OCES 3301 – Guest Lecture –

Reservoir Computing

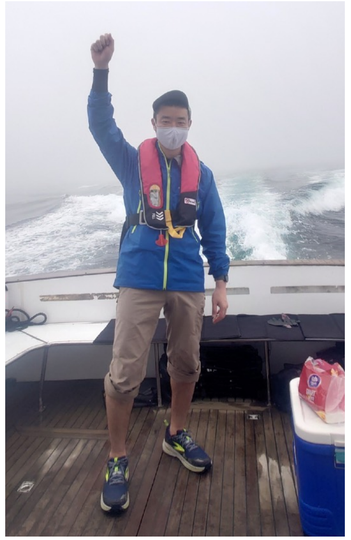
A simple and unique method to analyze time series

25 April 2023, 9:00–12:00

Masayuki USHIO, Assistant Professor at OCES ushio@ust.hk

Who am I?

- Masayuki USHIO (ushio@ust.hk, Room 5525)
- Assistant Professor at Department of Ocean Science (OCES)
- Ph.D at Kyoto University, Japan
- Joined OCES in September 2022



@HKUST

@Borneo

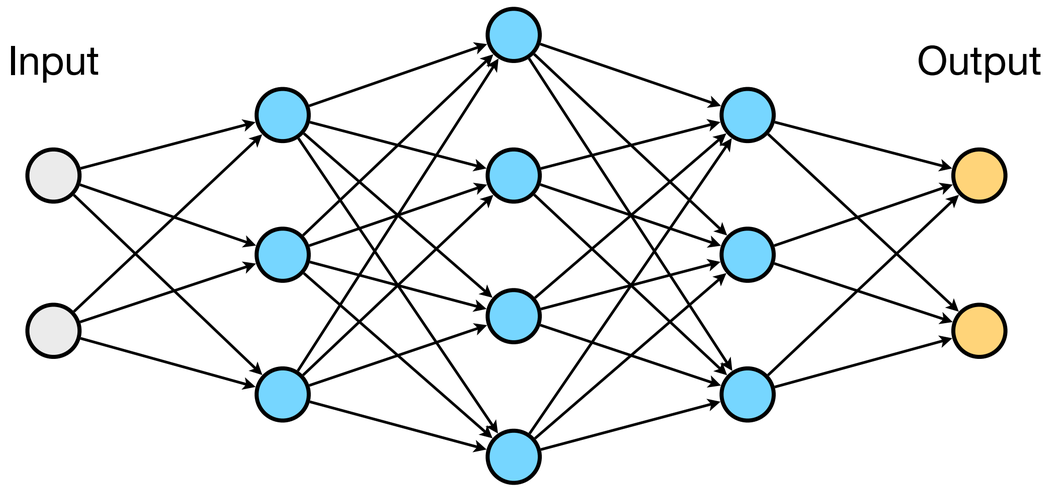
Outline

1. Brief introduction to artificial neural networks
   - Perceptron, feedforward neural networks, and recurrent neural networks
2. A special type of recurrent neural networks
   - Reservoir computing
3. A simple reservoir computing example
4. (Advanced topic) A unique application of reservoir computing

Brief introduction of artificial neural networks

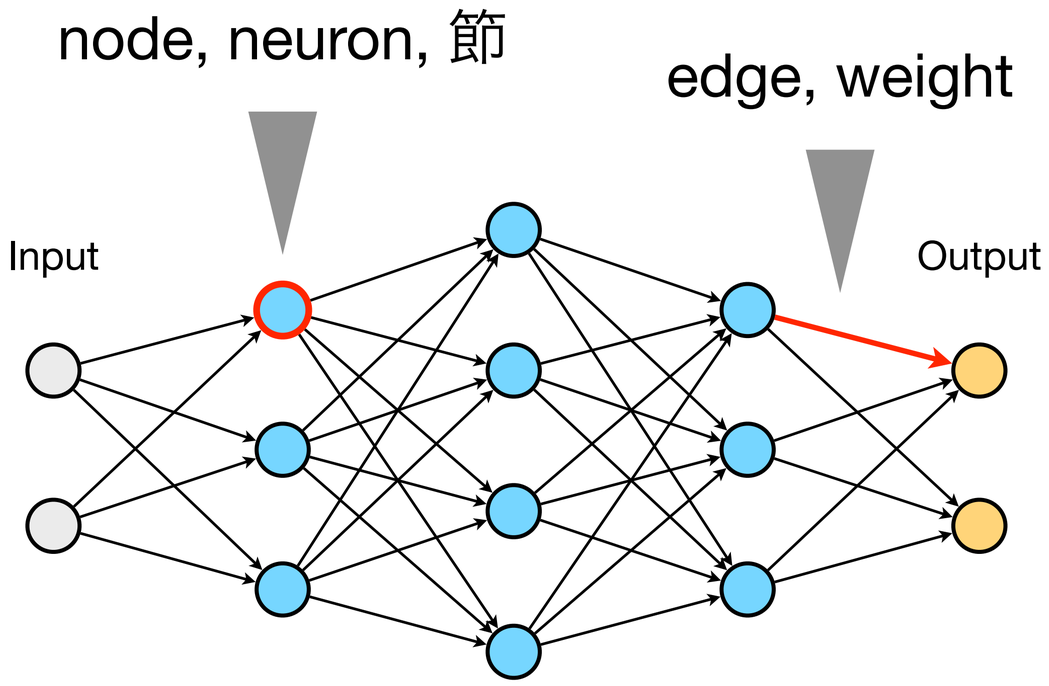
Brief introduction of artificial neural networks

- What is "**artificial neural network** (**ANN**)"?
  (https://en.wikipedia.org/wiki/Artificial_neural_network)
  - ANNs are computing systems inspired by the biological neural networks
  - ANN is a collection of **nodes** connected by (many) **edges** (links)



Brief introduction of artificial neural networks

- What is "**artificial neural network** (**ANN**)"?
  (https://en.wikipedia.org/wiki/Artificial_neural_network)



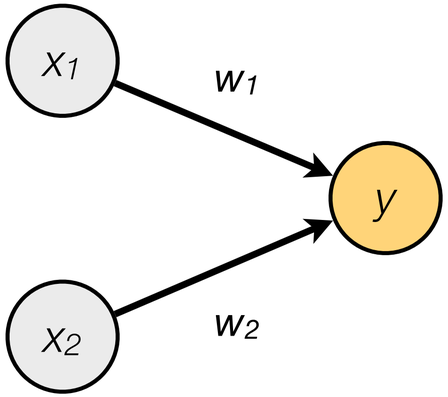
Brief introduction of artificial neural networks

https://github.com/oreilly-japan/deep-learning-from-scratch

- Perceptron

$$y = \begin{cases} 0 & (if\ w_1 \times x_1 + w_2 \times x_2 \leq \theta) \\ 1 & (if\ w_1 \times x_1 + w_2 \times x_2 > \theta) \end{cases}$$

Input Output



AND gate

$$(w_1, w_2, \theta) = (1,1,1)$$

$$y = \begin{cases} 0 & (if\ x_1 + x_2 \leq 1) \\ 1 & (if\ x_1 + x_2 > 1) \end{cases}$$

Example Output

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

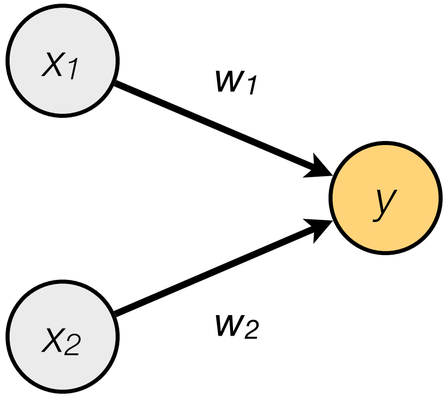
Brief introduction of artificial neural networks

https://github.com/oreilly-japan/deep-learning-from-scratch

- Perceptron

$$y = \begin{cases} 0 & (if\ w_1 \times x_1 + w_2 \times x_2 \leq \theta) \\ 1 & (if\ w_1 \times x_1 + w_2 \times x_2 > \theta) \end{cases}$$

Input Output



Not AND (or NAND) gate

$$(w_1, w_2, \theta) = (-0.5, -0.5, -0.7)$$

$$y = \begin{cases} 0 & (if\ -0.5 \times x_1 - 0.5 \times x_2 \leq -0.7) \\ 1 & (if\ -0.5 \times x_1 - 0.5 \times x_2 > -0.7) \end{cases}$$

Example Output

x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

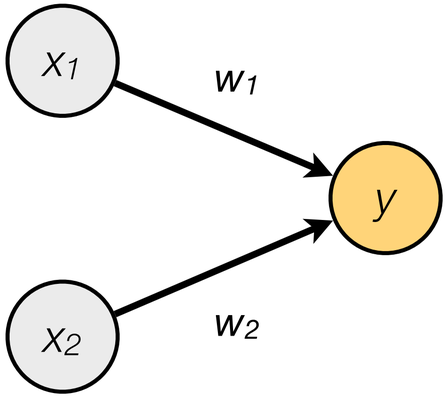
Brief introduction of artificial neural networks

https://github.com/oreilly-japan/deep-learning-from-scratch

- Perceptron

$$y = \begin{cases} 0 & (if\ w_1 \times x_1 + w_2 \times x_2 \leq \theta) \\ 1 & (if\ w_1 \times x_1 + w_2 \times x_2 > \theta) \end{cases}$$

Input Output



OR gate

$$(w_1, w_2, \theta) = (1,1,0)$$

$$y = \begin{cases} 0 & (if\ x_1 + x_2 \leq 0) \\ 1 & (if\ x_1 + x_2 > 0) \end{cases}$$

Example Output

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

Brief introduction of artificial neural networks

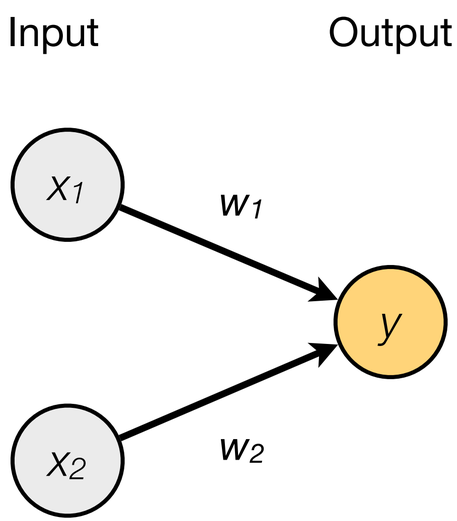
https://github.com/oreilly-japan/deep-learning-from-scratch
MIT License

- Perceptron

$$y = \begin{cases} 0 & (if\ w_1 \times x_1 + w_2 \times x_2 \leq \theta) \\ 1 & (if\ w_1 \times x_1 + w_2 \times x_2 > \theta) \end{cases}$$

Re-write

$$y = \begin{cases} 0 & (if\ b + w_1 \times x_1 + w_2 \times x_2 \leq 0) \\ 1 & (if\ b + w_1 \times x_1 + w_2 \times x_2 > 0) \end{cases}$$



b: bias
w_1, w_2: weight

Hands-on: Let's try!

https://github.com/oreilly-japan/deep-learning-from-scratch
MIT License

https://github.com/ong8181/OCES3301_guest

Open Jupyter notebook in your local PC

If Jupyter notebook is already installed in your PC, the most straightforward way to run the scripts is probably downloading all the files in this repository and open them using jupyter notebook. Alternatively, you can open them in Google Colab, but loading a demo file is a bit tricky.

Open Jupyter notebook in Google Colab

Perceptron

- Open in Colab 01_and_gate.ipynb
- Open in Colab 02_nand_gate.ipynb
- Open in Colab 03_or_gate.ipynb
- Open in Colab 04_xor_gate.ipynb

Brief introduction of artificial neural networks

https://github.com/oreilly-japan/deep-learning-from-scratch

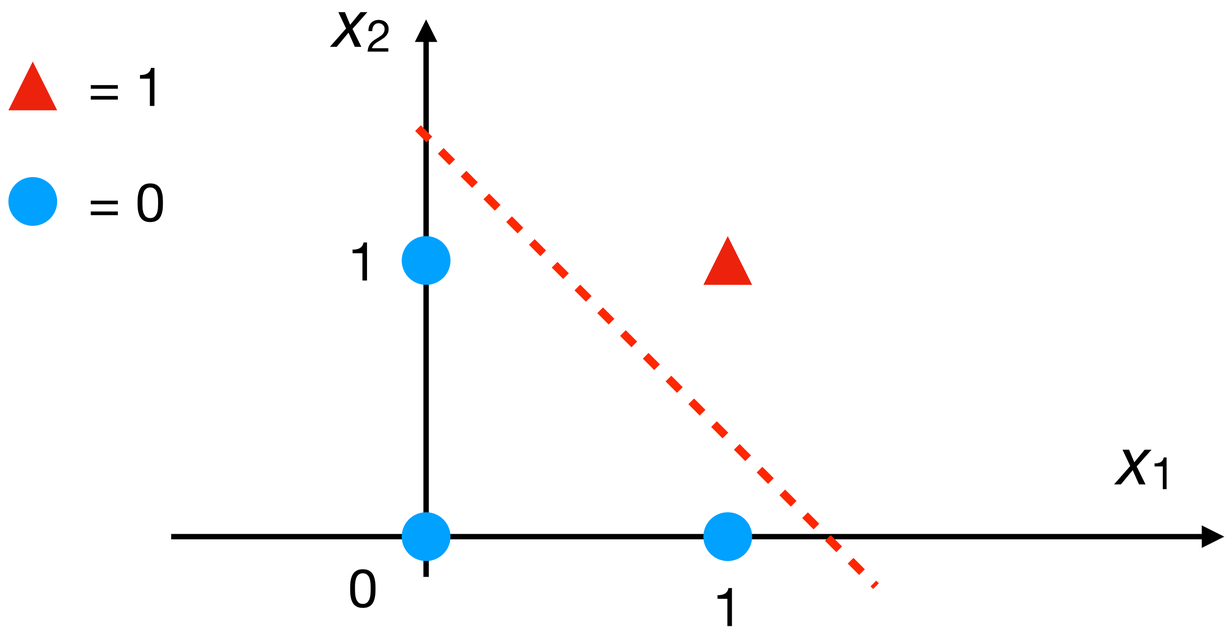
- Why are these simple transformations useful?

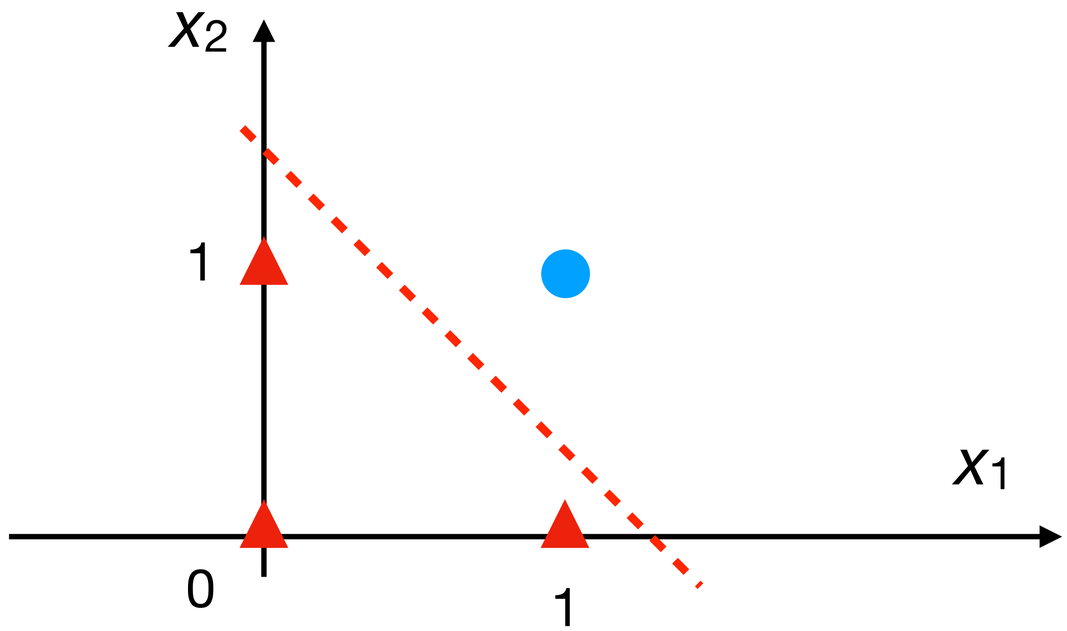
AND gate

$$y = \begin{cases} 0 & (if\ -1.2 + x_1 + x_2 \leq 0) \\ 1 & (if\ -1.2 + x_1 + x_2 > 0) \end{cases}$$

NAND gate

$$y = \begin{cases} 0 & (if\ 1.2 - x_1 - x_2 \leq 0) \\ 1 & (if\ 1.2 - x_1 - x_2 > 0) \end{cases}$$





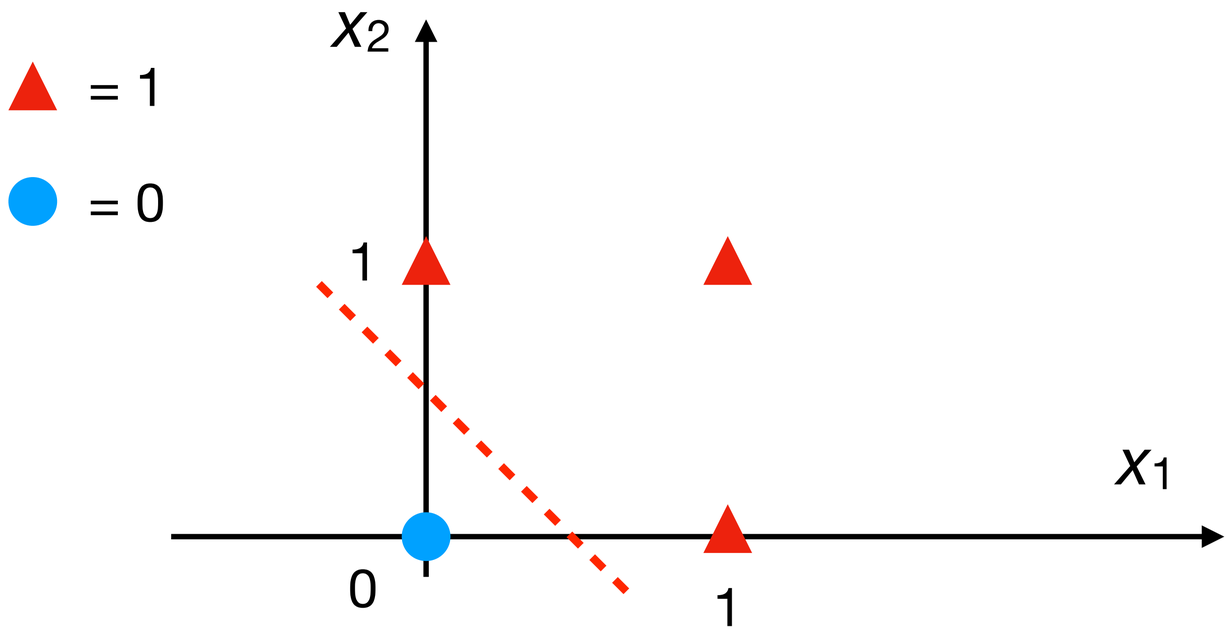
Brief introduction of artificial neural networks

https://github.com/oreilly-japan/deep-learning-from-scratch

- Why are these simple transformations useful?

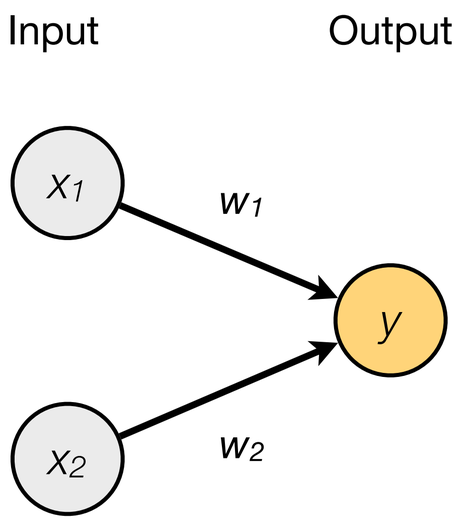
OR gate

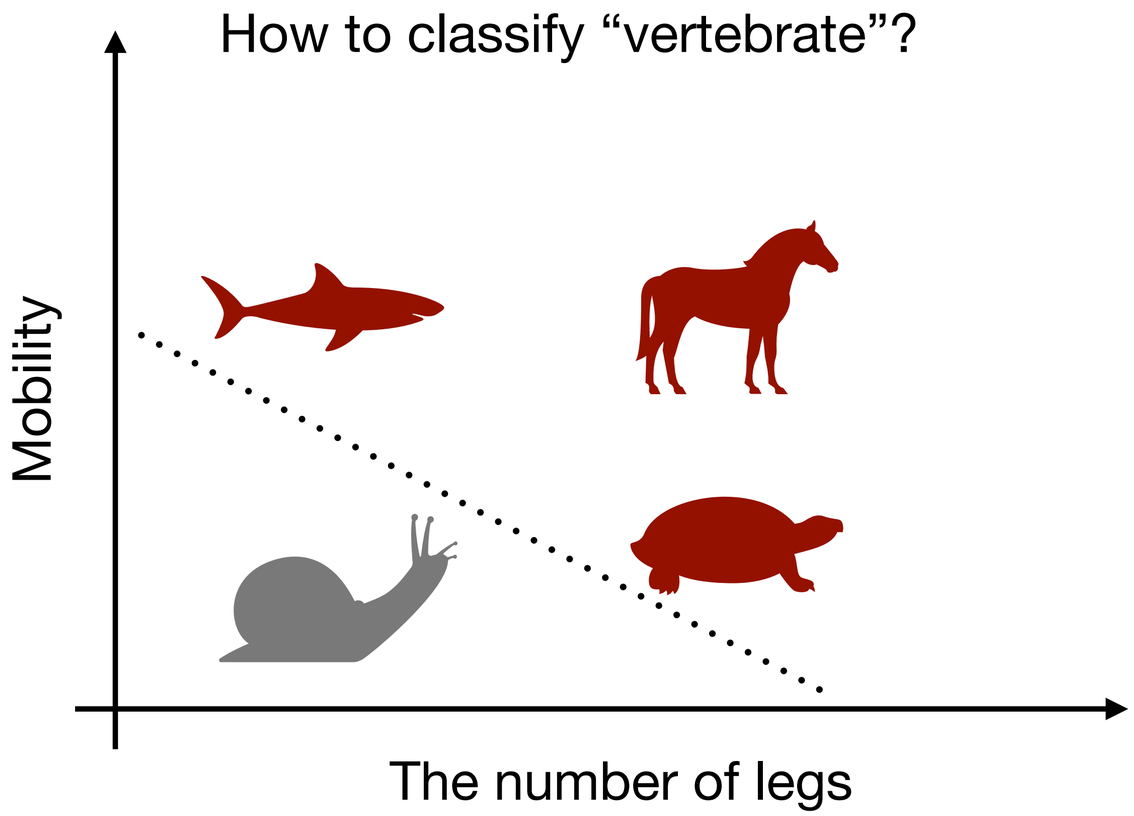
$$y = \begin{cases} 0 & (if \ -0.5 + x_1 + x_2 \leq 0) \\ 1 & (if \ -0.5 + x_1 + x_2 > 0) \end{cases}$$



Brief introduction of artificial neural networks

- Why are these simple transformations useful?



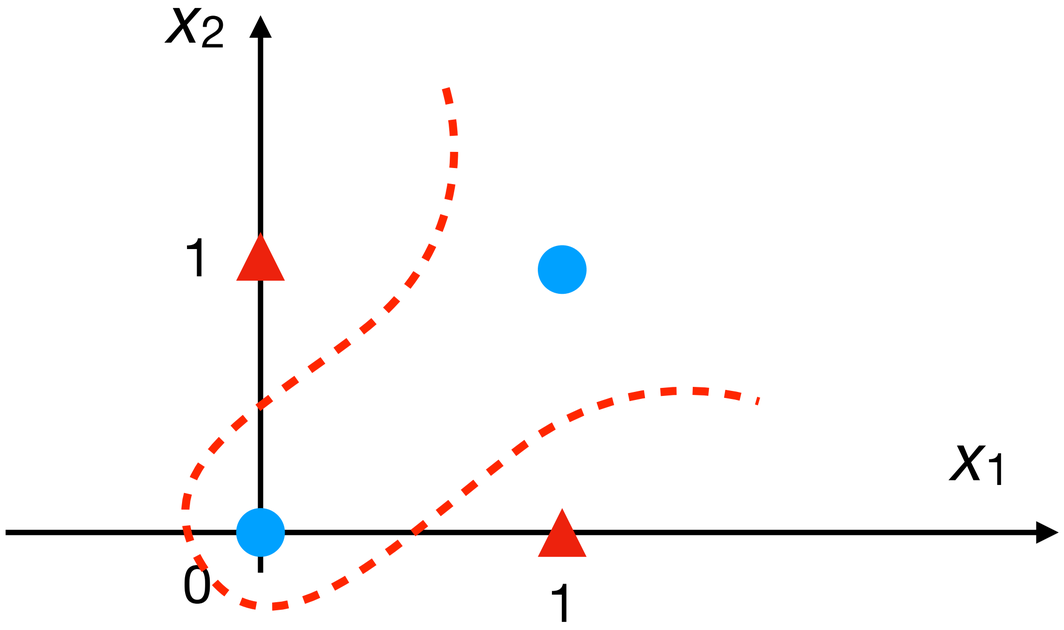


Brief introduction of artificial neural networks

https://github.com/oreilly-japan/deep-learning-from-scratch

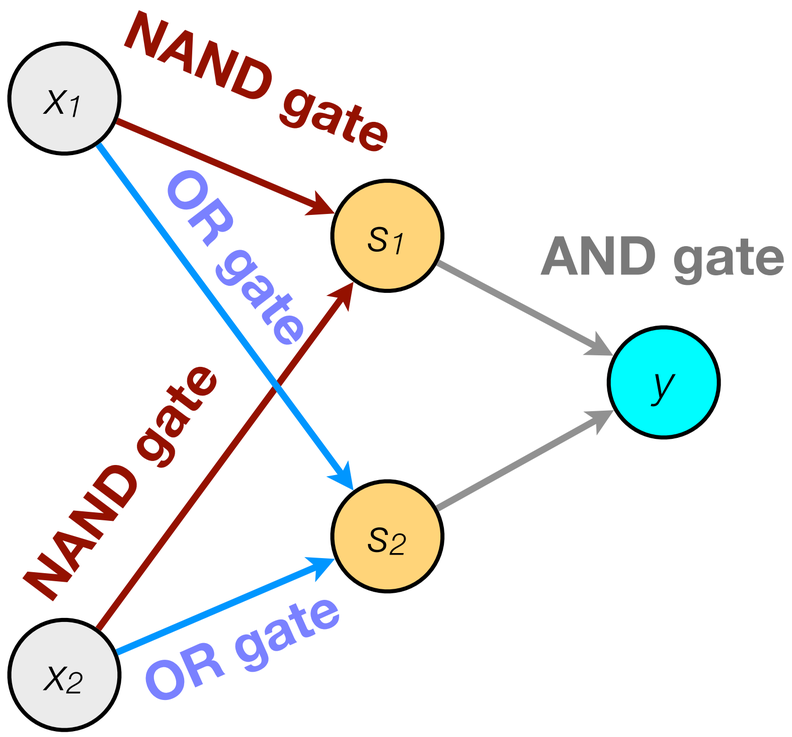
- Why are these simple transformations useful?

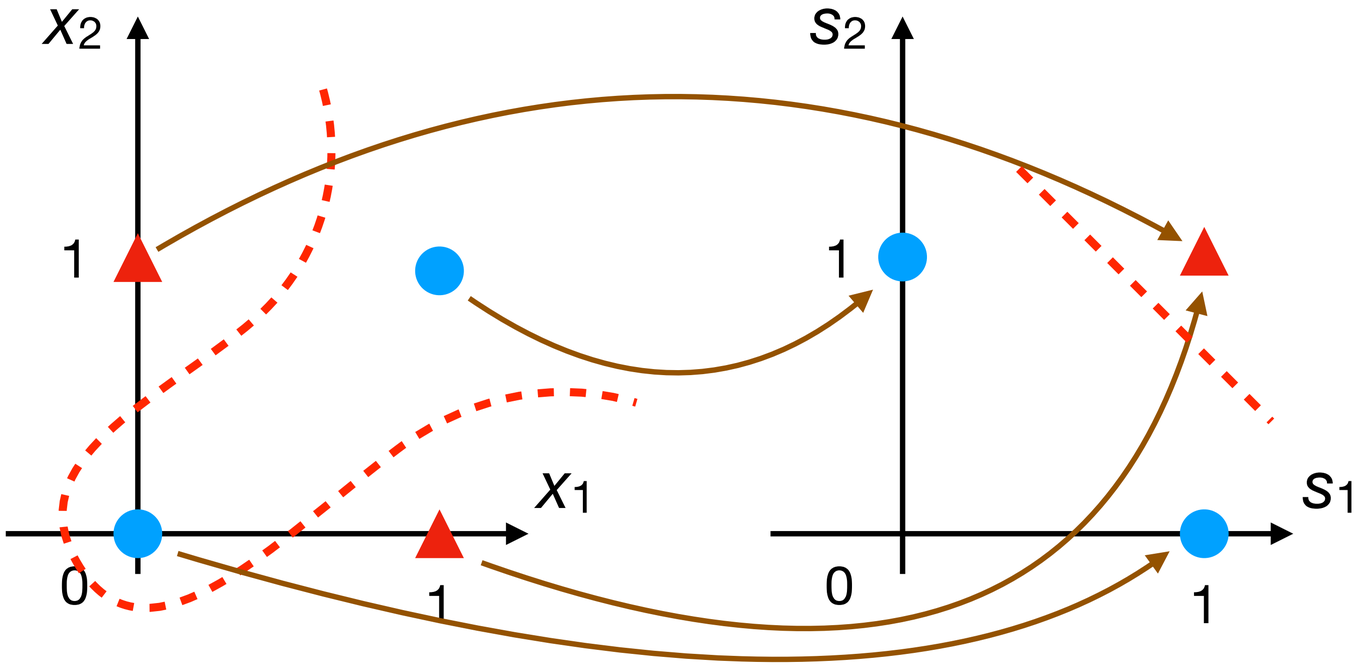
Nonlinear separation??



Brief introduction of artificial neural networks

- Use of two gates





x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

Hands-on: Let's try!

https://github.com/oreilly-japan/deep-learning-from-scratch
MIT License

https://github.com/ong8181/OCES3301_guest

Open Jupyter notebook in your local PC

If Jupyter notebook is already installed in your PC, the most straightforward way to run the scripts is probably downloading all the files in this repository and open them using jupyter notebook. Alternatively, you can open them in Google Colab, but loading a demo file is a bit tricky.

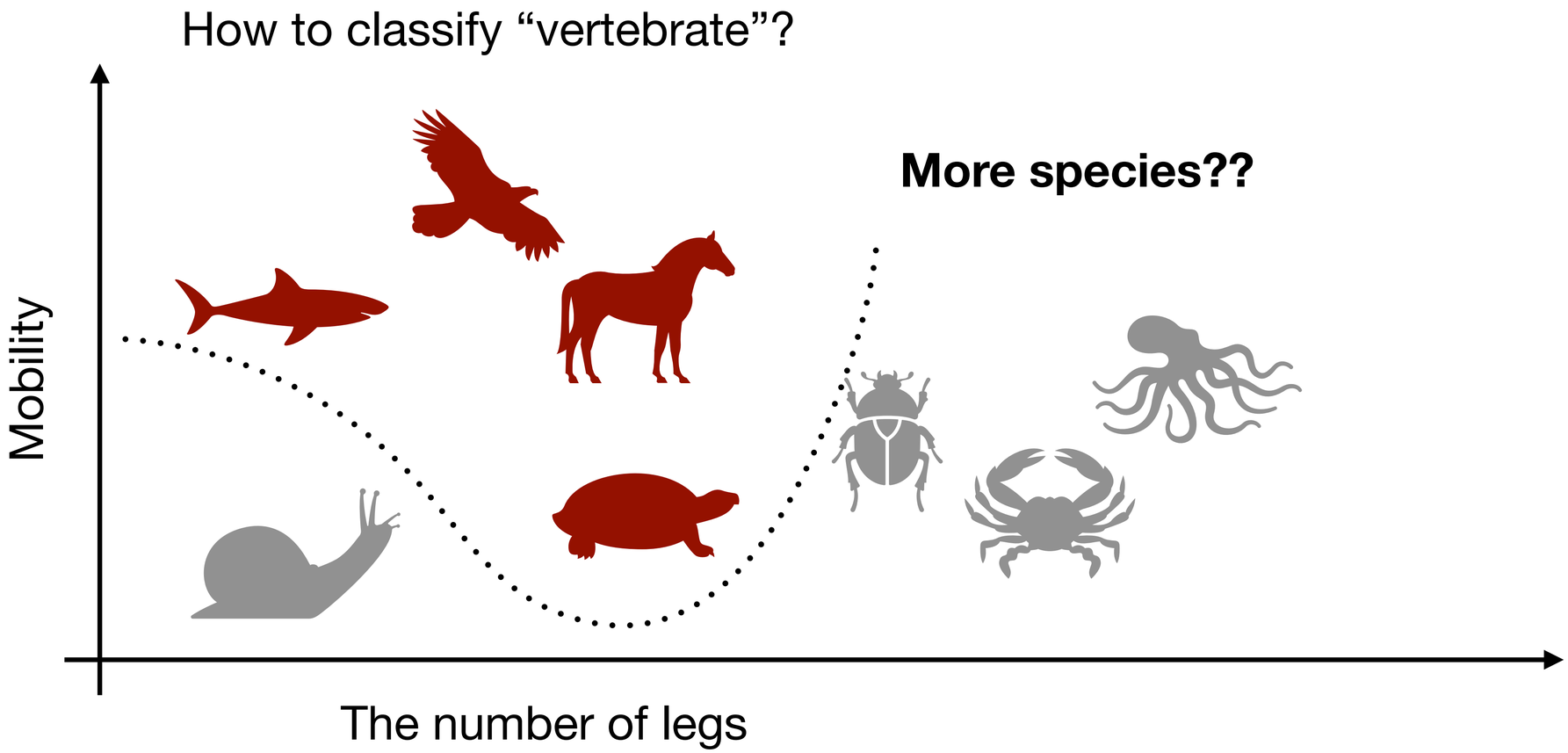
Open Jupyter notebook in Google Colab

Perceptron

- Open in Colab 01_and_gate.ipynb
- Open in Colab 02_nand_gate.ipynb
- Open in Colab 03_or_gate.ipynb
- Open in Colab 04_xor_gate.ipynb

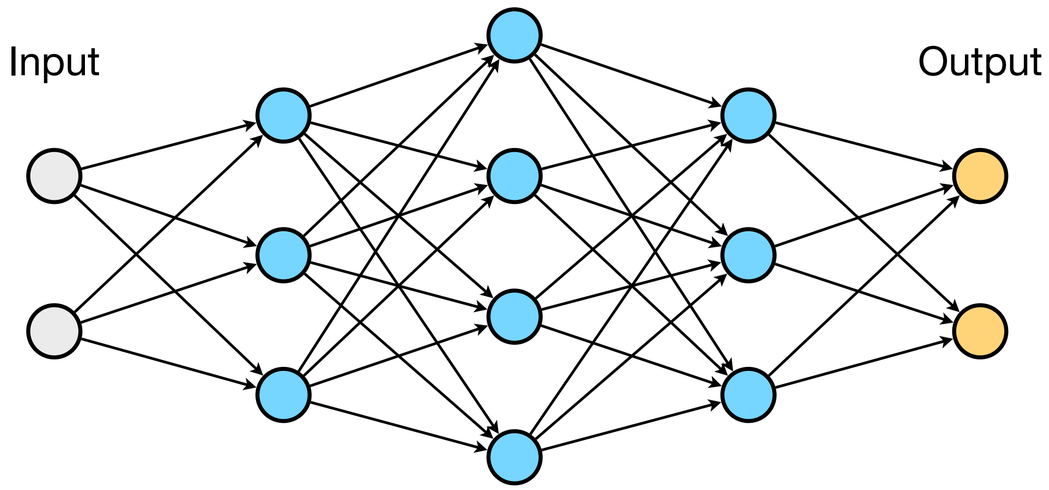
Brief introduction of artificial neural networks

- More layers may be useful for more complicated classifications



Brief introduction of artificial neural networks

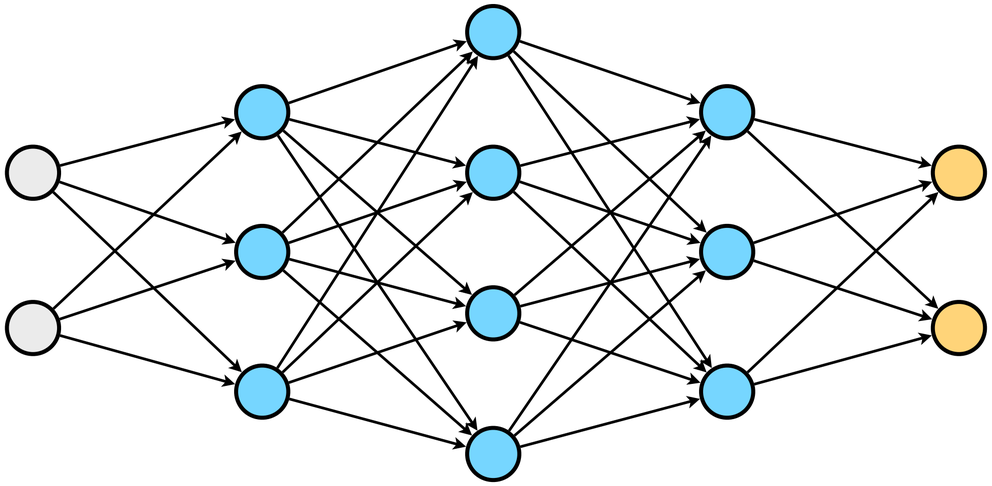
- More complicated task → more layers



Brief introduction of artificial neural networks

- Two types of ANNs

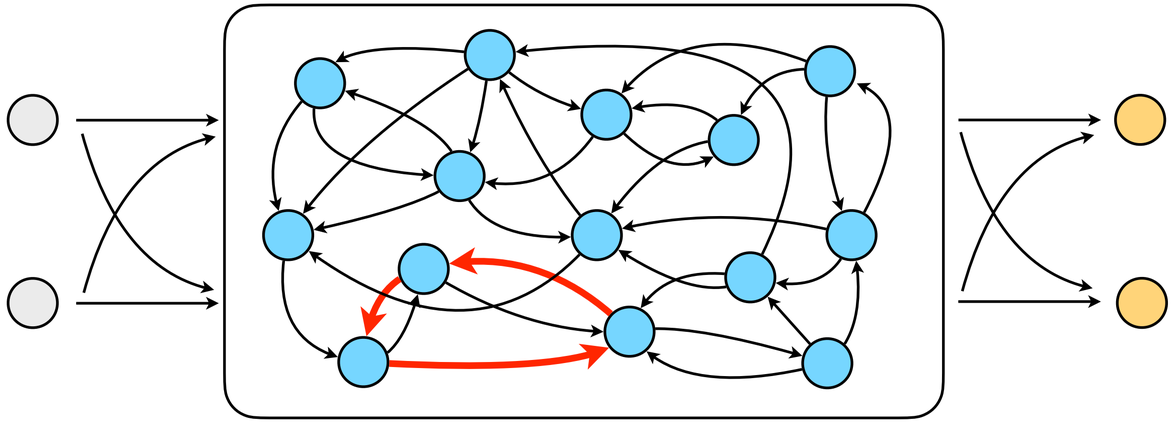
Feed-forward neural network



Signals transmitted one direction

Image analysis

Recurrent neural network

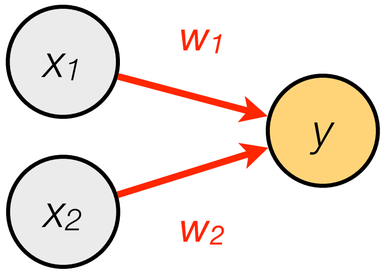


There are feedback loops

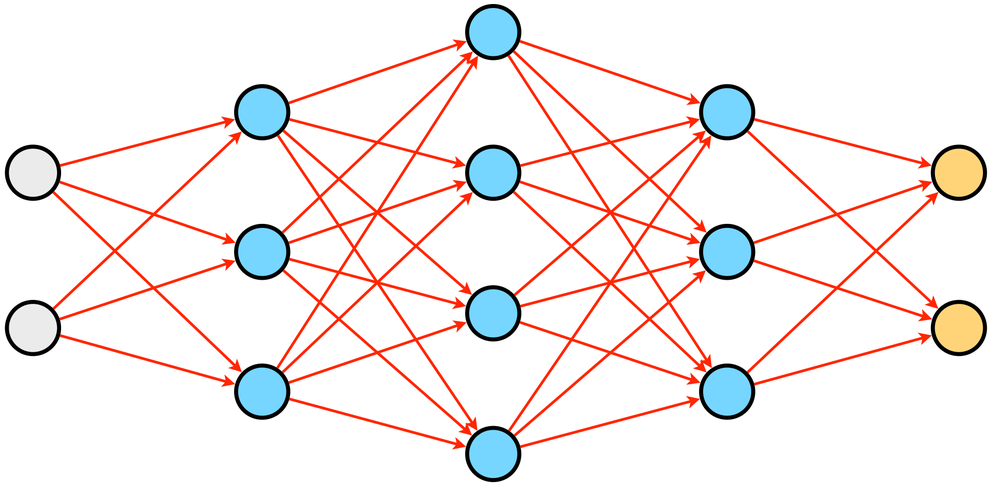
Time series analysis

Brief introduction of artificial neural networks

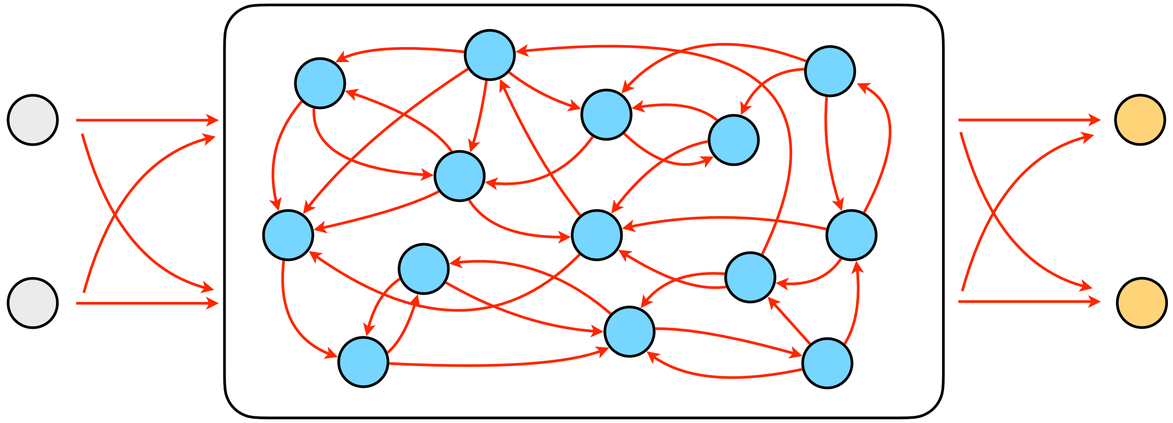
- **Learning optimal weights** is time-consuming



Feed-forward neural network



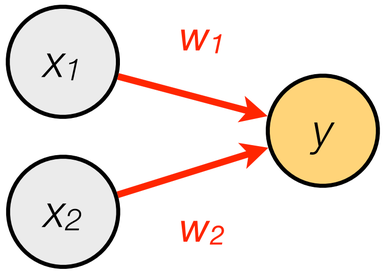
Recurrent neural network



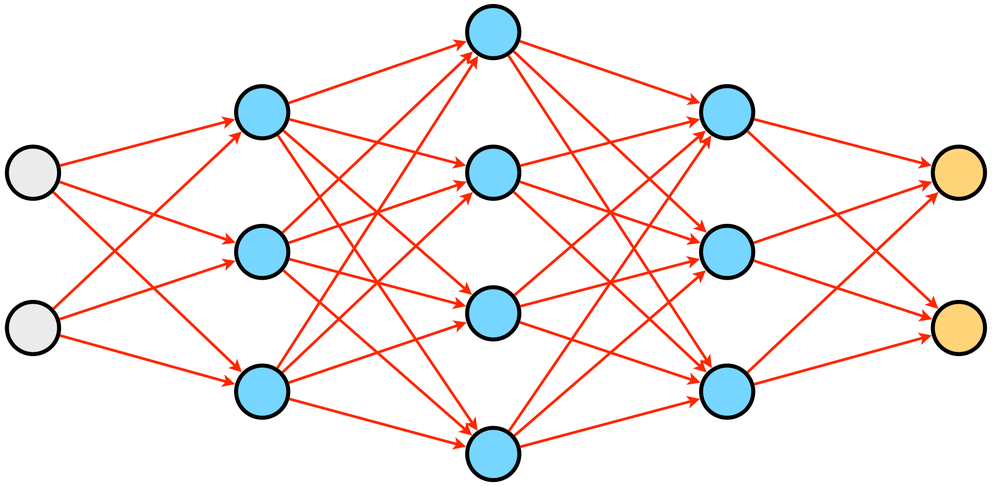
How to determine optimal weights?

Brief introduction of artificial neural networks

- **Learning optimal weights** is time-consuming

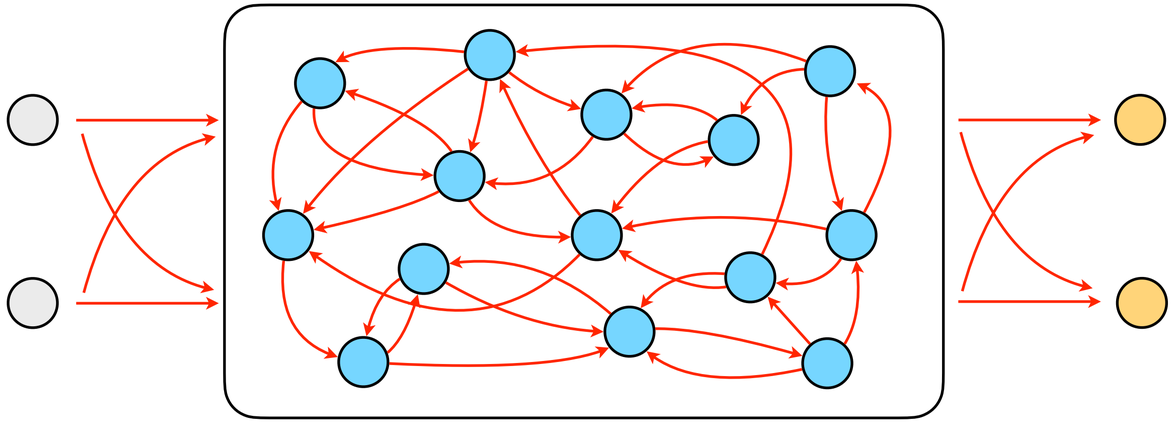


Feed-forward neural network



Back propagation

Recurrent neural network



Back propagation through time

Mini-summary

1. ANNs are computing systems inspired by the biological neural networks
2. ANN is a collection of **nodes** connected by (many) **edges** (links)
3. ANN transforms input data nonlinearly (= flexibly) and allows linear separation of the input data
4. Learning process (= optimization of edge weights) is time-consuming

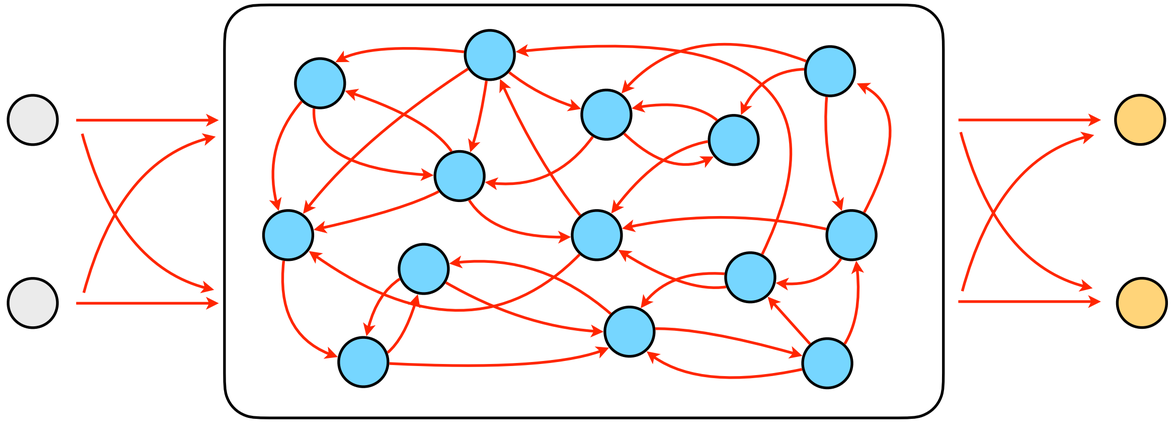
Reservoir computing

A special type of recurrent neural networks

Reservoir computing (RC)

- **Learning optimal weights** is time-consuming

Recurrent neural network

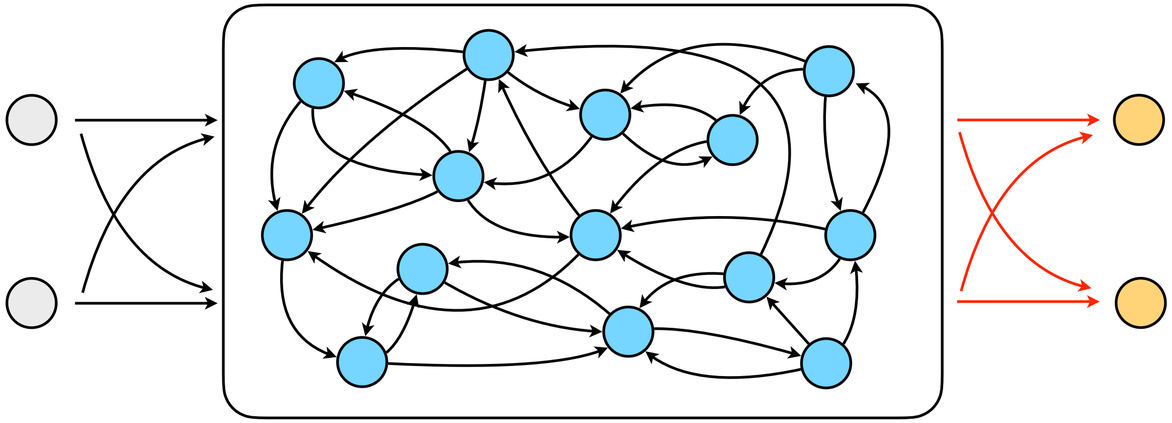


38 weights

Reservoir computing (RC)

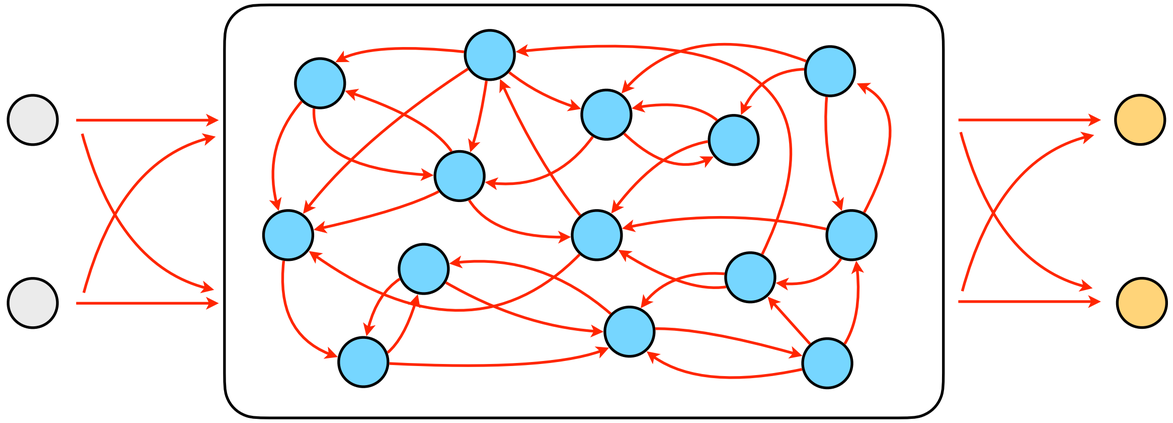
- **Learning optimal weights** is time-consuming

Reservoir computing



4 weights

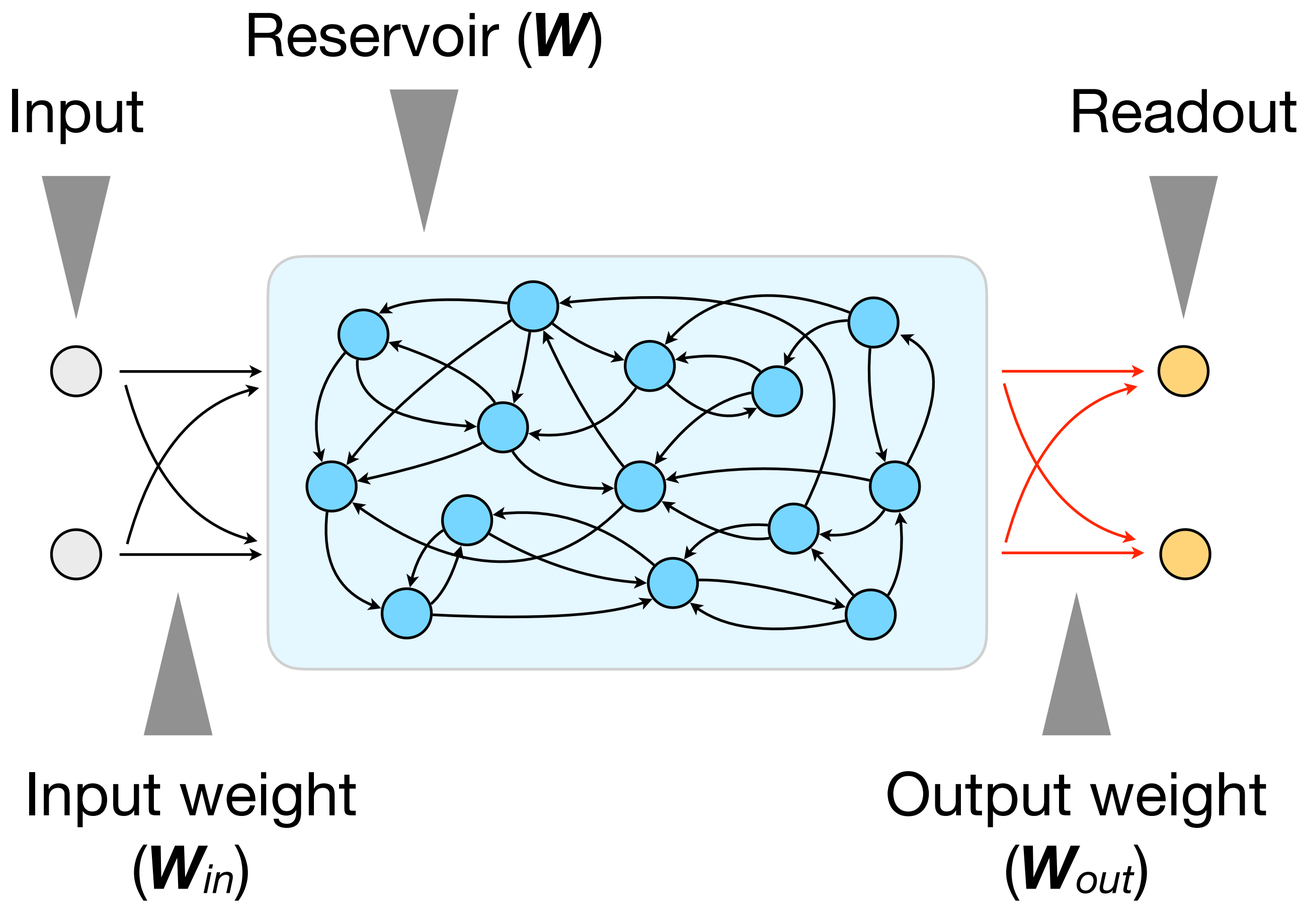
Recurrent neural network



38 weights

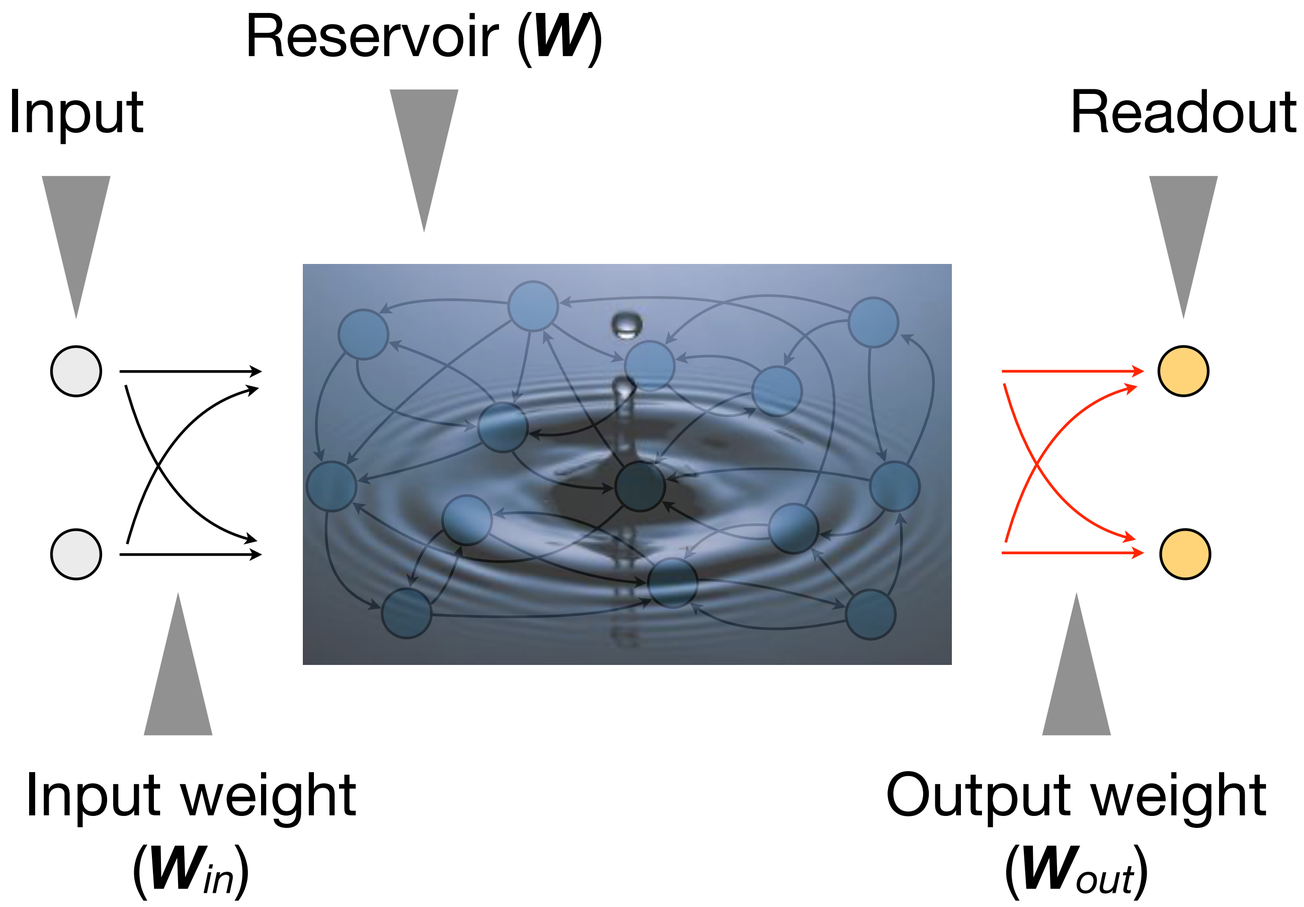
Reservoir computing (RC)

- Terminologies



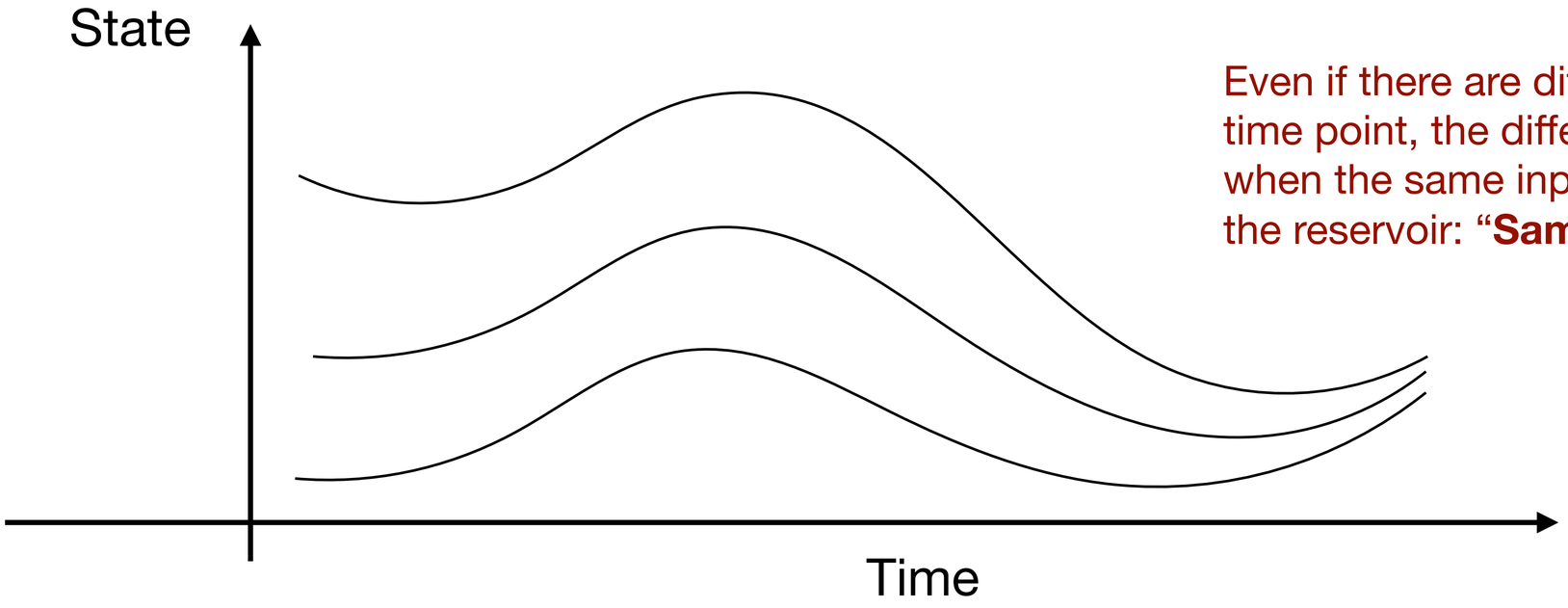
Reservoir computing (RC)

- Terminologies



Reservoir computing (RC)

- Some rules
  - ***W*** should satisfy **Echo State Property (ESP)**. ***W*** satisfies ESP when it returns identical results to identical input sequence (i.e., the process should be deterministic and does not depend on the initial state).



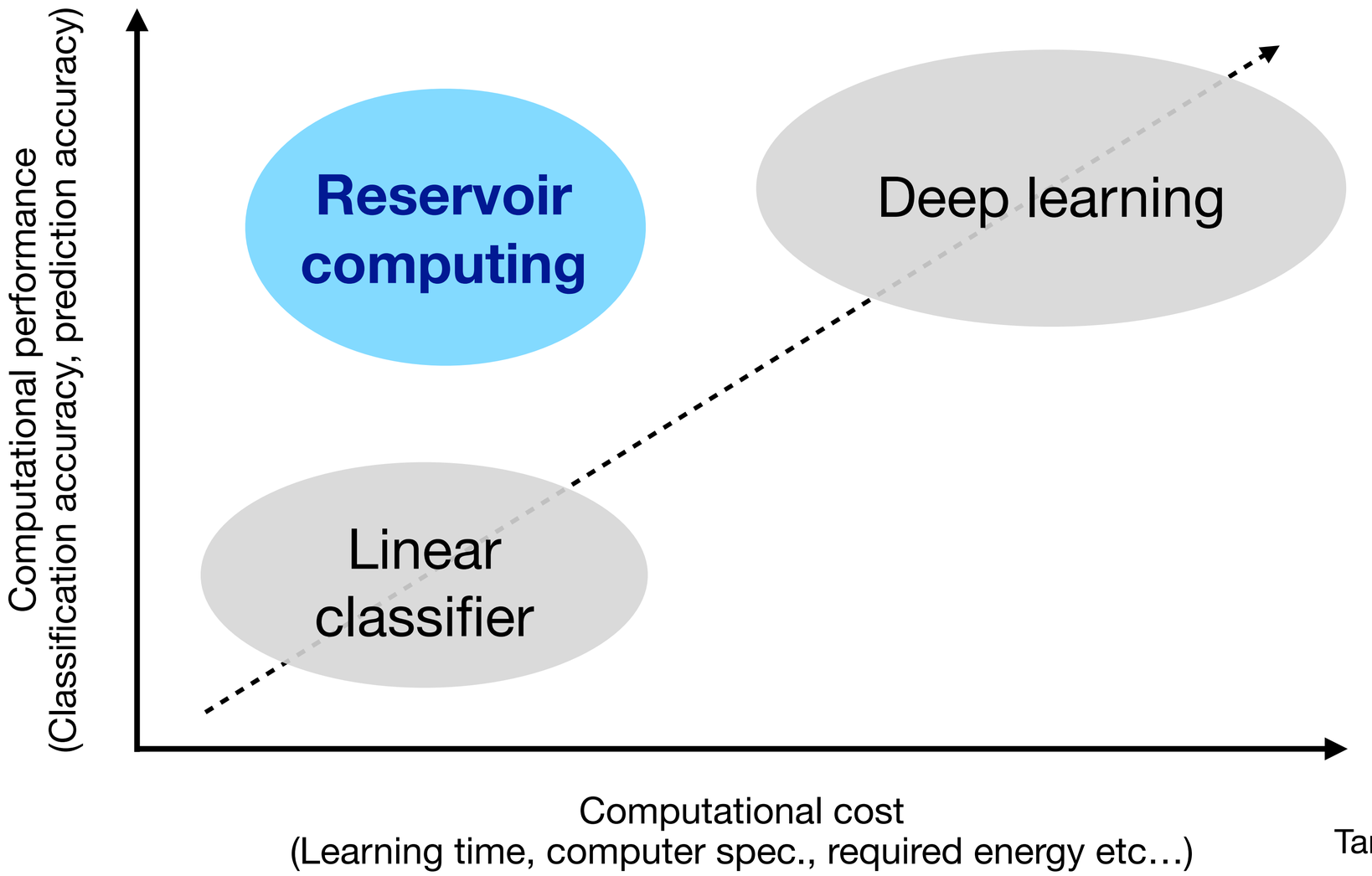
Even if there are differences at the initial time point, the differences should shrink when the same input sequence is added to the reservoir: "**Same input, same output**"

Modified from
Tanaka et al. (2021) "Reservoir Computing"
(In Japanese)

Reservoir computing (RC)

- Some rules
  - To satisfy ESP, the absolute value of the leading eigenvalue (so-called “spectral radius”) should *normally* be less than 1.
  - However, the exact condition required for ESP is still unknown, and sometimes the spectral radius larger than or equal to 1 is OK.
  - The more nodes W has, the better the performance is (usually up to 1000 or 2000 nodes).

Performance of RC



Modified from
Tanaka et al. (2021) “Reservoir Computing”
(In Japanese)

Mini-summary

1. Reservoir computing (RC) is a special type of recurrent neural network
2. RC does not train its network (= reservoir)
3. The computational performance of RC is similar to (or slightly less than) that of fine-tuned neural networks (e.g., deep learning)
4. The training cost of RC is much smaller than that of fine-tuned neural networks and similar to (or slightly higher than) that of linear classifiers

A simple reservoir computing example

Implementation with Python

Hands-on: Let's try!

https://mantas.info/code/simple_esn/
MIT License

https://github.com/ong8181/OCES3301_guest

Open Jupyter notebook in your local PC

If Jupyter notebook is already installed in your PC, the most straightforward way to run the scripts is probably downloading all the files in this repository and open them using jupyter notebook. Alternatively, you can open them in Google Colab, but loading a demo file is a bit tricky.

Open Jupyter notebook in Google Colab

Echo State Network

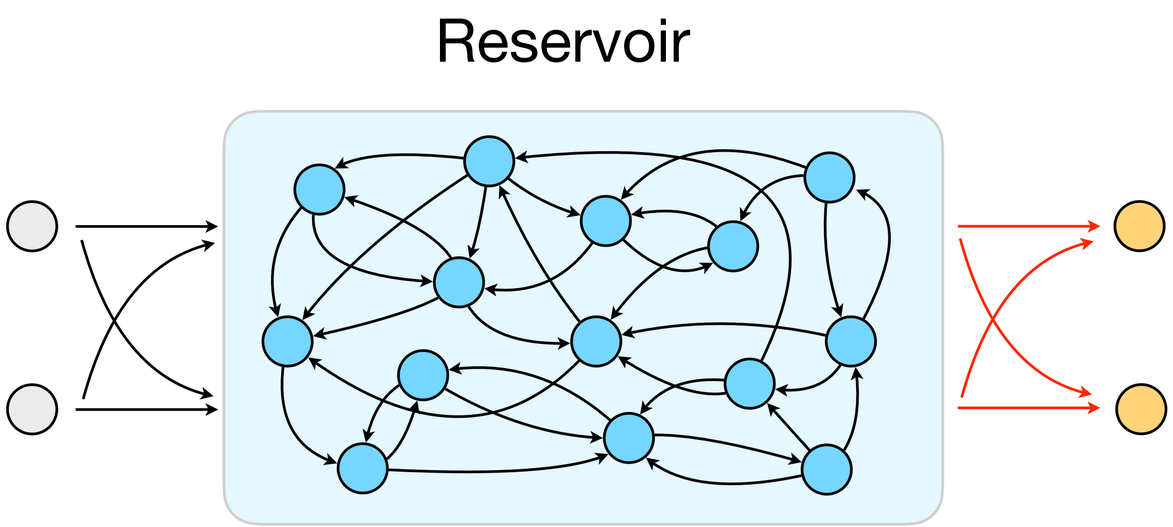
- Open in Colab 05_minimal_esn_lorenz.ipynb
- Open in Colab 06_minimal_esn_mackey.ipynb

A unique application of reservoir computing

Implementation with physical materials

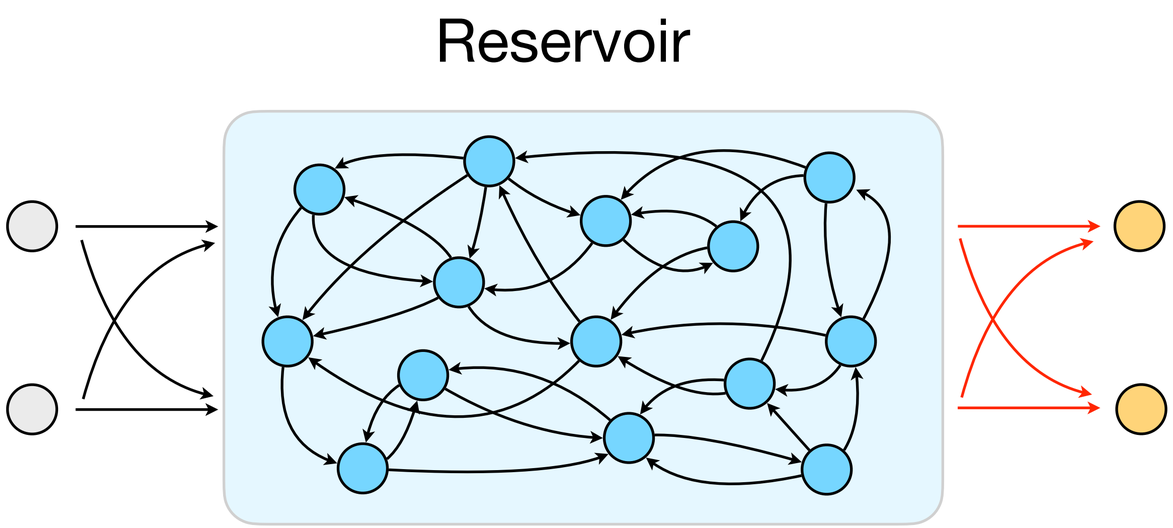
“Training” is not necessary for RC

- “Untrained” dynamical system may be used as a reservoir (= computational resource)



“Training” is not necessary for RC

- “Reservoir” is not necessarily be in a computer



Embodiment

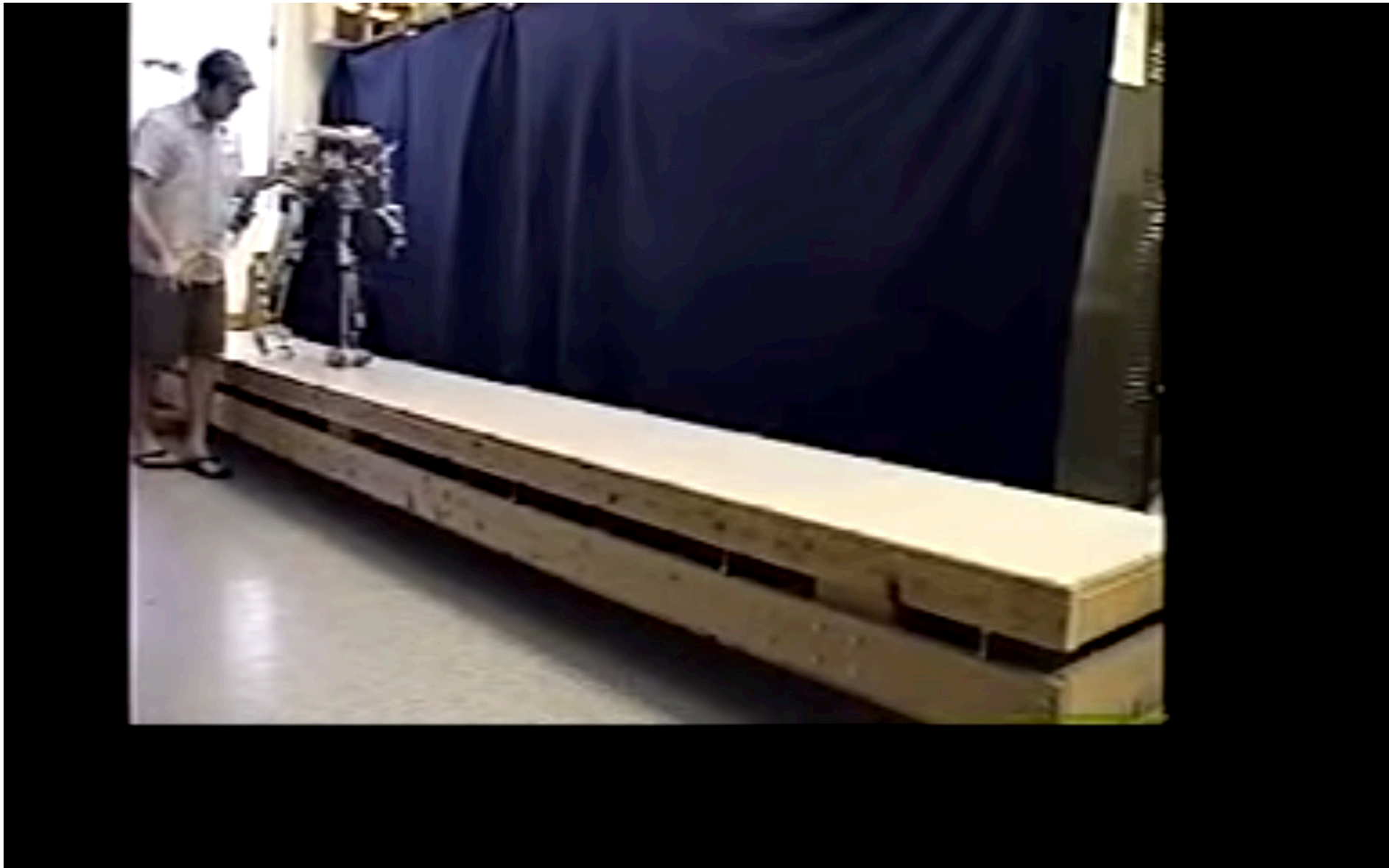
https://www.youtube.com/watch?v=_ZBWnhzYvts

Information processing without a brain: Dead fish can swim

Embodiment

https://www.youtube.com/watch?v=e2Q2Lx8O6Cg

Information processing without a brain: A physical body walks smoothly



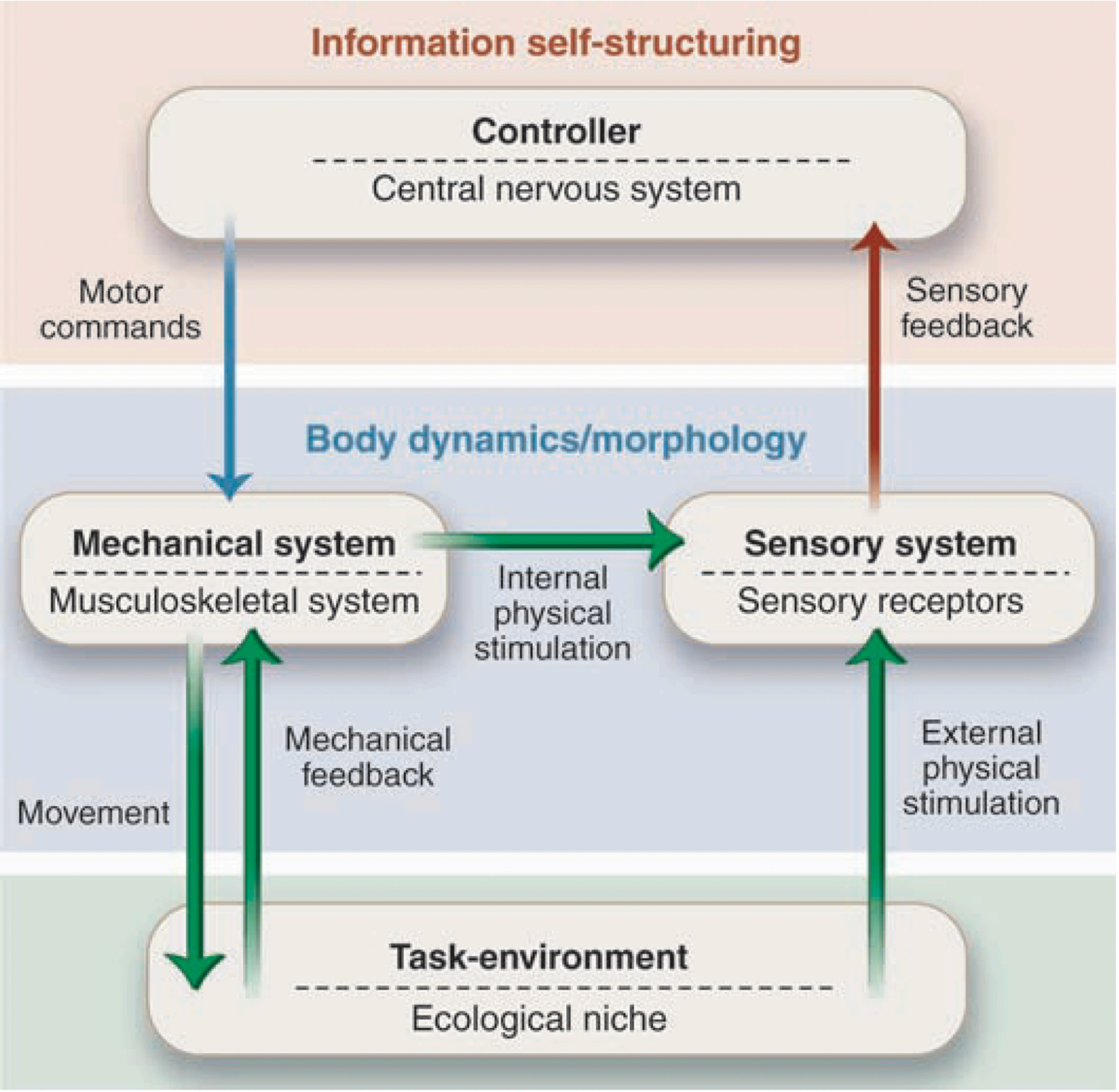
Embodiment

Brain

Body

Body can respond external forces **without** any command from a controller

Environment

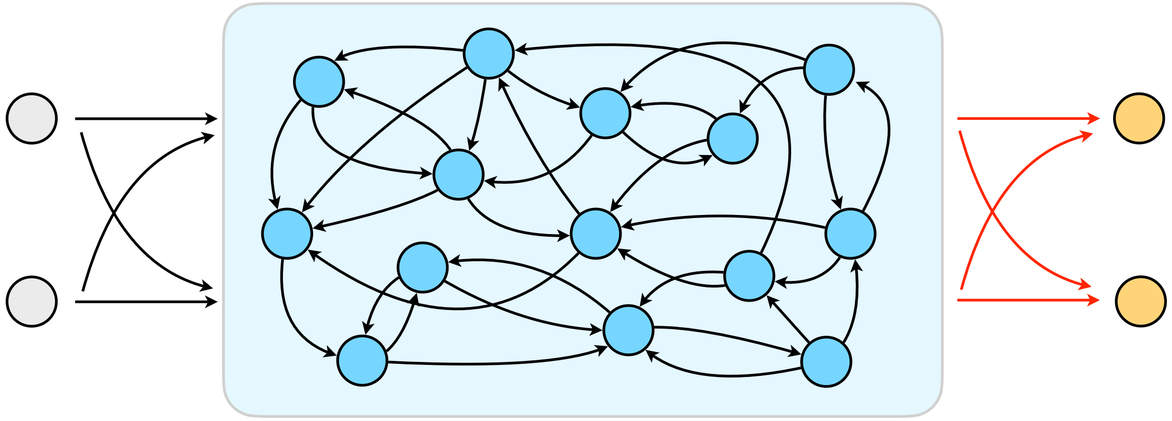


Pfeifer et al. (2007) *Science*

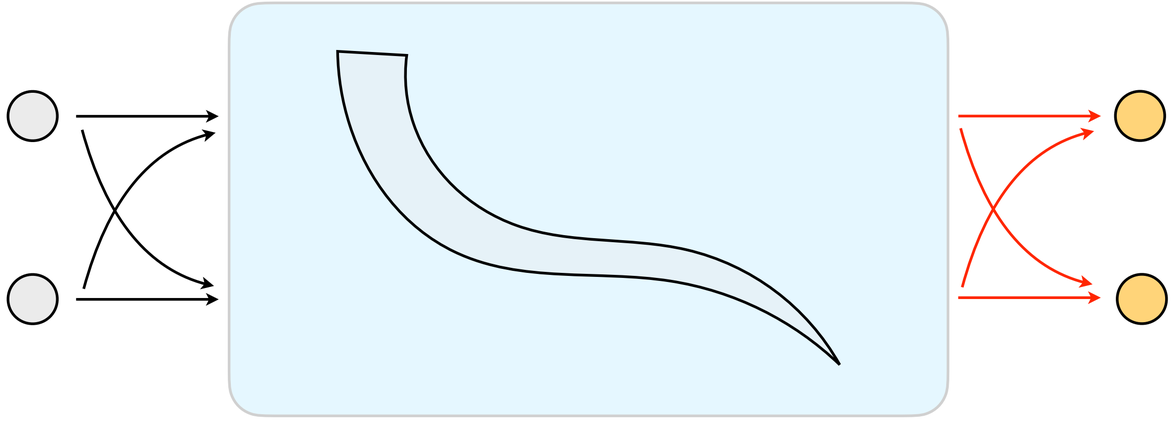
Physical materials can be a “reservoir”?

- “Reservoir” is not necessarily be in a computer
- Physical materials can process information (“embodiment”)
- Physical material can be a “reservoir” in terms of RC?

Reservoir



Physical reservoir?



Physical reservoir computing: An example

- Nakajima et al. (2015) *Scientific Reports*
- Tentacle-like silicon arm (like octopus)

SCIENTIFIC REPORTS

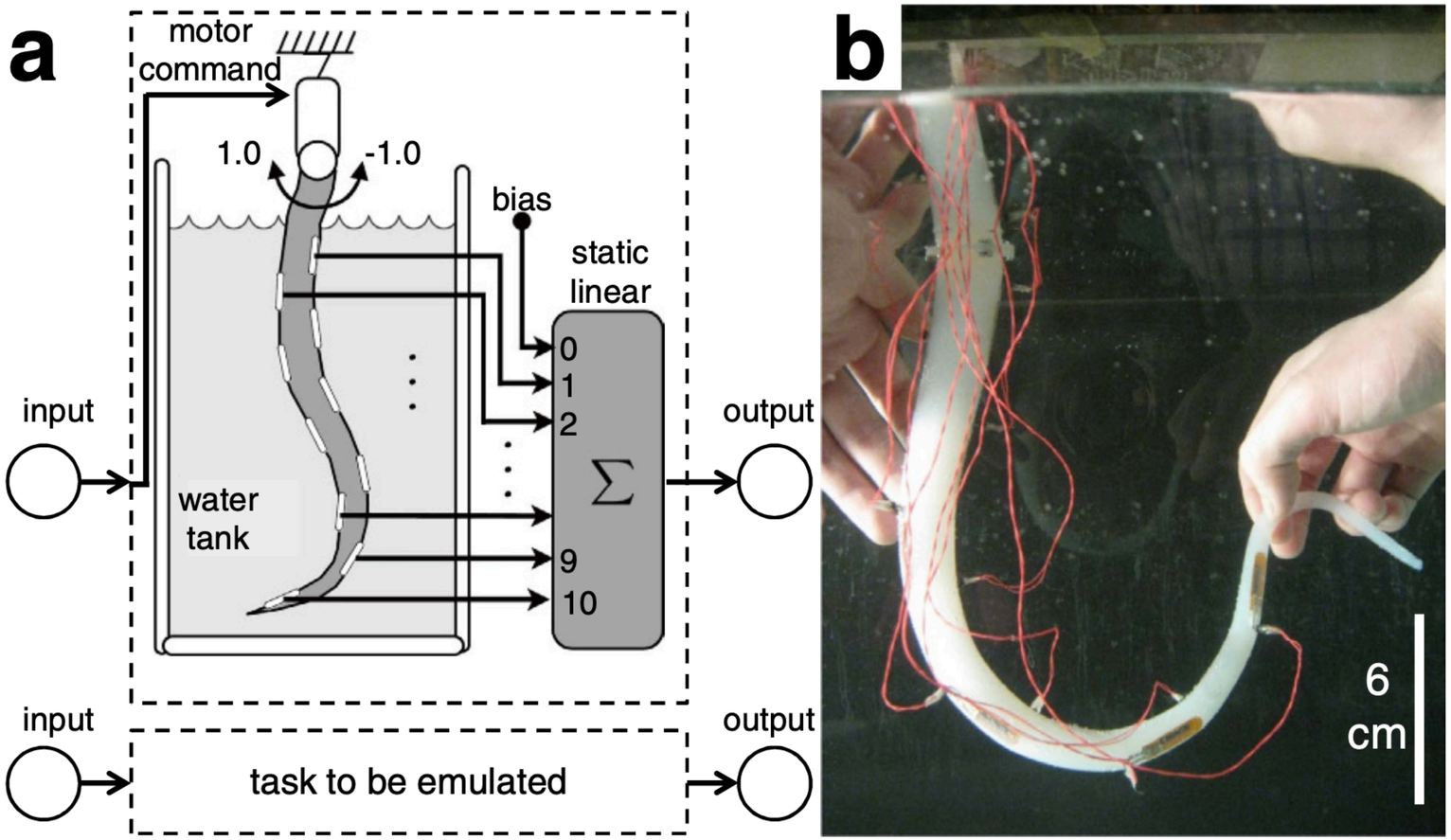
OPEN

Information processing via physical soft body

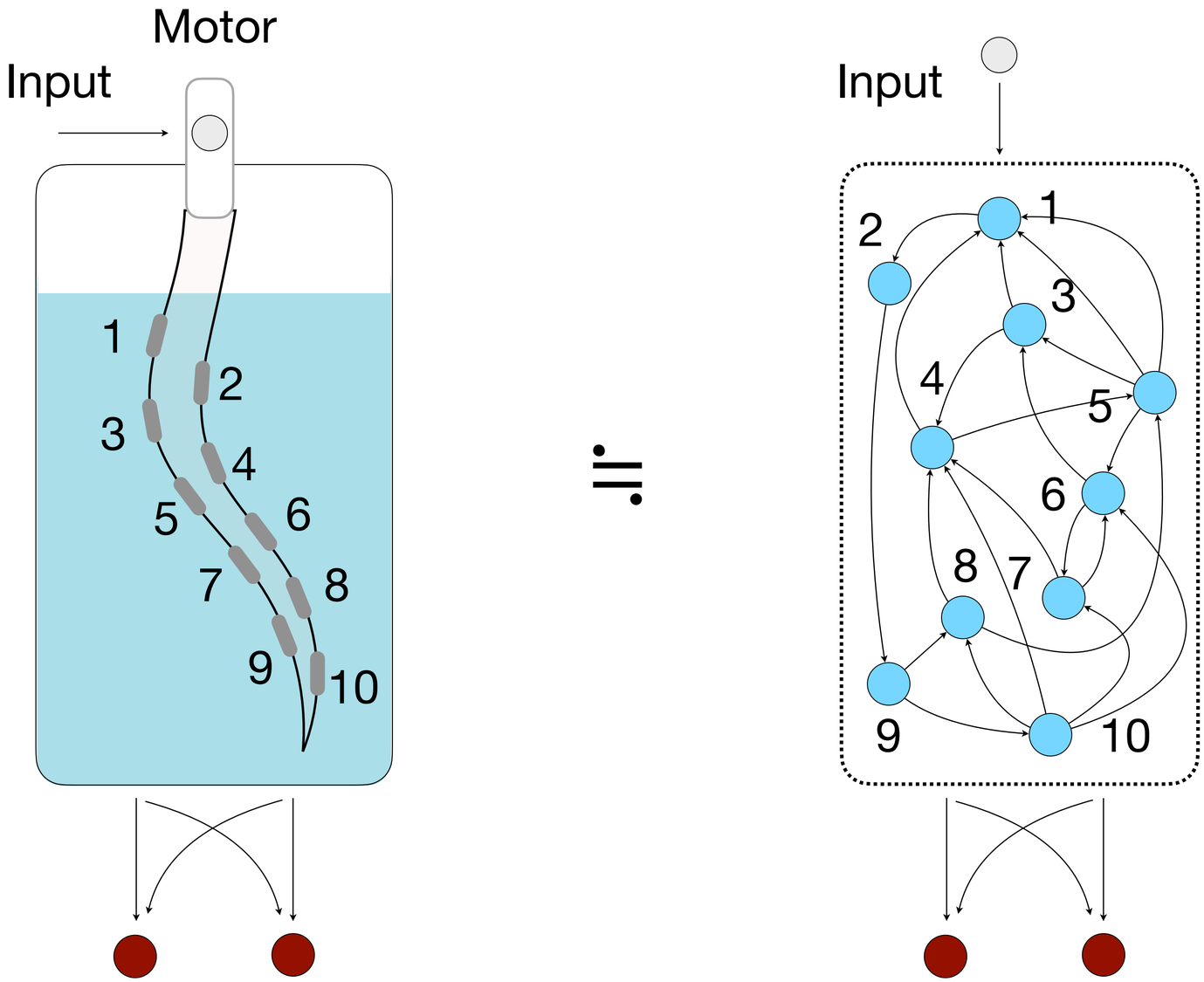
Kohei Nakajima[1], Helmut Hauser[2], Tao Li[3] & Rolf Pfeifer[4,5]

Physical reservoir computing: An example

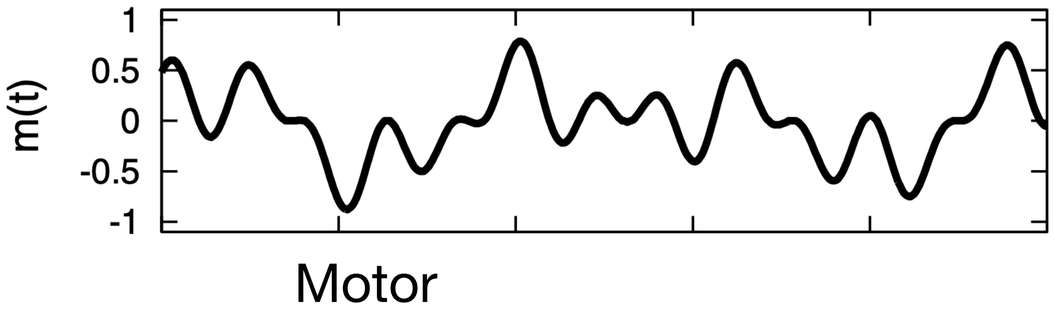
- Tentacle-like silicon arm (like octopus)

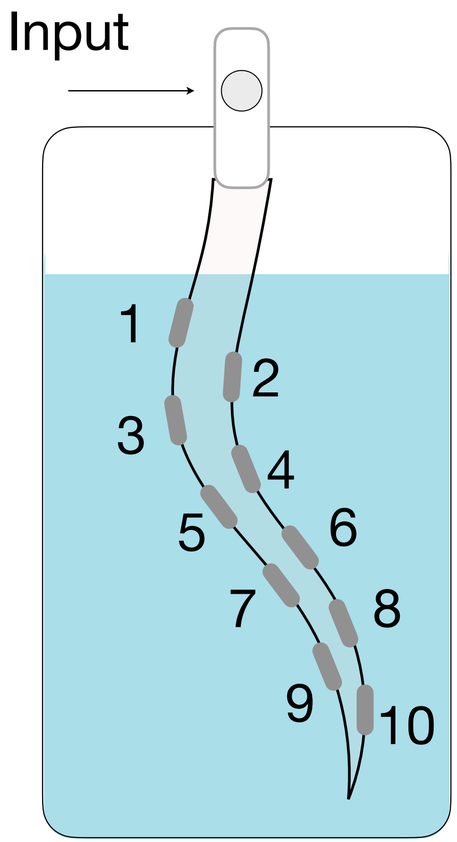


Physical reservoir computing: An example

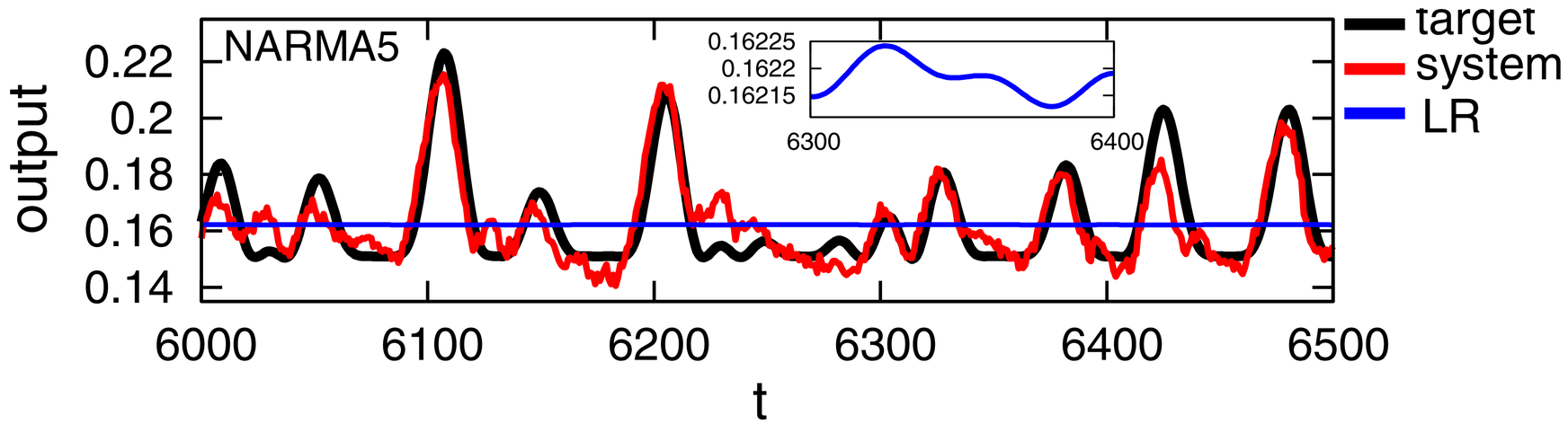


Task: Emulation of nonlinear time series

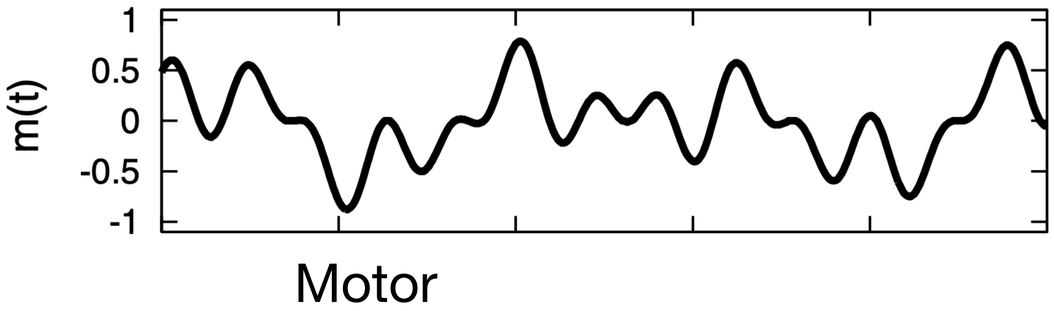


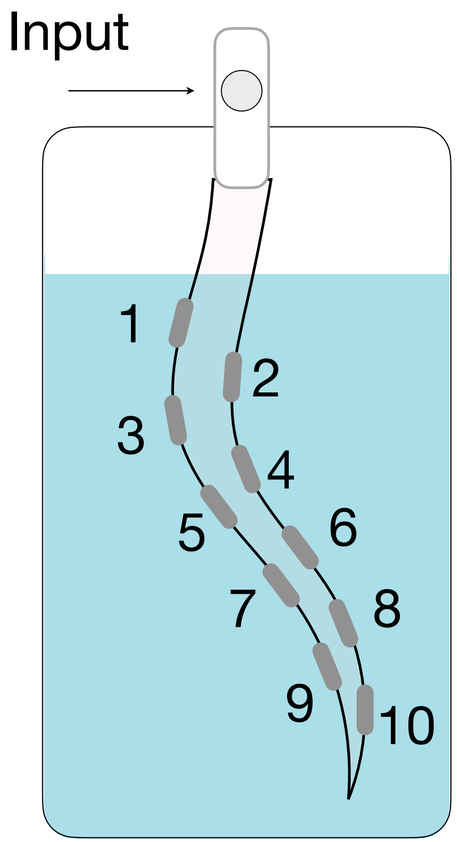


$$y(t+1) = \alpha y(t) + \beta y(t)\left(\sum_{j=0}^{n-1} y(t-j)\right) + \gamma I(t-n+1) I(t) + \delta$$



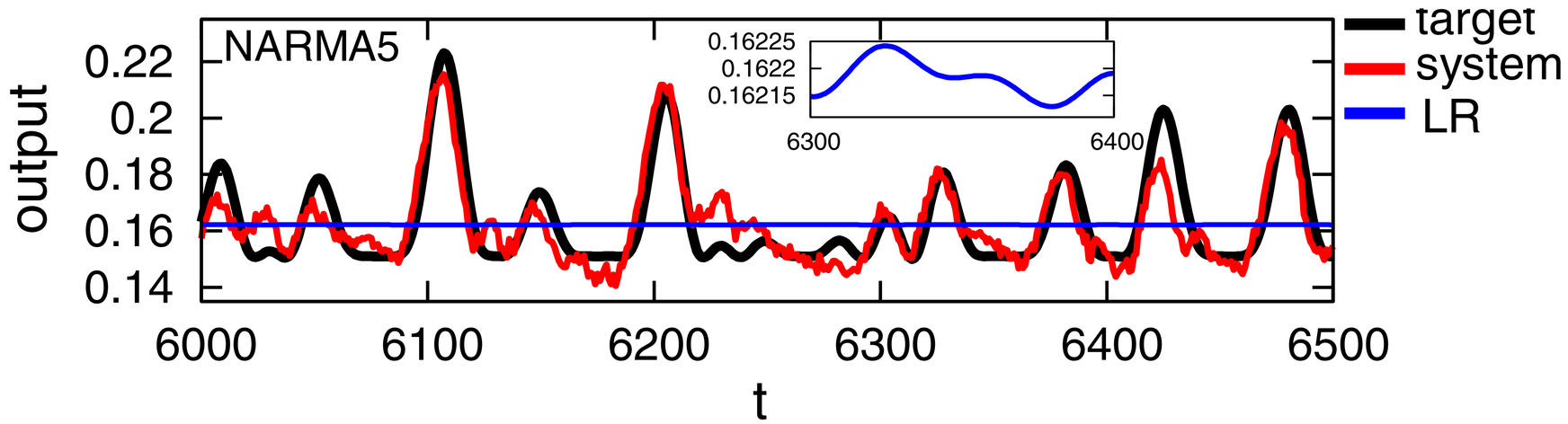
Task: Emulation of nonlinear time series



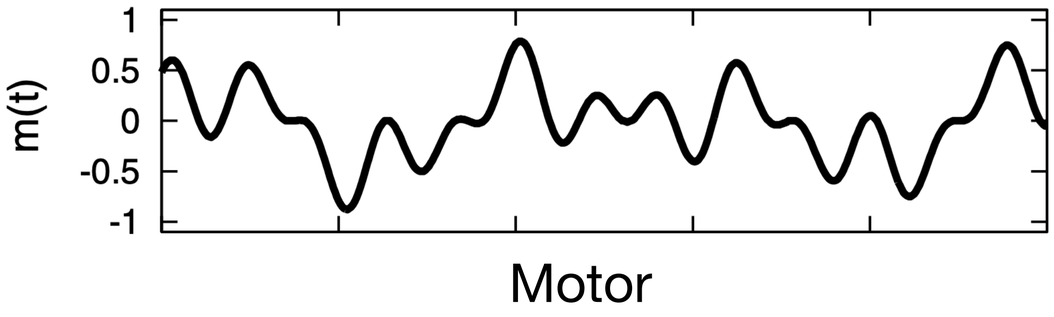


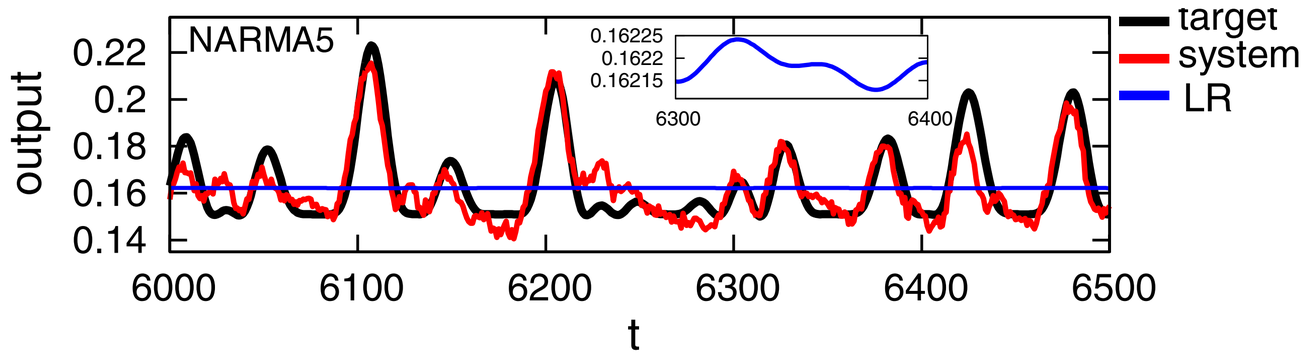
$$y(t+1) = \alpha y(t) + \beta y(t)\left(\sum_{j=0}^{n-1} y(t-j)\right) + \gamma I(t-n+1) I(t) + \delta$$

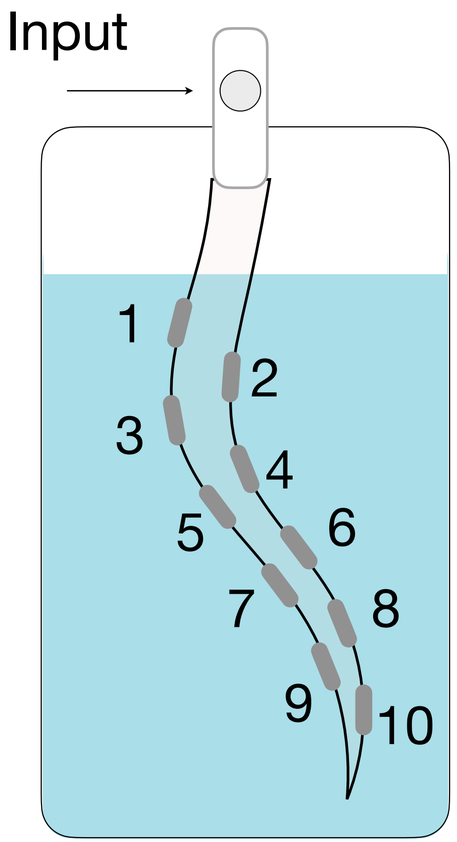
Does the soft silicon arm "emulate" the equation?



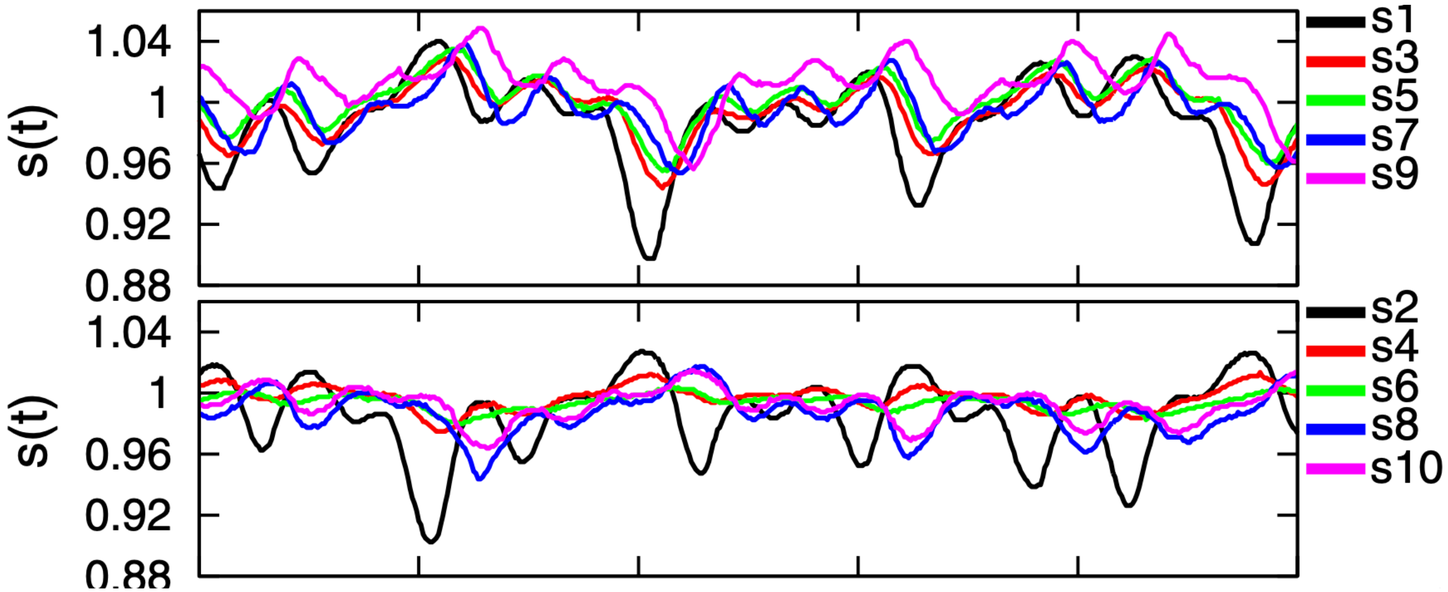
Task: Emulation of nonlinear time series



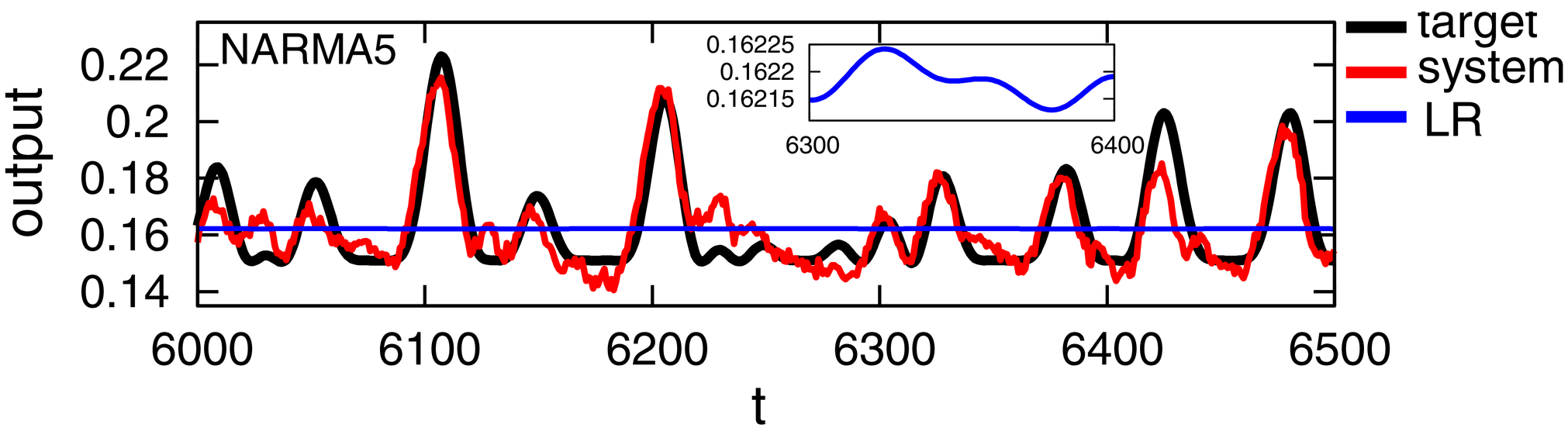




Ridge regression



Task: Emulation of nonlinear time series



Other examples

1. Incubated neural cells (Dockendorf et al. 2009)
2. Physical body of a fish-like robot (Horii et al. 2021)
3. Wing of a flying robot (Tanaka et al. 2021)
4. Origami (Bhovad & Li 2021)
5. Water surface dynamics (Fernando & Sojekka 2003)

Mini-summary

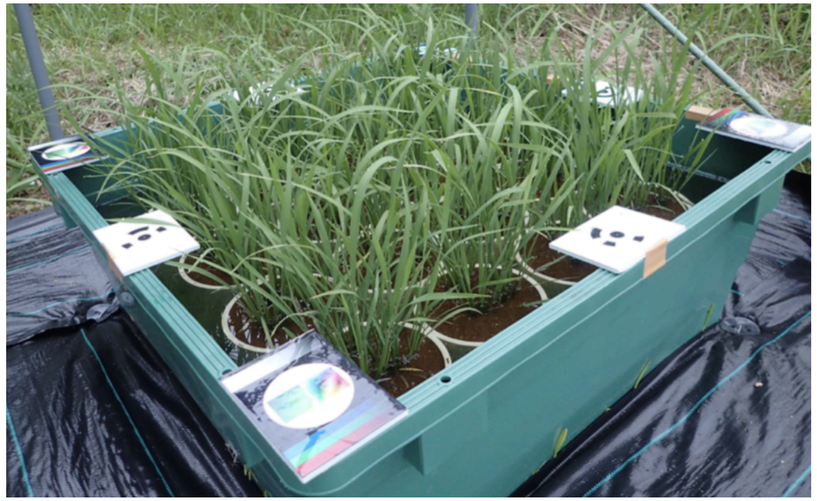
1. Reservoir computing (RC) can be implemented using physical materials
2. Physical reservoir computing (PRC) is a new computational method
3. PRC is energy efficient and high-speed computing method (but still deep learning performs better??)

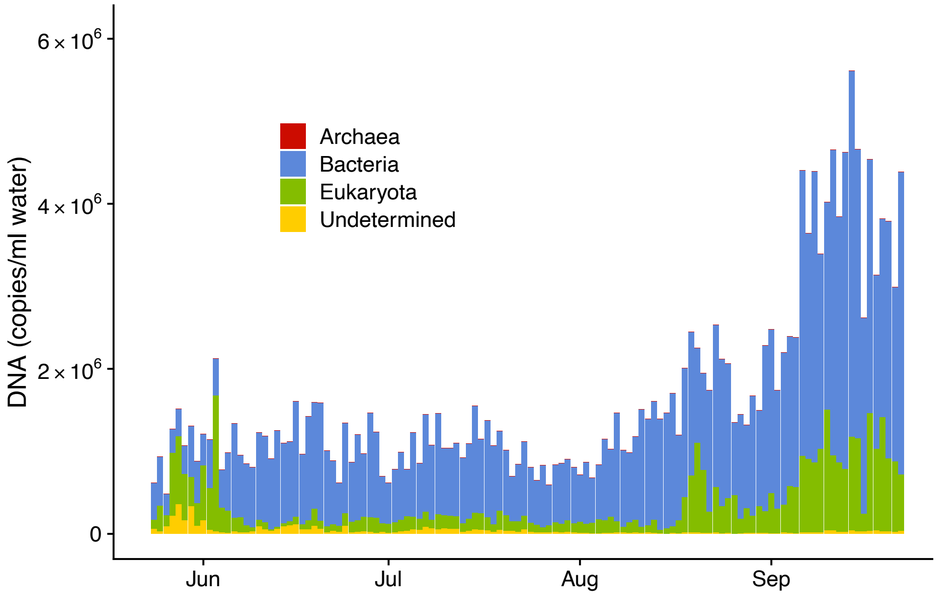
Bonus

Ecological reservoir computing

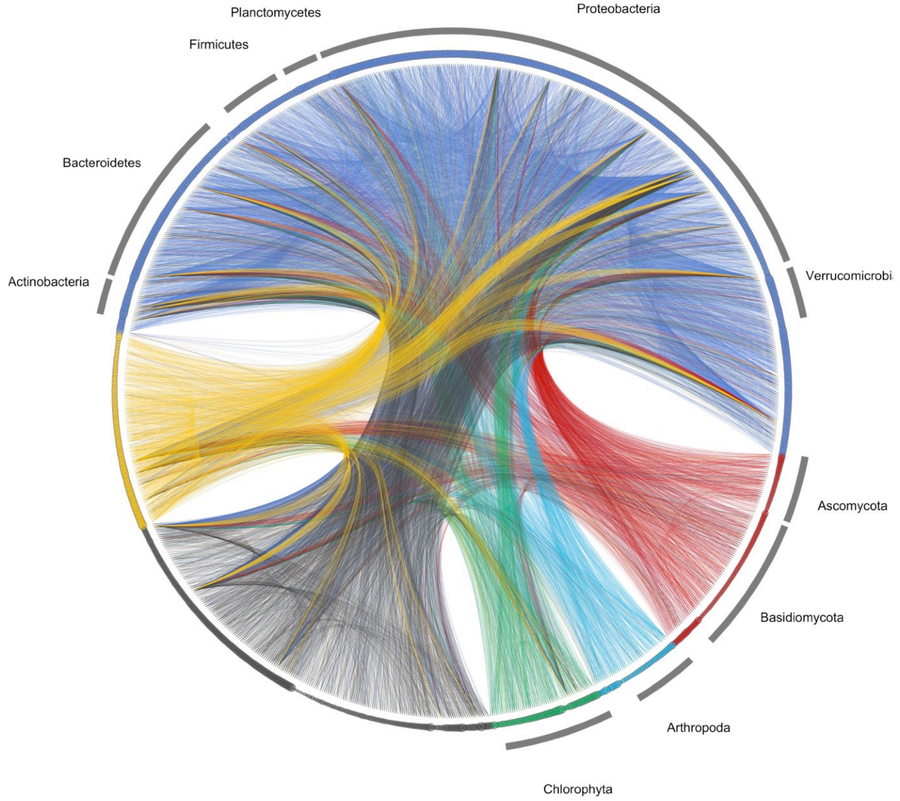
Interspecific interactions in nature

Ecological interaction networks





Ushio (2022)
Proceedings B

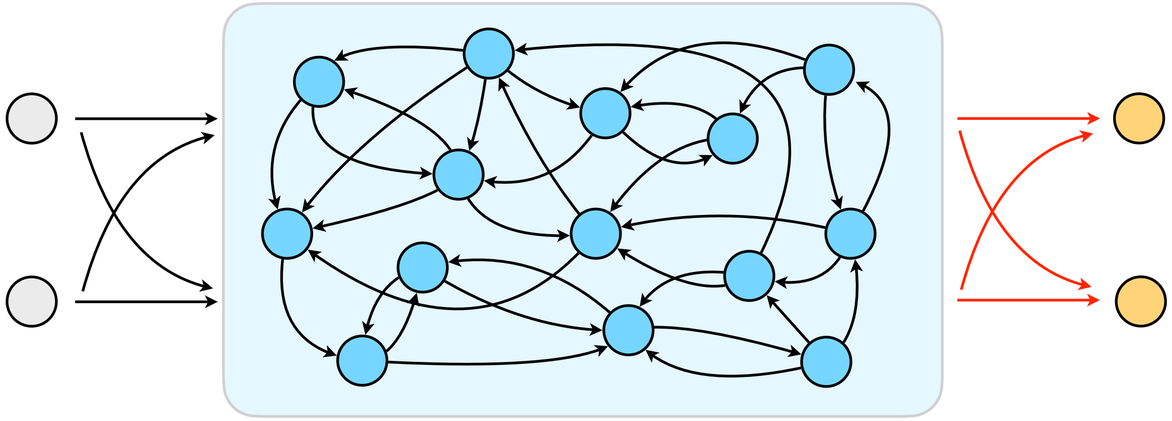


Time series based reconstruction of ecological interaction networks

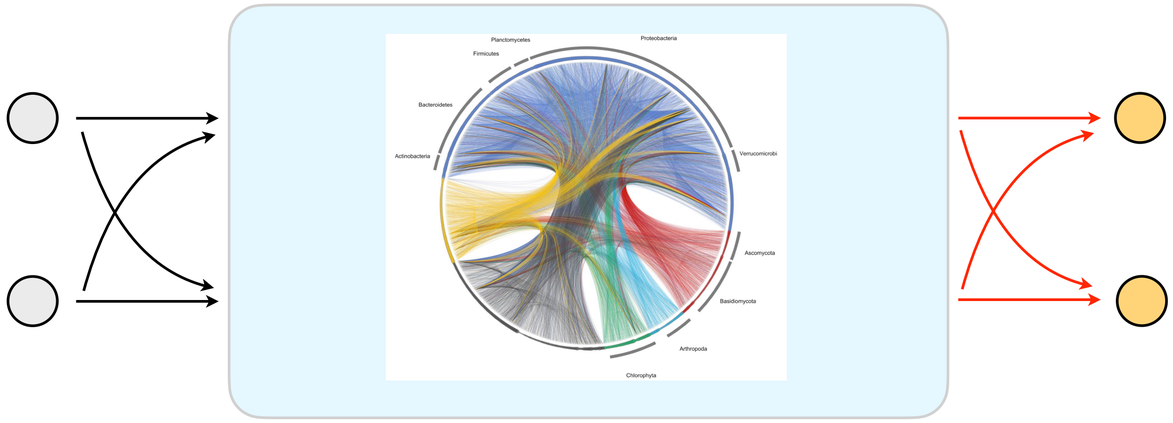
Ecological networks can be a “reservoir”?

- There are rules that drive ecological dynamics (e.g., changes in fish species abundance)
- Can we use the “rules” to transform any input data?

Reservoir



Ecological network



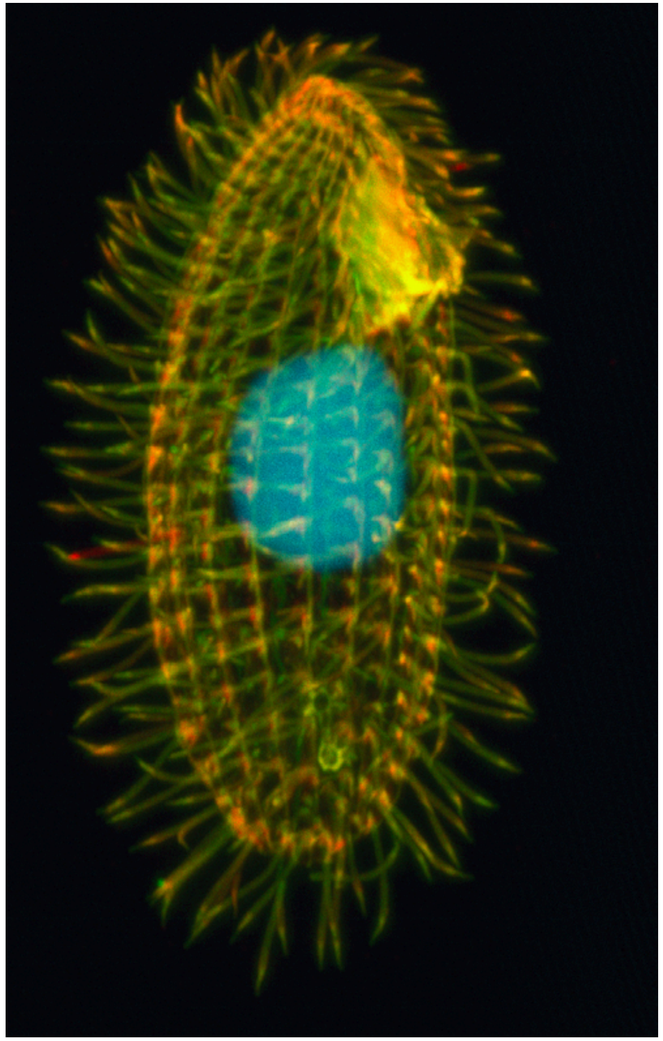
Ecological reservoir computing

Ushio et al. (2021) *bioRxiv* doi:10.1101/2021.09.15.460556

Computational capability of ecological dynamics

Masayuki Ushio[1,*], Kazufumi Watanabe[2], Yasuhiro Fukuda[3], Yuji Tokudome[4], Kohei Nakajima[4,*]

Target organisms: *Tetrahymena thermophila*

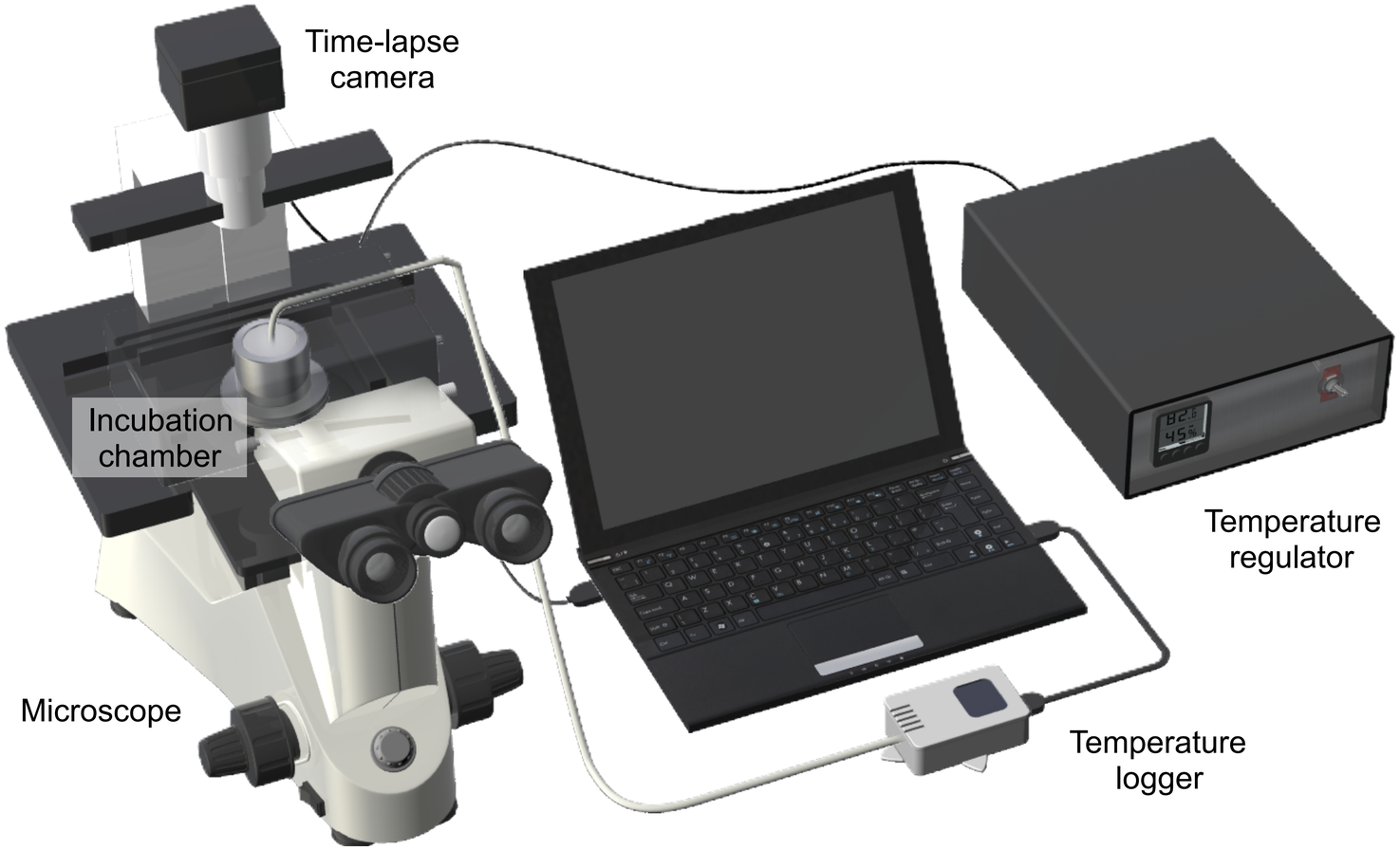


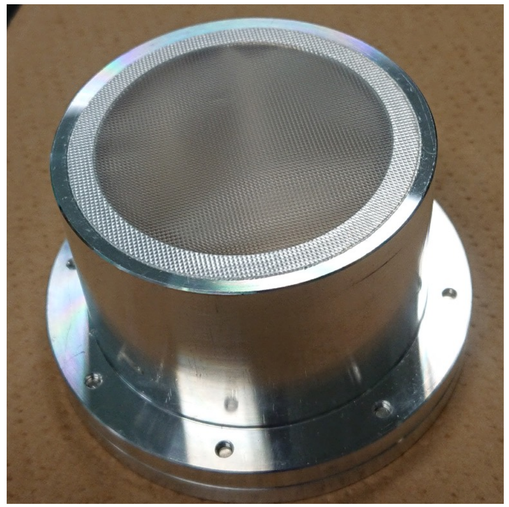
https://en.wikipedia.org/wiki/Tetrahymena

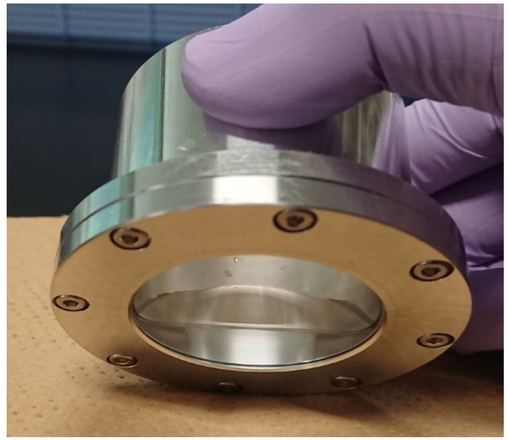
Tetrahymena thermophila

- A unicellular, eukaryotic microbial species
- Ciliates (繊毛虫)
- Cell size: 30–100 μm
- Doubling time: 2 hrs under optimal conditions
- Usually grow on a bacterial diet, but can easily be cultured on inanimate media
- Can be genetically modified

Target organisms: *Tetrahymena thermophila*



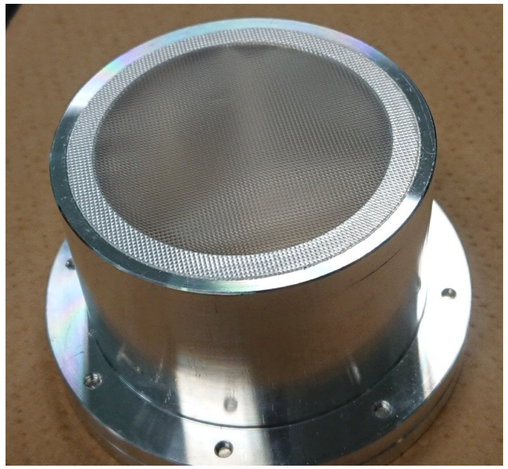


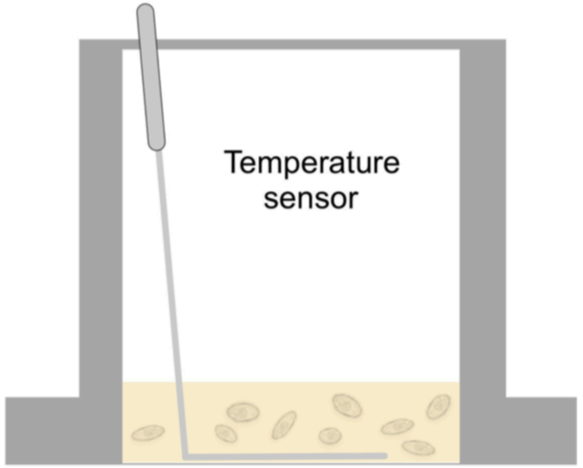


An experimental setting for monitoring *Tetrahymena* dynamics

(Time-lapse imaging + microscope + thermo-sensor + temperature regulator)

Target organisms: *Tetrahymena thermophila*

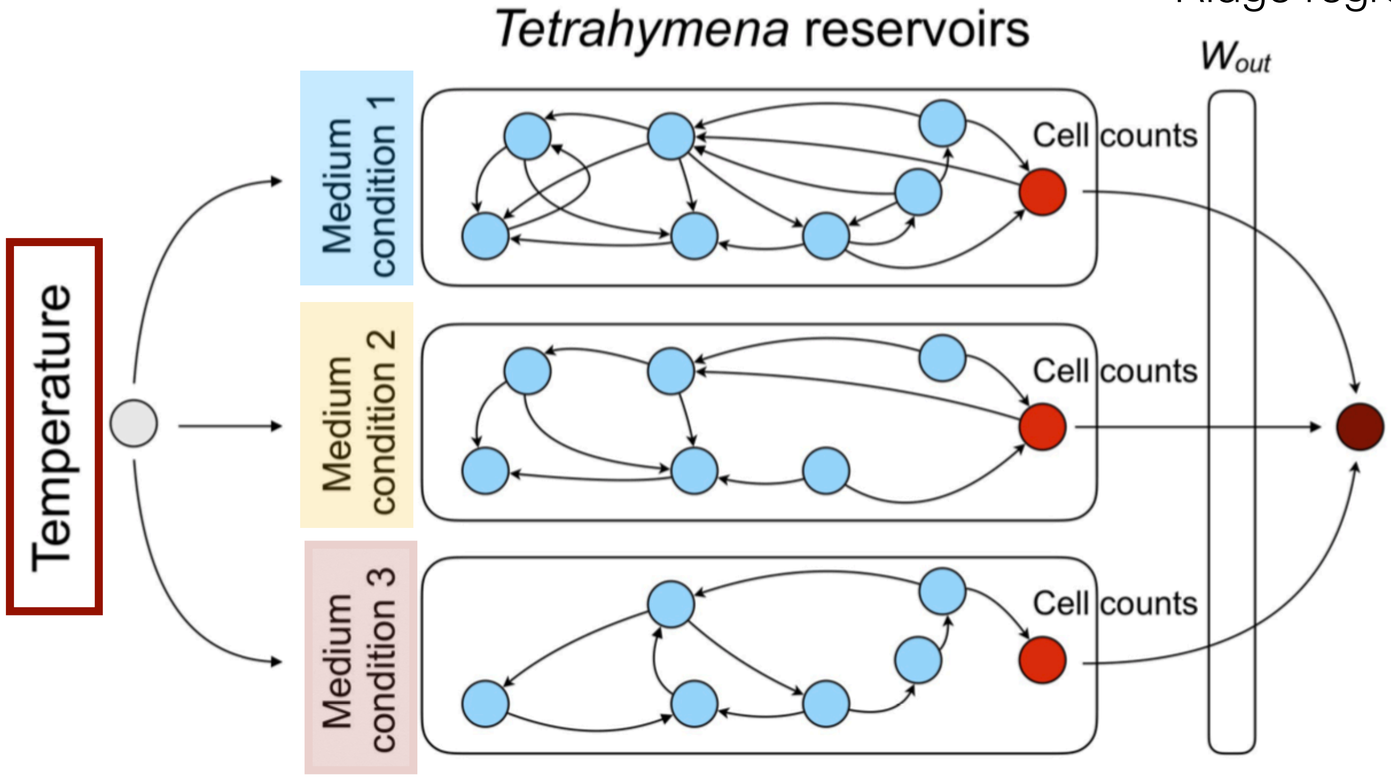




One input/5 mins
(256 temp)
One image/min
(1280 images)

Mapping inputs to feature space by *Tetrahymena* population dynamics

Ridge regression



Surface cell counts
Cell activity
Growth rate
Behavior

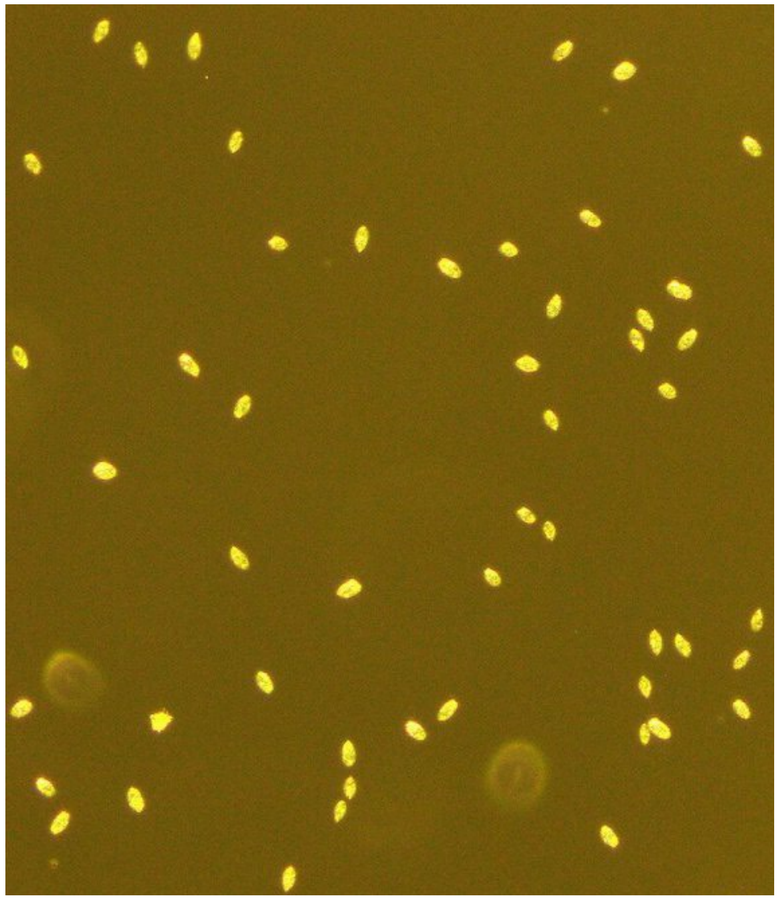
Bottom cell counts
…

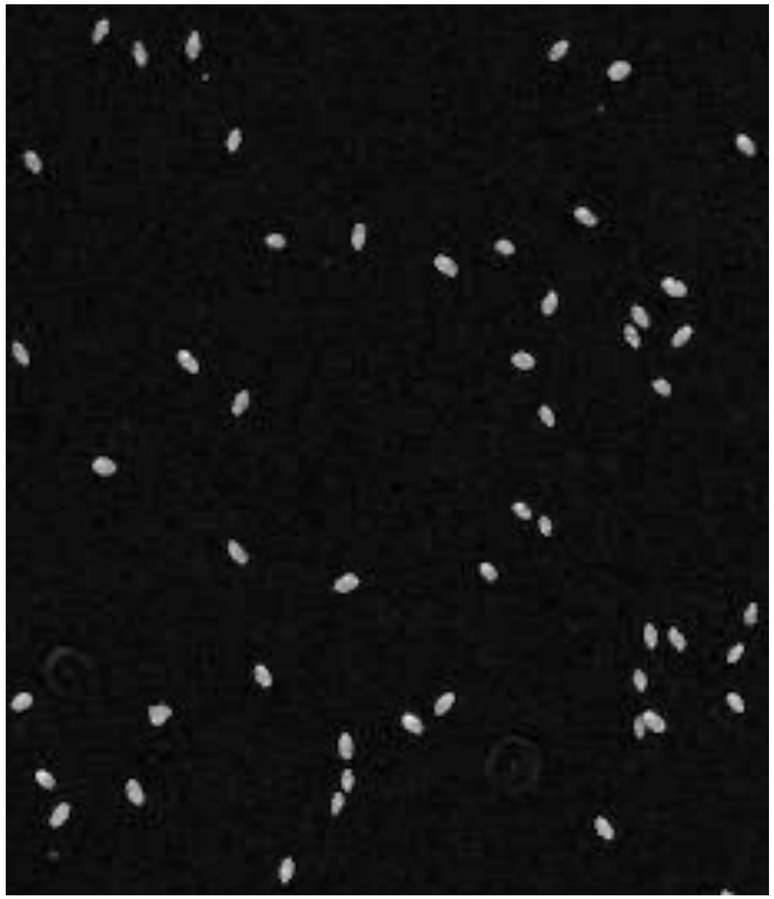
Target organisms: *Tetrahymena thermophila*

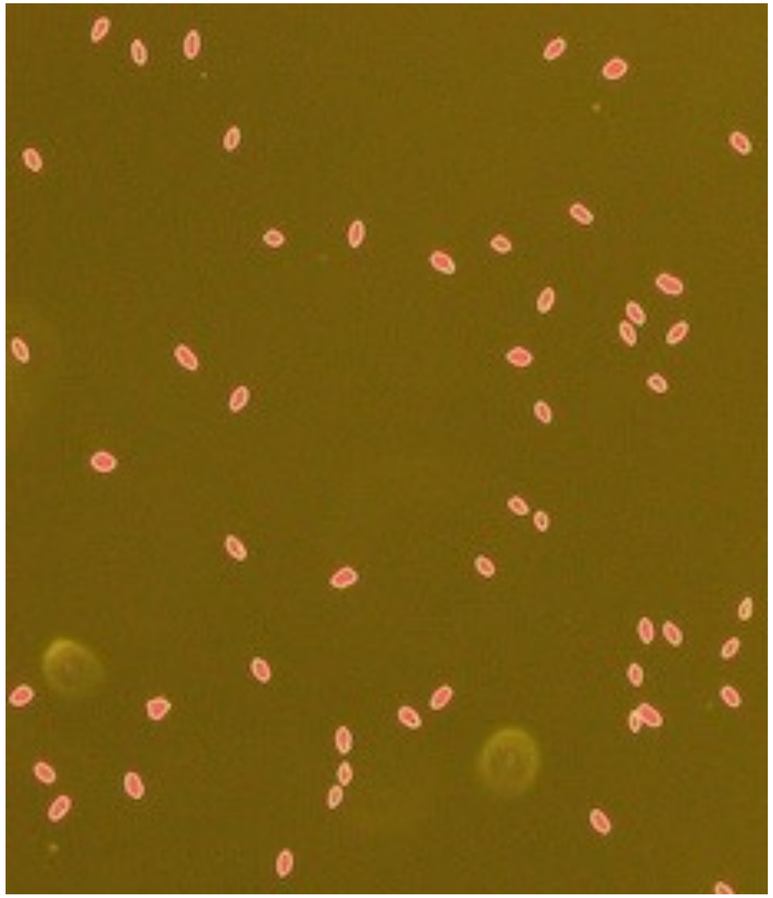
python + OpenCV

Binalization
Detection of boundary
Detection of maxima

- Cell counts
- (Cell area)
- (Cell location)
- (Cell-cell distance)



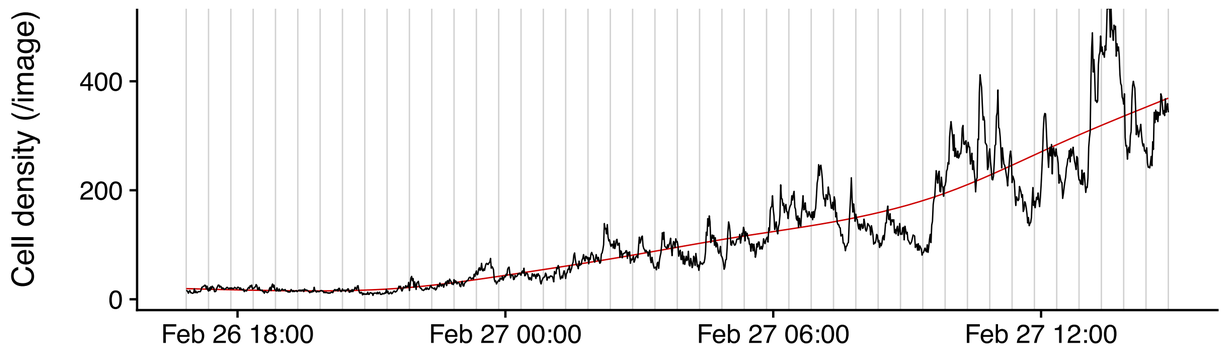




Target organisms: *Tetrahymena thermophila*

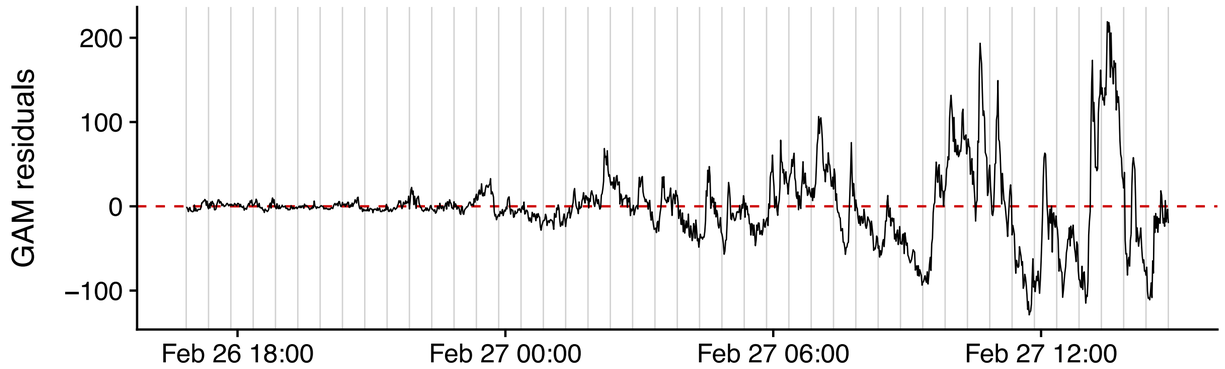
Original data

One image/min
(1280 images)
One input/5 mins
(256 temp)



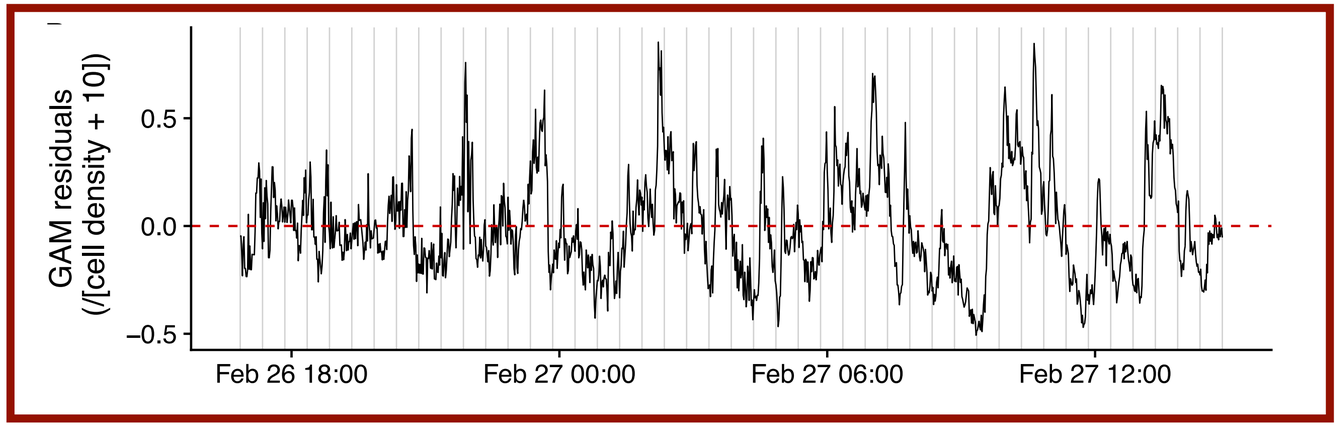
Estimation of trend

Residual data



Raw data – trend

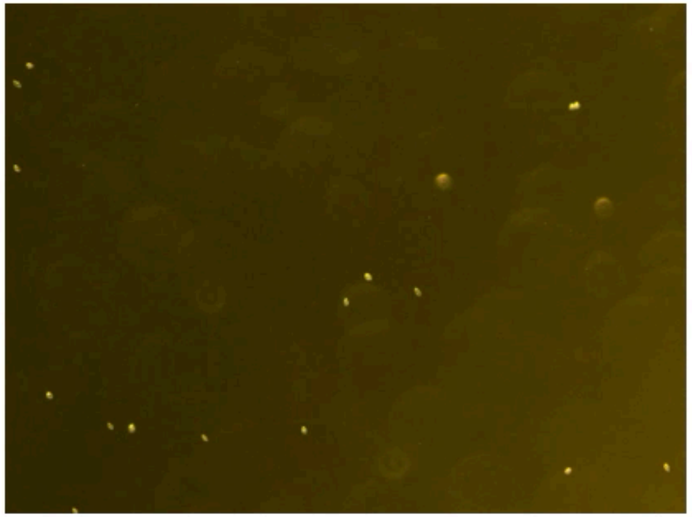
Relative residual data



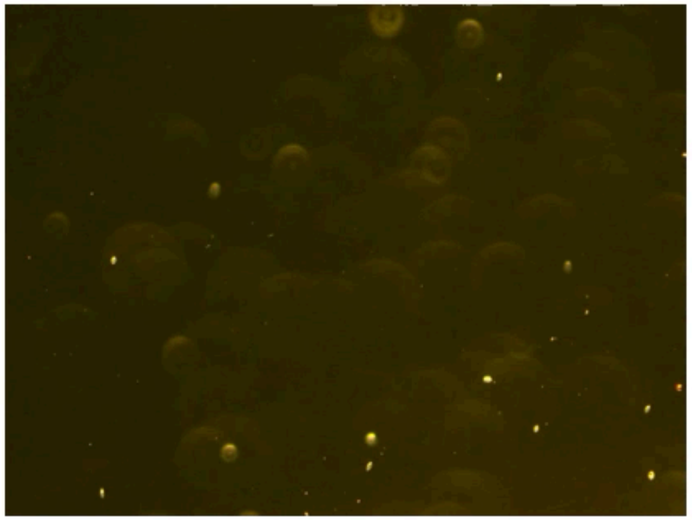
$$\frac{\text{Raw data} - \text{trend}}{\text{Trend}}$$

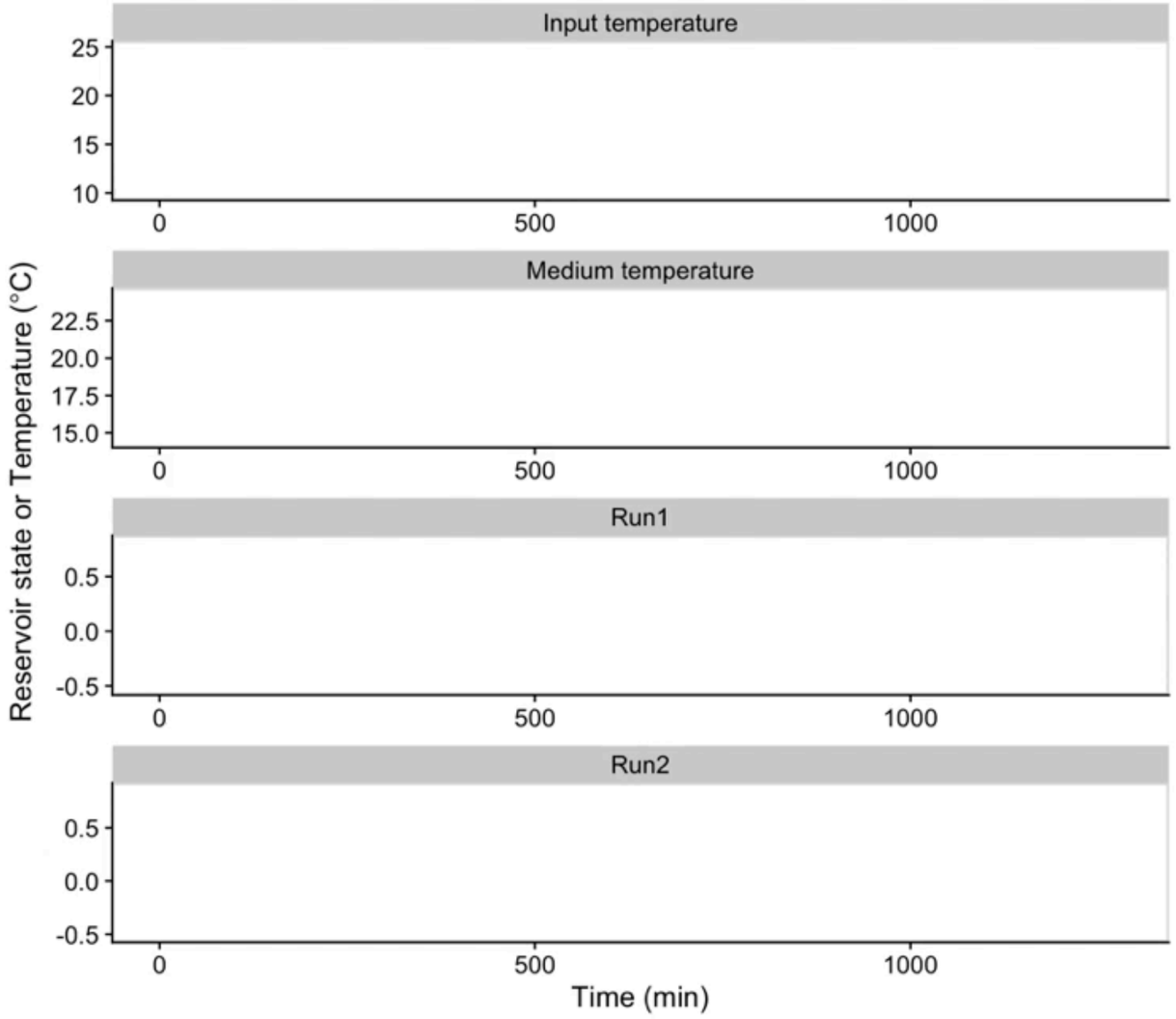
Examples of population dynamics

Medium nutrient concentration - Run1



Medium nutrient concentration - Run2





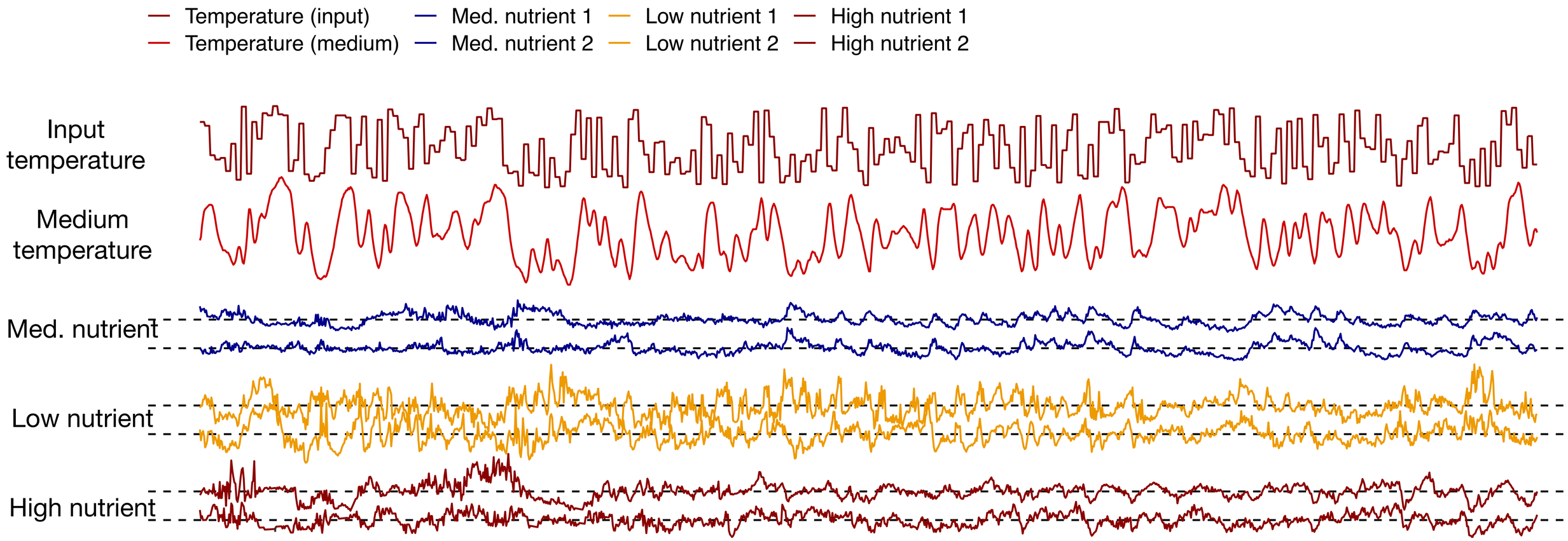
YouTube

https://youtu.be/z_QeEka4W3w

figshare

doi:10.6084/m9.figshare.16608802

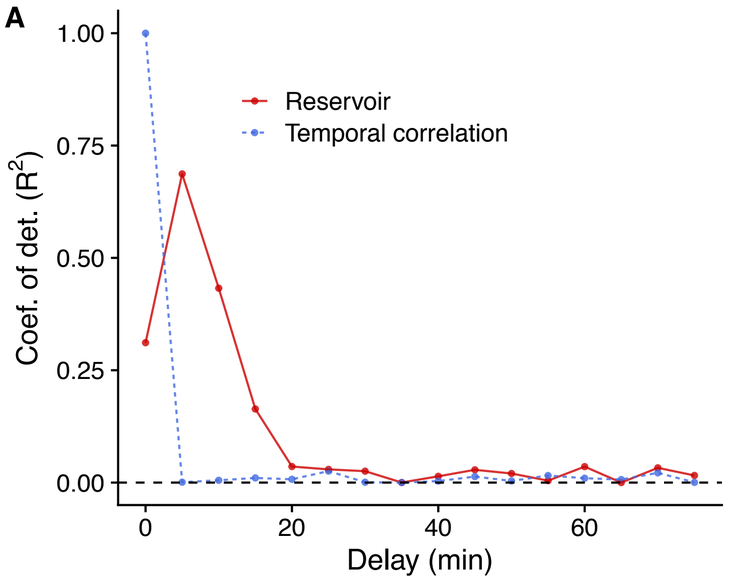
Examples of population dynamics

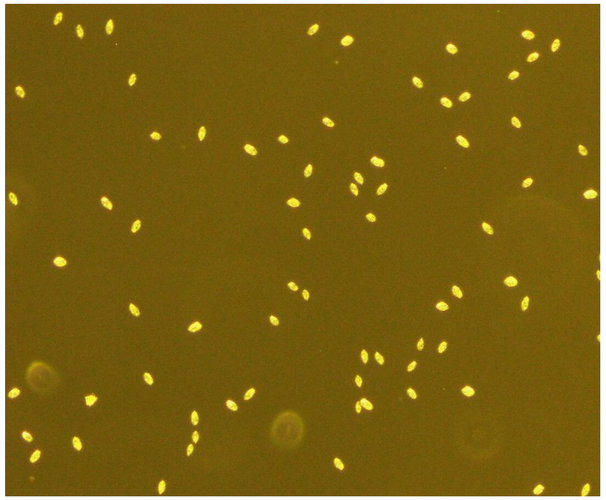


The reservoir responds to input temperatures accurately (nutrient-dependent response)

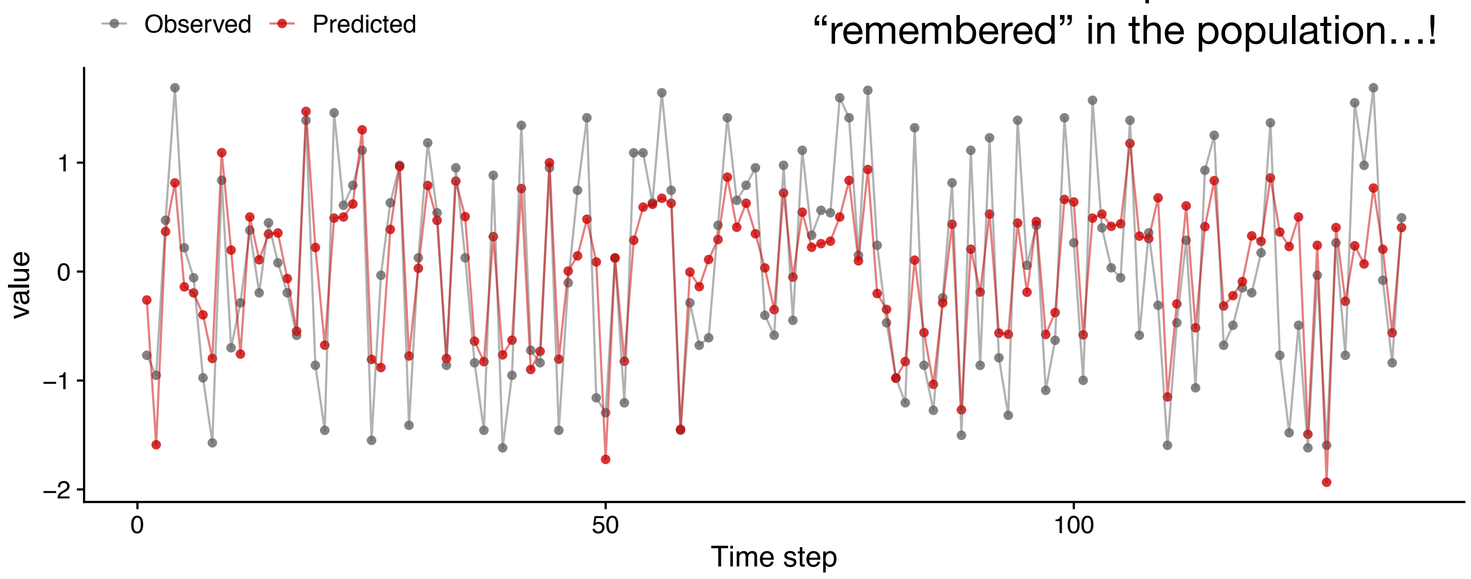
→ We can predict (recover) the input temperatures from the reservoir states?

Examples of population dynamics





Medium temperature is "remembered" in the population…!

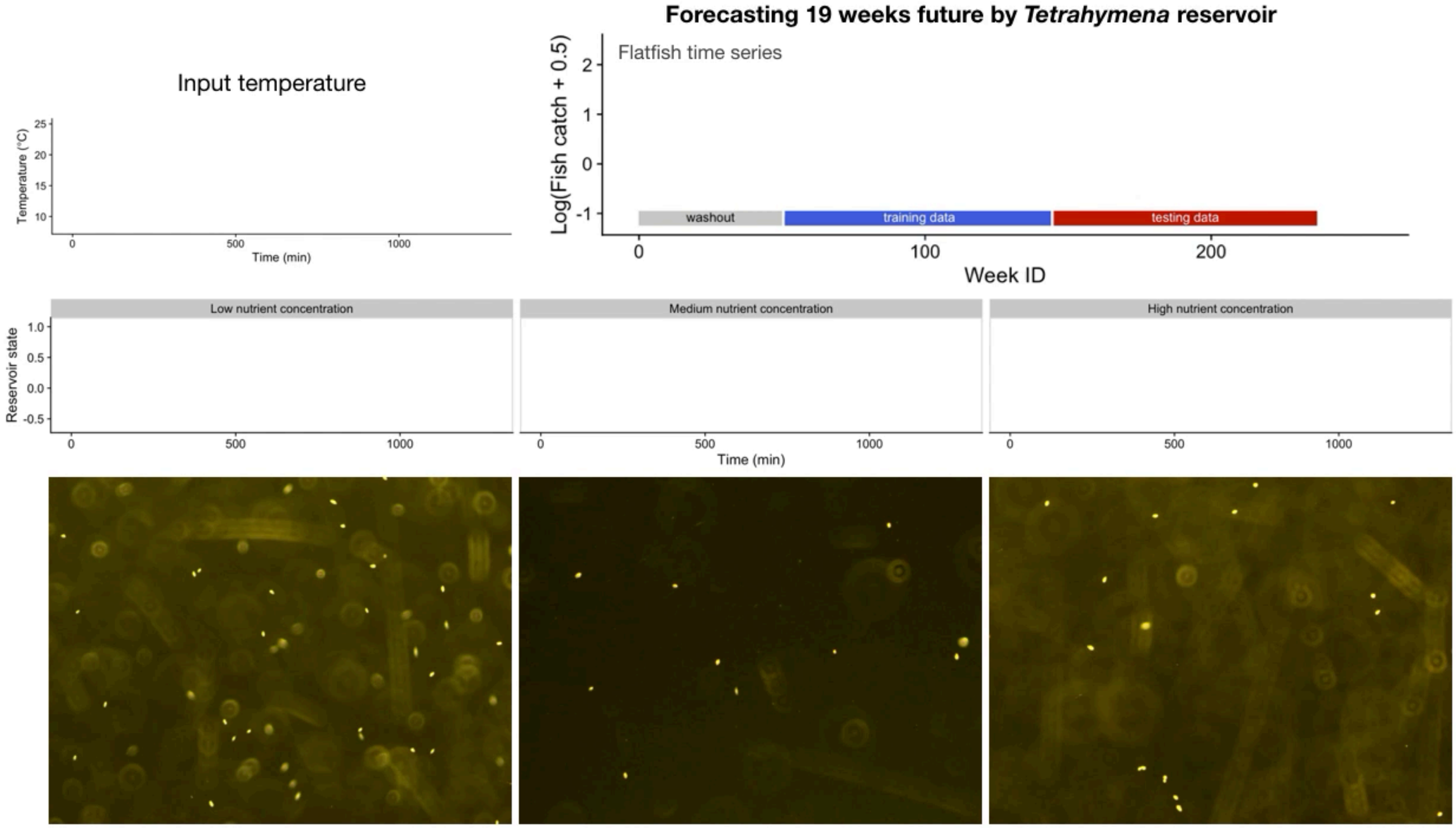


The population (not individuals) can remember 5–15 minutes ago!

Can this dynamics be used for time series analysis??

Ushio et al. (2021) *bioRxiv*

Predictions of near future of another time series by microbes



YouTube

https://youtu.be/SUmkYAnfjFk

figshare

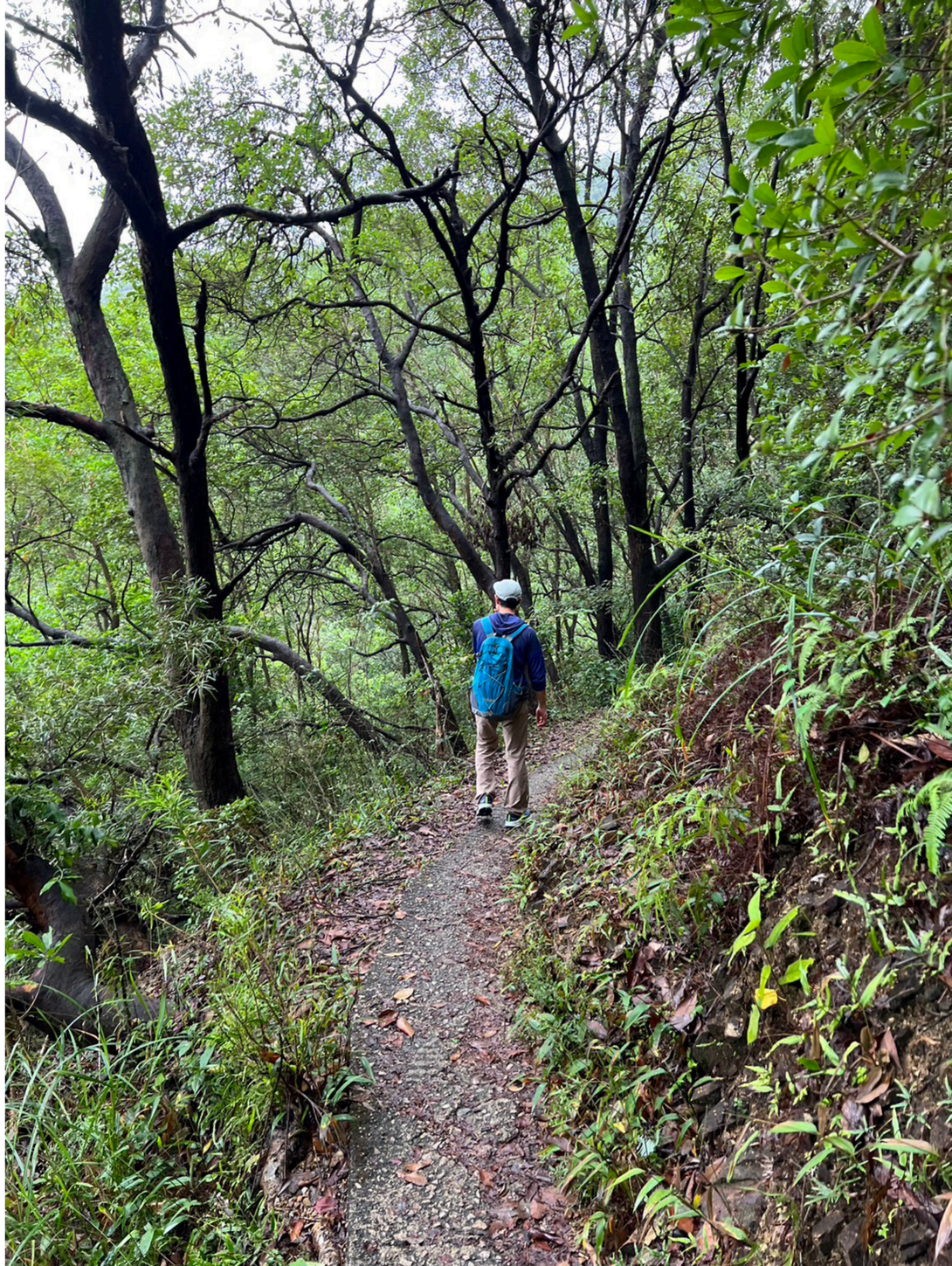
doi:10.6084/9.figshare.16608808

Mini-summary

1. Ecological dynamics might be utilized as a computational resource
2. (At least in our case) Near future forecasting was possible (but the performance is not high at present)
3. Further studies are necessary to understand potentials of ecological reservoir computing

Summary

Reservoir computing
Physical reservoir computing
(Ecological reservoir computing)



Summary

1. Artificial neural networks (ANNs) are powerful method to analysis data
2. There are two types of ANNs: Feedforward and recurrent neural networks
3. Reservoir computing (RC) is a special type of recurrent neural networks
4. RC does not train its network; the training cost is low, but good performance
5. RC implemented using physical material is physical reservoir computing
6. There might be more unique and powerful applications of RC in future