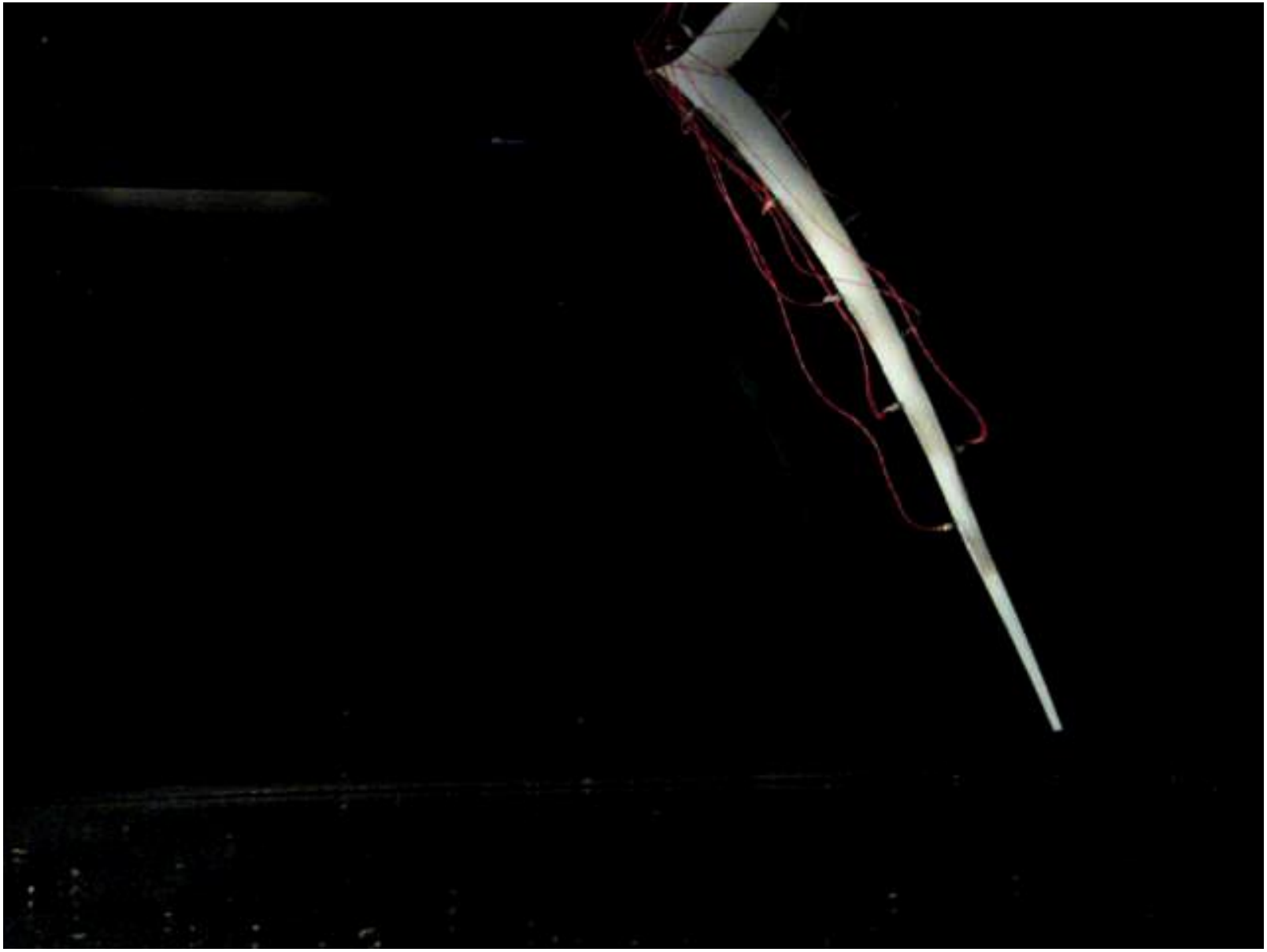
THE UNIVERSITY OF TOKYO

Chaos in reservoir computing

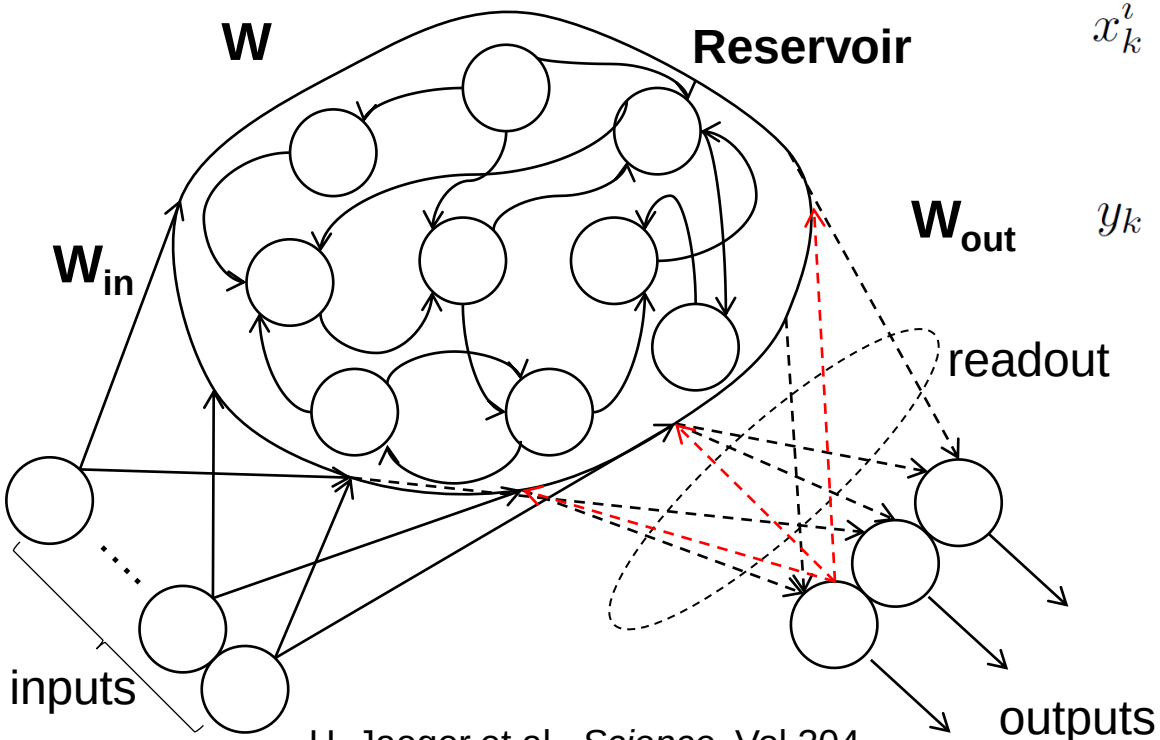
Kohei Nakajima
University of Tokyo

2024/07/02 Tuesday
Dynamics Days Asia Pacific 13,
Japan

Computing with soft body dynamics?



Reservoir computing: basic settings



H. Jaeger et al., *Science*, Vol.304. no.5667, pp.78–80 (2004).

$$x_k^i = f\left(\sum_{j=1}^{M} w^{ij} x_{k-1}^j + w_{in}^i u_k\right)$$

$$y_k = \sum_{i=0}^{M} w_{out}^i x_k^i, \quad f(x) = \tanh(x)$$

Adjust only the readout!

$$W_{out} = \left(X^T X\right)^{-1} X^T y$$

Use W_{out} for information processing!

$$\hat{y} = X W_{out}$$

(Good points)

- Learning is fast and stable!
- Feasible for physical platform.

(Computational power)

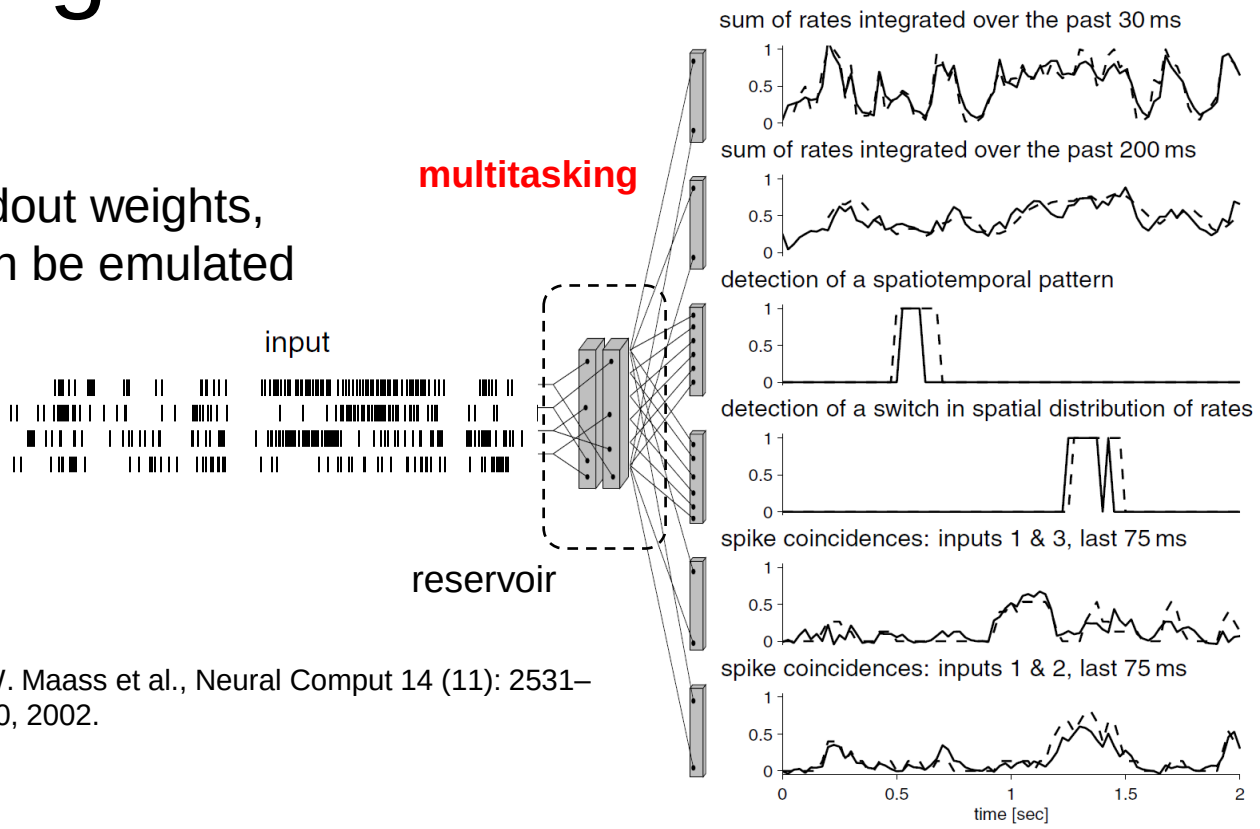
- **Nonlinearity**
- **Memory**

Typical settings

- **Open-loop**

  By attaching the readout weights, multiple functions can be emulated simultaneously!

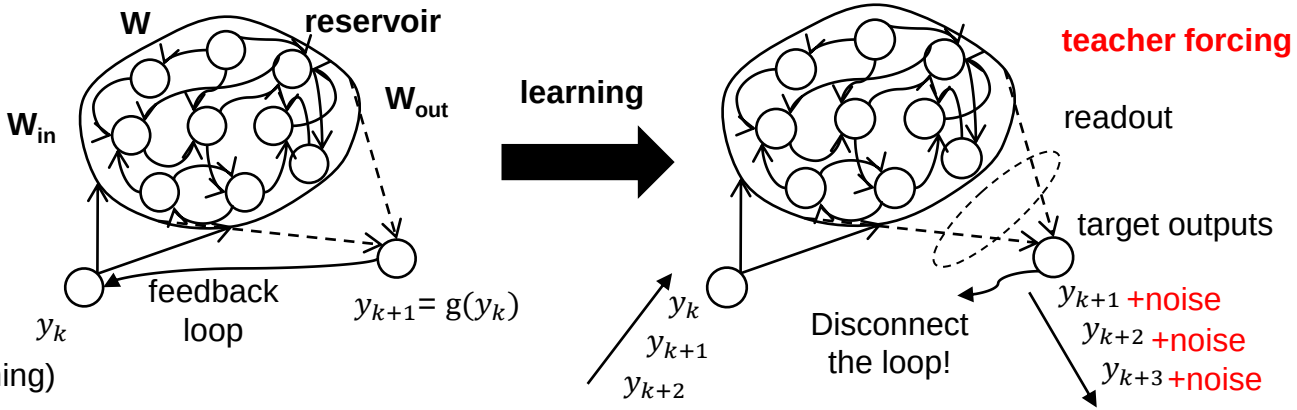
$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$$
$$\hat{\boldsymbol{y}}_k = \hat{\psi}(\boldsymbol{x}_k)$$



W. Maass et al., Neural Comput 14 (11): 2531–60, 2002.

- **Close-loop**

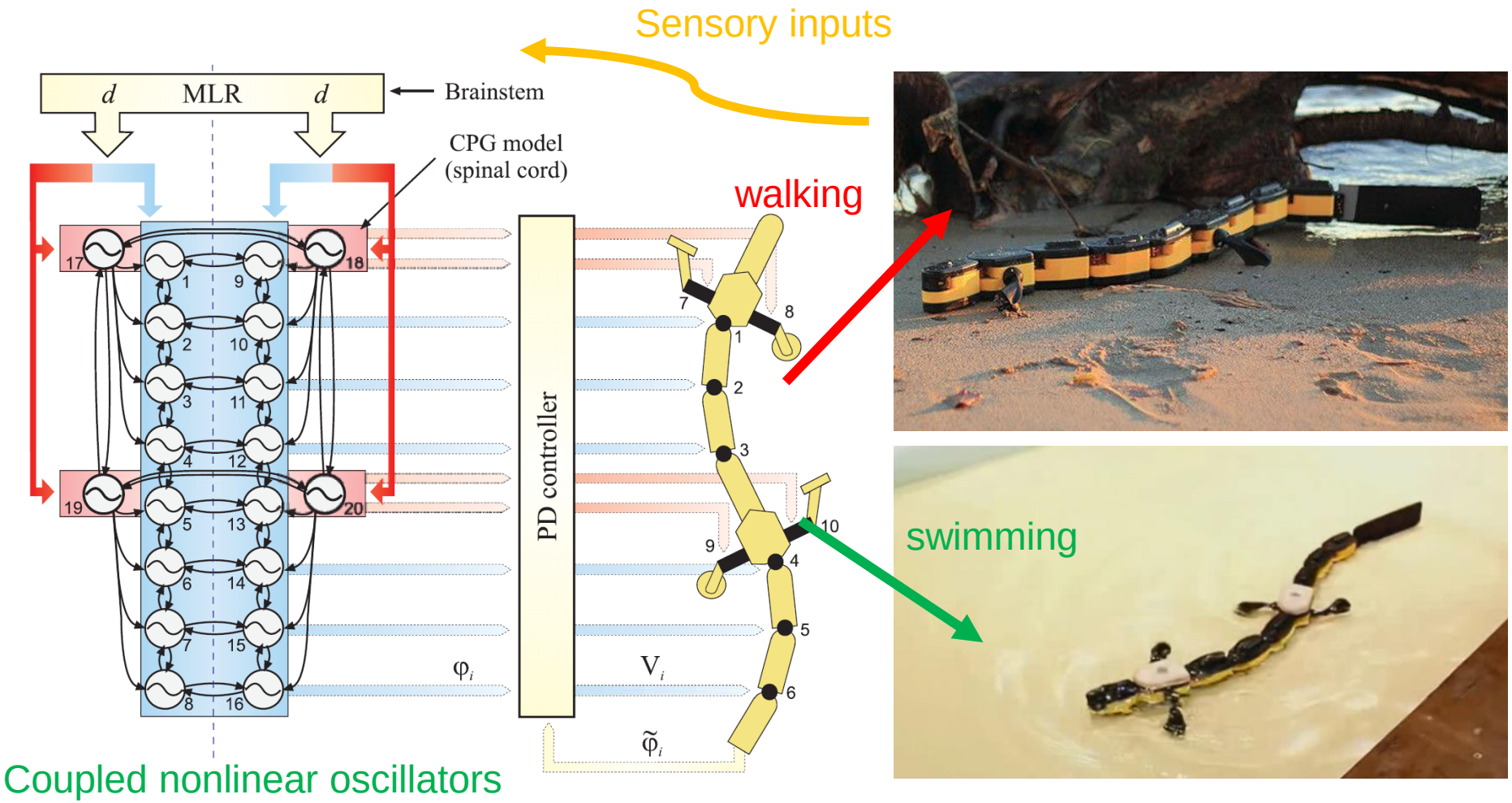
$$\hat{\boldsymbol{x}}_{k+1} = f(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k)$$
$$\hat{\boldsymbol{u}}_k = \hat{\psi}(\hat{\boldsymbol{x}}_k)$$



*On-line learning
(e.g. FORCE learning, Innate learning)

Sussillo, D., & Abbott, L. F. (2009). *Neuron*, 63(4), 544-557.

Designing locomotion patterns and switching them

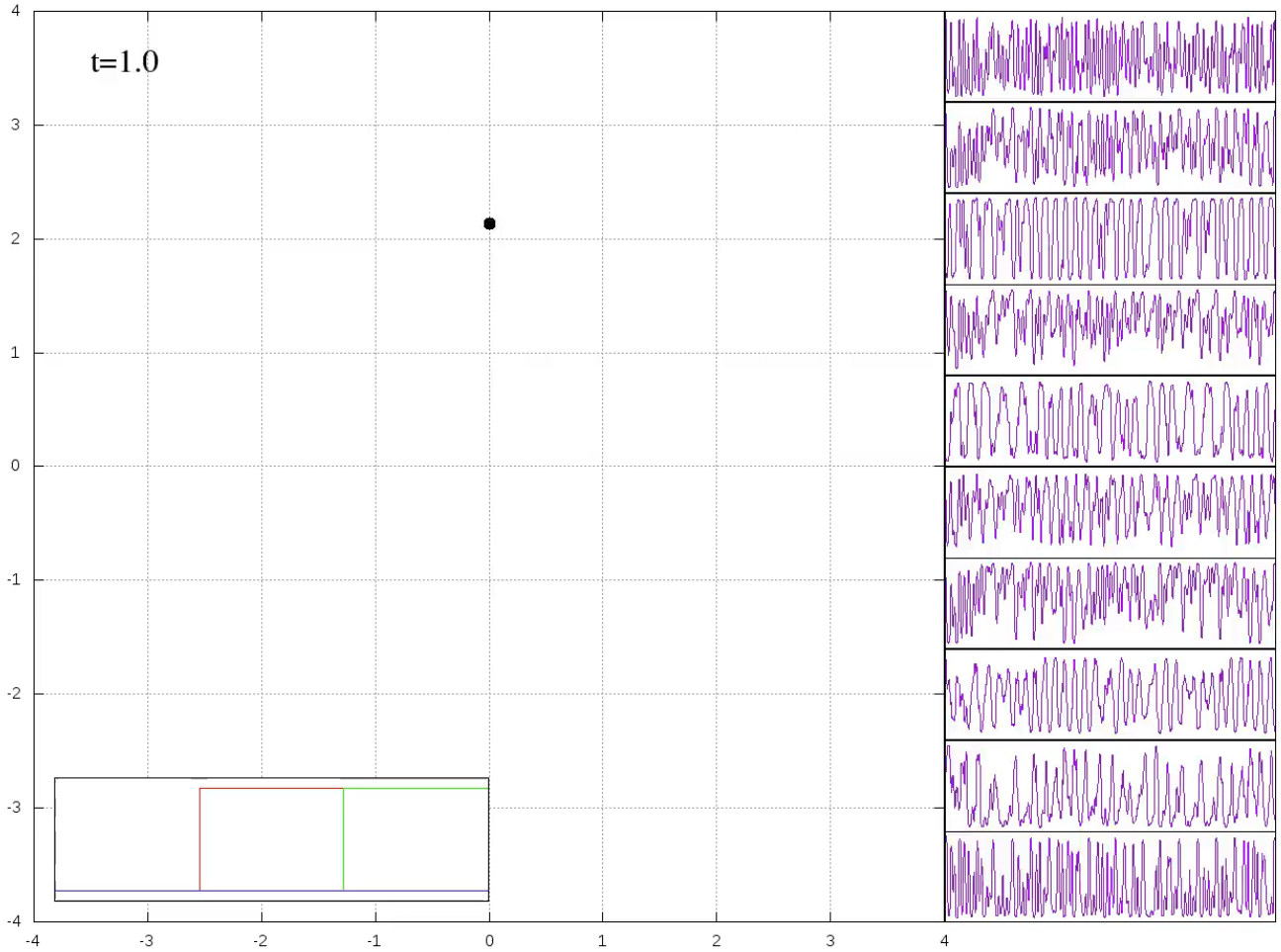


Ijspeert, A. J., Crespi, A., Ryczko, D., & Cabelguen, J. M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. Science, 315(5817), 1416-1420.

- Locomotion through central pattern generators!
- Switching the locomotion patterns via external stimuli! (e.g., sensors or external controllers)

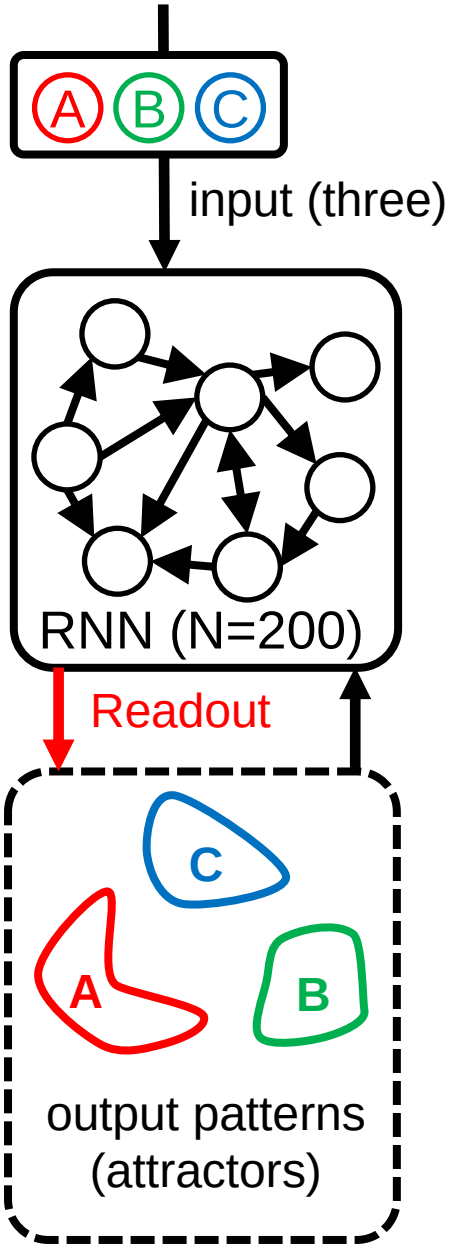
Embedding three limit cycles according to corresponding inputs

Output patterns



Input commands (three)

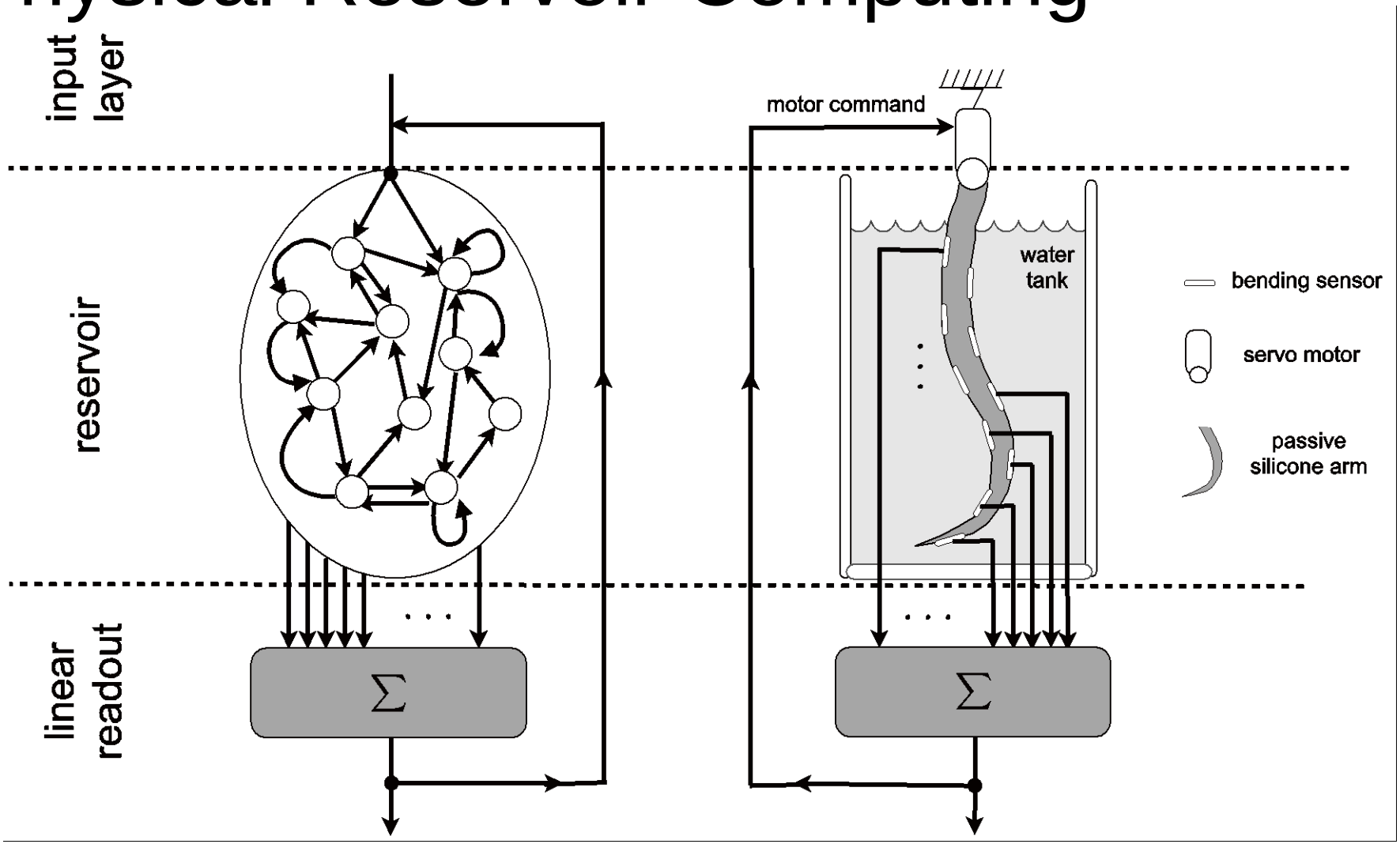
ESN dynamics (from 10 nodes)



High Operability!

K. Inoue, K. Nakajima, Y. Kuniyoshi, Proceedings of NOLTA2018, pp. 412-414, 2018.

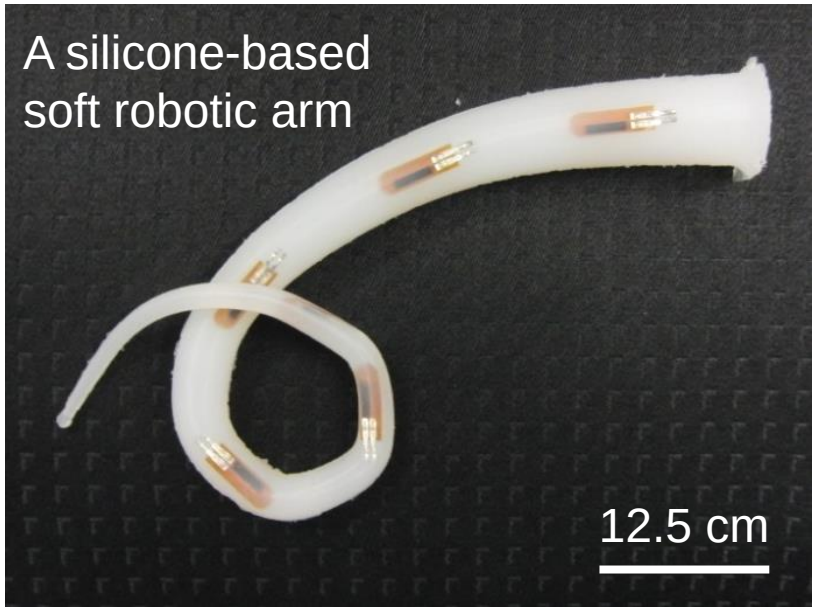
Physical Reservoir Computing

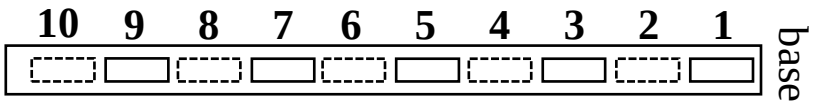


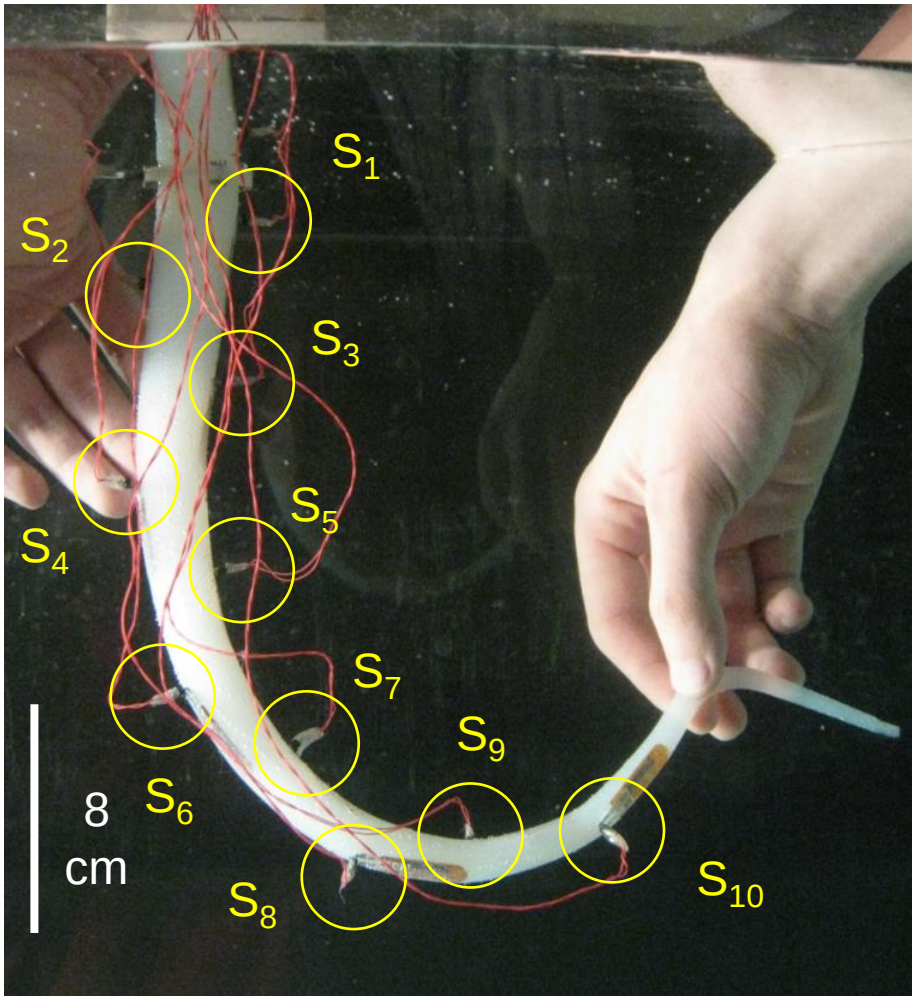
K. Nakajima et. al., *Front. Comput. Neurosci.*, 7 (91), 2013.
K. Nakajima et. al., *J. R. Soc. Interface* 11: 20140437, 2014.
K. Nakajima et. al., *Scientific Reports* 5: 10487, 2015.
K. Nakajima et. al., *Soft Robotics 5(3): pp.339-347*, 2018.

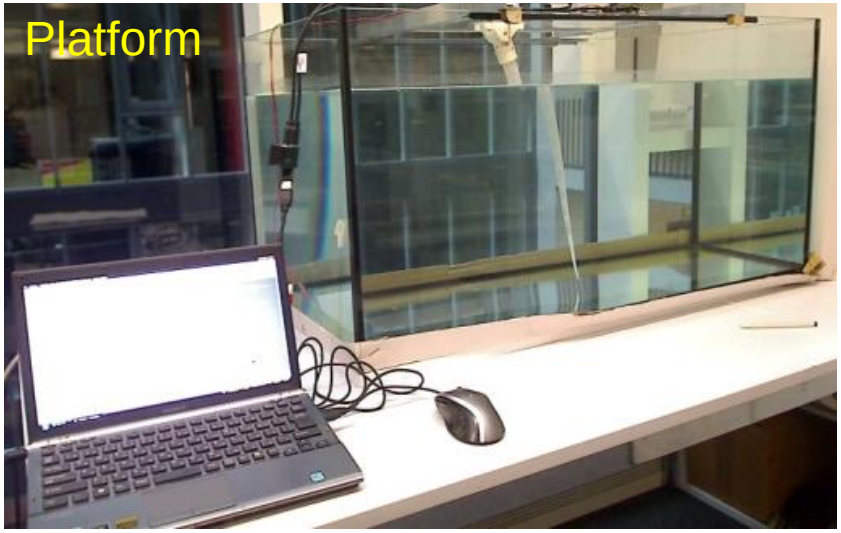
Soft body dynamics as computational resource!

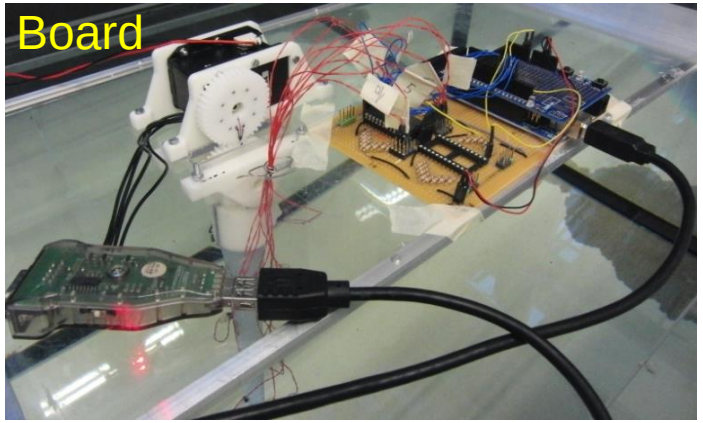
Physical platform

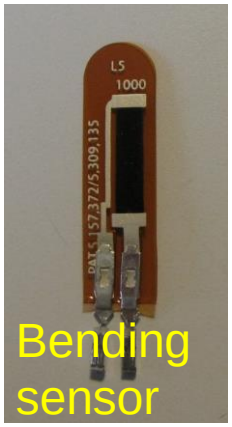




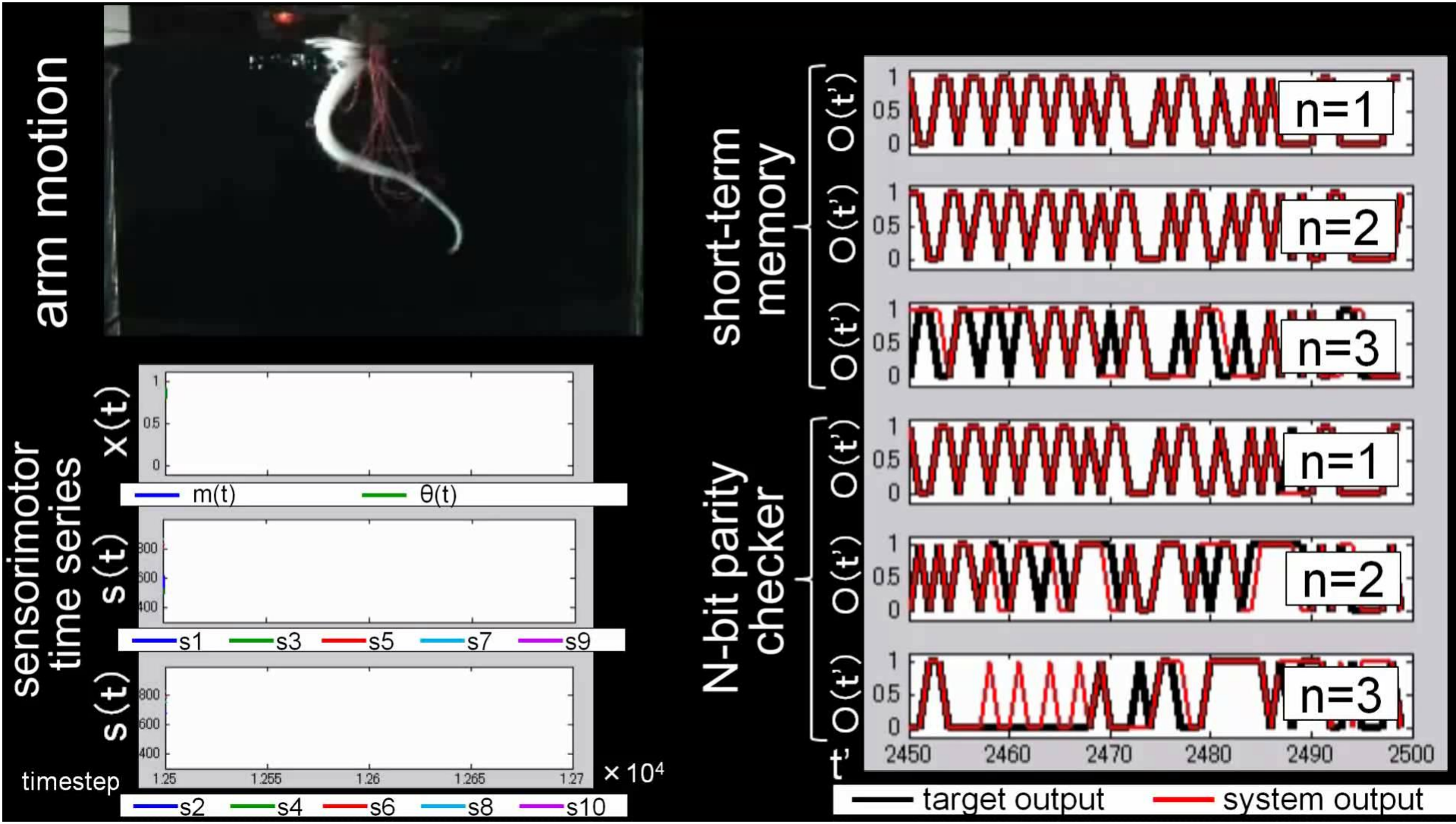






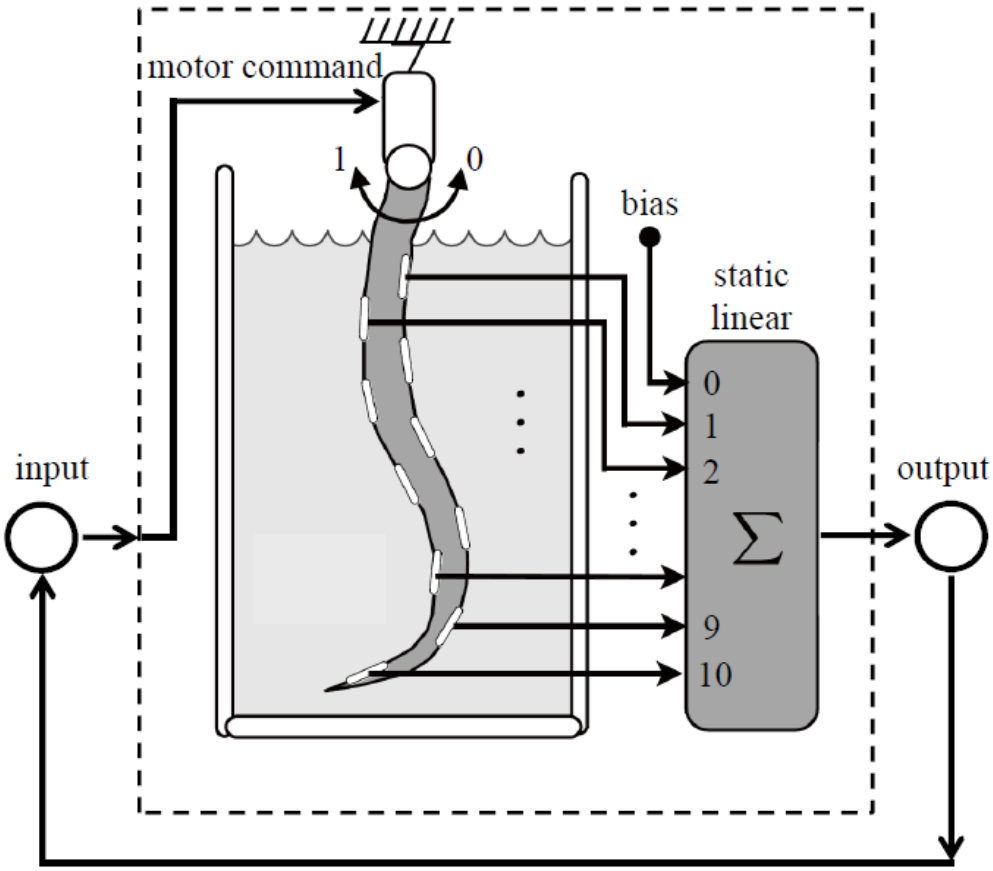


Arm behavior with τ_{state}=5



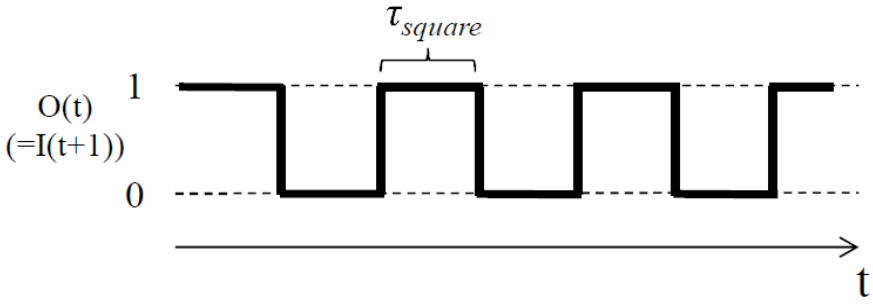
K. Nakajima et. al., *J. R. Soc. Interface* 11: 20140437, 2014.

Closed-loop control: embedding motor program into the body



(Oscillatory pattern)

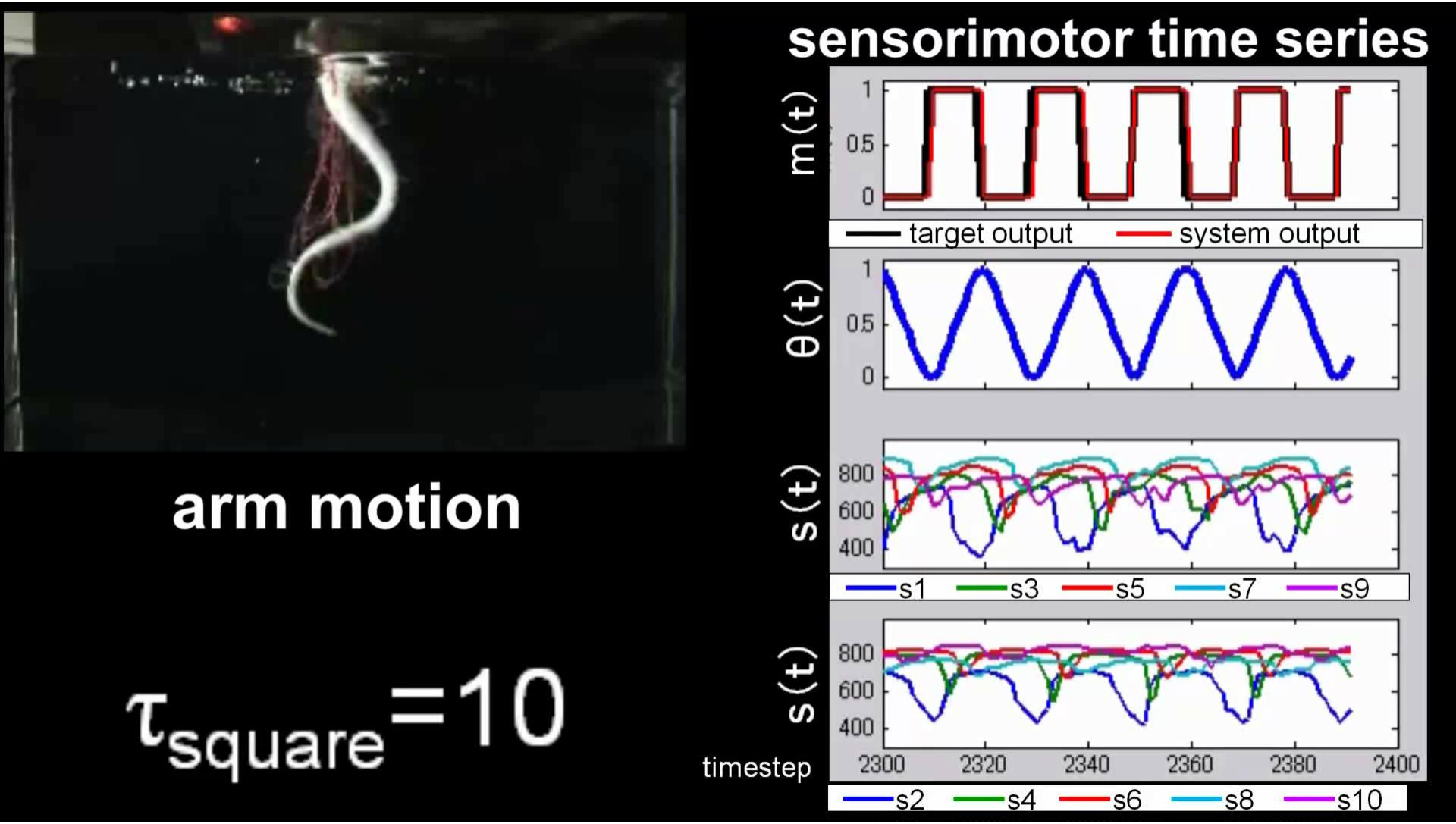
$$x(t) = \frac{1}{2}(sgn(\sin(\frac{2\pi}{\tau_{square}}t)) + 1)$$



Similar motor command with octopus swimming robot!

- No addition of memory from external controller!
- Autonomous behavior control by closing the loop!
- Investigate its robustness!

Closed loop control with τ_{square} =10 (manual perturbation)

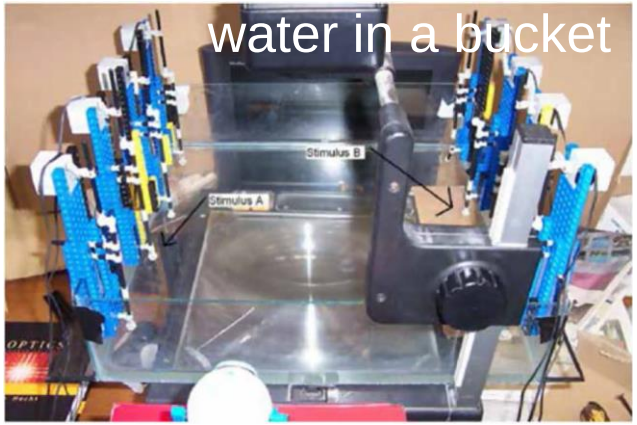


K. Nakajima et. al., *J. R. Soc. Interface* 11: 20140437, 2014.

Physical reservoirs …

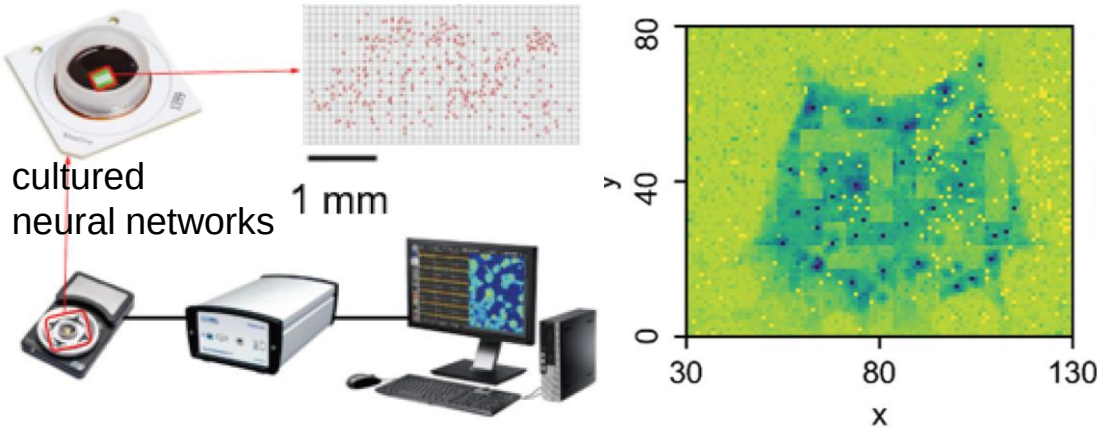
K. Nakajima, Physical reservoir computing---an introductory perspective, Jap. J. Appl. Phys. 59: 060501, 2020.

Liquid brain



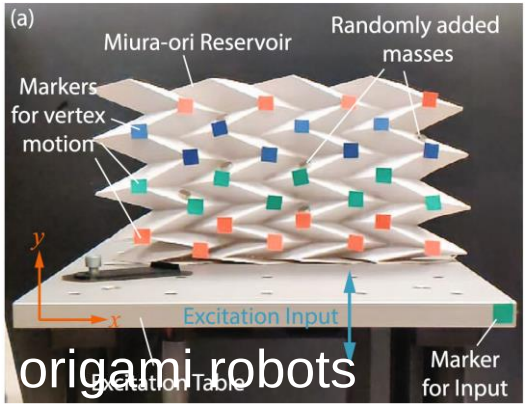
C. Fernando et. al., Lec. Comp. Sci. 2801 (2003).

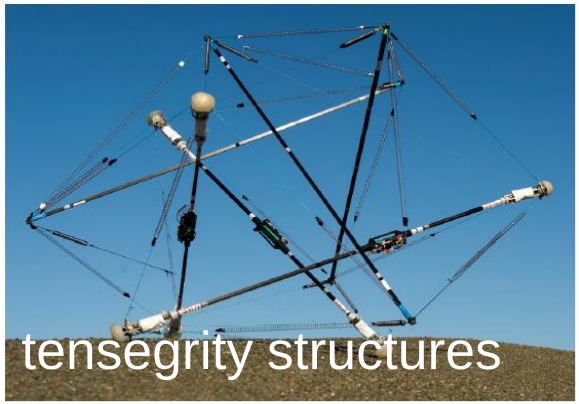
Cultured neural networks

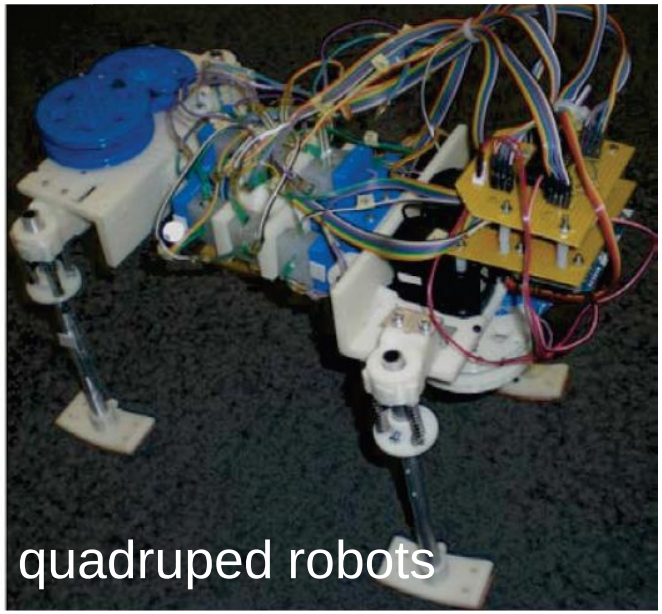


M. R. Dranias, et. al., J. Neurosci. 33, 1940 (2013).
T. Kubota, et. al., Lect. Comp. Sci. 11731 (2019).
Y. Yada, et. al., Appl. Phys. Lett., 119(17), 173701 (2021).

Soft robots



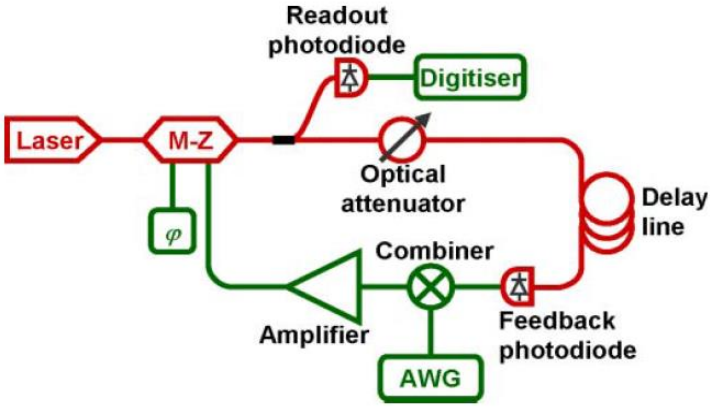




Q. Zhao et al., Proceedings of IROS, pp. 1445-1451 (2013).
K. Nakajima et al. J. R. Soc. Interface. 11: 20140437 (2014).
K. Caluwaerts et al. J. R. Soc. Interface 11:98 (2014).
K. Nakajima et al. Sci. Rep. 5: 10487 (2015).
K. Nakajima et al. Soft Robotics 5: 10487 (2018).
P. Bhovad, et. al., Sci. Rep. 11(1), 1-18 (2021).

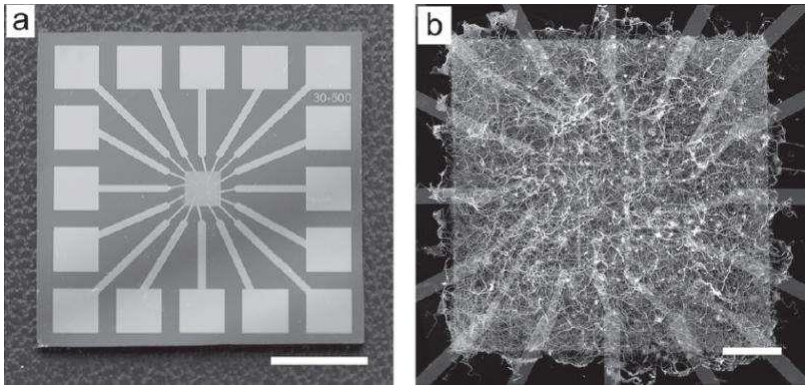
PRC for neuromorphic devices

Photonic reservoirs



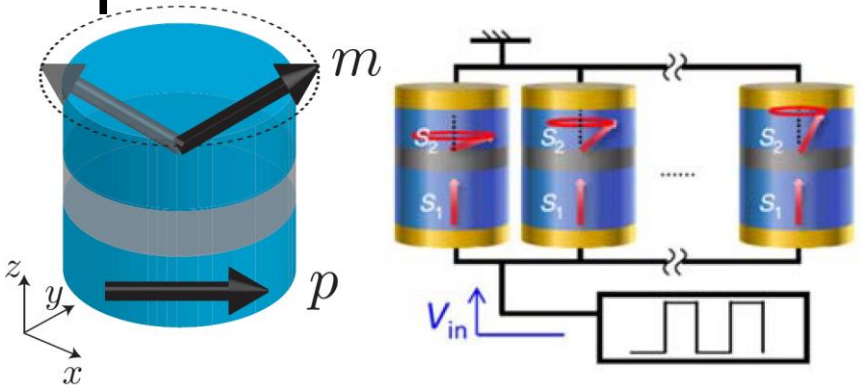
L. Larger, et. al., Opt. Express 20, 3241 (2012).
D. Brunner, et. al., Nat. Commun. 4, 1364 (2013).
K. Vandoorne, et. al., Nat. Commun. 5, 3541 (2014).
L. Larger, et. al., Phys. Rev. X 7, 011015 (2017).
M. Nakajima et al.. Nat. Commun. 13, 7847 (2022).

In-Materia reservoirs



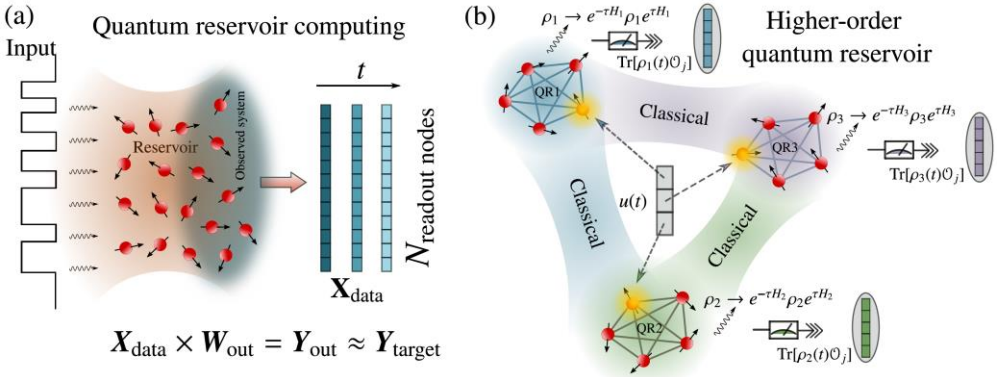
E. C. Demis et al. Nanotechnology 26:204003 (2015).
A. Z. Stieg et al. Adv. Mater. 24:286-293 (2012).
M. Cucchi, et. al., Science Advances, 7(34), eabh0693 (2021).
Y. Usami, et. al., Adv. Mater. (2021).

Spintronics reservoirs



J. Torrejon et al., Nature 547, 428 (2017).
T. Furuta, et. al., Phys. Rev. Appl. 10, 034063 (2018).
S. Tsunegi, et. al., Appl. Phys. Lett. 114, 164101 (2019).
N. Akashi, et. al., Phys. Rev. Res. 2: 043303 (2020).
Lee, O., Wei, T., Stenning, K.D. et al. Nat. Mater. (2023).

Quantum reservoirs



K. Fujii, K. Nakajima, Phys. Rev. Appl. 8: 024030 (2017).
K. Nakajima, et. al., Phys. Rev. Appl. 11: 034021 (2019).
S. Ghosh, et. al., Adv. Quantum Technol. 4: 2100053 (2021).
Q. H. Tran, K. Nakajima, Phys. Rev. Lett. 127: 260401 (2021).
T. Kubota et al.,Phys. Rev. Res., 5(2), 023057 (2023)..

*Prerequisite: "*reproducible response*"

(target function)

$$y_{k+1} = g(u_k, u_{k-1}, \dots)$$

(reservoir computing)

$$\begin{cases} \boldsymbol{x_{k+1}} = f(\boldsymbol{x_k}, u_k) \\ \hat{y}_{k+1} = \hat{\psi}(\boldsymbol{x_{k+1}}) \end{cases}$$

$$g(u_k, u_{k-1}, \dots) \approx \hat{\psi}(\boldsymbol{x_{k+1}})$$

We want to emulate (learn) function "g"!

(prerequisite)

- Reproducible response to the same input sequence!
- Reservoir states should not depend on the initial condition!

$$f(\boldsymbol{x_k}, u_k) - f(\boldsymbol{x_k^*}, u_k) \approx 0 \quad \longleftarrow \quad \boldsymbol{x_k} = \boldsymbol{\phi}(u_{k-1}, u_{k-2}, \dots)$$

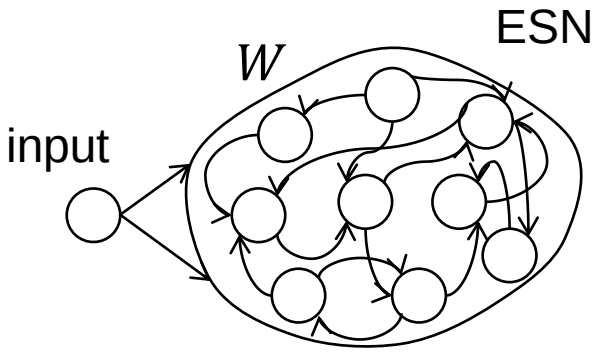
(common-signal-induced synchronization/ Negative conditional Lyapunov exponents)

Z. Lu, B. R. Hunt, E. Ott, *Chaos* 28, 061104 (2018).

(echo state property (ESP))

H. Jaeger, GMD Technical Report. 148 (2001).
I. B. Yildiz, et. al., *Neural netw.* 35 (2012).
G. Manjunath, et. al., *Neural comp.* 25 (2013).

Echo-state network (ESN)



- number of nodes : N
- state of neuron i at t : $x_i(t)$
- action potential of neuron i at t : $a_i(t)$
- input : $u(t)$
- internal weights : w_{ij}
- input weights : w_{in}
- activation function : $f(a) = \tanh(a)$
- dynamics :

spectral radius of W

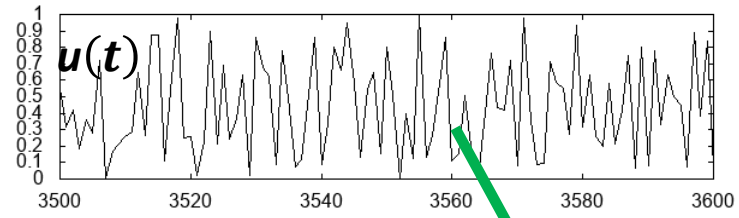
$$\rho(W) = \max\{|\lambda_1|, \ldots, |\lambda_N|\}$$

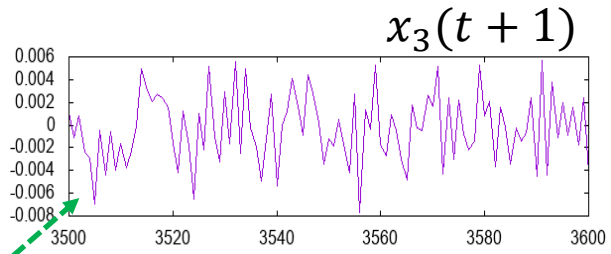
$$x_i(t+1) = f\big(a_i(t)\big), \quad a_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t) + w_{in} u(t)$$

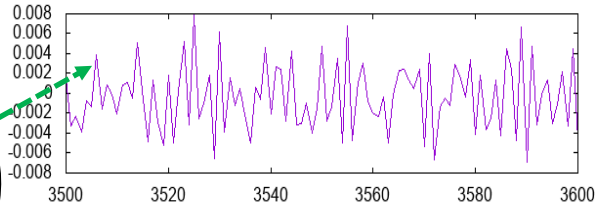
Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach (Vol. 5, p. 01). Bonn: GMD-Forschungszentrum Informationstechnik.

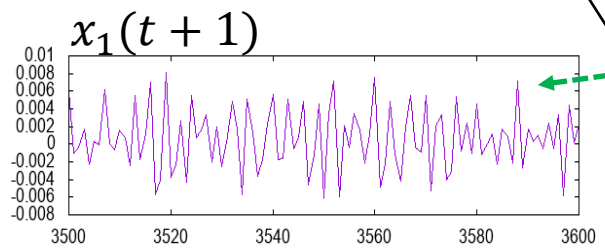
Example of ESN dynamics

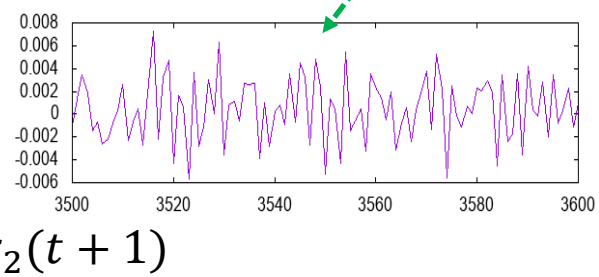
$(N = 200, \rho = 0.9)$

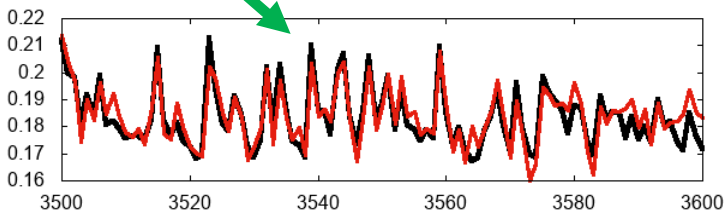












input

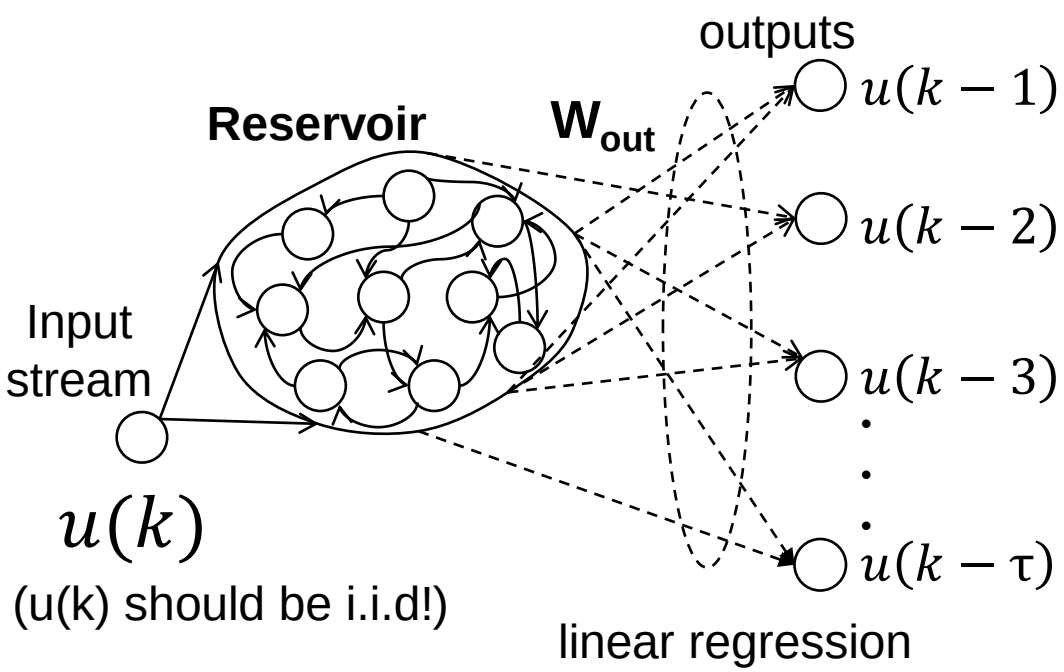
readout

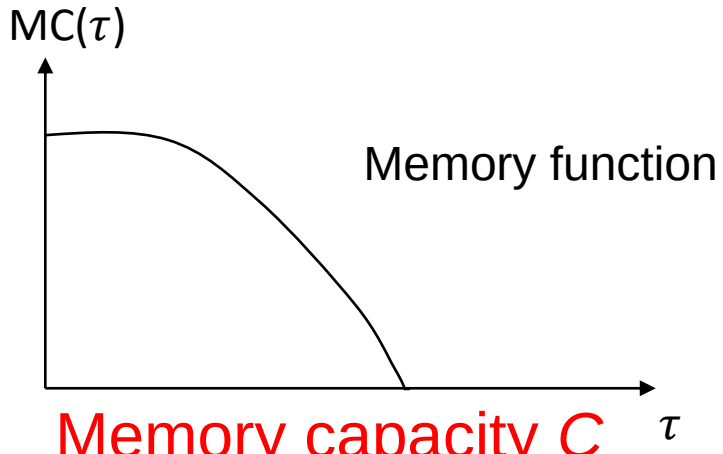
output

$$g(u(t), u(t-1), \dots) \approx \hat{\psi}(x(t+1))$$

- Diverse dynamics within the reservoir!

Memory Capacity





Memory capacity *C*

$$C = \sum_{\tau=1}^{\infty} MC(\tau)$$

Memory function

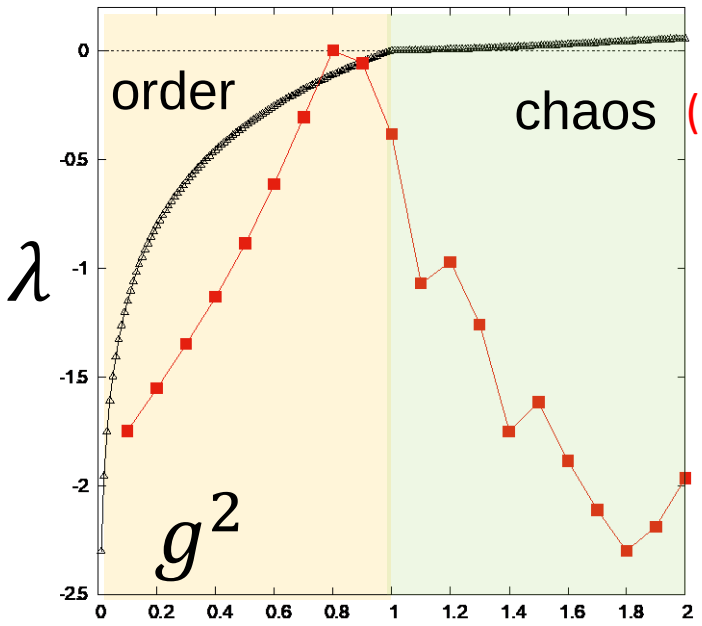
$$MC(\tau) = \frac{cov^2(y_s(k), u(k-\tau))}{\sigma^2(y_s(k))\sigma^2(u(k-\tau))}$$

H. Jaeger et al., “Short term memory in echo state networks," GMD Report 152 (2002).

Only by using current state of the reservoir, how much of the past input can be reconstructed?

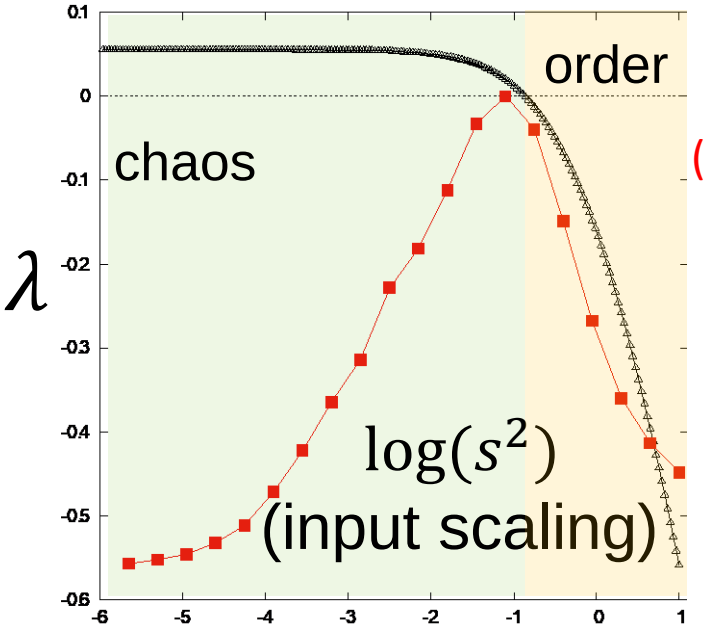
→*C* is called memory capacity of the reservoir

Bifurcation of ESN and its memory capacity



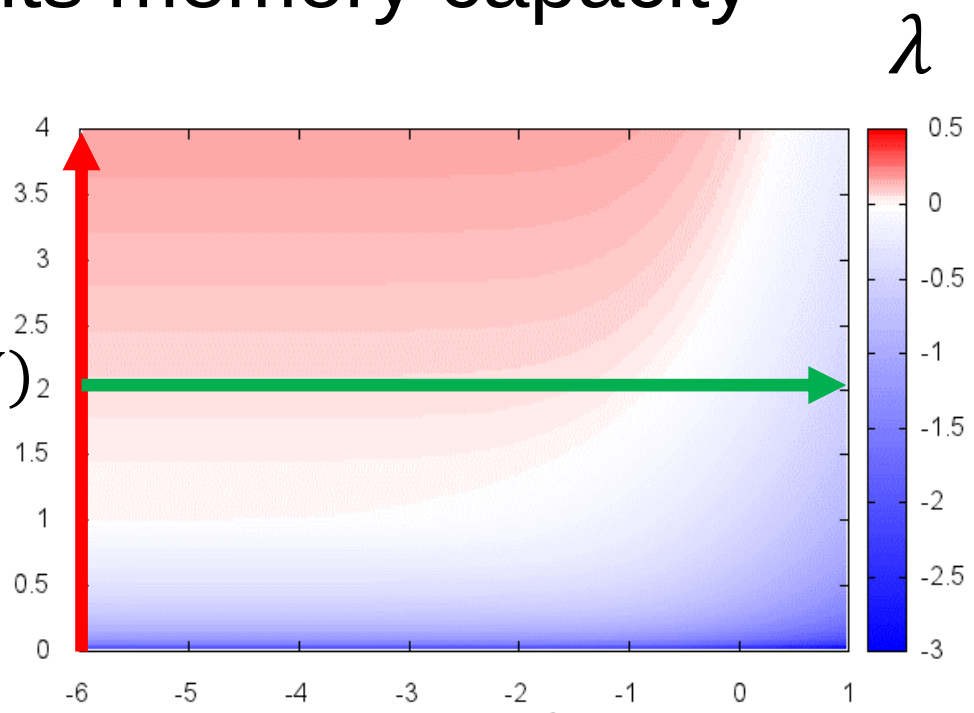
C_{max} (=65.8)

C



C_{max} (=13.3)

C



C is maximized around the bifurcation point!

(C is normalized in the left plots)

Is chaos useless in RC?

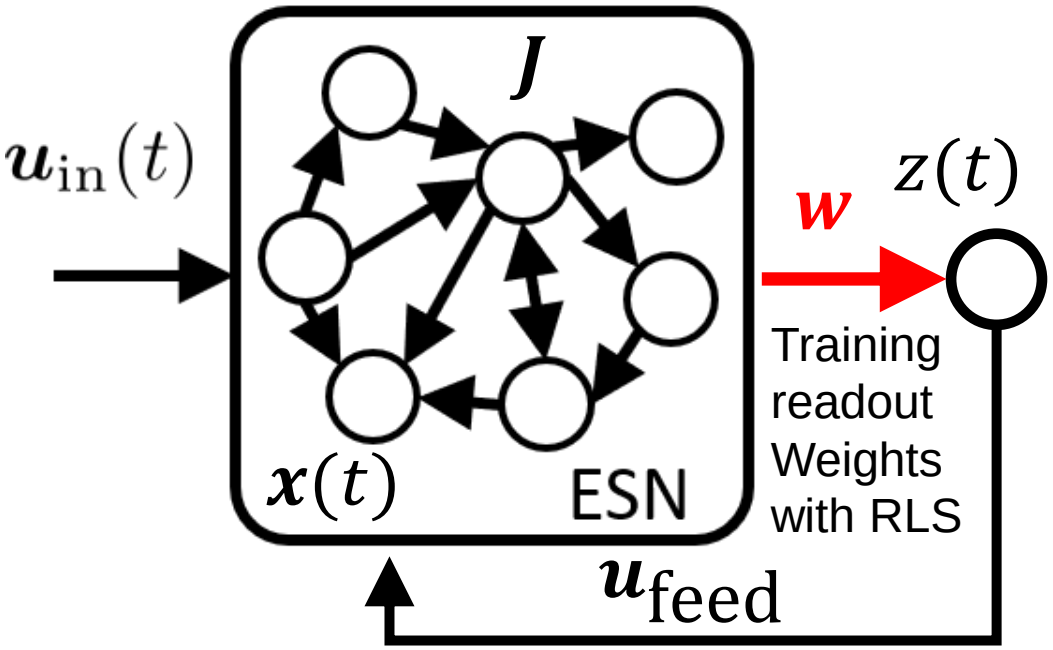
-> several ways proposed

Laje, R., & Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature neuroscience*, 16(7), 925.

Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science advances*, *6*(46), eabb3989.

Liu, S., Akashi, N., Huang, Q., Kuniyoshi, Y., & Nakajima, K. (2024). Exploiting Chaotic Dynamics as Deep Neural Networks. *arXiv preprint* arXiv:2406.02580.

Reservoir settings: chaotic ESN



τ: time constant,
$\boldsymbol{x}$: network state, $\boldsymbol{x} \in \mathbb{R}^N$
g: nonlinearity parameter
($g > 1$: chaotic)
$\boldsymbol{J}$: $N \times N$ matrix,
($random\ with\ spectral\ radius = 1$)
$\boldsymbol{u}_{\mathrm{in}}$: input function, $\boldsymbol{u}_{\mathrm{in}}: \mathbb{R} \to \mathbb{R}^N$

$\boldsymbol{u}_{\mathrm{feed}}$: $N \times 1$ matrix, $\boldsymbol{u}_{\mathrm{feed}} \sim [-1, 1]$
$\boldsymbol{w}$: readout weights, $N \times 1$ matrix

K. Inoue, K. Nakajima, Y. Kuniyoshi, Proceedings of NOLTA2018, pp. 412-414, 2018.

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \tanh\left(g\boldsymbol{J}\boldsymbol{x}(t) + \boldsymbol{u}_{\mathrm{feed}} z(t) + \boldsymbol{u}_{in}(t)\right)$$

$$z(t) = \boldsymbol{w}^T \boldsymbol{x}(t)$$

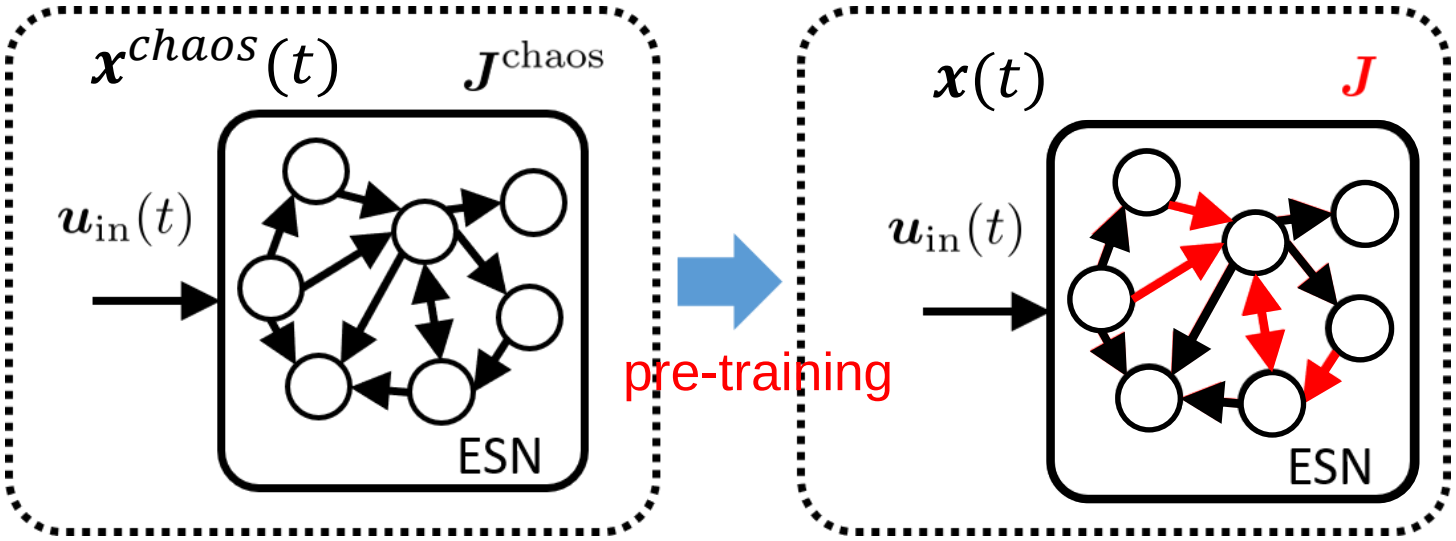
Use chaotic ESN ($g = 1.5$)

Innate training:
"exploiting intrinsic chaotic dynamics"

(Points)

1. Method with **chaotic reservoir**
2. **Pre-training**
3. **Semi-supervised** method (*innate* trajectory)
4. Robustness to **external perturbation**
5. Coexisting **chaotic and locally stable** trajectory

Innate training approach: exploiting chaos



Train *J* to minimize

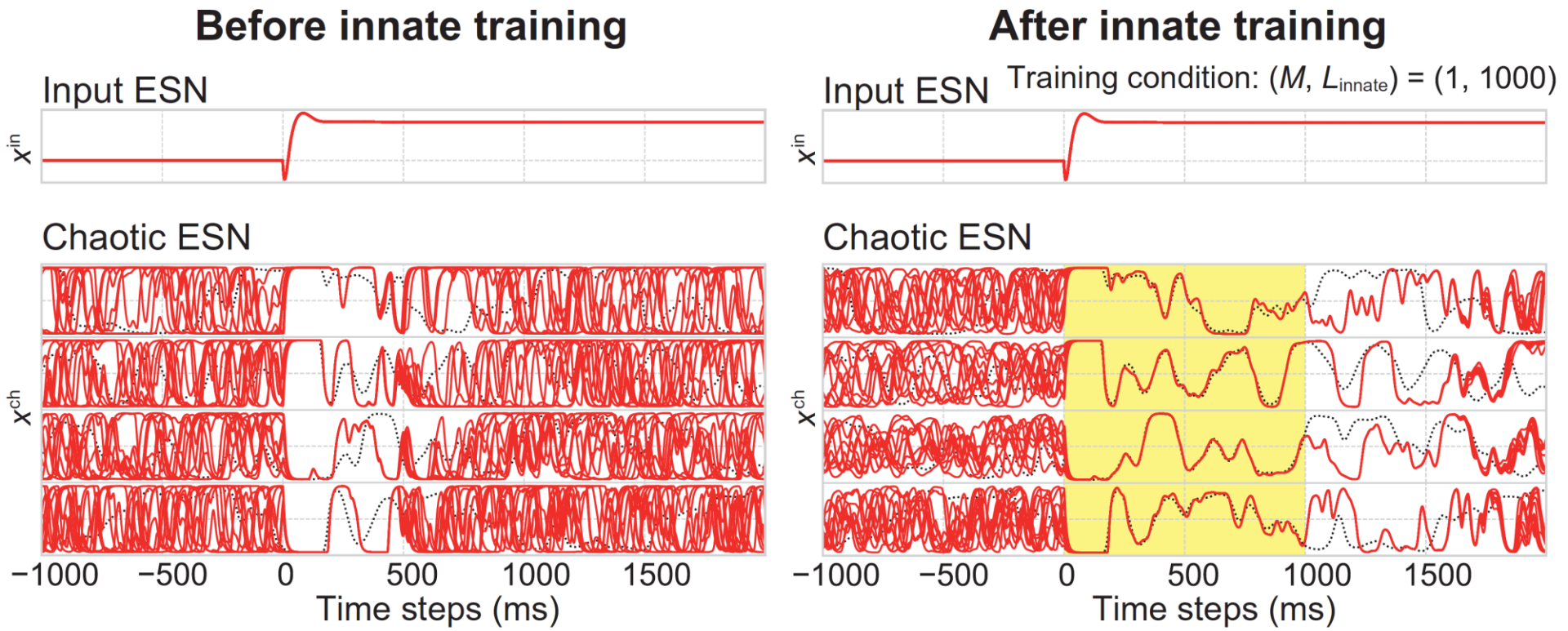
$$C_J = \int_{T_0}^{T_1} \| \; \boldsymbol{x}^{chaos}(t) - \boldsymbol{x}(t) \; \|^2 dt$$

Laje, R., & Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature neuroscience*, 16(7), 925.

The system is…

- globally chaotic
- locally stable and reproducible dynamics

Innate training: example

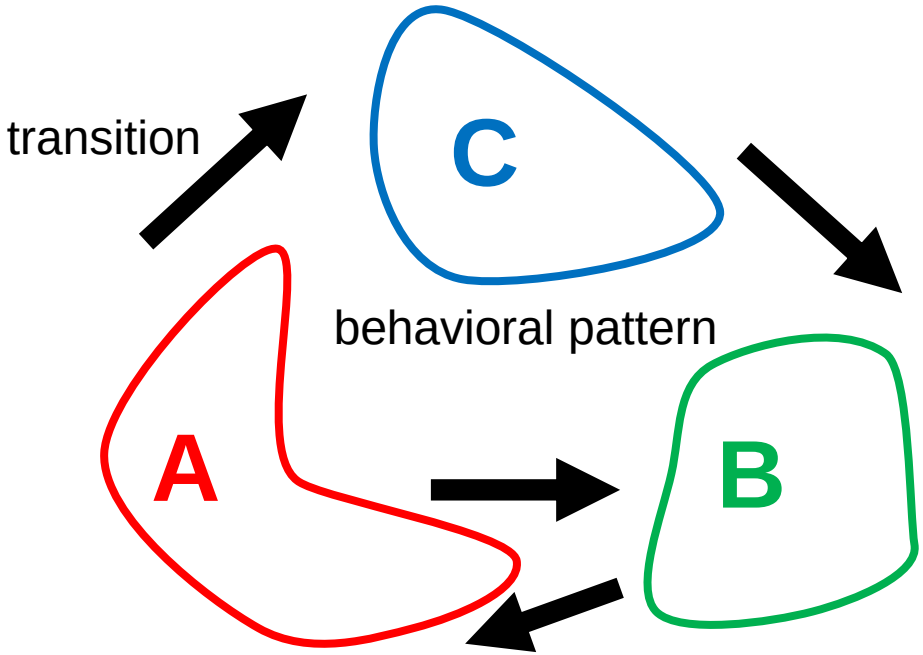


Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). *Science Advances* 6 (46), eabb3989.

(Features)

- Generate the same innate trajectory reproducibly according to the corresponding input!
- Learned innate trajectory is stable, but the ESN is still chaotic!

Designing spontaneous behavioral switching



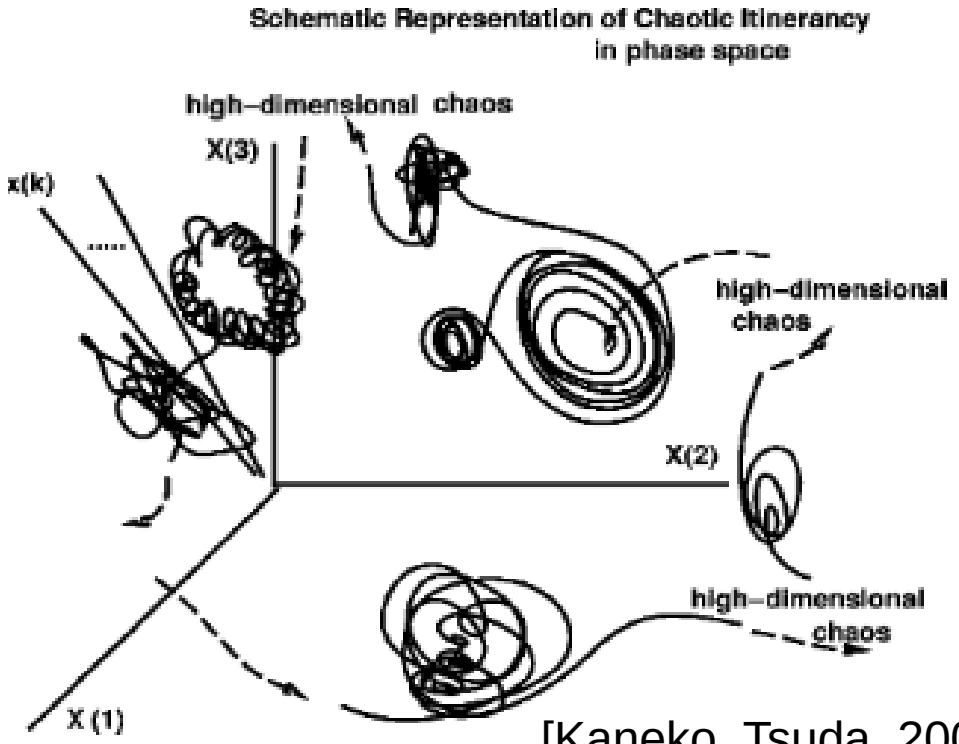
~~External control~~

Internal dynamics

- **Step 1**: Behavioral patterns
- **Step 2**: Periodic transitions among the patterns
- **Step 3**: Random transitions among the patterns

Step 3 is related to chaotic itinerancy!

What is chaotic itinerancy?



[Kaneko, Tsuda, 2003]

FIG. 1. Schematic representation of chaotic itinerancy.

First found in…

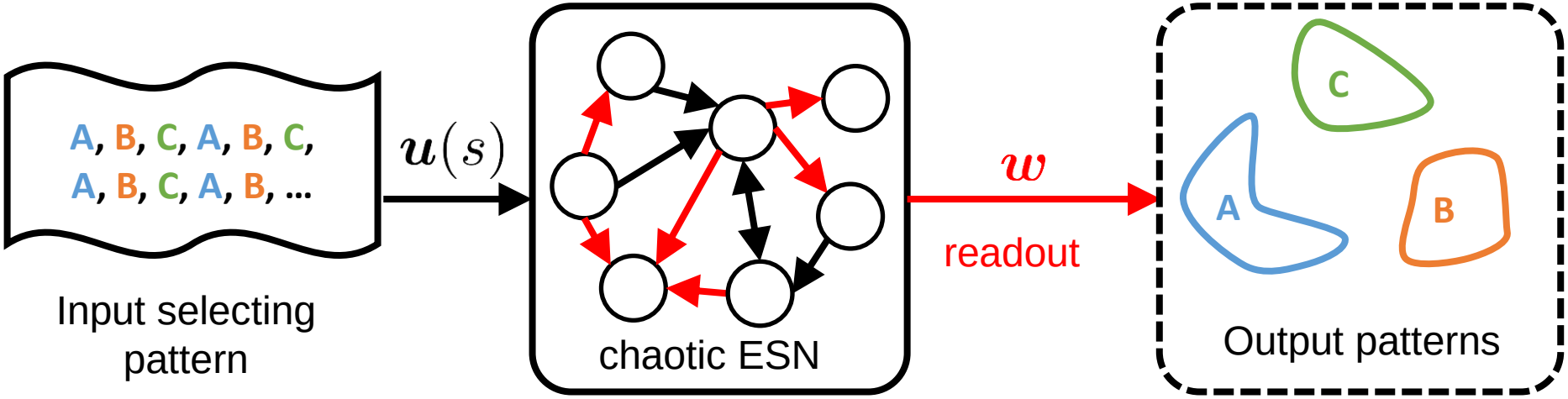
- Optical turbulence
  [K. Ikeda et. al., 1989]
- A globally coupled chaotic system
  [K. Kaneko, 1990; 1991]
- Nonequilibrium neural networks
  [I. Tsuda, 1991; 1992]

(Features)

- Frequently observed in high-dimensional nonlinear dynamical systems
- Seemingly random transitions among quasi-attractors

Propose a scheme to design CI!

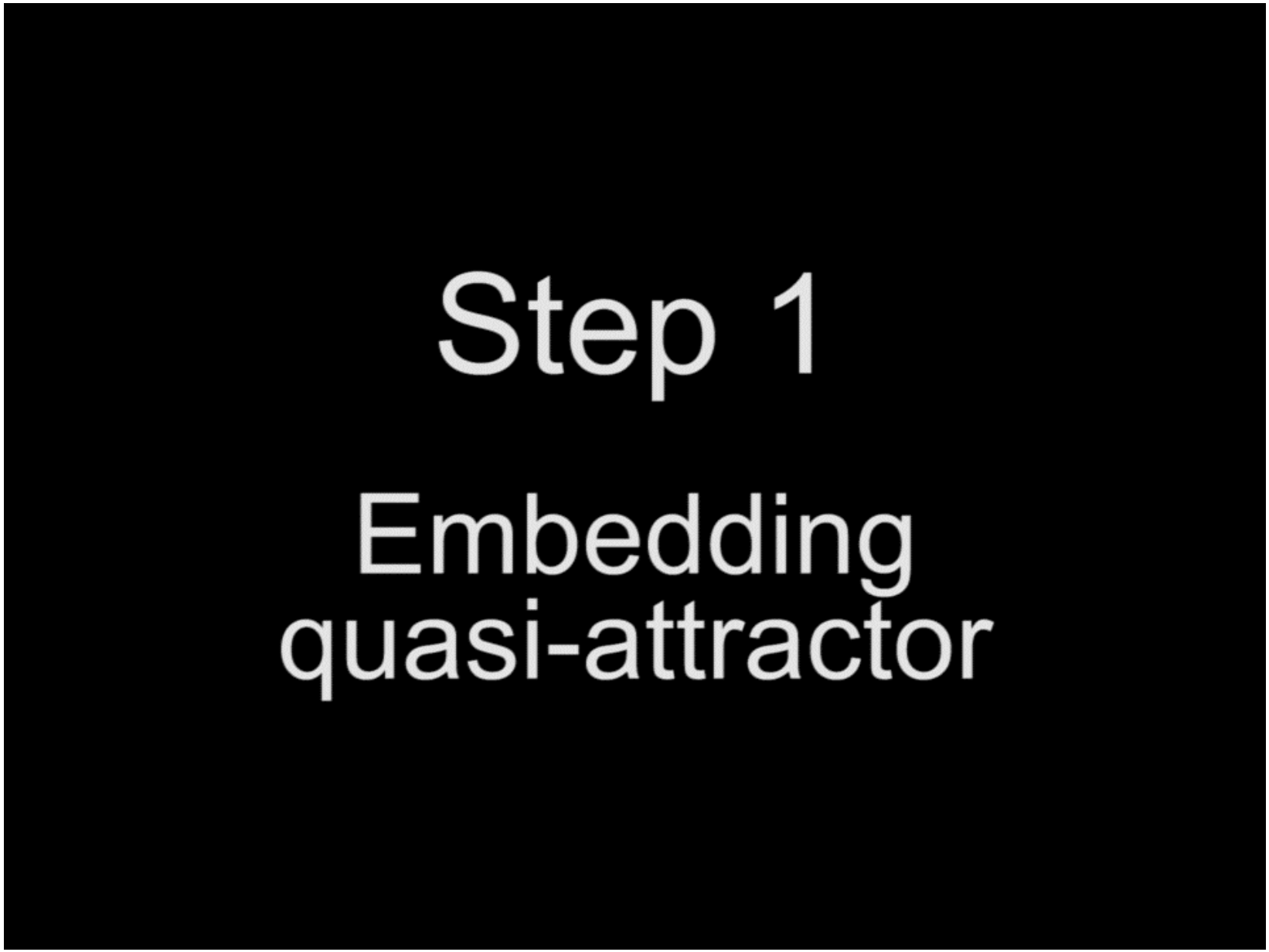
Step 1: Designing complex pattern corresponding to input signal



Pre-training chaotic ESN and embed complex patterns

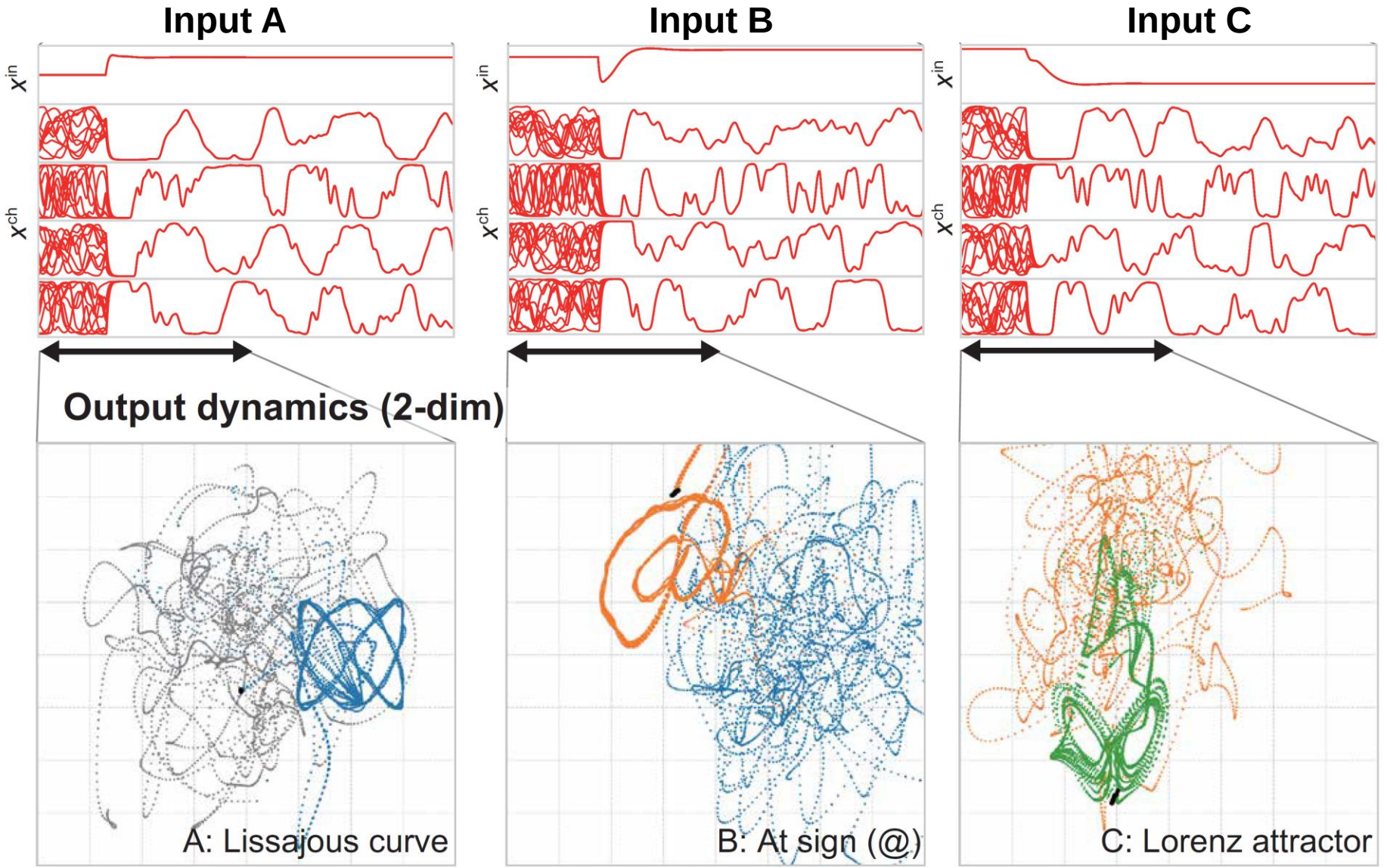
Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances* 6 (46), eabb3989.

*note that in the original paper, a construction called input reservoir is introduced, but this time we skip it for simplicity.



Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances* 6 (46), eabb3989.

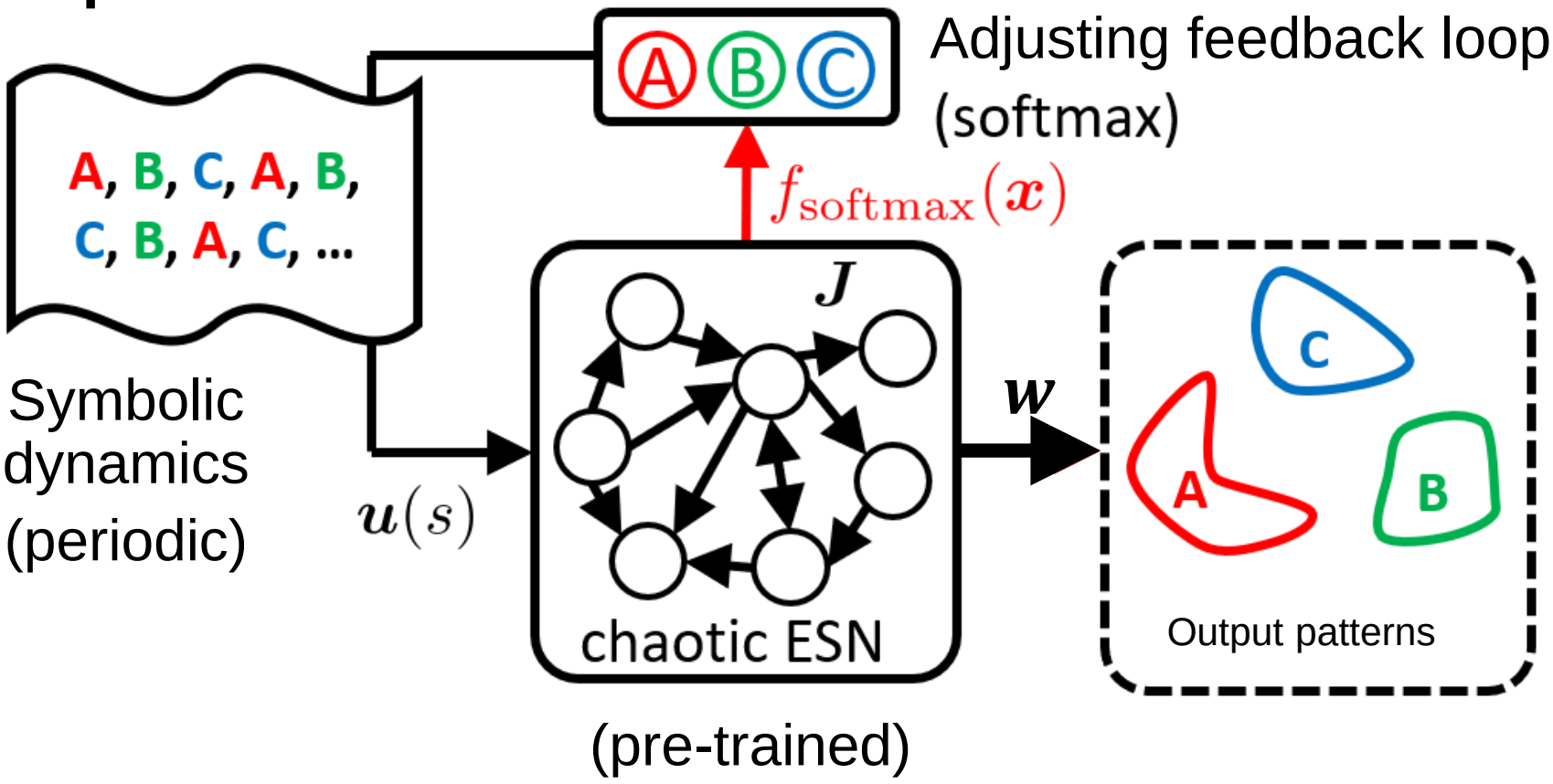
Generate complex patterns according to the input



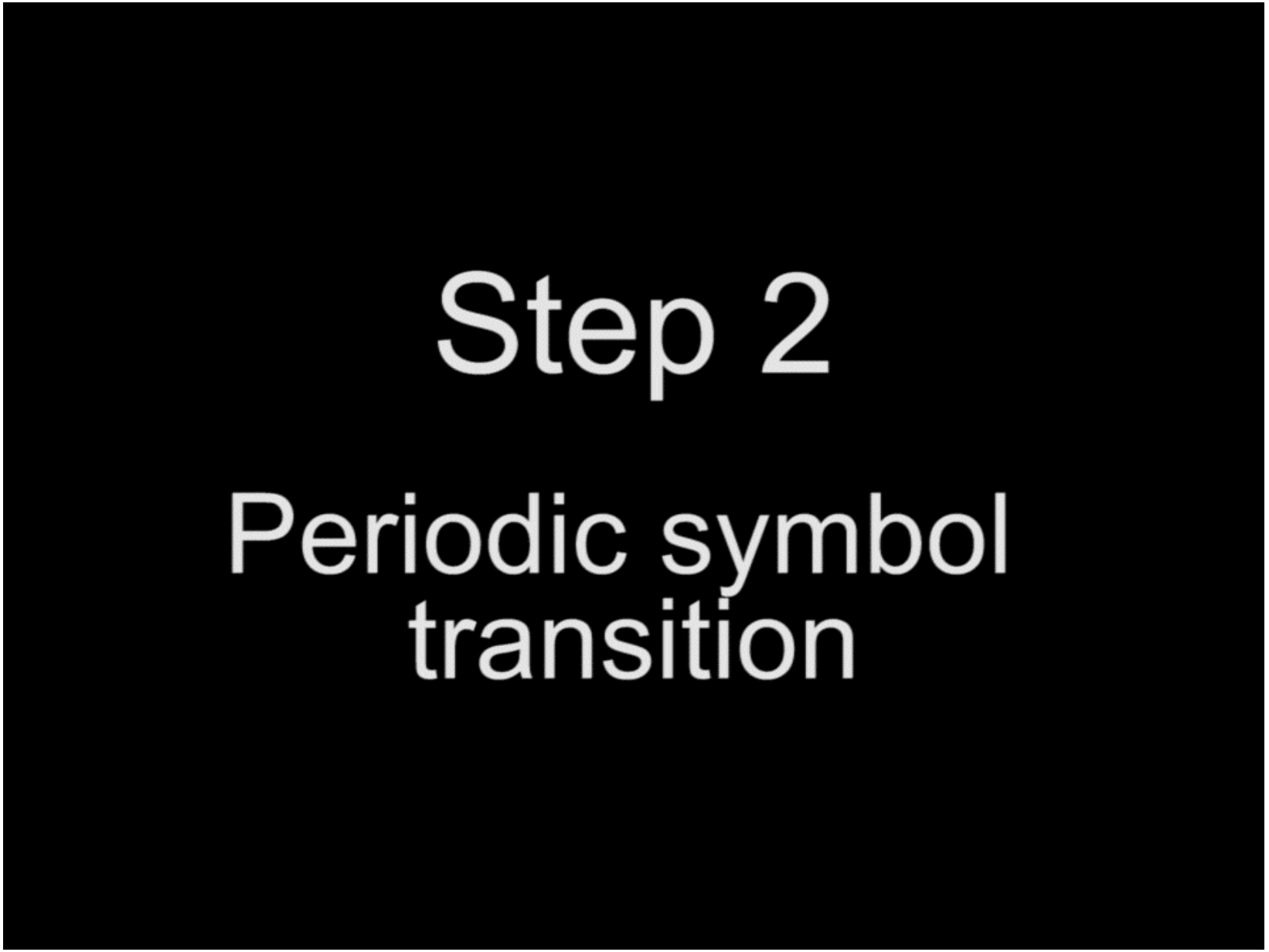
Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). *Science Advances* 6 (46), eabb3989.

- Required patterns are successfully learned!

Step 2: switching embedded patterns in a periodic manner

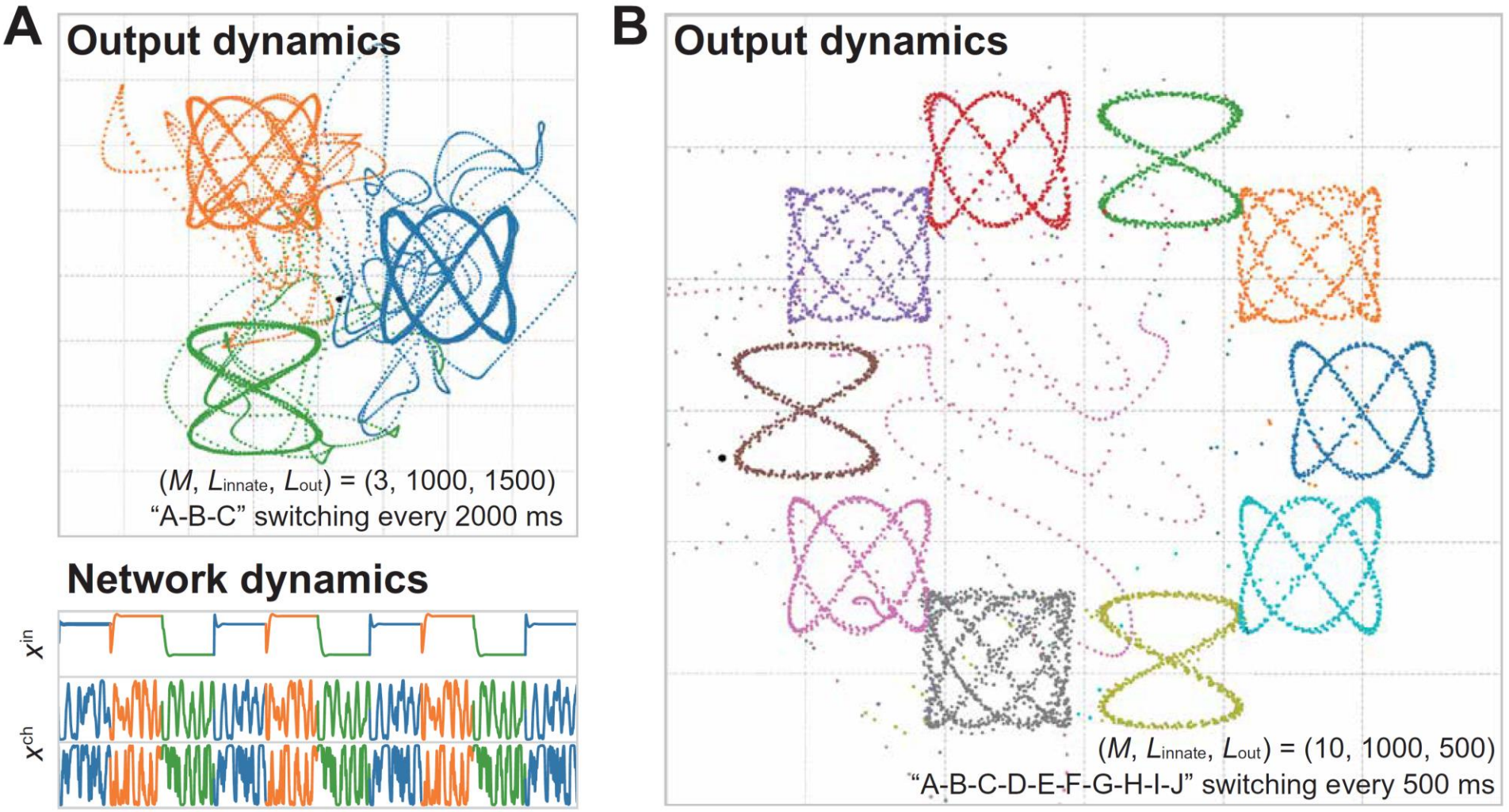


- Internalizing the external control through feedback loop!
- Should estimate the duration of time for each pattern (switch in appropriate timing) using the same reservoir!



Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances* 6 (46), eabb3989.

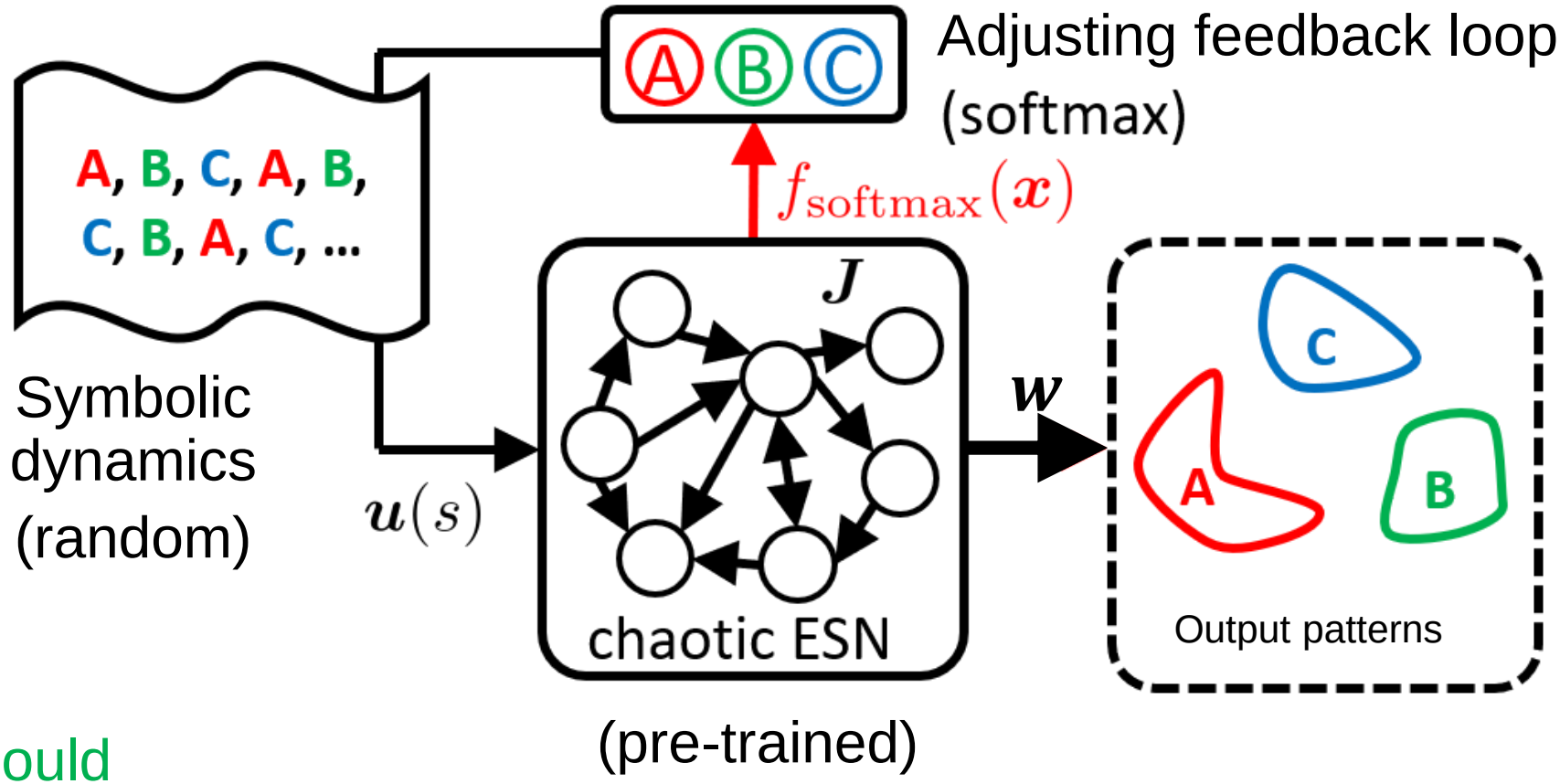
Periodic transitions are embedded!



Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). *Science Advances* 6 (46), eabb3989.

- Can design a fine structure in a limit cycle.

Step 3: switching embedded patterns in a random manner



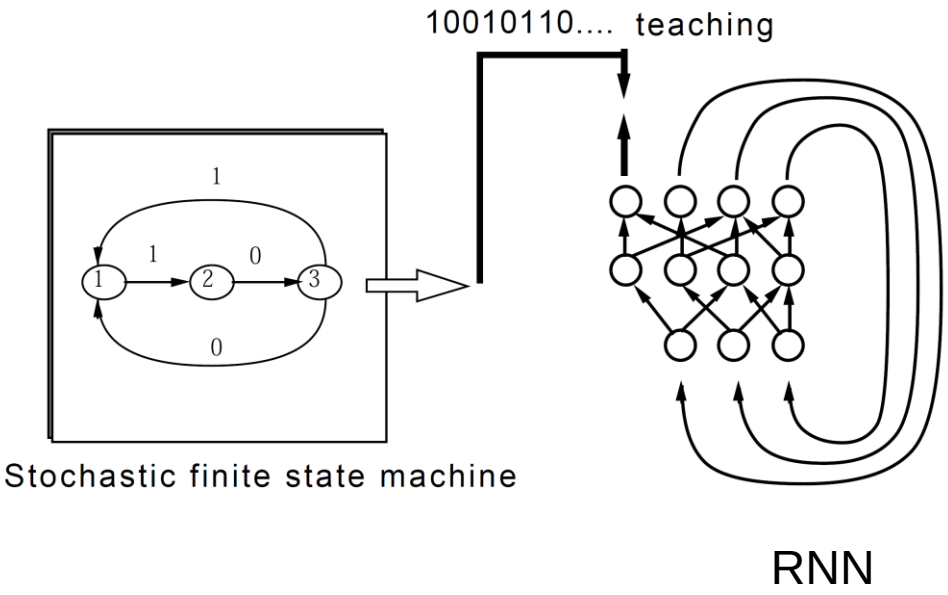
Should

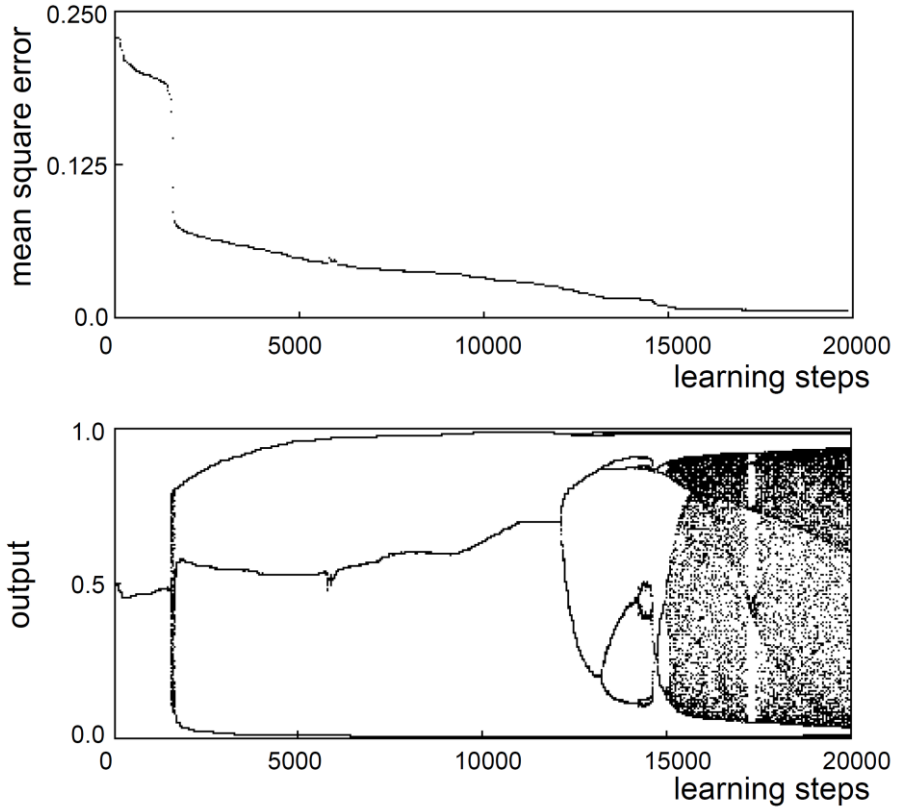
(**estimate the duration of time for each pattern**)

+ (**emulate transition probability**)

using the same chaotic reservoir!

Learning stochastic process with deterministic system generates chaos





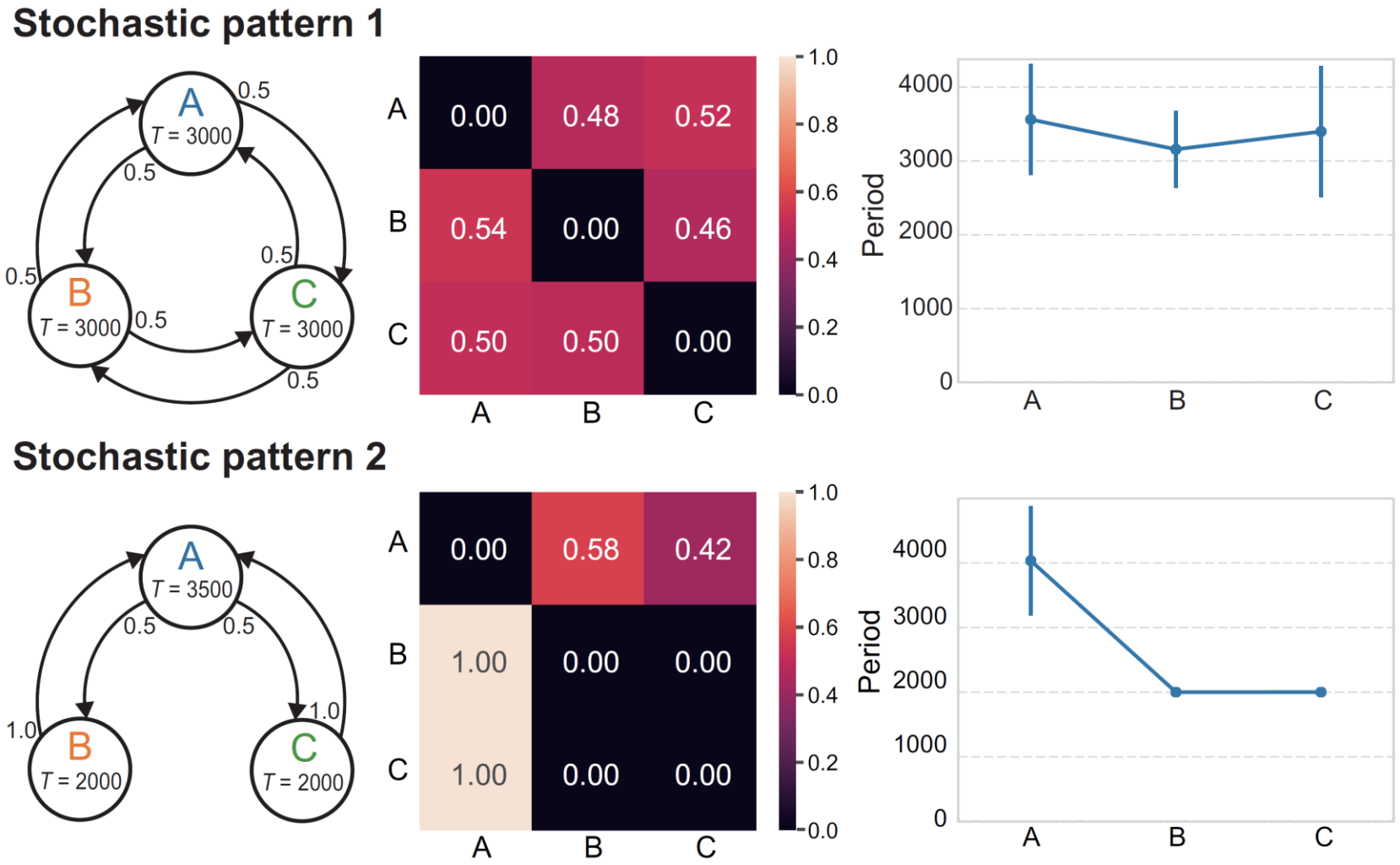
Tani, J., & Fukumura, N. (1995). Embedding a grammatical description in deterministic chaos: an experiment in recurrent neural learning. *Biological Cybernetics*, *72*(4), 365-370.

Chaos is generated!

- In our system, chaos already exists in the network!

Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science Advances* 6 (46), eabb3989.

Designing transition probabilities



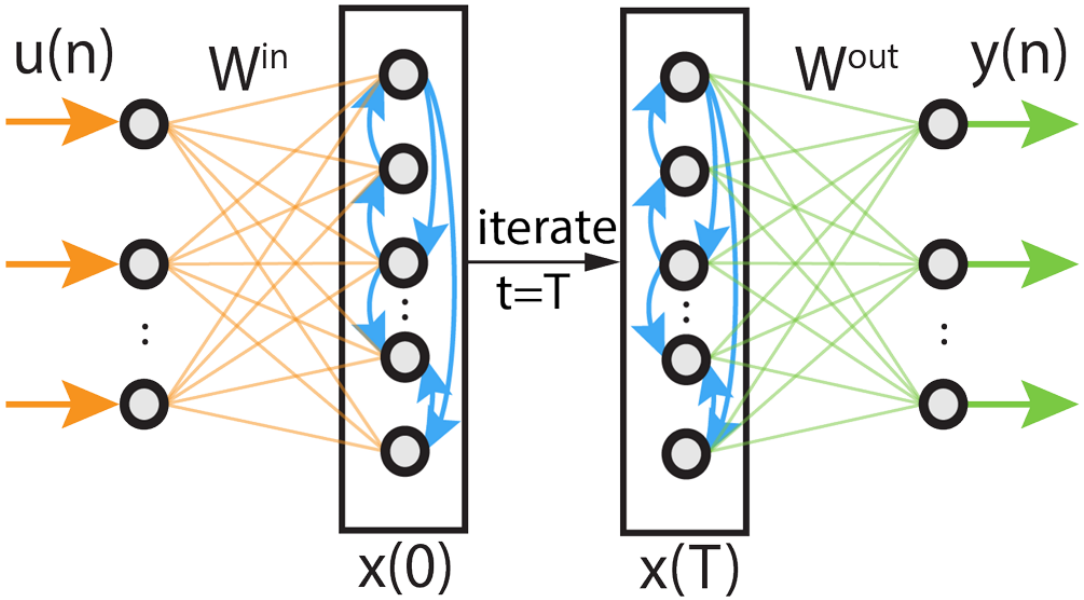
Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2020). *Science Advances* 6 (46), eabb3989.

Transition probability and duration for each pattern are designed successfully!

Liu, S., Akashi, N., Huang, Q., Kuniyoshi, Y., & Nakajima, K. (2024).
Exploiting Chaotic Dynamics as Deep Neural Networks.
arXiv preprint arXiv:2406.02580.

Feed-forward ESN

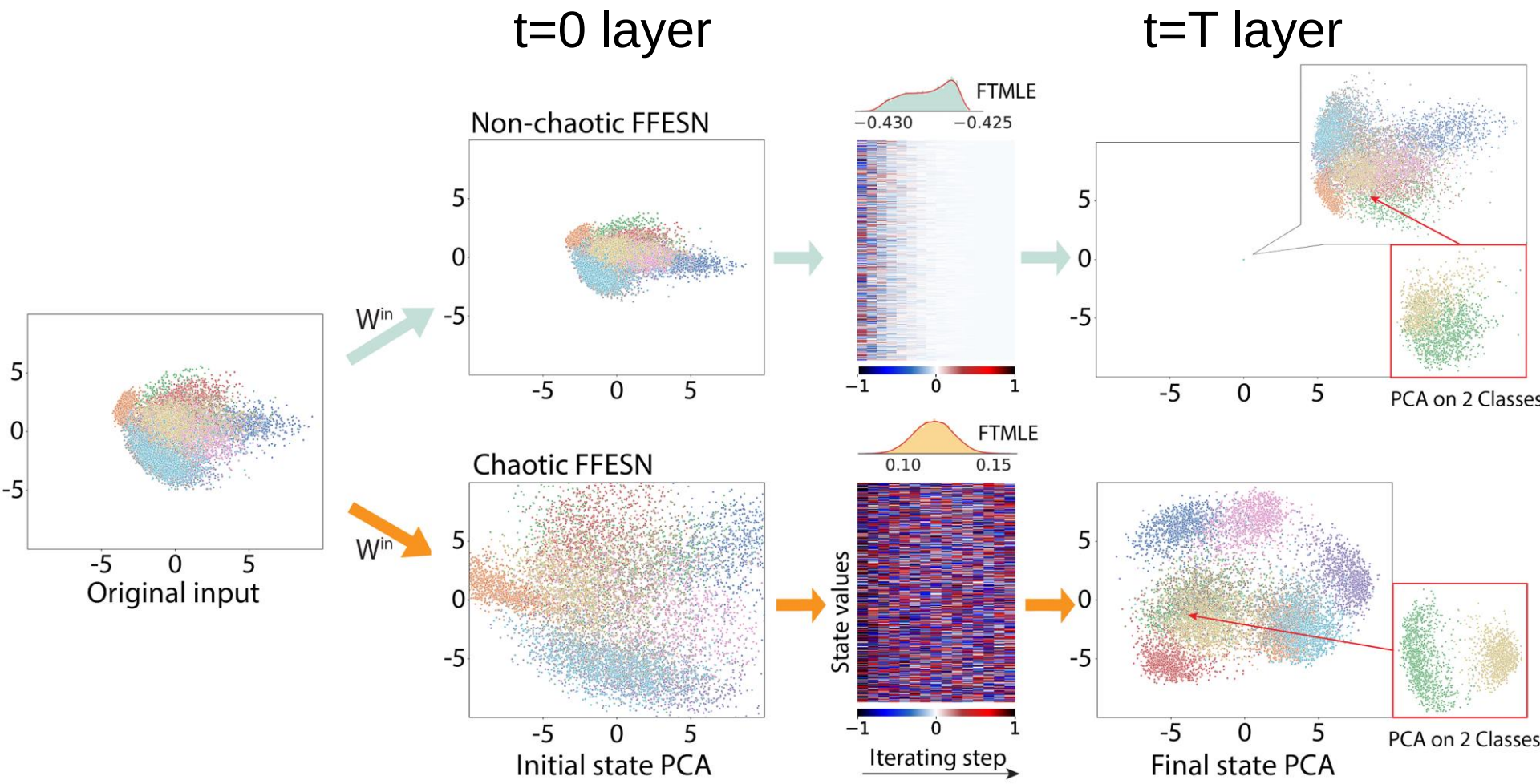
$$x(t+1) = \mathcal{F}(Wx(t))$$



$$x(0) = W_{\text{in}}u \ ,$$
$$x(i+1) = f(W'x(i)) \ ,$$
$$y = W_{\text{out}}x(n) \ .$$

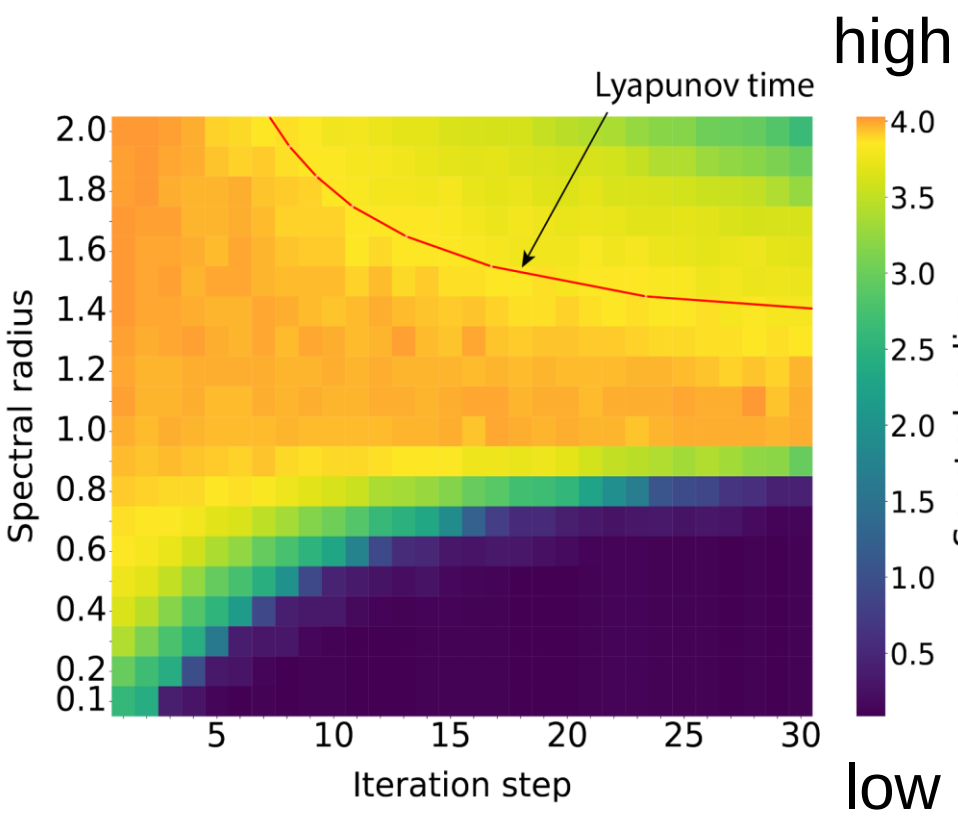
- Iteration timesteps as layers of deep neural networks (DNN)!
- W_{in}, W_{out} are trained through BP
- W_{in} determines how to set the initial conditions

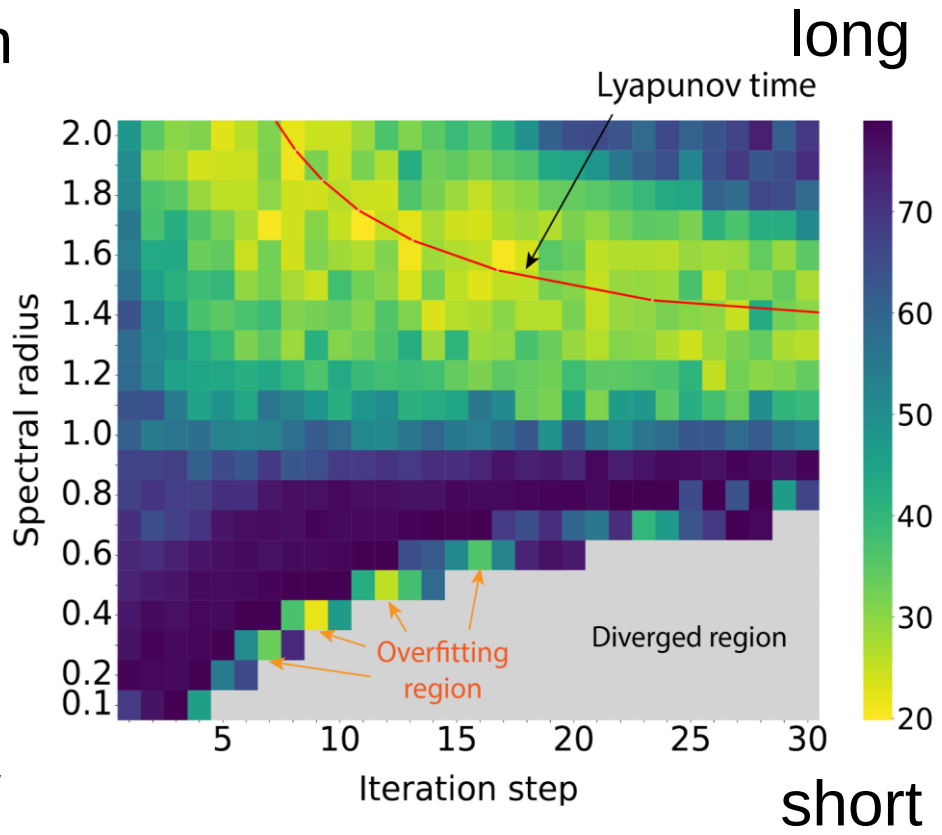
Implementing MNIST classification task



- Memory of initial conditions are rather required!
 (= broken down of ESP)
- Chaos generates high separability!

Performance and convergence time





- Lyapunov time characterizes the optimal timesteps (layers) of the network in chaotic region

High performance similar to DNN

System	High.(%)	Avg.(%)
Linear Regression	92.52	92.48
MLP ($N = 500$)	97.53	97.45
FFESN ($\rho = 0.9$, $N = 500$, $T = 1$)	98.16	98.08
FFESN ($\rho = 1.0$, $N = 500$, $T = 1$)	98.26	98.12
FFESN ($\rho = 1.8$, $N = 500$, $T = 2$)	98.34	98.23
Lorenz 96 ($F = 0.5$, $N = 500$, $T = 0.4$)	98.29	98.17
Lorenz 96 ($F = 4.75$, $N = 500$, $T = 0.2$)	98.20	98.02
CNN ($N = 600$)	98.77	98.69
CSTOs ($N = 600$, $A_{\mathrm{cp}} = 17.8$ Oe, $T = 300$ ps)	98.43	98.33
Deep CSTOs ($A_{\mathrm{cp}} = \{10, 0.1, 10\}$, $T = 100$ ps)	98.56	98.33
CNN + CSTOs ($N = 600$, cp $= 745$ Oe, $T = 200$ ps)	**99.05**	**99.00**

MLP and CNN all have only one hidden layer with N neurons.

- Chaotic dynamics is useful!

Thank you!

I would like to acknowledge all the collaborators, lab members, and RC seminar members!