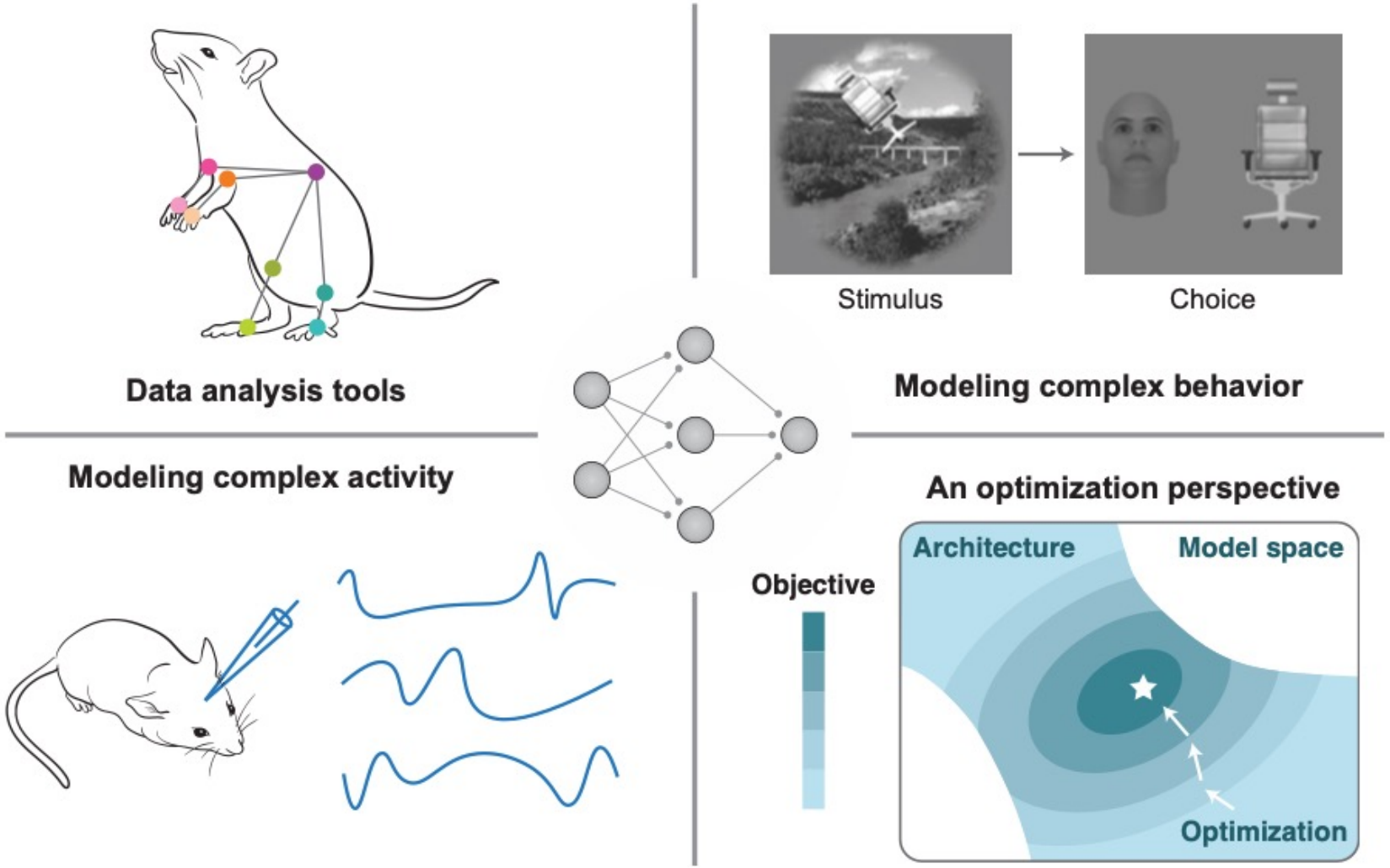
Reservoir Computing

张天秋

Introduction



Many recent reviews/opinions on this topic
Richards et al. 2019, Zador 2019, Sinz et al 2019, Hasson, Nastase, Goldstein 2020, Ma & Peters 2020, …

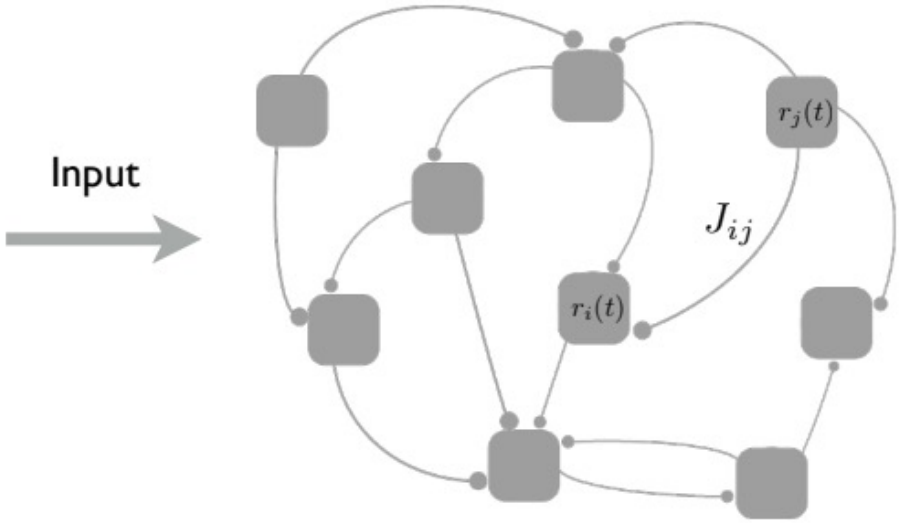
Figure from Yang & Wang *Neuron* 2020

From Robert Yang

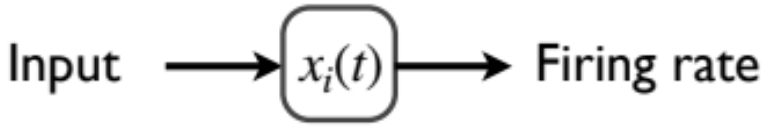
Introduction

北京大学 PEKING UNIVERSITY

RNN



- Individual neuron



Activation of unit i:

$$\tau \frac{dx_i}{dt} = -x_i + I_i^{(tot)}(t)$$

Firing rate:

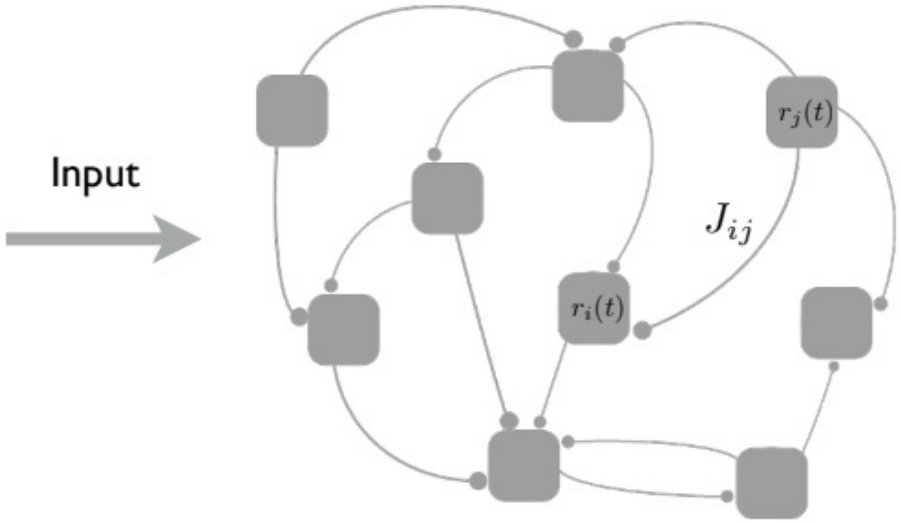
$$r_i(t) = \phi(x_i)$$

From Srdjan Ostojic

Introduction

北京大学 PEKING UNIVERSITY

RNN



- Connecting different units

Input to unit i from unit j:

$$I_{j\to i} = J_{ij}\, r_j(t)$$

Total input to unit i:

$$I_i^{(tot)} = \sum_{j=1}^{N} J_{ij}\, r_j(t) + I_i^{(ext)}$$

From Srdjan Ostojic

Introduction

北京大学 PEKING UNIVERSITY

RNN

Activation of unit i:

recurrent input external input

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij}\,\phi(x_j) + I_i^{(ext)}(t)$$

coupling strength from unit j to i

rate of unit j

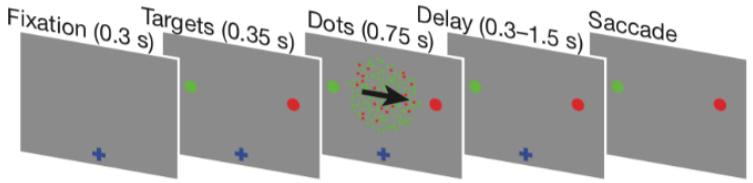
→ Connectivity matrix

$$r_j(t) = \phi(x_j)$$

From Srdjan Ostojic

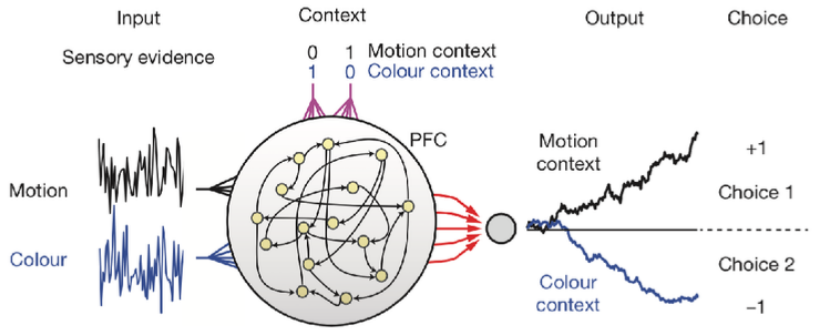
Introduction

Learning problem: Cognitive task



Mante*, Sussillo*, Shenoy, Newsome *Nature* 2013

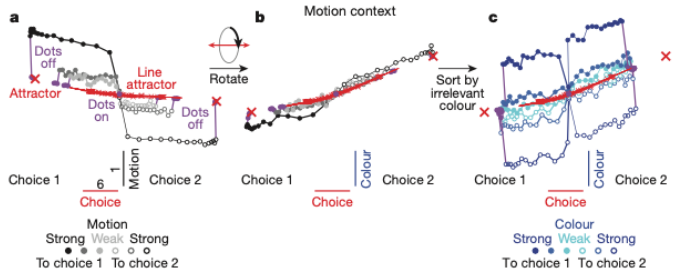
Architecture: Recurrent neural network (RNN)



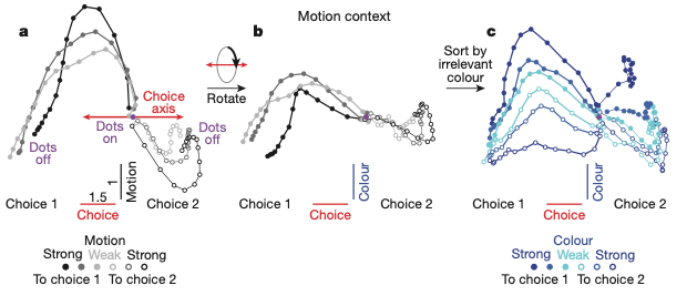
Learning rule: Gradient-based methods

Model/Data analysis: Representation/Dynamics/etc.

Recurrent network



Monkey PFC Data



A classical paradigm

Single task + Single RNN + Backpropagation = Dynamical/Mechanistic model of cognition?

Dating back two decades, e.g. Botvinick & Plaut *Psy. Rev.* 2004,
More recently, Sussillo, Barak, Rajan, Buonomano, many others

From Robert Yang

CONTENTS

01 Echo state machine

02 Constraints of echo state machine

03 Training of echo state machine

04 Echo state machine programming

05 Applications

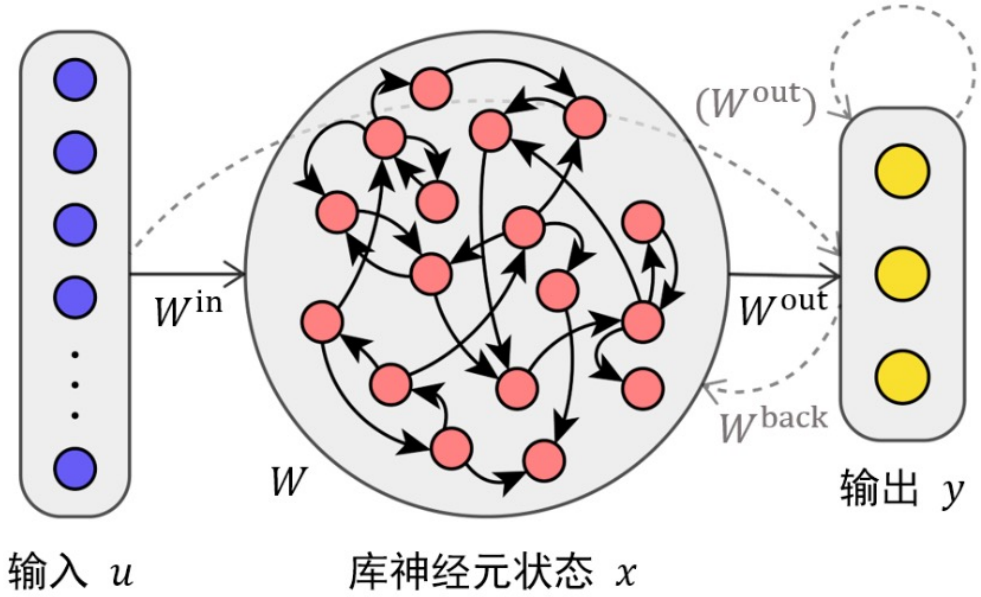
北京大学
PEKING UNIVERSITY

北京大学
PEKING UNIVERSITY

01
Echo state machine

Echo state machine

Echo State Networks (ESNs) are applied to supervised temporal machine learning tasks where for a given training input signal $x(n)$ a desired target output signal $y^{target}(n)$ is known. Here $n = 1, \ldots, T$ is the discrete time and T is the number of data points in the training dataset.



The "echo state" approach to analysing and training recurrent neural networks – with an Erratum note[1]

Herbert Jaeger
Fraunhofer Institute for Autonomous Intelligent Systems

January 26, 2010

$$\mathbf{x}(n+1) = f(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\text{back}}\mathbf{y}(n))$$

$$\mathbf{y}(n+1) = \mathbf{W}^{\text{out}}(\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n))$$

What is echo state?

北京大学 PEKING UNIVERSITY

- For an RNN, the state of its internal neurons reflects the historical information of the external inputs.

- Assuming that the updates of the network are discrete, the external input at the nth moment is u(n) and the neuron state is x(n), then x(n) should be determined by u(n), u(n - 1), ... uniquely determined. At this point, x(n) can be regarded as an "echo" of the historical input signals.

Definition 1 *Assume standard compactness conditions. Assume that the network has no output feedback connections. Then, the network has* echo states, *if the network state* $\mathbf{x}(n)$ *is uniquely determined by any left-infinite input sequence* $\bar{\mathbf{u}}^{-\infty}$. *More precisely, this means that for every input sequence* $\ldots, \mathbf{u}(n-1), \mathbf{u}(n) \in U^{-\mathbb{N}}$, *for all state sequences* $\ldots, \mathbf{x}(n-1), \mathbf{x}(n)$ *and* $\mathbf{x}'(n-1), \mathbf{x}'(n) \in A^{-\mathbb{N}}$, *where* $\mathbf{x}(i) = T(\mathbf{x}(i-1), \mathbf{u}(i))$ *and* $\mathbf{x}'(i) = T(\mathbf{x}'(i-1), \mathbf{u}(i))$, *it holds that* $\mathbf{x}(n) = \mathbf{x}'(n)$.

Jaeger, 2001

Echo state machine with leaky integrator

ESNs use an RNN type with leaky-integrated discrete-time continuous-value units. The typical update equations are

$$\hat{h}(n) = \tanh(W^{in}x(n) + W^{rec}h(n-1) + W^{fb}y(n-1) + b^{rec})$$
$$h(n) = (1-\alpha)x(n-1) + \alpha\hat{h}(n)$$

where $h(n)$ is a vector of reservoir neuron activations, W^{in} and W^{rec} are the input and recurrent weight matrices respectively, and $\alpha \in (0, 1]$ is the leaking rate. The model is also sometimes used without the leaky integration, which is a special case of $\alpha = 1$.

The linear readout layer is defined as

$$y(n) = W^{out}h(n) + b^{out}$$

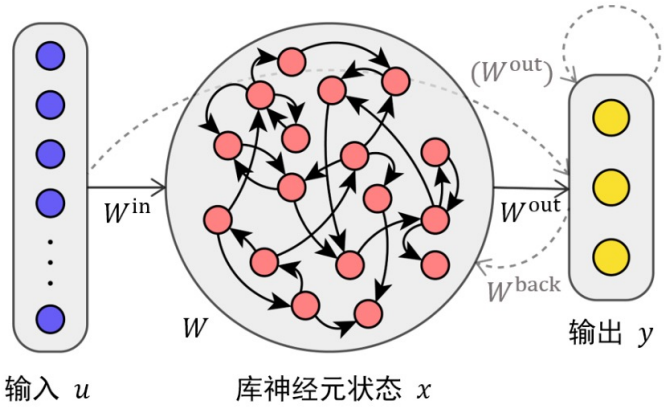
where $y(n)$ is network output, W^{out} the output weight matrix, and b^{out} is the output bias.

北京大学
PEKING UNIVERSITY

02

Constraints of echo state machine

Echo state property

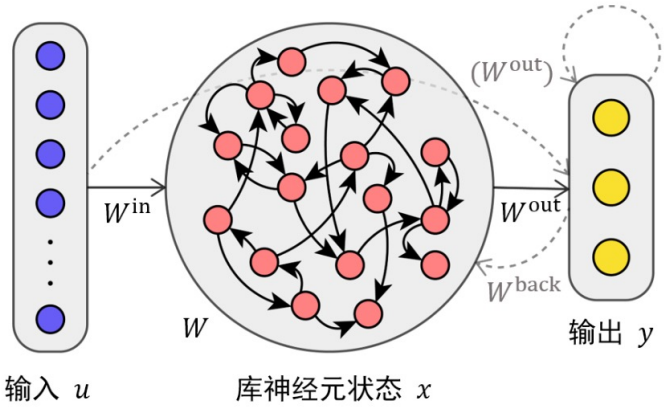


Theorem 1. For the echo state network defined above, the network will be echoey as long as the maximum singular value σ_{max} < 1 of the recurrent connectivity matrix W .

Provement:

$$
\begin{aligned}
d(\mathbf{x}(n+1), \mathbf{x}'(n+1)) &= d(T(\mathbf{x}(n), \mathbf{u}(n+1)), T(\mathbf{x}'(n), \mathbf{u}(n+1))) \\
&= d(f(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n)), f(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}'(n))) \\
&\leq d(\mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n), \mathbf{W}^{\text{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}'(n)) \\
&= d(\mathbf{W}\mathbf{x}(n), \mathbf{W}\mathbf{x}'(n)) \\
&= ||\mathbf{W}(\mathbf{x}(n) - \mathbf{x}'(n))|| \\
&\leq \sigma_{\max}(\mathbf{W})\, d(\mathbf{x}(n), \mathbf{x}'(n))
\end{aligned}
$$

Echo state property



Theorem 1. For the echo state network defined above, the network will be echoey as long as the maximum singular value σ_{max} < 1 of the recurrent connectivity matrix W .

Theorem 2. For the echo state network defined above, as long as the spectral radius $|\lambda_{max}|$ of the recurrent connection matrix W > 1, then the network must not be echogenic. The spectral radius of the matrix is the absolute value of the largest eigenvalue λ_{max}.

Echo state property

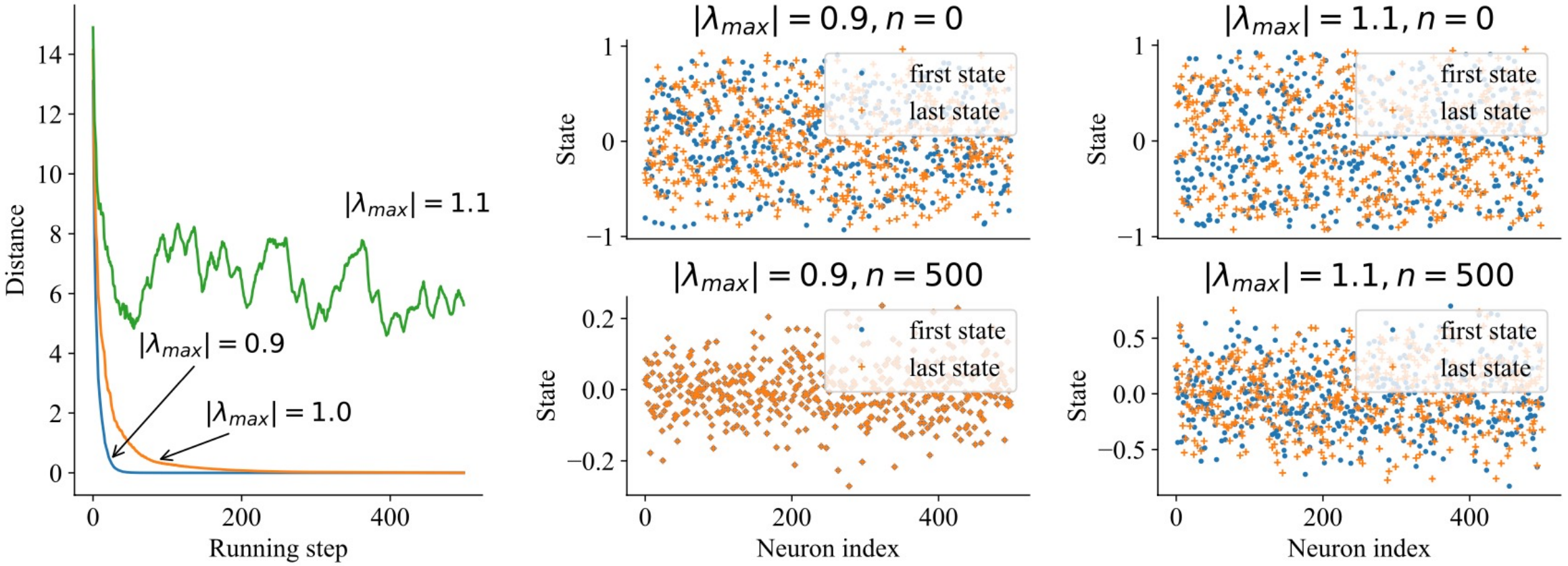
Using these two theorems, how should we initialize W so that the network has an echo property?

If we scale W, i.e., multiply it by a scaling factor α, then σ_{max} and $|\lambda_{max}|$ will also be scaled α.

For any square matrix, we have $\sigma_{max} \geq |\lambda_{max}|$. Therefore we set $\alpha_{min} = 1/\sigma_{max}(W)$, $\alpha_{max} = 1/|\lambda_{max}|(W)$. Then,

- If $\alpha < \alpha_{min}$, the network must have the echo state.
- If $\alpha > \alpha_{max}$, the network will not have the echo state.
- If$\alpha_{min} \leq \alpha \leq \alpha_{max}$, the network may have the echo state.

Echo state property



Global parameters of the reservoir

北京大学 PEKING UNIVERSITY

- The size N_x:
  - General wisdom: the bigger the reservoir, the better the obtainable performance.
  - Select global parameters with smaller reservoirs, then scale to bigger ones.
- Sparsity
- Distribution of nonzero elements:
  - Normal distribution
  - Uniform distribution
  - The width of the distributions does not matter
- spectral radius of W
  - scales the width of the distribution of its nonzero elements
  - determines how fast the influence of an input dies out in a reservoir with time, and how stable the reservoir activations are.
  - The spectral radius should be larger in tasks requiring longer memory of the input.
- Scaling(-s) to W^{in}:
  - For uniform distributed W^{in}, a is in the range of the interval $[-a; a]$.
  - For normal distributed W^{in}, one may take the standard deviation as a scaling measure.
- The leaking rate α

北京大学
PEKING UNIVERSITY

03

Training of echo state machine

Offline learning

The advantage of the echo state network is that it does not train recurrent connections within the reservoir, but only the readout layer from the reservoir to the output.

Ridge regression

$$\epsilon_{\text{train}}(n) = \mathbf{y}(n) - \hat{\mathbf{y}}(n)$$
$$= \mathbf{y}(n) - \mathbf{W}^{\text{out}}\mathbf{x}(n)$$

$$L_{\text{ridge}} = \frac{1}{N}\sum_{i=1}^{N} \epsilon_{\text{train}}^2(i) + \alpha||\mathbf{W}^{\text{out}}||^2$$

$$W^{out} = Y^{target}X^T(XX^T + \beta I)^{-1}$$

```
trainer = bp.OfflineTrainer(model, fit_method=bp.algorithms.RidgeRegression(1e-7), dt=dt)
```

Online learning

The training data is passed to the trainer in a certain sequence (e.g., time series), and the trainer continuously learns based on the new incoming data.

Recursive Least Squares (RLS) algorithm

$$E(\mathbf{y}, \mathbf{y}^{\text{target}}, n) = \frac{1}{N_{\text{y}}} \sum_{i=1}^{N_{\text{y}}} \sum_{j=1}^{n} \lambda^{n-j} \left(y_i(j) - y_i^{\text{target}}(j) \right)^2,$$

```
trainer = bp.OnlineTrainer(model, fit_method=bp.algorithms.RLS(), dt=dt)
```

Other methods: FORCE learning (Sussillo, Abbott, *Neuron*, 2009)

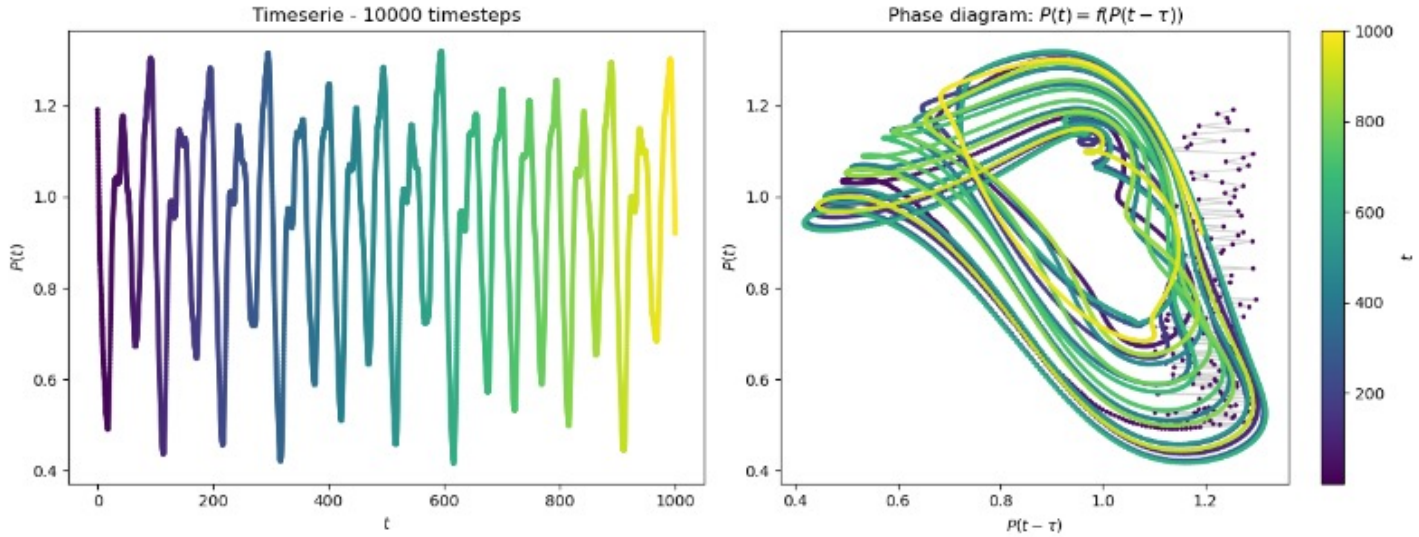
Dataset:

Mackey-Glass equation are a set of delayed differential equations describing the temporal behaviour of different physiological signal, for example, the relative quantity of mature blood cells over time.

The equations are defined as:

$$\frac{dP(t)}{dt} = \frac{\beta P(t-\tau)}{1+P(t-\tau)^n} - \gamma P(t)$$

where $\beta = 0.2$, $\gamma = 0.1$, $n = 10$, and the time delay $\tau = 17$. τ controls the chaotic behaviour of the equations (the higher it is, the more chaotic the timeserie becomes. $\tau = 17$ already gives good chaotic results.)



Dataset:

北京大学 PEKING UNIVERSITY

Neuromorphic and Cognitive Datasets for Brain Dynamics Modeling

python 3.7 | 3.8 | 3.9 | 3.10 | 3.1

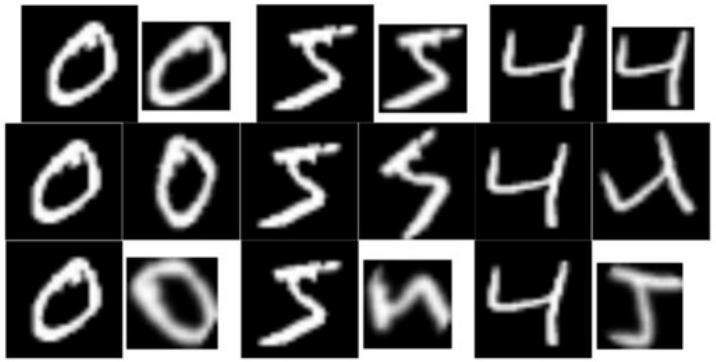
`brainpy-datasets` aims to provide datasets for brain dynamic
neuromorphic data, cognitive task or environment, traditional vi

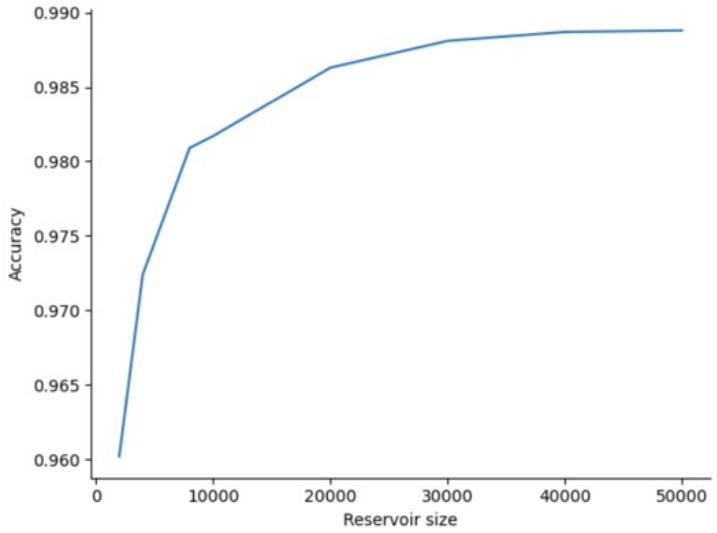
```python
from brainpy_datasets._src.chaos.base import (ChaosDataset as ChaosDataset)
from brainpy_datasets._src.chaos.one_dim_systems import (LogisticMap as LogisticMap)
from brainpy_datasets._src.chaos.three_dim_systems import (
  ModifiedLuChenEq as ModifiedLuChenEq,
  LorenzEq as LorenzEq,
  RabinovichFabrikantEq as RabinovichFabrikantEq,
  ChenChaoticEq as ChenChaoticEq,
  LuChenEq as LuChenEq,
  ChuaChaoticEq as ChuaChaoticEq,
  ModifiedChuaEq as ModifiedChuaEq,
  ModifiedLorenzEq as ModifiedLorenzEq,
  DoubleScrollEq as DoubleScrollEq,
)
from brainpy_datasets._src.chaos.two_dim_systems import (
  HenonMap as HenonMap,
  MackeyGlassEq as MackeyGlassEq,
  PWLDuffuingEq as PWLDuffuingEq,
)
```

```python
from brainpy_datasets._src.cognitive.decision_making import (
  RateSingleContextDecisionMaking,
  RateContextDecisionMaking,
  RatePerceptualDecisionMaking,
  RatePulseDecisionMaking,
  RatePerceptualDecisionMakingDelayResponse,
)
from brainpy_datasets._src.cognitive.others import (
  RateAntiReach,
  RateReaching1D,
)
from brainpy_datasets._src.cognitive.reasoning import (
  RateHierarchicalReasoning,
  RateProbabilisticReasoning,
)
from brainpy_datasets._src.cognitive.working_memory import (
  RateDelayComparison,
  RateDelayMatchCategory,
  RateDelayMatchSample,
  RateDelayPairedAssociation,
  RateDualDelayMatchSample,
  RateGoNoGo,
  RateIntervalDiscrimination,
  RatePostDecisionWager,
  RateReadySetGo,
)
```

Other tasks

MNIST dataset

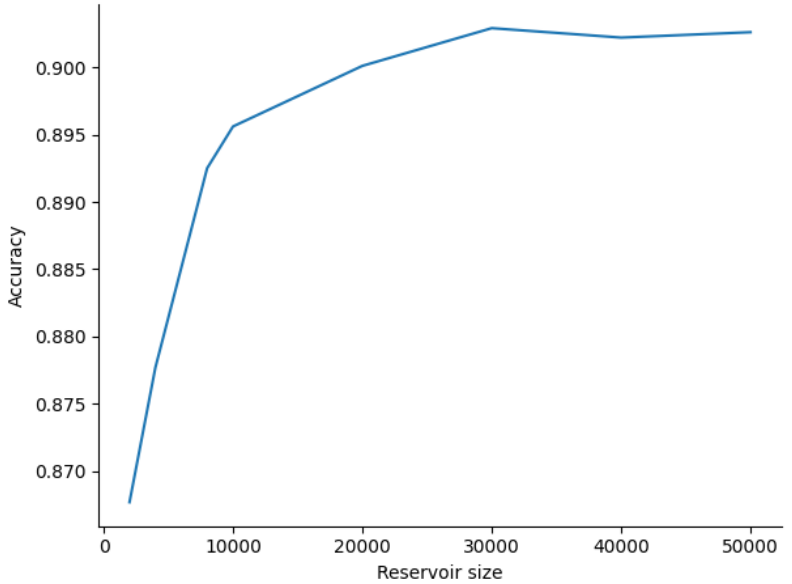




Two aspect:

- Running time
- Memory Usage

Fashion MNIST



北京大学
PEKING UNIVERSITY

04

Echo state machine programming

JIT connection operators

北京大学 PEKING UNIVERSITY

JIT connection operator

- Just-in-time randomly generated matrix.
- Support for Mat@Vec and Mat@Mat.
- Suppoer different random generation methods.(homogenous, uniform, normal)

```
import math, random

def jitconn_prob_homo(events, prob, weight, seed, outs):
  random.seed(seed)
  max_cdist = math.ceil(2/prob - 1)
  for event in events:
    if event:
      post_i = random.randint(1, max_cdist)
      outs[post_i] += weight
```

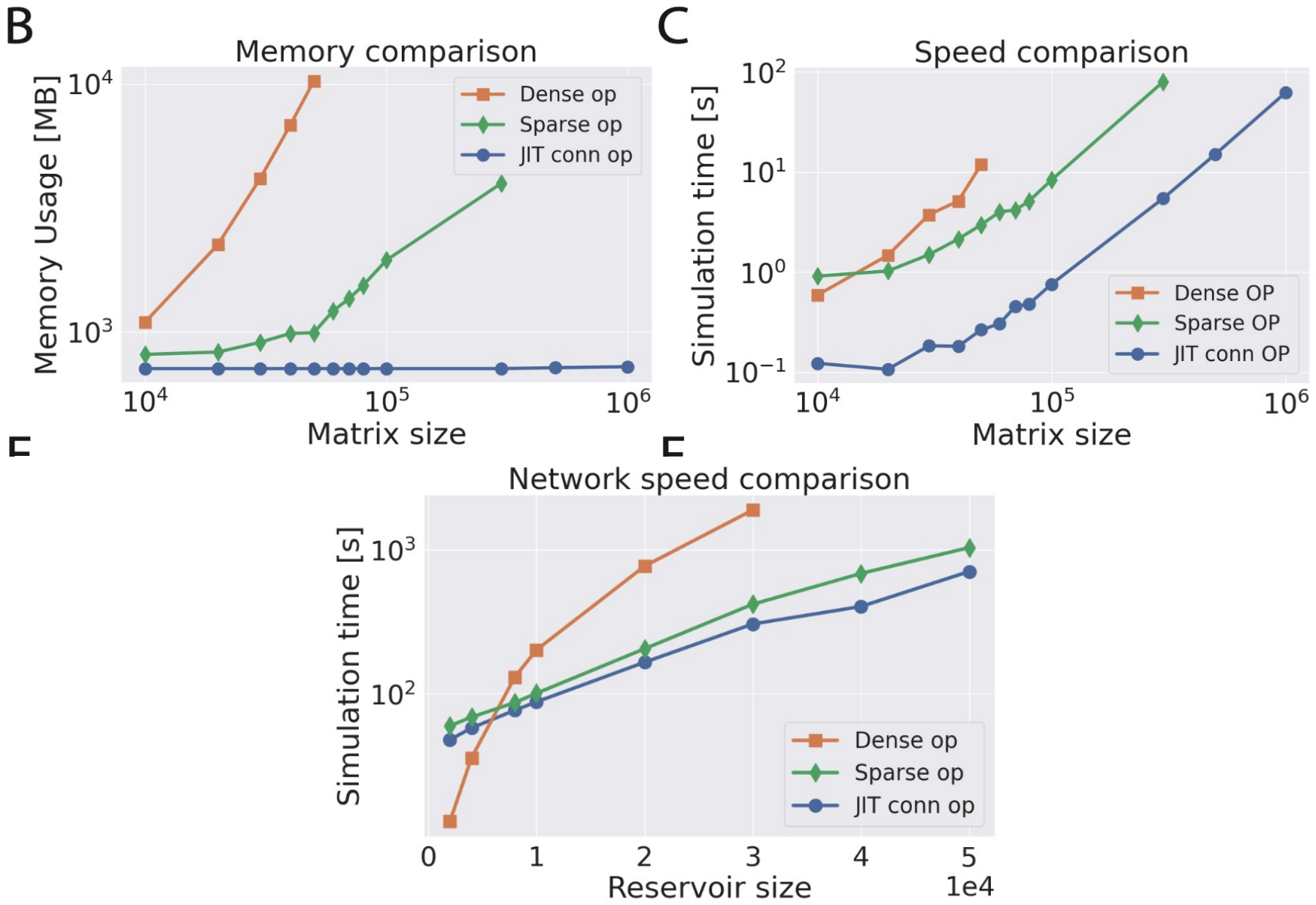
JIT connection operators

北京大学 PEKING UNIVERSITY

```python
from brainpy._src.math.jitconn import (
  event_mv_prob_homo as event_mv_prob_homo,
  event_mv_prob_uniform as event_mv_prob_uniform,
  event_mv_prob_normal as event_mv_prob_normal,

  mv_prob_homo as mv_prob_homo,
  mv_prob_uniform as mv_prob_uniform,
  mv_prob_normal as mv_prob_normal,
)
```

JIT connection operators



Building a ESN with BrainPy

```python
class ESN(bp.DynamicalSystemNS):
  def __init__(self, num_in, num_hidden, num_out, sr=1., leaky_rate=0.3,
               Win_initializer=bp.init.Uniform(0, 0.2)):
    super(ESN, self).__init__()
    self.r = bp.layers.Reservoir(
        num_in, num_hidden,
        Win_initializer=Win_initializer,
        spectral_radius=sr,
        leaky_rate=leaky_rate,
    )
    self.o = bp.layers.Dense(num_hidden, num_out, mode=bm.training_mode)

  def update(self, x):
    return x >> self.r >> self.o
```

```python
model = ESN(1, 100, 1)
model.reset_state(1)
trainer = bp.RidgeTrainer(model, alpha=1e-6)
```

```python
# warmup
_ = trainer.predict(x_warm)
```

```python
# train
_ = trainer.fit([x_train, y_train])
```

北京大学
PEKING UNIVERSITY

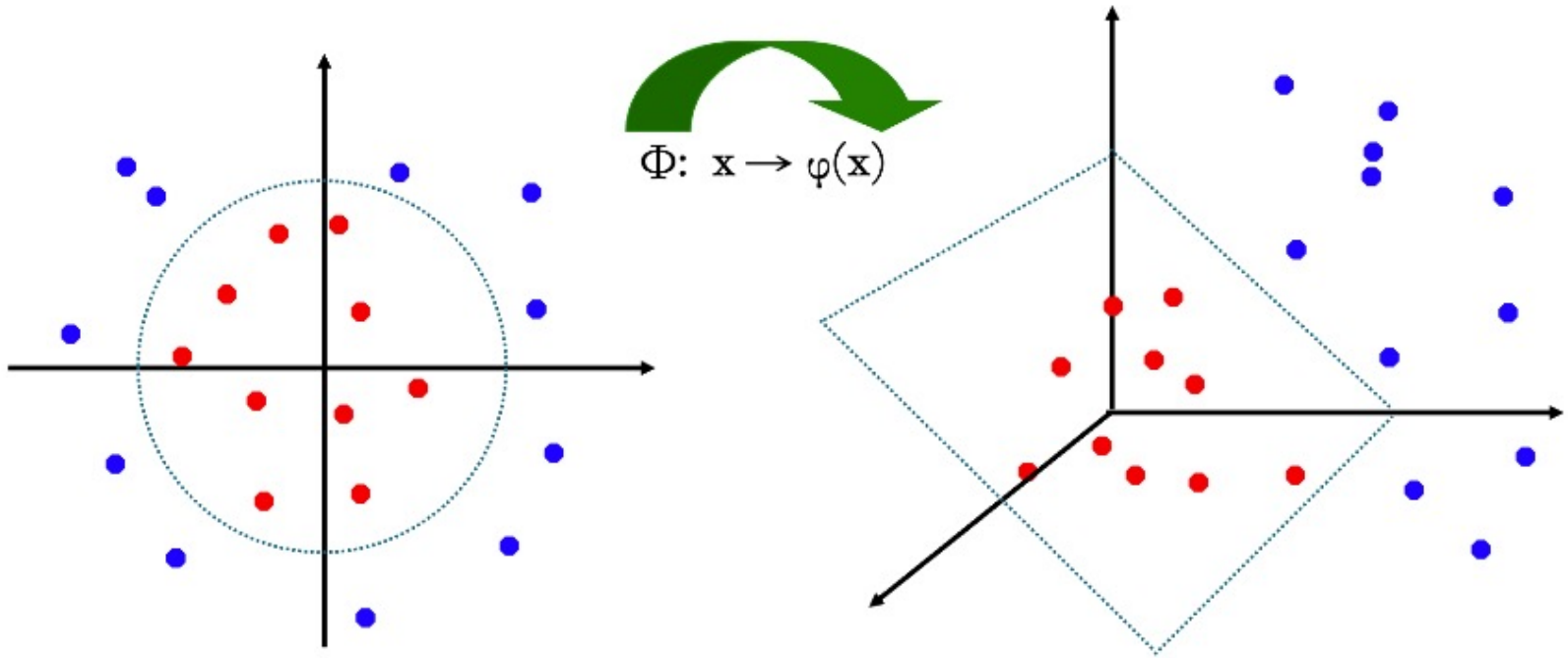
05
Applications

From the perspective of kernel methods

北京大学 PEKING UNIVERSITY

Non-linear SVMs: Kernel Mapping

- General idea: the original space is mapped to a higher-dimensional space where data becomes linearly separable:



From the perspective of kernel methods

北京大学 PEKING UNIVERSITY

The kernel trick

- The SVM only relies on the inner-product between vectors $x_i \cdot x_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: x \rightarrow \varphi(x)$, the inner-product becomes:

$$K(x_i,x_j)= \varphi(x_i) \cdot \varphi(x_j)$$

- $K(x_i,x_j)$ is called the kernel function.
- For SVM, we only need specify the kernel $K(x_i,x_j)$, without need to know the corresponding non-linear mapping, $\varphi(x)$.

From the perspective of kernel methods

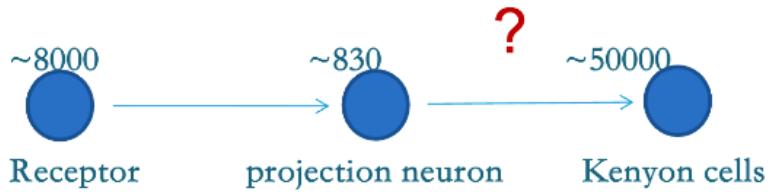
北京大学 PEKING UNIVERSITY

Key ideas of SVM

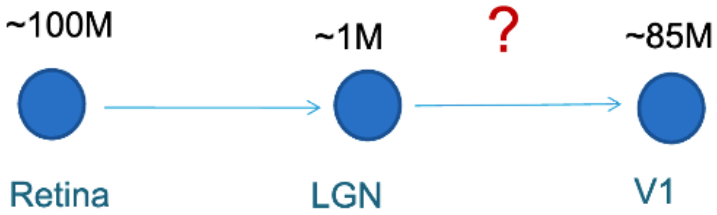
- A hyper-plane classifier
- A hyper-plane that maximizes the margin between two classes of data points
- Using a kernel function to map the original data into a high-dimensional space
- Soft-margin to accommodate noises

Kernel methods in neural system?

- Olfactory system of Locust



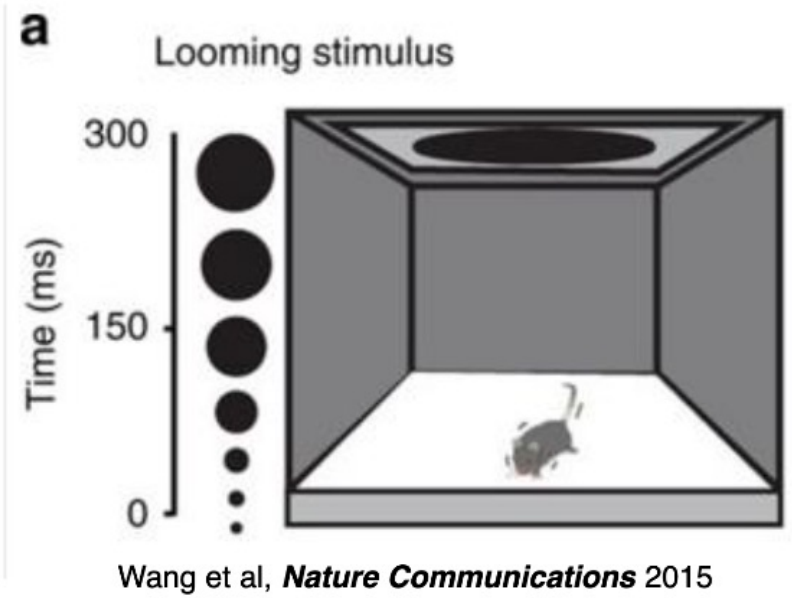
- Visual system of primates



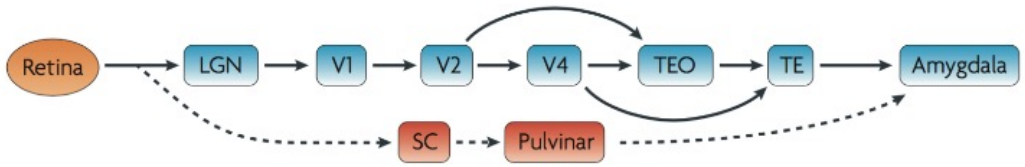
Subcortical pathway for rapid motion processing

The first two stages of subcortical visual pathway:
Retina → superior colliculus

Dimension expansion: retinal network.

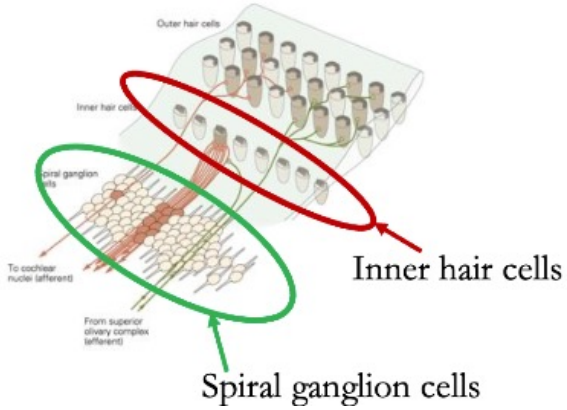


Wang et al, ***Nature Communications*** 2015

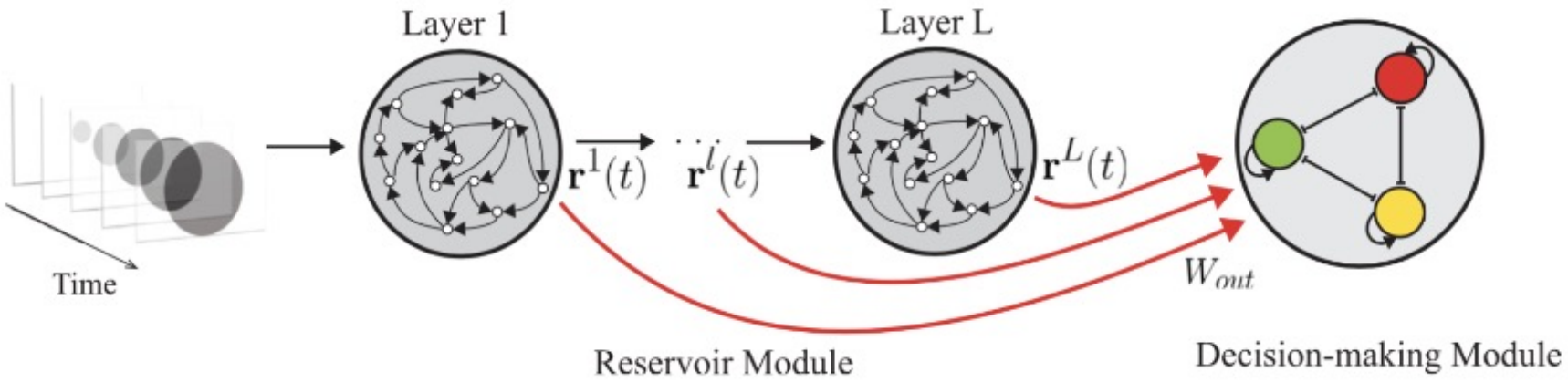


The first two stages of primary auditory pathway:
Inner Ear → Cochlear Nuclei

Dimension expansion: Each inner hair cell synapses on about 10 spiral ganglion cells; each spiral ganglion cell receives input from only 1 inner hair cell.



Spatial-temporal tasks

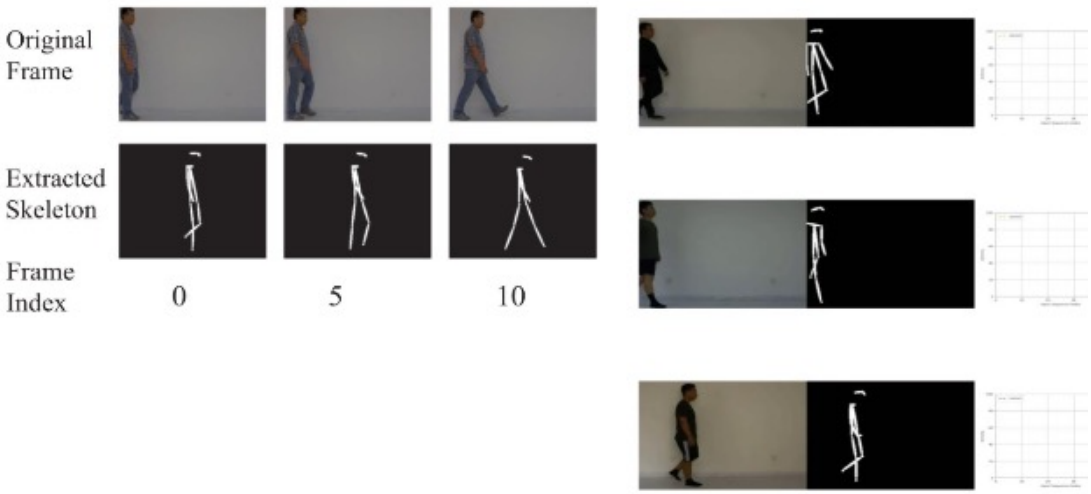


Dimension expansion

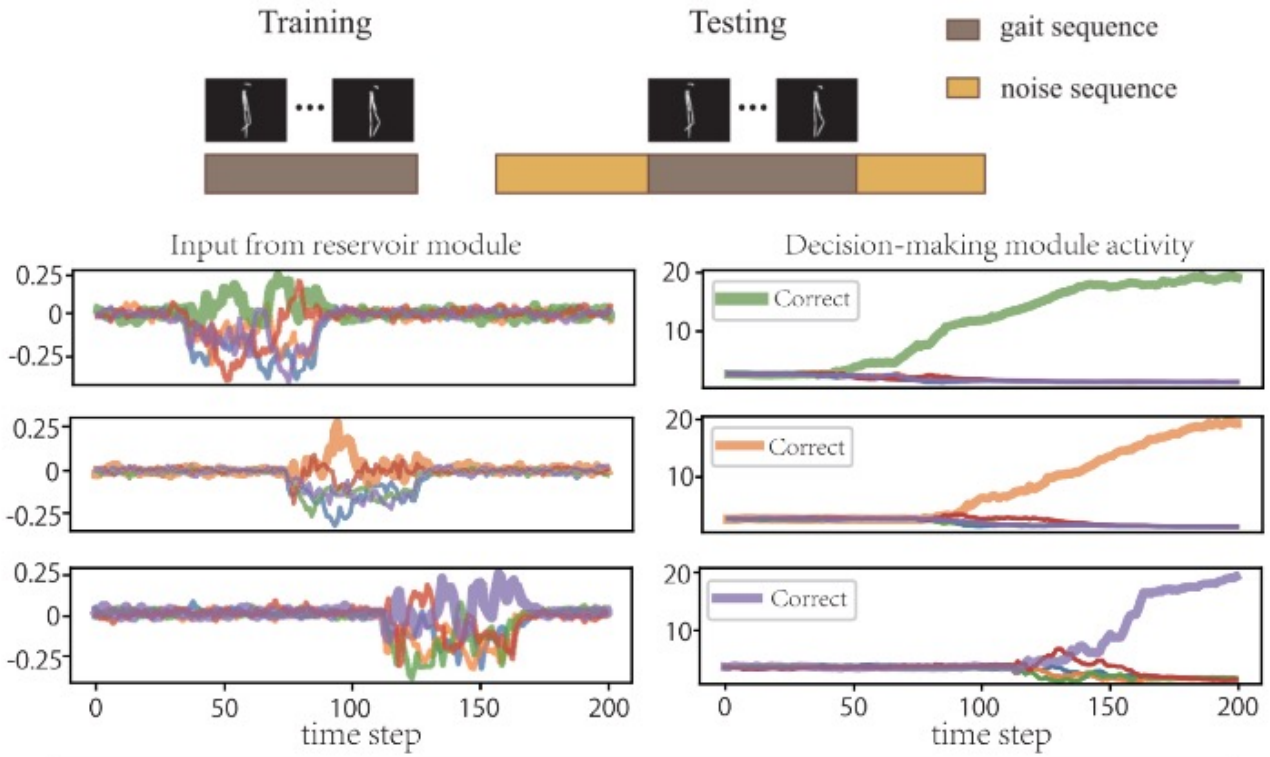
Evidence accumulation

Lin et al. Neural Networks 2021

Gait recognition



Model	5 classes	10 classes	15 classes
LSTM(20)	92.4 ± 3.9	83.9 ± 3.3	79.5 ± 3.9
LSTM(50)	94.3 ± 2.0	85.7 ± 2.9	81.5 ± 3.1
LSTM(100)	90.6 ± 3.6	79.5 ± 3.2	76.6 ± 2.1
GRU(20)	92.4 ± 2.5	82.2 ± 3.7	81.3 ± 2.9
GRU(50)	95.4 ± 2.1	88.2 ± 3.2	85.7 ± 1.8
GRU(100)	96.4 ± 2.0	90.5 ± 2.1	89.7 ± 1.9
RDMN	**98.3 ± 1.0**	**93.4 ± 2.2**	**92.4 ± 2.5**



Model	5 classes	10 classes	15 classes
Linear	93.3 ± 3.0	79.6 ± 2.7	76.4 ± 2.5
RDMN	**98.3 ± 0.7**	**93.2 ± 2.4**	**92.3 ± 2.7**

Lin et al. Neural Networks 2021

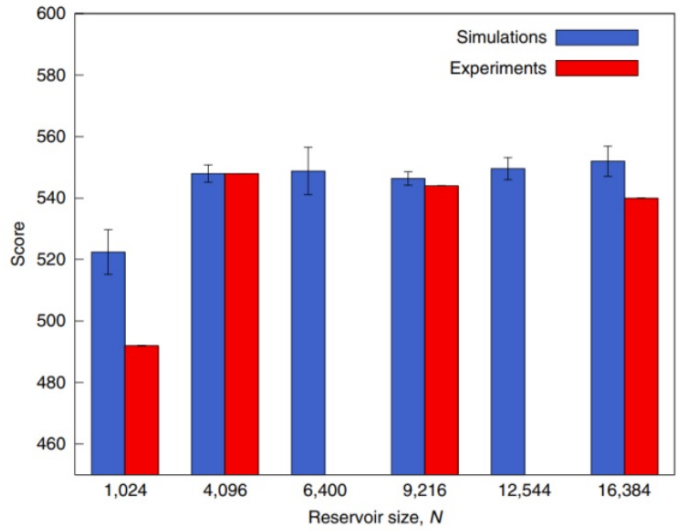
Spatial-temporal tasks

ARTICLES
https://doi.org/10.1038/s42256-019-0110-8
nature machine intelligence

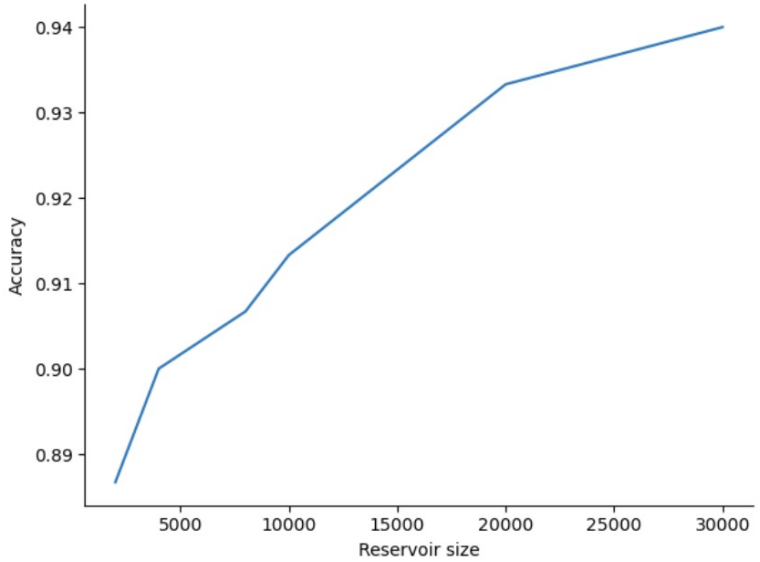
Human action recognition with a large-scale brain-inspired photonic computer

Piotr Antonik[1*], Nicolas Marsal[1], Daniel Brunner[2] and Damien Rontani[1*]

We report a classification accuracy of 91.3%, comparable to state-of-the-art digital implementations.
The experimental set-up can accommodate a reservoir of 16,384 nodes, while the physical limitation of the concept is set to be as high as 262,144 neurons.



KTH dataset



Related materials

Liquid state machine

Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations

Wolfgang Maass
maass@igi.tu-graz.ac.at
Thomas Natschläger
tnatschl@igi.tu-graz.ac.at
Institute for Theoretical Computer Science, Technische Universität Graz; A-8010 Graz, Austria

Henry Markram
henry.markram@epfl.ch
Brain Mind Institute, Ecole Polytechnique Federale de Lausanne, CH-1015 Lausanne, Switzerland

A **liquid state machine** (**LSM**) is a type of reservoir computer that uses a spiking neural network.

Advantages

1. Circuits are not hard coded to perform a specific task.
2. Continuous time inputs are handled "naturally".
3. Computations on various time scales can be done using the same network.
4. The same network can perform multiple computations.

Disadvantages

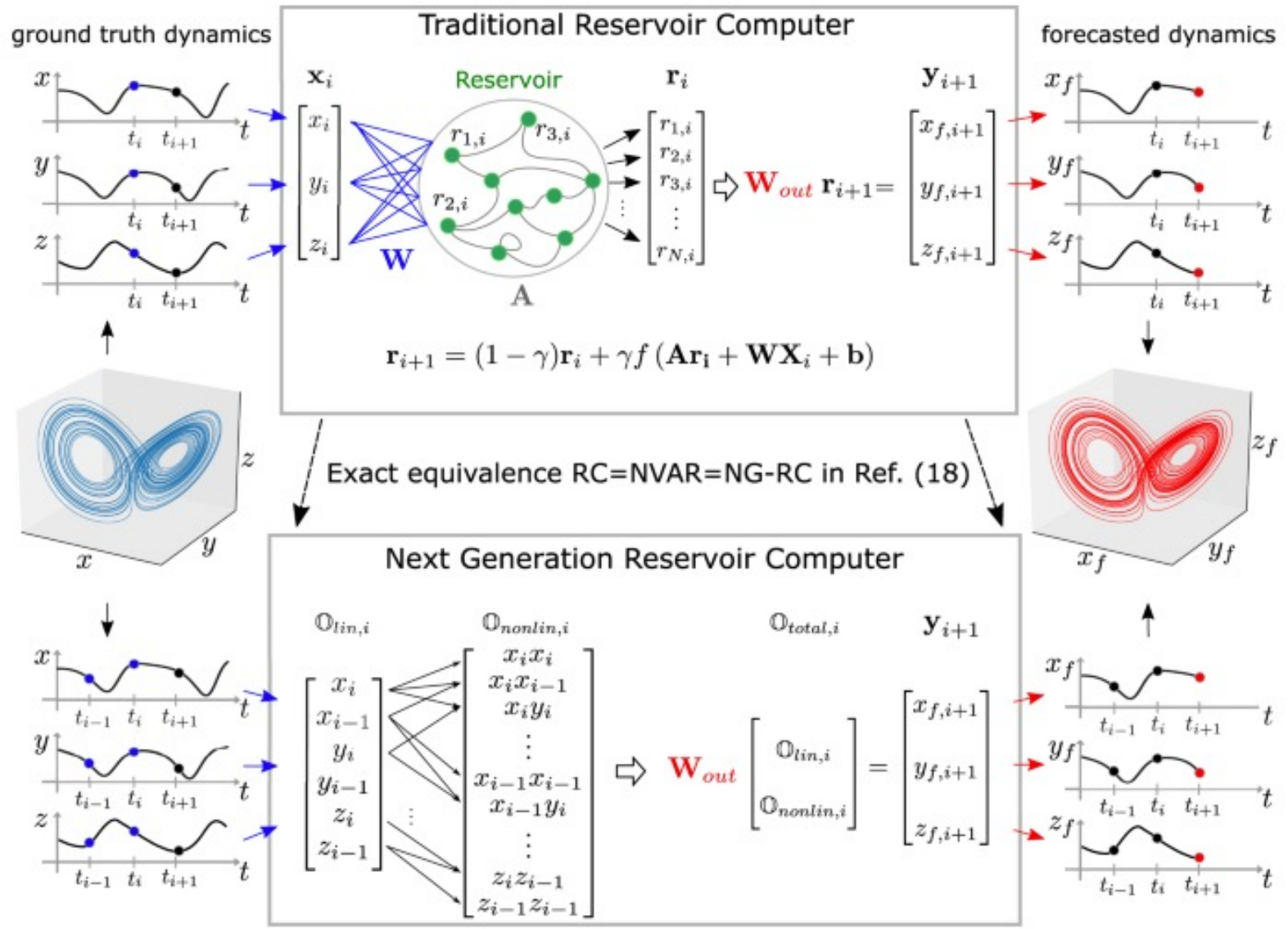
1. LSMs don't actually explain how the brain functions. At best they can replicate some parts of brain functionality.
2. There is no guaranteed way to dissect a working network and figure out how or what computations are being performed.
3. There is very little control over the process.

Related materials

Next generat

You can realize

- Nonlinear nc
- Linear reserv



, W. A. S. Barbosa,
outing,'

THANK YOU

北京大学
PEKING UNIVERSITY