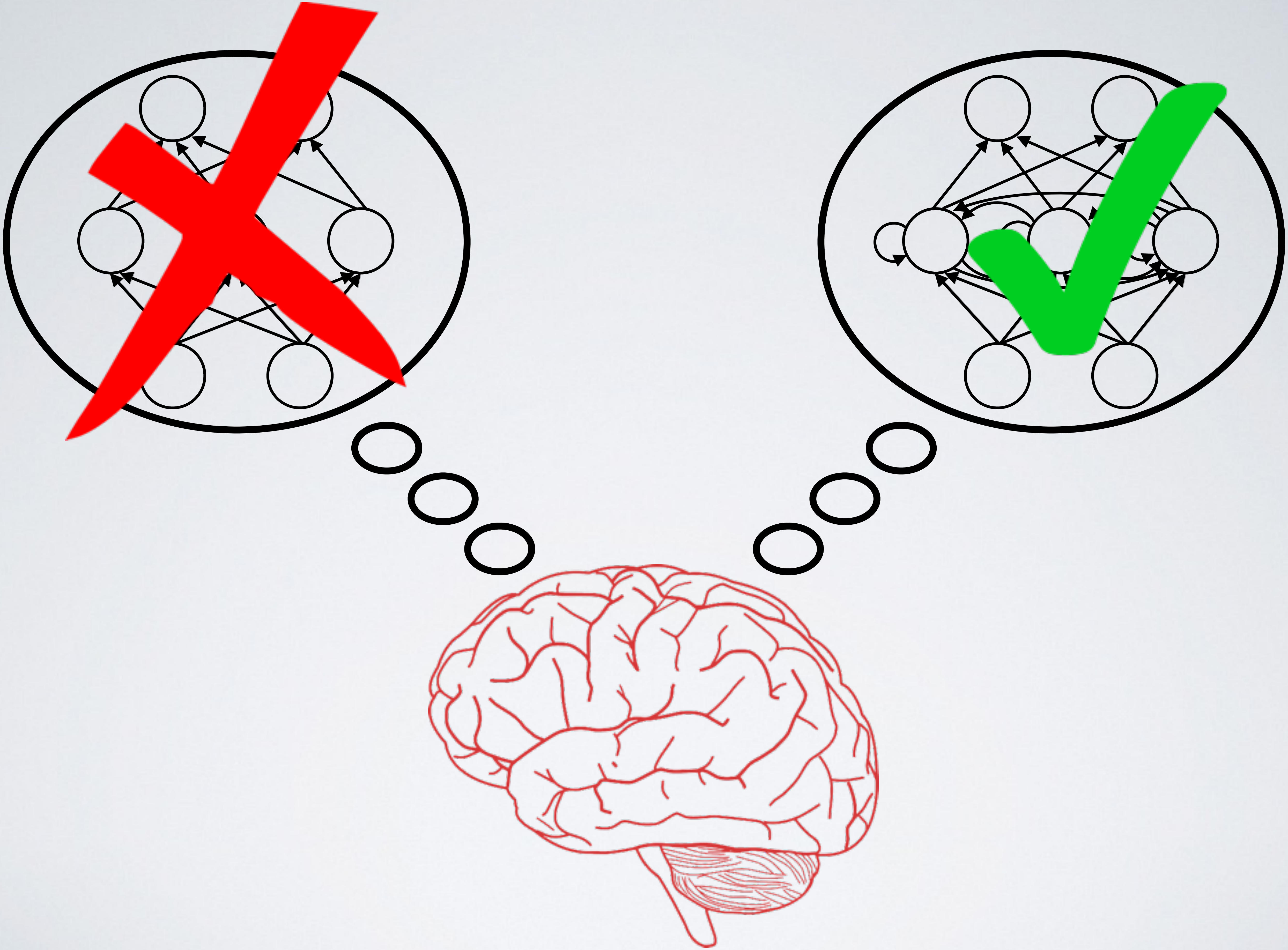
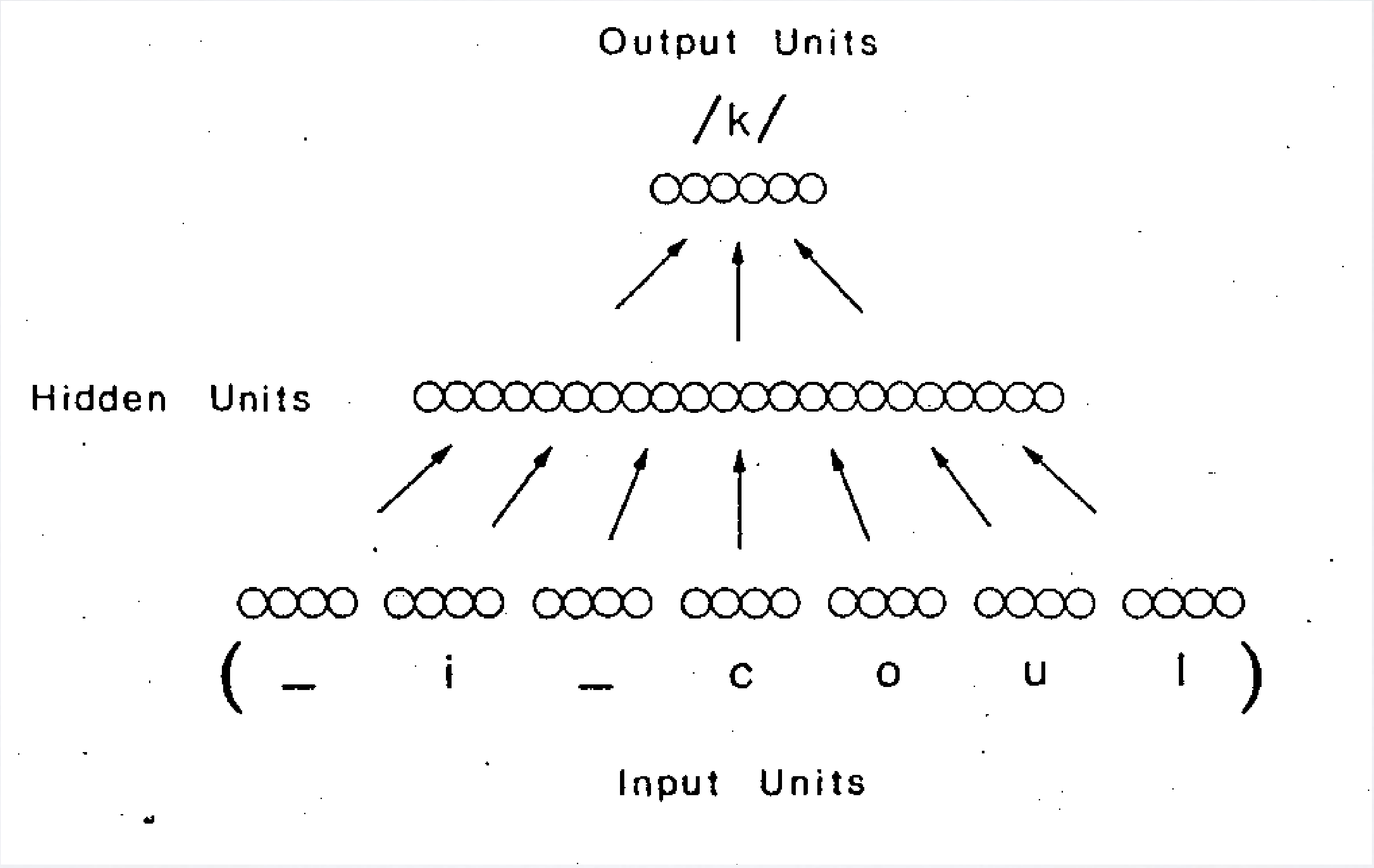
Learning with general recurrent neural networks

Guy Isely



Feedforward neural networks are "timeless"



Some problems with a feedforward model of temporal processes

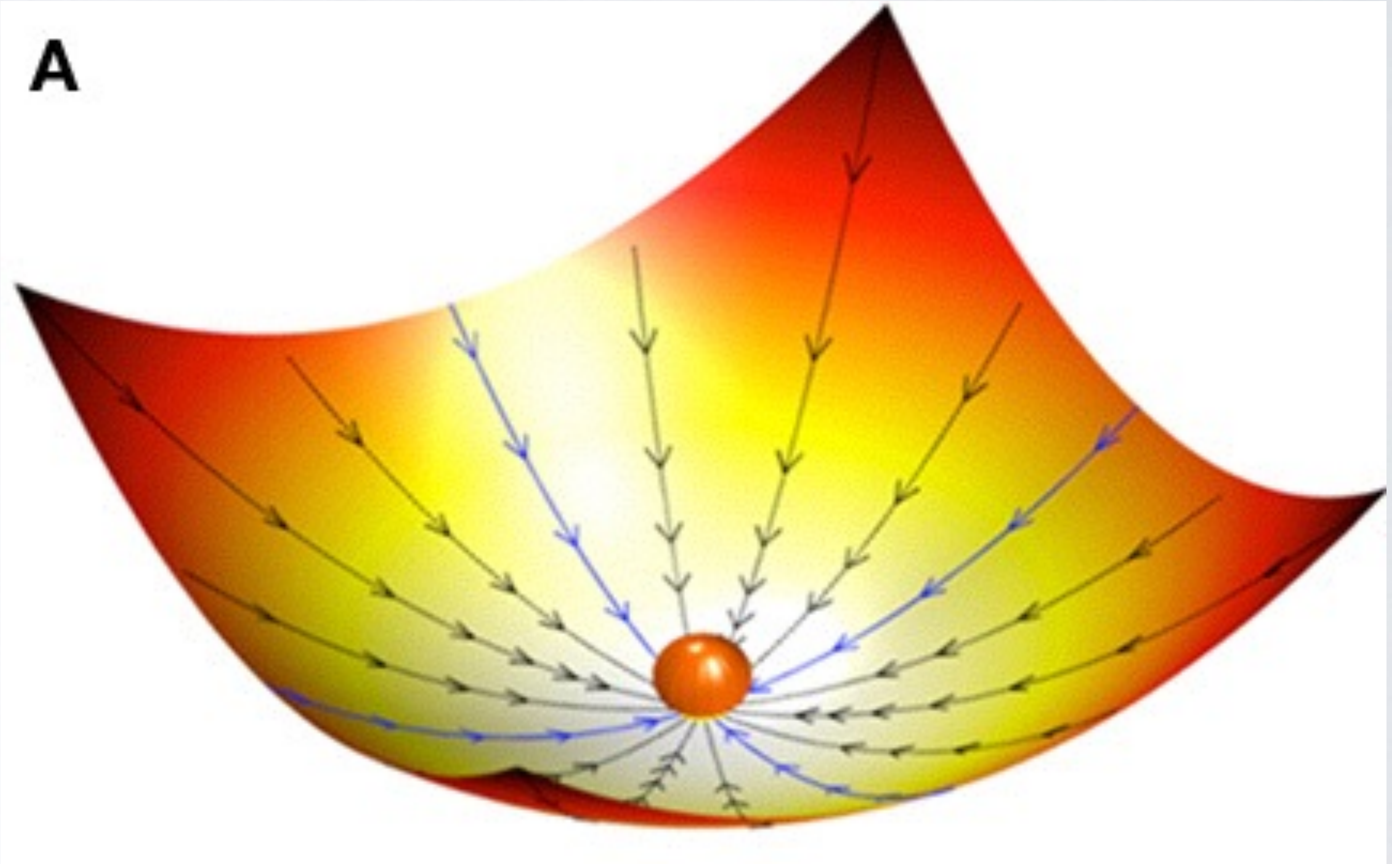
- Computational cost grows with temporal duration modeled
- Can't capture long-time contextual dependencies in sequences
- Networks don't have persistent state— "noise correlations" might be state!

Hopfield networks have fixed-point attractor dynamics

- Dynamics are gradient descent on an energy function (the lyapunov function)

- Autonomous after initial input

- Guaranteed to converge to a stable fixed point due to symmetric connectivity matrix

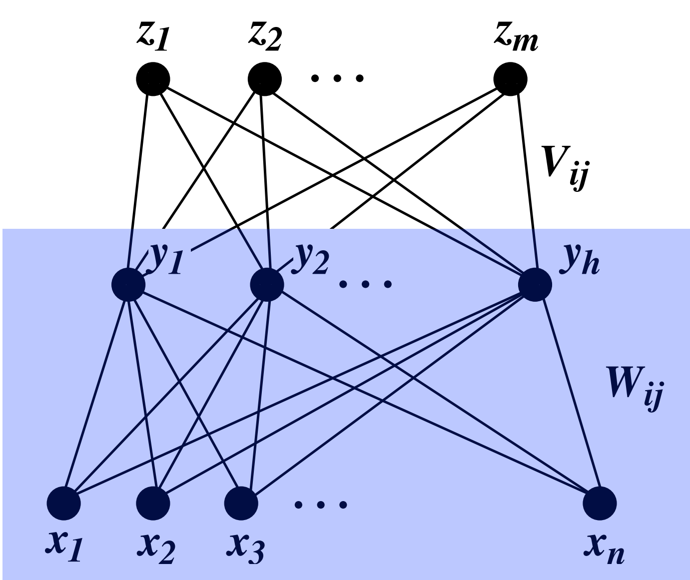


Can we you use gradient descent to train general RNNs?

Yes, yes we can!

…but there's a wrinkle.

Backpropagation review



$$\begin{aligned} \Delta W_{kl} &= -\eta \frac{\partial E}{\partial W_{kl}} \\ &= \eta \sum_i \left[T_i - z_i(\mathbf{x})\right] \frac{\partial z_i(\mathbf{x})}{\partial W_{kl}} . \end{aligned}$$

$$\frac{\partial z_i(\mathbf{x})}{\partial W_{kl}} = \frac{\partial z_i(\mathbf{x})}{\partial y_k} \frac{\partial y_k}{\partial W_{kl}}$$

It's just the chain rule!

$$\begin{aligned} \Delta W_{kl} &= \eta \sum_i \left[T_i - z_i(\mathbf{x})\right] \sigma'(u_{z_i}) V_{ik}\, \sigma'(u_{y_k})\, x_l \\ &= \boxed{\eta\, \delta_{y_k}\, x_l} \end{aligned}$$

where $\delta_{y_k} = \sigma'(u_{y_k}) \sum_i \delta_{z_i} V_{ik}$

back-propagation of error

Can we apply backpropagation directly to an RNN?

- Not exactly— the gradient of a RNN's error function w.r.t. to the weights depends on the network's state at all previous time steps.
- But we can unravel the network structure in time to get a feedforward network and perform backprop on this network.
- This is called backpropagation through time (BPTT).

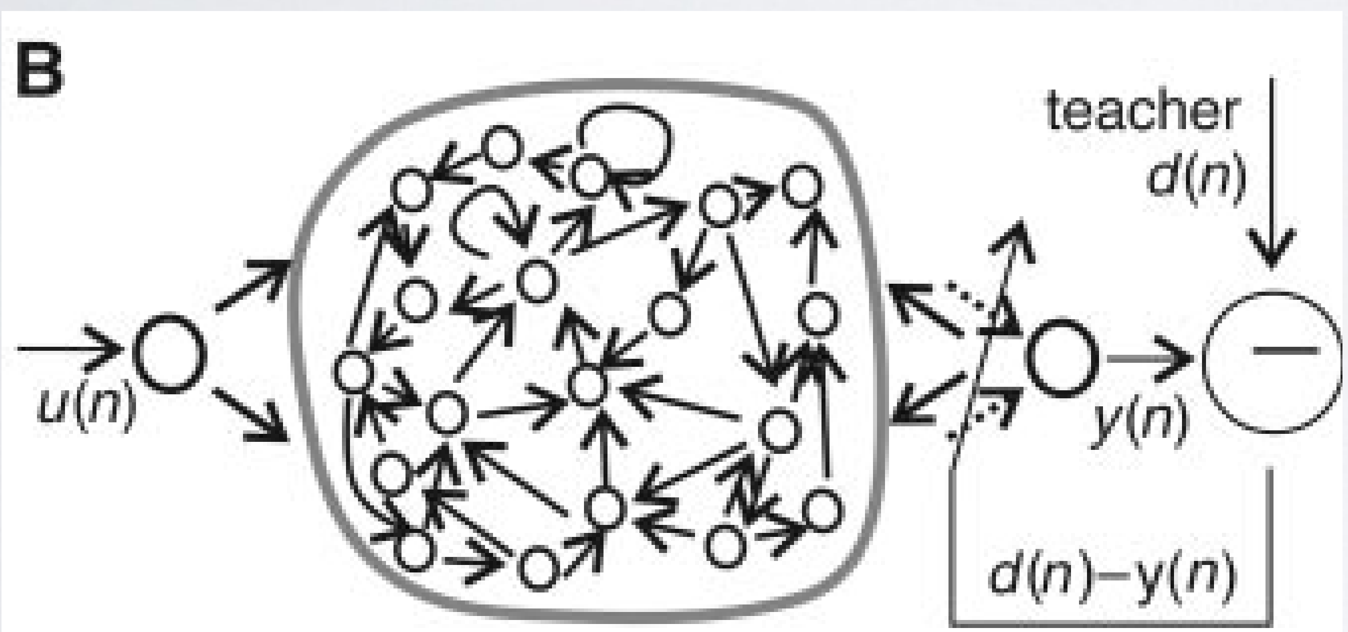
Realtime recurrent learning (Williams and Zipser 1989)

- $$\frac{\partial y(t)}{\partial W_r} = \text{diag}(\sigma'(y(t)))W^\top \cdot \frac{\partial y(t-1)}{\partial W_r}$$
- We can run the recurrence relation underlying the gradient computation forward in time!
- Downside: BPTT is $O(tn^2)$ per time step but RTRL is $O(n^3)$ per time step— prohibitive for large networks!

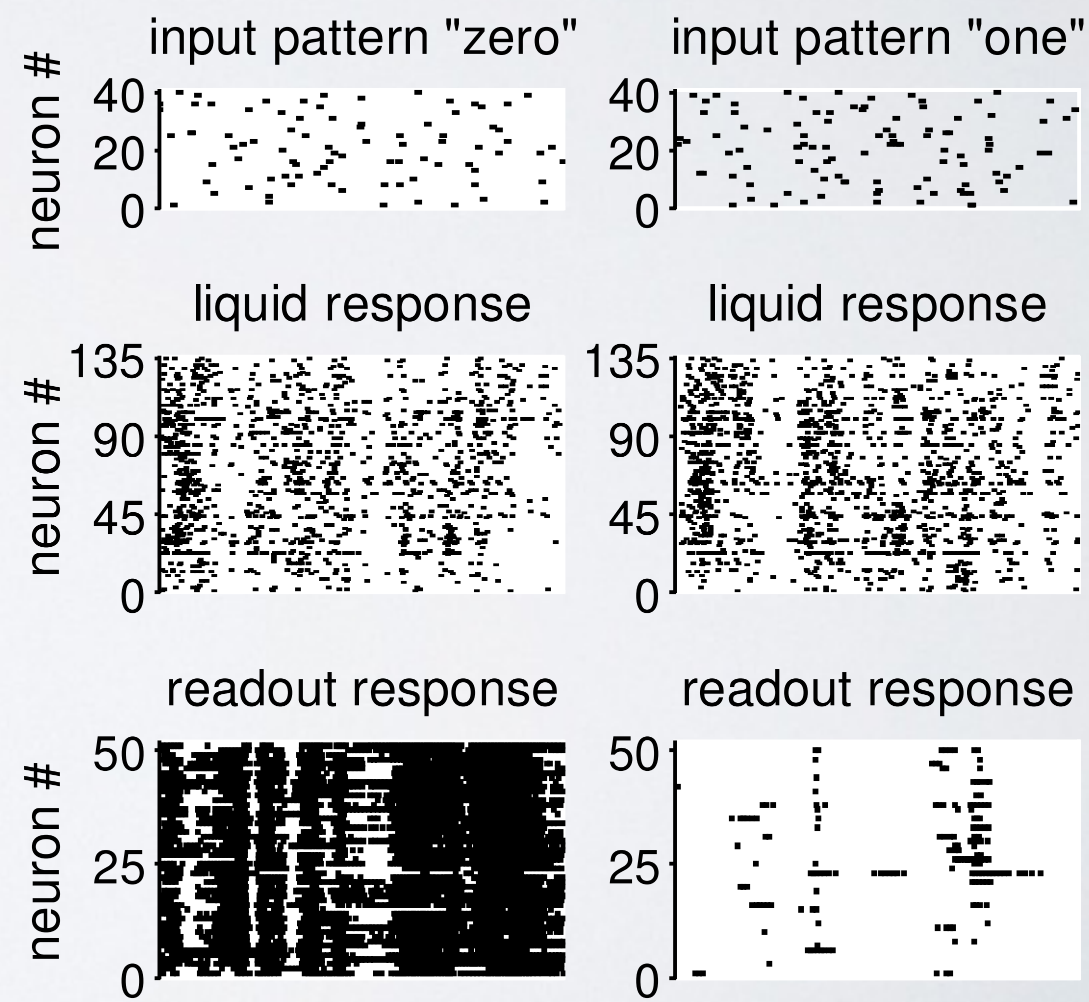
Intermission
(aka neural network winter)

Reservoir computing

Echo State Networks
(Jaeger & Haas 2004)



Liquid State Machines
(Maass et al. 2002)



Ideas from Echo State Networks

- Use an unoptimized random sparsely connected recurrent reservoir and do a linear readout.
- Only optimize the readout weights.
- Use **teacher forcing** to achieve appropriately tuned the reservoir dynamics

BPTT returns (with a vengeance)

Where to next?

- Address vanishing/exploding sensitivity problem with network units designed for specific temporal dynamics (e.g. Long Short Term Memory)

- Move beyond gradient descent based approaches to optimizing network parameters

- Incorporate addition biophysical features of real networks (e.g. STDP, metabotropic receptor dynamics, gap junctions, dendritic non-linearities)