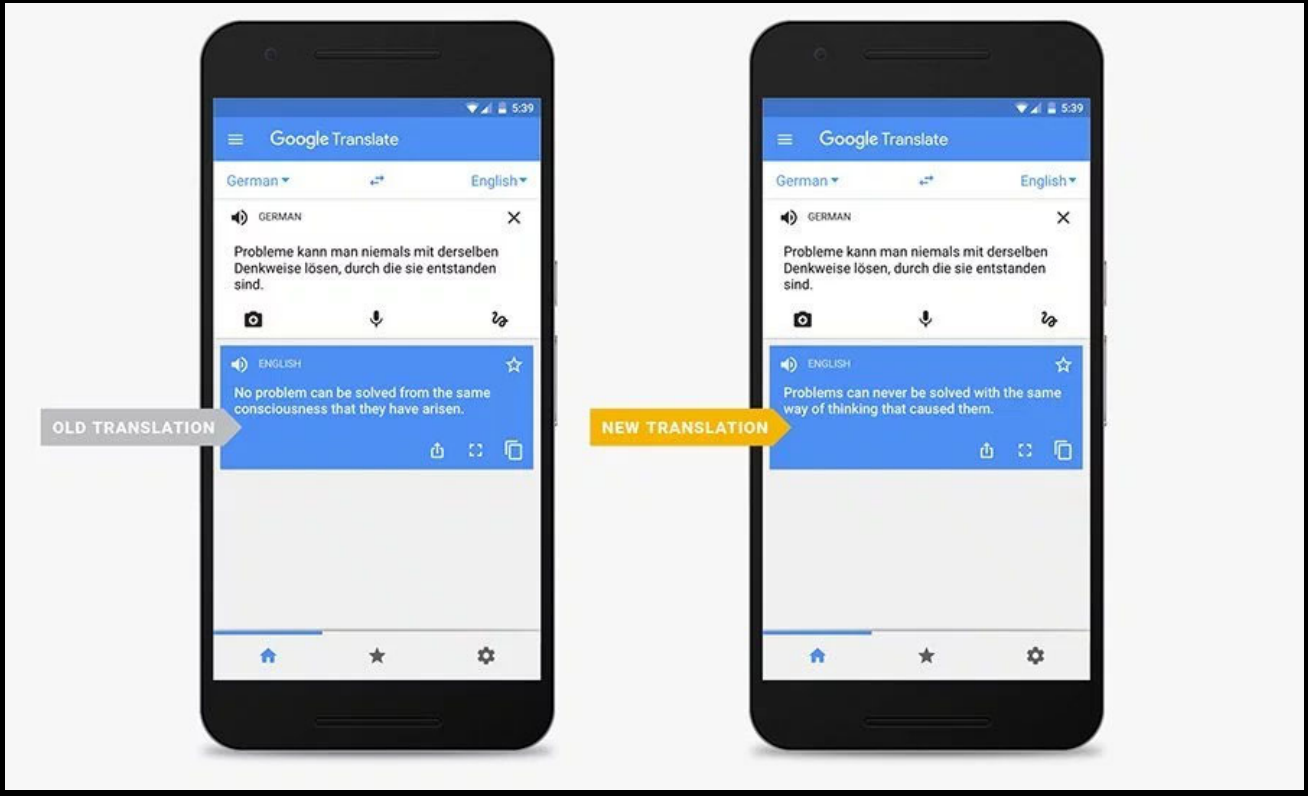
Deep Learning
Recurrent neural nets

COGS-Q355

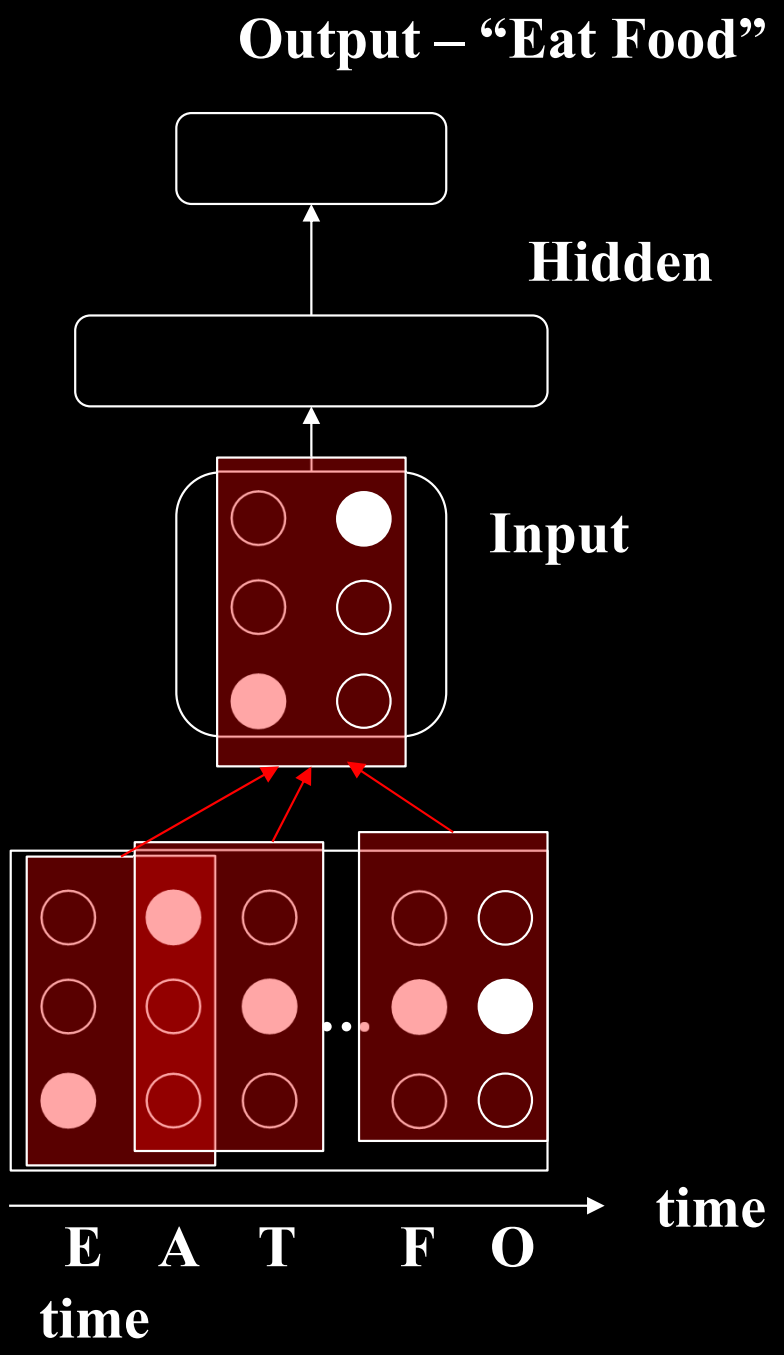
What about temporal patterns?

- Examples: Speech to text, Google translate – now with neural nets
- How to represent temporal patterns?



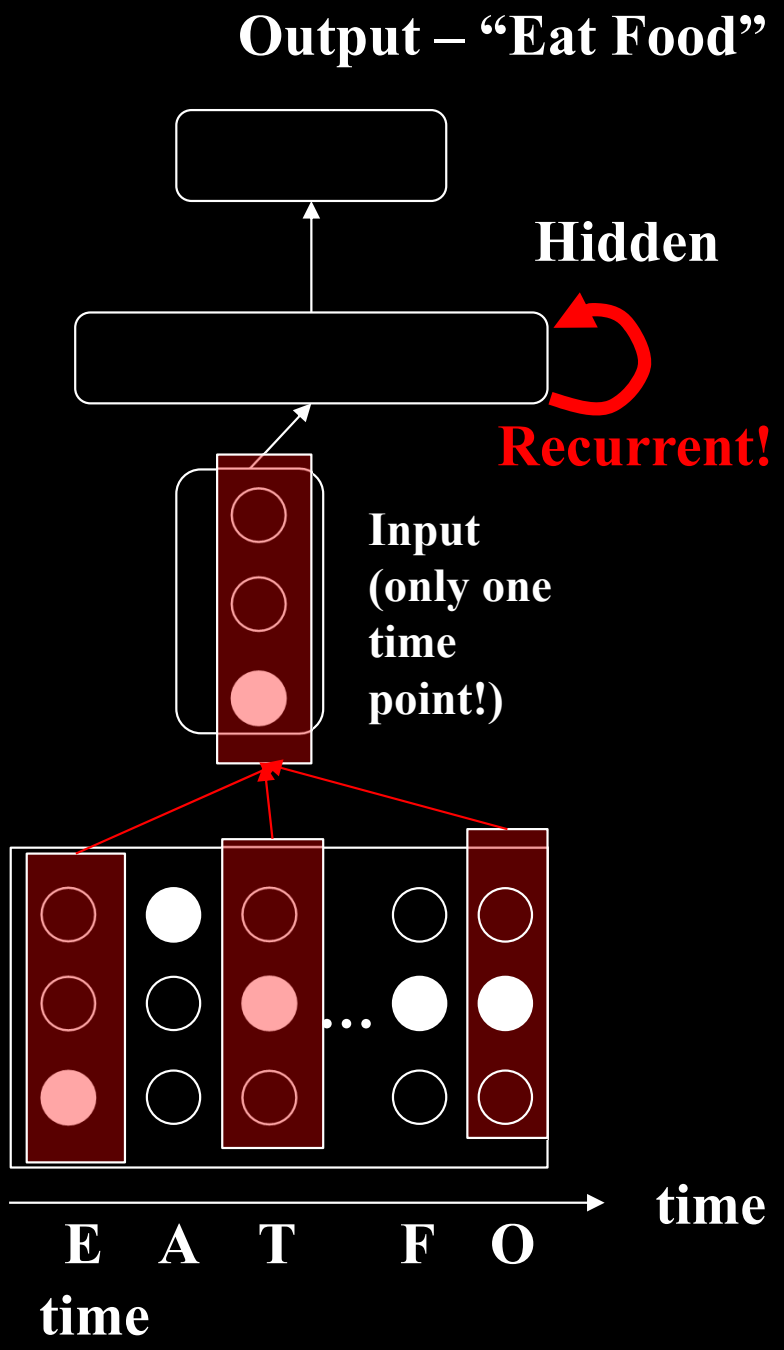
What about temporal patterns?

- One possibility: time delay neural networks (TDNN)
- Take a sample of input patterns over a window of time
- Shift the window later in time, repeat



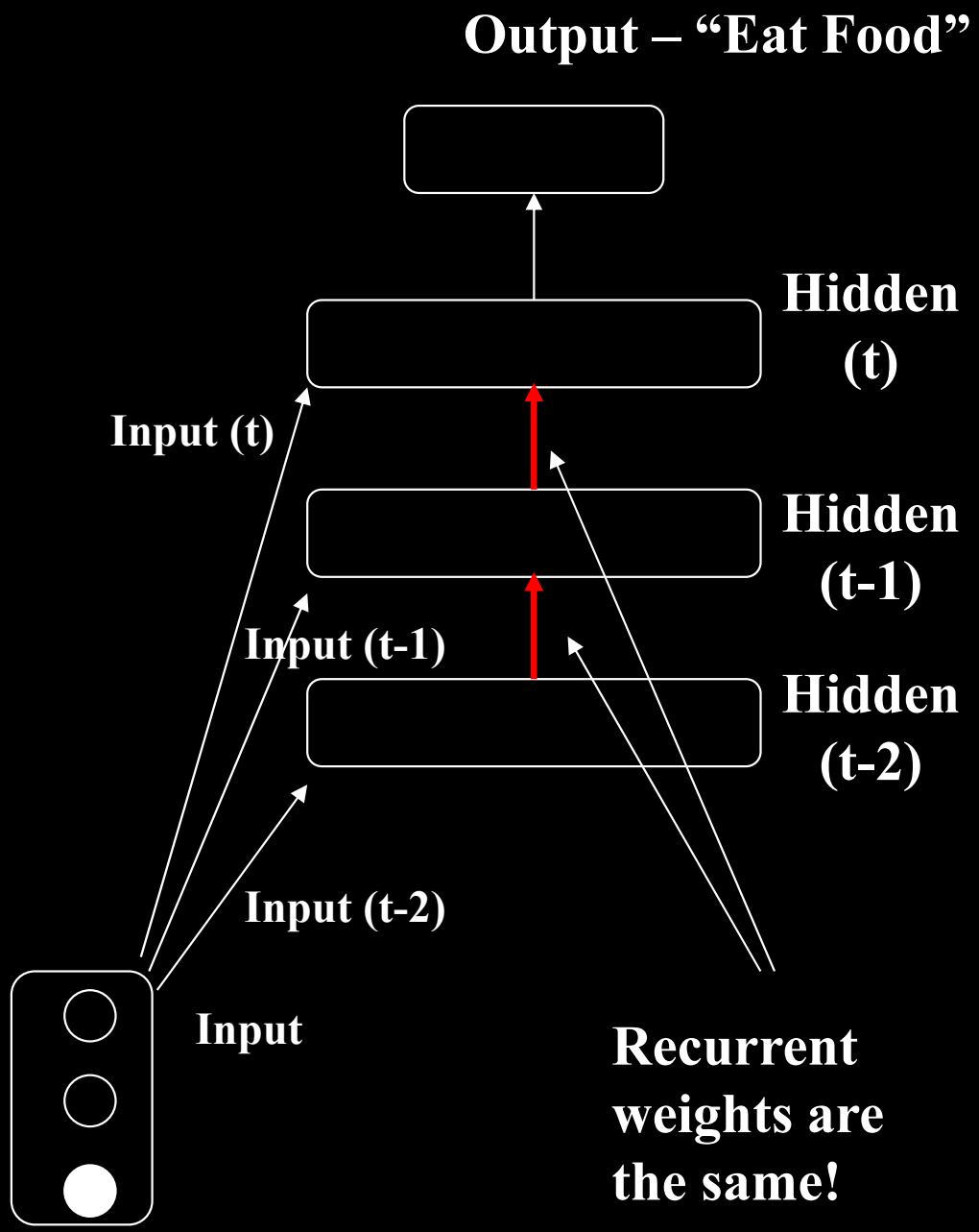
Recurrent Neural Nets

- Another possibility: "recurrent" network, in which the output of the hidden layer is fed back to the hidden layer
- Question: But how do we propagate the error??
- Answer: Remember the hidden layer activations at previous time steps, and unroll it in time!

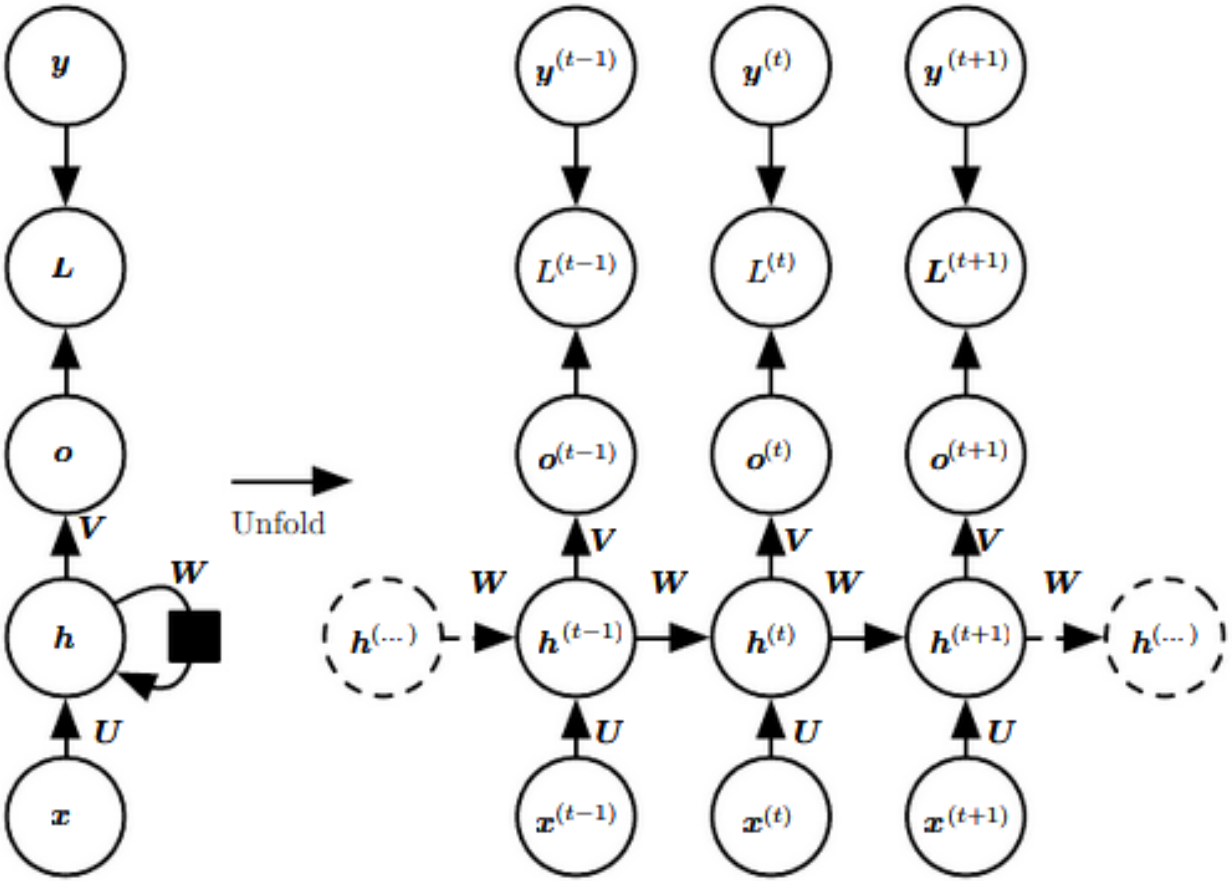


Recurrent Neural Nets – "Unrolled"

- E.g. backpropagation through time (BPTT) for 3 time steps
- Note the recurrent weights in red.
- For training, have to specify how many "unrolls" of the recurrent weight, i.e. how many steps backward in time



Recurrent Neural Nets



Note: vanishing gradient problem can be an issue. Derivative of tanh has max of 1, but deep RNNs can still have gradient vanish

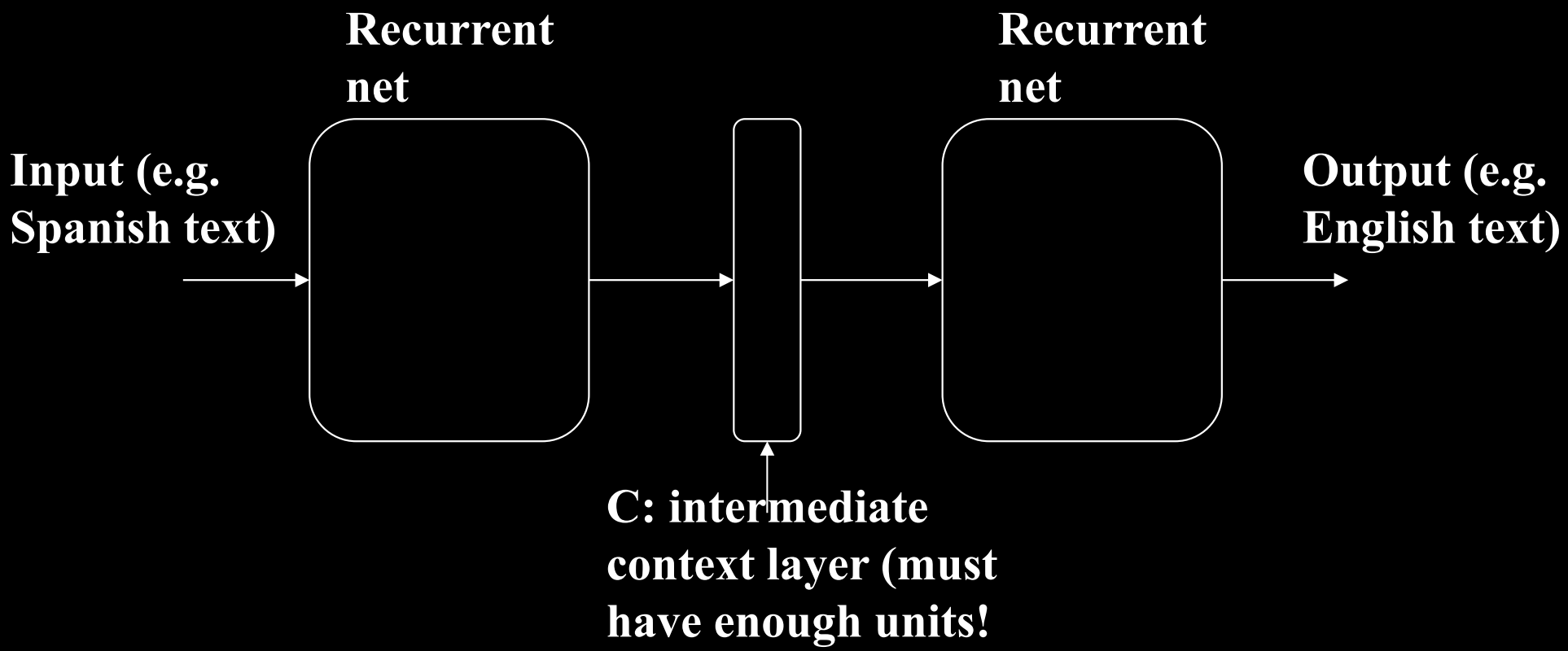
Forward (t to t-tau)

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$

W=recurrent wts
h(t-1) = previous hidden layer activity
X(t)=input
U=input weights
b=bias term
O=raw output layer
Y=softmax of output layer (probability estimate)
L=loss function

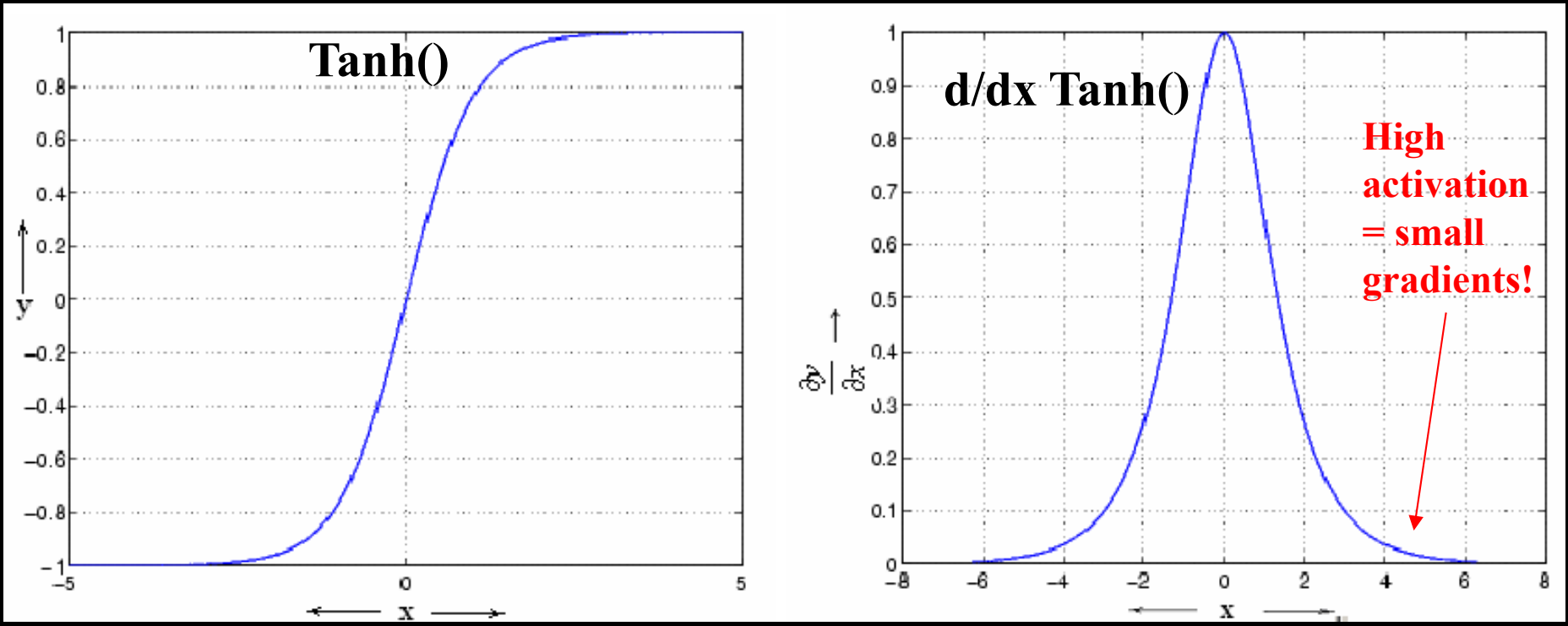
Recurrent Neural Nets

- Application: Encoder/Decoder sequence (speech to text, text to speech, text translation, etc.)
- Generally have recurrent nets for both source and output sequences! (E.g. -- Often can't translate word for word because of differences in grammar)



Recurrent Neural Nets

- Issue: Suppose a particular input signal should influence the output 100 time steps later, or 1000 steps later. How do we learn dependencies on something far back in time?
- Problem: The gradients tend to vanish through time. If the recurrent weights are too large, and the activation too large to maintain activation across time, then the activation will saturate (tanh, sigmoid, with zero derivative) or blow up (ReLU).



Recurrent Neural Nets

- One possible solution: Reservoir computing (liquid state machines, echo state networks)
- Basic idea is to:
  - (1) make the hidden layer really huge, with a *lot* of units, and
  - (2) initialize all the recurrent weights to random values, and
  - (3) simply shut off learning in the recurrent weights
- Then an input comes in and bounces around the hidden layers for a long time, basically creating an expanded spatiotemporal basis dimension. The output is learned only by the delta rule on the hidden-to-output weights.
- Seems counterintuitive to shut off learning, but it works, at the expense of requiring a lot more hidden units

Aside: hardware

- New Nvidia machine learning hardware

JETSON NANO DEVELOPER KIT

$99 AI Computer

128 CUDA Cores | 4 Core CPU

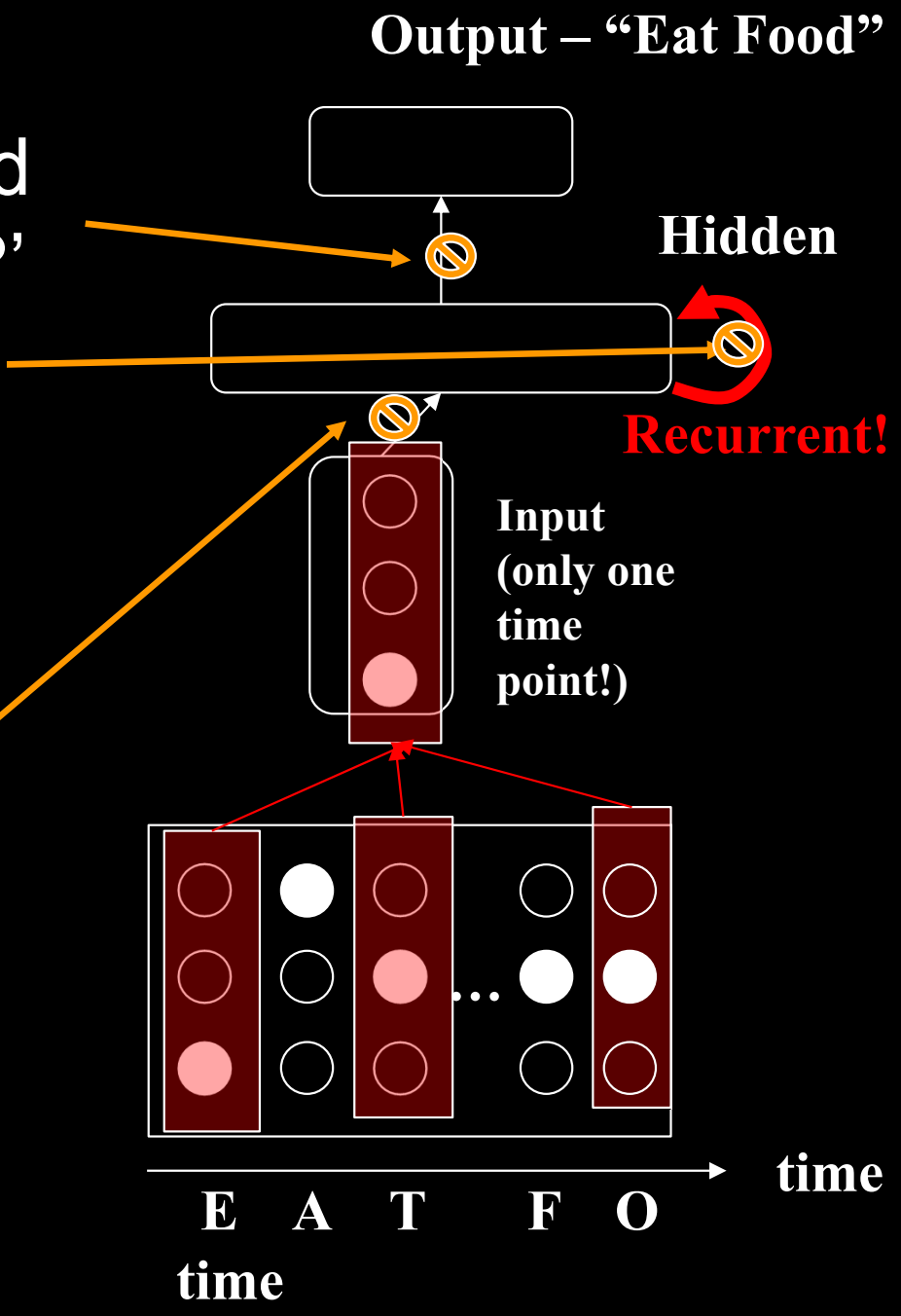
472 GFLOPs

5W | 10W

https://www.androidauthority.com/nvidia-jetson-nano-966609/

Recurrent Neural Nets - gating

- What if we had better control over what was gated OUT of the recurrent layer?'
- What if we could control whether the hidden layer activity decayed away or not?
- What if we could control what was gated IN to the recurrent layer?



Recurrent Neural Nets

- Another idea: What if we could train a hidden unit to **maintain** its activity over a long period of time?
- Add 3 parts to the RNN, to turn it into "Long Short Term Memory" (LSTM):
  - 1) An "Input" gate, to control what inputs are allowed to activate a hidden unit, and in what context
  - 2) A "forgetting" gate, to control whether a hidden unit activity will be maintained, perhaps indefinitely, or whether it will decay away
  - 3) An "output" gate, to control when a hidden unit is allowed to influence the output

Hochreiter & Schmidhuber, 1997

LSTM

- Each gate is controlled by a set of weights and input from the input layer and hidden layer, and the gate weights are adjusted with error backprop

1) An "Input" gate, $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
2) A "forgetting" gate, $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
3) An "output" gate, $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$

Note similarities of gating signals!

All gate signals generally vary between 0 and 1

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- The hidden unit (h) has a corresponding state (C), which can carry over short term memories
- To update C, first compute a new candidate C value ($\tilde{C}$) from the input and hidden layers:

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

- Finally, update C:
  - $C_t = f_t C_{t-1} + i_t \tilde{C}_t$

Carry over old Or forget

Add in new input to hidden layer

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- To update h, use the output gate to determine how much of C to pass into h at the next time step

$$h_t = o_t * \tanh(C_t)$$

- h is the network output

- What about learning? Basically it's the same as with RNNs: apply the chain rule to compute dE/dW, then subtract a fraction of that off of the weights. All gating weights can be trained this way.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

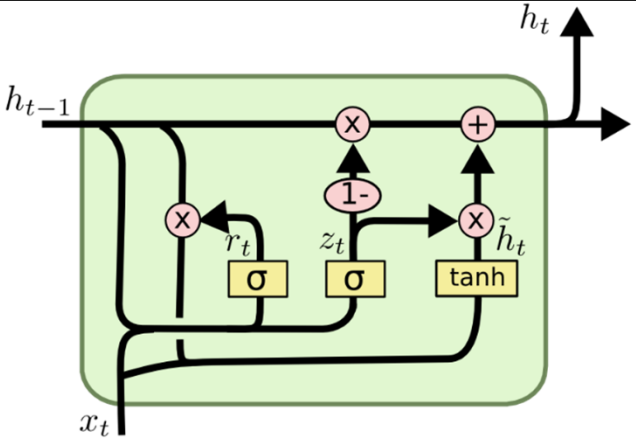
LSTM

- Some variations on LSTM:
- 1) Peepholes – the gates get to look at the states, not just the hidden layers:
  - $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ →
  - $i_t = \sigma(W_i[C_{t-1}, h_{t-1}, x_t] + b_i)$
- 2) Coupled Input and Forgetting (i.e. no separate input gate) –
  - $C_t = f_t C_{t-1} + i_t \widetilde{C}_t$ → $C_t = f_t C_{t-1} + (1 - f_t)\widetilde{C}_t$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- Some variations on LSTM, continued:
- 3) Gated Recurrent Units (GRU) – simpler
  - the input and hidden gates are combined
  - The cell (C) and hidden (h) are combined



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$ **Forgetting gate**

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$ **Input gate**

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- Other variations…

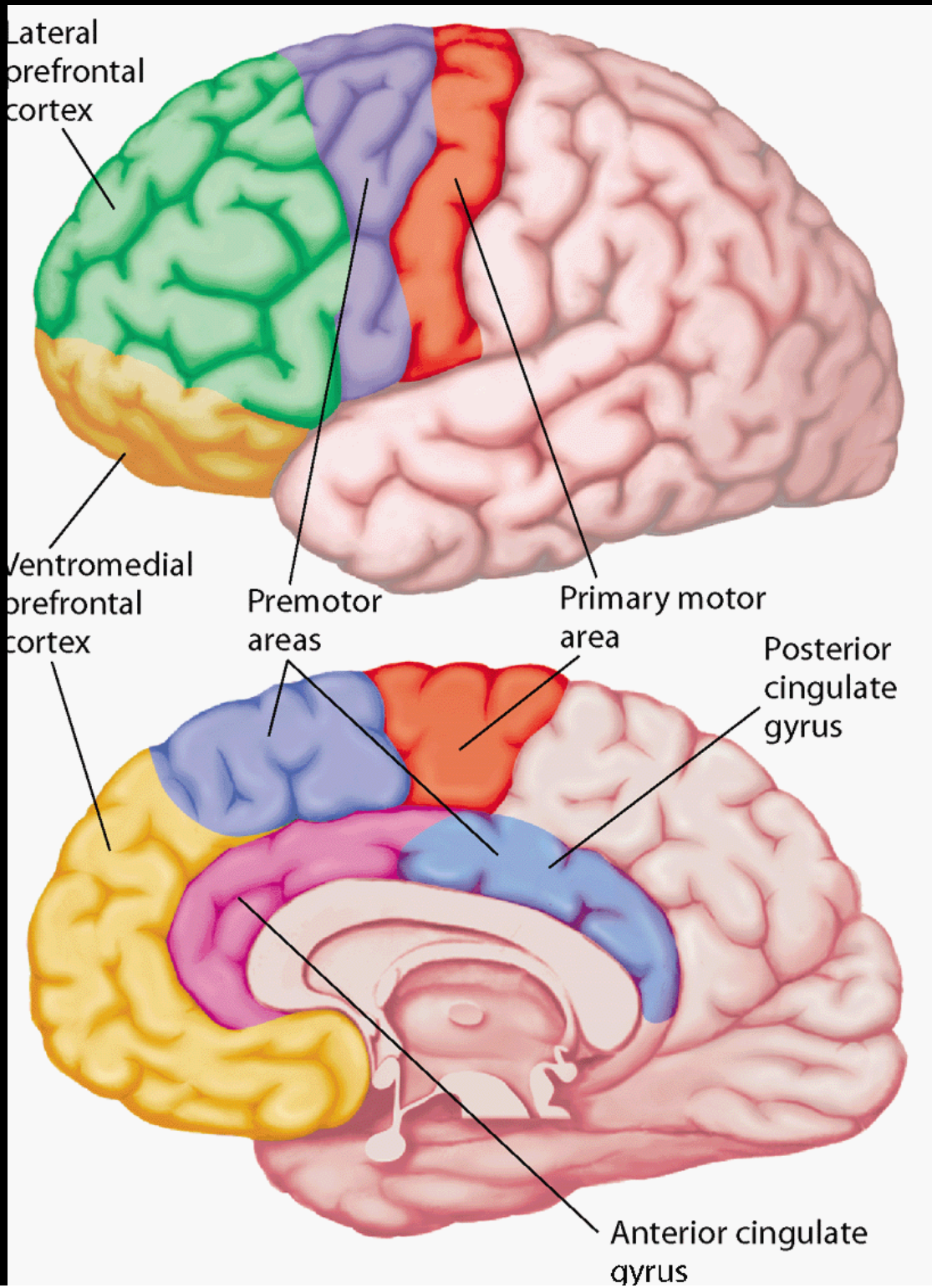
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Nets – Biology of working memory

- Animal working memory?
- How does the brain do working memory?

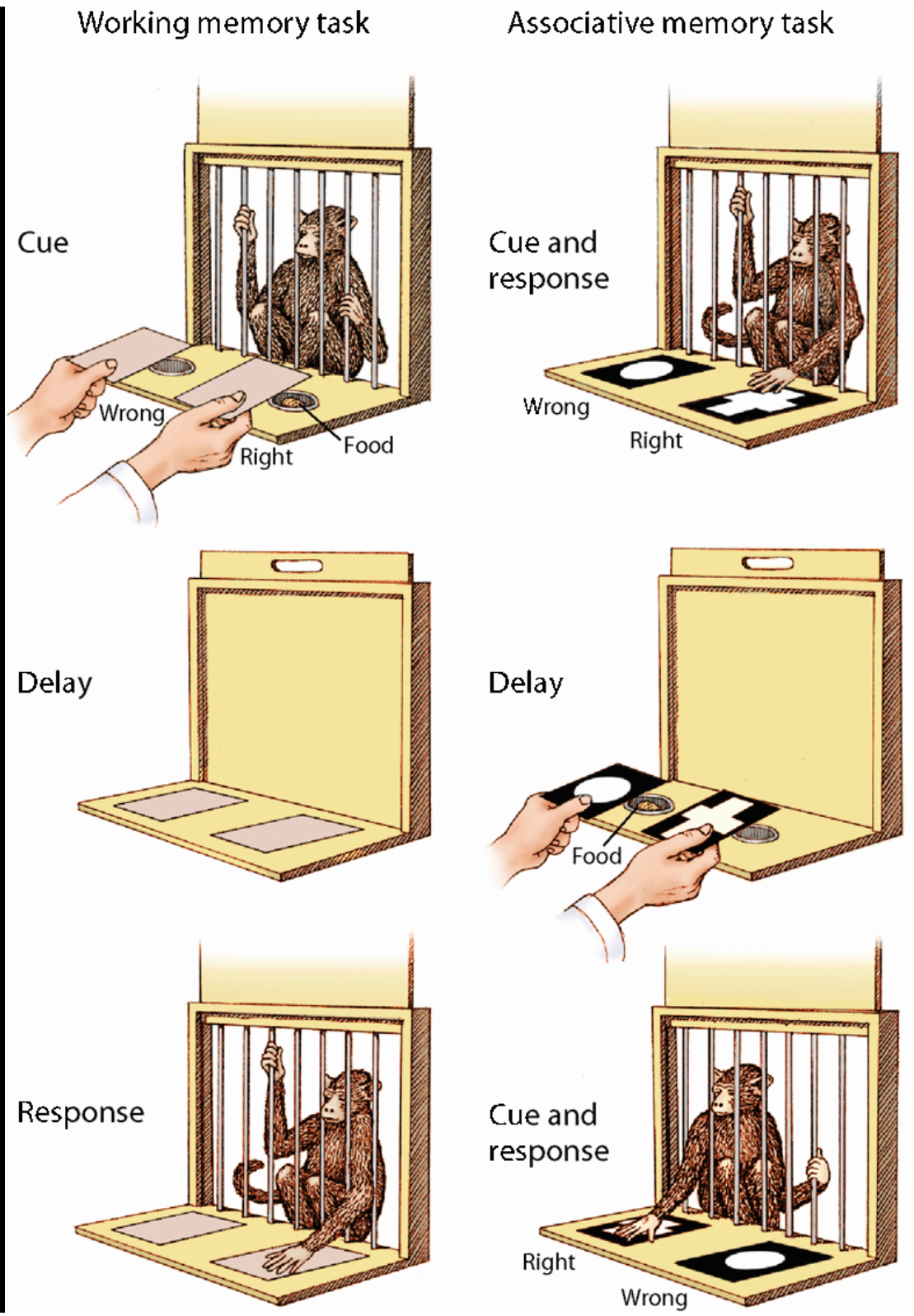
Working memory

- The frontal lobes, executive function
- The prefrontal cortex
- Lateral prefrontal cortex – working memory
- Sensory working memory – posterior cortical regions



Delayed response task

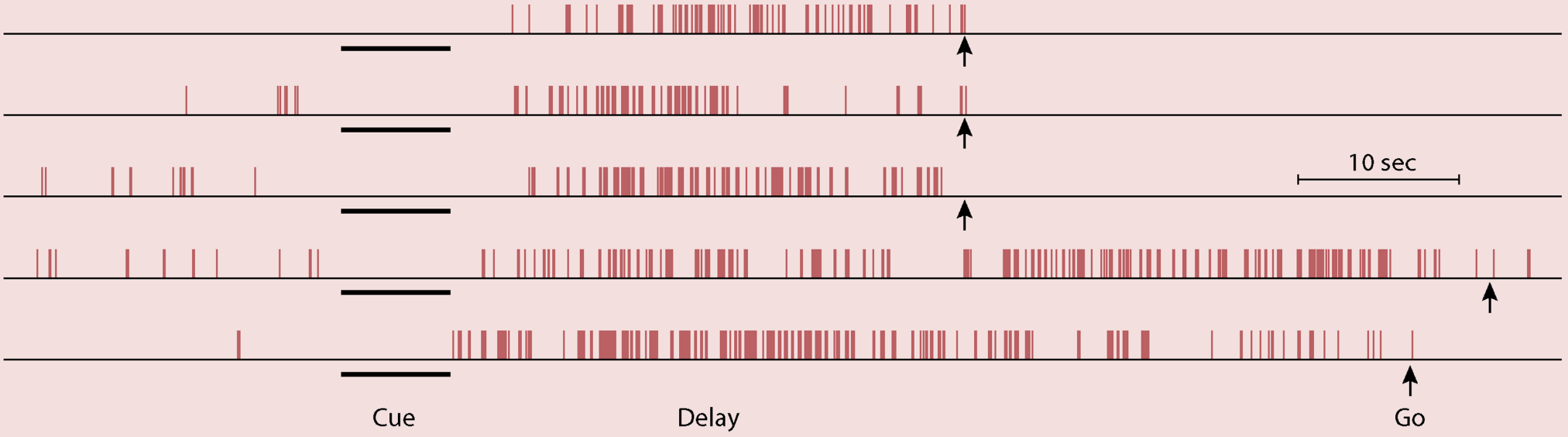
- Left (WHERE task): Monkey shown where food is hidden
- The food wells are hidden for a few moments
- The monkey attempts to find the food
- Must remember where the food is – short-term memory or working memory
- Right (WHAT task): Monkey is shown food hidden underneath a particular symbol card
- The food wells are hidden for a few moments
- After that, the monkey has to remember which symbol covered the food in order to get the food
- Animals with damage to lateral prefrontal cortex have difficulty in these tasks



Working memory: prefrontal cortex cell activity

- Animals shown a cue
- Animal trained to remember the cue location across a delay
- When a "GO" signal is given, then the animal must make a response to the cue

- Dorsolateral prefrontal cortex cells are active during the delay period. They maintain the "working memory" of the cue.
- But what does this cell's activity represent?

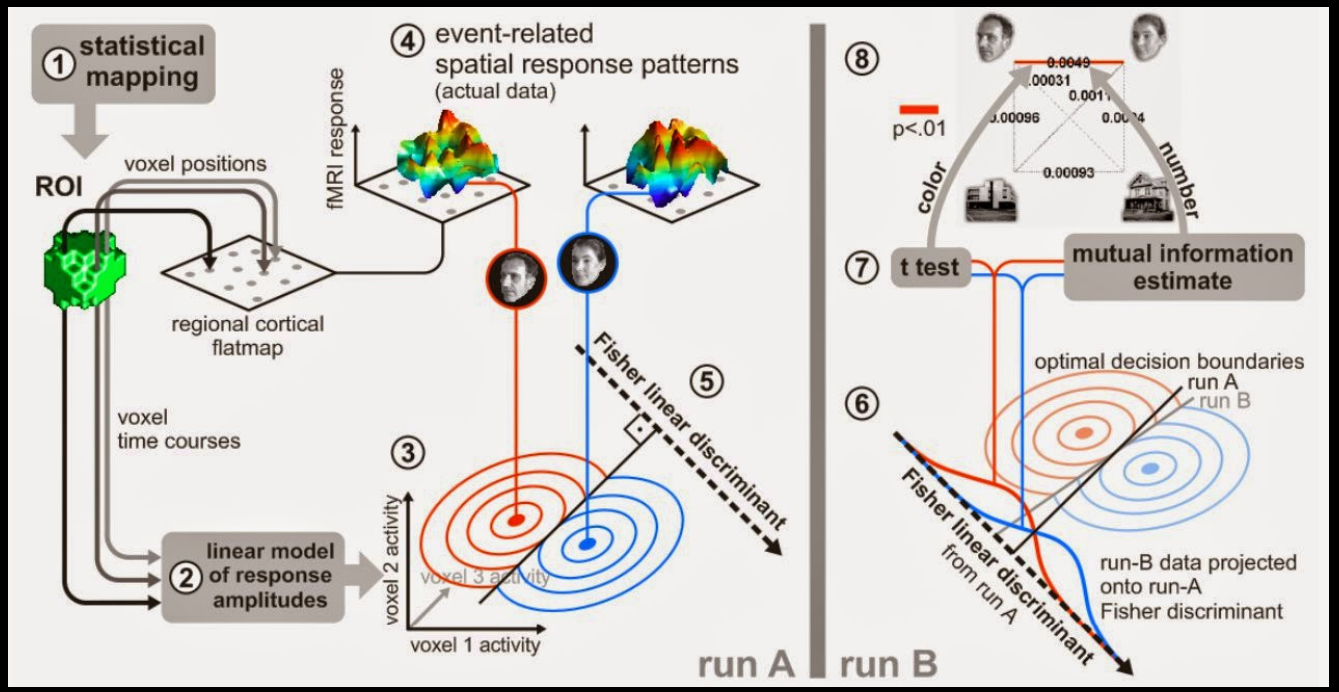


Adapted from Fuster, J.M., *The Prefrontal Cortex: Anatomy, Physiology, and Neuropsychology of the Frontal Lobe*, 2nd edition. New York: Raven Press, 1989.

Visual Working memory – where in the brain?

Where are patterns of activity that represent visual working memory in the brain?

fMRI method: MVPA (multi-voxel pattern analysis)



Visual Working memory – posterior cortex

Where are patterns of activity that represent visual working memory in the brain (fMRI, with MVPA)?

Mostly occipital, temporal, and (for trial dimension, direction or speed) parietal

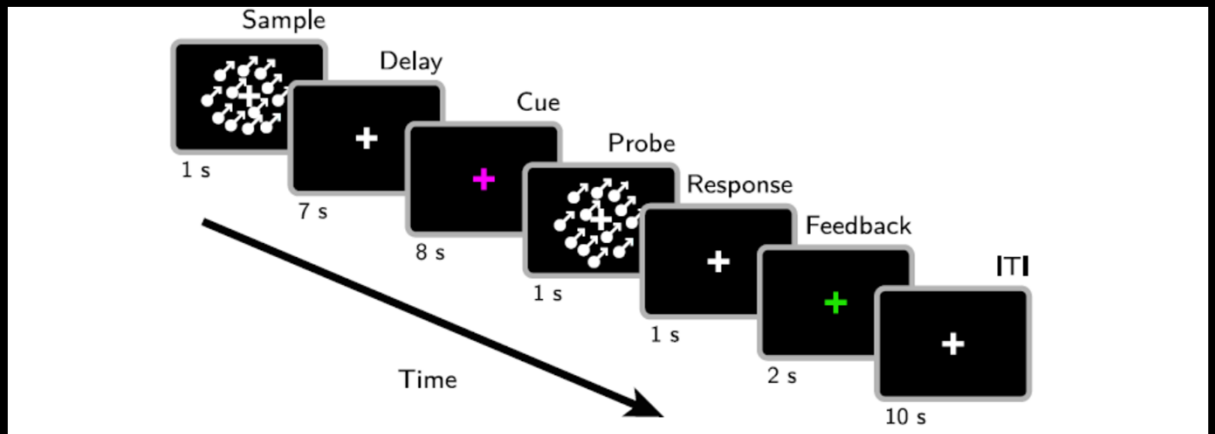


Figure 1. Behavioral task. Participants maintained the direction and speed of a sample motion stimulus over a long delay period. Midway through this delay period, they were cued as to the dimension on which they would be making an upcoming comparison, either direction or speed. At the end of the delay period, they were presented with a probe motion stimulus and had to indicate with a button press whether it matched or did not match the sample stimulus on the cued dimension.

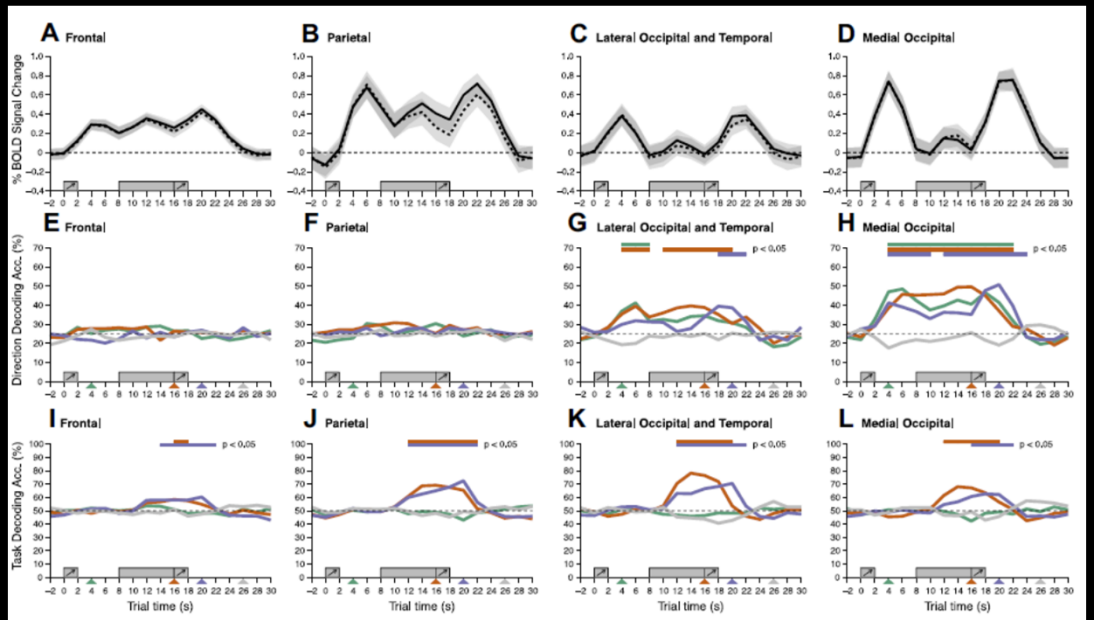
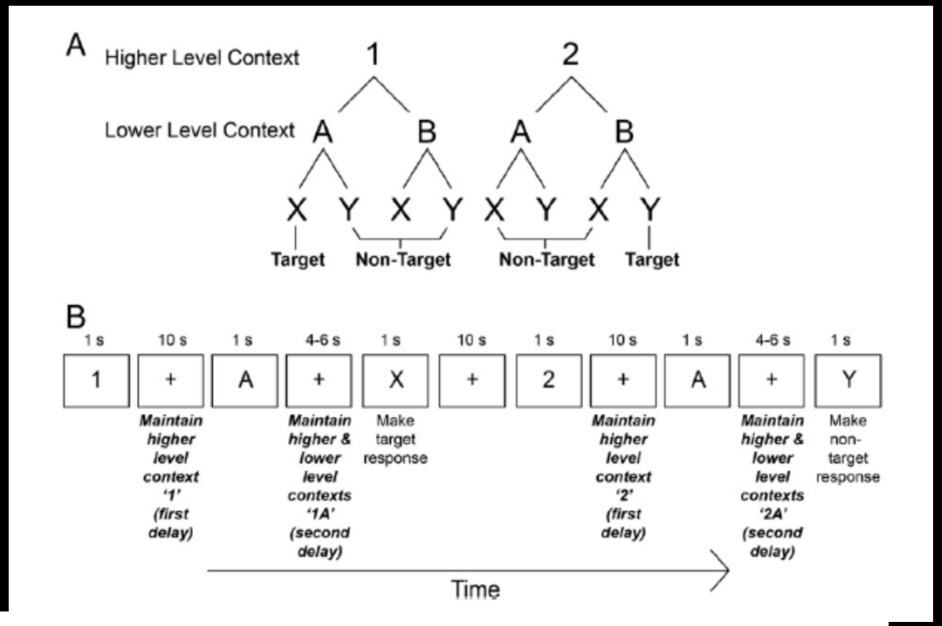


Figure 4. ROI BOLD and decoding time courses. *A–D*, Average ROI BOLD activity. Data from direction-cued trials use solid lines, and speed-cued trials use dashed lines; bands cover average SE across subjects. ROI stimulus-direction decoding results (*E–H*) and ROI trial-dimension decoding results (*I–L*). Graphical conventions same as Figure 3. All averaged across individual data from seven subjects.

DOI:10.1523/JNEUROSCI.1892-12.2012

Task working memory – frontal cortex

* For task instructions, working memory decodable from prefrontal cortex, with a hierarchy of representation



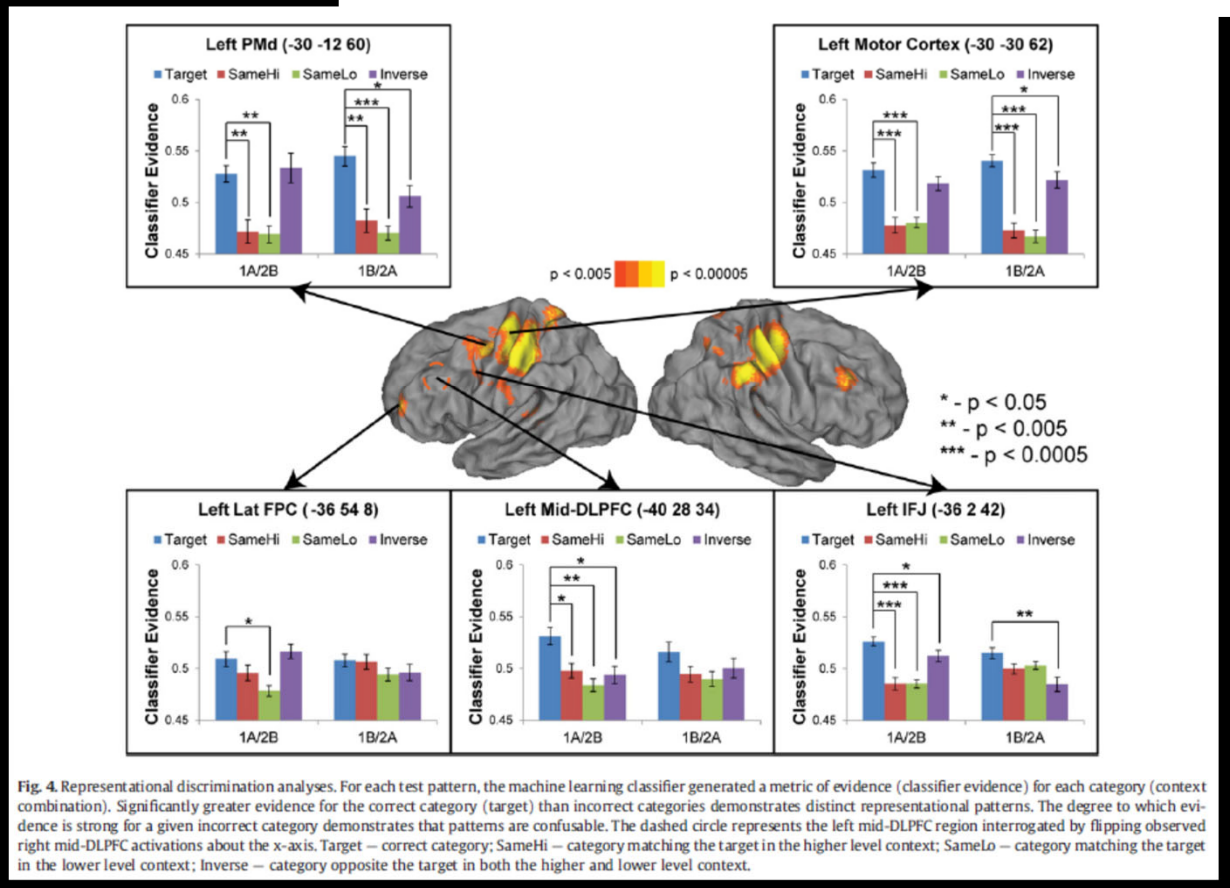
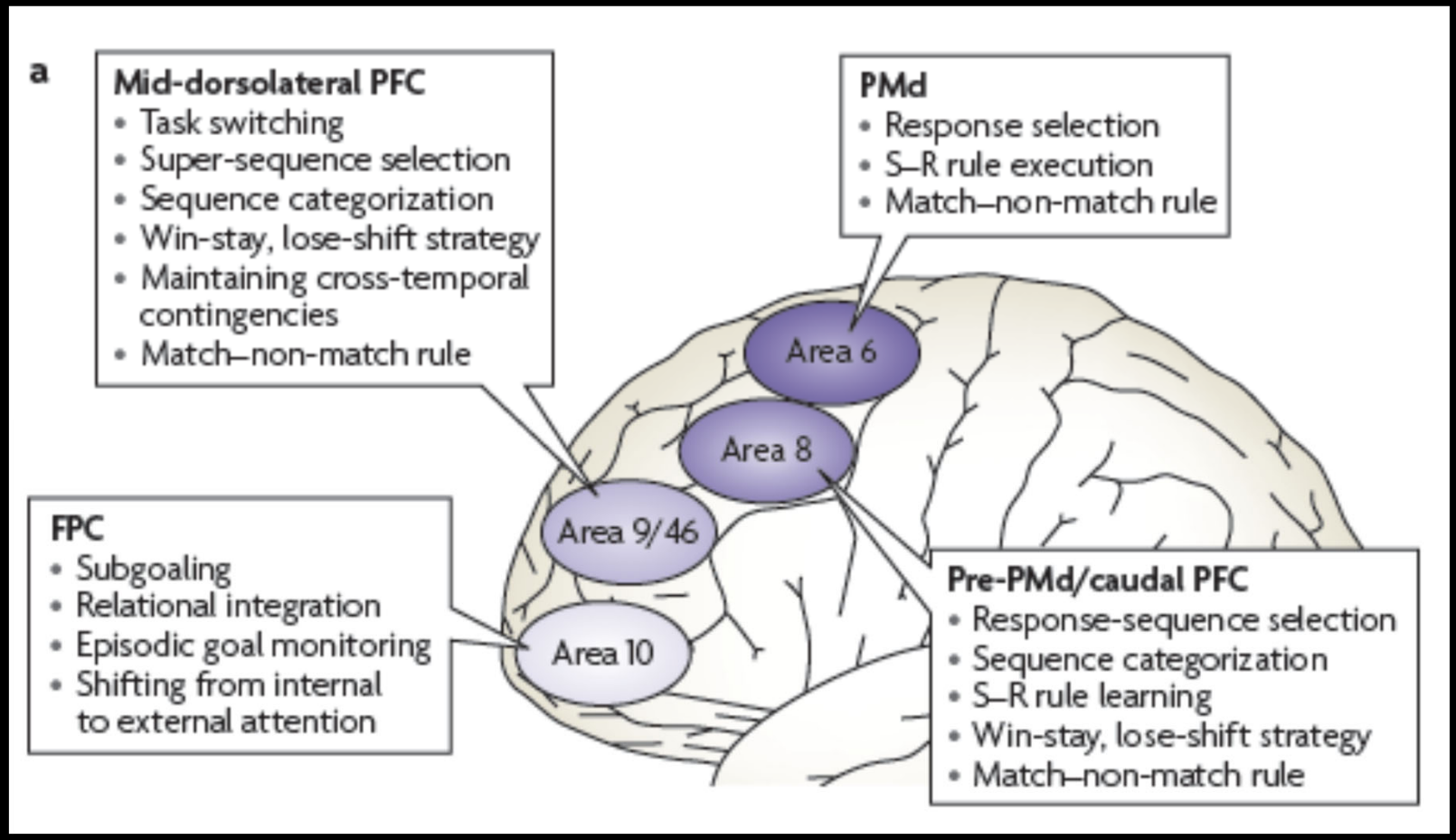
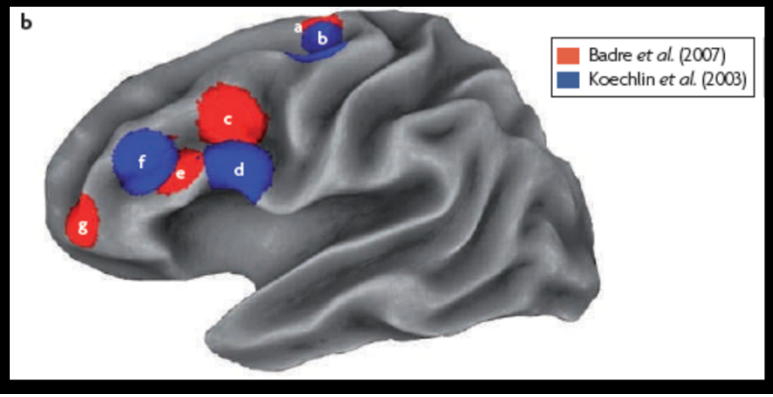


Fig. 4. Representational discrimination analyses. For each test pattern, the machine learning classifier generated a metric of evidence (classifier evidence) for each category (context combination). Significantly greater evidence for the correct category (target) than incorrect categories demonstrates distinct representational patterns. The degree to which evidence is strong for a given incorrect category demonstrates that patterns are confusable. The dashed circle represents the left mid-DLPFC region interrogated by flipping observed right mid-DLPFC activations about the x-axis. Target — correct category; SameHi — category matching the target in the higher level context; SameLo — category matching the target in the lower level context; Inverse — category opposite the target in both the higher and lower level context.

http://dx.doi.org/10.1016/j.neuroimage.2012.08.034

Lateral PFC is hierarchical





Badre & D'Esposito (2009)

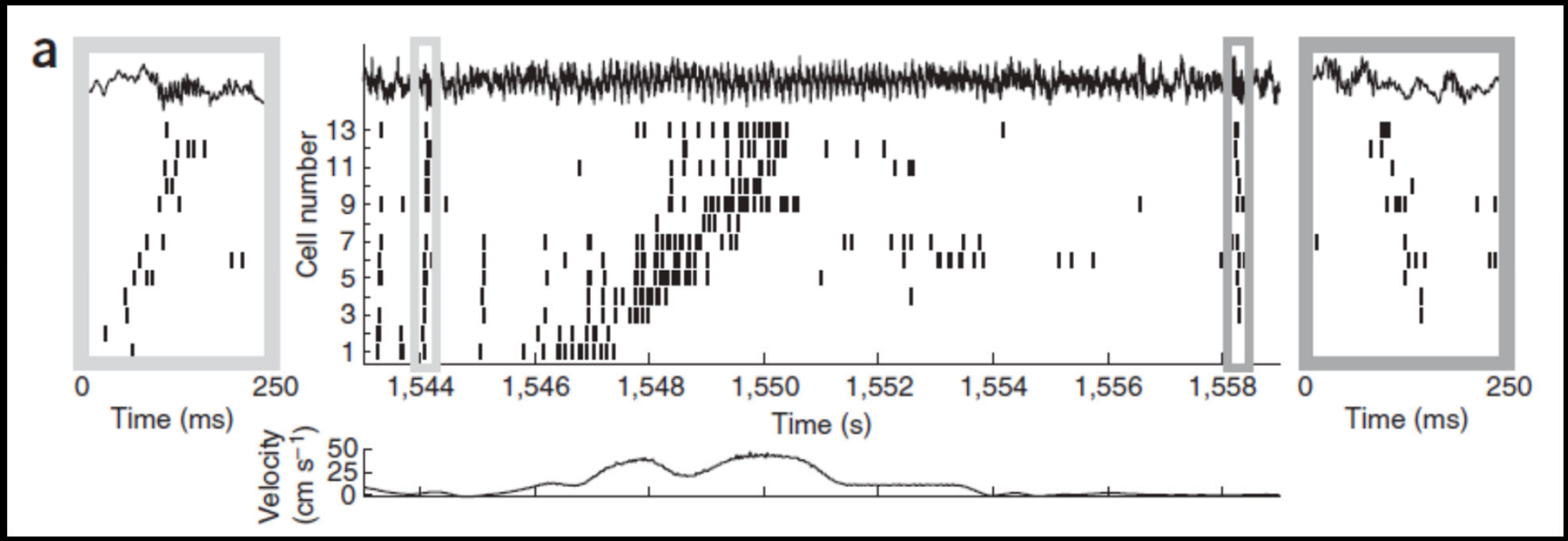
Biological considerations – Backprop through time

- How might recurrent nets be implemented in the brain?
- Could we use a feedback alignment version of recurrent network learning?
- R2N2 model (Reversible Recurrent Neural Network)
  - Use Feedback alignment to do backprop through time
  - Reconstruct past activity by running the net backwards instead of forwards
  - Provides full gradient descent with purely localist information

Cheng, Newman, and Brown (in prep)

R2N2 model

- Clue in the hippocampus: forward and reverse replays



Diba & Buszáki, 2007

R2N2 model

Problems:

- A) Recurrent backprop requires error propagated backward in time. How could the brain do that??
- B) Also requires a memory of each neuron's activity at each point in time in the past. How??

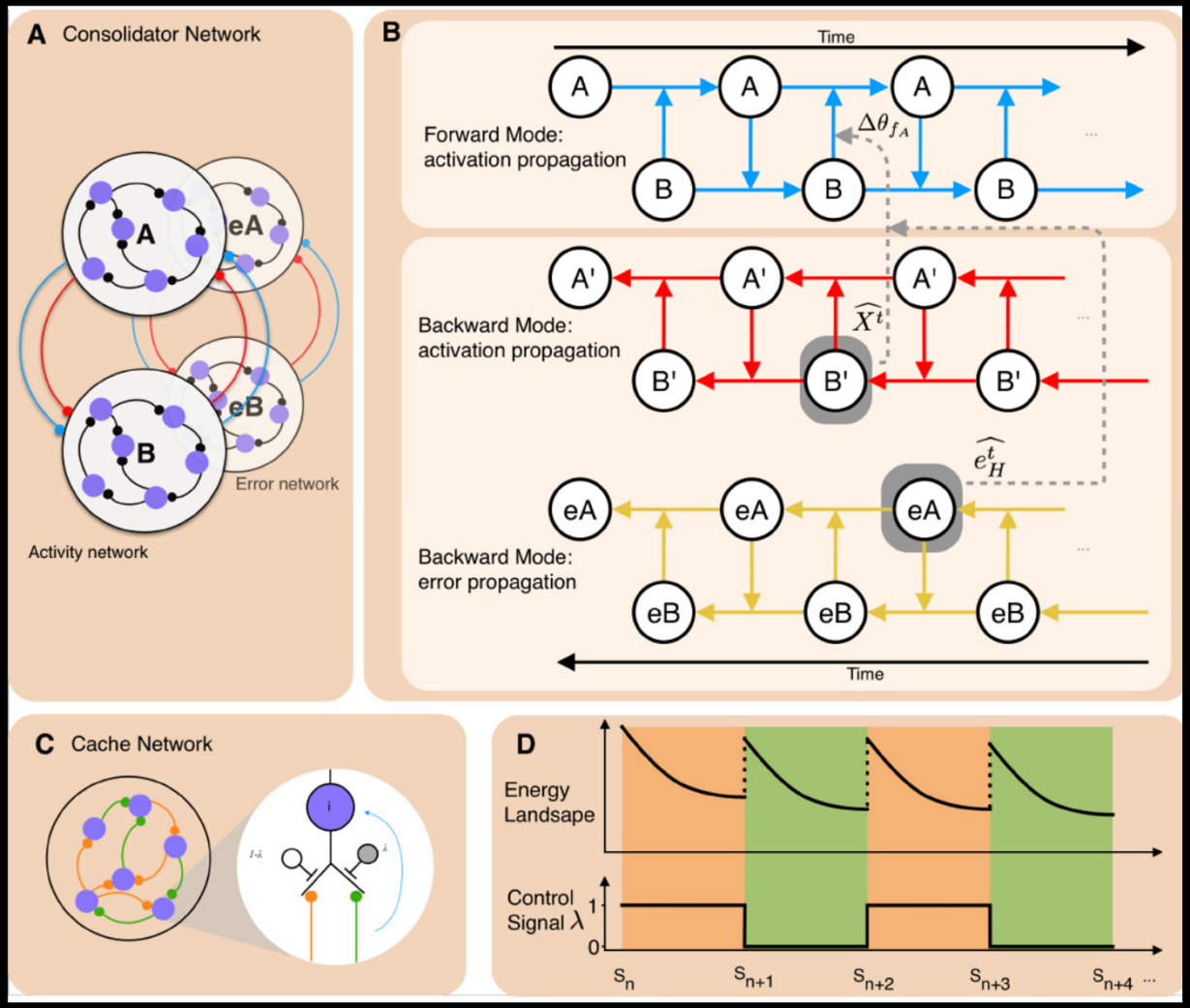
Solutions:

- The Reversible Recurrent Neural Network (R2N2)
- A') Recurrent feedback alignment can propagate error E for N steps backwards in time – B^N*E

- B') Instead of remembering activity at each point in time, simply reconstruct it by running the network backward!
- Delta rule = learning_rate * error * activity
- (BUT must store the sequence initially, then play it in reverse to TRAIN the network – where stored initially?)

Cheng, Newman, and Brown (in prep)

R2N2 model



consolidator (different weights become active, learn for forward vs. reverse)

Backward error propagation

Cache (2 Hopfield networks oscillate, serve as “pump”)

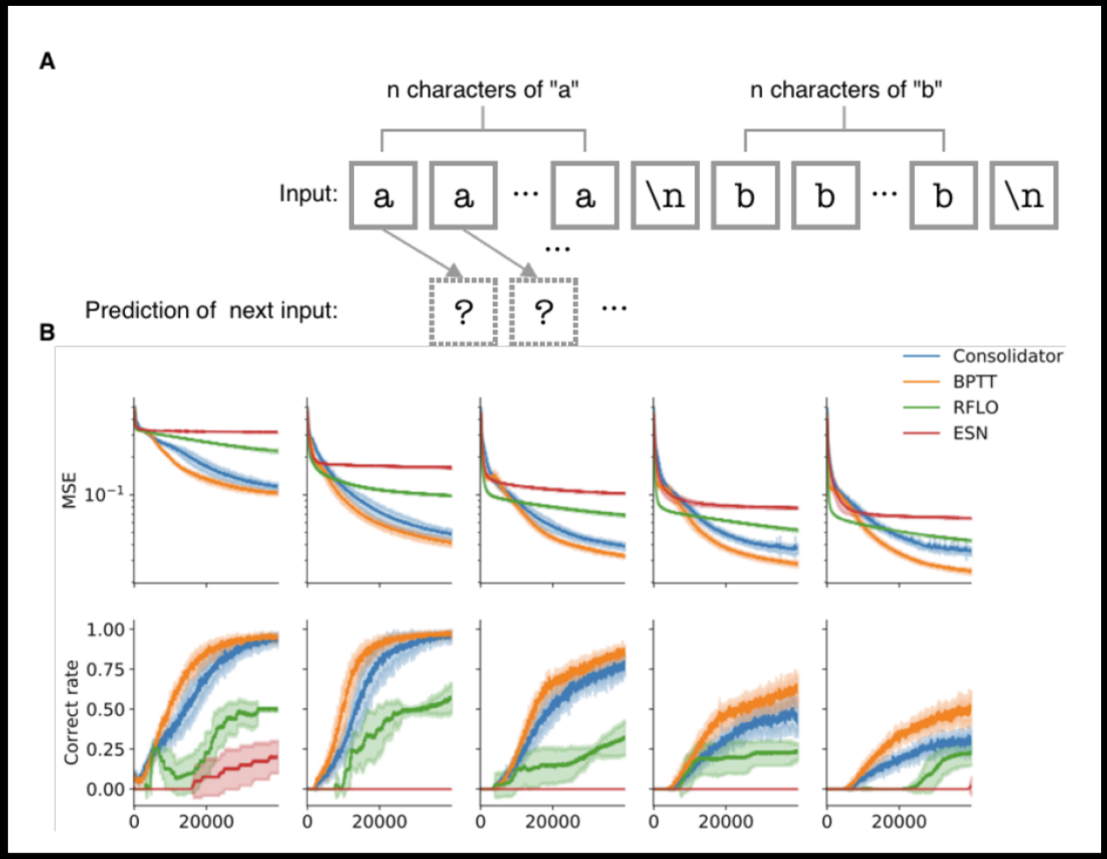
Cheng, Newman, and Brown (in prep)

R2N2 model – how it works

1. Learn a sequence one-shot in a temporary cache network
2. Run a longer term storage consolidator network forward, and compute the output error (E) after T time steps
3. Run 3 networks in reverse simultaneously for each t:
    1. The cache, to calculate the desired output at time T-t
    2. The consolidator, to calculate the previous activity at T-t
    3. The recurrent error B^t*E

4. → At each step t, compute the delta rule and update weights:
    1. dW = learning_rate * Error * previous activity

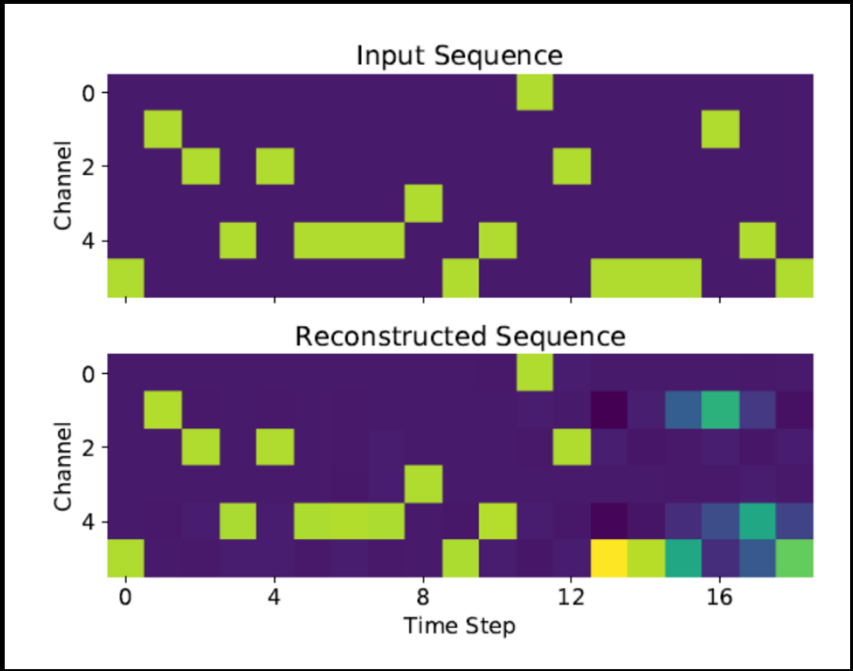
R2N2 model



- Outperforms previous biological sequence learning models
- Task: predict the input at t+1 given sequence [0,t] – predictive coding
- AnBn task – e.g. if n=3, sequence is "AAABBBAAABBB…"

Cheng, Newman, and Brown (in prep)

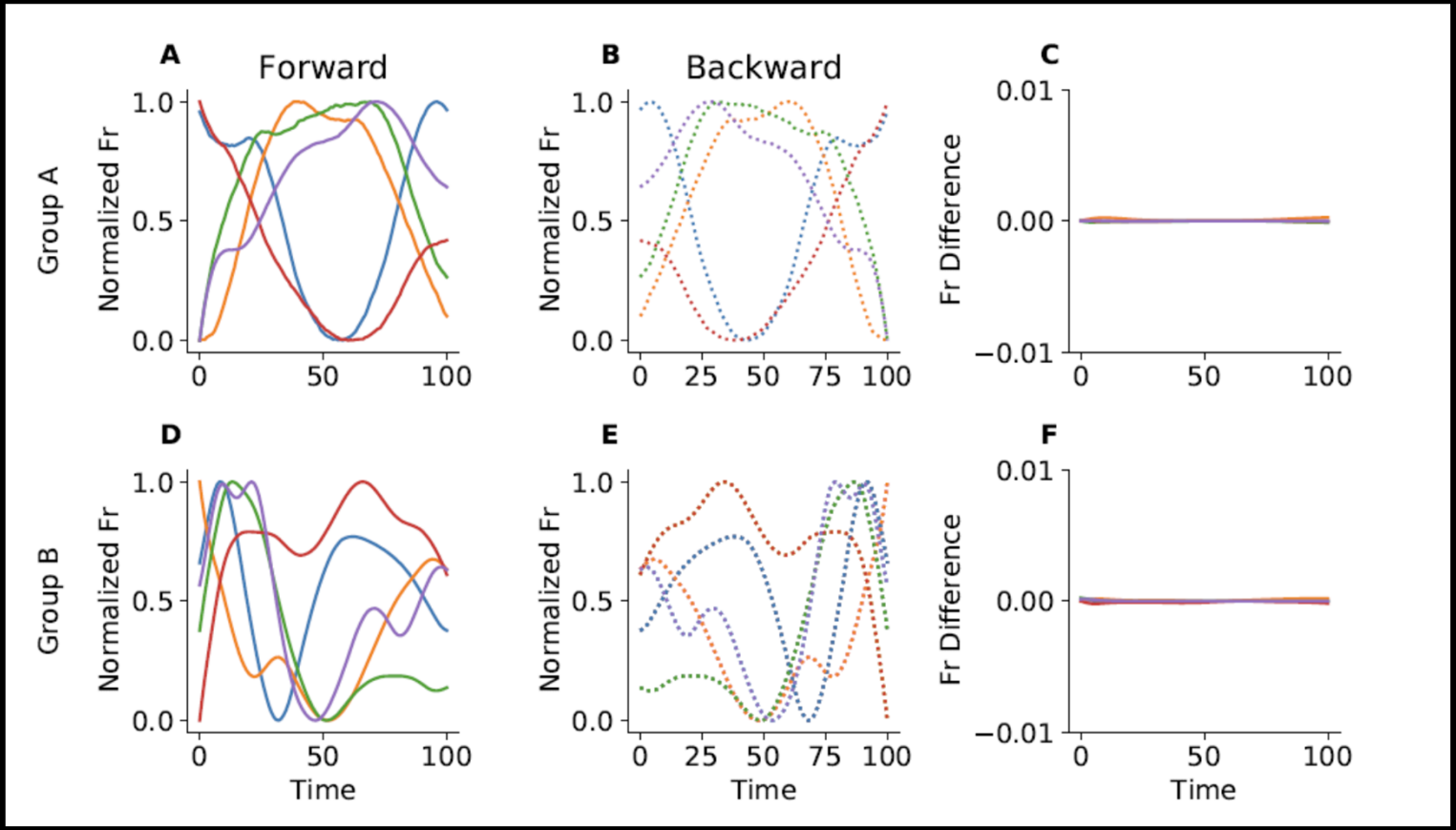
R2N2 model



- Consolidator learns to reconstruct sequence forward

Cheng, Newman, and Brown (in prep)

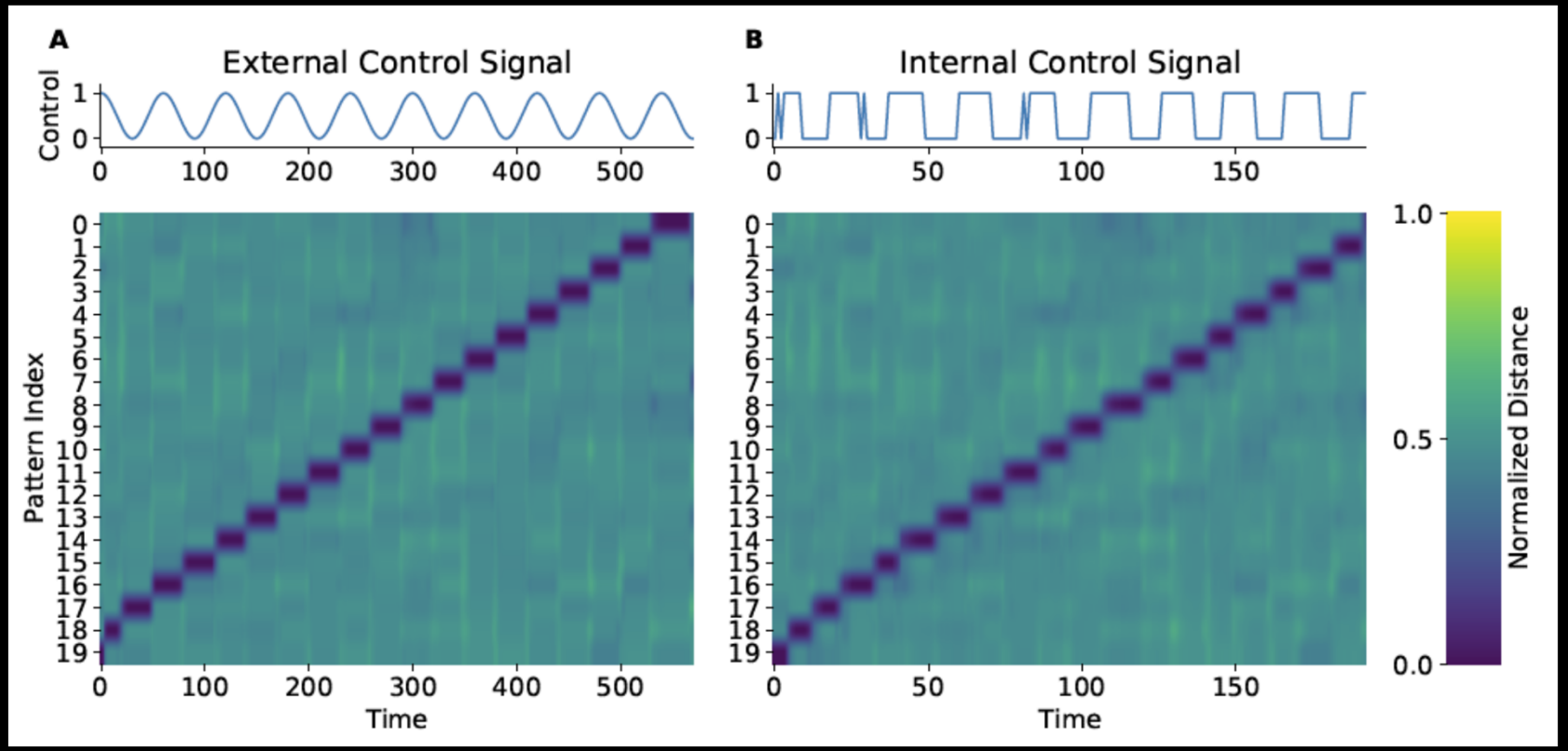
R2N2 model



- Consolidator can replay both forward and backward

Cheng, Newman, and Brown (in prep)

R2N2 model - Cache

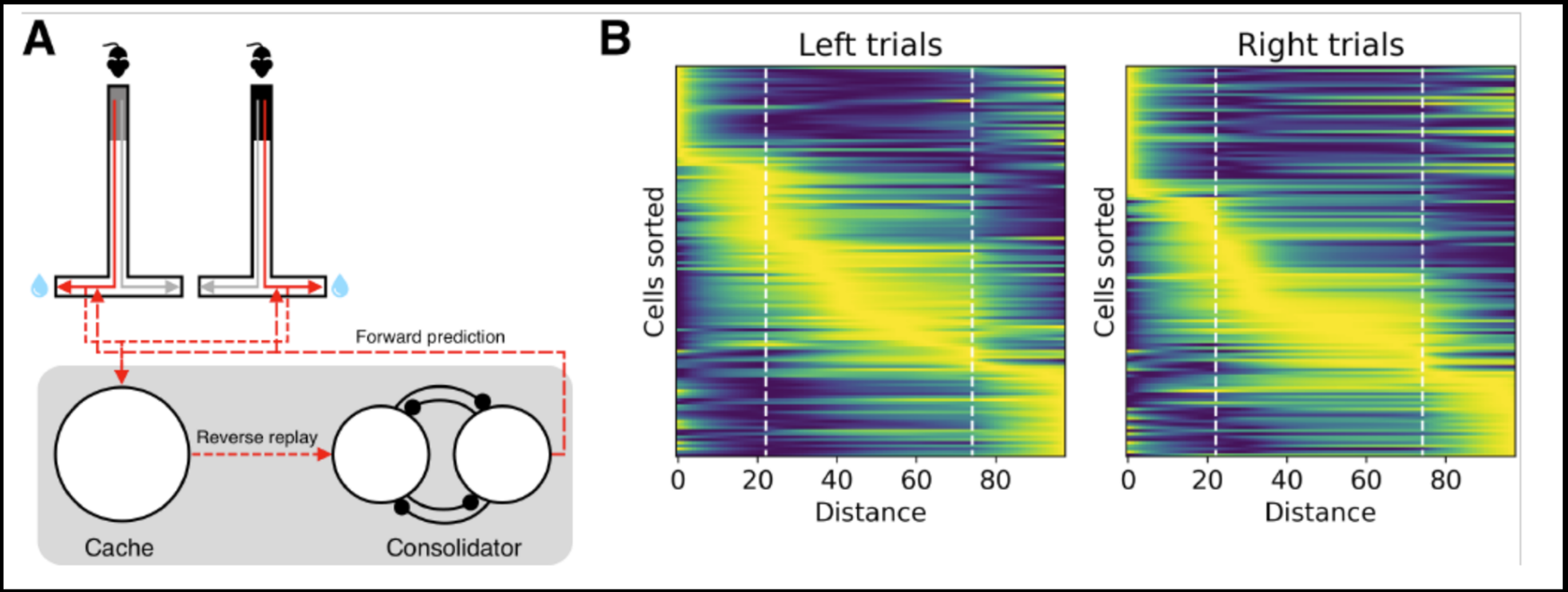


- One-shot learning in the cache, plays back learned arbitrary sequence

Cheng, Newman, and Brown (in prep)

Consolidator place cells

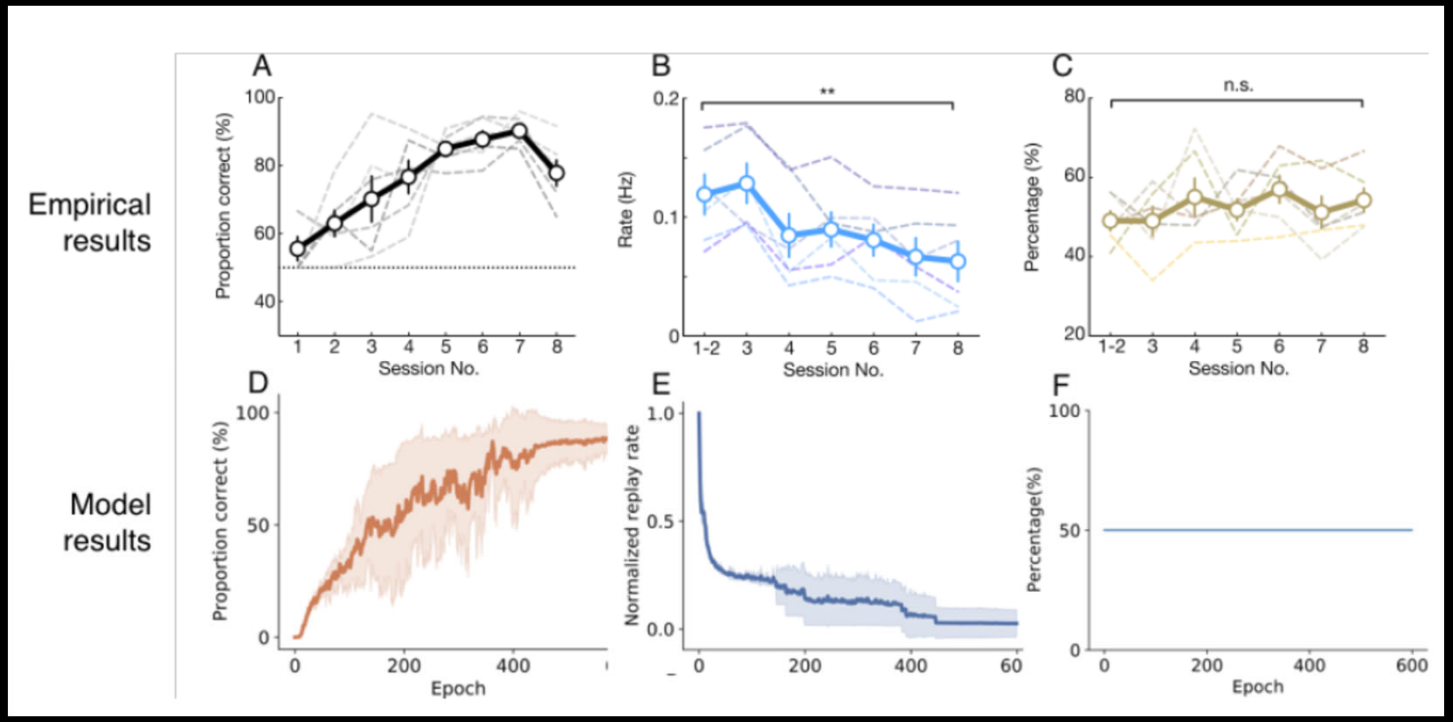
- Combined model with Consolidator and Cache, learning T-maze, shows place cells



Cheng, Newman, and Brown (in prep)

R2N2 model

- A) Performance improves steadily
- B) During training, the replay rate decreases steadily
- C) During training, the proportion of forward and reverse replays remains even (forward replay trains backward circuits, & vice versa)



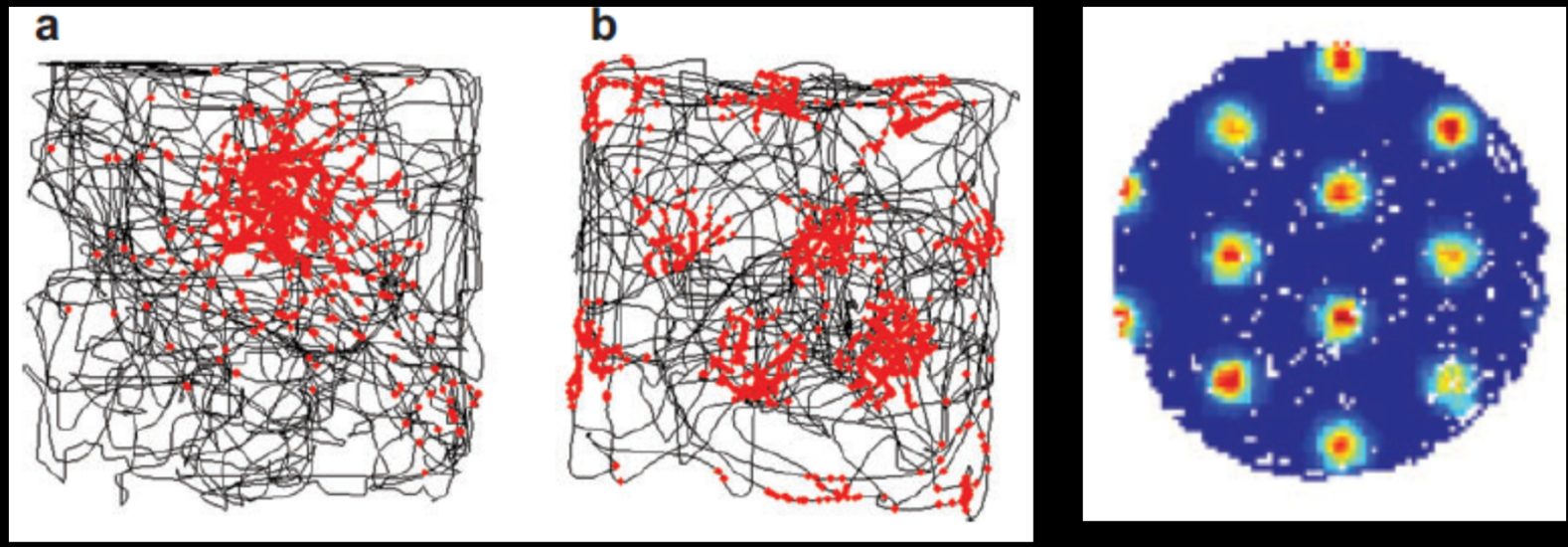
Empirical data from Shin et al. 2019

Cheng, Newman, and Brown (in prep)

RNNs and information representation

- The basic question: when a cell is active (lots of action potentials per second), what information does that represent?
- Place cells (in the hippocampus) – active when animal (rodent) at a particular location
- Grid cells – active when the animal is at a particular set of locations, which are arranged in a hexagonal grid
- Receptive field – the particular stimulus conditions which cause a cell to fire

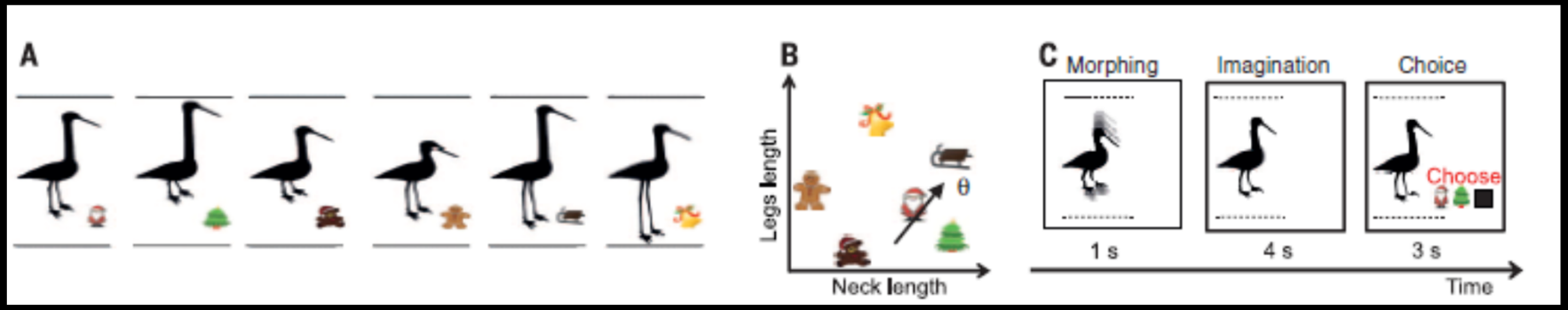
Place cell **Grid Cell** **Grid Cell**



Moser, 2008, Ann Rev Neuro

RNNs and information representation

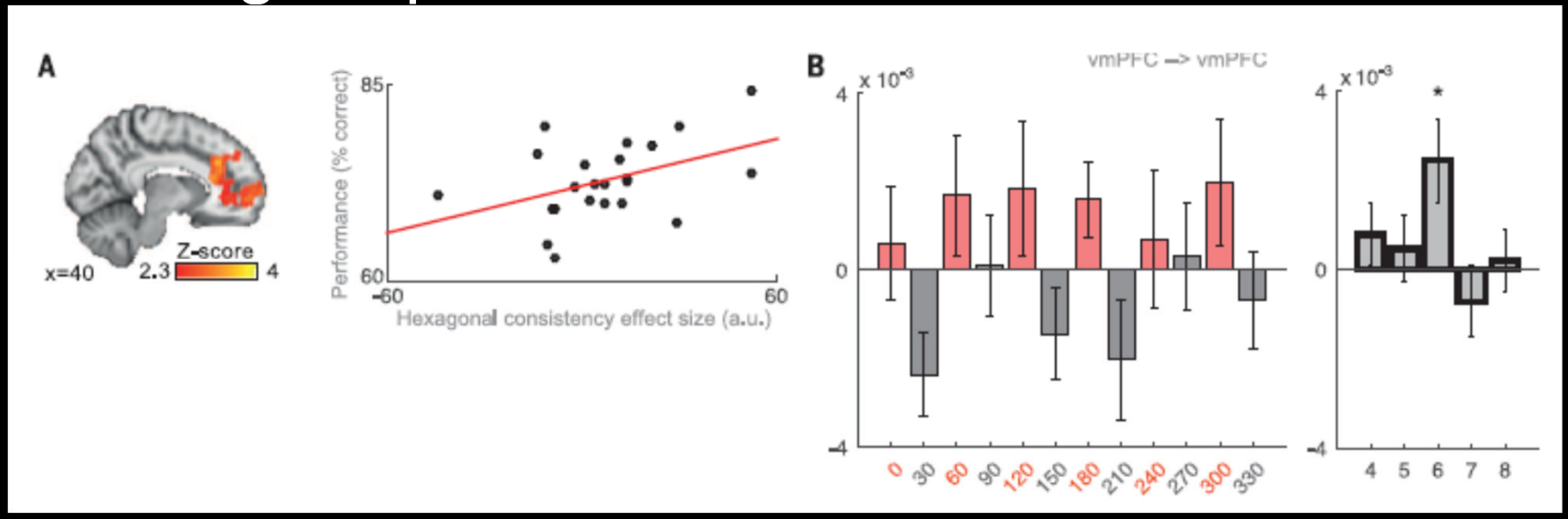
- Are grid cells only about 2D spatial representation? No
- Subjects trained to morph bird images in "bird space" with 2 dimensions – neck length, and leg length. Each bird morphing followed a direction (angle θ) in bird space.



Constantinescu et al. 2016, Science

RNNs and information representation

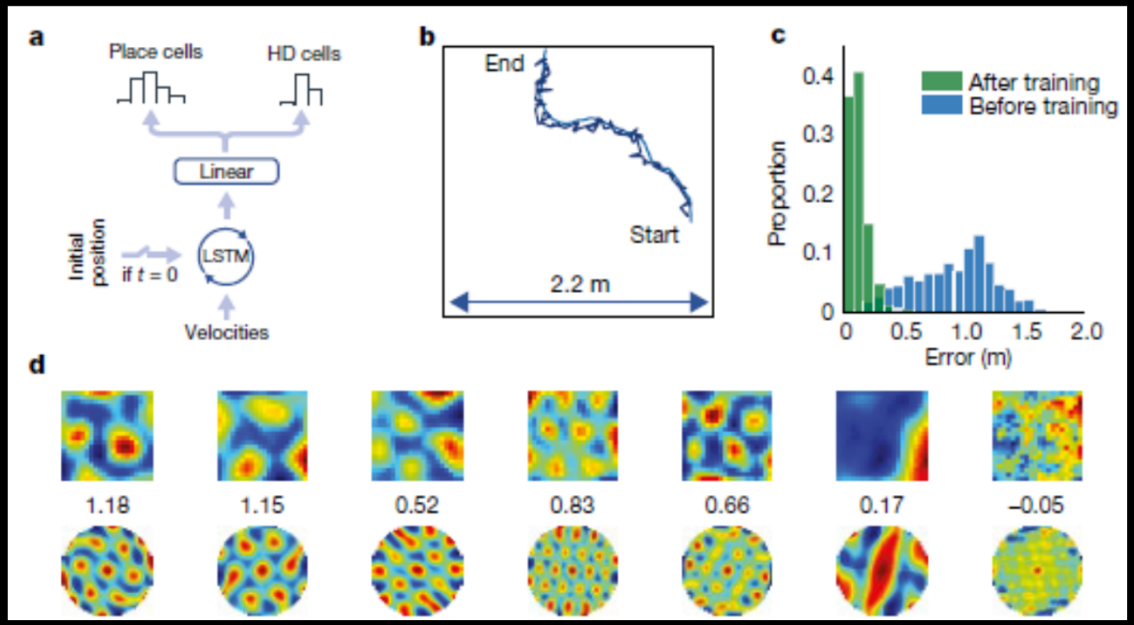
- Results show that brain areas including ventromedial prefrontal cortex are more active when the trajectory in “bird space” follows a multiple of 60 degrees, but are less active when following between one of the 60 degree directions.
- Confirms “grid cell” like representation of abstract knowledge representation

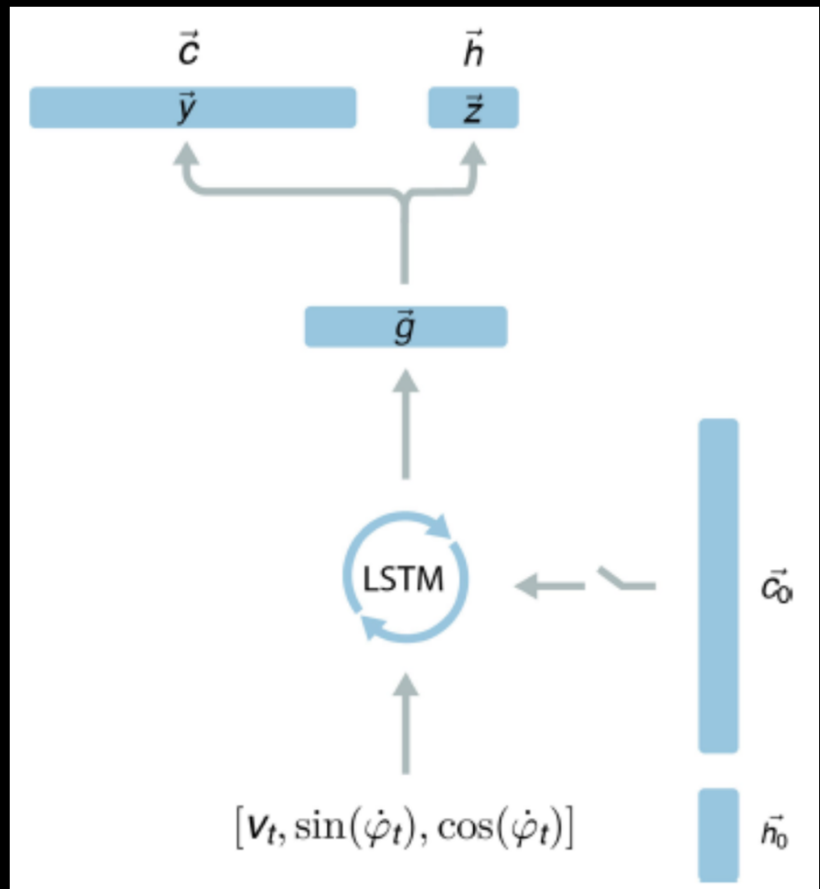


Constantinescu et al. 2016, Science

RNNs and information representation

- Do deep neural networks show hexagonal knowledge representation??
- Train a deep LSTM network:
  - Input location, head direction
  - Output agent position, head direction
- Result: LSTM linear layer (g) shows grid cell representation, BUT only when dropout (20%) is applied during training!!





Extended Data Fig. 1 | Network architecture in the supervised learning experiment. The recurrent layer of the grid cell network is an LSTM with 128 hidden units. The recurrent layer receives as input the vector $[\vec{v}, \sin(\dot{\varphi}), \cos(\dot{\varphi})]$. The initial cell state and hidden state of the LSTM, $\vec{l}_0$ and $\vec{m}_0$, respectively, are initialized by computing a linear transformation of the ground truth place $\vec{c}_0$ and head-direction activity $\vec{h}_0$ at time 0. The output of the LSTM is followed by a linear layer on which dropout is applied. The output of the linear layer, $\vec{g}_t$, is linearly transformed and passed to two softmax functions that calculate the predicted head direction cell activity, $\vec{z}_t$, and place cell activity, $\vec{y}_t$. We found evidence of grid-like and head direction-like units in the linear layer activations $\vec{g}_t$.

Banino et al, 2018, Nature