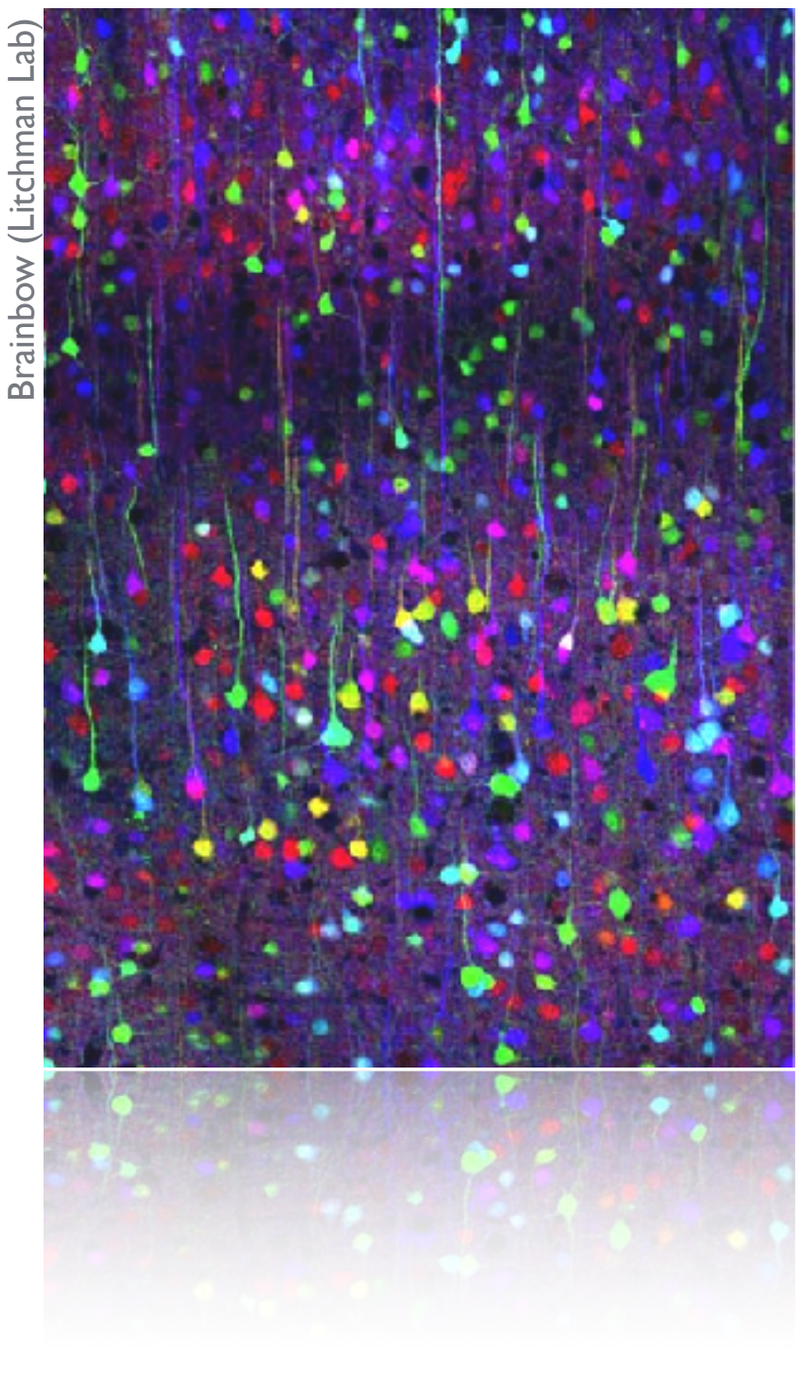
# Bio-deep learning workshop

University of BRISTOL



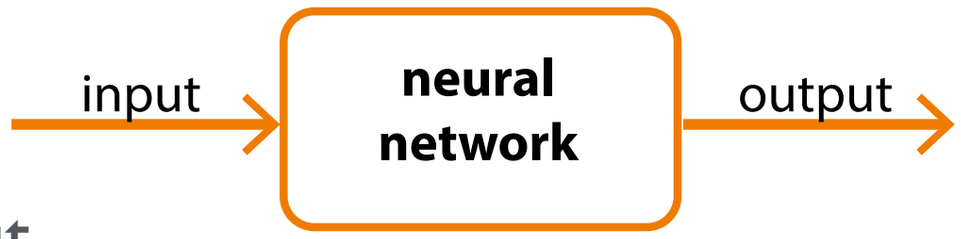

## Neural circuits and learning:

Supervised learning:
backprop & cerebellum

# Previously on *IPB*…

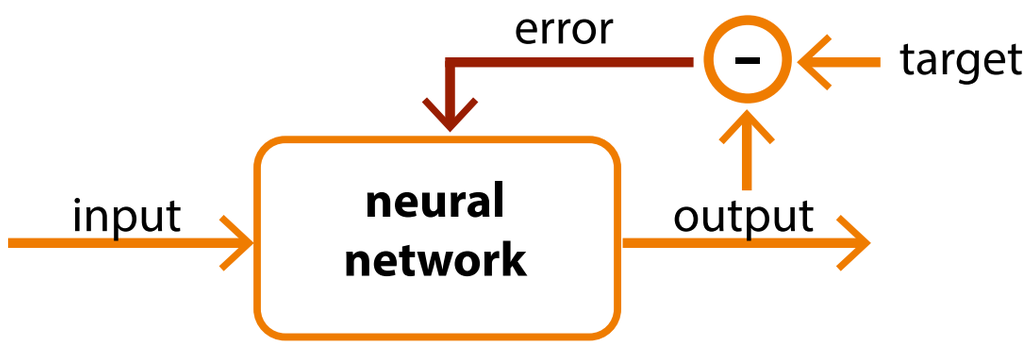
## Unsupervised Learning:

Extracts useful representations of input



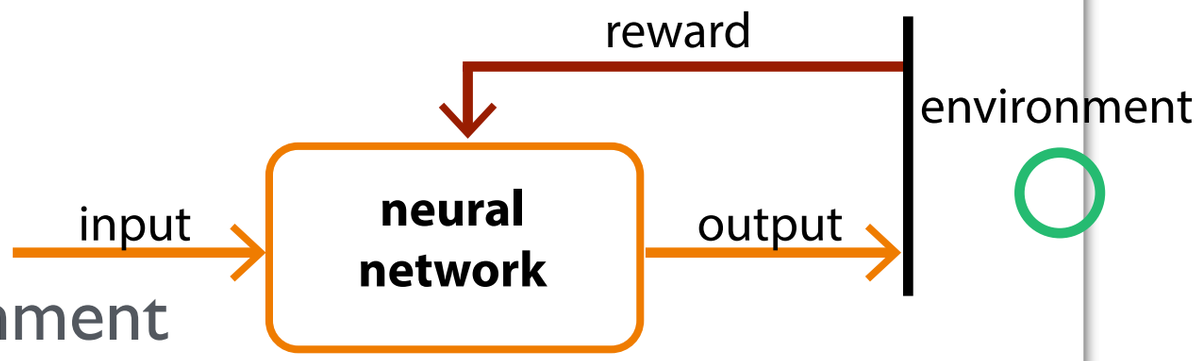

## Supervised Learning:

Relies on a teaching signal




## Reinforcement Learning:

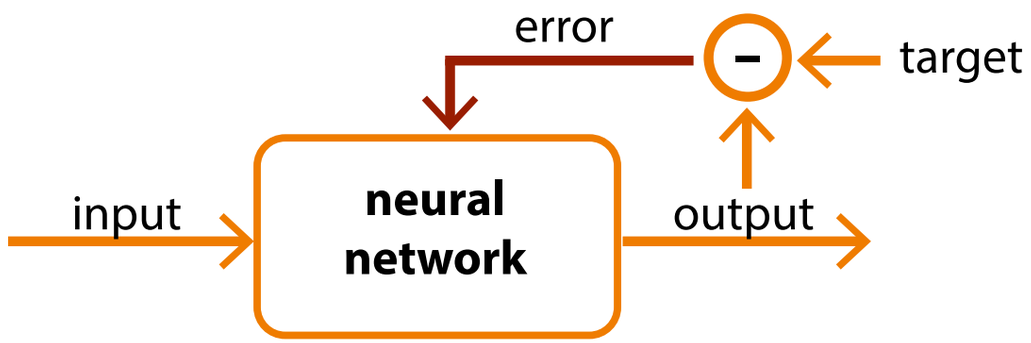
Learn to navigate/survive an environment



# But, how to exactly train neural networks?

## Supervised Learning:

Relies on a teaching signal




Solution: The backprop algorithm which is the workhorse of deep learning!

**Note**: Backprop is a general optimization algorithm, that can also be used in unsupervised and reinforcement learning

# Outline

1. **The backpropagation algorithm**

2. **Backprop in the brain**

3. **Does the cerebellum implement supervised learning?**

# The backpropagation of errors algorithm

- **Backpropagation** is an algorithm used to train artificial neural networks. It calculates a gradient from the error function with respect to a given parameter (i.e. weight). With this gradient errors are effectively back propagated during training throughout the network.

$$\Delta w = \eta(v - \text{target})u$$

- It is a generalisation of the **delta rule** (above) to deep feedforward networks, which relies on the use of the chain rule to compute gradients for each layer (see next slides).

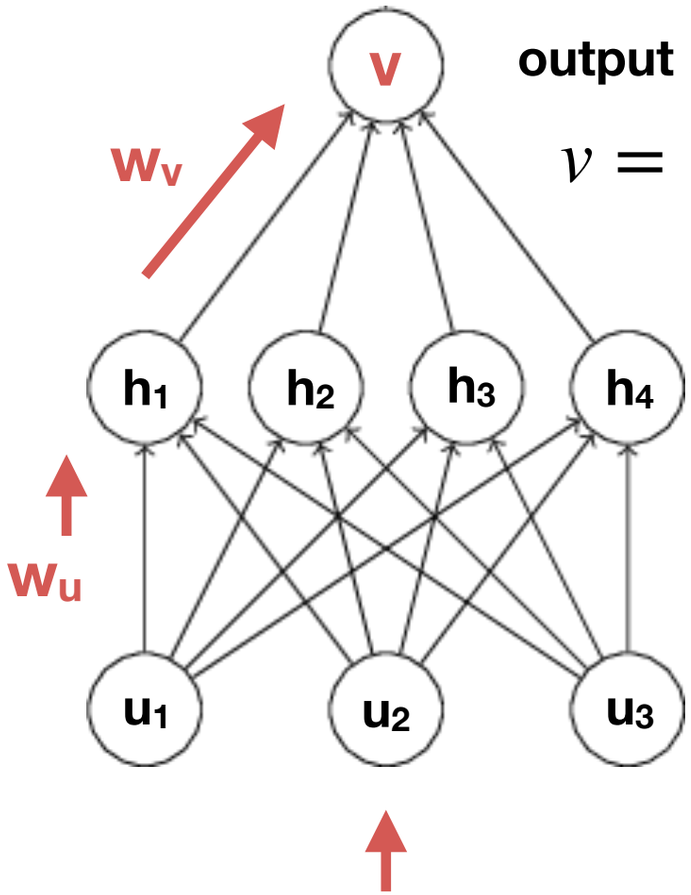
- **Backpropagation** is typically used to perform gradient descent optimisation, adjusting the weights of neurons while optimising a desired loss function (error).

Rumelhart et al. 1986 PDP

# The backpropagation algorithm

**The backpropagation algorithm:**

**1.** Calculate forward activity




$$v = W_v h$$

**Note:** Here, we are assuming a linear activation for ***v***

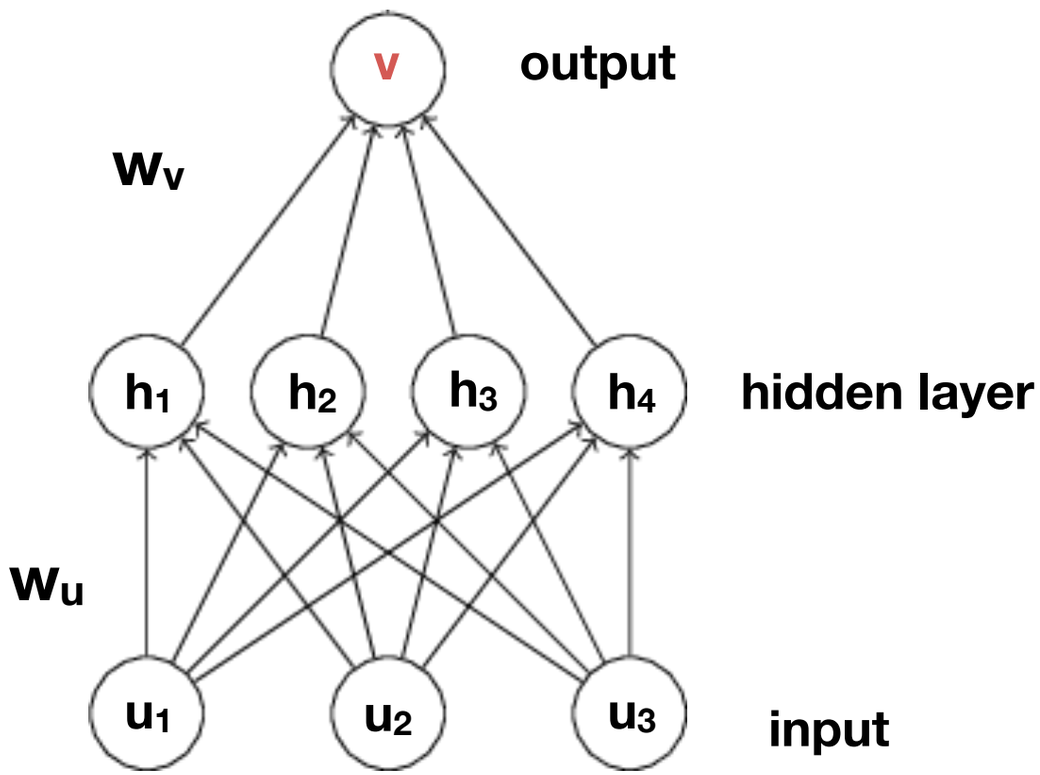
$$h = f(W_u u)$$

***f*** denotes the activation function (i.e. input-output function), which is typically modelled as a sigmoid or rectifying linear unit function (ReLU)

Rumelhart et al. 1986 PDP

# The backpropagation algorithm

$$\text{error} = \frac{1}{2}(v - \text{target})^2$$

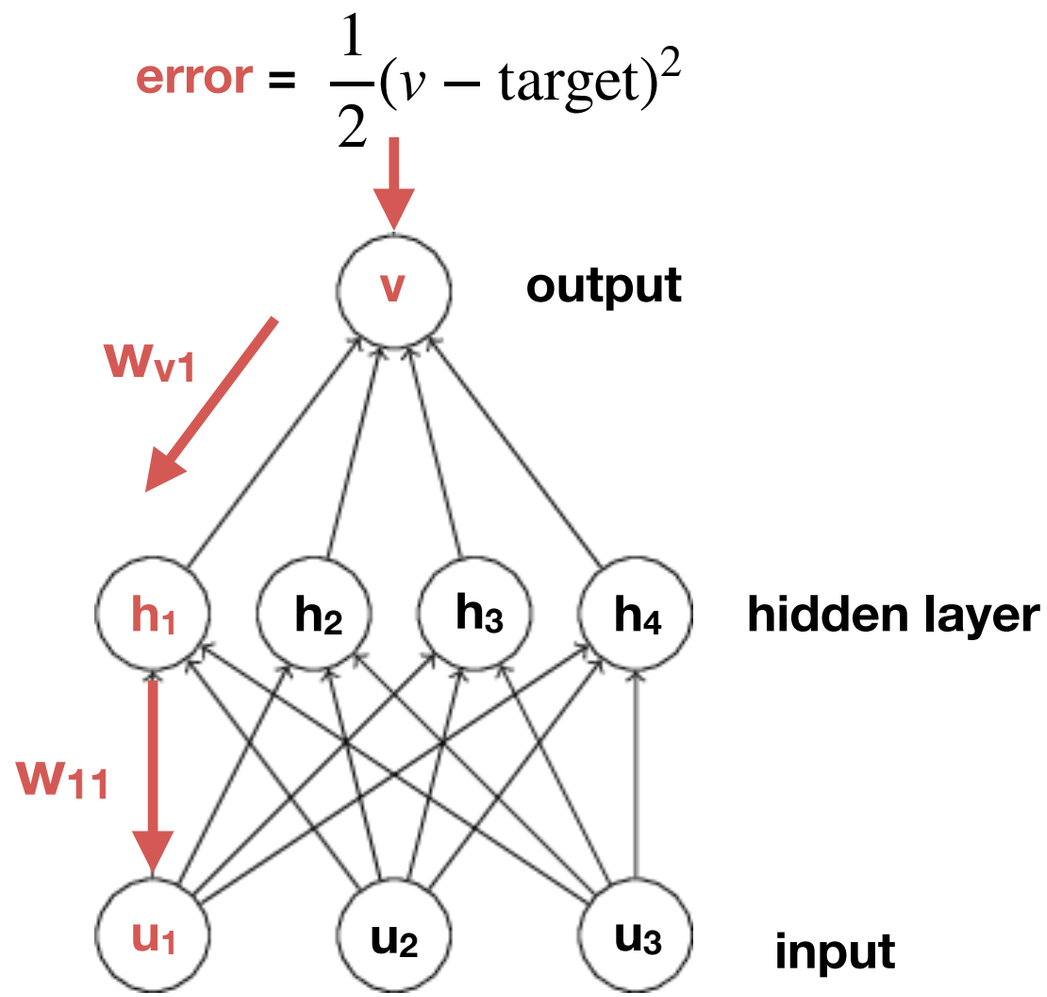



**The backpropagation algorithm:**

1. Calculate forward activity
2. Calculate **error**

Rumelhart et al. 1986 PDP

# The backpropagation algorithm




**The backpropagation algorithm:**

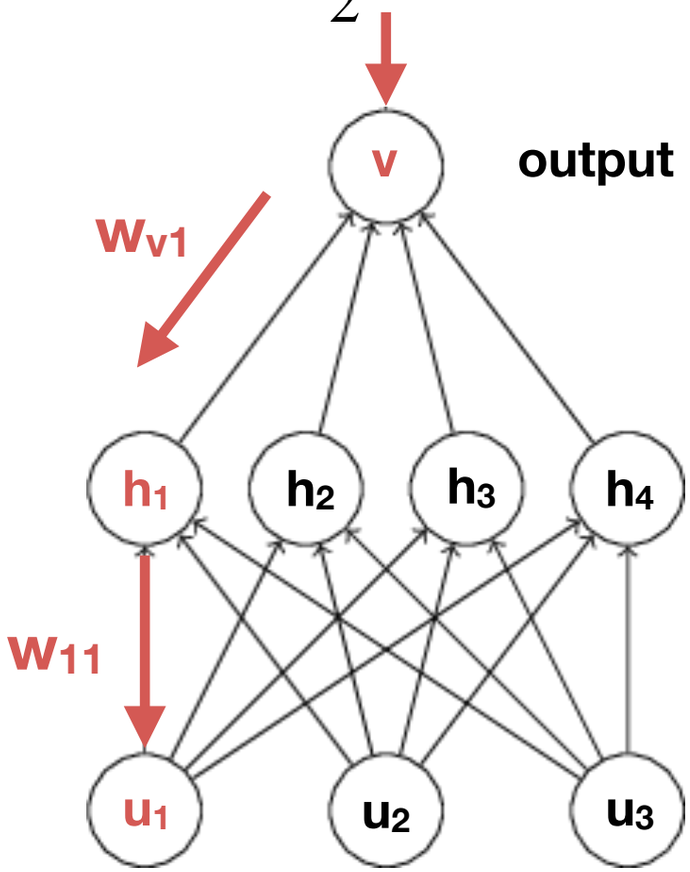
1. Calculate forward activity
2. Calculate error
3. **Backpropagate** error, e.g. how to calculate the gradient wrt $w_{11}$:

$$\frac{\partial \text{Error}}{\partial w_{11}} = \underbrace{\frac{\partial \text{Error}}{\partial v}\frac{\partial v}{\partial h_1}\frac{\partial h_1}{\partial w_{11}}}_{\text{chain rule}}$$

Rumelhart et al. 1986 PDP

# The backpropagation algorithm

$$\textbf{error} = \frac{1}{2}(v - \text{target})^2$$




**Note:** Write down the gradient wrt a given weight for practice!

**The backpropagation algorithm:**

1. Calculate forward activity
2. Calculate error
3. **Backpropagate** error, e.g. how to calculate the gradient wrt $w_{11}$:

$$\frac{\partial \text{Error}}{\partial w_{11}} = \underbrace{\frac{\partial \text{Error}}{\partial v}\frac{\partial v}{\partial h_1}\frac{\partial h_1}{\partial w_{11}}}_{\text{chain rule}}$$

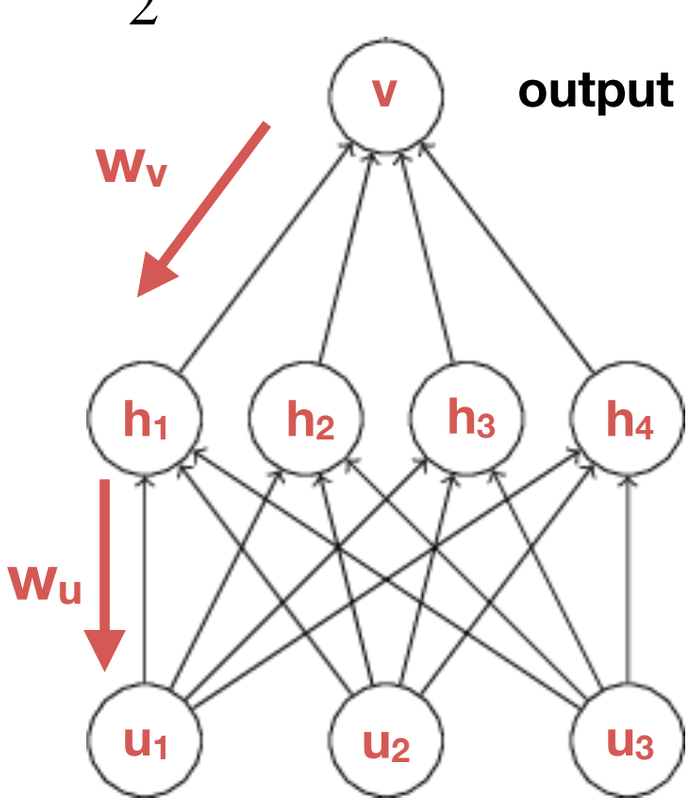
**weight update:**

$$\Delta w_{11} = -\eta \frac{\partial \text{Error}}{\partial w_{11}} = \underbrace{(\text{v} - \text{target})}_{\text{error}}\, w_{v1} f'_{h_1} u_1$$

Rumelhart et al. 1986 PDP

# The backpropagation algorithm general weight update

**error =** $\frac{1}{2}(v - \text{target})^2$




**The general form for the weight update:**

$$\Delta w_u = -\eta \underbrace{(\text{v} - \text{target})}_{\text{error}} \; w_v^T f_h' u$$

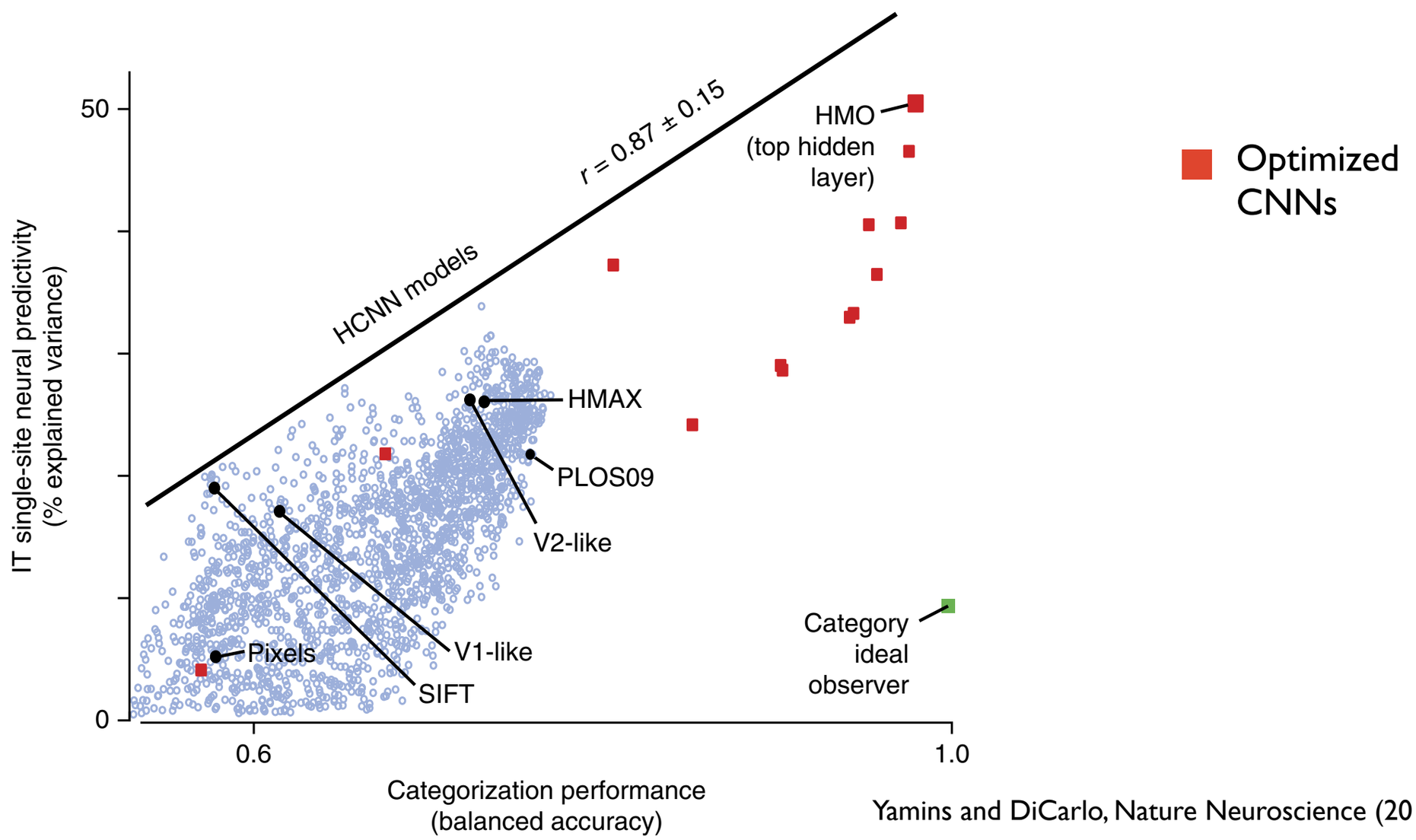
**Note:** Write down the gradient wrt a weight for practice!

Rumelhart et al. 1986 PDP

# The backpropagation algorithm in the brain

**Next lecture**: Neural nets trained with backprop can predict neural data to a high degree!

So should we consider backprop as a potential mechanism for learning in the brain?




Yamins and DiCarlo, Nature Neuroscience (2016)

# Backprop in the brain
# and
# why should we care?

This would provide for the **first time** a **good model of learning in the brain**!

Once we have done this mapping, we can then use the brain as **inspiration** for better/**more robust optimisation algorithms**!

# The backpropagation algorithm
## is **NOT** = backpropagating action potential

In biology there is a phenomenon known as action potential backpropagation.
This does **NOT** directly relate to the backpropagation algorithm!

**Backpropagating action potential ≠ Backpropagation algorithm**

see Stuart et al. TiNS (1997) for more info

# The backpropagation algorithm in the brain

However, at a first glance the backpropagation algorithm seems to not be biologically plausible.

**Key issues:**

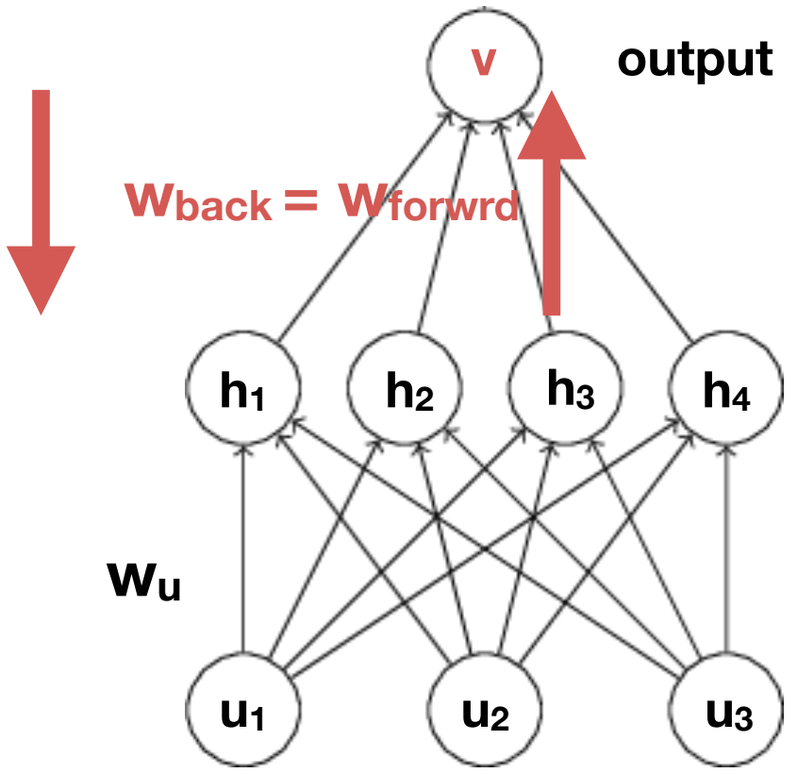
1. **Weight transport problem**: Suggests that feedback weights = feedforward weights
2. **Derivative of activation functions**: Relies on calculating derivatives of the activation functions
3. **Two phase learning**: (1) feedforward propagation of activity and (2) error backpropagation
4. **Separate error network**: Suggest the need for a separate biological network computing the learning rules
5. **Non-local learning rules**: The weight update depends on non-local information
6. **Target**: The target label guides supervised learning

Useful references:
Roelfsema and Holtmaat, Nature Neuroscience Rev (2018)
Richards and Lillicrap, Current Opinion in Neurobiology (2019)

# The backpropagation algorithm
## and why it is at odds with biology
### 1. Weight transport problem

Backprop: The chain rule predicts feedback weights that are equal to feedforward weights: $W_v^T = W_v$

Biology: No evidence that feedback connections exactly mirror feedforward ones.



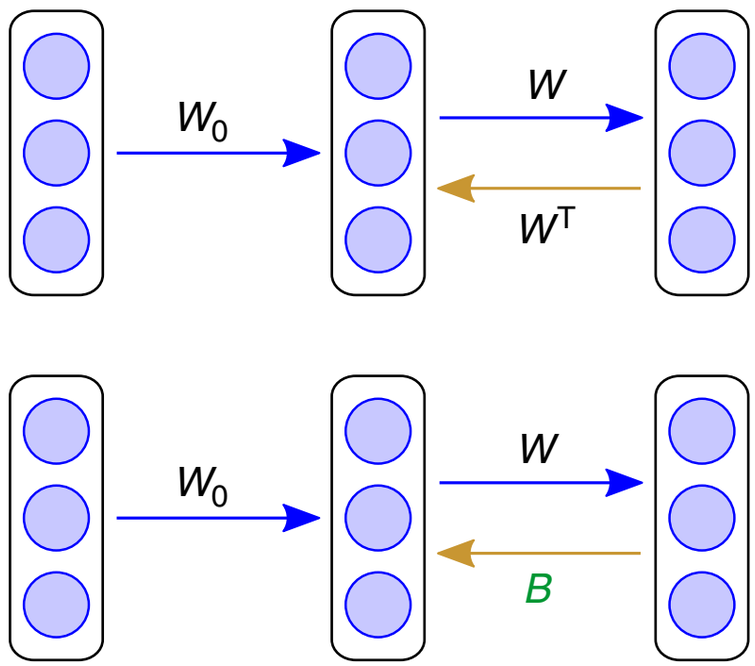

$$\Delta w_u = -\eta \underbrace{(\mathrm{v} - \mathrm{target})}_{\text{error}} \; w_v^T f_h' u$$

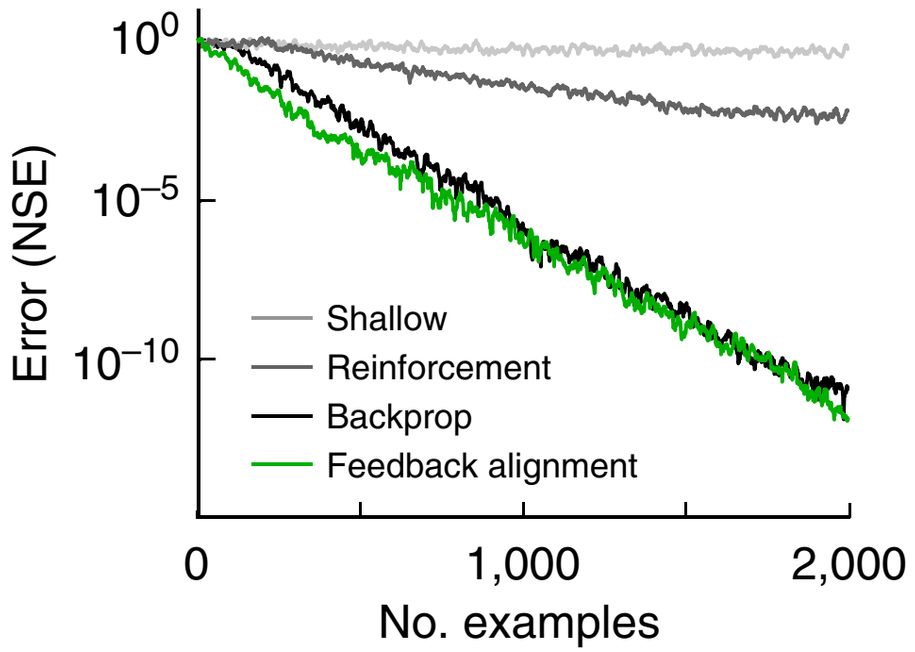
**Backprop/Feedback weights = Feedforward weights**

$$v = W_v h$$

# The backpropagation algorithm
## and why it is at odds with biology
### 1. Weight transport problem

**Solution**: Unclear; but *Lillicrap et al.* found that exact feedback weights can be replaced by random weights, without a major impact on performance. This method is known as **feedback alignment**, and suggests that the brain may not need an exact feedback signal to implement 'backprop'.




**B** is a random matrix, replacing $W^T$

Lillicrap et al., Nature Comms (2016)

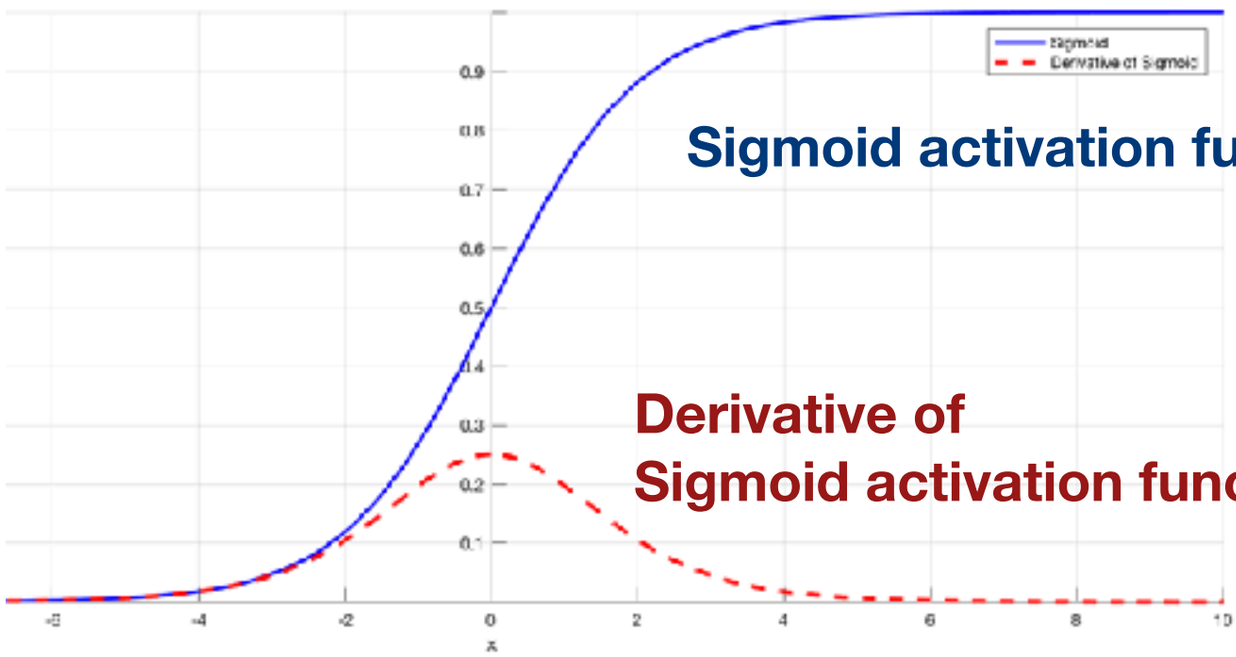
# The backpropagation algorithm
## and why it is at odds with biology
### 2. Derivative of activation function

Backprop: Relies on the derivative of the activation function.

Biology: Not clear how neurons could calculate their own derivative.

**Solution**: Unclear; but as long as it doesn't impact on the gradient direction this derivative is not critical for performance (similar to feedback alignment).

$$\Delta w_u = -\eta \underbrace{(\mathrm{v} - \mathrm{target})}_{\text{error}} \; w_v^T f_h' u$$




**Sigmoid activation function**

**Derivative of Sigmoid activation function**

**Derivative of activation function**

# The backpropagation algorithm
## and why it is at odds with biology
### 3. Two-phase learning

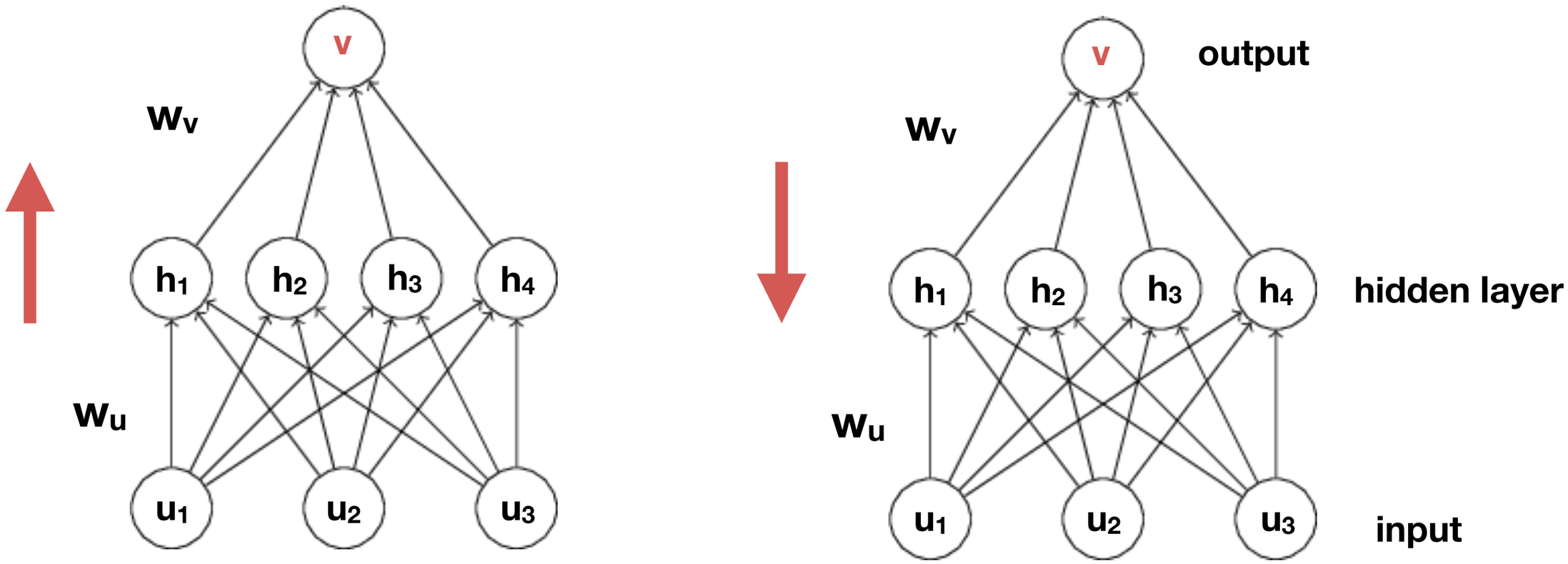
Backprop: The backprop relies on **two phases**, first a feedforward pass of the activity and then a backward pass of errors.

Biology: In the brain there is no clear separation between perception and learning.

**Forward phase**
(perception/inference)

**Backward phase** (learning)

$$\text{error} = \frac{1}{2}(v - \text{target})^2$$



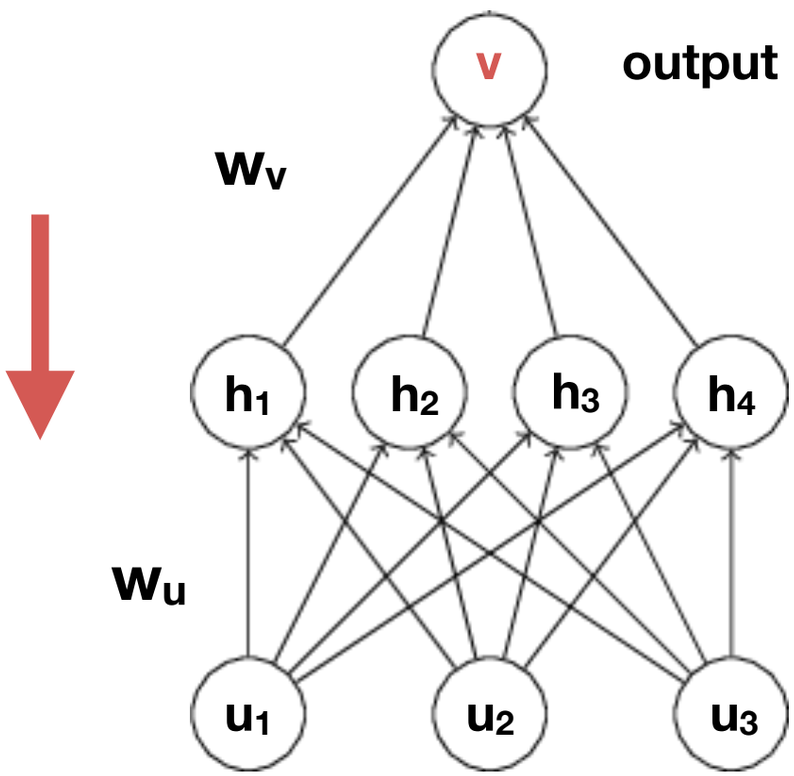
# The backpropagation algorithm
## and why it is at odds with biology
### 4. Separate error network

Backprop: The chain rule can be seen as a separate (error/gradients) network.

Biology: Minimal evidence for the existence of such gradient or error networks.

**Backward phase ~ error network (EN)**

error = $\frac{1}{2}(v - \text{target})^2$




EN weights

$$\Delta w_u = -\eta \underbrace{(\text{v} - \text{target})}_{\text{error}} w_v^T f_h' u$$

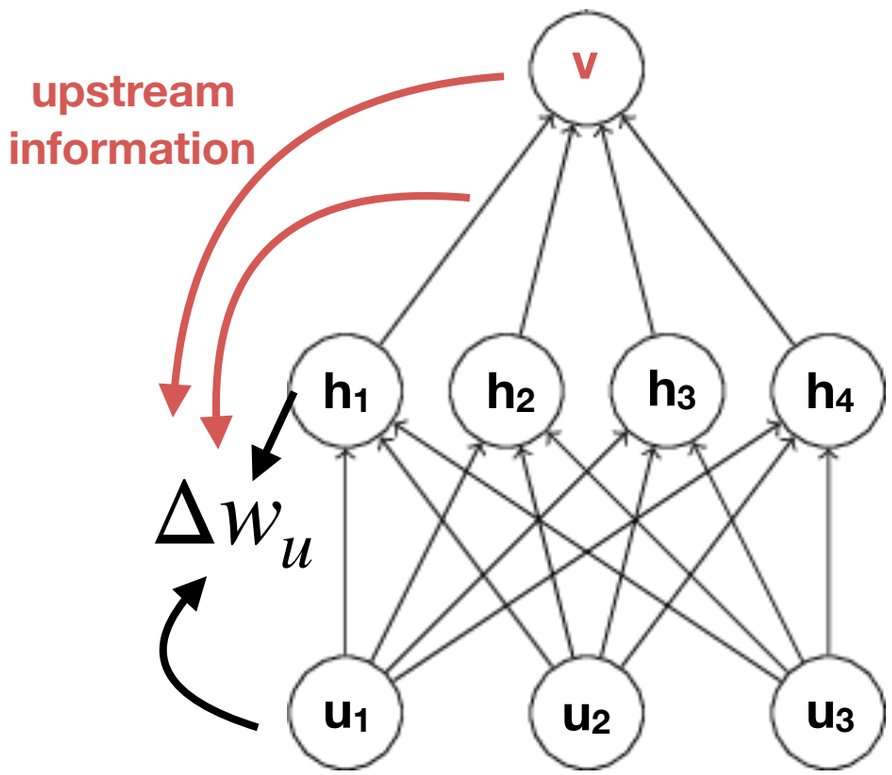
EN activation function

# The backpropagation algorithm
## and why it is at odds with biology
### 5. Non-local learning rules

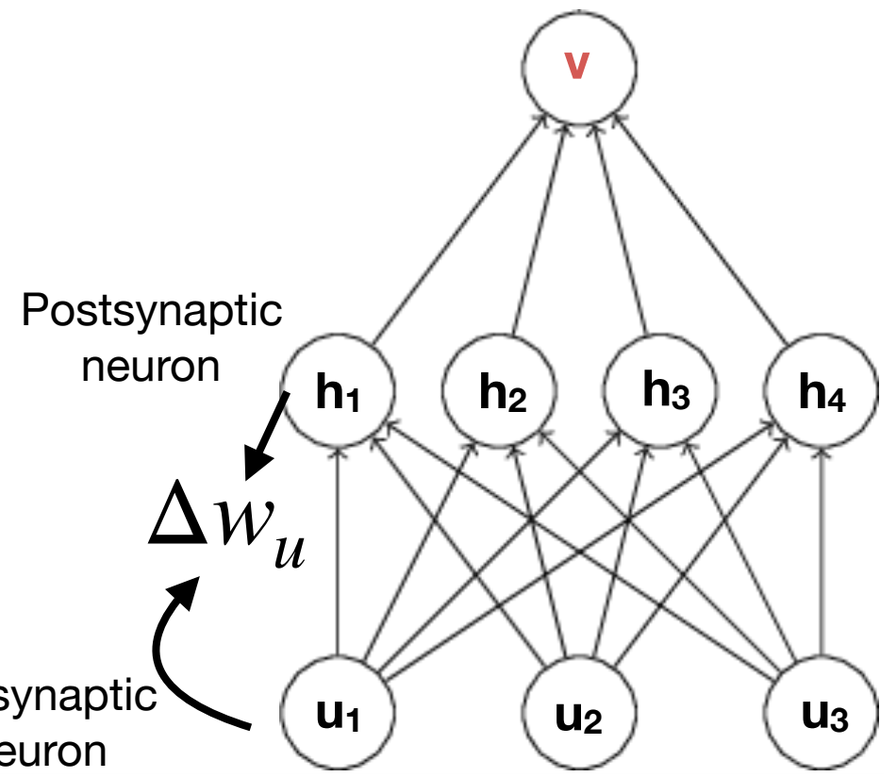
Backprop: Relies on non-local learning rules (i.e. a given neuron needs information about other neurons and layers).

Biology: Learning is believed to occur through local changes at synapses, a process known as synaptic plasticity which can only access locally available information (i.e. within the pre and postsynaptic neuron).

**Non-local learning rule**
(backprop)




**Local learning rule**
(biology)



# Quiz time!

**Please go to BB**
**and solve quiz 2.**

It should take you just a couple of minutes.

# The backpropagation algorithm in the brain

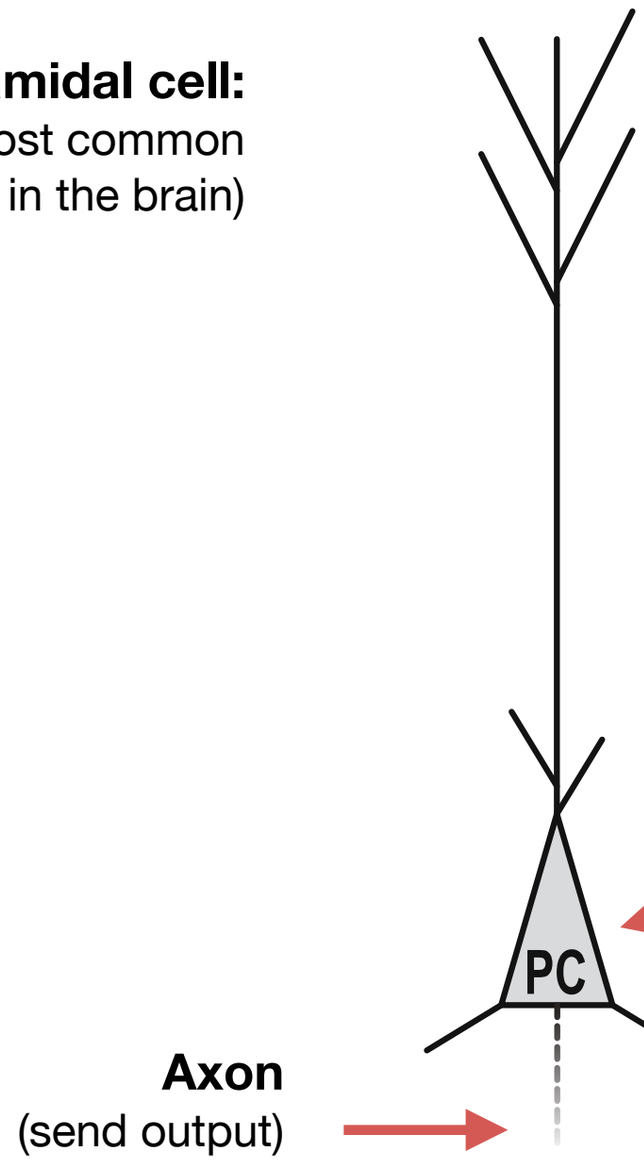
**How to solve:**

1. **Weight transport problem**: It proposes feedback weights equal to feedforward
2. **Derivative of activation functions**: Relies on calculating derivatives of the activation functions
3. **Two phase learning**: (1) feedforward propagation of activity and (2) error backpropagation
4. **Separate error network**: Suggest the need for a separate biological network computing the learning rules
5. **Non-local learning rules**: The weight update depends on non-local information
6. **Target**: The target label guides supervised learning

# The backpropagation algorithm in the brain

## Inspiration from real neurons: pyramidal cells

**Pyramidal cell:**
(one of the most common neurons in the brain)




**Distal dendrites**
(receive feedback/topdown input)

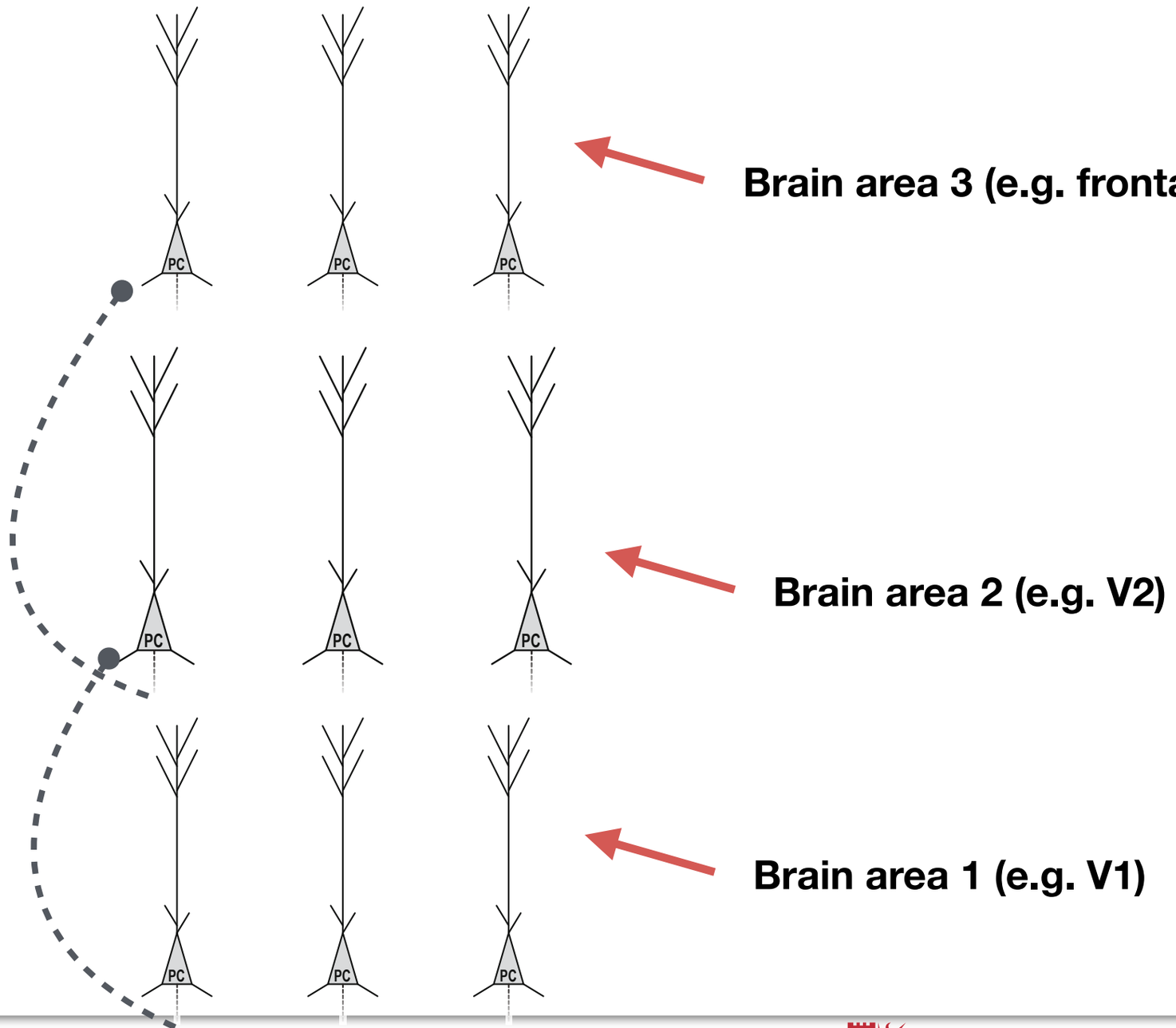
**Soma**
(integrates input from all dendrites)

**Proximal dendrites**
(receive feedforward/bottom-up input)

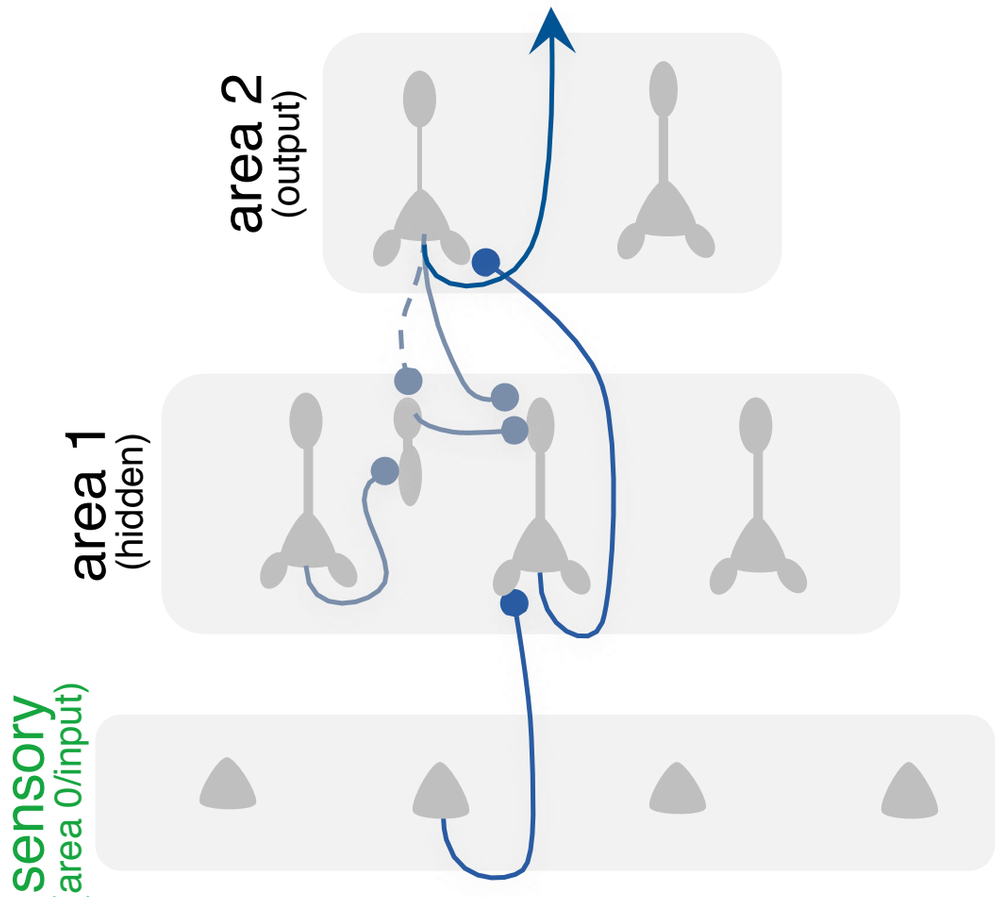
**Axon**
(send output)

# The backpropagation algorithm in the brain

## Inspiration from real neurons: pyramidal cells

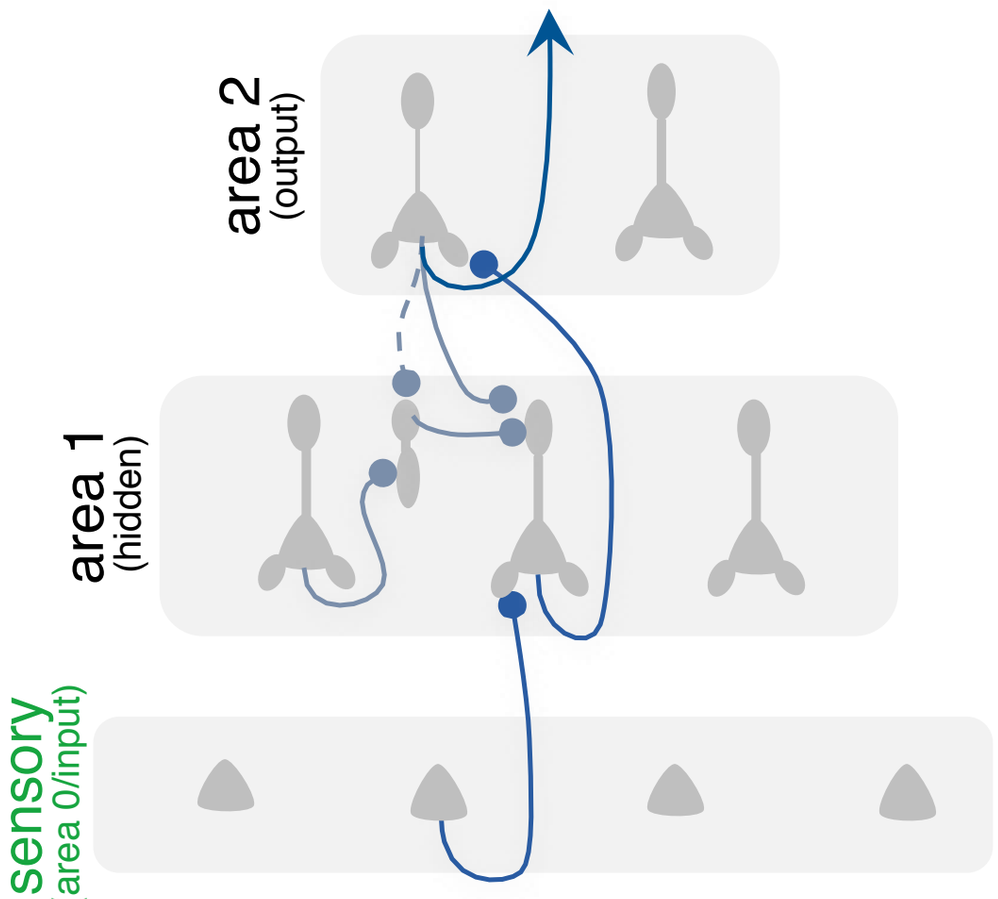


# Dendritic microcircuits approximate backprop



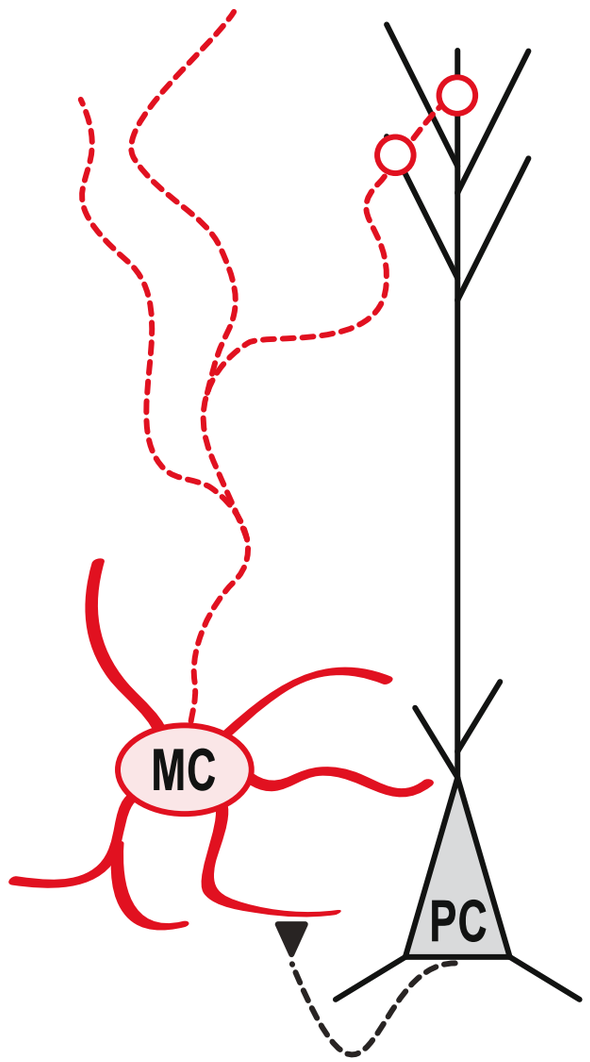

Sacramento et al. 2018 (NIPS)

# Dendritic microcircuits approximate backprop



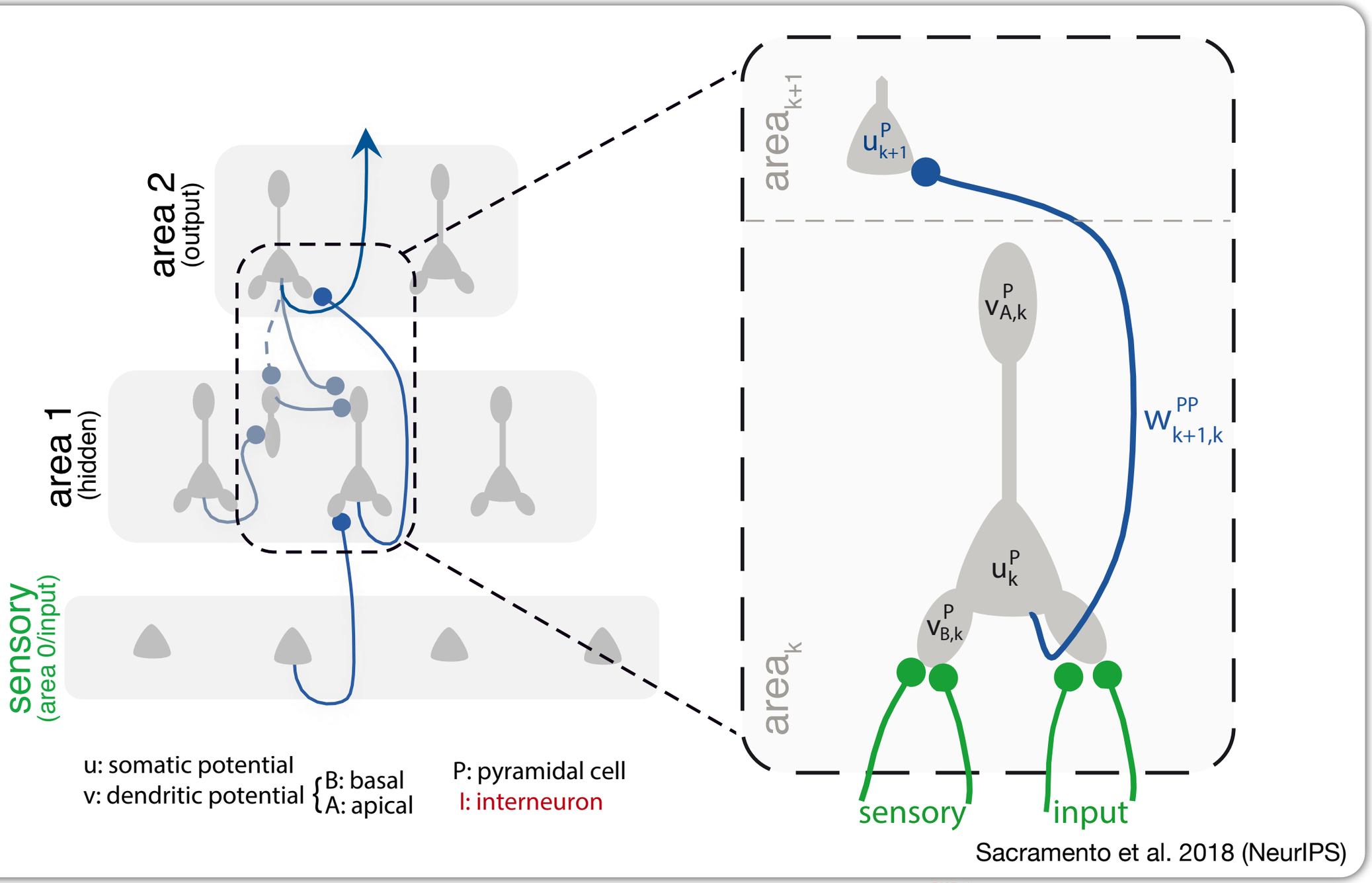

Inspired by two key cell types in the cortex:

- **Pyramidal cells** (excitatory cell)
- **Martinotti cells** (inhibitory cell)



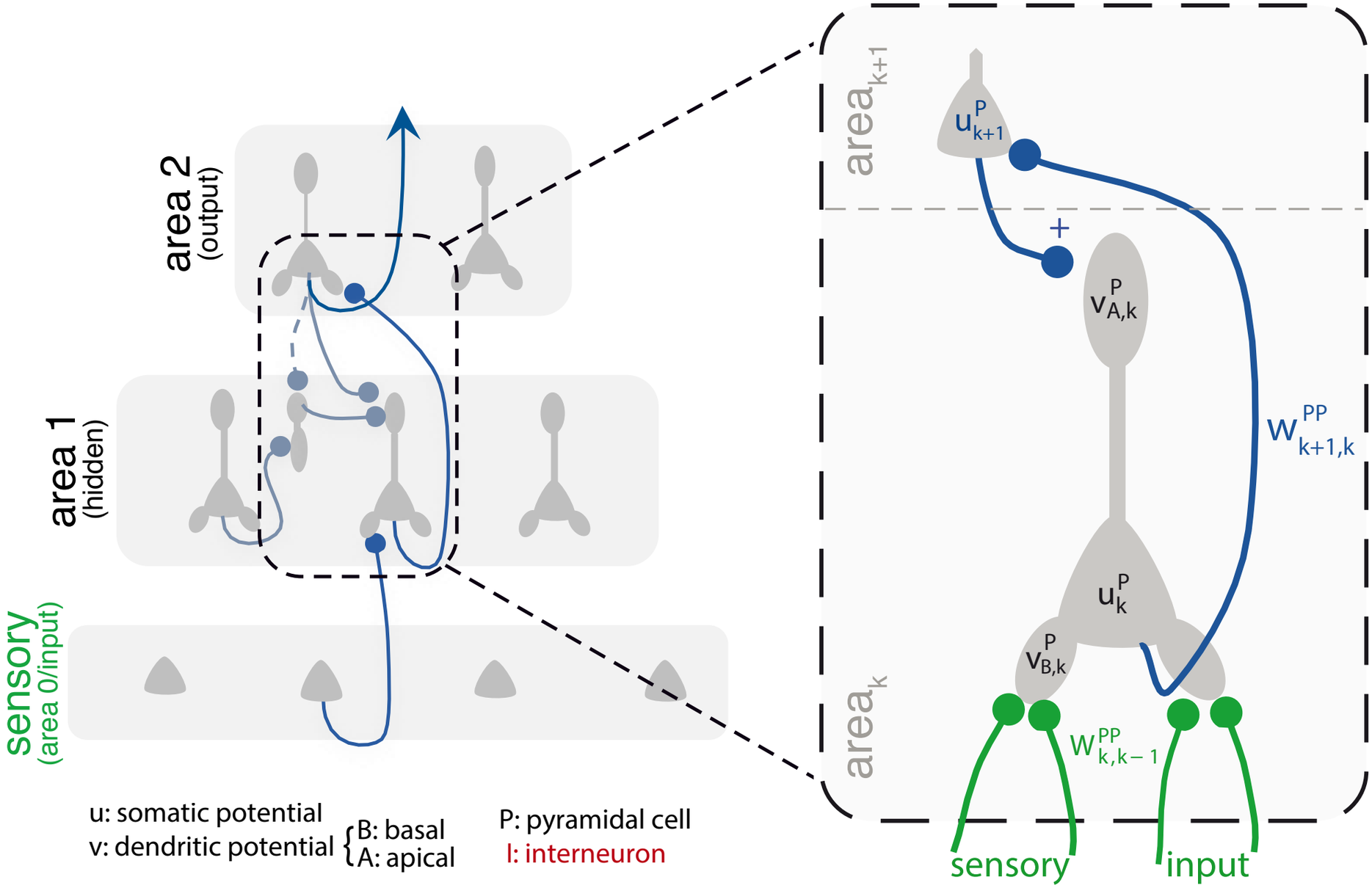

Sacramento et al. 2018 (NeurIPS)

# Dendritic microcircuits approximate backprop



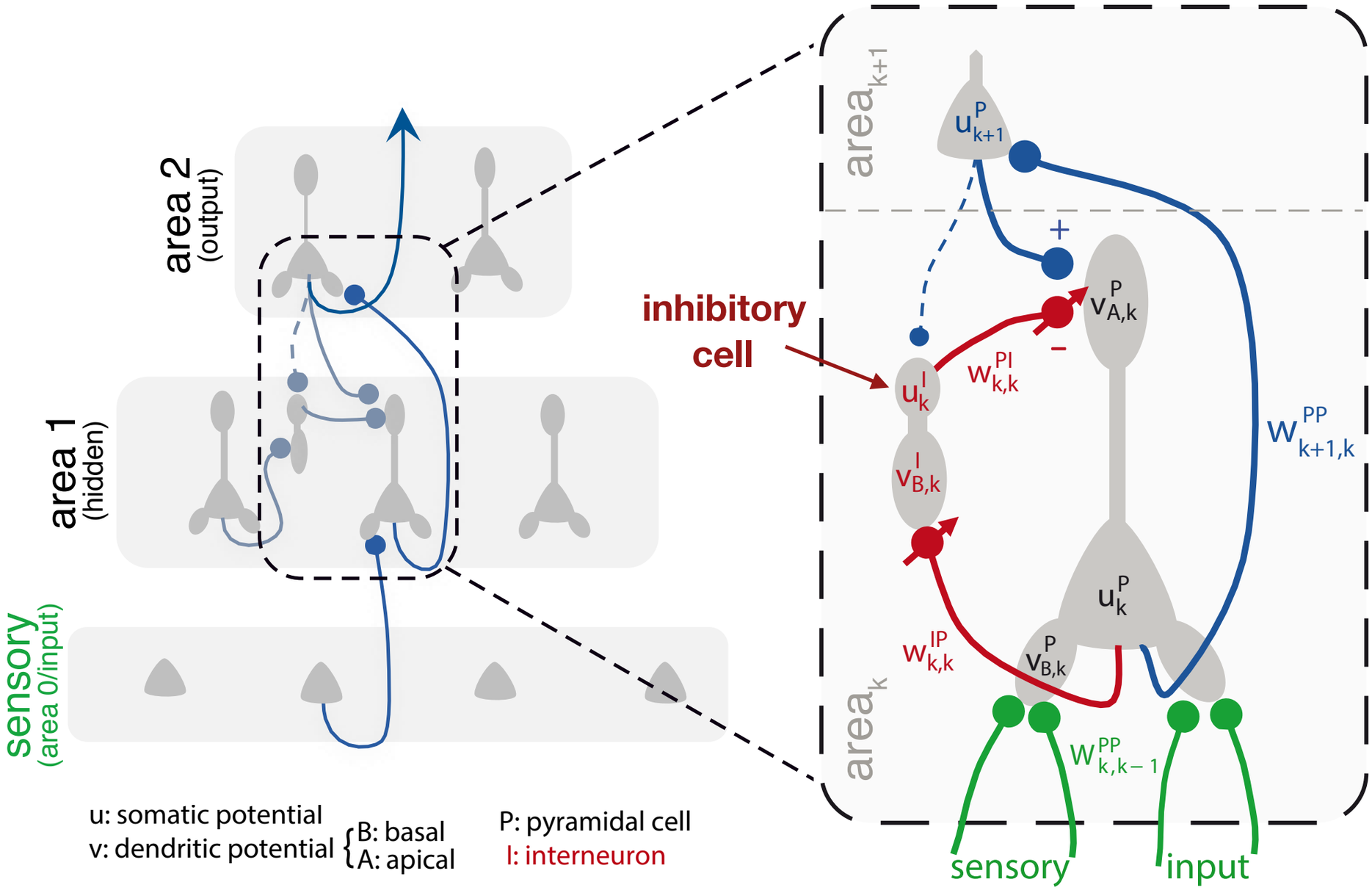

Sacramento et al. 2018 (NeurIPS)

# Dendritic microcircuits approximate backprop



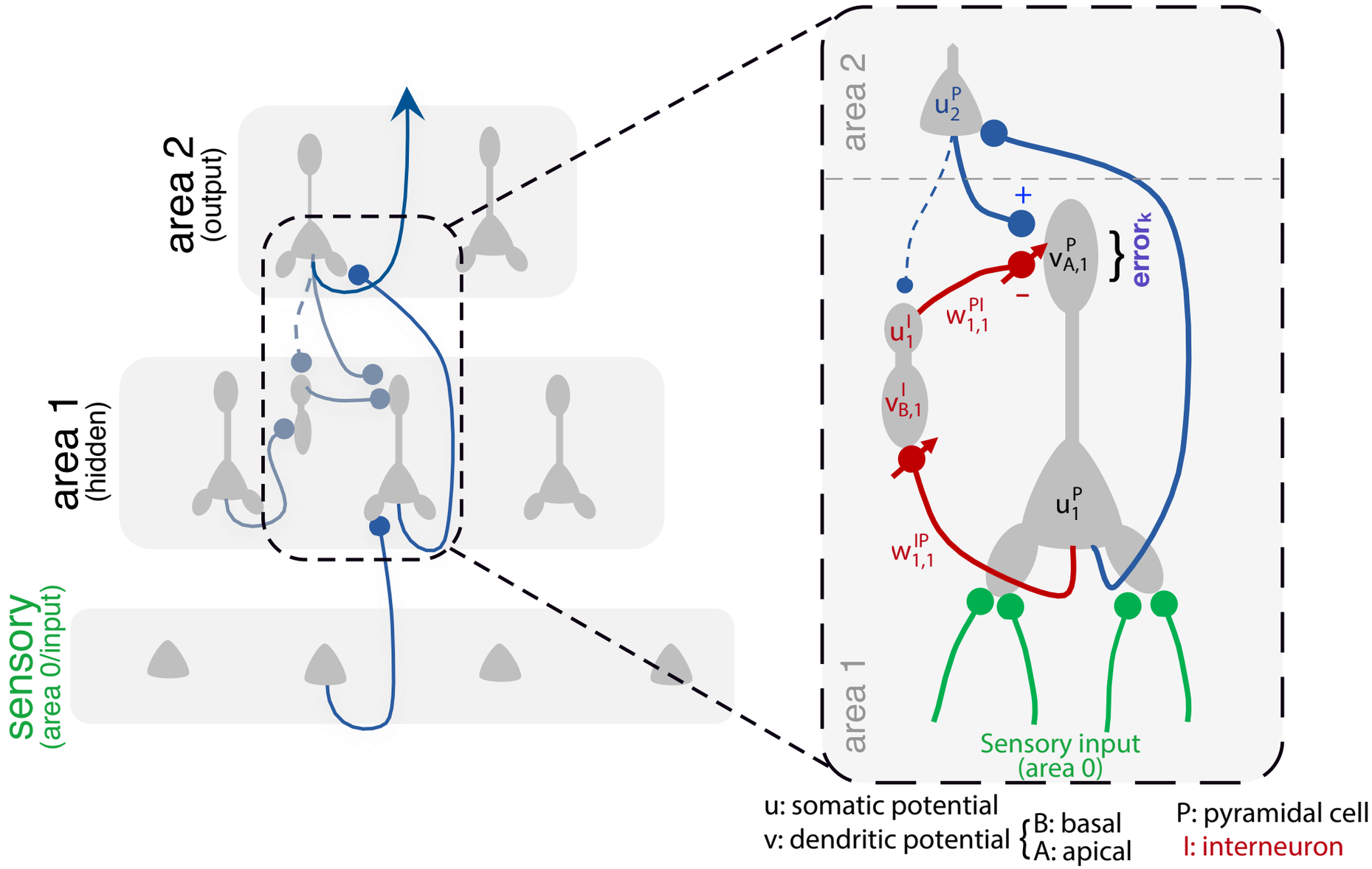

Sacramento et al. 2018 (NeurIPS)

# Dendritic microcircuits approximate backprop




Sacramento et al. 2018 (NeurIPS)

# Dendritic microcircuits approximate backprop

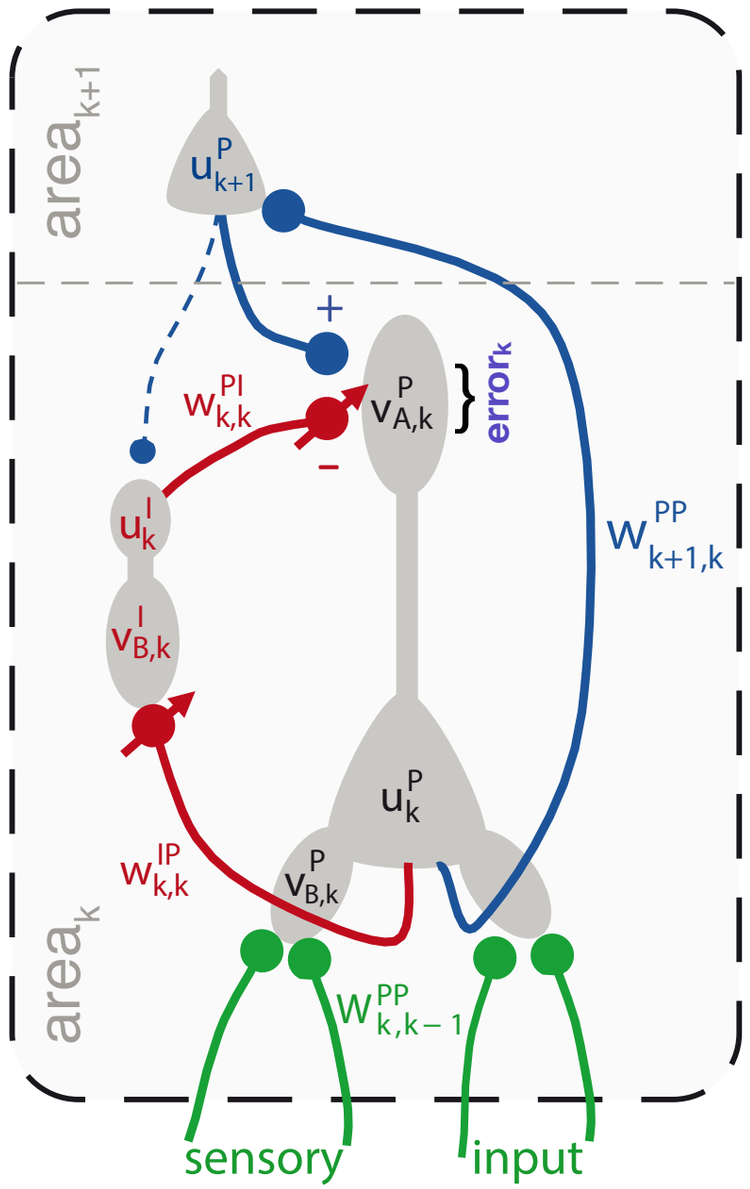



Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits

## local error-correcting learning rules

**Important note**: For this model *u* and *v* have a different meaning from before!




**General idea**:

1. Difference between topdown activity from $u^P_{k+1}$ and lateral inhibition $u^I_k$ generates an **error** signal:

$$v^P_{A,k} \sim u^P_{k+1} - u^I_k$$

The **error** signal triggers changes in bottom weights:

$$w^{PP}_{k,k-1}$$

2. The model learns recursively across areas/layers (as in backprop) to correctly predict an output. During this process lateral inhibition learns to predict the top-down input, leading to a zero error after learning, i.e.:

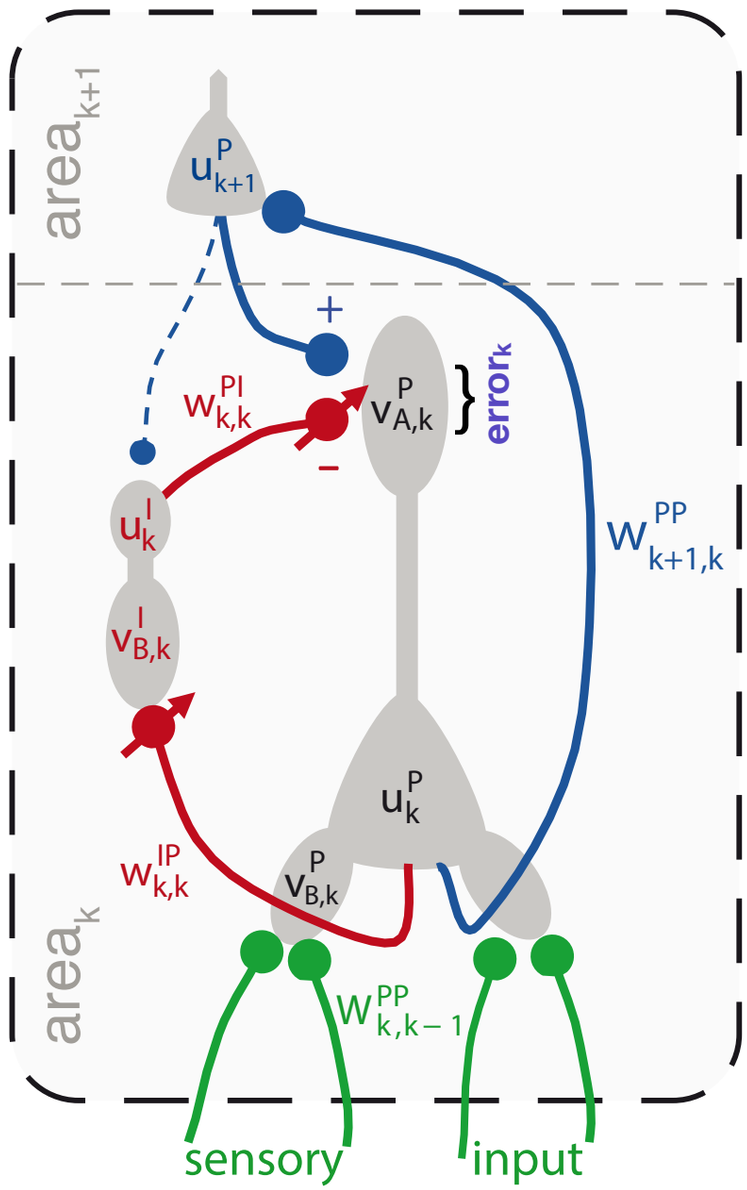
$$u^I_k \sim u^P_{k+1}$$

Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits

## local error-correcting learning rules

**Important note**: For this model *u* and *v* have a different meaning from before!




General form of the (local) learning rule:

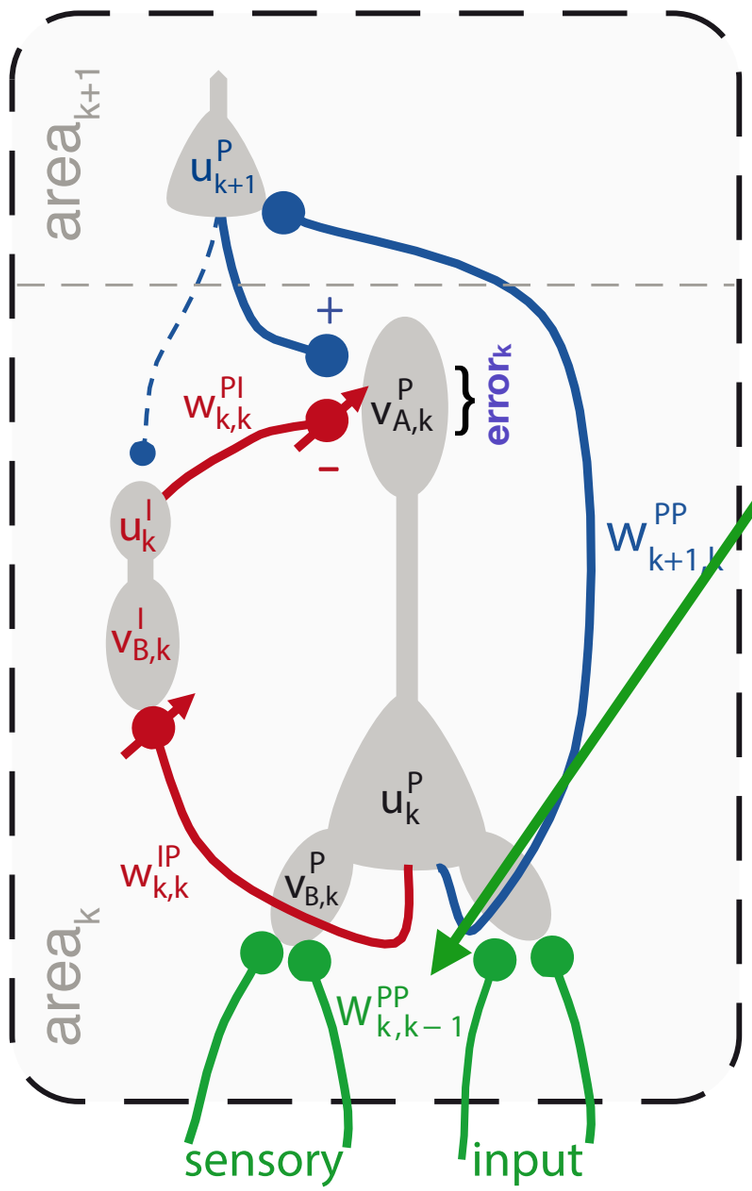
$$\frac{d}{dt}w \sim \eta(u - v)r$$

Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits

## local error-correcting learning rules

**Important note**: For this model *u* and *v* have a different meaning from before!



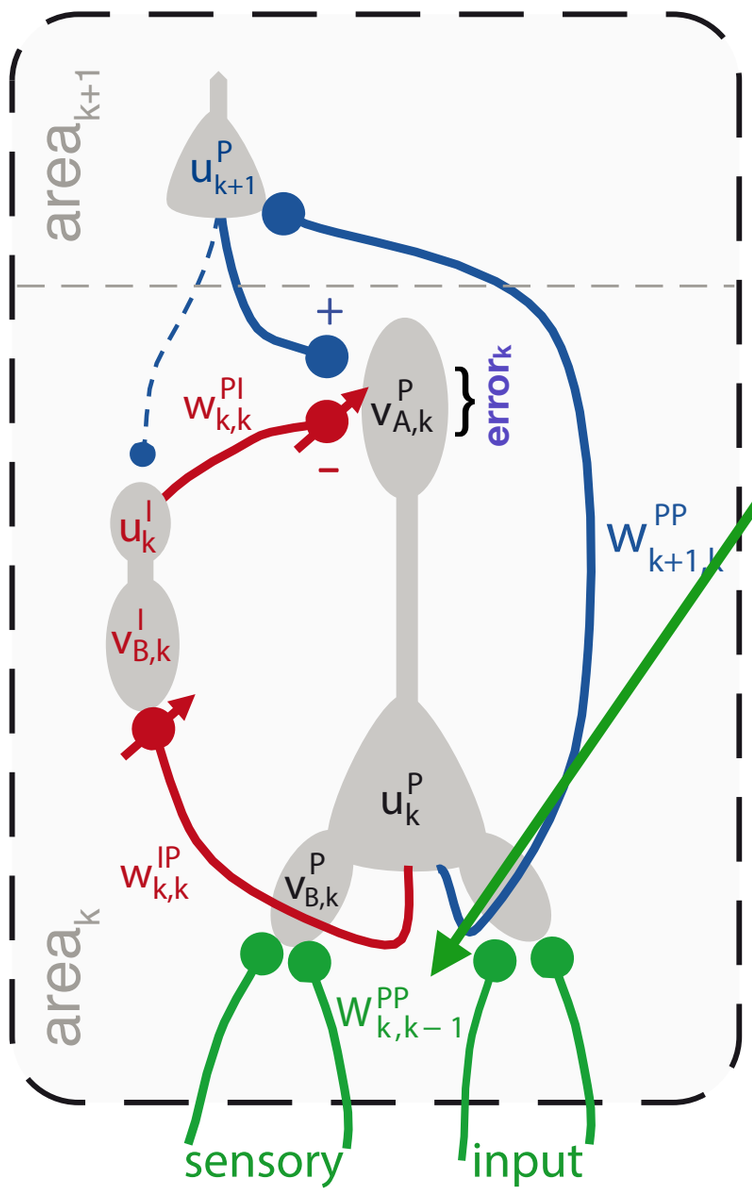

Learning rule for bottom-up synapses:

$$\frac{d}{dt} w^{PP}_{k,k-1} \sim \eta (u^P_k - v^P_{B,k}) r^P_{k-1}$$

Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits

## local error-correcting learning rules




Learning rule for bottom-up synapses:

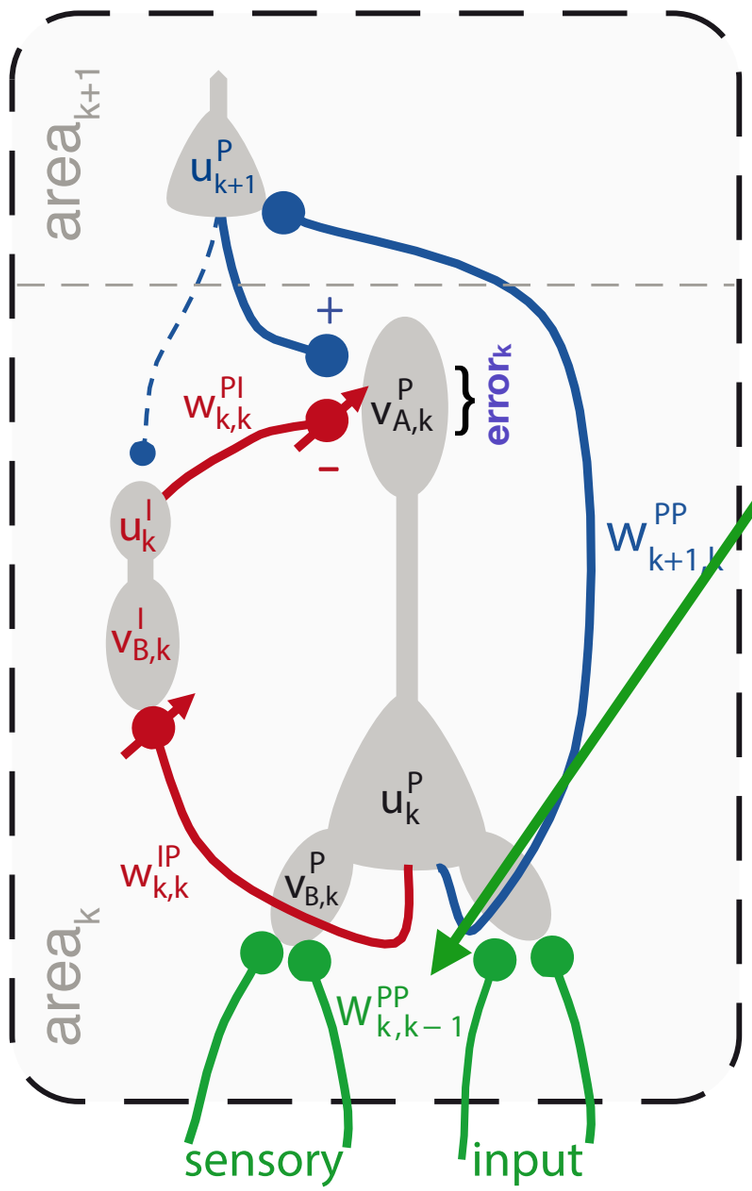
$$\frac{d}{dt} w^{PP}_{k,k-1} \sim \eta (u^P_k - v^P_{B,k}) r^P_{k-1}$$

$$u^P_k \sim \underbrace{(u^P_{k+1} - u^I_k)}_{v^P_{A,k} \sim \text{error}} + v^P_{B,k})$$

Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits

## local error-correcting learning rules




Learning rule for bottom-up synapses:

$$\frac{d}{dt} w^{PP}_{k,k-1} \sim \eta (u^P_k - v^P_{B,k}) r^P_{k-1}$$

$$u^P_k \sim \underbrace{(u^P_{k+1} - u^I_k)}_{v^P_{A,k} \sim \text{error}} + v^P_{B,k}$$

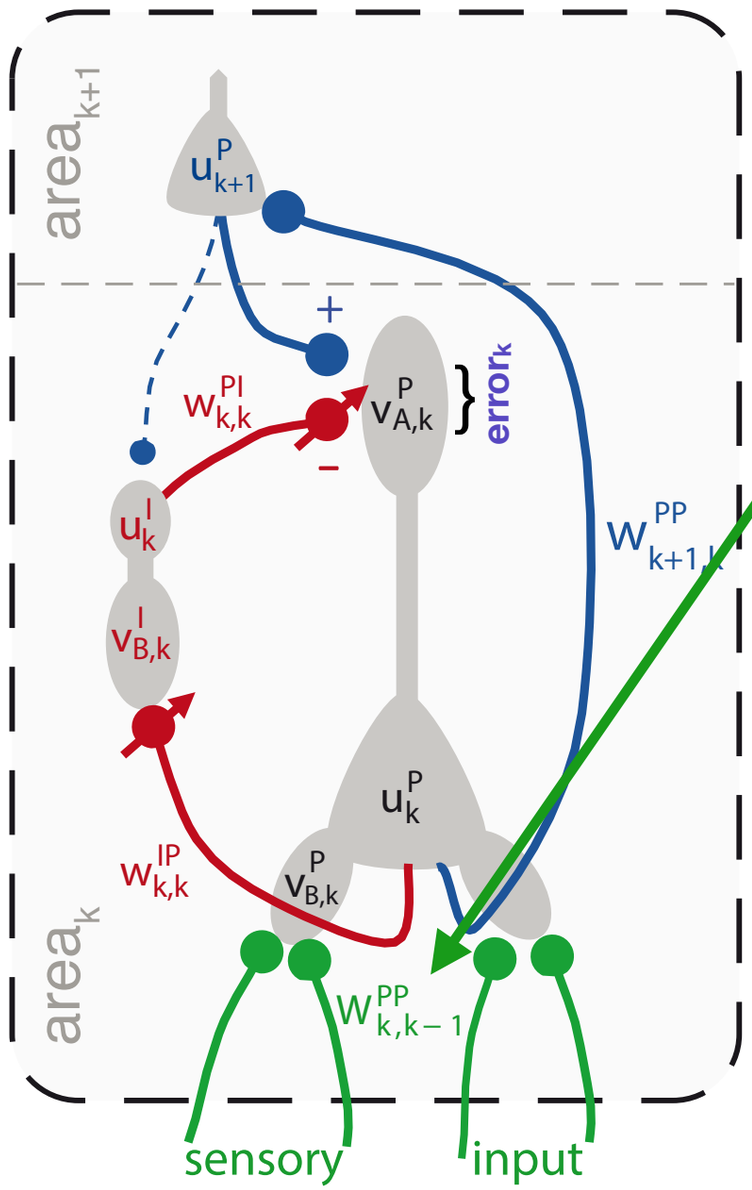
$$\frac{d}{dt} w^{PP}_{k,k-1} \sim \eta \underbrace{(u^P_{k+1} - u^I_k)}_{v^P_{A,k} \sim \text{error}} r^P_{k-1}$$

**Note**: To a first approximation the model dynamics and learning rules are equal to backprop (see details in paper).

Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits

## local error-correcting learning rules




Learning rule for bottom-up synapses:

$$\frac{d}{dt} w^{PP}_{k,k-1} \sim \eta \underbrace{(u^P_{k+1} - u^I_k)}_{v^P_{A,k} \sim \text{error}_k} r^P_{k-1}$$

Note: similar to **backprop learning rule**:

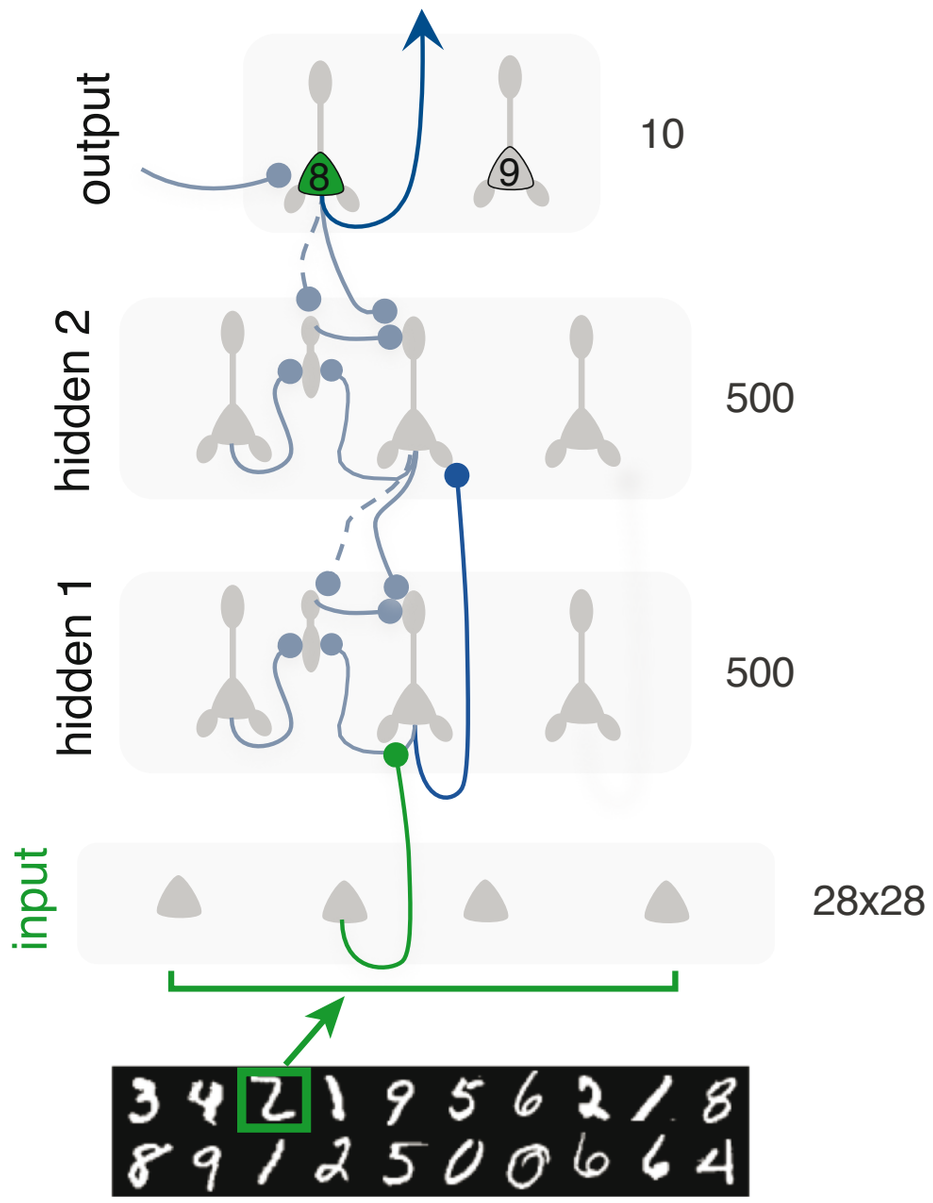
$$\Delta w^{PP} \sim \eta \, \text{error}_k \, r^P_{k-1}$$

$$r^P_{k-1} \sim u^P_{k,k-1}$$

$\text{error}_k$ : Error (or gradient) for area *k* backpropagated through all the areas above it

To a first approximation the model dynamics and learning rules give backprop. see details in paper:
Sacramento et al. 2018 (NeurIPS)

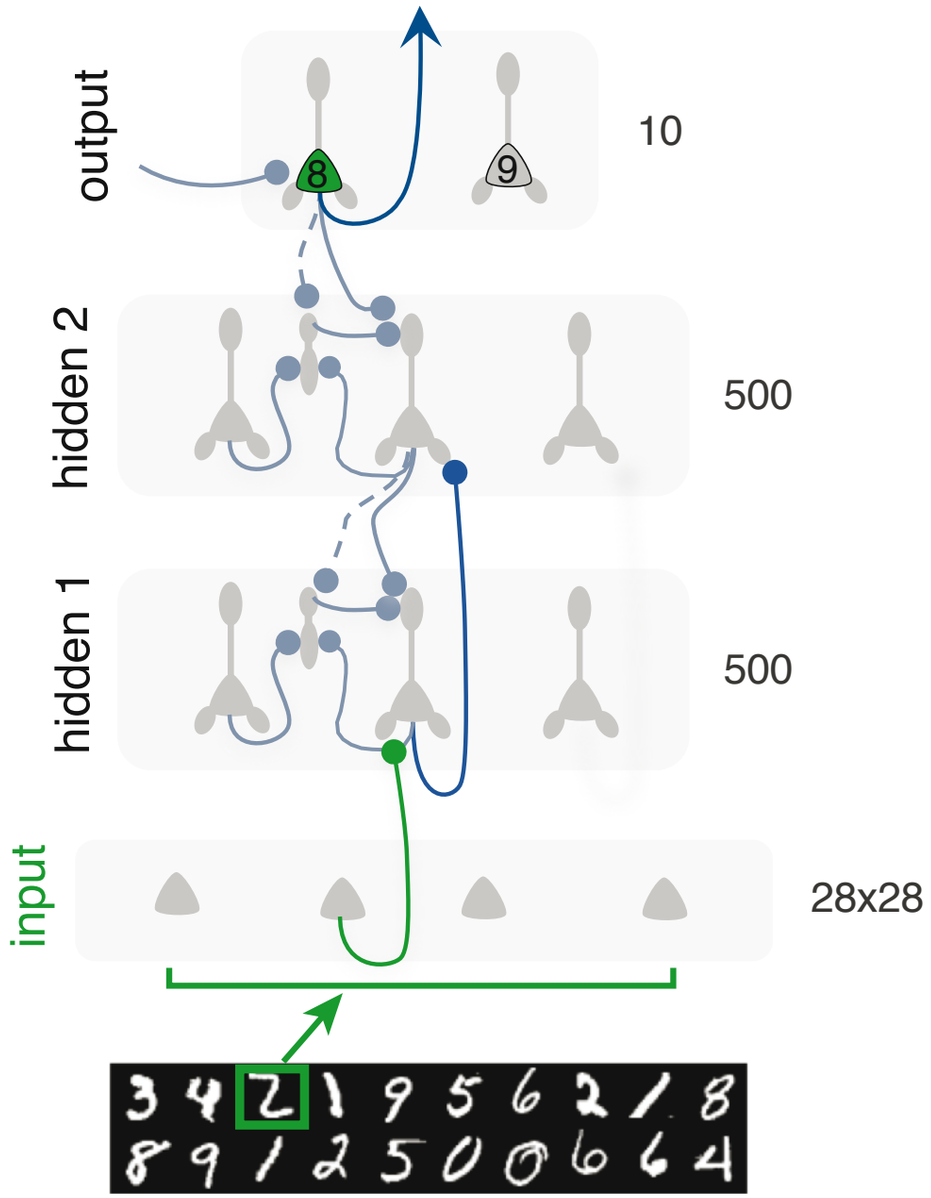
# Dendritic error microcircuits learn to classify hand-written digits



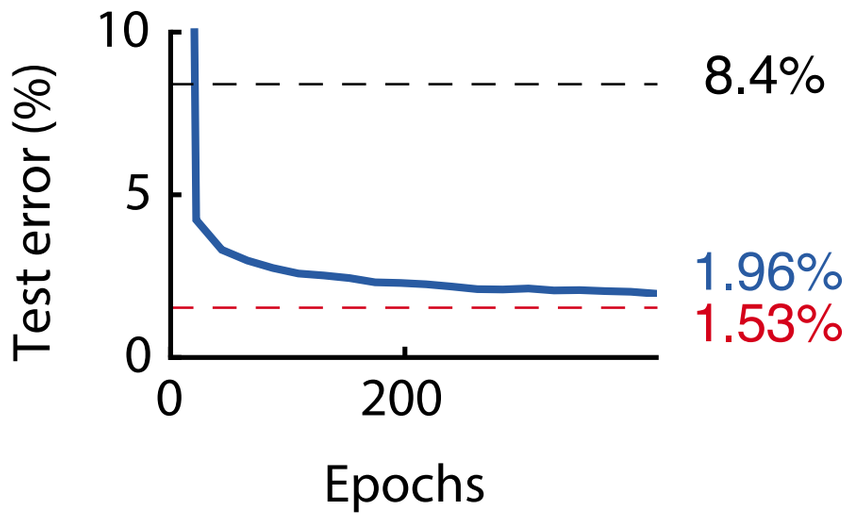

MNIST handwritten digit images

Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits learn to classify hand-written digits

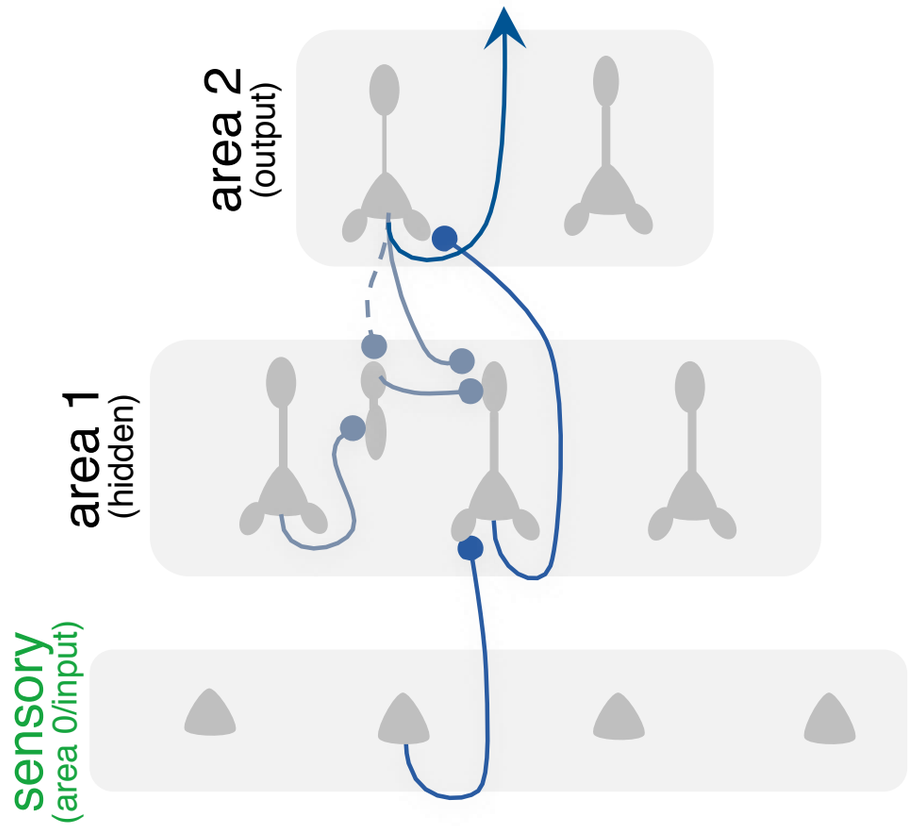



MNIST handwritten digit images




Sacramento et al. 2018 (NeurIPS)

# Dendritic error microcircuits approximate backprop




Sacramento et al. 2018 (NeurIPS)

**By being closer to biology neural networks ‘solve’ three key problems:**

3. **Two phase learning**: No need for two separate phases
4. **Separate error network**: Neuron encodes both activity and errors
5. **Non-local learning rules**: Learning rules are local

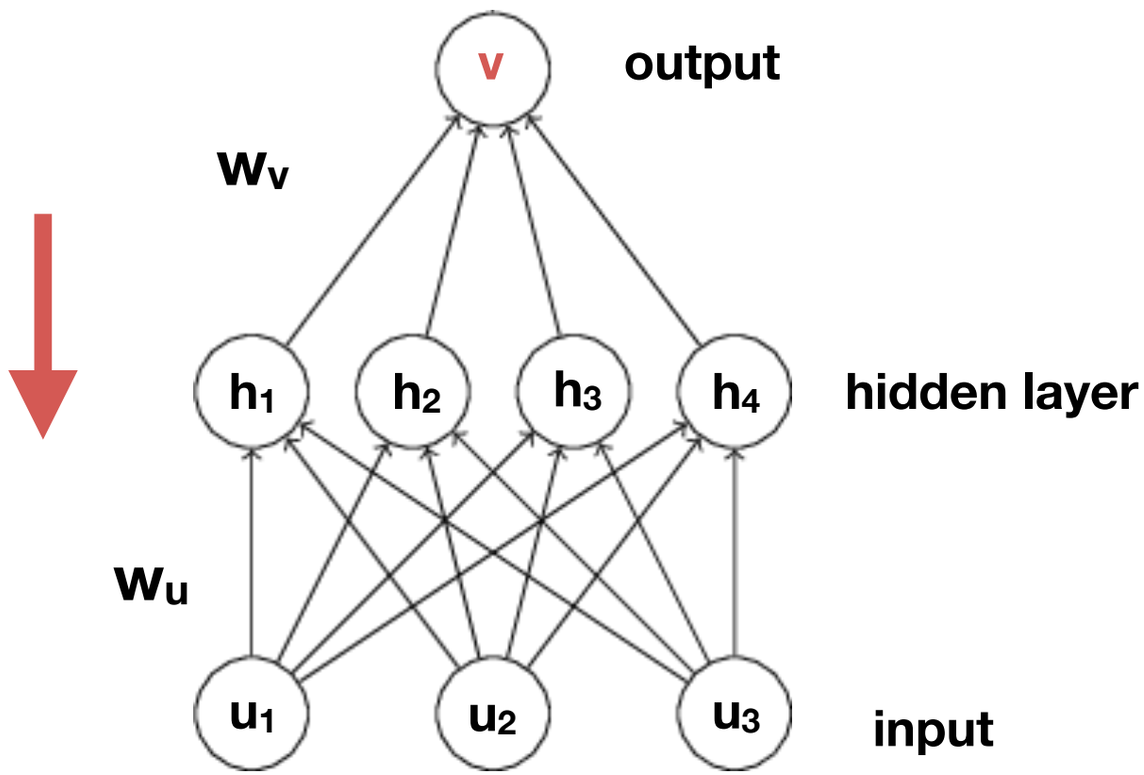
see also Guerguiev et al. eLife 2017

# The backpropagation algorithm
## and why it is at odds with biology
### 6. Target

Backprop: Supervised learning needs a specific target (teaching signal).

Biology: It is far from clear how such target signals could be generated.

$$\textbf{error} = \frac{1}{2}(v - \text{target})^2$$



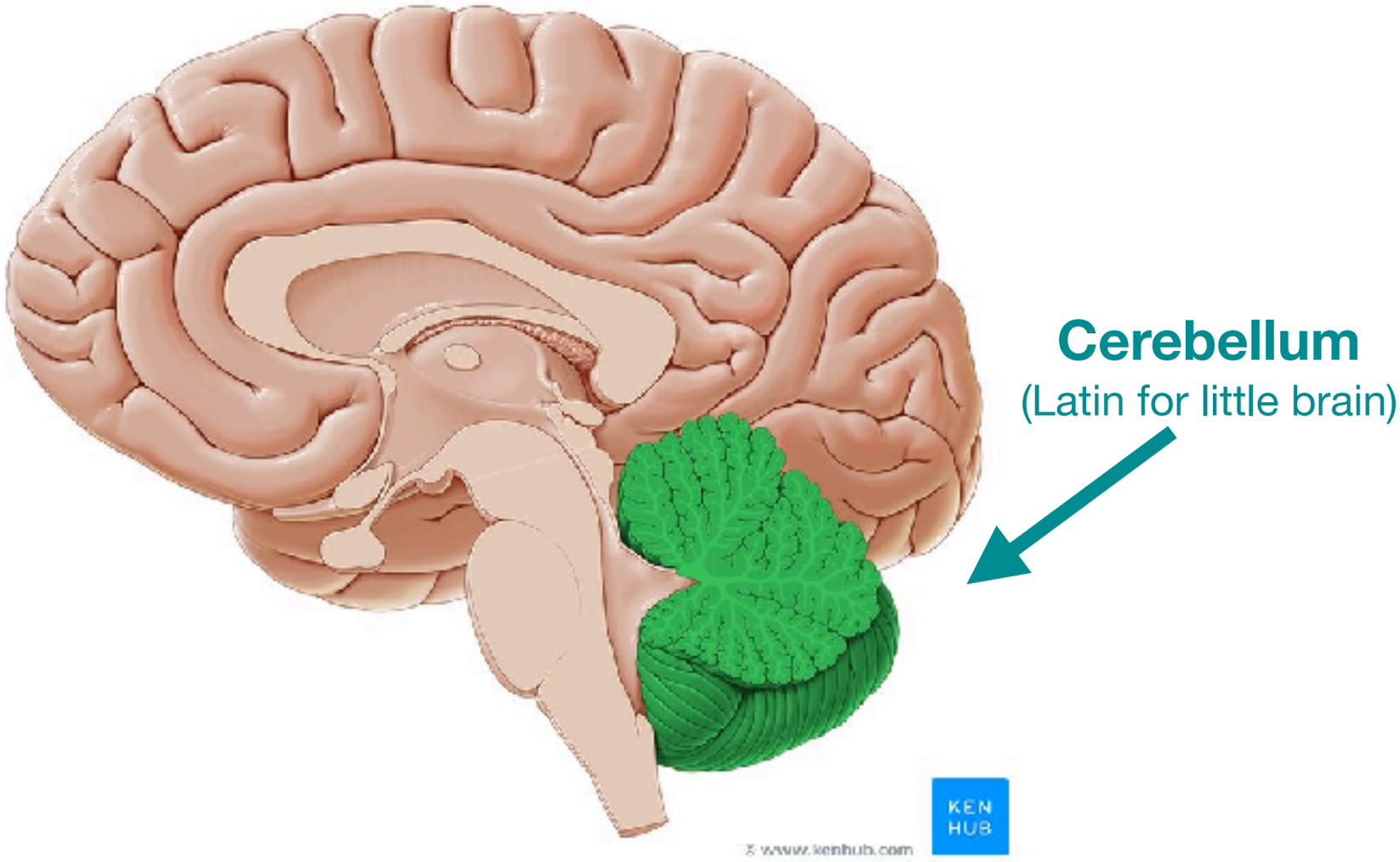
# The backpropagation algorithm
## and why it is at odds with biology
### 6. Target

**Solution 1**: Backprop also works in the context of unsupervised (e.g. autoencoders) and reinforcement learning (e.g. deep Q-learning). We are going to cover some of this in the coming lectures.

**Solution 2**: Specific brain areas (e.g. **cerebellum**) may be calculating teaching signals, which are then used for internal supervised learning at other brain areas (see end of this lecture).

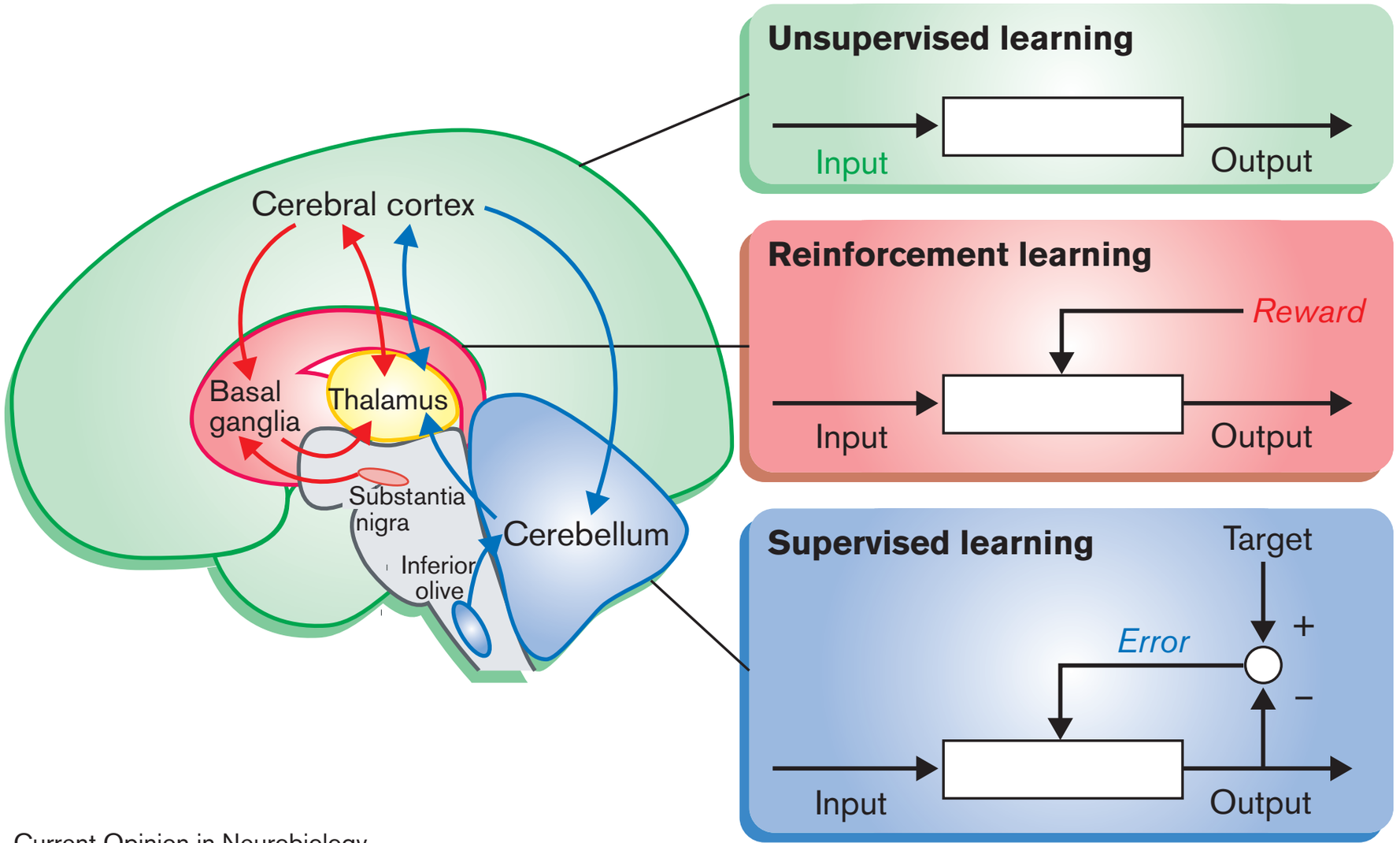
# Supervised learning in the Cerebellum

The **cerebellum** (small brain in Latin) has a stereotypical structure, and *contains more neurons than the rest of the brain*! It is classically involved in motor error correction, but growing evidence suggests to be also involved in regulating many other aspects of behaviour.



# Supervised learning in the Cerebellum

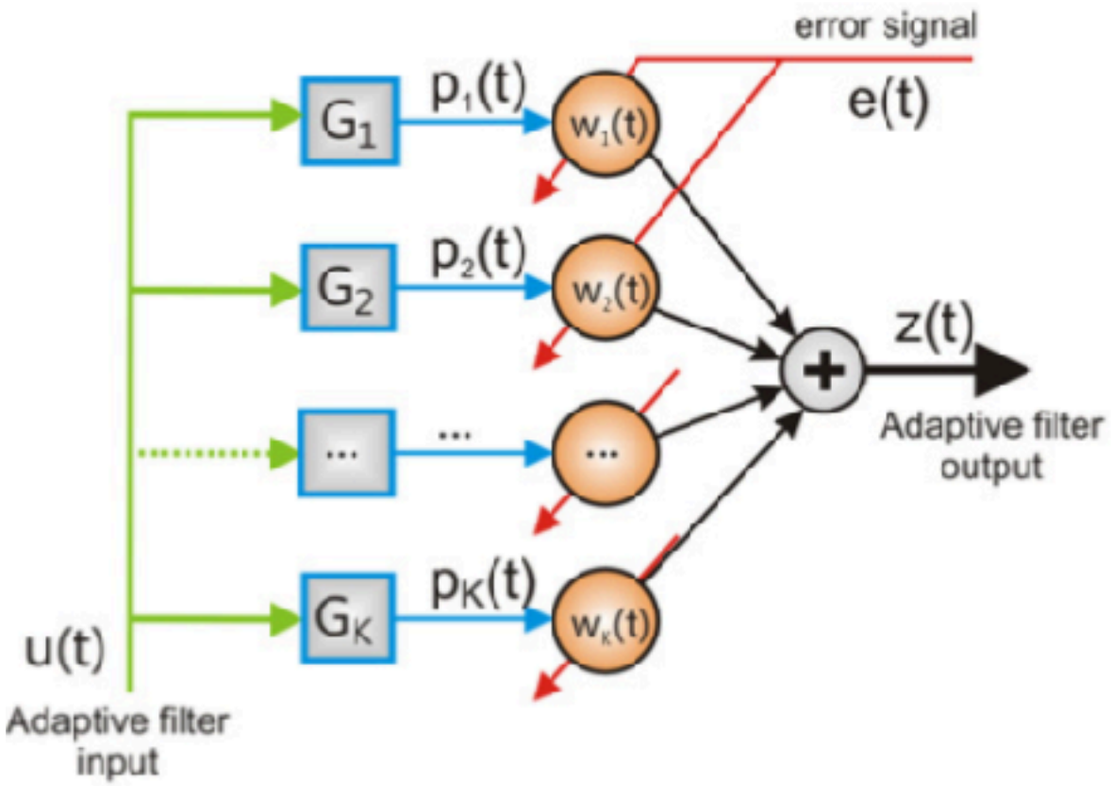
The **cerebellum** itself seems to use a form of supervised learning, but may also provide teaching signals to the cerebral cortex.




Current Opinion in Neurobiology

Doya, Curr Opin Neurobiol (2000)

# Cerebellum as a adaptive filter/controller



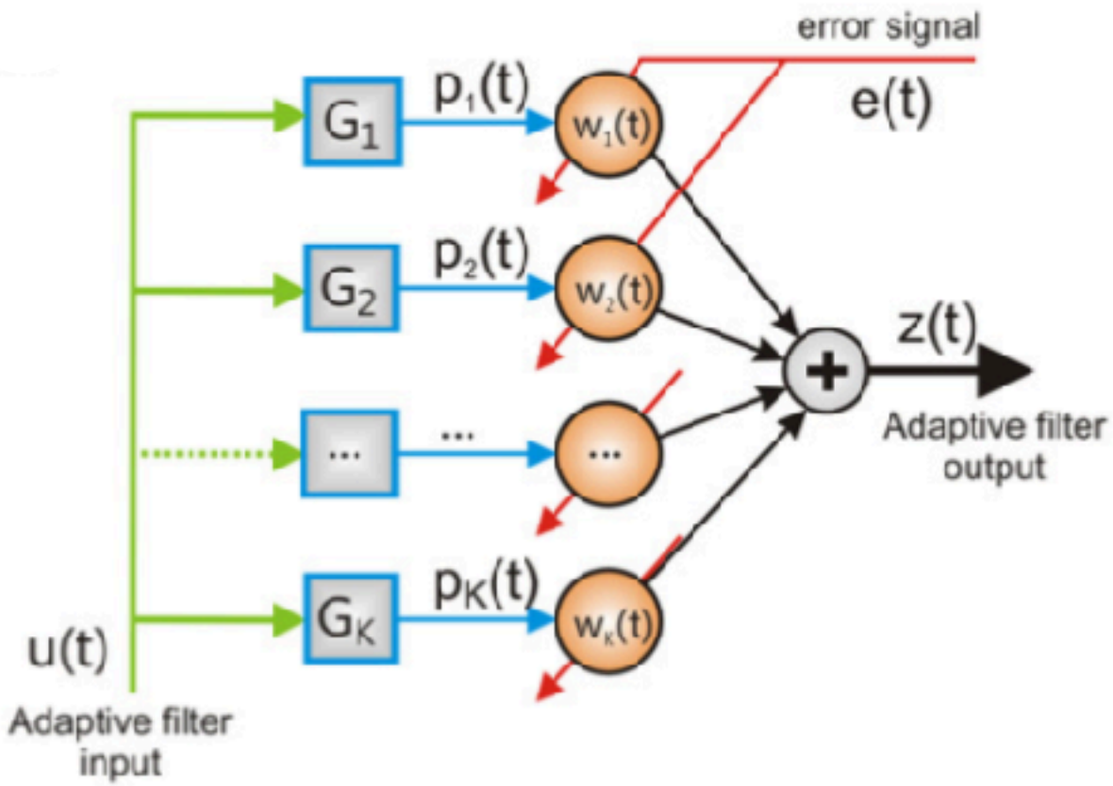

**Goal**: Online control (via z(t)) of movements to achieve a desired outcome (learning occurs via e(t))?

Porrill et al. Neural Networks (2013)

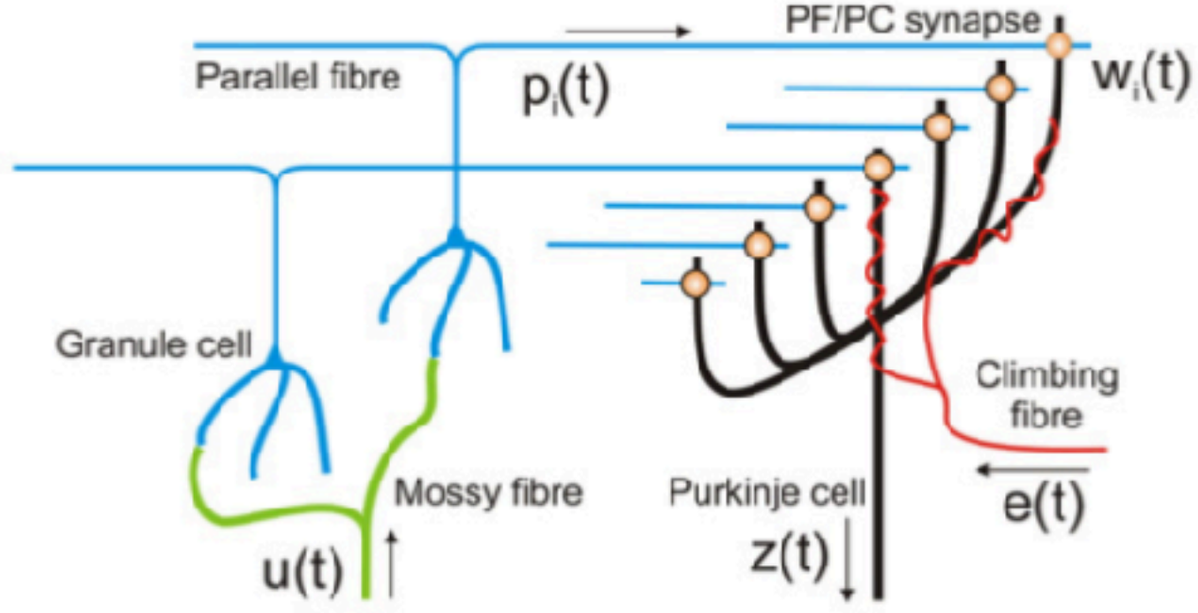
Raymond and Medina, Annual Rev Neurosci (2018)

# Cerebellum as a adaptive filter

**Controller/filter model**




**Neurobiology of the cerebellum**




Sensory input/ motor copy

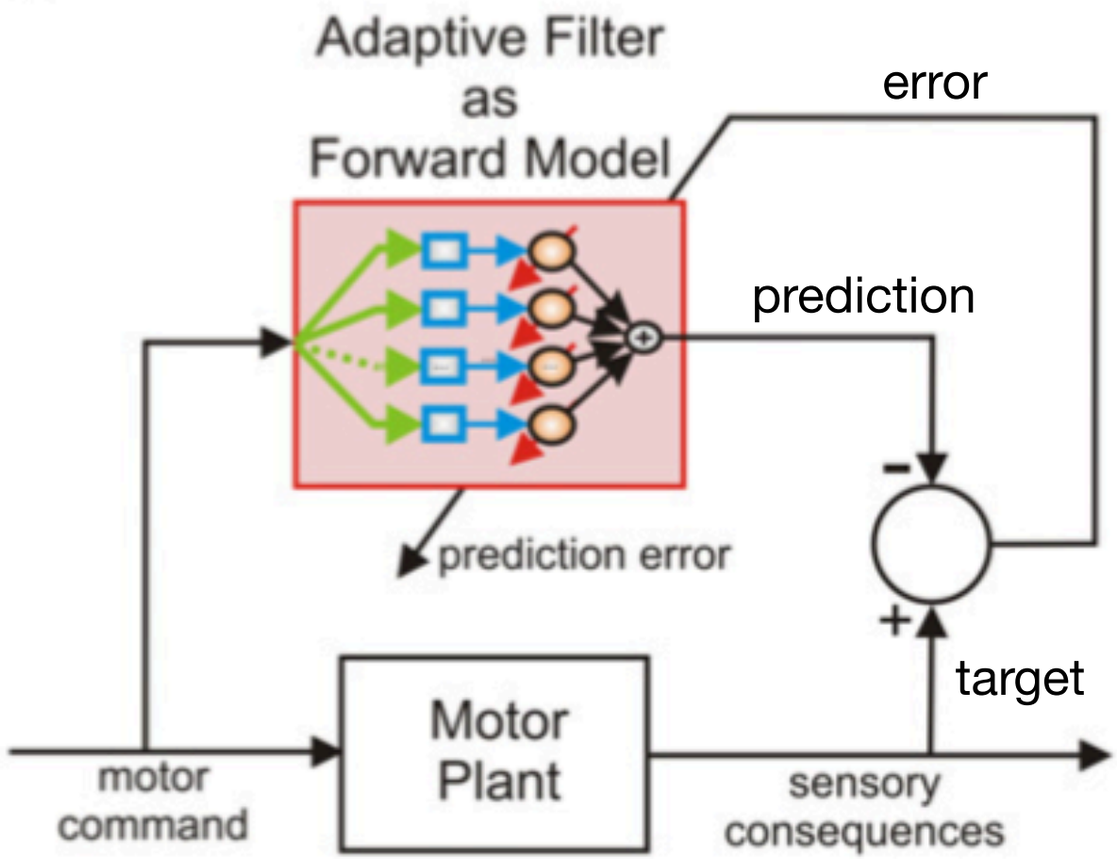
Prediction

Porrill et al. Neural Networks (2013)

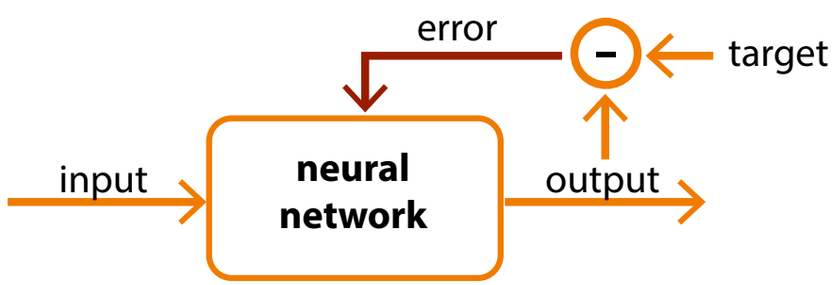
Raymond and Medina, Annual Rev Neurosci (2018)

# Forward model of the cerebellum

**Forward model:**
Given a motor command predict its outcome in the environment (using sensory info).




Note that this is similar to:

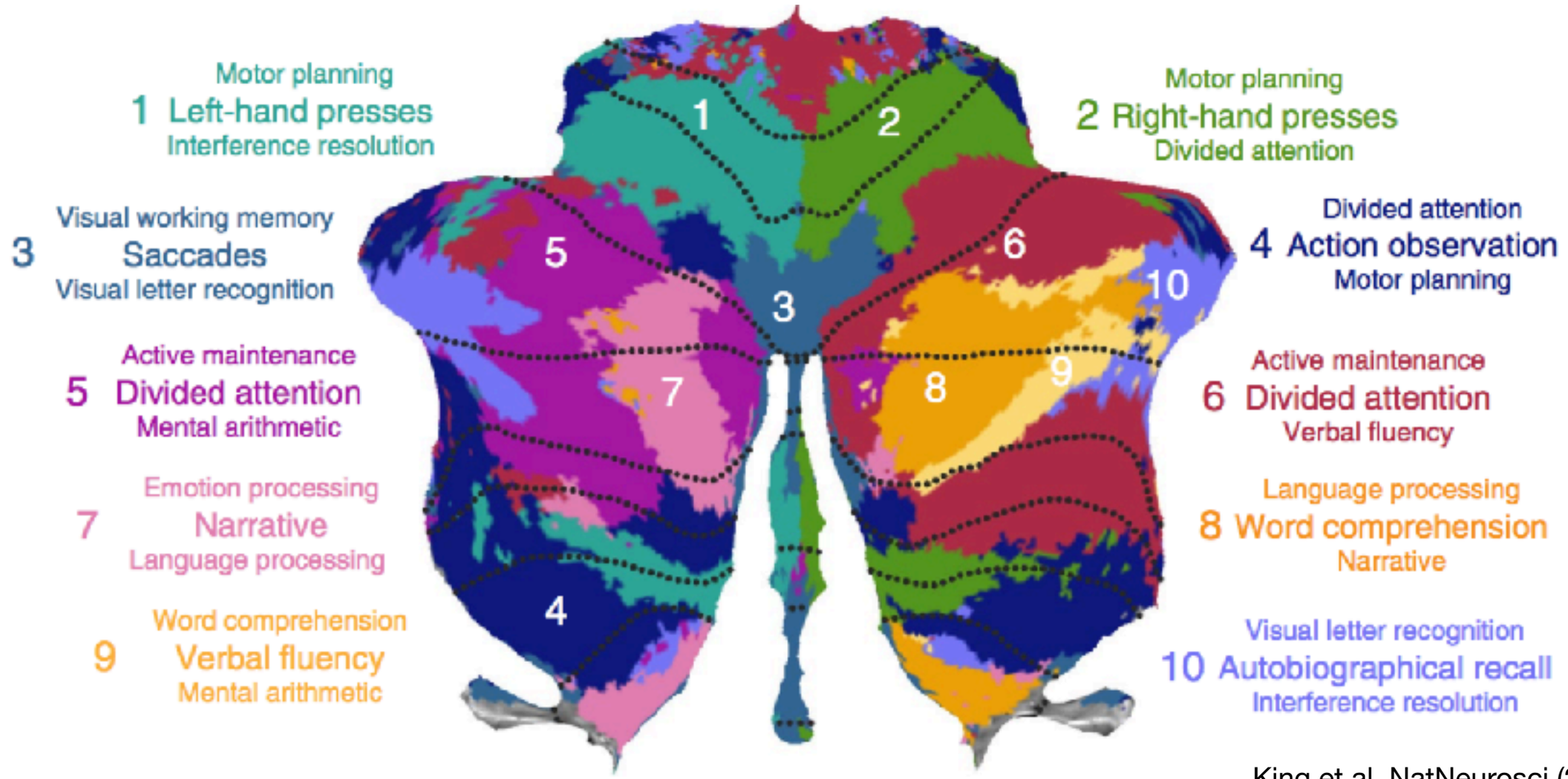



Porrill et al. Neural Networks (2013)

Raymond and Medina, Annual Rev Neurosci (2018)

# The cognitive cerebellum

A new emerging field showing that the cerebellum is involved in much more than simple motor control. 2D cognitive map of the cerebellum showing activations to many different tasks:




King et al. NatNeurosci (2019)

# Summary

1. **Deep learning successes relies on the backprop algorithm**

2. **Backprop is slowly becoming a possible model for learning in the brain, but several issues remain unsolved**

4. **Dendritic microcircuits provide a powerful model of learning in the brain that approximates backprop**

6. **Cerebellum seems to use supervised learning**

**Exercise:** Practice writing down the gradient wrt a given weight in a simple neural network!

# References

**Text books:**

Parallel Distributed Processing, Rumelhart et al. 1986 (classical book on neural networks)

**Relevant papers:**

- Roelfsema and Holtmaat, Nature Neuroscience Rev (2018) (recent review on the credit assignment problem in the brain)
- Doya, Curr Opin Neurobiol (2000) (review on how different brain areas learn)
- Richards and Lillicrap, Current Opinion in Neurobiology (2019) (recent review on dendritic credit assignment)
- Sacramento et al., NeurIPS (2018) (shows how dendritic microcircuits may approximate backprop)

# Upcoming lectures

- L1[4]: Neural circuits and learning: introduction
- L2[4]: Supervised learning & backprop
- **L3[5]: Visual system: deep learning?**
- L4[5]: Reinforcement learning
- L5[6]: Unsupervised learning
- L6[6]: Temporal processing
- L7[7]: Recurrent neural networks

# Lab 1

On Github IPB page:
comsm0075.github.io/2022-23.html

- Implement and study the behaviour and biological plausibility of supervised learning/backprop.

- Teaching assistants: Joe Pemberton
  Dabal Pedamonti