Neuromorphic Computing with Reservoir Neural Networks on Memristive Hardware

Aaron Stockdill

September 2016

Neuromorphic Computing with Reservoir Neural Networks on Memristive Hardware

Aaron Stockdill
September 2016

Neuromorphic Computing

Brain **Shaped**

Trying to build an artificial brain

Software: Artificial Neural Networks

Hardware: this project!

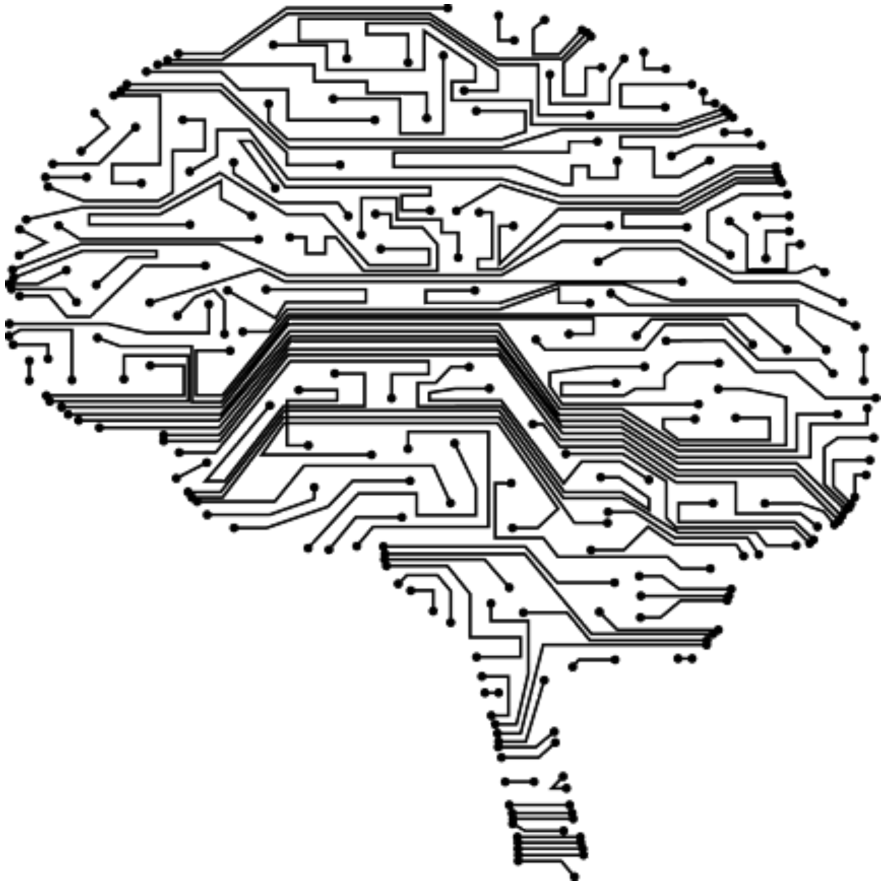


Image Source: Middleton Lab, http://middleton-lab.com/

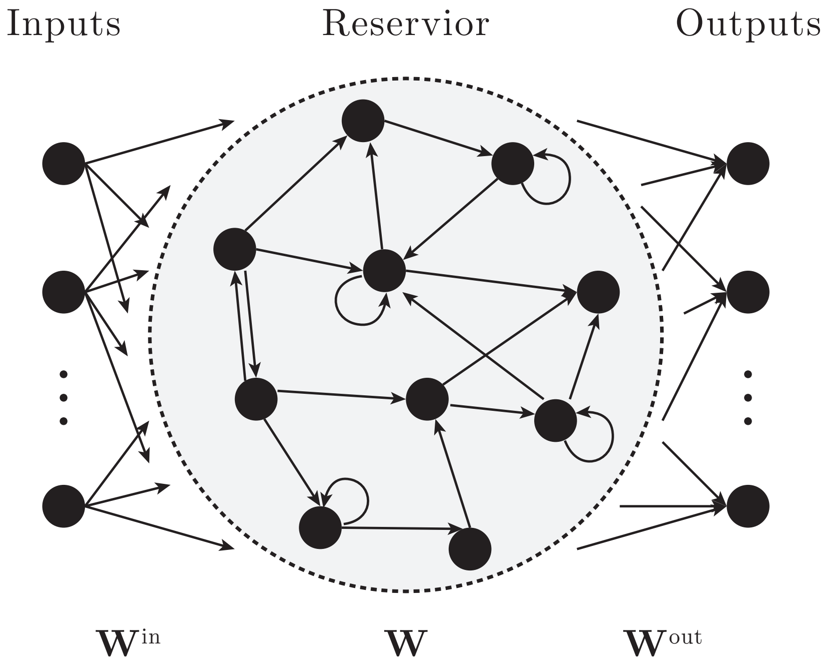
Reservoir Neural Networks

Recurrent neural network

Much easier to train – single readout layer

Echo State Network (ESN), Jaeger, 2001

Liquid State Machine (LSM), Maass, 2002

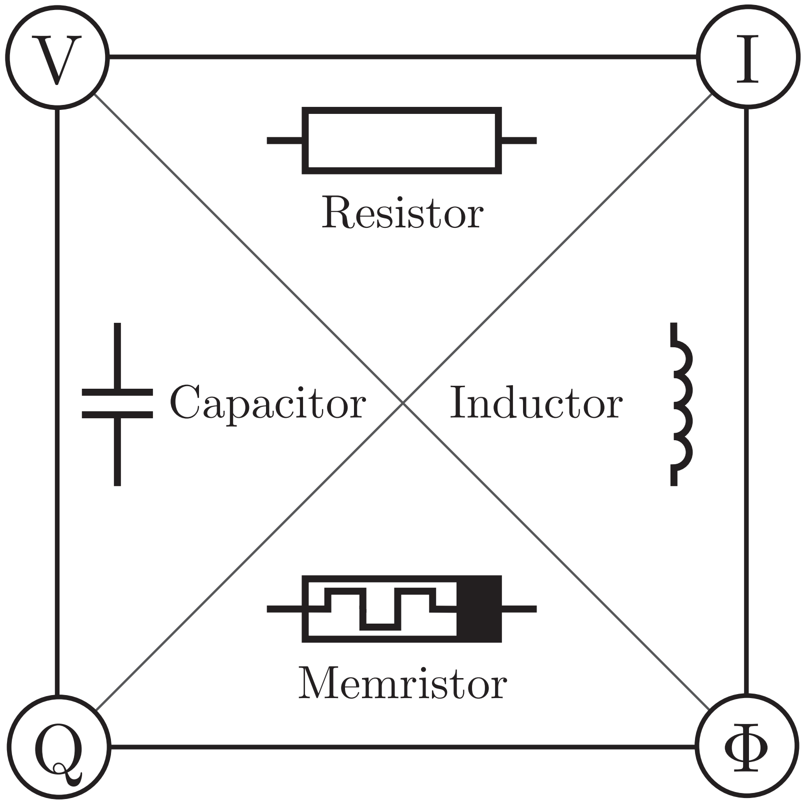


Memristors & Atomic Switches

Memristors: Leon Chua, 1989

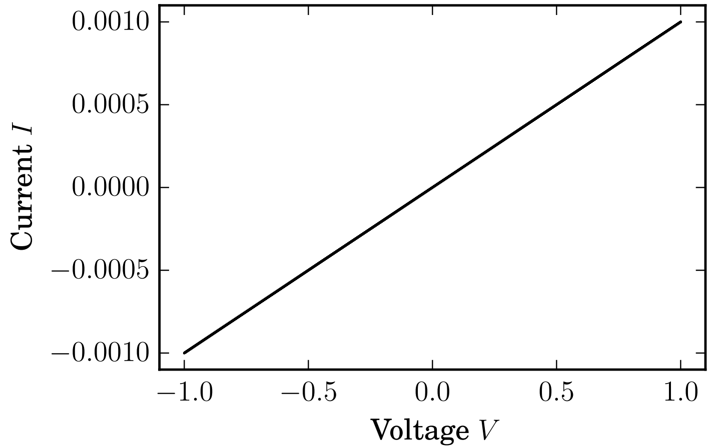
Resistance is based on past voltage/current

Atomic switches are very similar, but instead of a gradual change, it's high/low.

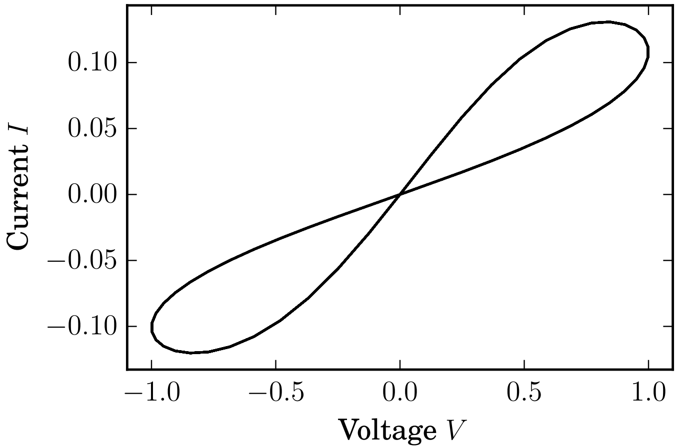


Memristors & Atomic Switches

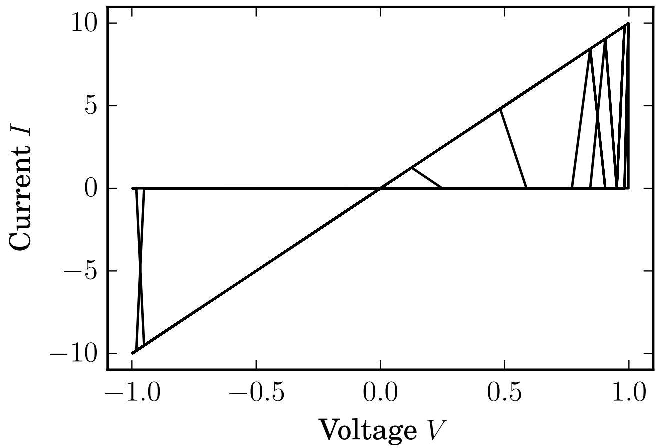
Resistor



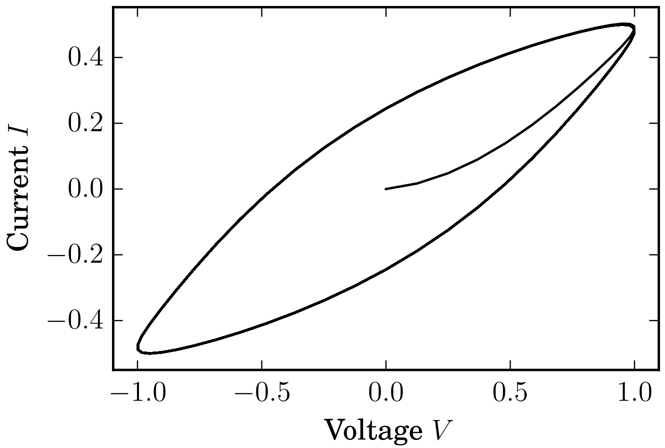
Memristor



Atomic Switch



Echo Neuron



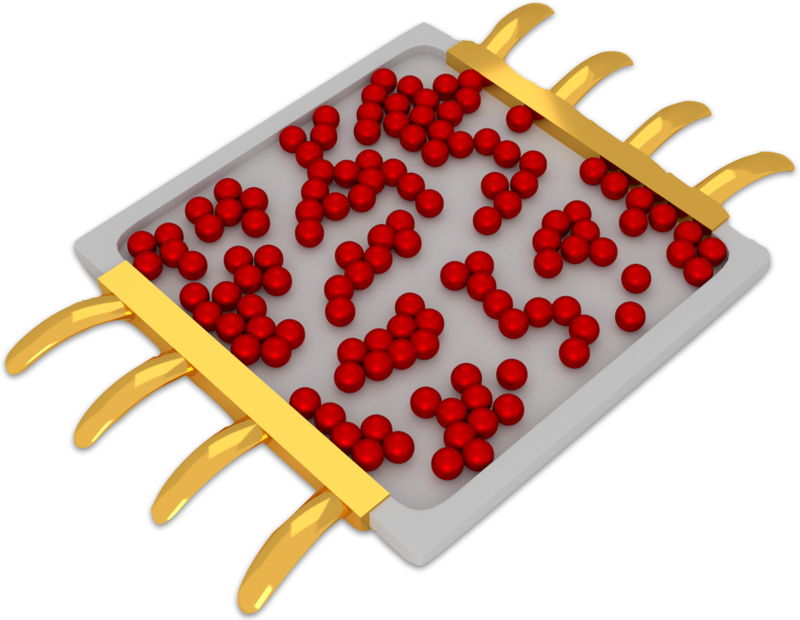
Motivation & Goals

Build a simulation of neuromorphic hardware

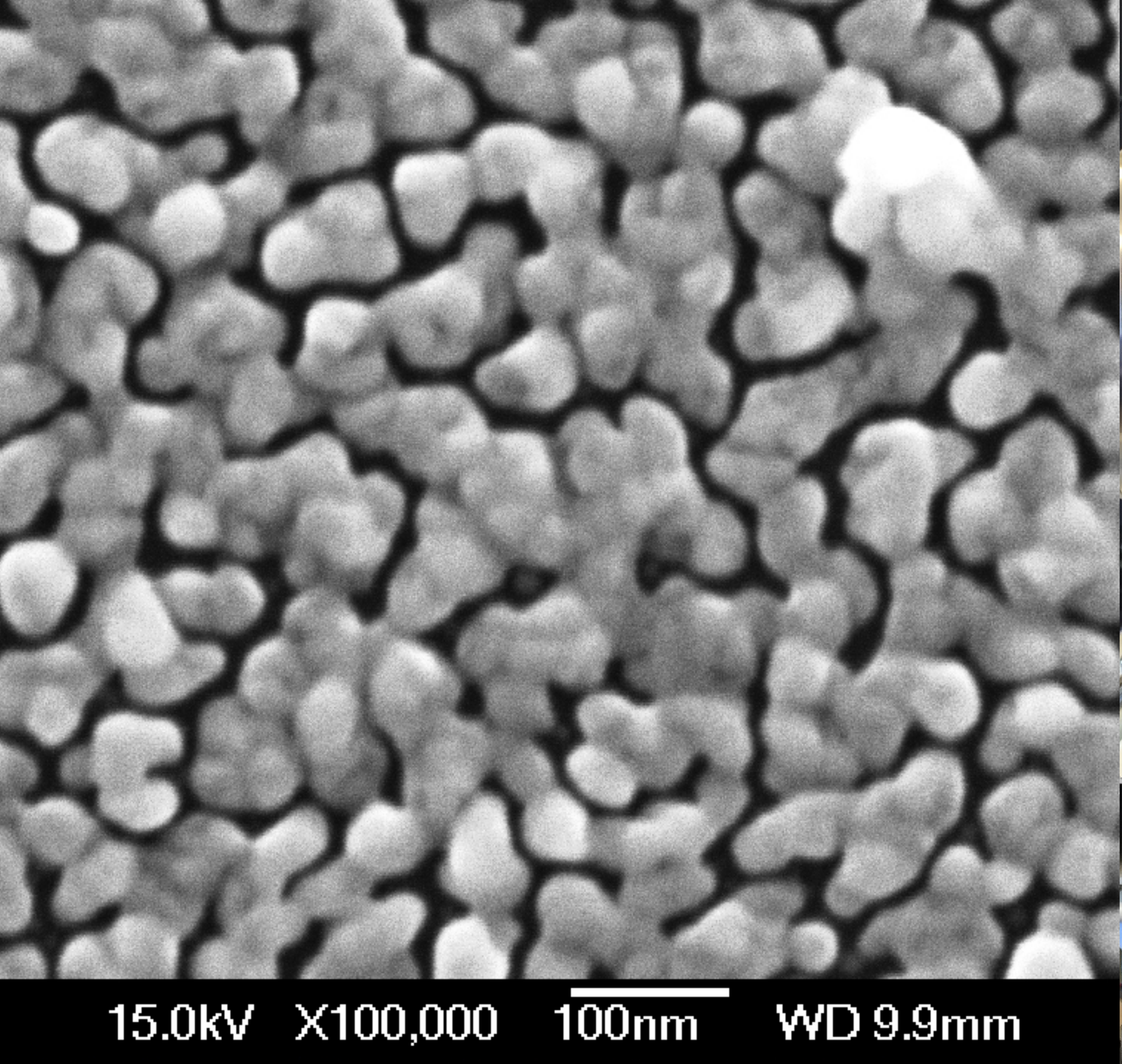
See how atomic switches compare to memristors

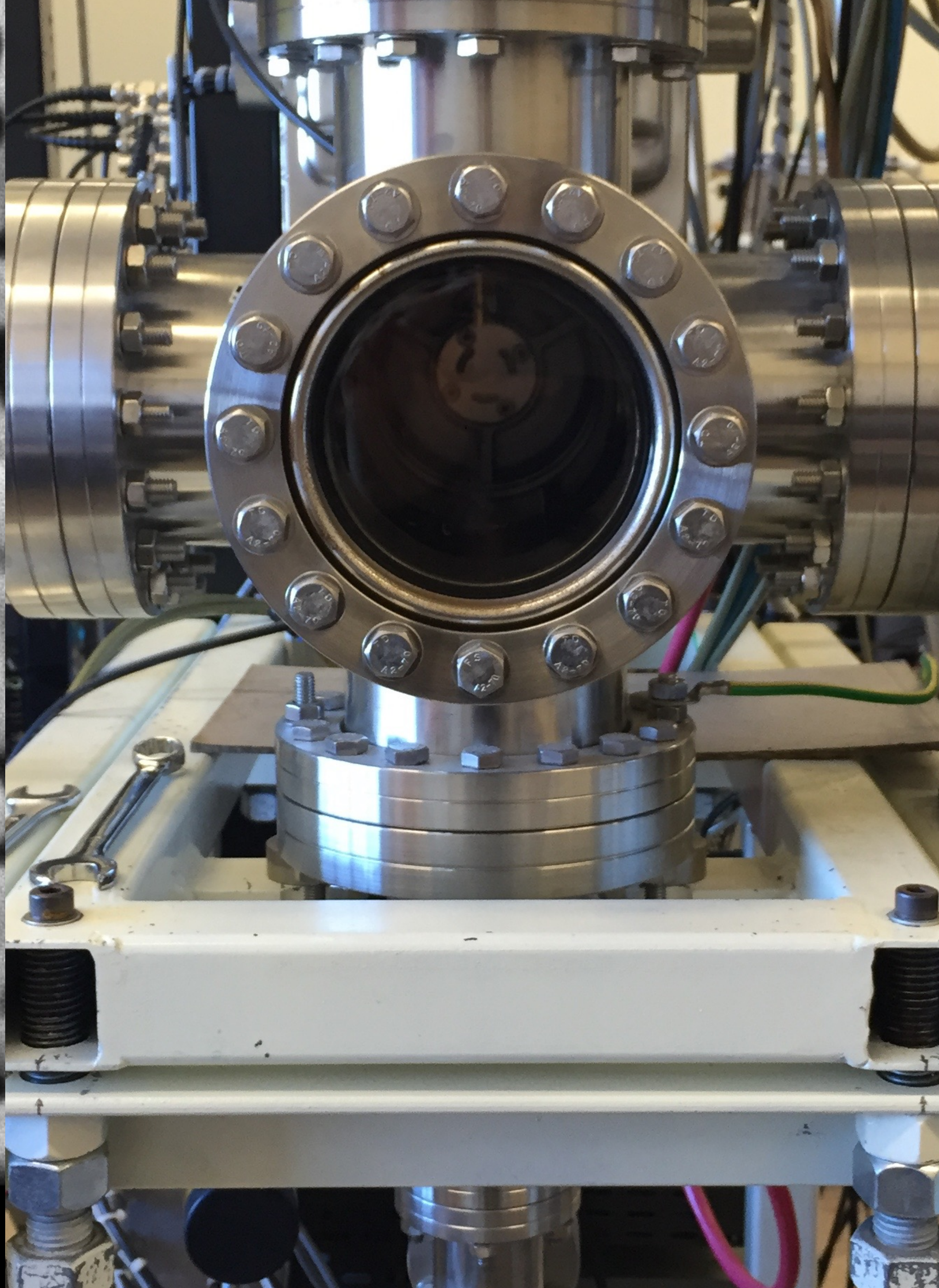
Determine what kinds of problems this hardware can solve

Build the most amazing machine learning tool ever



Simulating Novel Hardware


15.0kV X100,000 100nm WD 9.9mm



Fostner & Brown

Focussed only on atomic switches

Matlab

Workflow:

- Deposit particles
- Find groups
- Calculate connections between groups.

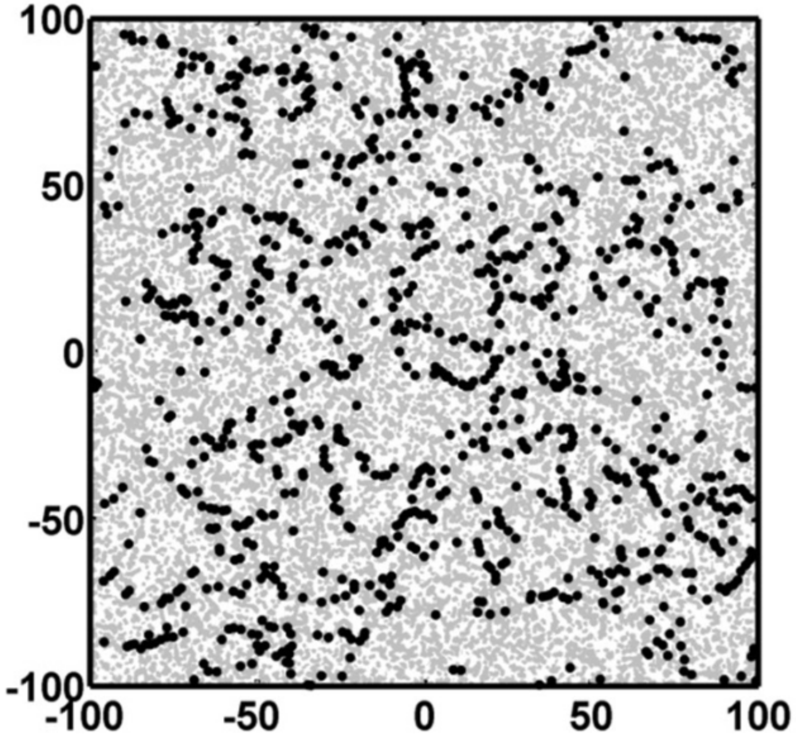


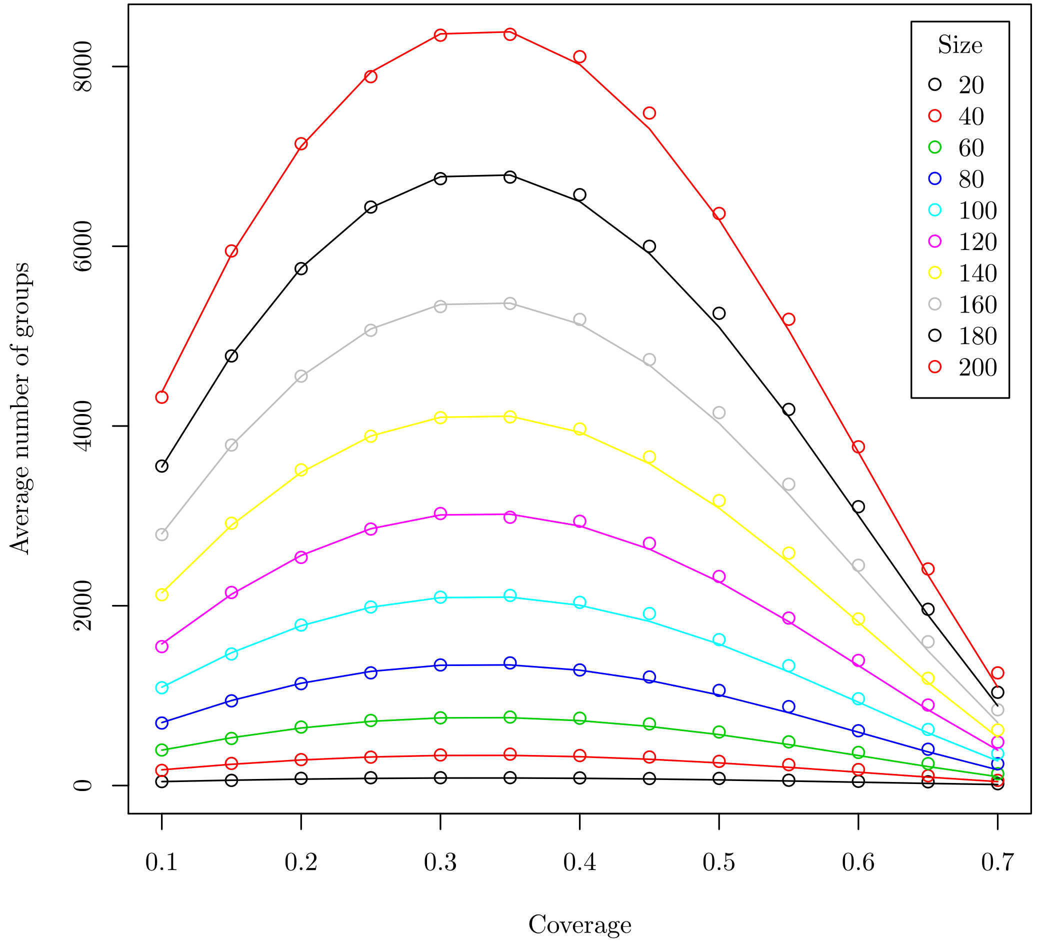
Image Source: Fostner & Brown,
Neuromorphic behavior in percolating nanoparticle films

Statistical Depositions

Depositing individual particles is really, really slow

Use distributions around known averages to “guess” groups

Same trick for the gaps between them

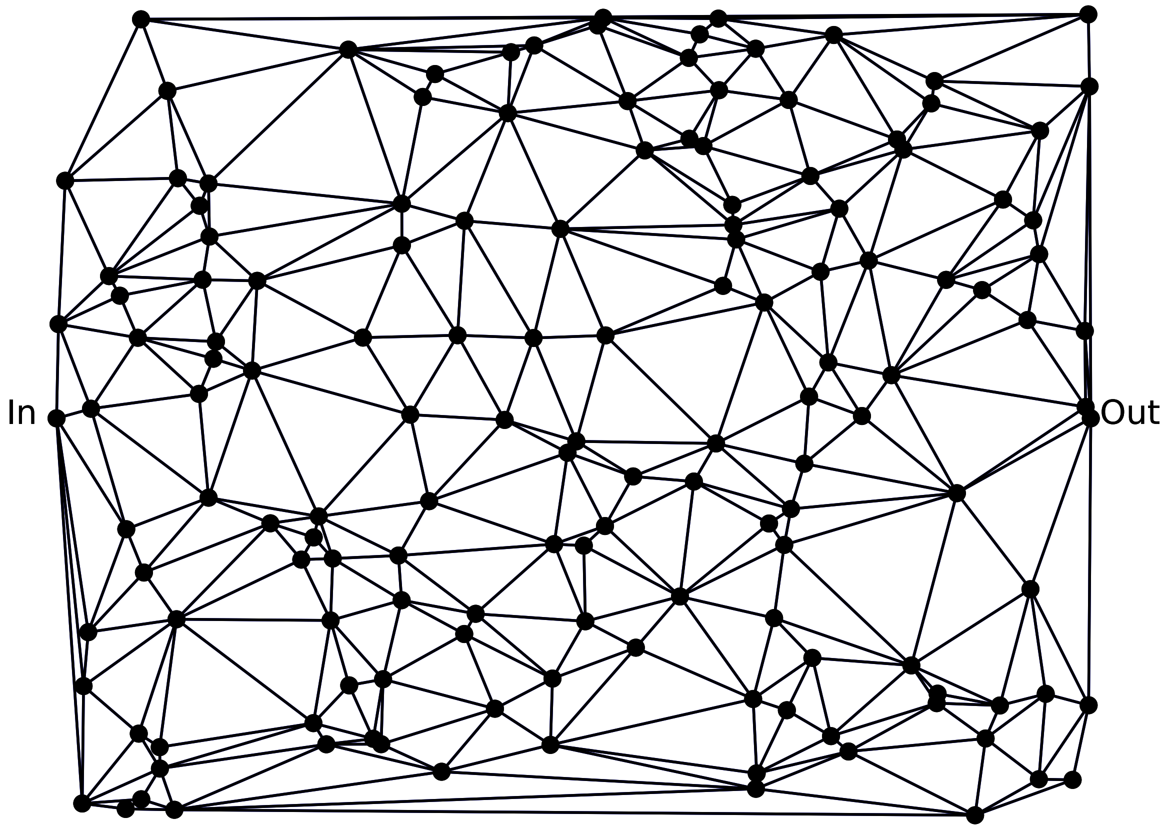


Statistical Depositions

Networks can be generated quickly

Nodes of the graph are the group centroids

Edges are the memristive connections between the groups.



Kirchhoff's Laws

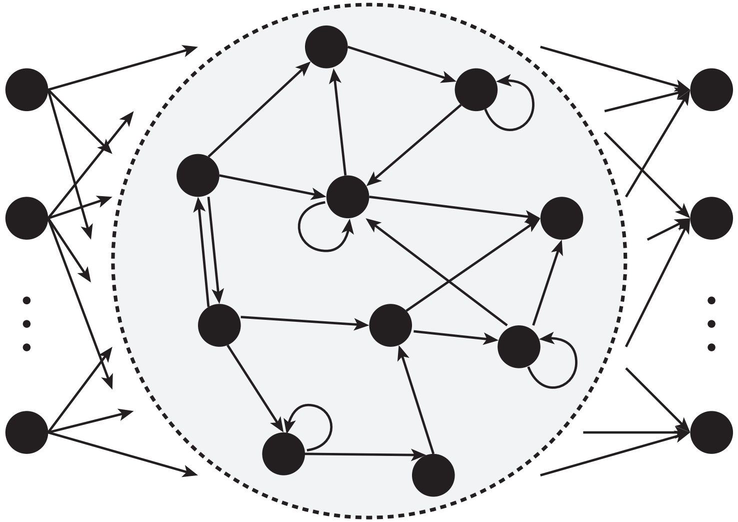
Current Law & Voltage Laws:

- Current In = Current Out
- Use all the voltage

Used for circuit simulation

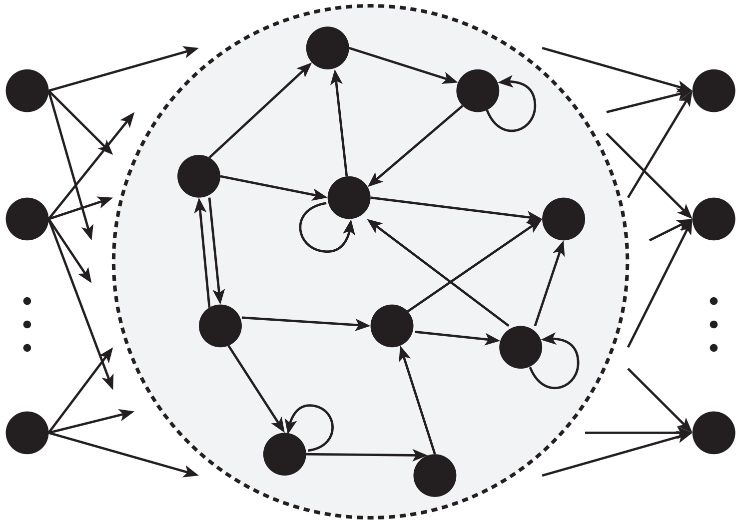
Build a big matrix

$$
\left[\begin{array}{cccc:ccc:ccc}
-(\sum G_{1i}) & G_{12} & \cdots & & 1 & & & & & \\
G_{21} & -(\sum G_{2i}) & \cdots & & & \ddots & & & & \\
\vdots & \vdots & \ddots & & & & 1 & & & \\
 & & & & & & & -1 & & \\
 & & & & & & & & \ddots & \\
 & & & & & & & & & -1 \\
\hdashline
1 & & & & & & & & & \\
 & \ddots & & & & 0 & & & 0 & \\
 & & 1 & & & & & & & \\
\hdashline
 & & 1 & & & & & & & \\
 & & & \ddots & & 0 & & & 0 & \\
 & & & 1 & & & & & & \\
\end{array}\right]
\left[\begin{array}{c}
V_1 \\ V_2 \\ \vdots \\ \\ \\ \\ I_1^{\text{in}} \\ I_2^{\text{in}} \\ \vdots \\ I_1^{\text{out}} \\ I_2^{\text{out}} \\ \vdots
\end{array}\right]
=
\left[\begin{array}{c}
0 \\ \vdots \\ \\ \\ \vdots \\ 0 \\ V_1^{\text{in}} \\ V_2^{\text{in}} \\ \vdots \\ 0 \\ \vdots \\ 0
\end{array}\right]
$$



Constructing a Reservoir

Framework



$$\text{input layer} \rightarrow \text{reservoir} \rightarrow \text{output layer}$$

Swap out each section as needed. Many variations tested quickly.

Readout Weights

Ridge regression to penalise high weights

$$\mathbf{W}^{\text{out}} = \mathbf{Y}^{\text{target}}\mathbf{X}^{\text{T}}\left(\mathbf{X}\mathbf{X}^{\text{T}} + \beta\mathbf{I}\right)^{-1}$$

Simple, linear optimisation

Actually very powerful in its own right

Faster in Fortran

Based on:

d-length input sequence,

n groups on the chip,

k loops to solve the DE for memristors,

$$O(dn^3k)$$

Lower bounded by LU matrix decomposition.

Slowest sections are now in Fortran!

Comparisons and Results

Handwriting Recognition

Memristors

```
Confusion matrix:
[[87  0  0  0  0  0  0  0  1  0]
 [12 61  0  2  0  0  0  1 12  3]
 [ 4  0 77  1  0  0  0  0  0  4]
 [ 0  1  0 67  0  2  1  4  9  7]
 [ 0  0  0  0 85  0  0  3  4  0]
 [ 0  0  0  0  0 83  1  0  2  5]
 [ 0  0  4  0  1  0 79  3  1  3]
 [ 3  1  0  0  0  0  0 76  9  0]
 [ 2  0  0  0  0  1  1  0 83  1]
 [ 3  0  0  2  0  1  0  0  8 78]]
```

90%, Woohoo! It works!

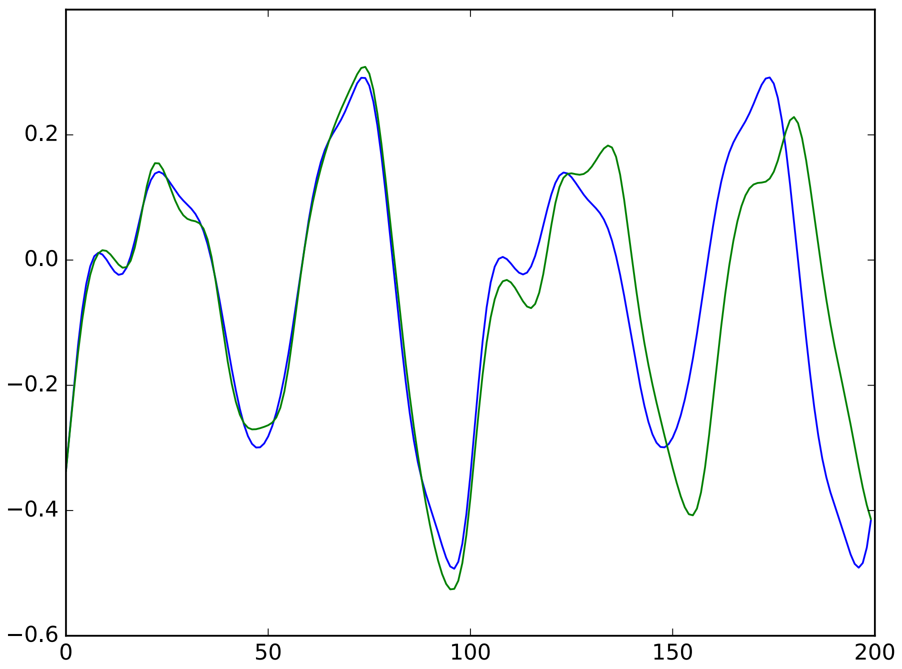
No reservoir at all

```
Confusion matrix:
[[87  0  0  0  0  0  1  0  0  0]
 [ 2 79  1  1  1  0  1  0  2  4]
 [ 3  3 78  2  0  0  0  0  0  0]
 [ 0  2  1 77  0  5  0  2  3  1]
 [ 1  1  0  0 81  0  0  2  2  5]
 [ 0  0  0  1  0 83  1  0  0  6]
 [ 0  0  1  0  0  0 90  0  0  0]
 [ 0  0  0  1  2  2  0 82  2  0]
 [ 0  3  1  2  0  4  4  1 65  8]
 [ 7  0  0  2  0  2  1  1  4 75]]
```

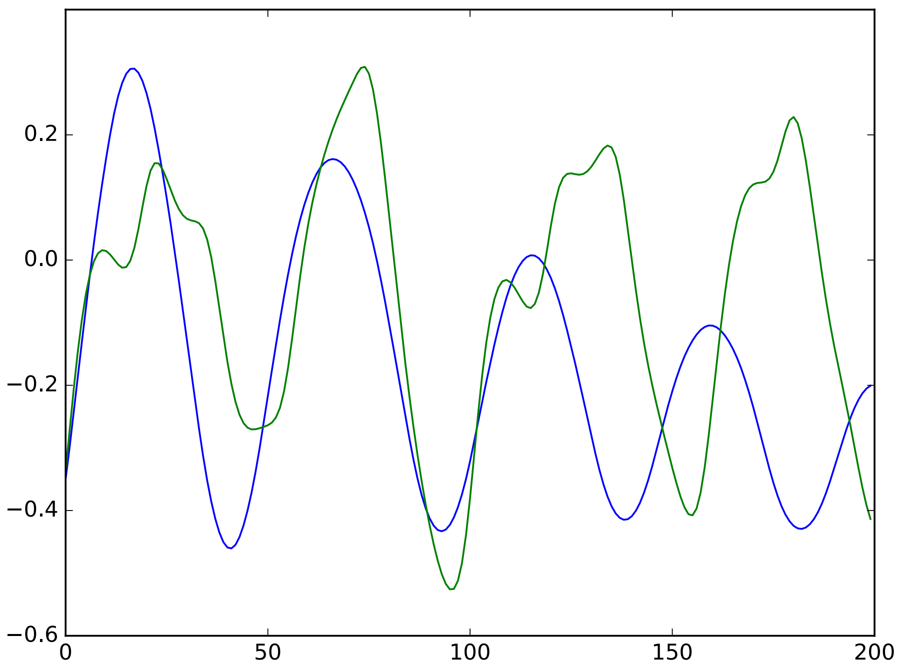
90%, Oh no! It still works!

Mackey-Glass

Echo State Network



Memristors



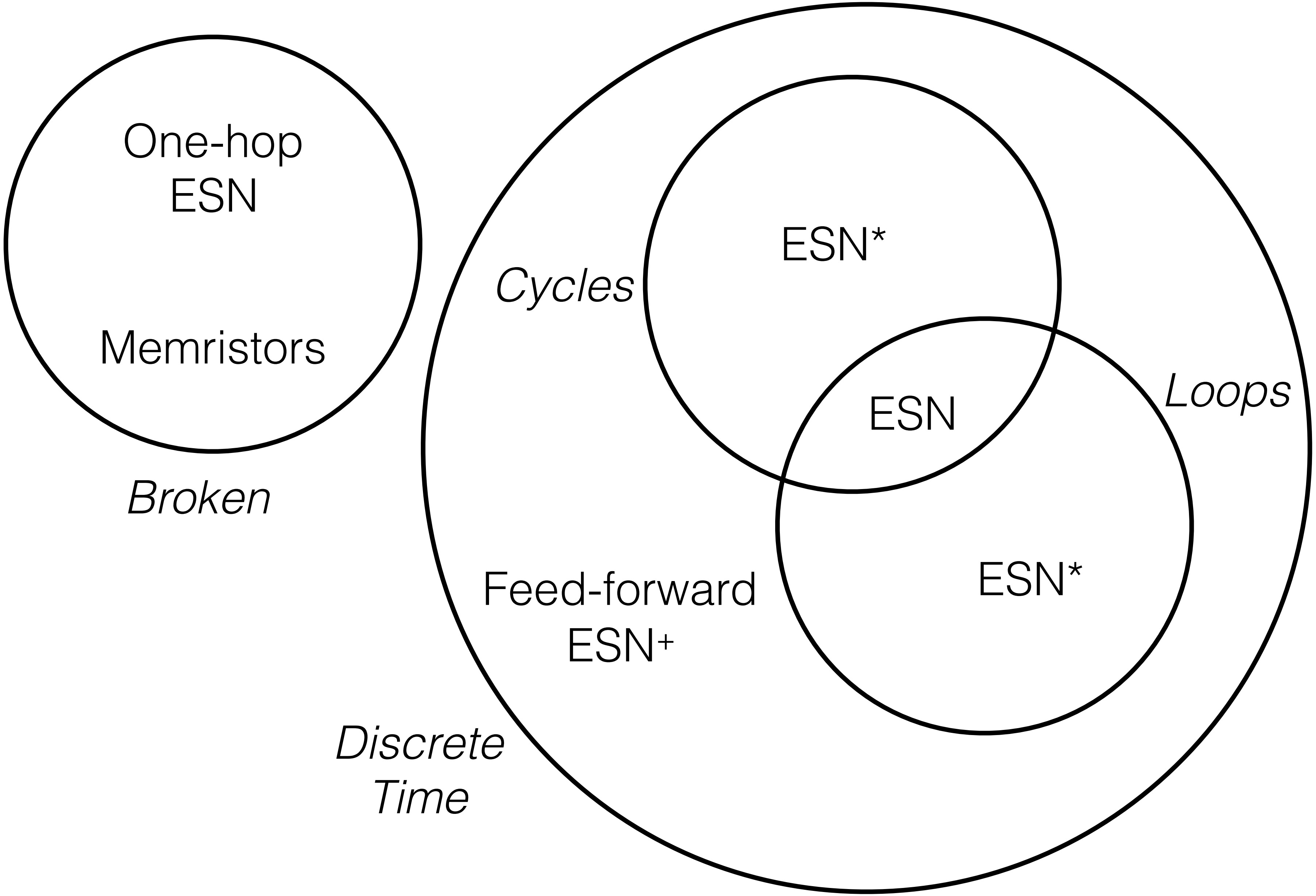
Memory

Echo state networks have four distinct sources of memory:

- Leaking
- Cycles
- Loops
- Discrete time steps

Memristive networks have two sources of memory:

- Leaking
- Conductive state



* Loops and Cycles can mimic each other

+ Čerňansky and Makula

Summary

Summary

We can speed up simulations with statistics

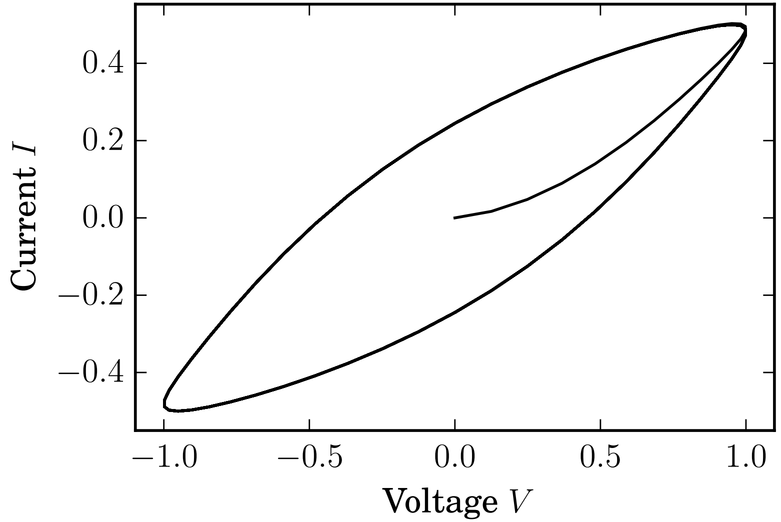
Homogeneous neuromorphic hardware is missing key features

Cycles can mimic loops, and loops can mimic cycles

Discrete time is the single most important part of reservoir learners

Future Work

Other network components: the ESN “IV” curve looks like a capacitor or inductor IV curve



Heterogenous networks with “neurons,” e.g. delay mechanisms

Alternative information encoding that the network may be able to handle better

Specialised hardware layout: Solving mazes with memristors: A massively parallel approach - Pershin and Di Ventra

Neuromorphic Computing with Reservoir Neural Networks on Memristive Hardware

Aaron Stockdill

September 2016