UCI

Brain-Inspired Learning Machines

Neural Models of Associative Memory and Working Memory

Emre Neftci

Department of Cognitive Sciences, UC Irvine,

November 18, 2016

Types of Declarative Memory

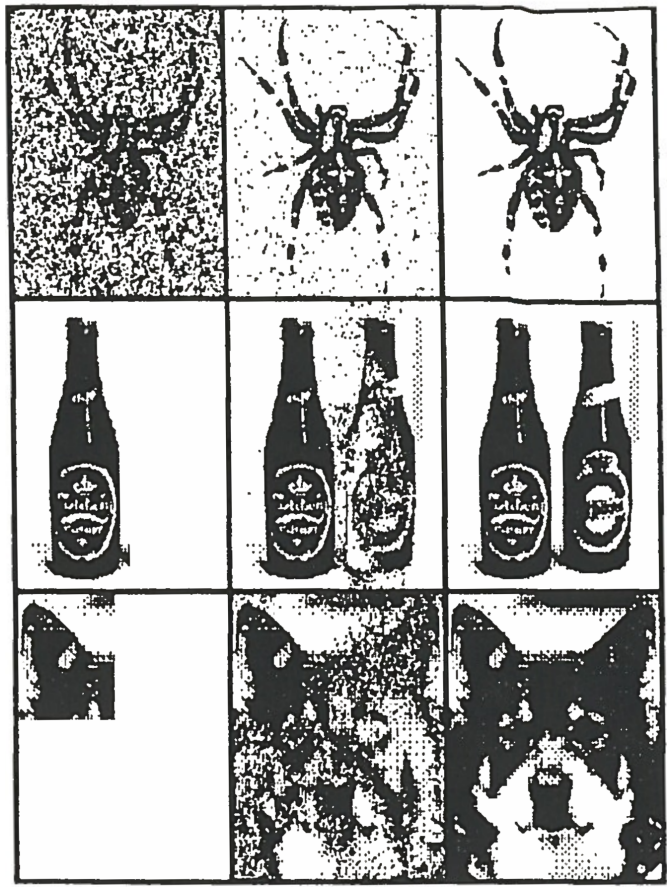
- **Working Memory** (<1m) - Prefrontal lobe, *Example:* Remembering digits of a phone number
- **Long-term Memory** (years) - Hippocampus, Inferior Temporal (IT) lobe, *Example:* Recognition Memory, Autoassociative memory

Associative Memory

Associative Memory

"In associative memory, a partial or approximate representation of a stored item is used to recall the full item"

Dayan and Abbott,, 2001



Hertz, Krogh, and Palmer,, 1991

In computer science, autoassociative memory is called Content-Addressable Memory (CAM)

Address-based Memory vs. Content-based Memory

Address-based memory *e.g.* Random Access Memory (RAM):

- High capacity
- Need to know the “address” to retrieve memory
- Recall is not robust/fault-tolerant

Content-based memory *e.g.* Cache:

- Low capacity
- Ability to recall a memory for partial cues
- Errors in recall cues can be corrected
- Hardware can be fault tolerant

Biological memory mechanisms for long-term storage are CAMs

(a) Stored Memories

moscow------russia
lima----------peru
london-----england
tokyo--------japan
edinburgh-scotland
ottawa------canada
oslo--------norway
stockholm---sweden
paris-------france

(b) Recall

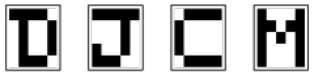
moscow---::::::::: $\Longrightarrow$ moscow------russia

::::::::::--canada $\Longrightarrow$ ottawa------canada

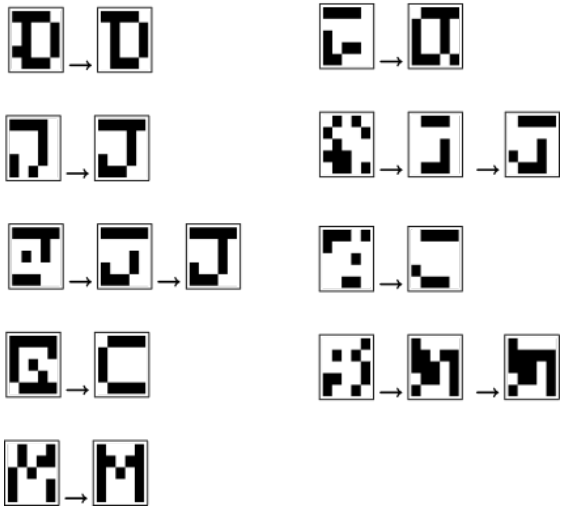
otowa-------canada $\Longrightarrow$ ottawa------canada

egindurrh-sxotland $\Longrightarrow$ edinburgh-scotland

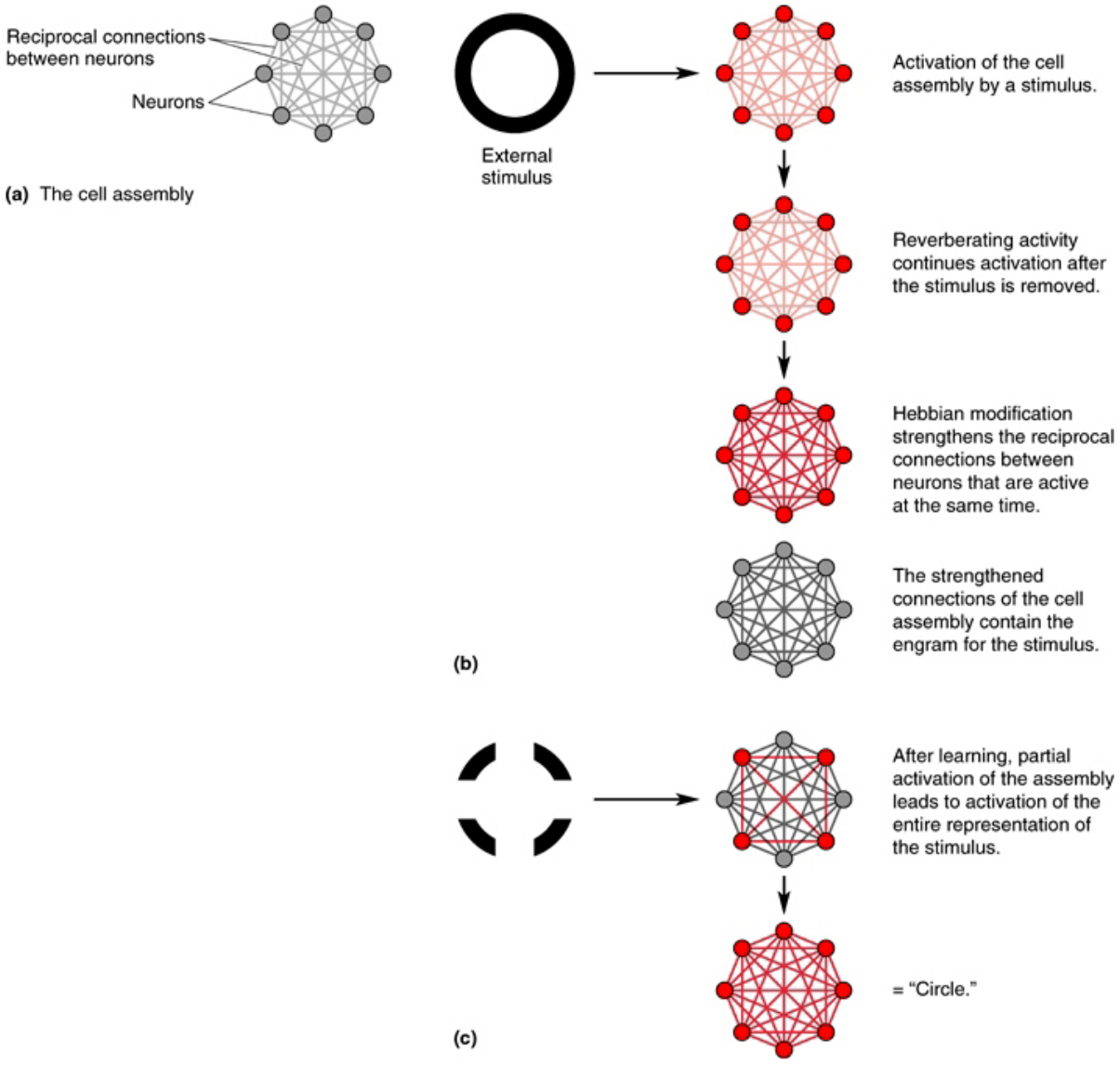
(a) Stored Memories

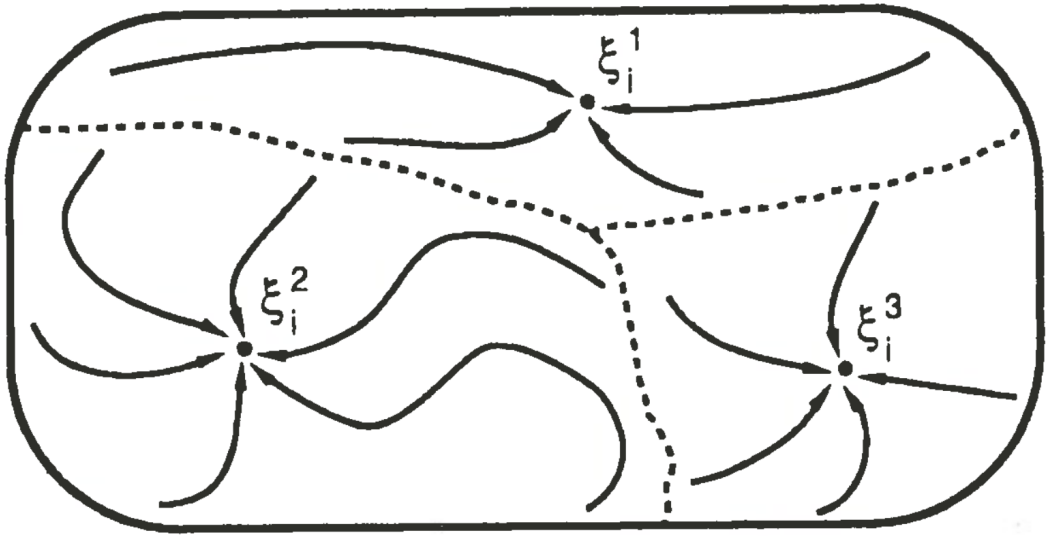


(b) Recall

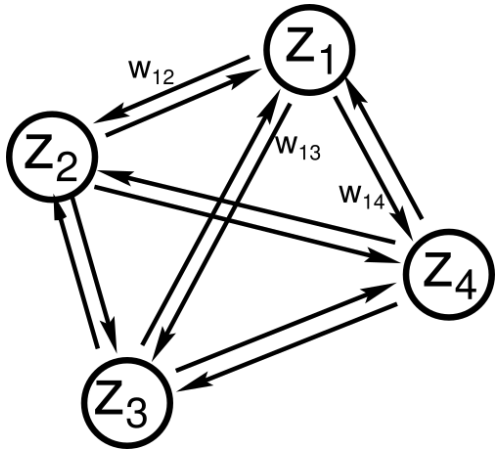


Hebb's Cell Assembly





- The rectangle is the **state space**.
- ξ_i^1, ξ_i^2, ξ_i^3 are **attactors**
- Regions bounded by dashed lines represent **basins of attraction.**



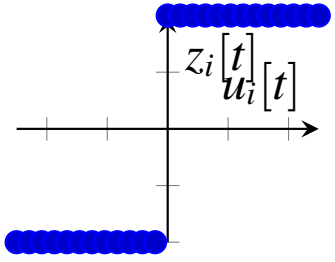
A Hopfield network is a recurrent neural net for content-addressable memory

Hopfield, *Proceedings of the national academy of sciences*, 1982

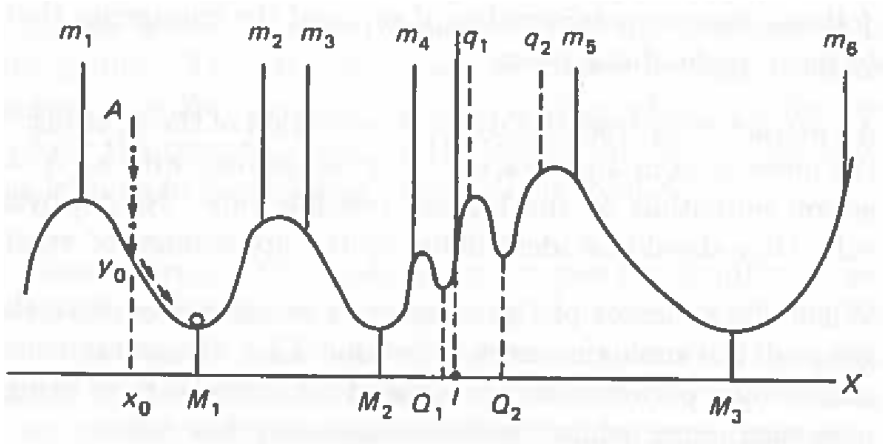
Neuron model:

$$u_i[t] = \sum_j w_{ij} z_i[t] + \theta_i$$

$$z_i[t] = \Theta(u_i[t]) = \begin{cases} 1 & \text{if } u_i[t] > 0 \\ -1 & \text{if } u_i[t] \leq 0 \end{cases}$$



Energy landscape metaphor and learning



Amit,, 1992

"Energy" of a Hopfield Network:

$$E(\mathbf{z}) = -\frac{1}{2}\sum_{i,j} w_{ij} z_i z_j - \sum_i \theta_i \, z_i$$

Learning of autoassociative memory = placing of attractors on or near regions of interest

Hopfield Learning Rule

$$w_{ij} = \frac{1}{N} \sum_{k \in data} z_i^k z_j^k$$

Example

code/ann_hopfieldnet.py

```
from npamlib import *

data = np.array([[1,1,-1,-1],[1,-1,1,-1]])

#"Train" Weight Matrix
W = np.zeros([len(data), 4, 4])
for i in range(len(data)):
    W[i, :, :] += np.outer(data[i], data[i].T)
W = np.mean(W, axis=0)
b = data.mean(axis=0)

out = FRN(N=4, b=b, W=W, rates_in = [1,1,1,-1], activation_function=hopfield, T=5)
```

Energy

$$E(\mathbf{z}) = -\frac{1}{2}\sum_{i,j} w_{ij} z_i z_j - \sum_i \theta_i \, z_i$$

z0	z1	z2	z3	Energy
-1	-1	-1	-1	-0.0
-1	-1	-1	1	-2.0
-1	-1	1	-1	-4.0
-1	-1	1	1	-6.0
-1	1	-1	-1	-4.0
-1	1	-1	1	-6.0
-1	1	1	-1	-0.0
-1	1	1	1	-2.0
1	-1	-1	-1	-6.0
1	-1	-1	1	-0.0
1	-1	1	-1	-10.0
1	-1	1	1	-4.0
1	1	-1	-1	-10.0
1	1	-1	1	-4.0
1	1	1	-1	-6.0
1	1	1	1	-0.0

Example Letters

code/ann_hopfieldnet_letter.py

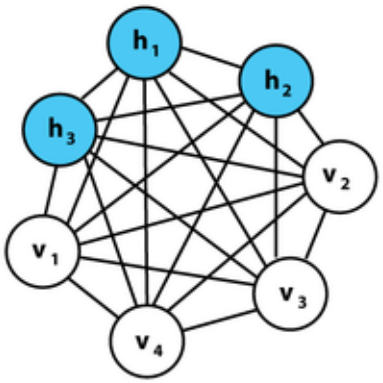
```
from npamlib import *
data, labels = data_load_letters()
#Transform into +1/-1
data = (data>.5)*2-1

#"Train" Weight Matrix
W = np.zeros([len(data), 25, 25])
for i in range(len(data)):
    W[i, :, :] += np.outer(data[i], data[i].T)
W = np.mean(W, axis=0)-np.eye(25)
b = data.mean(axis=0)

res = np.zeros([len(data),25])
data_corrupt = np.empty_like(data)

#Number of pixels to corrupt
N_corrupt = 5
for i in range(len(data)):
    d = data[i,:]
    #Flip pixel value
    d[np.random.choice(range(25),N_corrupt)] *= -1
    data_corrupt[i,:] = d
    res[i] = FRN(N=25, b=b, W=W, rates_in = data[i], activation_function=hopfield, T=10)
```

Example Letters

code/ann_hopfieldnet_letter.py

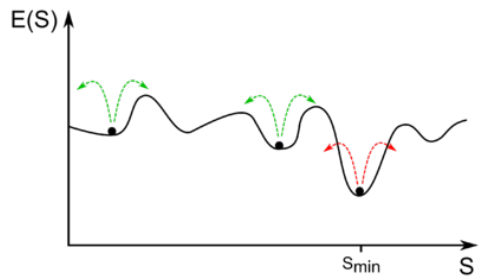
```
from npamlib import *
data, labels = data_load_letters()
#Transform into +1/-1
data = (data>.5)*2-1

#"Train" Weight Matrix
W = np.zeros([len(data), 25, 25])
for i in range(len(data)):
    W[i, :, :] += np.outer(data[i], data[i].T)
W = np.mean(W, axis=0)-np.eye(25)
b = data.mean(axis=0)

res = np.zeros([len(data),25])
data_corrupt = np.empty_like(data)

#Number of pixels to corrupt
N_corrupt = 5
for i in range(len(data)):
    d = data[i,:]
    #Flip pixel value
    d[np.random.choice(range(25),N_corrupt)] *= -1
    data_corrupt[i,:] = d
    res[i] = FRN(N=25, b=b, W=W, rates_in = data[i], activation_function=hopfield, T=10)
```

No labels were used for learning: Unsupervised Learning

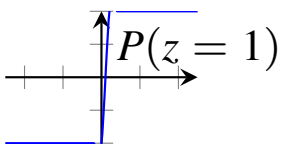


$$p^{BM}(\mathbf{z}) = \frac{1}{Z}\exp\big(-E(\mathbf{z})\big), \quad E(\mathbf{z}) = -\frac{1}{2}\sum_{i,j} w_{ij} z_i z_j - \sum_i \theta_i \, z_i$$

Intuition behind the BM: add stochasticity to a Hopfield network to escape local minima

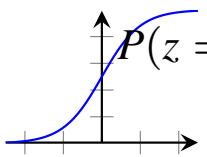


Deterministic unit (Hopfield Network)



$$\Delta w_{ij} \propto \langle z_i z_j \rangle_{data}$$

Stochastic unit (Boltzmann Machine)



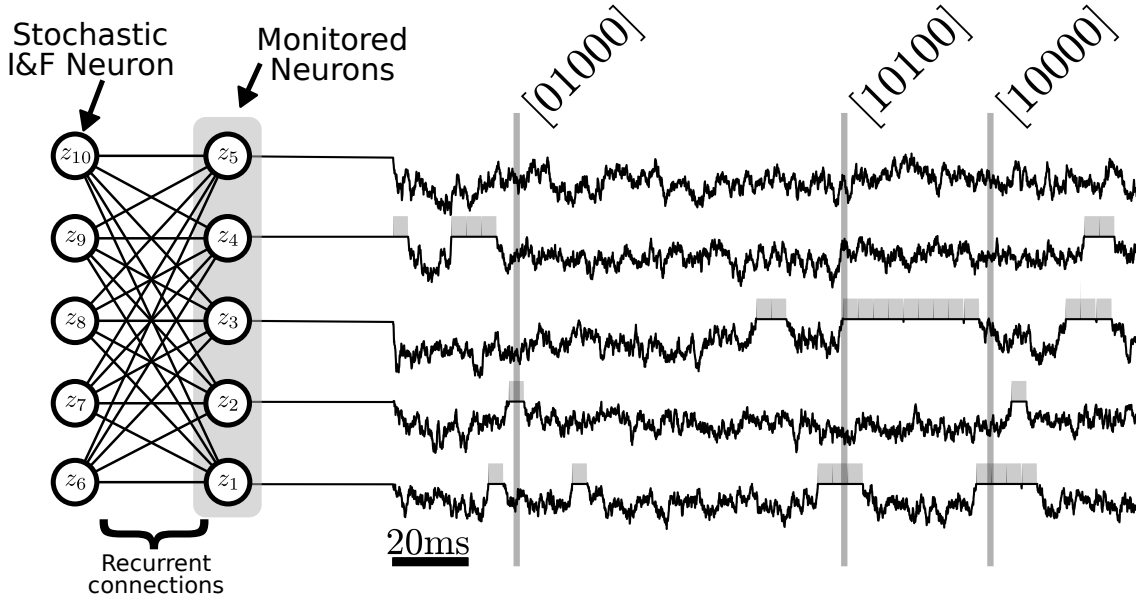
$$\Delta w_{ij} \propto \langle z_i z_j \rangle_{data} - \underbrace{\langle z_i z_j \rangle_{reconstruction}}_{\text{difficult to evaluate}}$$

Hinton and Sejnowski, 1986

The additional term is often estimated using Markov Chain Monte Carlo (MCMC) sampling

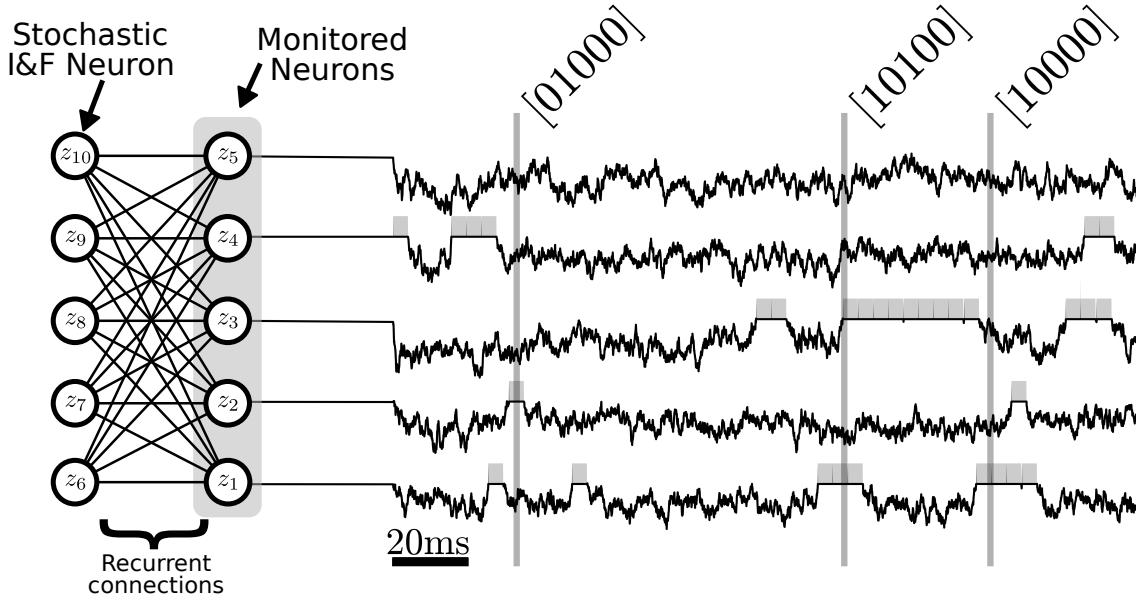
Neural sampling and learning with noisy Integrate & Fire (I&F) neurons

Neuron state associated with a binary random variable $z_k = 0, 1$

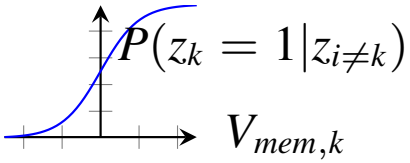


Neural sampling and learning with noisy I&F neurons

Neuron state associated with a binary random variable $z_k = 0, 1$



- Network states are Markov Chain Monte Carlo samples of the Boltzmann distribution if:



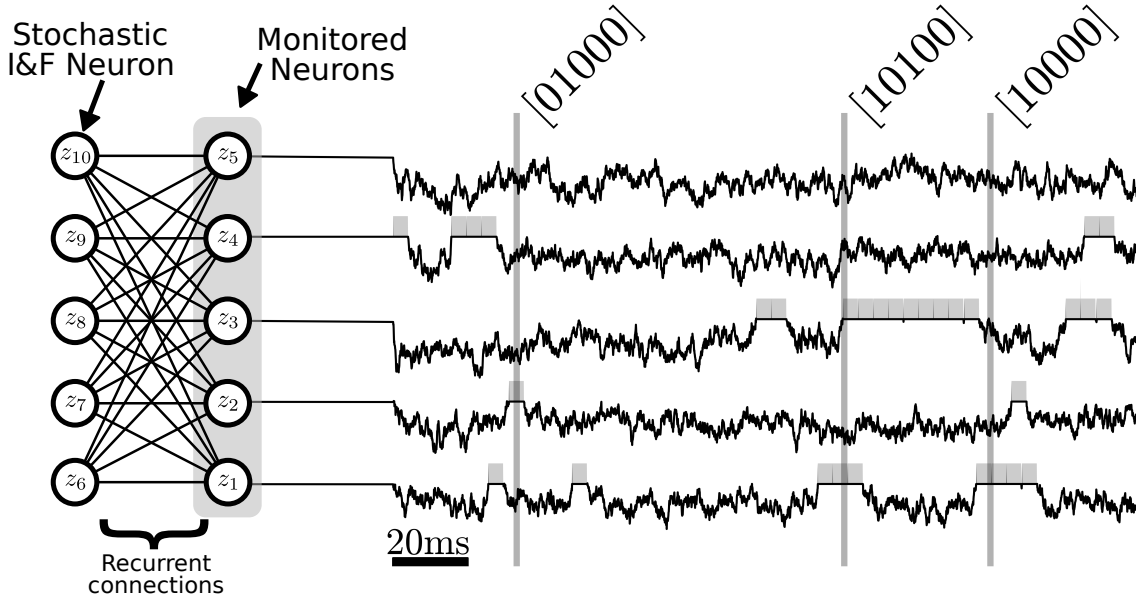
$$V_{mem,k}(t) = \sum_i W_{ki} z_i(t) + b_k$$

Linear sum of inputs

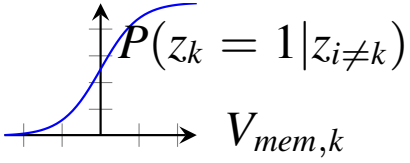
Buesing, Bill, Nessler, and Maass, *PLoS Computational Biology*, 2011

Neural sampling and learning with noisy I&F neurons

Neuron state associated with a binary random variable $z_k = 0, 1$



- Network states are Markov Chain Monte Carlo samples of the Boltzmann distribution if:



$$V_{mem,k}(t) = \sum_i W_{ki} z_i(t) + b_k$$

Linear sum of inputs

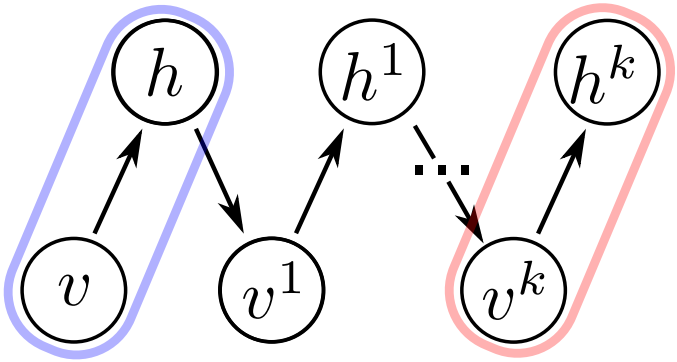
Buesing, Bill, Nessler, and Maass, *PLoS Computational Biology*, 2011

- Biophysical neuron models can meet these conditions
- Learning with STDP-based *Event-driven Contrastive Divergence*

Neftci, Posch, and Chicca, *Computational Intelligence*, 2014

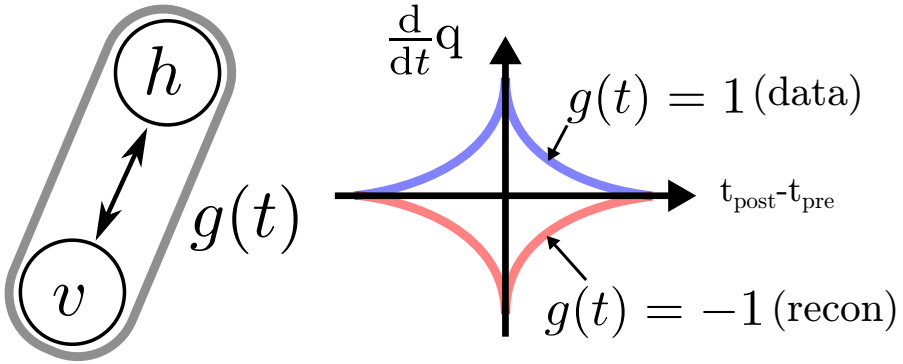
Event-driven Contrastive Divergence

Gibbs Sampling +
Standard CD



$$\Delta w \propto \langle vh \rangle_{\text{data}} - \langle v^k h^k \rangle_{\text{recon}}$$

Neural Sampling +
Event-driven CD

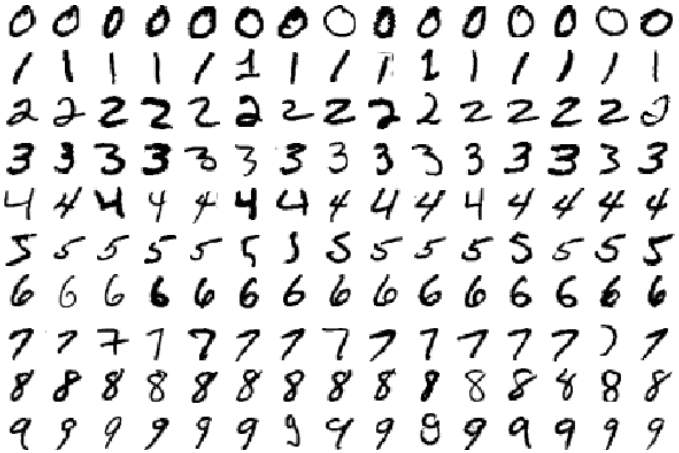


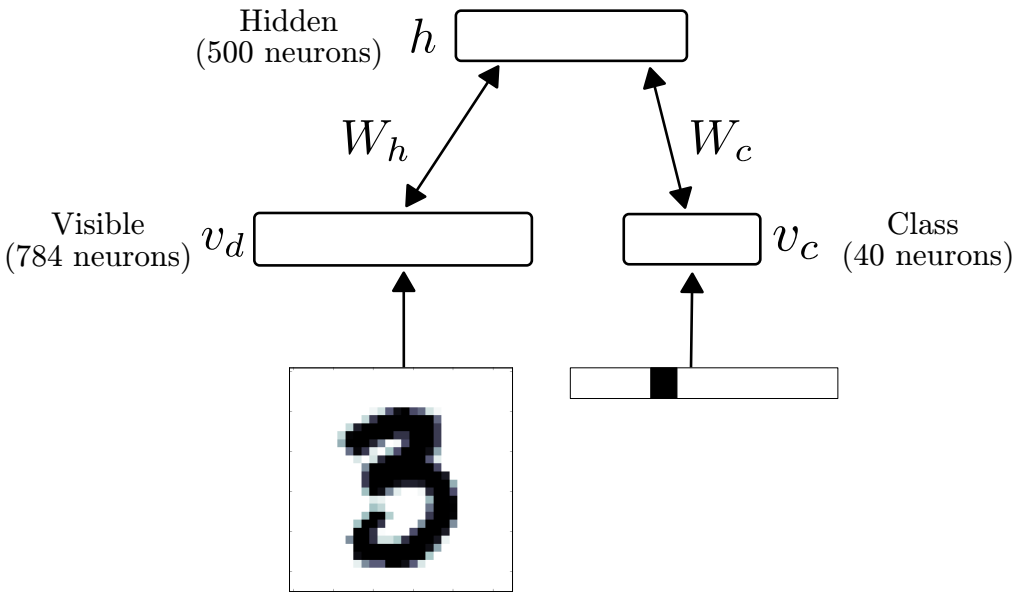
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{q} = g(t)\mathrm{STDP}(v(t), h(t))$$

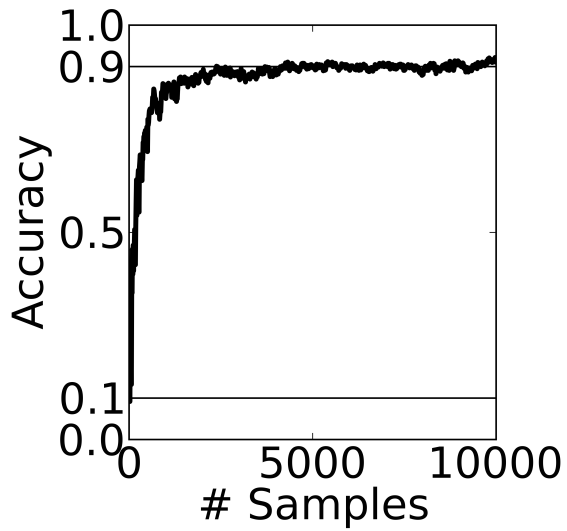
In average, eCD behaves like standard CD

Neftci_etal14 Neftci_etal14Neftci_etal14 Neftci_etal14

Learning a model of MNIST hand-written digits (software simulations)



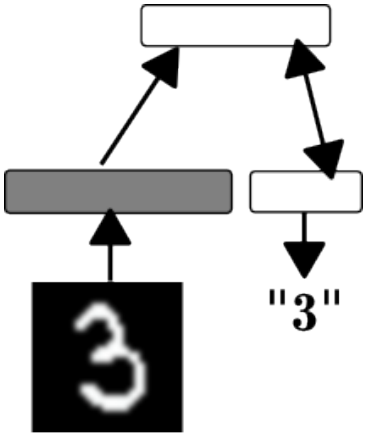




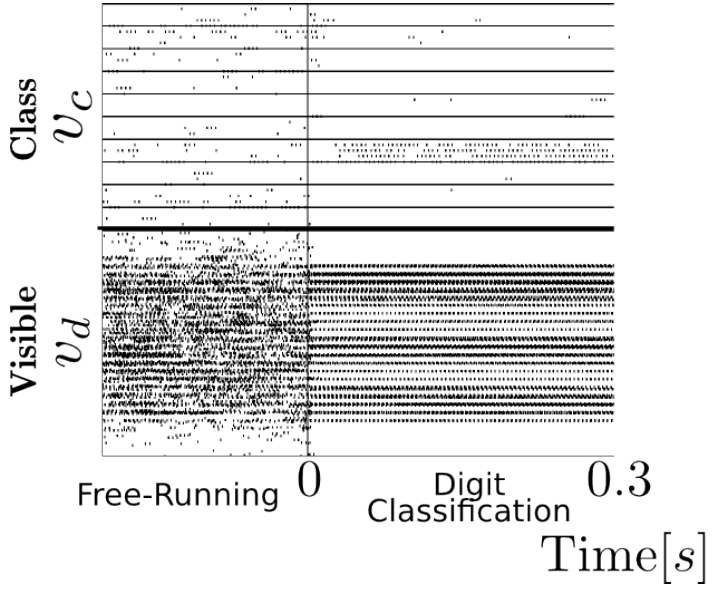
Recognition Accuracy: 91.9%, (*vs.* standard CD: 93.6%)

Neftci, Posch, and Chicca, 2014

The trained network performs Bayesian inference

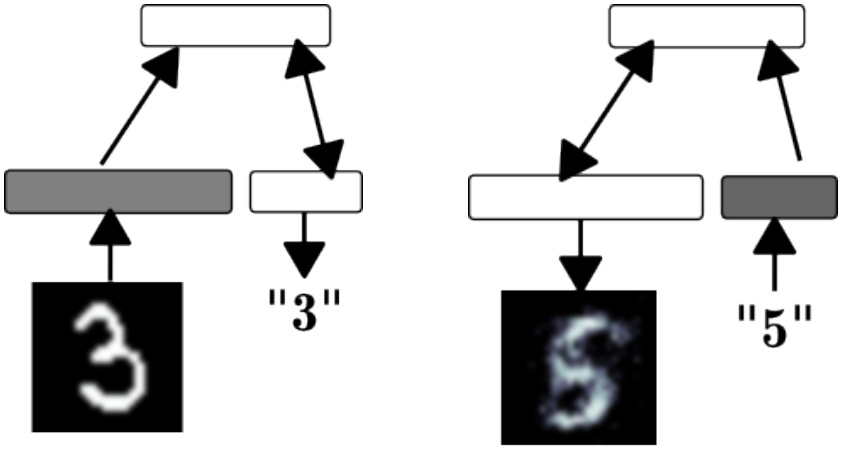


Digit
Classification



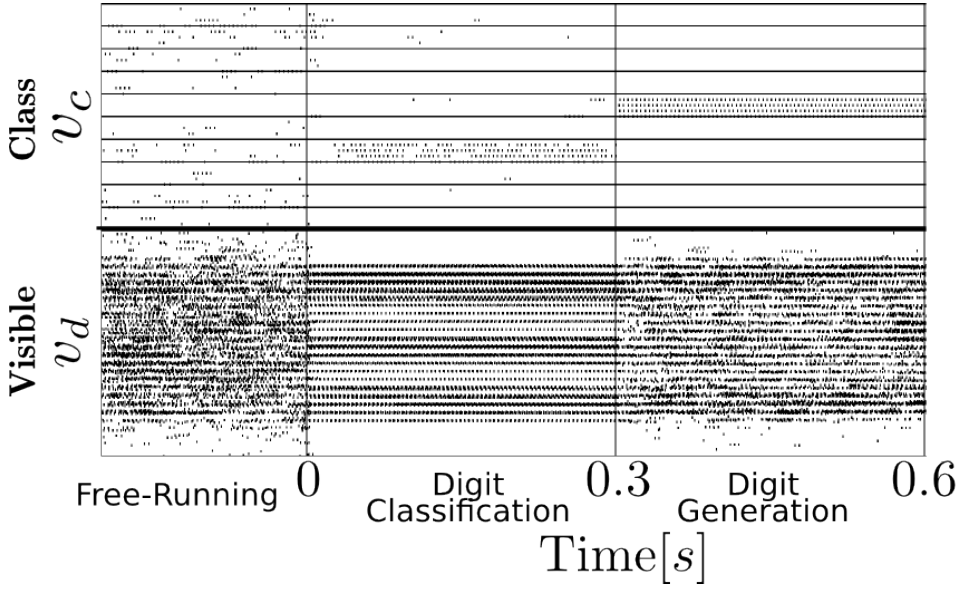
Neftci, Posch, and Chicca, 2014

The trained network performs Bayesian inference



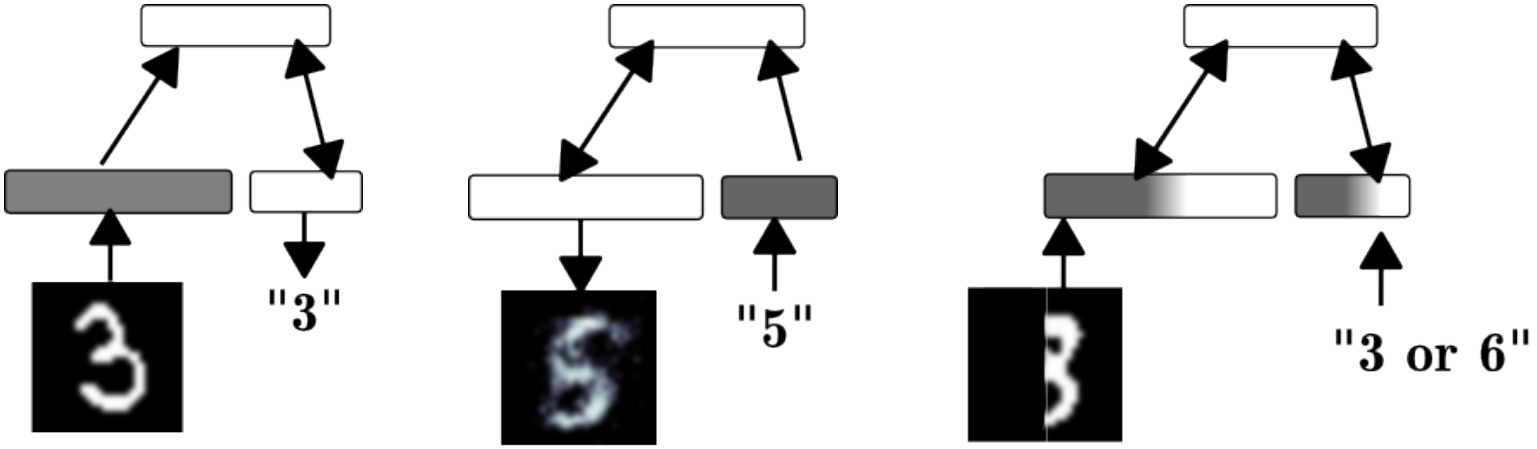
Digit Classification

Digit Generation



Neftci, Posch, and Chicca, 2014

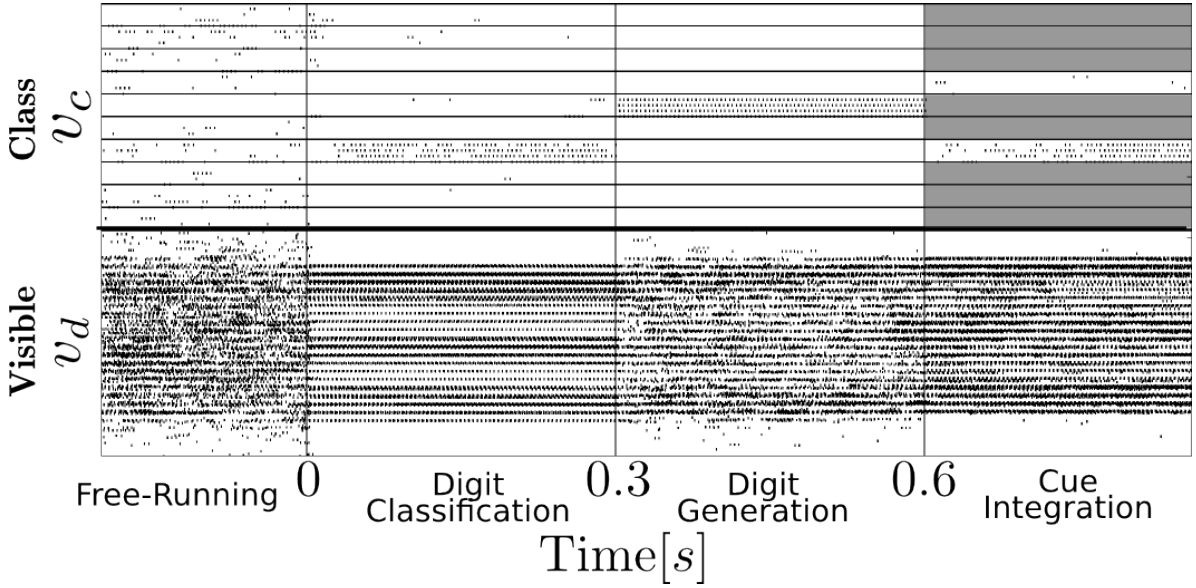
The trained network performs Bayesian inference



Digit Classification

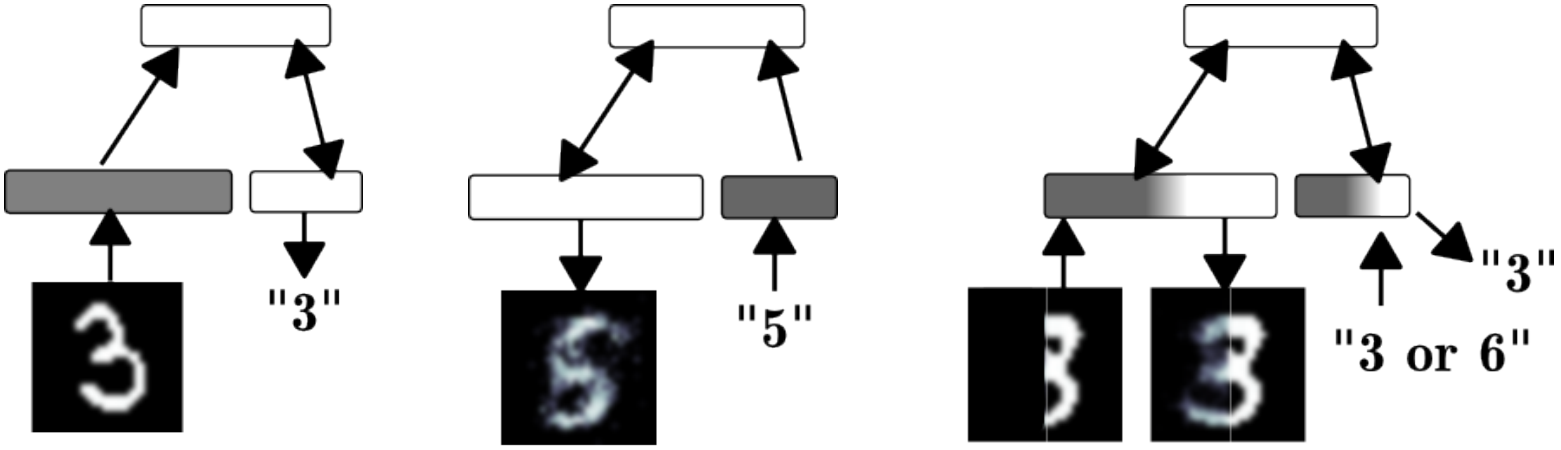
Digit Generation

Cue Integration



Neftci, Posch, and Chicca, 2014

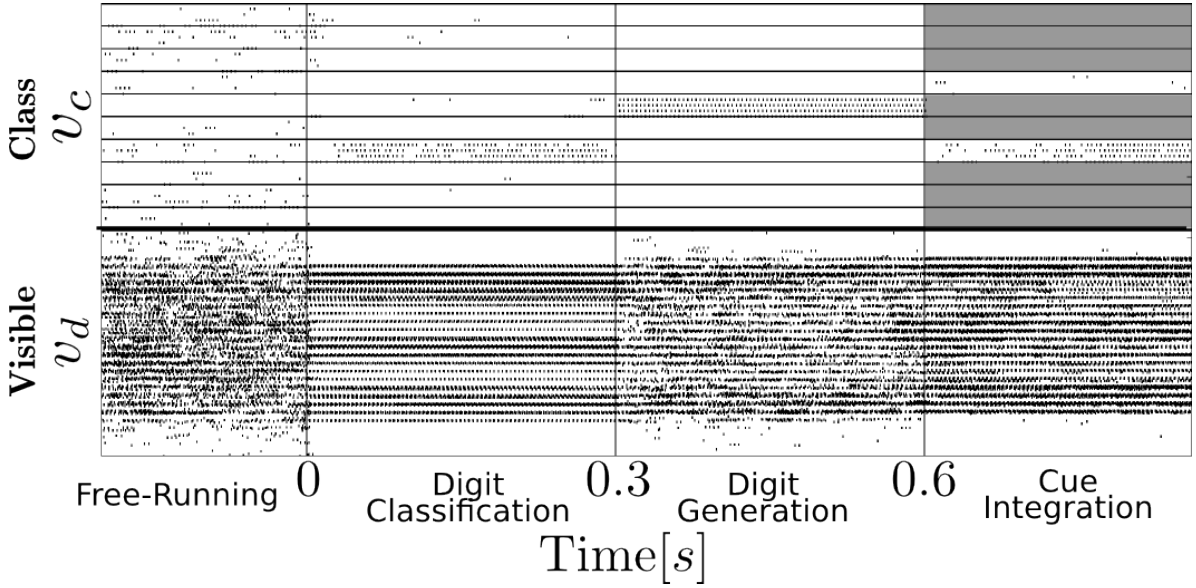
The trained network performs Bayesian inference



Digit Classification

Digit Generation

Cue Integration

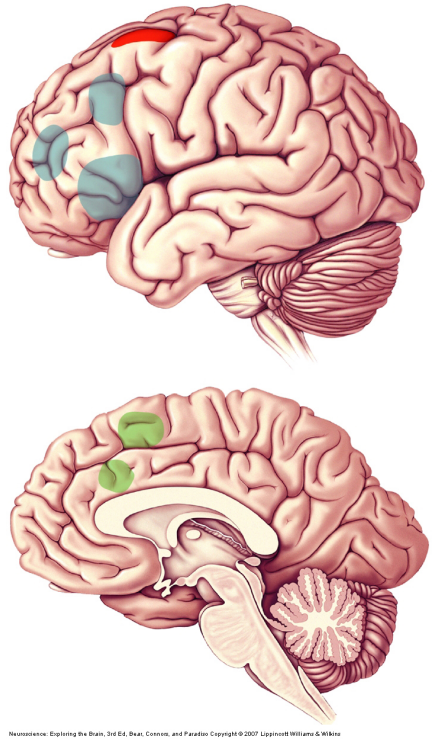


Neftci, Posch, and Chicca, 2014

Working Memory

Working Memory:

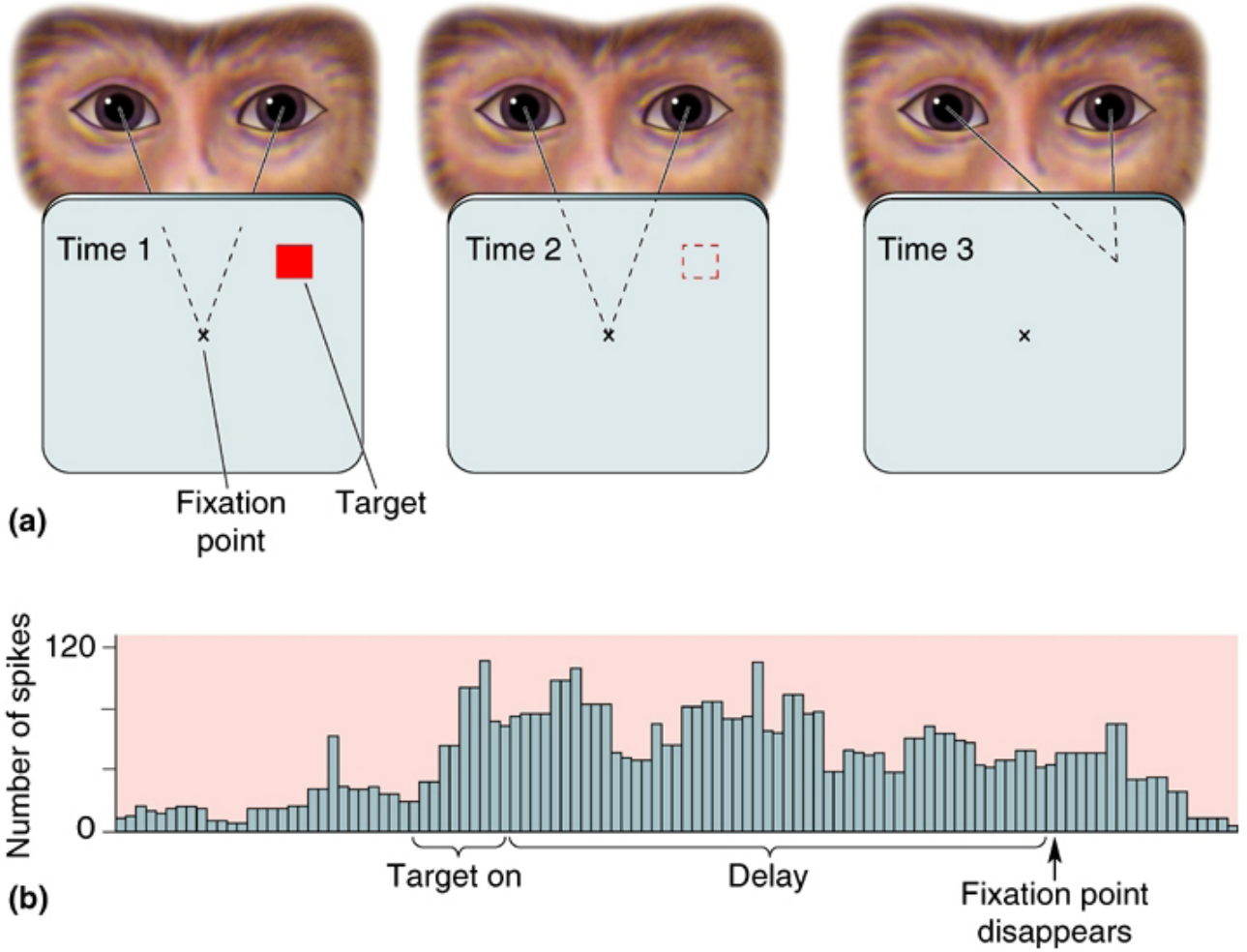
- A type of short-term memory
- Limited in capacity
- Task- and sensory modality-dependent
- Necessary for cognitive control

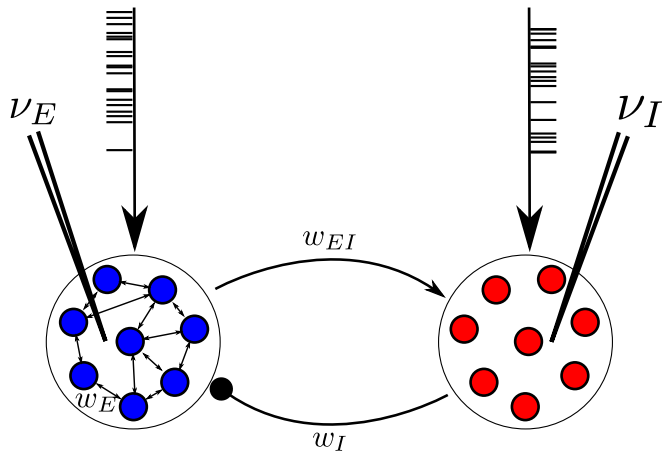


Human brain areas for working memory of face identity and location

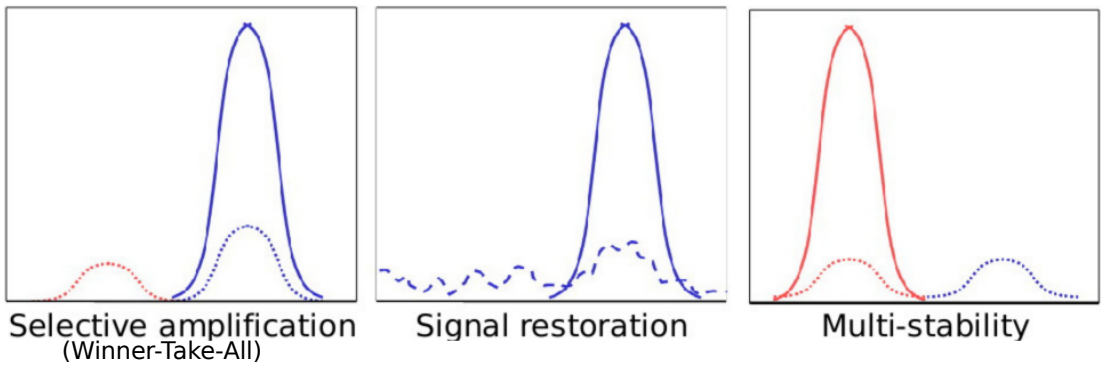
Working Memory is a "Mental Sketch Pad"

Neural Correlates of Working Memory

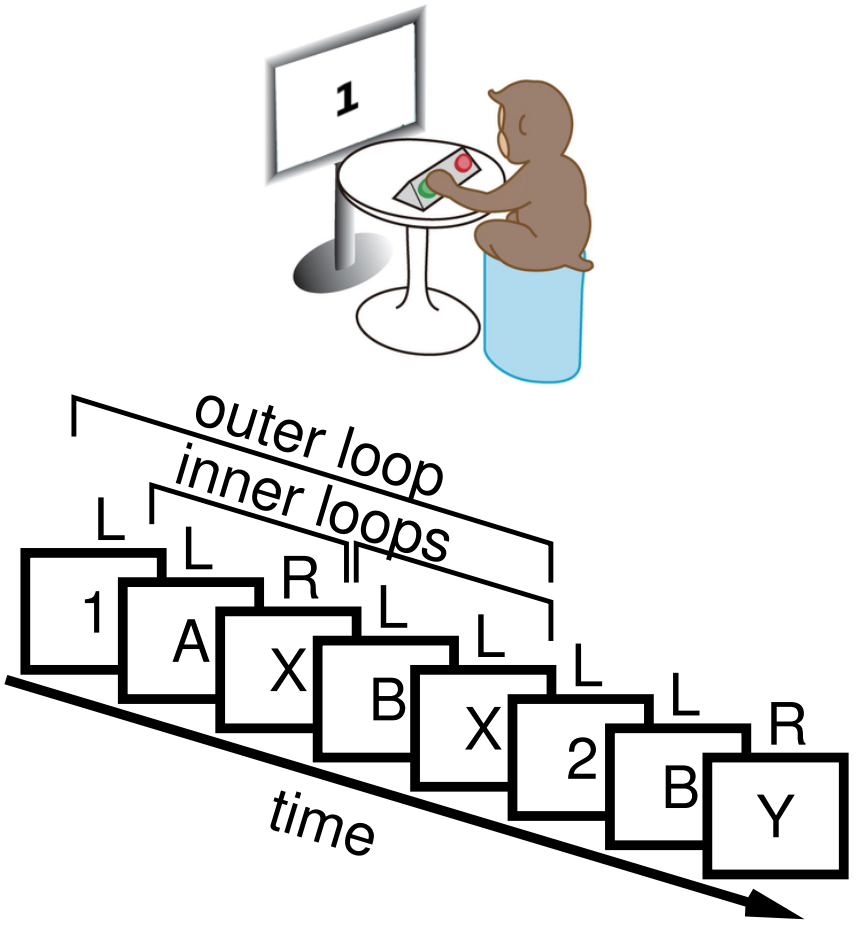



ν_E
ν_I
w_{EI}
w_E
w_I

Winner-Take-All (WTA) Properties



Douglas and Martin, *Annual Review of Neuroscience*, 2004



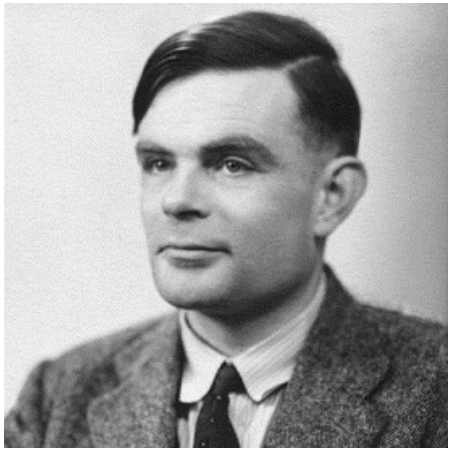
O'Reilly and Frank, 2006

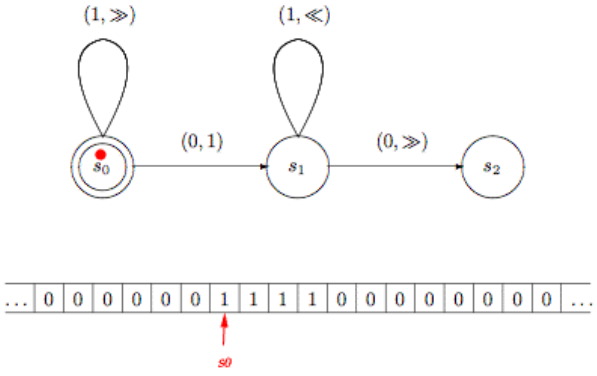
Finite State Machines (FSM)

State Machines are the Main Components of Turing Machines

A Turing machine is a theoretical computing machine invented by Alan Turing (1937) to serve as an idealized model for mathematical calculation.

Wolfram Mathworld





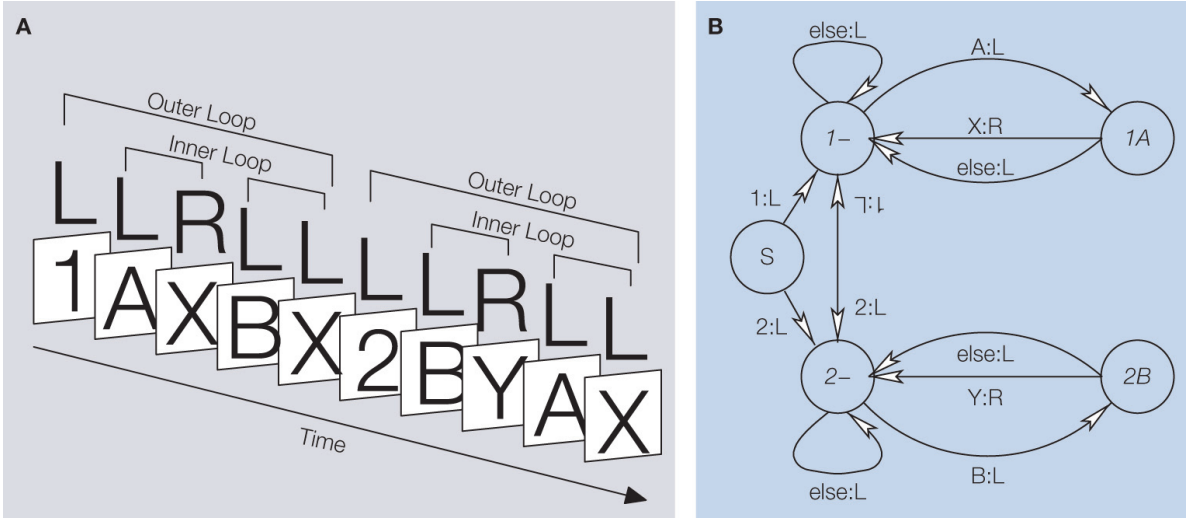
`http://www.google.com/doodles/alan-turings-100th-birthday`

Any real-world computation can be translated into an equivalent computation involving a Turing machine

The 12-AX Cognitive task

12AX Task

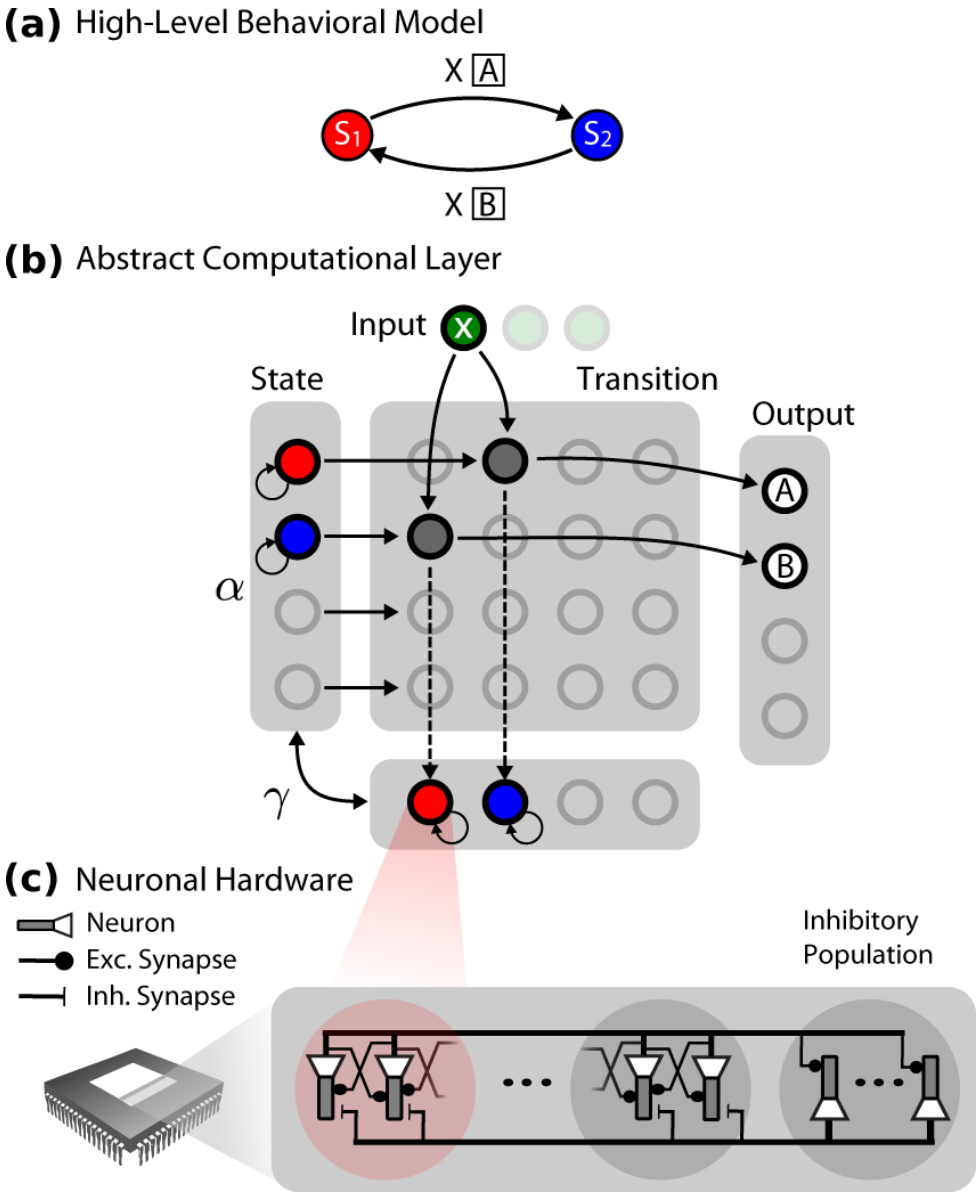
State machine solving 12AX



O'Reilly and Frank, Dayan, 2006, 2008

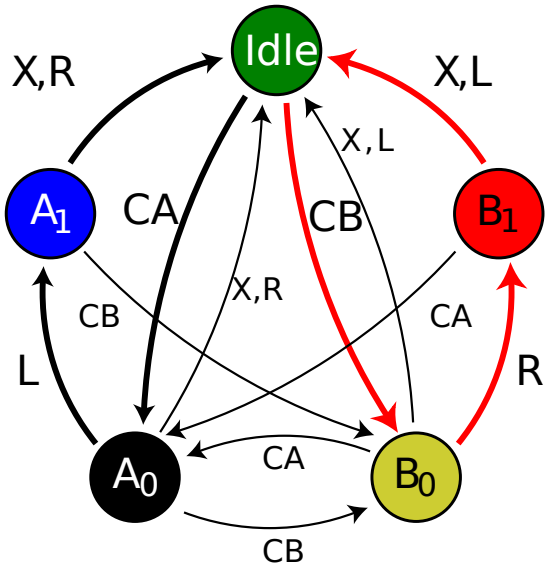
Can neurons implement state machines?

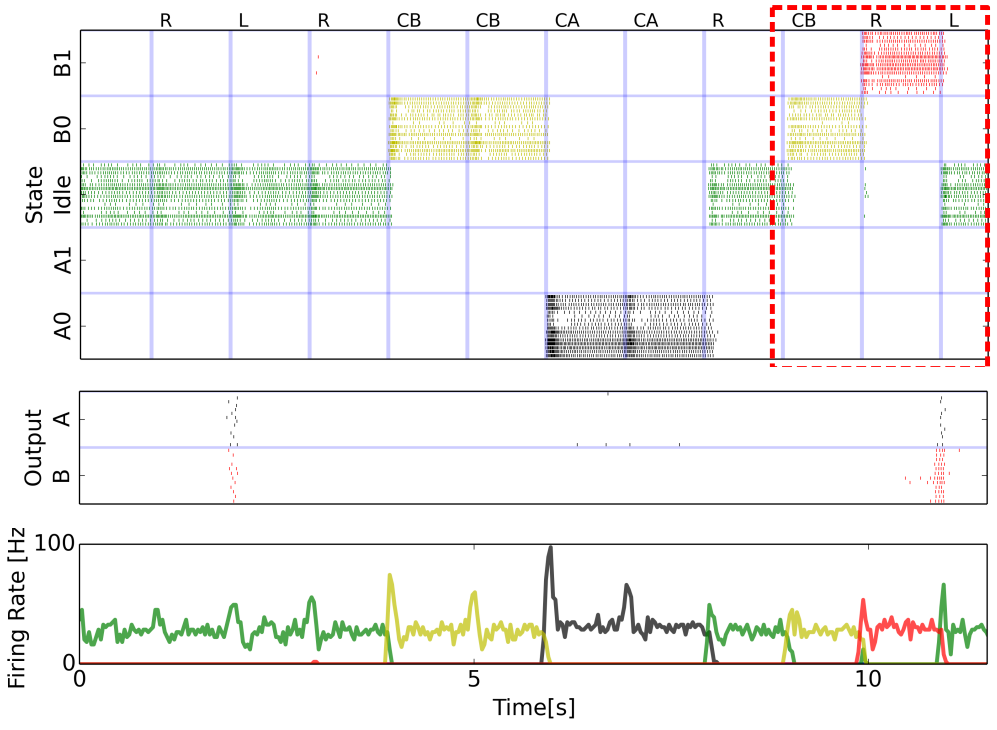
Synthesizing State-Dependent Behavior in WTA



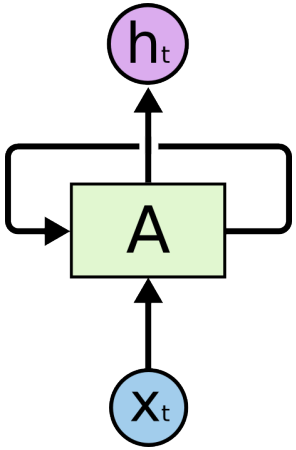
Neftci, Binas, Rutishauser, Chicca, Indiveri, and Douglas, *PNAS*, 2013

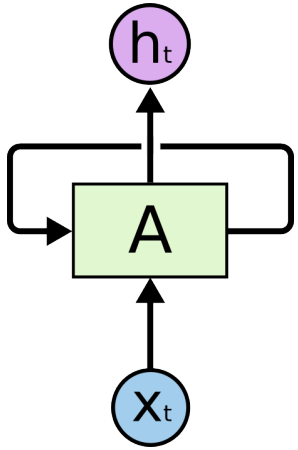
Example: Synthesis of 12-AX task state machine

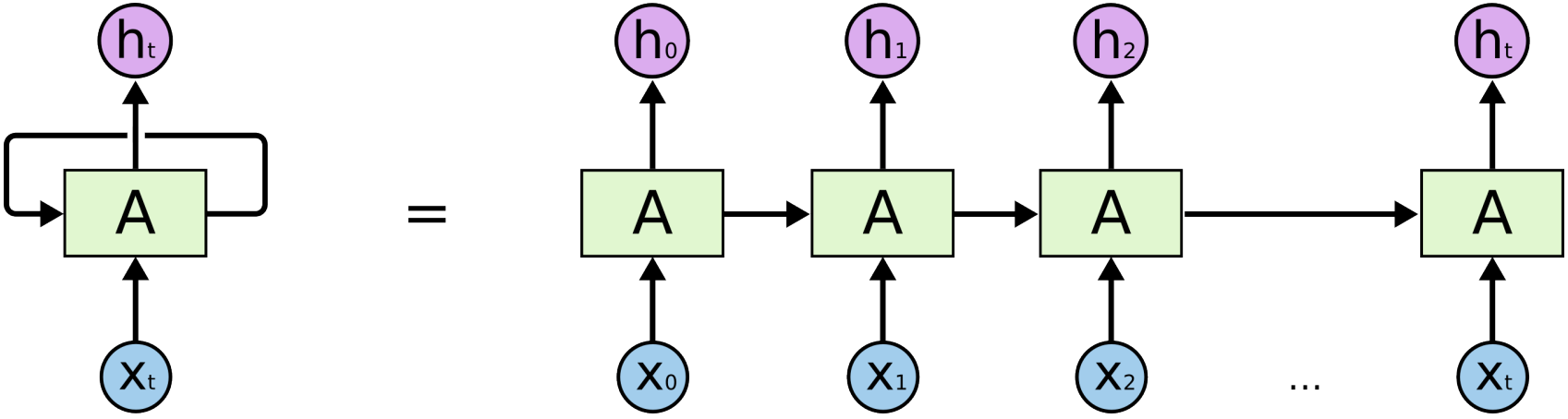


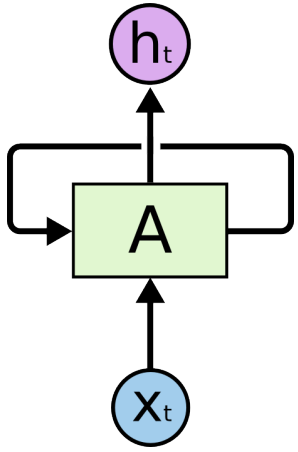


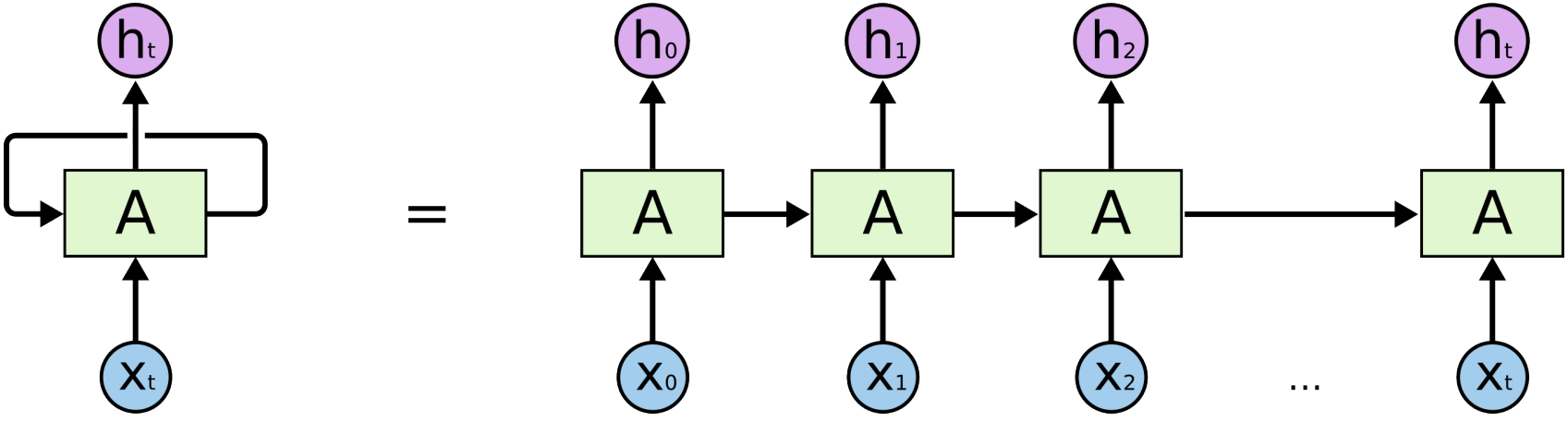
Neftci, Binas, Rutishauser, Chicca, Indiveri, and Douglas, *PNAS*, 2013


h_t
A
x_t


h_t
A
x_t


h_t
A
x_t
=
h_0
h_1
h_2
h_t
A
A
A
A
x_0
x_1
x_2
...
x_t





Learning with Back-Propagation on the “unrolled” network

Final Project and Presentations

- Topic in brain-inspired machine learning using what you have learned in class
- No class Nov. 25
- Dec 2: 10+5 min presentation
- Dec 7: Project report, using NIPS 2015 template `https://nips.cc/Conferences/2015/PaperInformation/StyleFiles`
- Discuss topic with instructor
- Groups of two OK, but delineate contributions