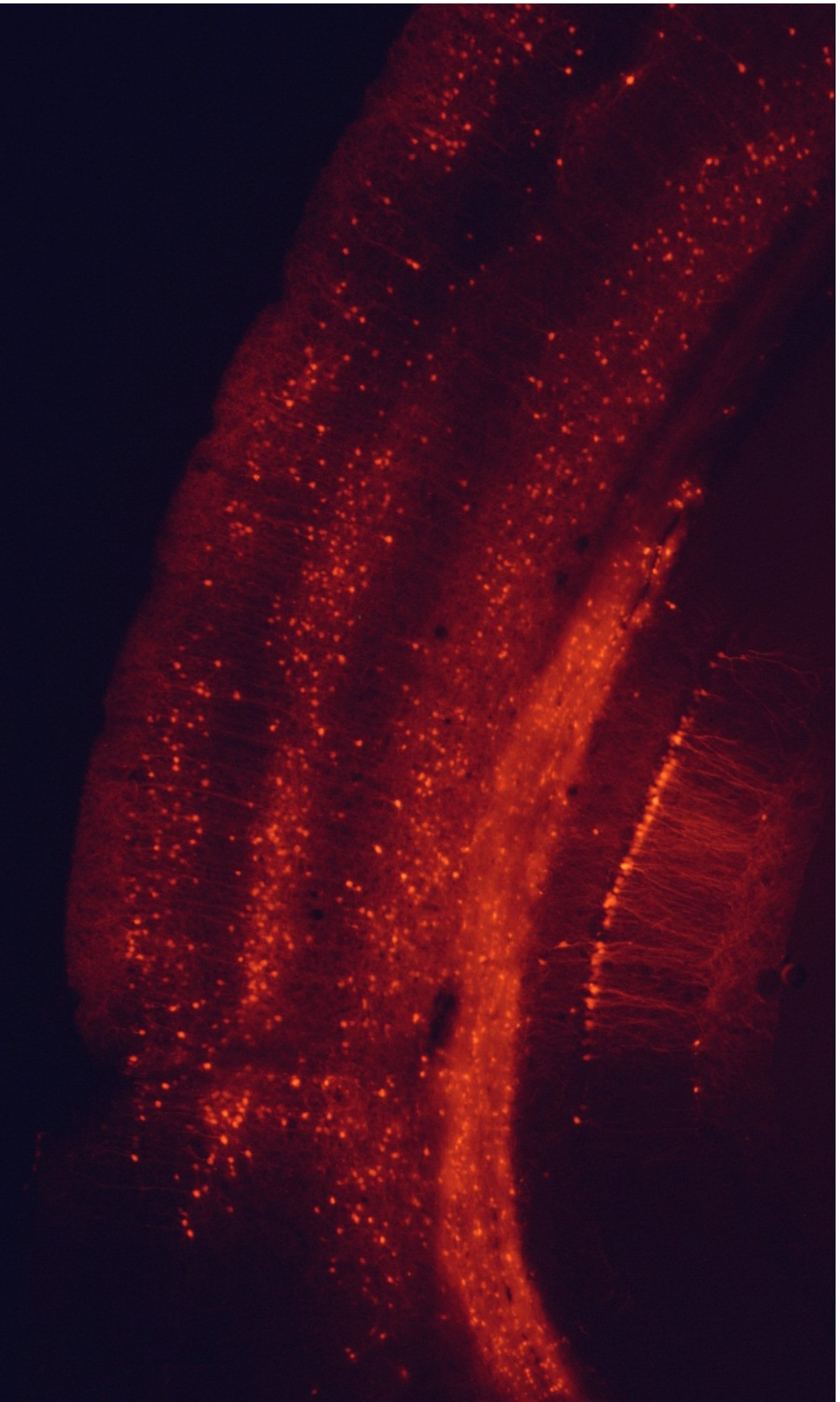


There's a good reason that deep learning has taken over AI – *it works*

The reasons it works in AI apply equally to our own brains – evolution must have come to use some end-to-end optimization because learning in such high-D space is too hard otherwise (too many directions!)

Our brains may not do backprop specifically, but we are getting to the point where the old objections that prevented us from trying to link deep learning and neuroscience no longer hold water

11. Conclusion



I'll leave you with the following cool/scary thought...

If our brains are approximating gradient descent, and we can determine the signals used to estimate the gradient, then in the future (with good enough neural prostheses) we could do backprop through an AI back into the brain!

This could give us seamless brain-AI interfaces, which would open the door to some pretty crazy tech...

Questions?

Deep learning summer school
Toronto 2018

Special thanks to:
Jordan Guerguiev (U of Toronto)
Timothy Lillicrap (Google DeepMind)
Richard Naud (U of Ottawa)
Thomas Mesnard (MILA)

Financial support

NSERC CRSNG

HFSP

Google

CIFAR CANADIAN INSTITUTE FOR ADVANCED RESEARCH | ICRA INSTITUT CANADIEN DE RECHERCHES AVANCÉES

LiNC LAB
linclab.org

UNIVERSITY OF TORONTO SCARBOROUGH