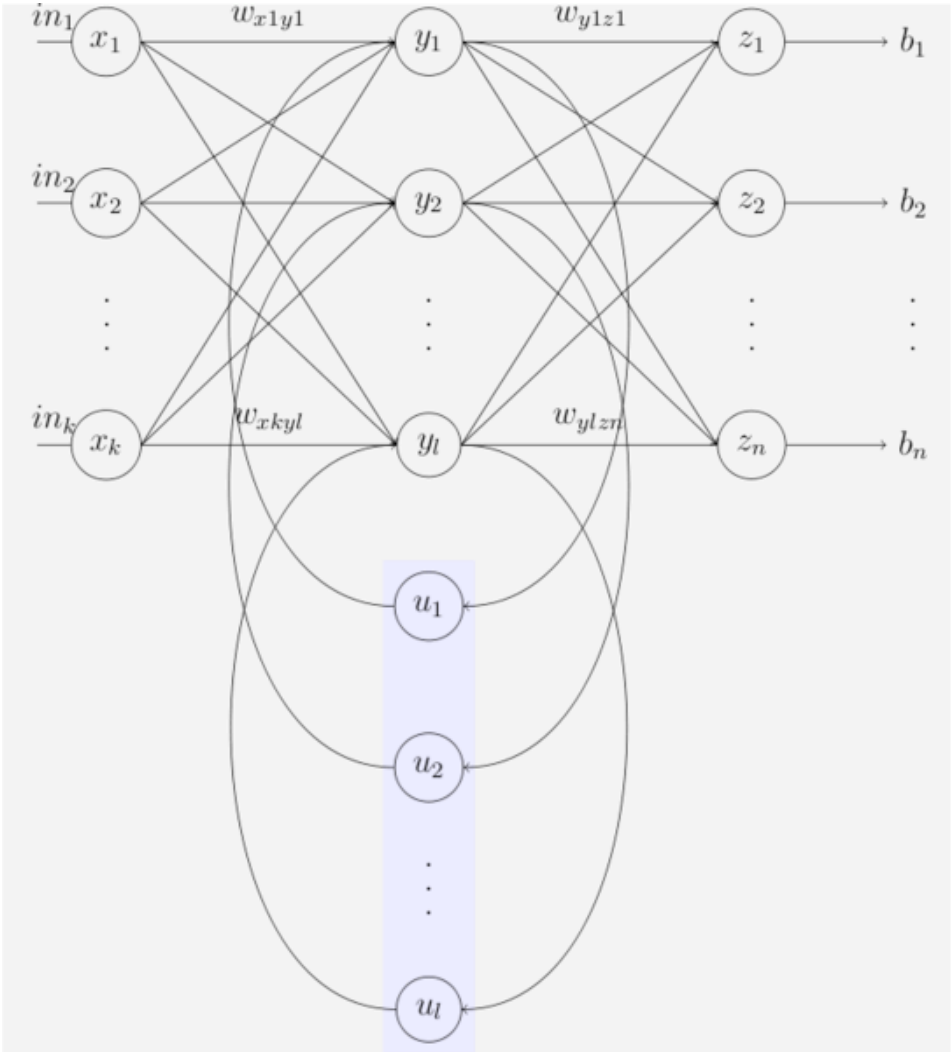
Recurrent (or Auto-Associative) Neural Networks

Elman Network

- includes a layer of "context units" connected from hidden layer with weight of 1
- at each iteration, hidden unit outputs copied into context units



Jordan Network

- similar to Elman network but:
  - context units connected to output layer
  - values in context units (c_i) are updated, not replaced by values of output units (o_i), i.e., $c_i = o_i + \gamma c_i$

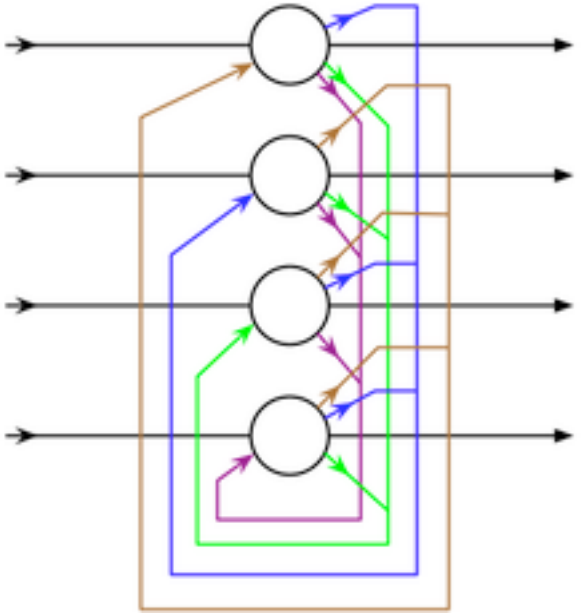
Associative Memory

- store a set of *p* patterns
- when presented with a new pattern, network responds by producing closest stored pattern “in memory”
- uses relaxation learning

Relaxation Learning

- Important questions:
  - will the network settle?
  - how long does it take to settle?
  - how are the weights acquired?
  - how much information does each unit need to convey to neighbours?

Hopfield Networks



- N interconnected neurons, potentially completely recurrent
- all neurons are both input & output; all weight are symmetric (but $w_{ii}=0$)
- activation values are binary (typically -1/+1)

- activation rule: $a_j = \begin{cases} +1 & \text{if } \Sigma_i w_{ij} a_i > \theta_j \\ -1 & \text{otherwise} \end{cases}$

  where θ_j is the threshold

Learning and Recall

- set weights to ensure learned patterns are stable
- pattern x^p stable if, when clamped (activation fixed for the given neurons), all neurons stable
- recalling a pattern is achieved by activating (possibly a subset of) neurons with the corresponding inputs and then letting the network settle

Hebbian learning

- $w_{ij} \leftarrow$ correlation of activations (+1)
- for n patterns x^p

  $w_{ij} = \sum_p x_i^p x_j^p$ if $i \neq j$, else 0
- limits of learning: for a network of N neurons, recall errors become severe above $0.15N$ "memories"
  - stored patterns become unstable
  - spurious stable states appear

Why simple Hebbian Learning Fails

- don’t know how to set weights to or from hidden units

  ← can’t generalize beyond simple Hopfield Networks