Reservoir Computing

Andrea Ceni
University of Pisa, Italy

Contact Information

Andrea Ceni, Ph.D.
Research Fellow,
University of Pisa

Research interests:

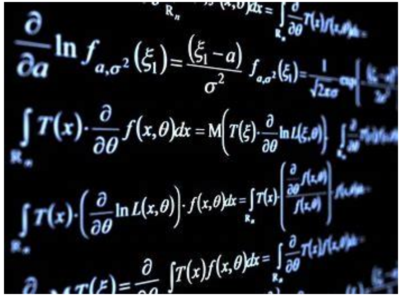
Learning long-term dependencies with RNNs
Reservoir computing – Randomised NNs
Convolutional neural networks
Modular composition of NNs
Linking cognitive features to attractor’s geometries
State space models for sequence learning
Graph Neural Networks
Criticality in complex systems
Dynamical recurrent neural models (bio-inspired)

Do you like these topics? Email me!

email: andrea.ceni@di.unipi.it

My path

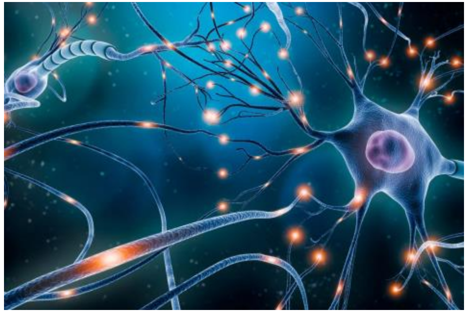
UNIVERSITÀ DEGLI STUDI FIRENZE



Neurosciences, neurologie, psychiatrie

Inserm

Institut national de la santé et de la recherche médicale



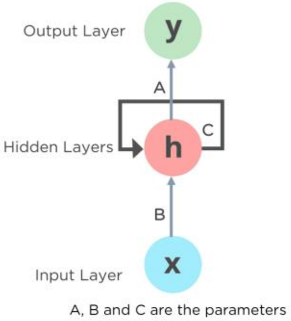
CPT

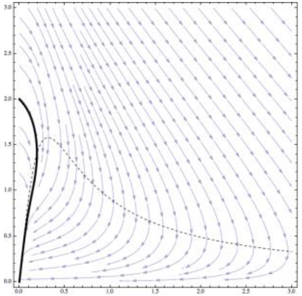
Centre de Physique Théorique

Aix*Marseille université

Initiative d'excellence

University of Exeter



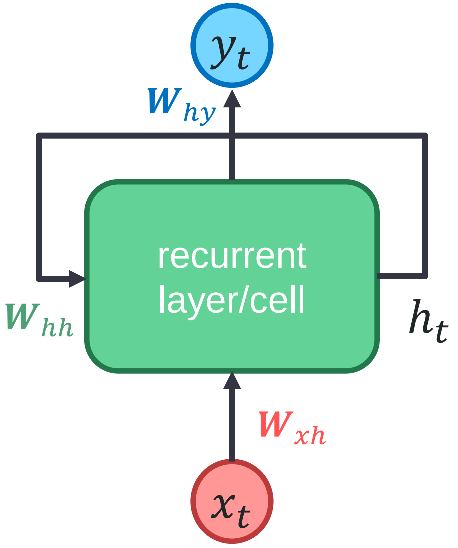


Reservoir Computing

=

Extremely efficient way of designing and training RNNs

Recurrent Neural Networks



- **State update:**

input

previous state

$$h_t = tanh(x_t\,W_{xh} + h_{t-1}\,W_{hh})$$

state

input weight matrix

recurrent weight matrix

- **Output function:**

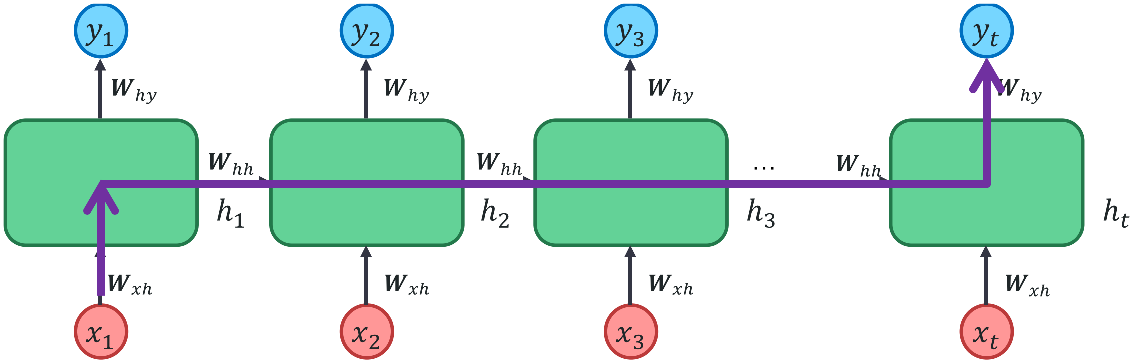
$$y_t = h_t\,W_{hy}$$

output

state

output weight matrix

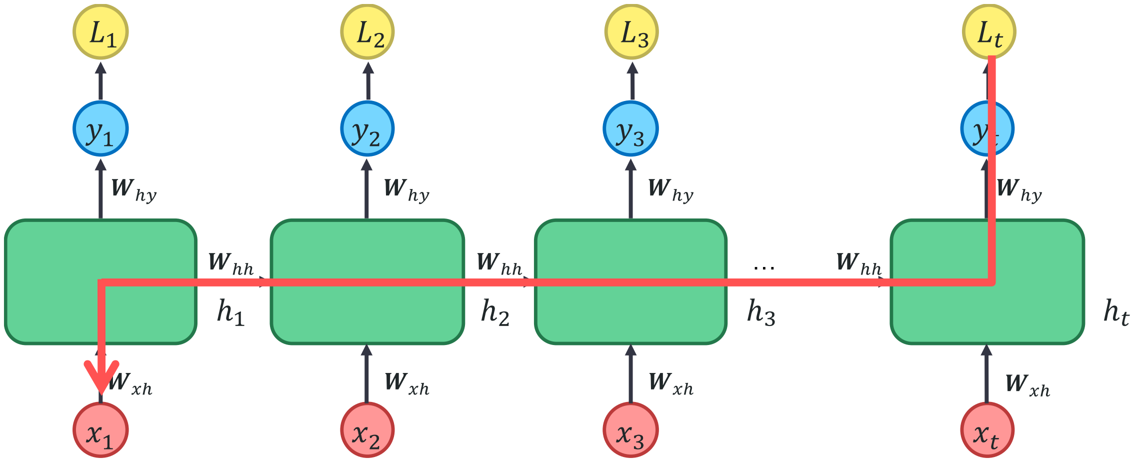
Forward Computation



Fading/Exploding memory:

- the influence of inputs far in the past vanishes/explodes in the current state
- many (non-linear) transformations

Backpropagation Through Time (BPTT)



Gradient Propagation

- gradient might vanish/explode through many non-linear transformations
- difficult to train on long-term dependencies

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

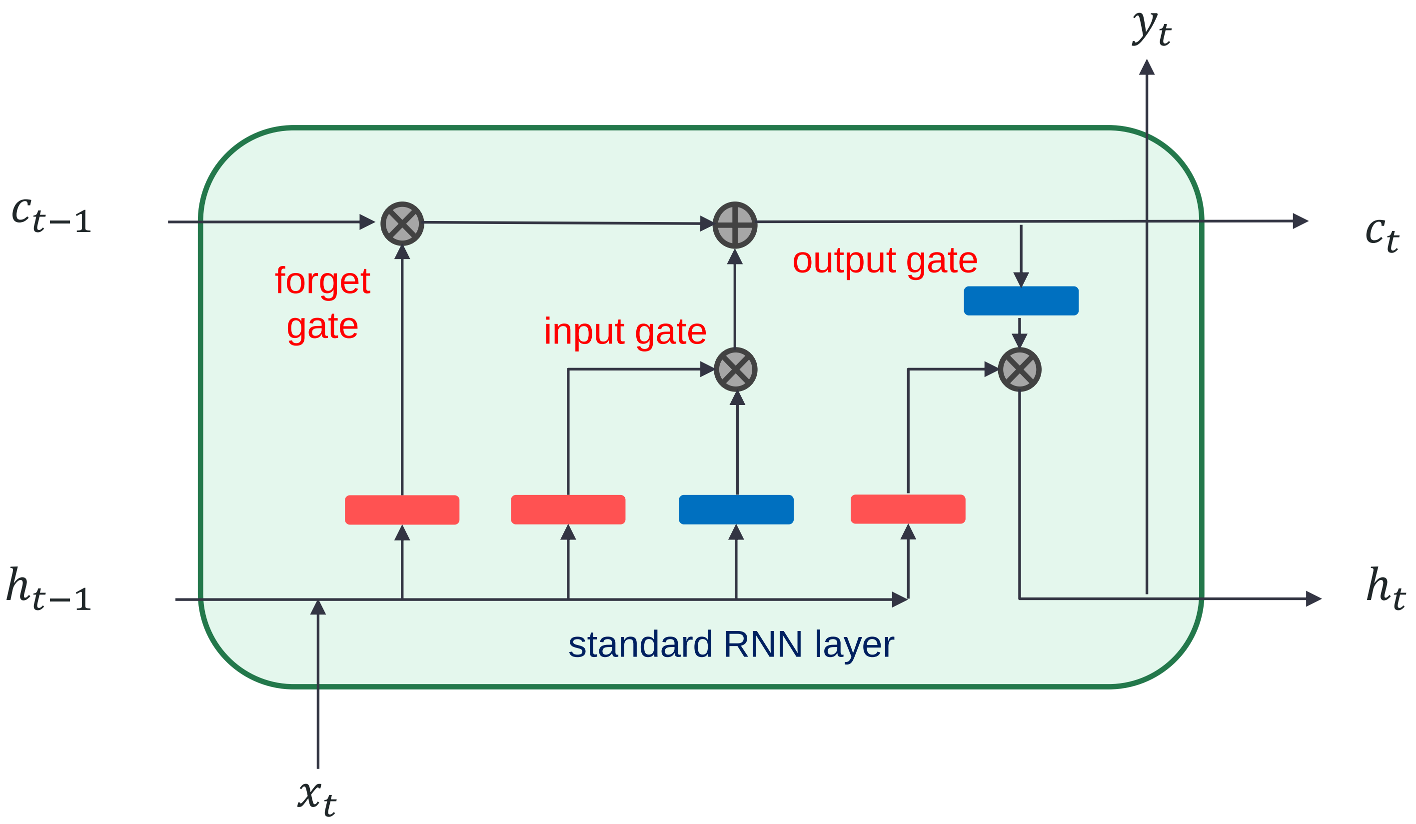
Ceni, A. (2022). Random orthogonal additive filters: a solution to the vanishing/exploding gradient of deep neural networks. *arXiv preprint arXiv:2210.01245*.

Approaches

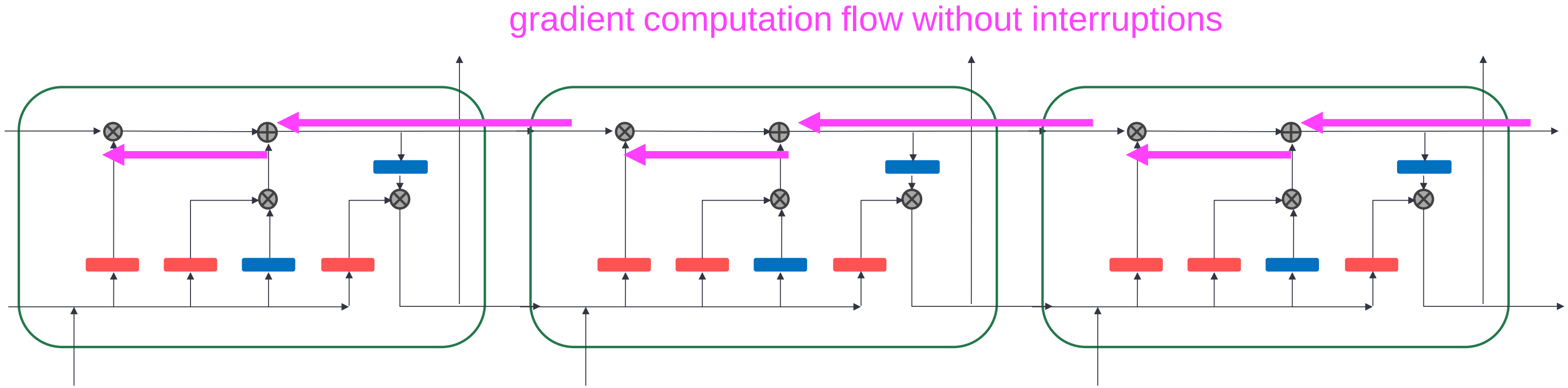
- **Gated architectures**
  - create a pathway for uninterrupted gradient propagation
  - LSTM, GRU
  - training is slow

- **Smart initialization**
  - Reservoir Computing
  - training is limited

LSTM cell


y_t
c_{t-1}
c_t
forget gate
input gate
output gate
h_{t-1}
h_t
standard RNN layer
x_t

Long-term dependencies



LSTM equations

vanilla RNN

- $g_t = tanh(h_{t-1}W_{hg} + x_t\, W_{xg} + b_g)$
- $f_t = \sigma\left(h_{t-1}\underline{W_{hf}} + x_t\, \underline{W_{xf}} + \underline{b_f}\right)$
- $i_t = \sigma(h_{t-1}\underline{W_{hi}} + x_t\, \underline{W_{xi}} + \underline{b_i})$
- $c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$
- $o_t = \sigma(h_{t-1}\underline{W_{ho}} + x_t\, \underline{W_{xo}} + \underline{b_o})$
- $h_t = o_t \otimes \tanh(c_t)$

extra computation

extra parameters

training is slow

(computationally intensive)

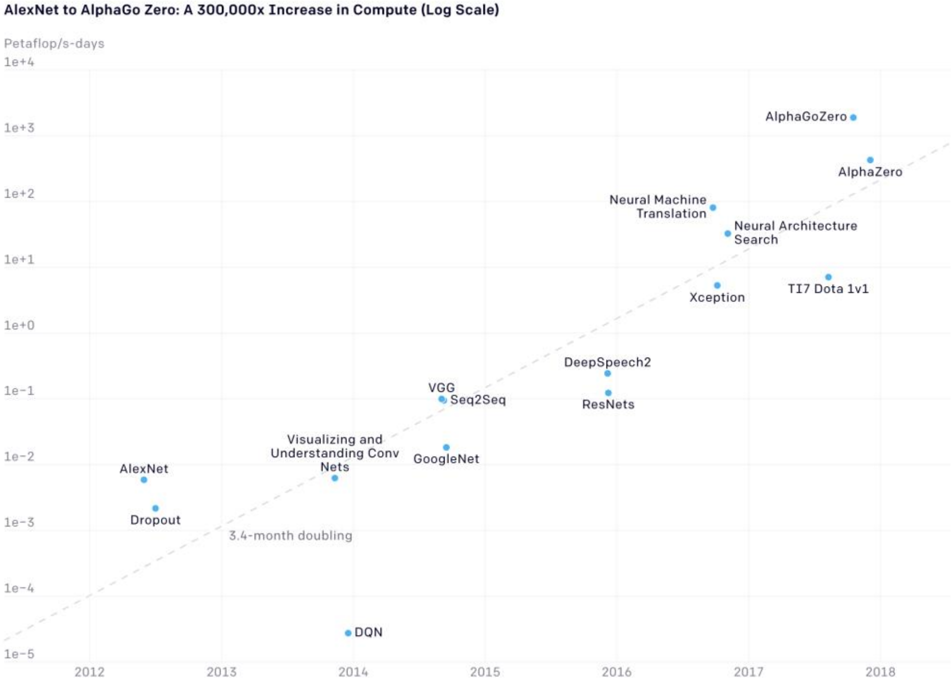
The Philosophy

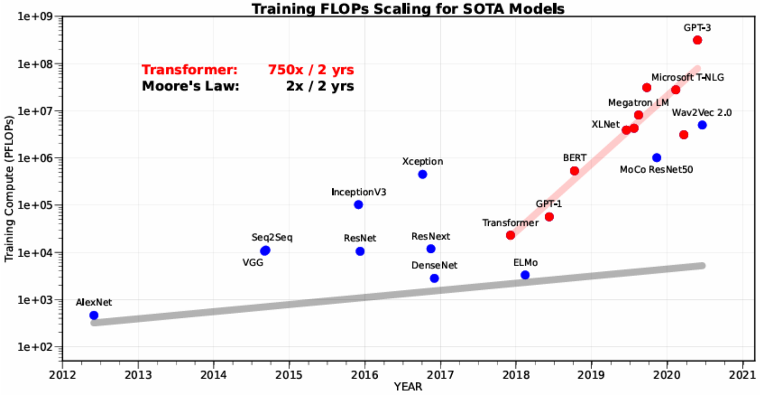
“Randomization is computationally cheaper than optimization”

Rahimi, A. and Recht, B., 2008. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. Advances in neural information processing systems, 21, pp.1313-1320.

Rahimi, A. and Recht, B., 2007. Random features for large-scale kernel machines. Advances in neural information processing systems, 20, pp. 1177-1184.

Energy consumption matters!





Gholami, Amir, et al. "Ai and memory wall." *IEEE Micro* (2024).

- **2012-2017: 300000x**
- **3.4-month doubling time**

Dario Amodei and Danny Hernandez. AI and compute, 2018. Blog post.

https://openai.com/blog/ai-and-compute/

Green AI

Roy Schwartz* ♢ Jesse Dodge*♢♣ Noah A. Smith♢♡ Oren Etzioni♢

♢Allen Institute for AI, Seattle, Washington, USA
♣ Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
♡ University of Washington, Seattle, Washington, USA

July 2019

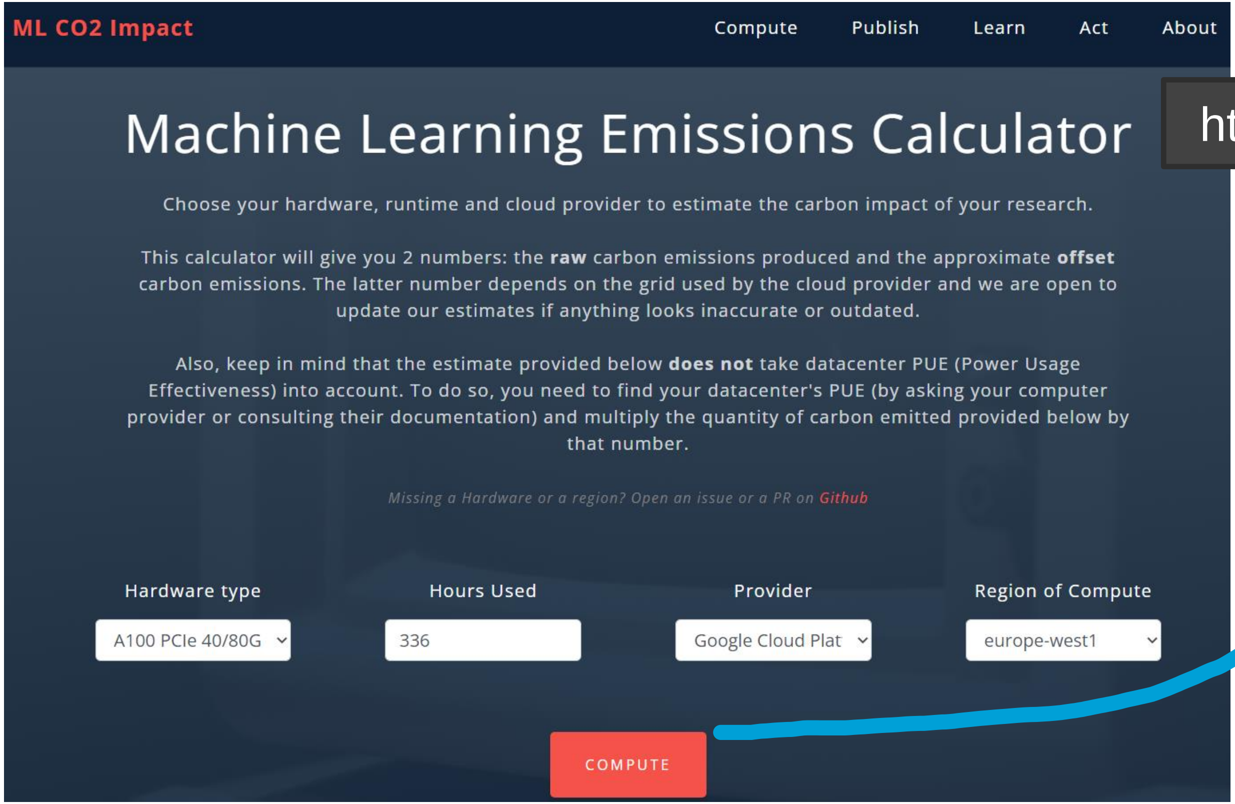
Abstract

The computations required for deep learning research have been doubling every few months, resulting in an estimated 300,000x increase from 2012 to 2018 [2]. These computations have a surprisingly large carbon footprint [40]. Ironically, deep learning was inspired by the human brain, which is remarkably energy efficient. Moreover, the financial cost of the computations can make it difficult for academics, students, and researchers, in particular those from emerging economies, to engage in deep learning research.

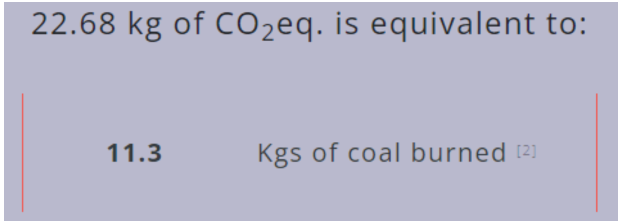
This position paper advocates a practical solution by making **efficiency** an evaluation criterion for research alongside accuracy and related measures. In addition, we propose reporting the financial cost or "price tag" of developing, training, and running models to provide baselines for the investigation of increasingly efficient methods. Our goal is to make AI both greener and more inclusive—enabling any inspired undergraduate with a laptop to write high-quality research papers. Green AI is an emerging focus at the Allen Institute for AI.

Schwartz, Roy, et al. "Green ai." *arXiv preprint arXiv:1907.10597* (2019).

Quantifying the carbon emissions of ML



https://mlco2.github.io/impact/



Lacoste, Alexandre, et al. "Quantifying the carbon emissions of machine learning." *arXiv preprint arXiv:1910.09700* (2019).

Energy consumption matters!

Artificial intelligence / Machine learning

Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao**

June 6, 2019

The artificial-intelligence **industry is often compared to the oil industry: once** mined and refined, data, like oil, can be a highly lucrative commodity. Now it seems the metaphor may extend even further. Like its fossil-fuel counterpart, the process of deep learning has an outsize environmental impact.

ImageNet Training in 24 Minutes

Yang You, Zhao Zhang, James Demmel, Kurt Keutzer, Cho-Jui Hsieh

(Submitted on 14 Sep 2017)

Finishing 90-epoch ImageNet-1k training with ResNet-50 on a NVIDIA M40 GPU takes 14 days. This training requires 10^18 single precision operations in total. On the other hand, the world's current fastest supercomputer can finish 2 * 10^17 single precision operations per second (Dongarra et al 2017). If we can make full use of the supercomputer for DNN training, we should be able to finish the 90-epoch ResNet-50 training in five seconds. However, the current bottleneck for fast DNN training is in the algorithm level. Specifically, the current batch size (e.g. 512) is too small to make efficient use of many processors
For large-scale DNN training, we focus on using large-batch data-parallelism synchronous SGD without losing accuracy in the fixed epochs. The LARS algorithm (You, Gitman, Ginsburg, 2017) enables us to scale the batch size to extremely large case (e.g. 32K). We finish the 100-epoch ImageNet training with AlexNet in 24 minutes, which is the world record. Same as Facebook's result (Goyal et al 2017), we finish the 90-epoch ImageNet training with ResNet-50 in one hour. However, our hardware budget is only 1.2 million USD, which is 3.4 times lower than Facebook's 4.1 million USD.

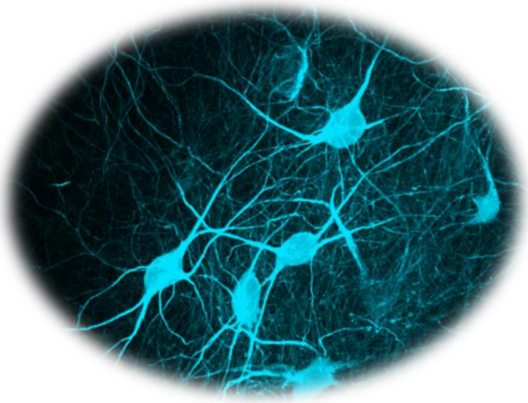
Yet another accelerated sgd: Resnet-50 training on **imagenet** in 74.7 **seconds**

M Yamazaki, A Kasagi, A Tabuchi, T Honda... - arXiv preprint arXiv ..., 2019 - arxiv.org

... on **ImageNet** using 81,920 mini-batch size in 74.7 **seconds**. ... -50 training on **ImageNet** in 74.7 **seconds** with 75.08% ... 2, the dotted line denotes the ideal throughput of images-per-**second**, ...

vs the Brain…

…Neuromorphic Computing



≈30 PFlops

10 MW vs 20 W

memory and computing are co-located

10^{11} neurons, 10^{15} synapses

10000 synapses/neuron

Deep Learning

Deep Learning models achieved tremendous success over the years. This comes at very high cost in terms of

- Time
- Parameters

Do we really need this all the time?

Example: embedded applications

Source: https://www.eenewsembedded.com/news/raspberry-pi-3-now-compute-module-format

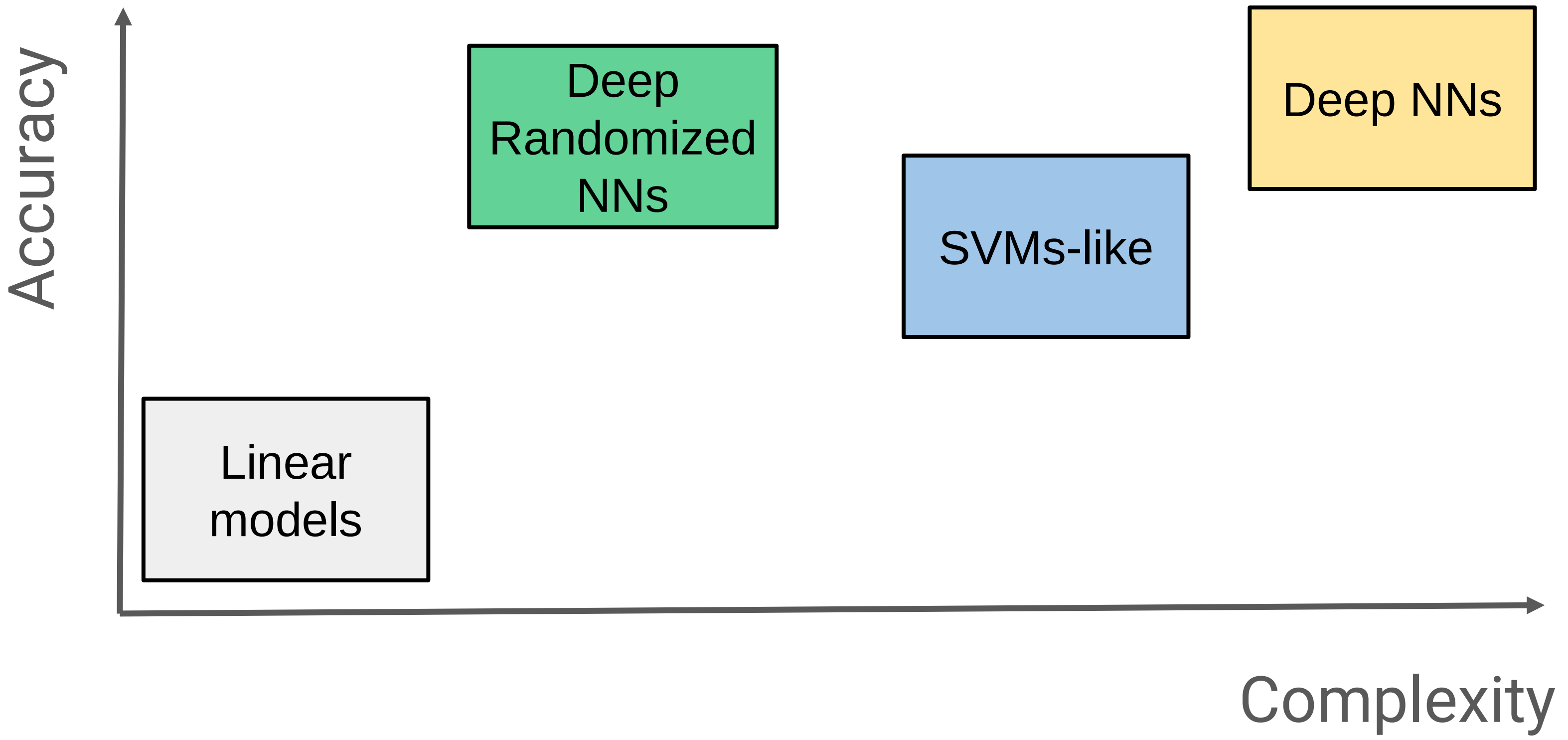
Source: https://bitalino.com/en/freestyle-kit-bt

Deep Neural Networks

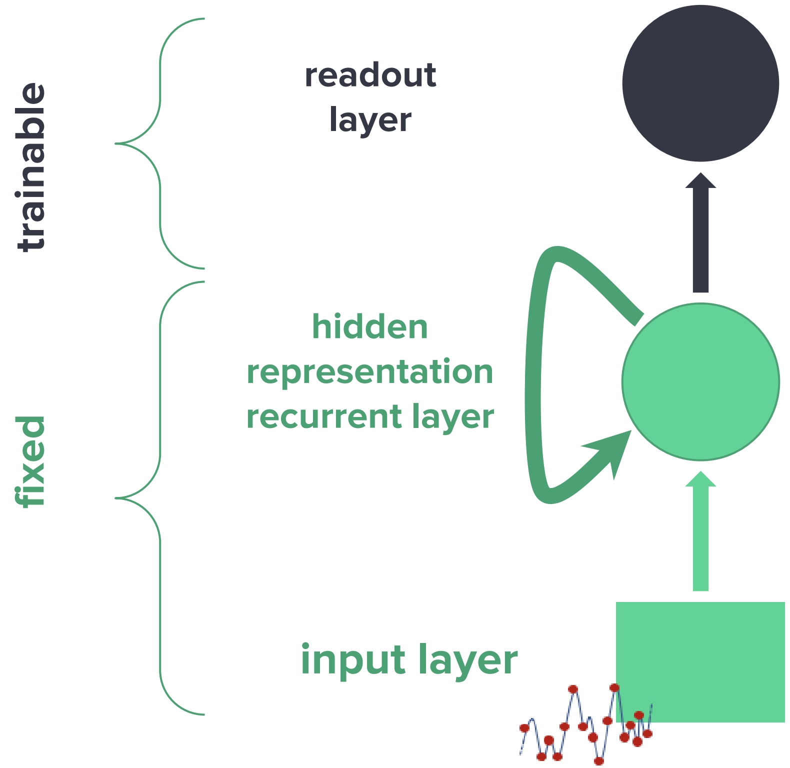
Powerful representations by applying multiple non-linear levels of transformation

Deep Learning = Architectural Biases + Learning Algorithms

Complexity / Accuracy Tradeoff



Randomized Recurrent Neural Networks



$$\boldsymbol{y}_t = \mathrm{g} \circ \mathrm{f}_{\mathrm{R}}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

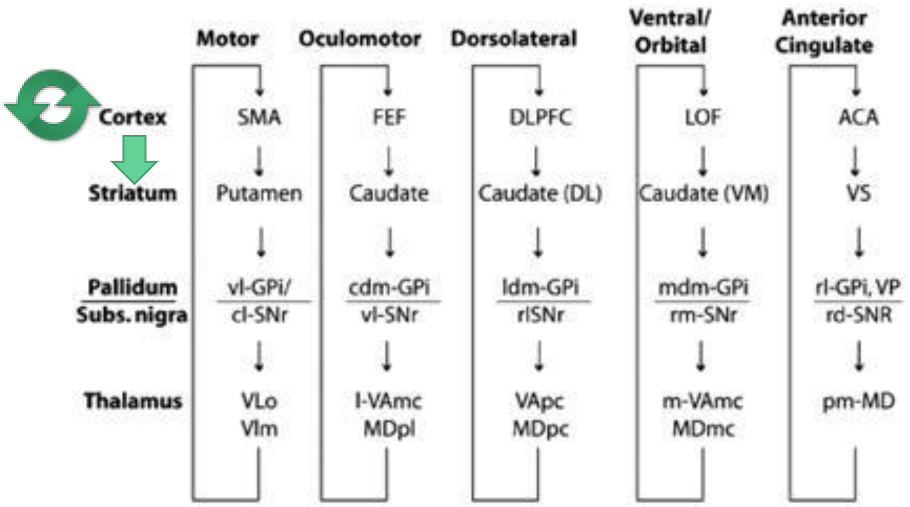
$$\boldsymbol{h}_t = \mathrm{f}_{\mathrm{R}}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

$$\boldsymbol{x}_t$$

Randomization = Efficiency

- Training algorithms are cheaper and simpler
- Model transfer: don’t need to transmit all the weights
- Amenable to neuromorphic implementations

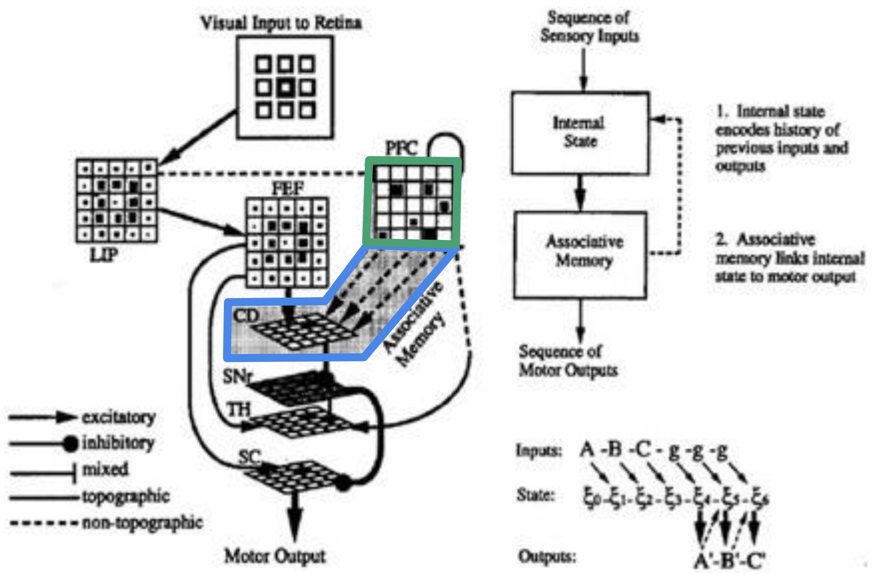
Historical note: the cortico-striatal model



Dominey, P.F., 2013. Recurrent temporal networks and language acquisition—from corticostriatal neurophysiology to reservoir computing. Frontiers in psychology, 4, p.500.

- Structured projections from cortex to striatum is a major architectural property of primate brains
- Recurrent cortico-cortical connections
- Dopamine-regulated plasticity in cortico-striatal connections

Historical note: the cortico-striatal model

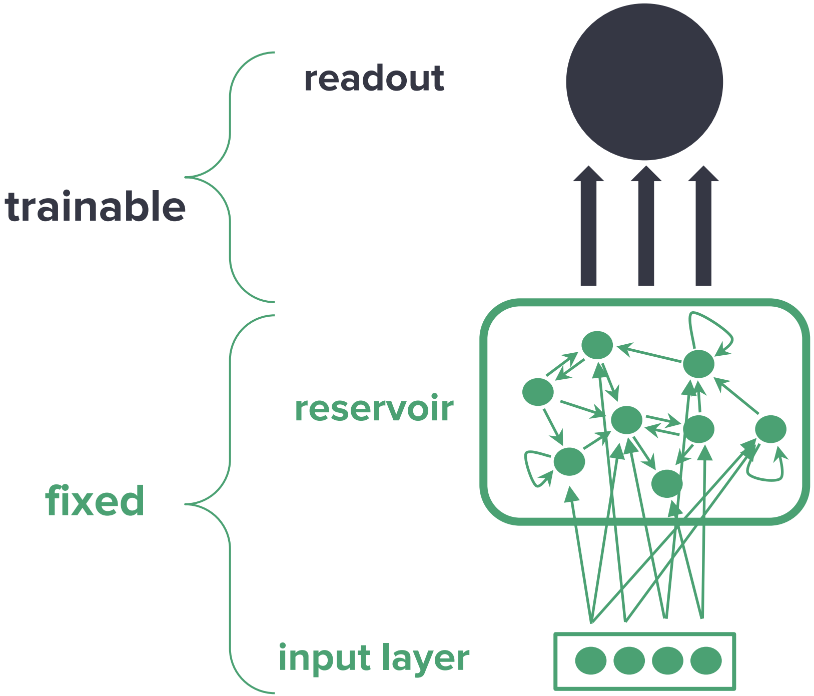


- Fixed recurrent connections in the PFC
- Modifiable connections between PFC and neurons in the striatum (CD)

Dominey, P.F., 2013. Recurrent temporal networks and language acquisition—from corticostriatal neurophysiology to reservoir computing. Frontiers in psychology, 4, p.500.

Reservoir Computing

Reservoir Computing: focus on the dynamical system



$$\boldsymbol{h}_t = \tanh(\boldsymbol{x}_t \underline{\boldsymbol{W}_{xh}} + \boldsymbol{h}_{t-1} \underline{\boldsymbol{W}_{hh}})$$

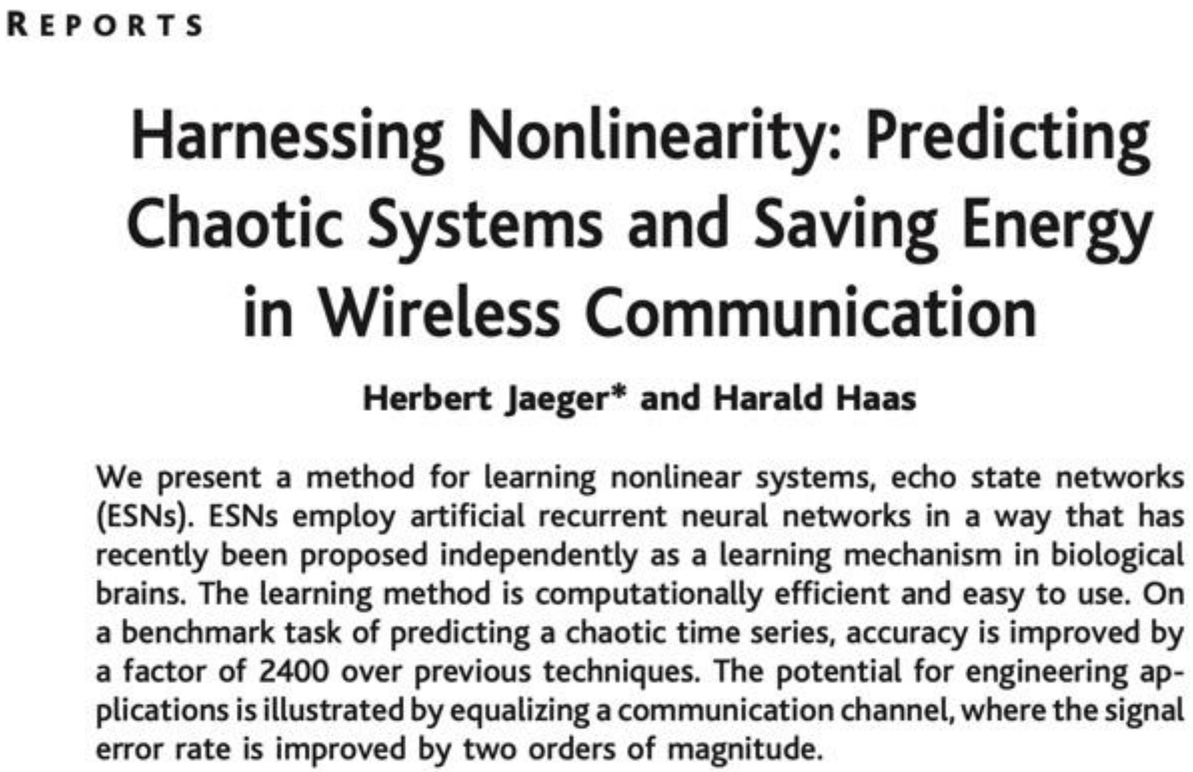
Randomly initialized under stability conditions on the dynamical system

Stable dynamics - Echo State Property

Verstraeten, David, et al. *Neural networks* 20.3 (2007).
Lukoševičius, Mantas, and Herbert Jaeger. *Computer Science Review* 3.3 (2009).

Jaeger, Herbert, and Harald Haas. Science 304.5667 (2004): 78-80.

Echo State Network


REPORTS

Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication

Herbert Jaeger* and Harald Haas

We present a method for learning nonlinear systems, echo state networks (ESNs). ESNs employ artificial recurrent neural networks in a way that has recently been proposed independently as a learning mechanism in biological brains. The learning method is computationally efficient and easy to use. On a benchmark task of predicting a chaotic time series, accuracy is improved by a factor of 2400 over previous techniques. The potential for engineering applications is illustrated by equalizing a communication channel, where the signal error rate is improved by two orders of magnitude.

Maass, Wolfgang, Thomas Natschläger, and Henry Markram. Neural computation 14.11 (2002): 2531-2560.

Liquid State Machine

ARTICLE — Communicated by Rodney Douglas

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

Tino, Peter, and Georg Dorffner. Machine Learning 45.2 (2001): 187-217.

Fractal Prediction Machine

Predicting the Future of Discrete Sequences from Fractal Representations of the Past

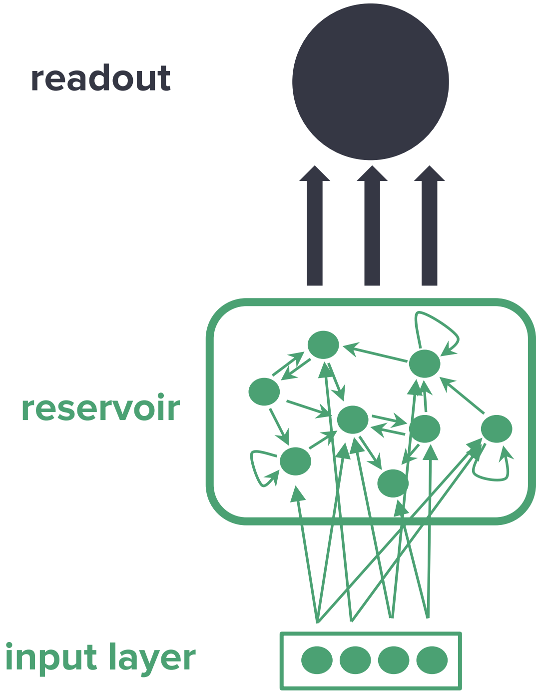
PETER TIÑO petert@ai.univie.ac.at
Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Vienna, Austria; Department of Computer Science and Engineering, Slovak University of Technology, Ilkovicova 3, 812 19 Bratislava, Slovakia

GEORG DORFFNER georg@ai.univie.ac.at
Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Vienna, Austria; Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, Freyung 6/2, A-1010 Vienna, Austria

Editor: Michael Jordan

Abstract. We propose a novel approach for building finite memory predictive models similar in spirit to variable memory length Markov models (VLMMs). The models are constructed by first transforming the n-block structure of the training sequence into a geometric structure of points in a unit hypercube, such that the longer is the common suffix shared by any two n-blocks, the closer lie their point representations. Such a transformation embodies a Markov assumption—n-blocks with long common suffixes are likely to produce similar continuations. Prediction contexts are found by detecting clusters in the geometric n-block representation of the training sequence via vector quantization. We compare our model with both the classical (fixed order) and variable memory length Markov models on five data sets with different memory and stochastic components. Fixed order Markov models (MMs) fail on three large data sets on which the advantage of allowing variable memory length can be exploited. On these data sets, our predictive models have a superior, or comparable performance to that of VLMMs, yet, their construction is fully automatic, which, is shown to be problematic in the case of VLMMs. On one data set, VLMMs are outperformed by the classical MMs. On this set, our models perform significantly better than MMs. On the remaining data set, classical MMs outperform the variable context length strategies.

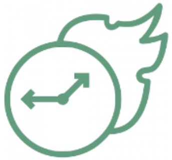
Echo State Networks (ESNs)



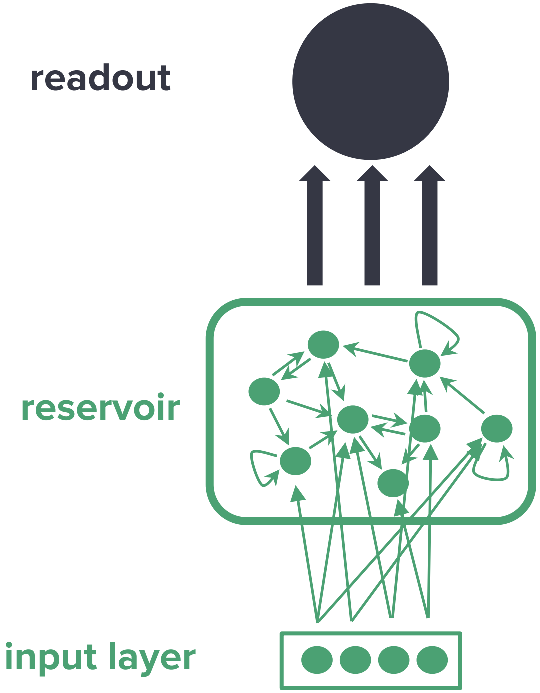
Reservoir

$$\boldsymbol{h}_t = \tanh(\boldsymbol{x}_t \boldsymbol{W}_{xh} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hh})$$

- large layer of recurrent units
- sparsely connected
- randomly initialized under the ESP
- untrained



Echo State Networks (ESNs)



Readout

$$y_t = h_t W_{hy}$$

- linear combination of the reservoir state variables
- can be trained in closed form

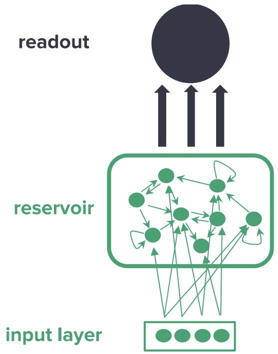
$$W_{hy} = (H^T H)^{-1} H^T D$$

ESNs in a nutshell

- Architecture of the Echo State Network:
  - Reservoir: untrained non-linear recurrent hidden layer
  - Readout: (linear) output layer
- Setup of the Neural Network:
  - Initialize W_{xh} and W_{hh} randomly
  - Scale W_{hh} to meet the contractive/stability property
- Training of the Neural Network
  - Drive the network with the input signal
  - Discard an initial transient
  - Train the readout

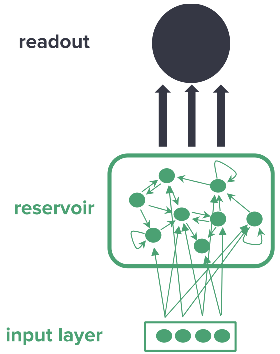
Reservoir

- Non-linearly embed the input into a higher dimensional feature space where the original problem is more likely to be solved linearly (Cover's Th.)
- Randomized basis expansion computed by a pool of randomized filters
- Provides a "rich" set of input-driven dynamics



Readout

- Use the features in the reservoir state space for the output computation
- Typically implemented by using linear models
- Learning involves convex optimization



Reservoir: State Transition Function

Reservoir = discrete-time input-driven dyn. system

- Dynamics are driven by the state transition function

$$F: \mathbb{R}^{N_X} \times \mathbb{R}^{N_H} \rightarrow \mathbb{R}^{N_H}$$

$$\begin{aligned}\mathbf{h}_t &= \mathrm{F}(\mathbf{x}_t, \mathbf{h}_{t-1}) \\ &= \tanh(\mathbf{x}_t \mathbf{W}_{\mathrm{xh}} + \mathbf{h}_{t-1} \mathbf{W}_{\mathrm{hh}})\end{aligned}$$

Iterated version of state transition function

- given an input sequence $\boldsymbol{s} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_t]$
- return the final state $\boldsymbol{h}_t$

$$\hat{F}: (\mathbb{R}^{N_X})^* \times \mathbb{R}^{N_H} \to \mathbb{R}^{N_H}$$

input sequence $\boldsymbol{s}$

initial state $\boldsymbol{h}_0$

final state $\boldsymbol{h}_t$

Reservoir: State Transition Function

Iterated version of state transition function

$$\hat{F}(\boldsymbol{s}, \mathbf{h}_0) = \begin{cases} \boldsymbol{h}_0 & if\ \boldsymbol{s} = [\,] \\ F(\boldsymbol{x}_t, \underline{\hat{F}([\boldsymbol{x}_1, \dots, \boldsymbol{x}_{t-1}], \boldsymbol{h}_0)}) & if\ \boldsymbol{s} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_t] \end{cases}$$

iterated application:
the state after seeing all but 1 input

Echo State Property (ESP)

Yildiz, Izzet B., Herbert Jaeger, and Stefan J. Kiebel. "Re-visiting the echo state property." *Neural networks* 35 (2012): 1-9.

A valid ESN should satisfy the "Echo State Property"

- (DEF) An ESN satisfies the ESP whenever:

$$\forall \boldsymbol{s} \in (\mathbb{R}^{N_X})^N, \forall \boldsymbol{h_0}, \boldsymbol{z}_0 \in \mathbb{R}^{N_H}:$$

input sequence of length N — couple of initial states

$$\left\| \hat{F}(\mathbf{s}, \mathbf{h}_0) - \hat{F}(\mathbf{s}, \mathbf{z}_0) \right\| \to 0, \qquad as\ N \to \infty$$

the distance between the final states goes to 0

Echo State Property

- The state of the network asymptotically depends only on the driving input signal
- Dependencies on the initial conditions are progressively lost
- Fading memory

Conditions for the ESP

- **Sufficient Condition**, involving the control of the maximum singular value of $\mathbf{W}_{\mathrm{hh}}$
- **Necessary Condition**, involving the control of the maximum eigenvalue in modulus of $\mathbf{W}_{\mathrm{hh}}$
- Active area of intense mathematical research: link it to the input

Sufficient Condition for the ESP

- **Theorem.** If the maximum singular value of $\mathbf{W}_{\mathrm{hh}}$ is < 1 then (under mild assumptions) the ESN satisfies the ESP for any possible input.

Sufficient condition for the ESP

- contractive dynamics for every input

  $\sigma_{max}(\mathbf{W}_{\mathrm{hh}}) = \|\mathbf{W}_{\mathrm{hh}}\|_2 < 1$

Necessary Condition for the ESP

- **Theorem.** If the spectral radius of $\mathbf{W}_{\mathrm{hh}}$ is not smaller than 1 then (under mild assumptions) the ESN does not satisfy the ESP.

Necessary condition for the ESP

- globally asymptotically stable dynamics around the 0 state
$\rho(\mathbf{W}_{\mathrm{hh}}) = \max(abs(eig(\mathbf{W}_{\mathrm{hh}}))) < 1$

Relation between the ESP conditions

- Known linear algebra fact: $\rho(\mathbf{W}_{\mathrm{hh}}) \leq \|\mathbf{W}_{\mathrm{hh}}\|_n$
- Applying the sufficient condition is OK in theory, but often impractical: it is too strong!
- Often, the necessary condition is used as an easy way for initialization of the reservoir

Reservoir Initialization

Initialization of $\mathbf{W}_{\mathrm{hh}}$:

1) Generate a random matrix $\boldsymbol{W}$

   e.g. from a uniform distribution on [-1,1]

2) Scale by the desired spectral radius

$$\mathbf{W}_{\mathrm{hh}} \leftarrow \rho_{desired} \frac{\mathbf{W}}{\rho(\mathbf{W})}$$

- Note that now $\rho(\mathbf{W}_{\mathrm{hh}}) = \rho_{desired}$ (choose a value < 1)
- The spectral radius is a key hyper-parameter of the reservoir

Reservoir Initialization

Initialization of $\boldsymbol{W}_{xh}$:

1) Generate a random matrix $\boldsymbol{W}_{xh}$, whose elements are drawn e.g. from a uniform distribution on [-1,1]
2) Scale by an input scaling parameter ω_{in}
   - by range: $\boldsymbol{W}_{xh} \leftarrow \omega_{in} \boldsymbol{W}_{xh}$ (now weights are in $[-\omega_{in}, \omega_{in}]$
   - by norm: $\boldsymbol{W}_{xh} \leftarrow \omega_{in} \frac{\boldsymbol{W}_{xh}}{\|\mathbf{W}_{xh}\|_2}$ (now the 2-norm of $\boldsymbol{W}_{xh}$ equals ω_{in})

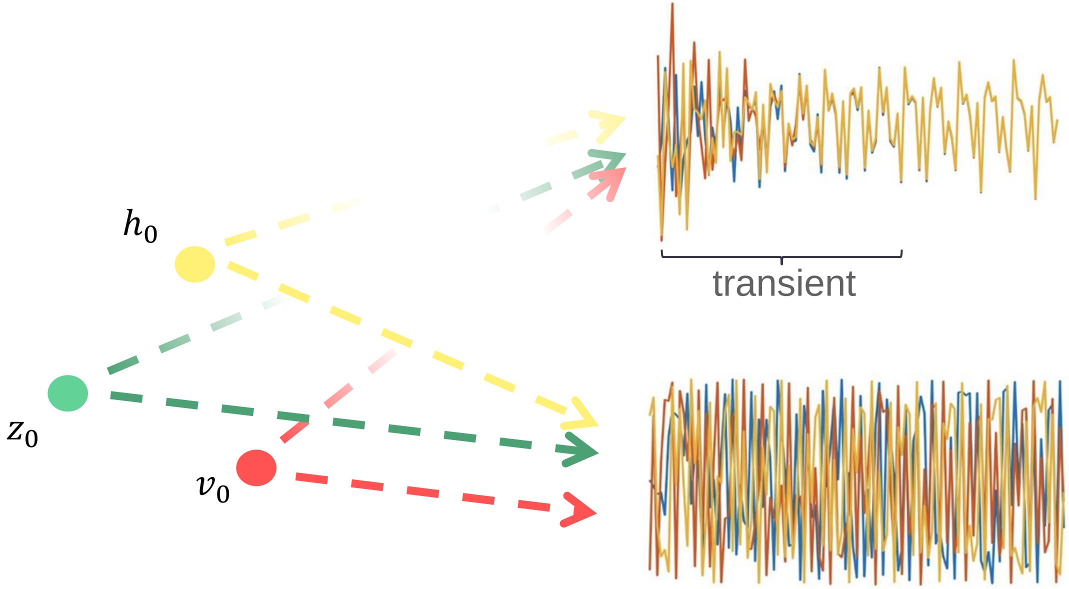
- The input scaling is a key hyper-parameter of the reservoir

Dynamical Transient

- If the system is globally asymptotically stable, then all possible (input-driven) trajectories will synchronize *after a transient*
- **Washout**: initial part of the time-series in which the state could be still affected by initialization condition (i.e. here the ESP could still not hold)
    - the washout states of the reservoir should be discarded

Gallicchio, Claudio. "Chasing the echo state property." ESANN 2019

Stability in Practice



Stable dynamics

- orbits synchronize after a transient

Unstable dynamics

- orbits are sensitive to initial conditions

ESN Training

Given a training set $\{(\boldsymbol{x}_t, \boldsymbol{d}_t)\}_{t=1}^{N}$

1. Run the reservoir on the input sequence & collect the states

$$H = [\mathbf{h}_1, \dots, \mathbf{h}_N]$$

2. Washout the initial transient. $\mathbf{H} \leftarrow \mathbf{H}(N_w : N, :)$
3. Collect the target data similarly into a matrix

$$\mathbf{D} = \left[\mathbf{d}_{N_w}, \dots, \mathbf{d}_N\right]$$

4. Solve the linear regression problem for the readout

$$\min_{\mathbf{W}_{\mathrm{hy}}} \left\| \mathbf{H}\mathbf{W}_{\mathrm{hy}} - \mathbf{D} \right\|_2^2$$

Readout Training

- Typically training performed off-line in closed-form
  - pseudo-inverse $W_{hy} = (H^T H)^{-1} H^T D$
  - ridge-regression $W_{hy} = (H^T H + \underbrace{\lambda}_{\text{Tikhonov regularizer}} I)^{-1} H^T D$
- Online learning by Least Mean Squares has problems
  - high eigenvalue spread of H
  - alternatives: Recursive Least Squares
  - … use any other modern optimizer for training the readout layer, use deep readouts

Hyper-parameters tuning by model selection

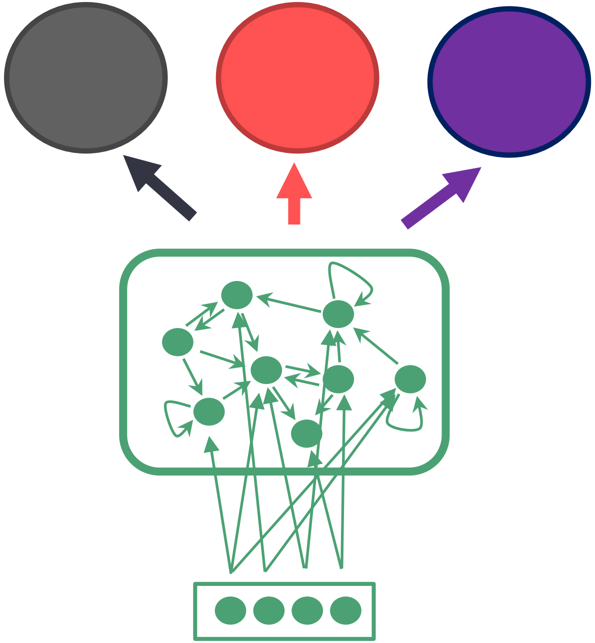
Like for any other **ML/NN** model, hyp-params tuning is important in applications

- reservoir dimension N_H
- spectral radius ρ
- input scaling ω_{in}
- readout regularization λ
- ...

Practical tips

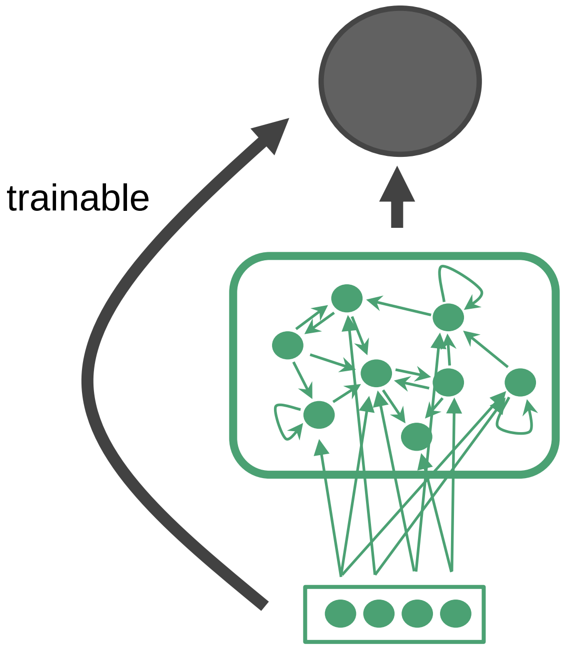
- Using sparse reservoir matrices to boost computing times
- Using $\rho < 1$ gives the ESP in practice in most situations
  - in fine-tuning explore also values >1 (e.g., $\rho \in (0.1, 1.5)$)
- Use larger values of ρ when more memory is needed
- In case of time-series regression:
  - For long sequences discard an initial transient
- In case of time-series classification:
  - Use the last state as representative for the whole input sequence

Architectural variants: Multiple readouts



- The reservoir is operating in unsupervised mode
- The same reservoir can serve to tackle multiple learning problems

Architectural variants: Input-Readout connections

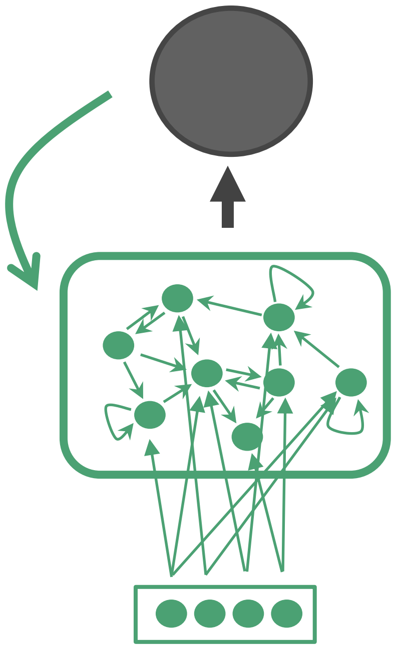


Direct input-readout connections

$$\mathbf{h}_t = \tanh(\boldsymbol{x}_t \mathbf{W}_{\mathrm{xh}} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hh})$$
$$\boldsymbol{y}_t = [\boldsymbol{h}_t; \boldsymbol{x}_t] \boldsymbol{W}_{hy}$$

useful, e.g., when instantaneous (i.e., non-temporal) I/O transformations can be useful

Architectural Variants: Output feedback



Feedback connections from the readout

$$y_t = h_t W_{hy}$$

$$\mathbf{h}_t = \tanh\left(x_t \mathbf{W}_{\mathrm{xh}} + h_{t-1} W_{hh} + y_{t-1} W_{yh}\right)$$

Might impact on the stability of the reservoir dynamics.

Note: in this case the reservoir dynamics are adapted...

$$\mathbf{h}_t = \tanh\left(x_t \mathbf{W}_{\mathrm{xh}} + h_{t-1}(W_{hh} + W_{hy} W_{yh})\right)$$

Jaeger, Herbert, et al. *Neural networks* 20.3 (2007): 335-352.

Leaky Integrator ESN (LI-ESN)

Use leaky integrators reservoir neurons:

$$\boldsymbol{h}_t = (1 - \underline{\alpha})\boldsymbol{h}_{t-1} + \underline{\alpha} \tanh(\boldsymbol{x}_t \boldsymbol{W}_{xh} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hh})$$

leaking rate hyper-parameter

$\alpha \in (0,1]$

A few examples of RC applications

Reservoir Computing in practice

Lukoševičius, M., 2012. A practical guide to applying echo state networks. In Neural networks: Tricks of the trade (pp. 659-686). Springer, Berlin, Heidelberg.

A Practical Guide to Applying Echo State Networks

Mantas Lukoševičius

Jacobs University Bremen, Campus Ring 1,
28759 Bremen, Germany
m.lukosevicius@jacobs-university.de

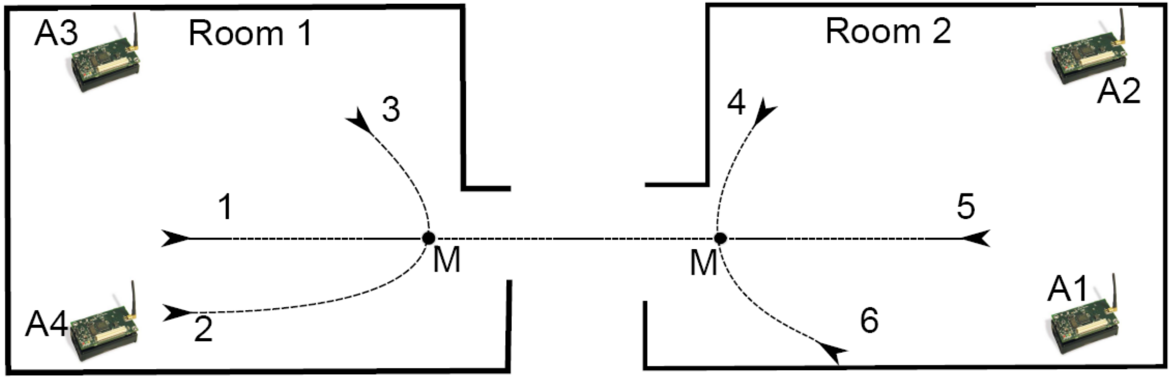
Abstract. Reservoir computing has emerged in the last decade as an alternative to gradient descent methods for training recurrent neural networks. Echo State Network (ESN) is one of the key reservoir computing "flavors". While being practical, conceptually simple, and easy to implement, ESNs require some experience and insight to achieve the hailed good performance in many tasks. Here we present practical techniques and recommendations for successfully applying ESNs, as well as some more advanced application-specific modifications.

Distributed Intelligence Applications

Dragone, Mauro, et al. "A cognitive robotic ecology approach to self-configuring and evolving AAL systems." Engineering Applications of Artificial Intelligence 45 (2015): 269-280.

Human Activity Monitoring

Bacciu, Davide, et al. *Neural Computing and Applications* 24.6 (2014): 1451-1464.



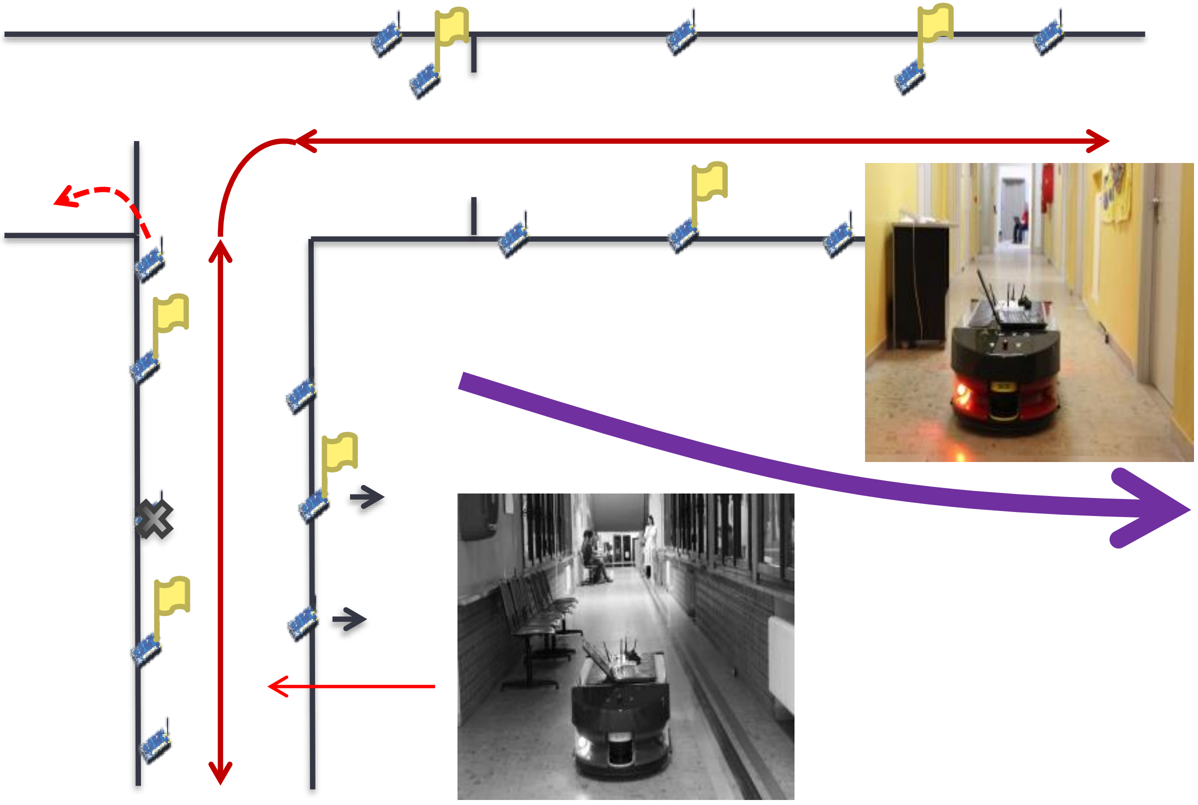
Forecasting human indoor mobility

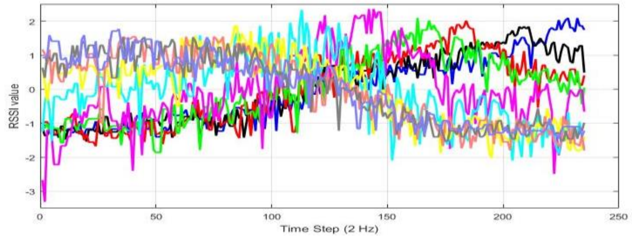
Dataset is available online on the UCI repository

https://archive.ics.uci.edu/ml/datasets/Indoor+User+Movement+Prediction+from+RSS+data

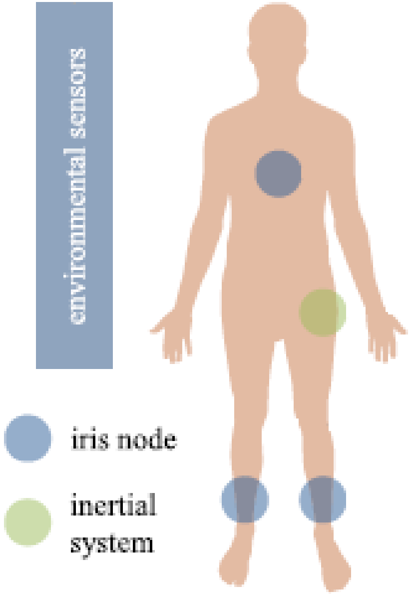
Dragone, Mauro, et al. *ESANN*. 2016.

Robot localization in critical environments





Human Activity Recognition



- Classification of human daily activities from RSS data generated by sensors worn by the user

EvAAL

UCI Machine Learning Repository Dataset is available online on the UCI repository

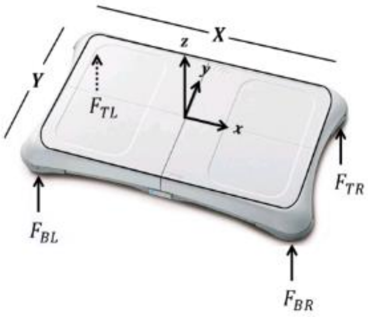
http://archive.ics.uci.edu/ml/datasets/Activity+Recognition+system+based+on+Multisensor+data+fusion+%28AReM%29

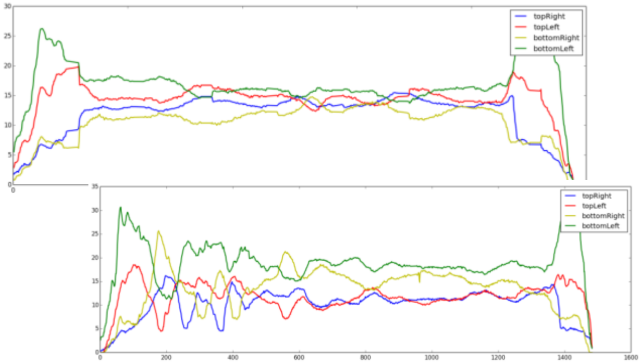
Clinical applications

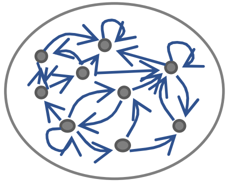
Bacciu, Davide, et al. *Engineering Applications of Artificial Intelligence* 66 (2017): 60-74.

- Automatic assessment of balance skills
- Predict the outcome of the Berg Balance Scale (BBS) clinical test from time-series of pressure sensors

Wii Balance Board



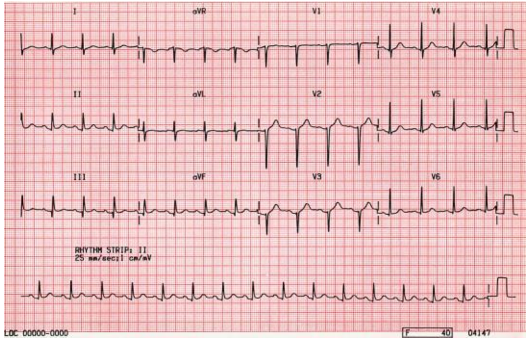




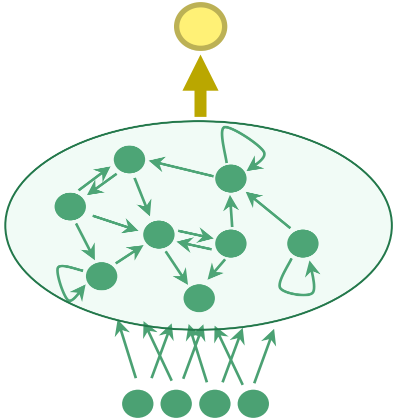
BBS

Brugada Syndrome

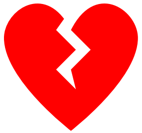
Dimitri, Giovanna Maria, et al. "A preliminary evaluation of Echo State Networks for Brugada syndrome classification." SSCI, IEEE, 2021.



ECG leads in input



processed by ESNs



Brugada Type 1 syndrome diagnosis
≈ 80% accurate

TEACHING

PROJECT CONSORTIUM RESULTS NEWS & EVENTS

TEACHING

A computing Toolkit for building Efficient Autonomous appliCations leveraging Humanistic INtelliGence is an EU-funded project that designs a computing platform and the associated software toolkit supporting the development and deployment of autonomous, adaptive and dependable CPSoS applications, allowing them to exploit a sustainable human feedback to drive, optimize and personalize the provisioning of their services.

https://www.teaching-h2020.eu

https://twitter.com/TEACHING_H2020

https://www.linkedin.com/company/teaching-horizon-2020/

TEACHING

RC in Autonomous Vehicles

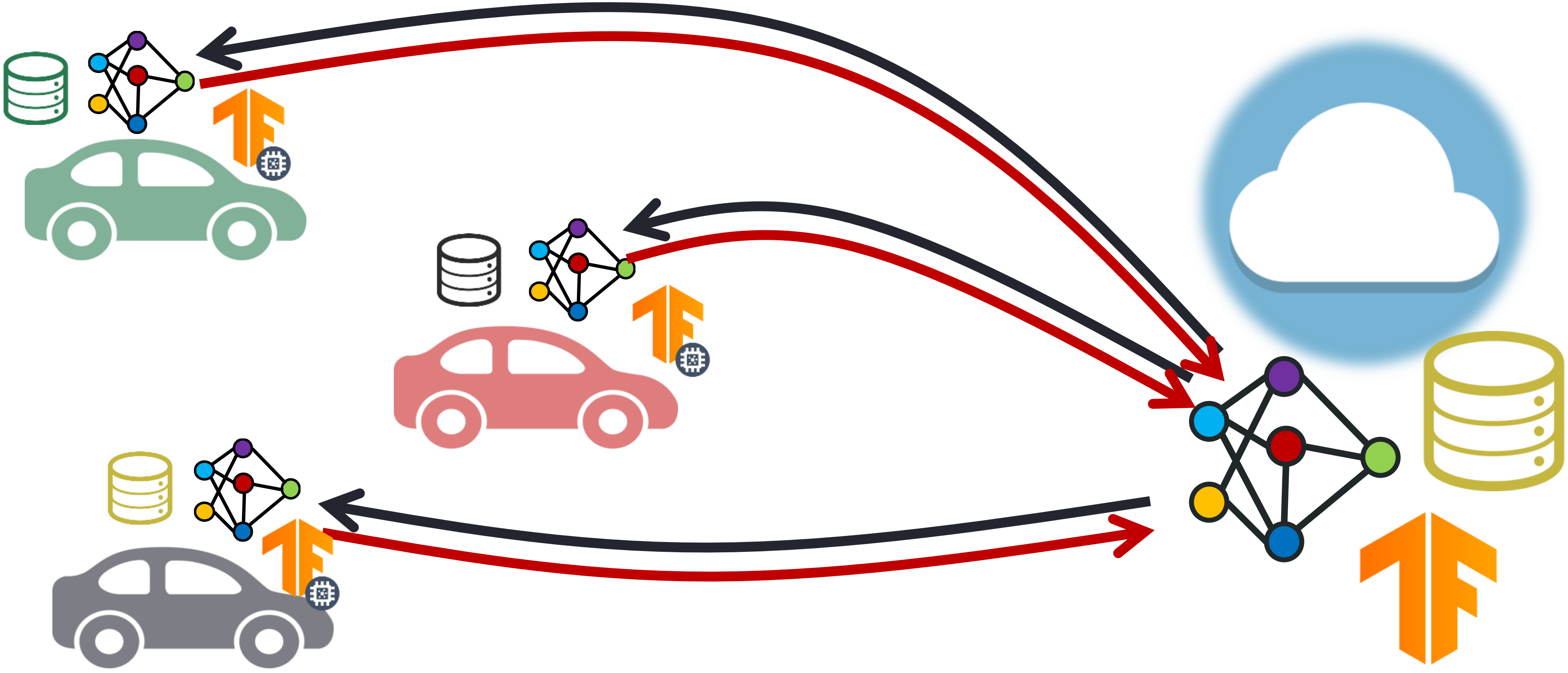
- Automatic detection of physiological, emotional, cognitive state of the human → Human-centric personalization
- Good performance in human state monitoring + efficiency

	WESAD		HHAR		PAMAP2		OPPORTUN.		ASCERTAIN	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
RNN	94.62	2.84	78.54	2.04	96.00	3.39	96.84	2.58	94.77	0.78
ESN	94.96	2.60	89.79	3.81	97.50	2.74	94.74	5.77	96.54	0.77
LSTM	95.48	1.17	92.71	2.72	96.50	1.22	93.08	2.88	94.63	0.00
GRU	98.13	1.16	98.54	0.83	98.50	2.00	96.84	2.58	94.63	0.00

D. Bacciu, D. Di Sarli, C. Gallicchio, A. Micheli, N. Puccinelli, "Benchmarking Reservoir and Recurrent Neural Networks for Human State and Activity Recognition", IWANN 2021

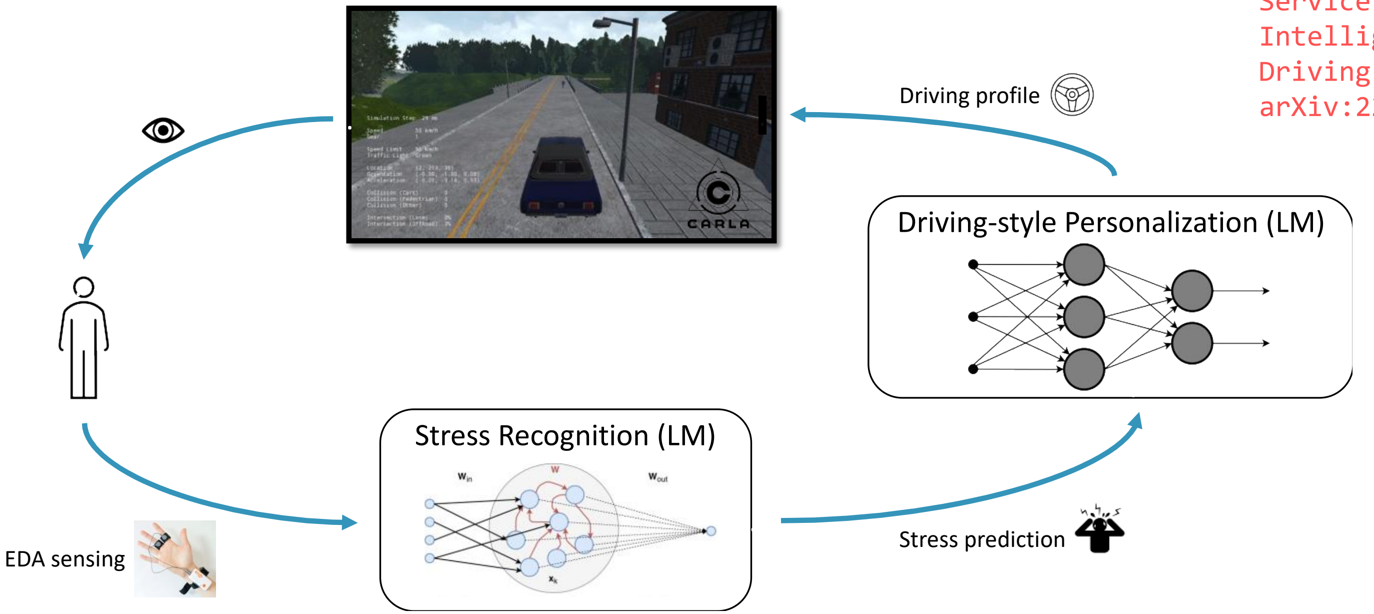
Bacciu, Davide, et al. COINS, IEEE, 2021.

Distributed, embeddable and federated learning



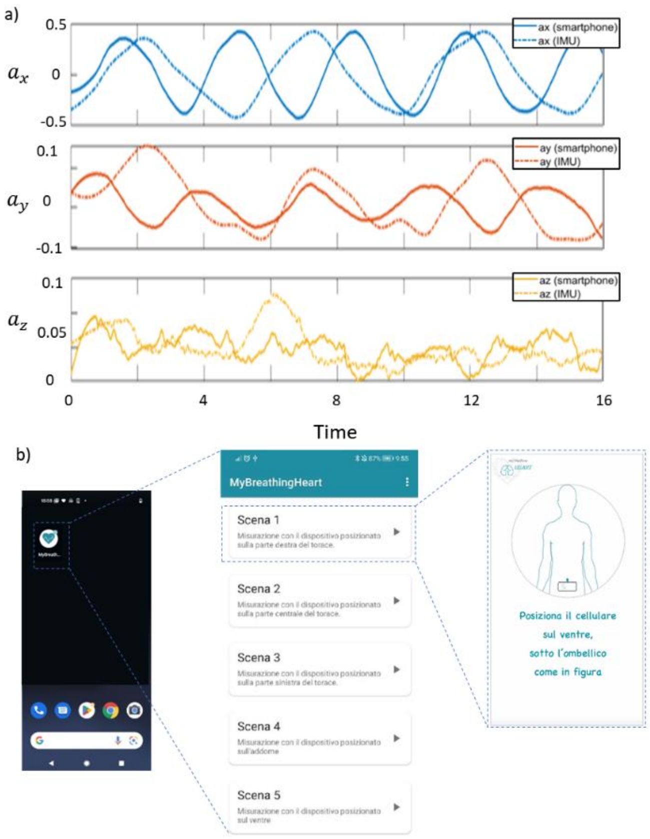
Driving-Style Personalization Based on Driver Stress

De Caro, Valerio, et al. "AI-as-a-Service Toolkit for Human-Centered Intelligence in Autonomous Driving." arXiv preprint arXiv:2202.01645, PERCOM 2022



YouTube https://www.youtube.com/watch?v=QrGsqlhjSRA

MyBreathingHeart



Sviluppo ed implementazione di un'applicazione per smartphone per il monitoraggio remoto di problemi cardio-respiratori durante crisi pandemica

EMERGE

https://eic-emerge.eu

Reservoir Computing: Research

Quality of Reservoir dynamics

- **Entropy** of recurrent units activations
  - Unsupervised adaptation of reservoirs using Intrinsic Plasticity
- Study the **short-term memory ability** of the system
  - Memory Capacity and relations to linearity
- **Edge of stability/chaos**: reservoir at the border of stability
  - Recurrent systems close to instability show optimal performances whenever the task at hand requires long short-term memory

Intrinsic Plasticity

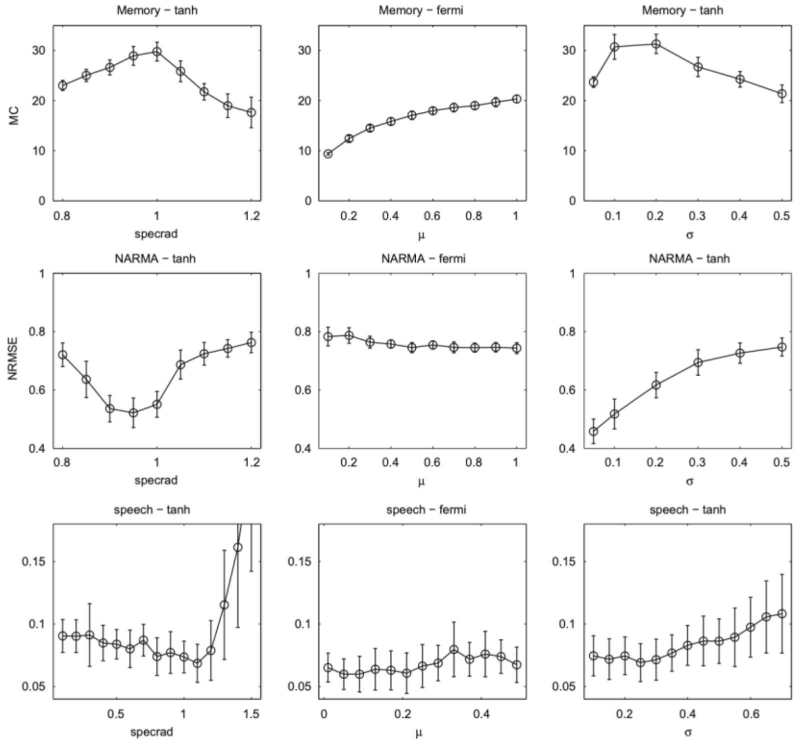


Fig. 4. Results for all three benchmarks for tanh with spectral radius ranging (left column), exponential IP for fermi nodes (middle column), and Gaussian IP for tanh nodes (right column).

- Adapt gain and bias of the act. function
- Tune the probability density of reservoir neurons to maximum entropy

$$f_{\text{gen}}(x) = f(ax + b)$$

$$\Delta b = -\eta\left(-\frac{\mu}{\sigma^2} + \frac{y}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y)\right),$$

$$\Delta a = \frac{\eta}{a} + \Delta b x.$$

Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J.J. and Stroobandt, D., 2008. Improving reservoirs using intrinsic plasticity. Neurocomputing, 71(7-9), pp.1159-1171.

Edge of chaos

Boedecker, J., Obst, O., Lizier, J.T., Mayer, N.M. and Asada, M., 2012. Information processing in echo state networks at the edge of chaos. Theory in Biosciences, 131(3), pp.205-213.

Improved dynamics near the transition between ordered and chaotic

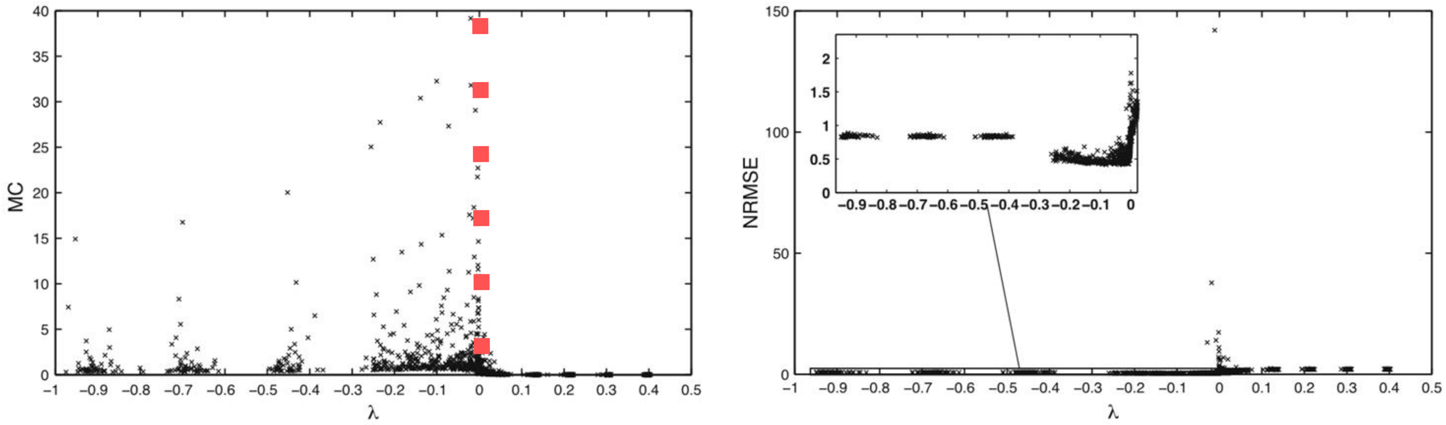
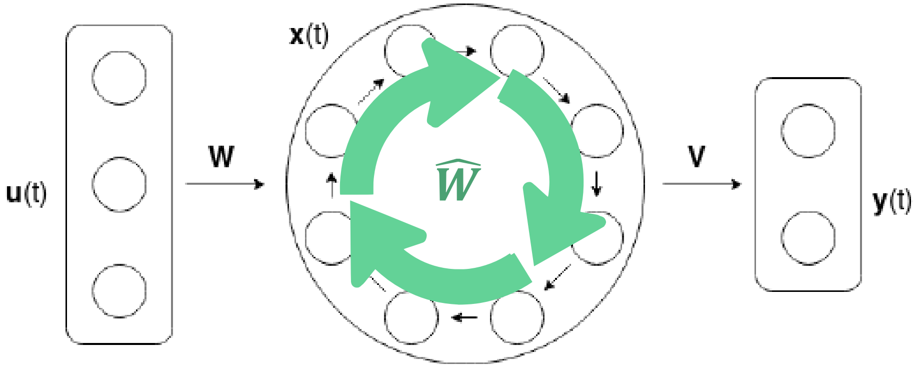


Fig. 3 *Left* Memory capacity versus estimated Lyapunov exponent. *Right* Normalized root mean squared error (NRMSE) versus estimated Lyapunov exponent

Simple Cycle Reservoir (SCR)

The reservoir layer has an easy-to-build orthogonal structure



$$\hat{\mathbf{W}} = \begin{pmatrix} 0 & 0 & \cdots & 0 & \hat{w} \\ \hat{w} & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & \hat{w} & 0 \end{pmatrix}$$

Rodan, A. and Tino, P., 2010. Minimum complexity echo state network. IEEE transactions on neural networks, 22(1), pp.131-144.

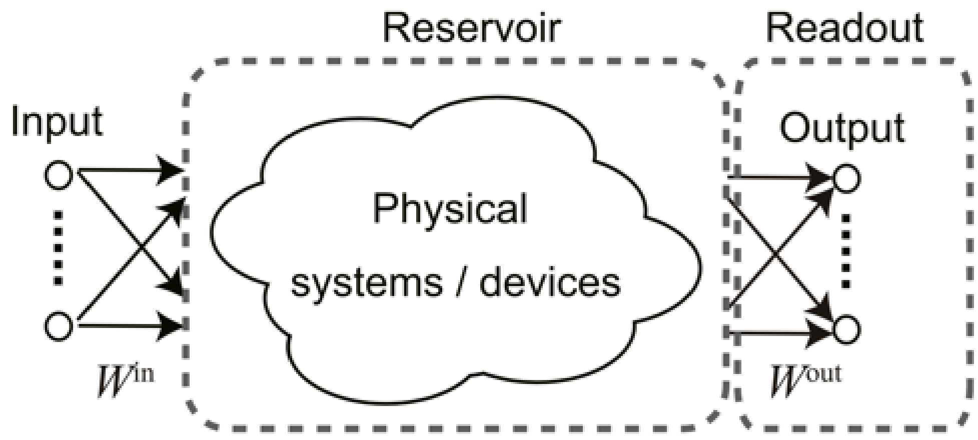
Approximation Capabilities

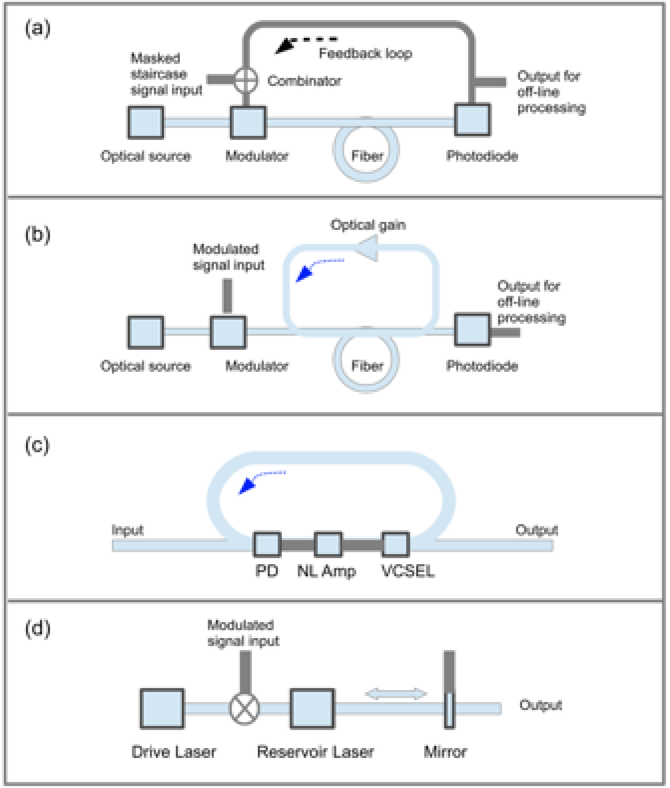
- Echo State Networks can approximate any fading memory filter
- non-linear reservoir + linear readout
- trigonometric state affine reservoir systems + linear readout
- linear reservoir systems + non-linear readout (e.g., MLP)

Lyudmila Grigoryeva and Juan-Pablo Ortega. Echo state networks are universal. Neural Networks, 108:495–508, 2018.

Lukas Gonon and Juan-Pablo Ortega. Reservoir computing universality with stochastic inputs. IEEE transactions on neural networks and learning systems, 2019

Physical Reservoir Computing

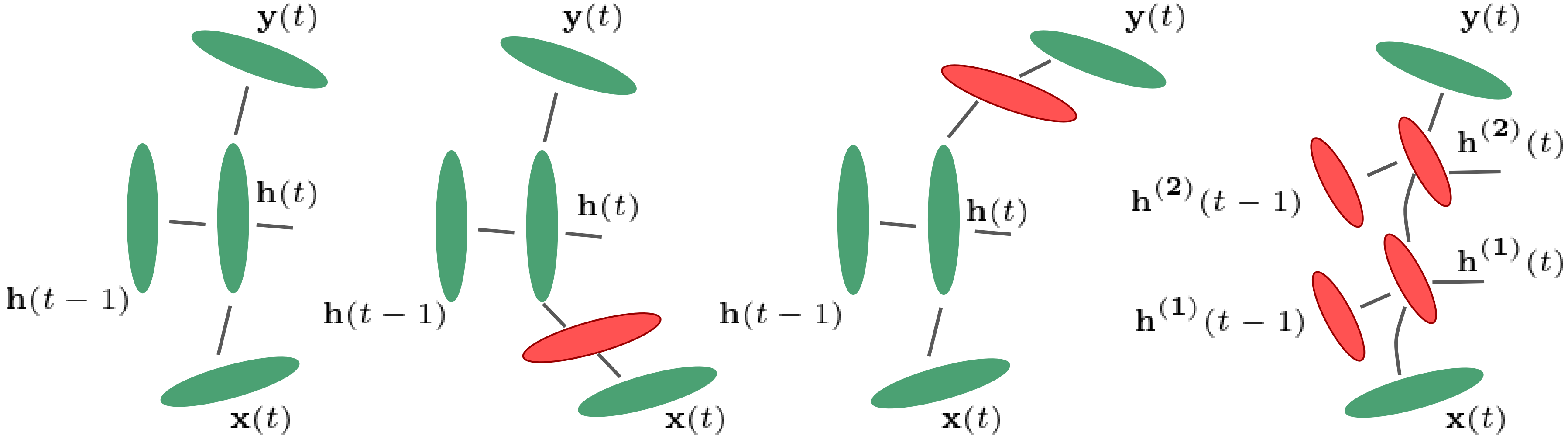




Tanaka, G., Yamane, T., Héroux, J.B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D. and Hirose, A., 2019. Recent advances in physical reservoir computing: A review. Neural Networks, 115, pp.100-123.

Pascanu, R., Gulcehre, C., Cho, K. and Bengio, Y., 2013. How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026.

Depth in RNNs



shallow

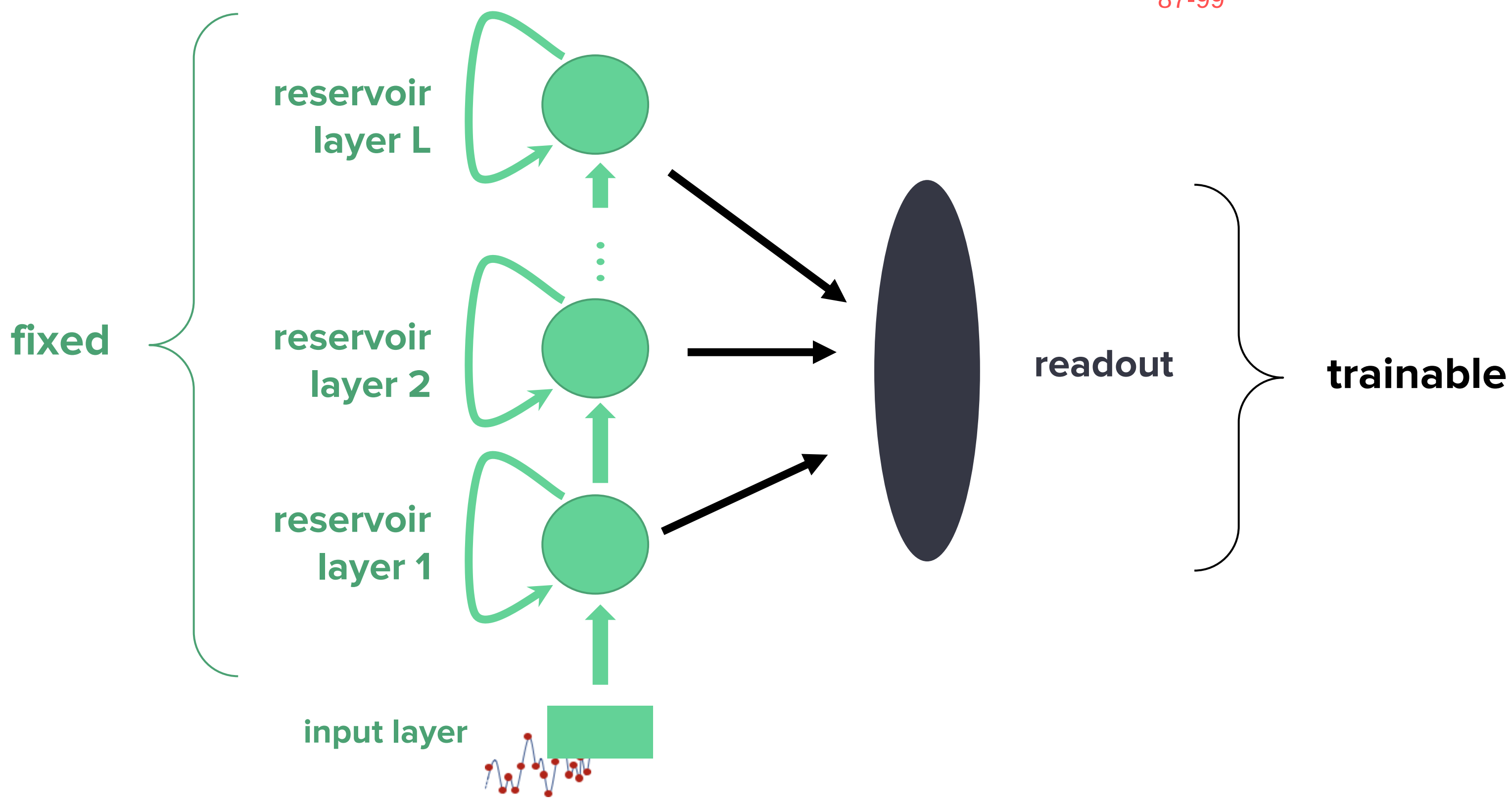
deep input

deep readout

deep reservoir

Deep Echo State Networks

Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli. "Deep reservoir computing: A critical experimental analysis." Neurocomputing 268 (2017): 87-99



Deep Echo State Networks

Deep reservoir = nested set of dynamical systems

$$\boxed{\mathbf{h}^{(\mathbf{L})}(t)} = \tanh(\mathbf{W}^{(L)}\boxed{\mathbf{h}^{(L-1)}(t)} + \mathbf{W_R}^{(L)}\mathbf{h}^{(L)}(t-1))$$

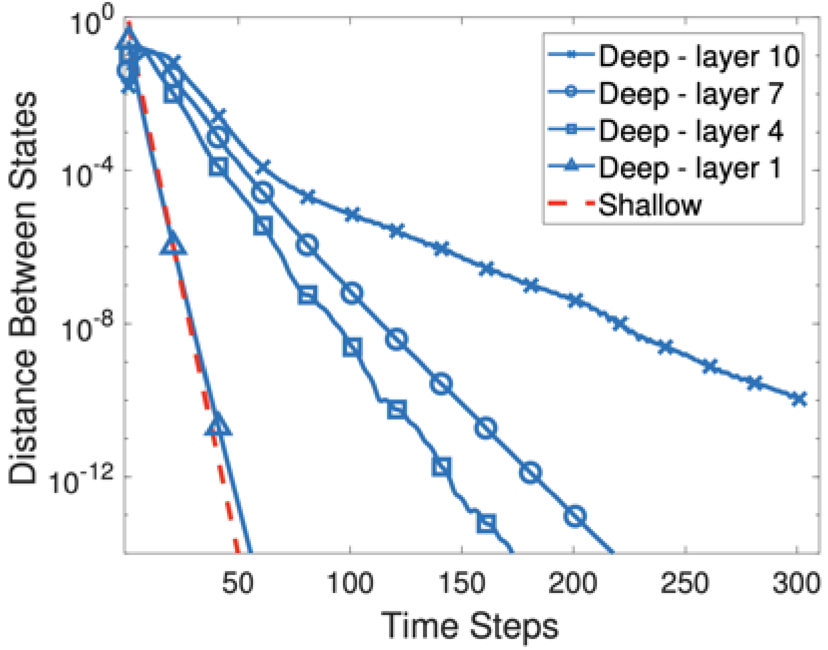
$$\vdots$$

$$\boxed{\mathbf{h}^{(\mathbf{2})}(t)} = \tanh(\mathbf{W}^{(2)}\boxed{\mathbf{h}^{(1)}(t)} + \mathbf{W_R}^{(2)}\mathbf{h}^{(2)}(t-1))$$

$$\boxed{\mathbf{h}^{(\mathbf{1})}(t)} = \tanh(\mathbf{W}^{(1)}\boxed{\mathbf{x}(t)} + \mathbf{W_R}^{(1)}\mathbf{h}^{(1)}(t-1))$$

Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli. "Deep reservoir computing: A critical experimental analysis." Neurocomputing 268 (2017): 87-99

Multiple time-scales



- Effects of input perturbations last longer in the higher reservoir layers
- Multiple time-scales representation is intrinsic

Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli. "Deep reservoir computing: A critical experimental analysis." Neurocomputing 268 (2017): 87-99

Gallicchio, C. and Micheli, A., 2018, July. Why Layering in Recurrent Neural Networks? A DeepESN Survey. In 2018 International Joint Conference on Neural Networks (IJCNN)(pp. 1-8). IEEE.

Richer dynamics: short-term memory

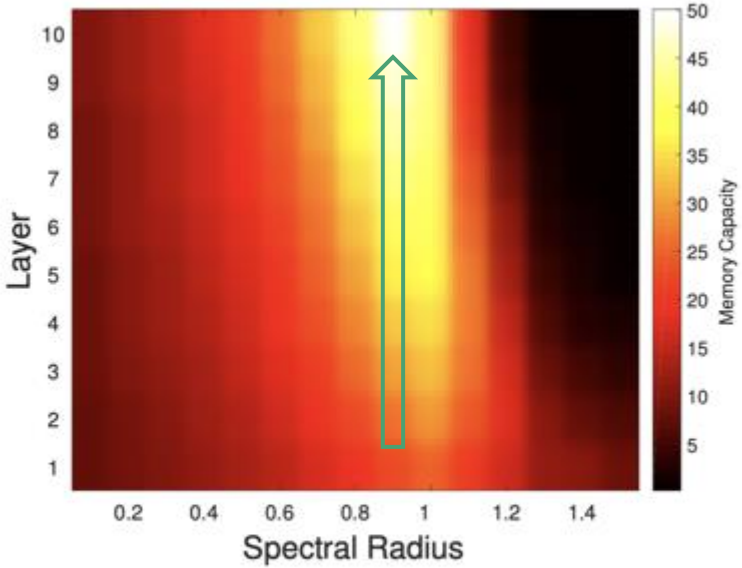
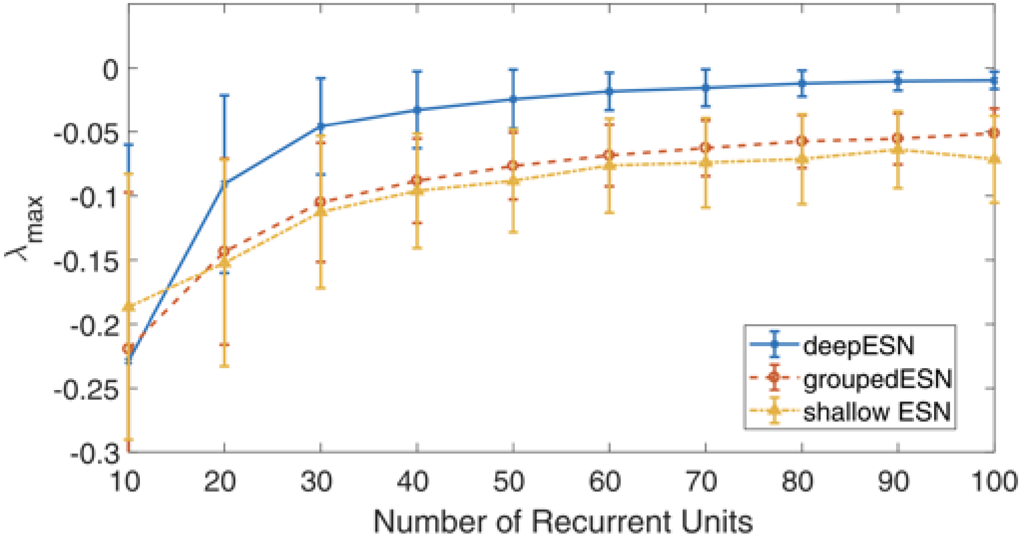


Fig. 7. Averaged Memory Capacity of individual layers of DeepESN, shown for increasing values of the spectral radius, as computed in [49]. Results correspond to a DeepESN setting in which each layer comprises a number of 100 recurrent reservoir units. Differently from the results in Figure 6, Memory Capacity in this plot refers to reservoir-readout connections that are trained separately for each layer. Further details and results can be found in [49].

Gallicchio, C. and Micheli, A., 2018, July. Why Layering in Recurrent Neural Networks? A DeepESN Survey. In 2018 International Joint Conference on Neural Networks (IJCNN)(pp. 1-8). IEEE.

Richer dynamics: stability regime



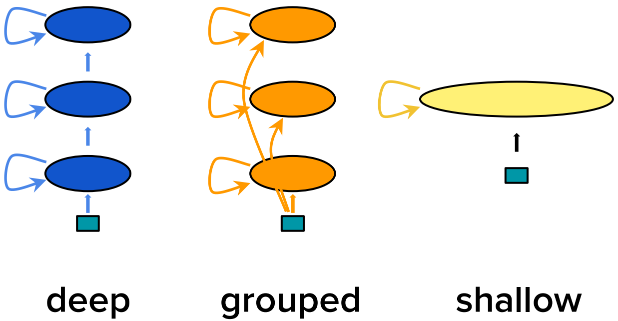
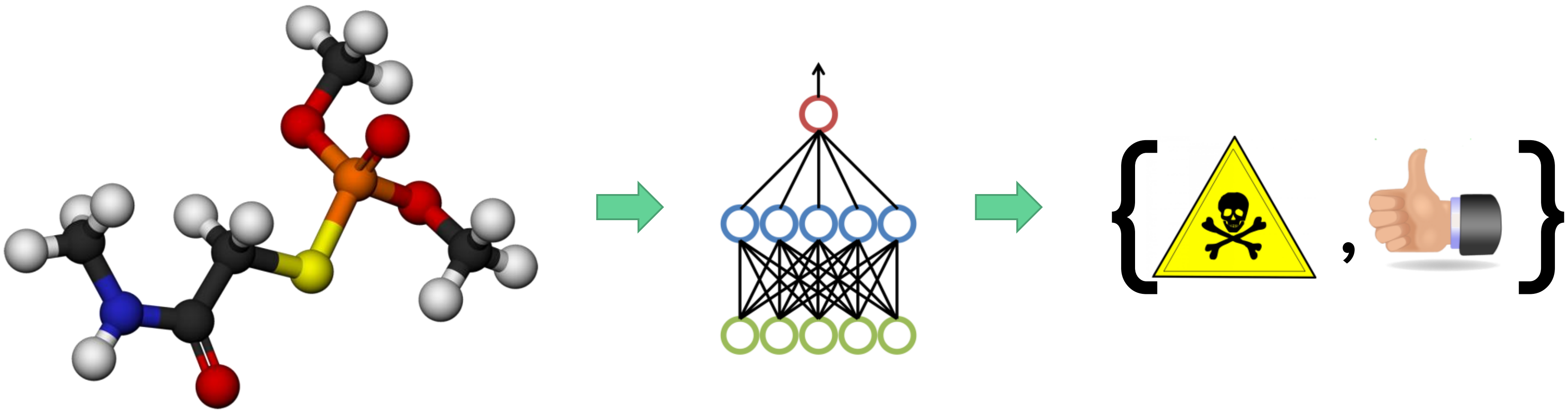


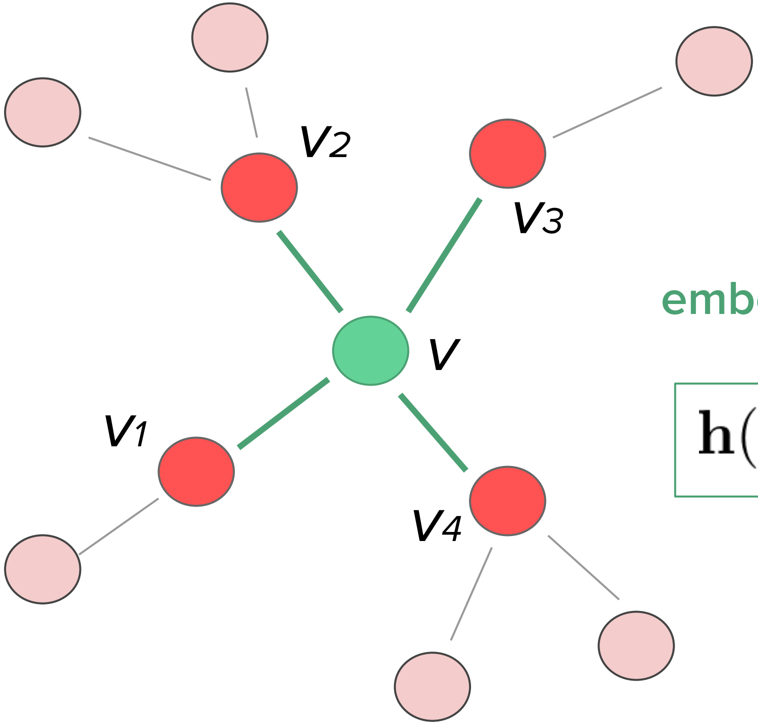
Fig. 4. Averaged values of λ_{max} obtained by DeepESN for increasing number of reservoir units, organized in layers of 10 units each. The considered hyper-parameterization corresponds to $\rho = 1$, $a = 1$, $scale_{in} = 1$ and $scale_{il} = 0.5$. Results achieved by a shallow ESN and groupedESN with the same hyper-parameterization and the same number of reservoir units are reported as well for the sake of comparison.

Gallicchio, C., Micheli, A. and Silvestri, L., 2018. Local lyapunov exponents of deep echo state networks. Neurocomputing, 298, pp.34-45.

Neural networks for graphs



Vertex-wise graph encoding



- time-step → vertex
- previous time step → neighborhood

embedding (state)

$$\boxed{\mathbf{h}(v)} = \tanh\left(\mathbf{W}\boxed{\mathbf{x}(v)} + \sum_{v' \in \mathcal{N}(v)} \mathbf{W}_{\mathbf{R}}\boxed{\mathbf{h}(v')}\right)$$

input features

embeddings of neighbors

Reservoir Computing for graphs

$$\mathbf{H} = \tanh\left(\mathbf{W}\mathbf{X} + \mathbf{W}_{\mathbf{R}}\mathbf{H}\tilde{\mathbf{A}}\right)$$

- Basic idea: encode the input graph as the fixed point of a dynamical system
- Impose stability of the iterated map - Graph Embedding Stability (GES)
    - E.g., $\rho(\mathbf{W}_{\mathbf{R}}) < 1$

Reservoir Layer for graphs

$$\mathbf{H}[t] = \tanh(\mathbf{W}\mathbf{X} + \mathbf{W}_{\mathbf{R}}\mathbf{H}[t-1]\tilde{\mathbf{A}})$$

- Initialize randomly under the GES condition
- For each graph in your dataset:

1. Initialize H[0] (e.g., to 0)
2. Iterate the above equation until convergence

Gallicchio, C. and Micheli, A., 2020. Fast and Deep Graph Neural Networks. In AAAI (pp. 3898-3905).

Fast and Deep Graph Neural Networks (FDGNN)

$$\boxed{\mathbf{H}^{(L)*}}$$

$$\mathbf{H}^{(L)}[t] = \tanh(\mathbf{W}^{(L)}\mathbf{H}^{(L-1)*} + \mathbf{W_R}^{(L)}\mathbf{H}^{(L)}[t-1]\tilde{\mathbf{A}})$$

$$\boxed{\mathbf{H}^{(L-1)*}}$$

$$\vdots$$

$$\boxed{\mathbf{H}^{(2)*}}$$

$$\mathbf{H}^{(2)}[t] = \tanh(\mathbf{W}^{(2)}\mathbf{H}^{(1)*} + \mathbf{W_R}^{(2)}\mathbf{H}^{(2)}[t-1]\tilde{\mathbf{A}})$$

$$\boxed{\mathbf{H}^{(1)*}}$$

$$\mathbf{H}^{(1)}[t] = \tanh(\mathbf{W}^{(1)}\mathbf{X} + \mathbf{W_R}^{(1)}\mathbf{H}^{(1)}[t-1]\tilde{\mathbf{A}})$$

Gallicchio, C. and Micheli, A., 2020. Fast and Deep Graph Neural Networks. In AAAI (pp. 3898-3905).

Gallicchio, C. and Micheli, A., 2020. Fast and Deep Graph Neural Networks. In AAAI (pp. 3898-3905).

It's accurate

	MUTAG	PTC	COX2	PROTEINS	NCI1
FDGNN	**88.51** ± 3.77	**63.43** ± 5.35	**83.39** ± 2.88	**76.77** ± 2.86	77.81 ± 1.62
FDGNN$_{(L=1)}$	87.38 ± 6.55	**63.43** ± 5.35	82.41 ± 2.67	**76.77** ± 2.86	77.11 ± 1.52
GNN (Uwents et al. 2011)	80.49 ± 0.81	-	-	-	-
RelNN (Uwents et al. 2011)	87.77 ± 2.48	-	-	-	-
DGCNN (Zhang et al. 2018)	85.83 ± 1.66	58.59 ± 2.47	-	75.54 ± 0.94	74.44 ± 0.47
PGC-DGCNN (Tran, Navarin, and Sperduti 2018)	87.22 ± 1.43	61.06 ± 1.83	-	76.45 ± 1.02	76.13 ± 0.73
DCNN (Tran, Navarin, and Sperduti 2018)	-	-	-	61.29 ± 1.60	56.61 ± 1.04
PSCN (Tran, Navarin, and Sperduti 2018)	-	-	-	75.00 ± 2.51	76.34 ± 1.68
GK (Zhang et al. 2018)	81.39 ± 1.74	55.65 ± 0.46	-	71.39 ± 0.31	62.49 ± 0.27
DGK (Yanardag and Vishwanathan 2015)	82.66 ± 1.45	57.32 ± 1.13	-	71.68 ± 0.50	62.48 ± 0.25
RW (Zhang et al. 2018)	79.17 ± 2.07	55.91 ± 0.32	-	59.57 ± 0.09	-
PK (Zhang et al. 2018)	76.00 ± 2.69	59.50 ± 2.44	81.00 ± 0.20	73.68 ± 0.68	82.54 ± 0.47
WL (Zhang et al. 2018)	84.11 ± 1.91	57.97 ± 2.49	83.20 ± 0.20	74.68 ± 0.49	**84.46** ± 0.45
KCNN (Nikolentzos et al. 2018)	-	62.94 ± 1.69	-	75.76 ± 0.28	77.21 ± 0.22
CGMM (Bacciu, Errica, and Micheli 2018)	85.30	-	-	-	-

Gallicchio, C. and Micheli, A., 2020. Fast and Deep Graph Neural Networks. In AAAI (pp. 3898-3905).

It's accurate

	IMDB-b	IMDB-m	REDDIT	COLLAB
FDGNN	**72.36**$_{\pm 3.63}$	**50.03**$_{\pm 1.25}$	**89.48**$_{\pm 1.00}$	74.44$_{\pm 2.02}$
FDGNN$_{(L=1)}$	71.79$_{\pm 3.37}$	49.34$_{\pm 1.70}$	87.74$_{\pm 1.61}$	73.82$_{\pm 2.32}$
DGCNN (Zhang et al. 2018)	70.03$_{\pm 0.86}$	47.83$_{\pm 0.85}$	-	73.76$_{\pm 0.49}$
PGC-DGCNN (Tran, Navarin, and Sperduti 2018)	71.62$_{\pm 1.22}$	47.25$_{\pm 1.44}$	-	**75.00**$_{\pm 0.58}$
PSCN (Tran, Navarin, and Sperduti 2018)	71.00$_{\pm 2.29}$	45.23$_{\pm 2.84}$	-	72.60$_{\pm 2.15}$
GK (Yanardag and Vishwanathan 2015)	65.87$_{\pm 0.98}$	43.89$_{\pm 0.38}$	77.34$_{\pm 0.18}$	72.84$_{\pm 0.56}$
DGK (Yanardag and Vishwanathan 2015)	66.96$_{\pm 0.56}$	44.55$_{\pm 0.52}$	78.04$_{\pm 0.39}$	73.09$_{\pm 0.25}$
KCNN (Nikolentzos et al. 2018)	71.45$_{\pm 0.15}$	47.46$_{\pm 0.21}$	81.85$_{\pm 0.12}$	74.93$_{\pm 0.14}$

It’s fast

Gallicchio, C. and Micheli, A., 2020. Fast and Deep Graph Neural Networks. In AAAI (pp. 3898-3905).

Table 3: Running times required by FDGNN (in single core mode, without GPU acceleration). Results are averaged (and std are computed) on the outer 10 folds.

Task	Training	Test
MUTAG	$0.56''_{\pm 0.33}$	$0.06''_{\pm 0.04}$
PTC	$0.16''_{\pm 0.03}$	$0.02''_{\pm 0.00}$
COX2	$1.36''_{\pm 0.42}$	$0.15''_{\pm 0.05}$
PROTEINS	$2.16''_{\pm 0.47}$	$0.24''_{\pm 0.04}$
NCI1	$2.00'_{\pm 0.45}$	$13.36''_{\pm 3.02}$
IMDB-b	$7.46''_{\pm 3.14}$	$0.83''_{\pm 0.35}$
IMDB-m	$8.68''_{\pm 1.73}$	$0.98''_{\pm 0.22}$
REDDIT	$2.47'_{\pm 0.01}$	$16.49''_{\pm 0.28}$
COLLAB	$22.86'_{\pm 4.70}$	$2.54'_{\pm 0.52}$

Table 4: Comparison of training times required on MUTAG by FDGNN, GNN, GIN and WL. Results are averaged (and std are computed) on the outer 10 folds.

FDGNN	GNN	GIN	WL
$\mathbf{0.56''}_{\pm 0.33}$	$202.28''_{\pm 166.87}$	$499.24''_{\pm 2.25}$	$1.16''_{\pm 0.03}$

Reservoir Computing by discretizing ODEs

Gallicchio, C., Reservoir Computing by discretizing ODEs. ESANN 2021

- Euler State Networks (EuSN)
    - stable dynamics + non-dissipation of the input over time

$$h' = \tanh(W_x\, x + W_h h + b)$$

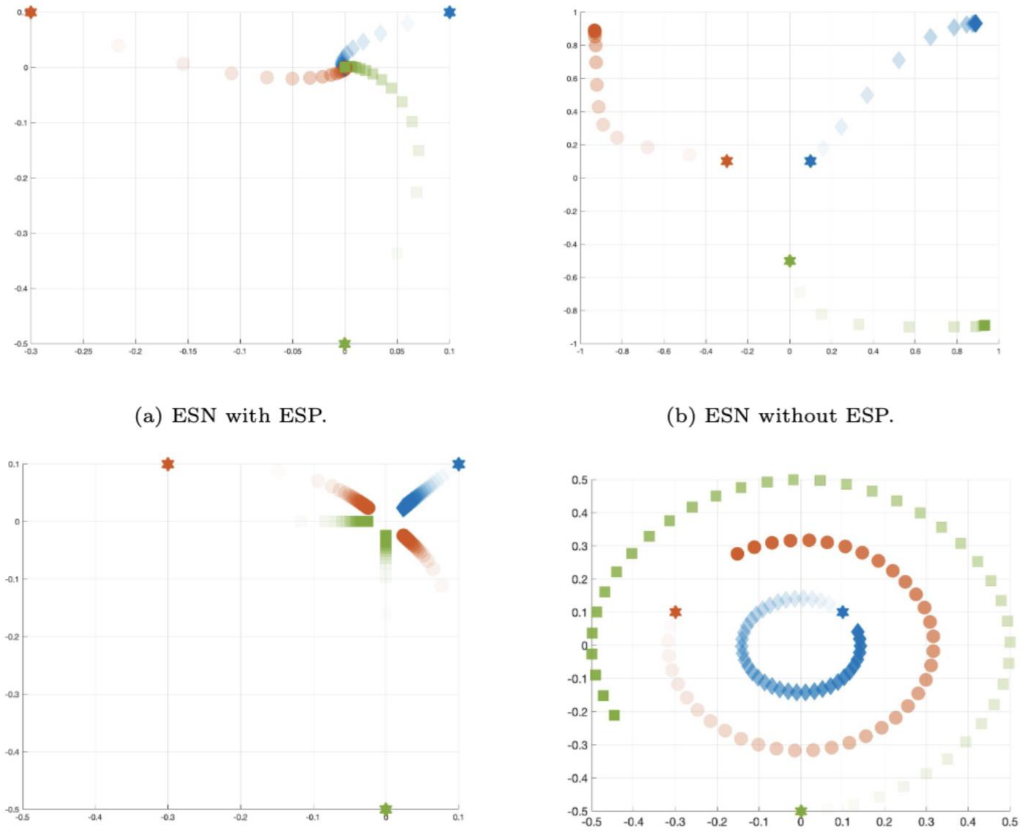
1. impose antisymmetric recurrent weight matrix to enforce critical dynamics
2. discretize the ODE

$$h_t = \mathrm{h}_{\mathrm{t}-1} + \epsilon \tanh\left(W_x\, x_{\mathrm{t}} + (W_h - W_h^T - \gamma I)\, h_{\mathrm{t}-1} + b\right)$$

step size

diffusion coefficient

The input signal is preserved without exploding nor dying



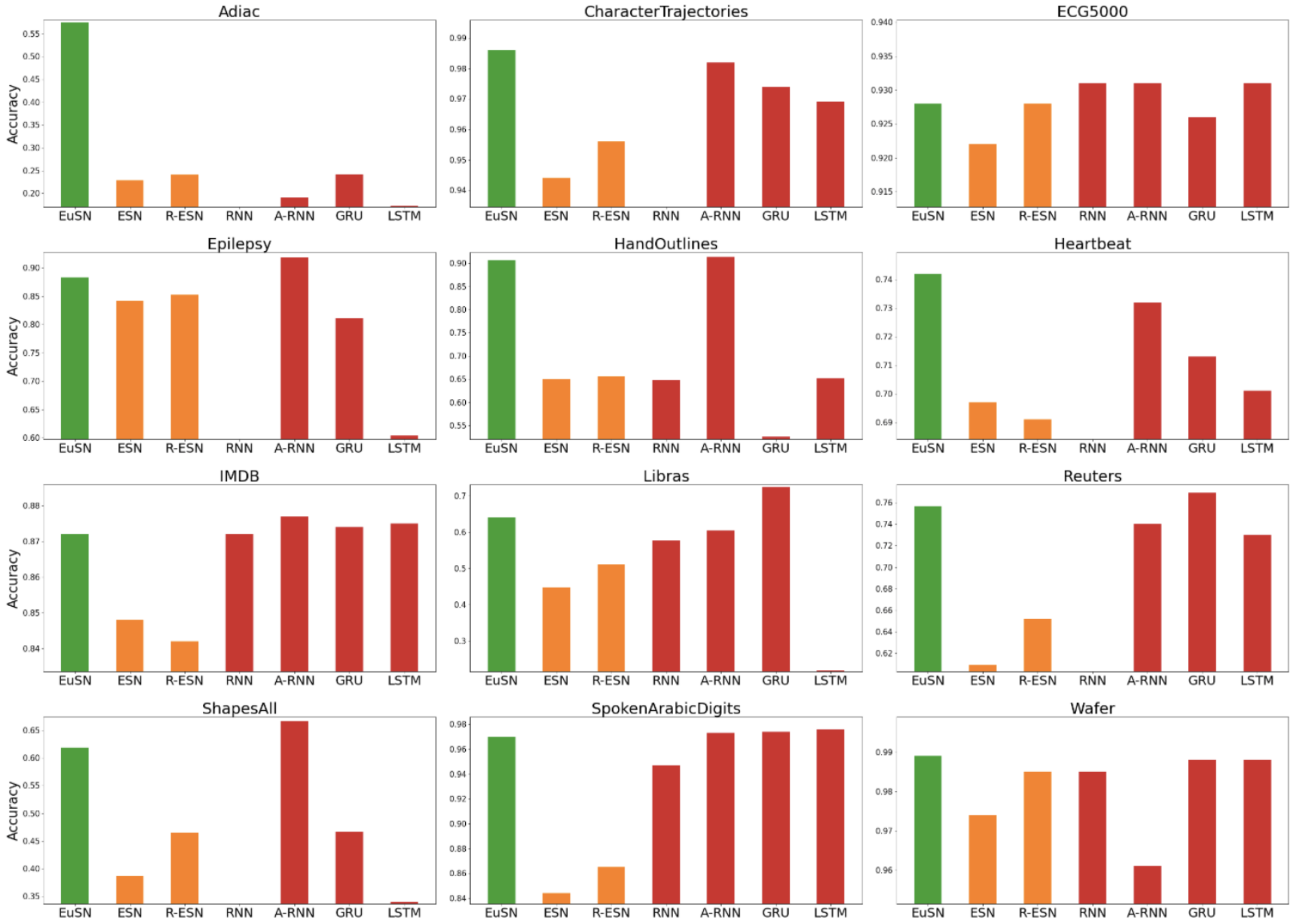
(a) ESN with ESP.

(b) ESN without ESP.

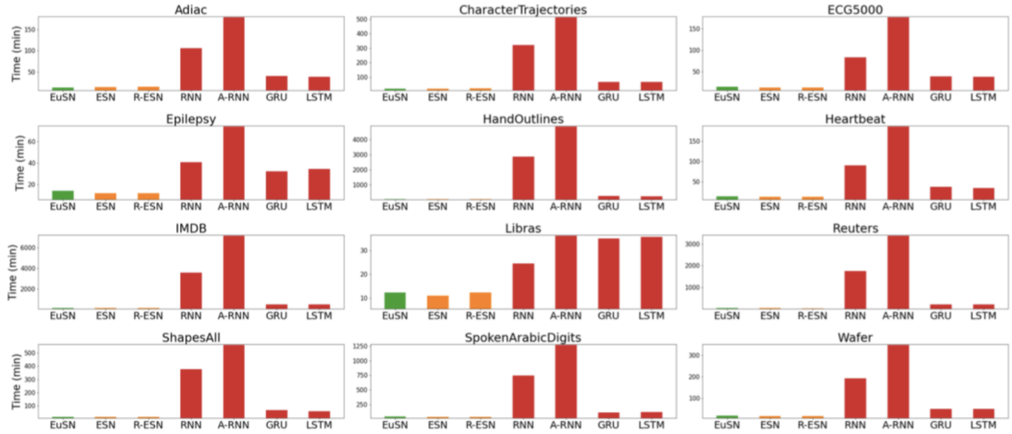
(c) R-ESN.

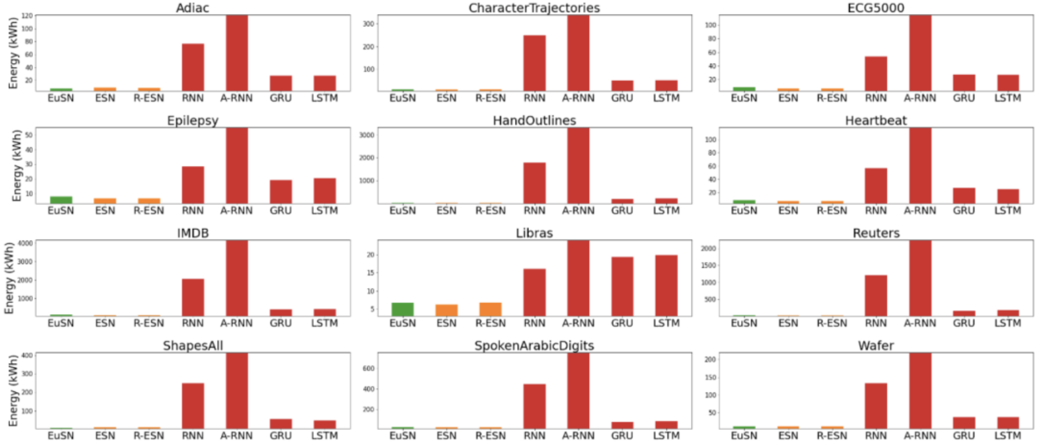
(d) EuSN.

Gallicchio, Claudio. "Euler State Networks." arXiv preprint arXiv:2203.09382 (2022).



High accuracy vs state-of-the-art fully trainable models & ESNs





Extremely more efficient (up to 100x) than fully trainable models

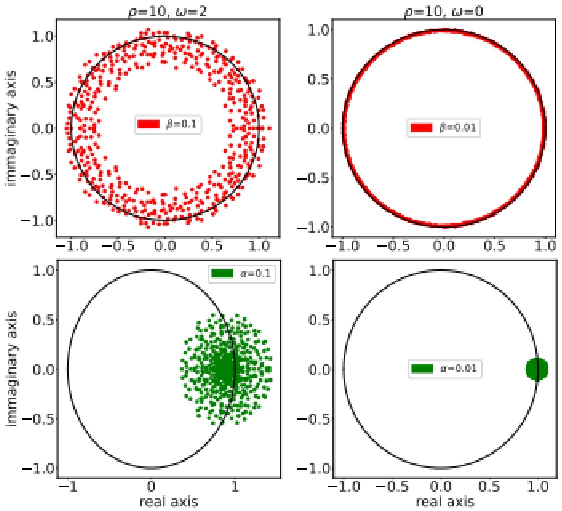
Edge of Stability Reservoir Computing

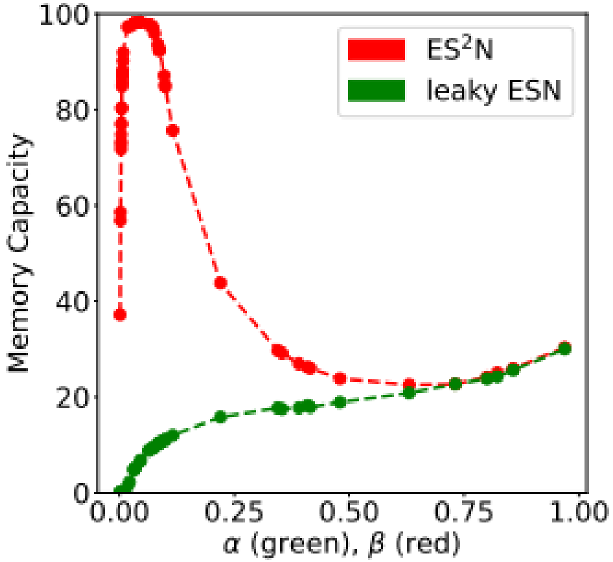
Put a random orthogonal matrix here transforming the hidden state

$$\boldsymbol{h}_t = (1-\alpha)\boldsymbol{h}_{t-1} + \alpha \tanh(\boldsymbol{x}_t \boldsymbol{W}_{xh} + \boldsymbol{h}_{t-1}\boldsymbol{W}_{hh}) \quad \text{Leaky ESN}$$

$$\boldsymbol{h}_t = (1-\beta)\ \boldsymbol{O}\ \boldsymbol{h}_{t-1} + \beta \tanh(\boldsymbol{x}_t \boldsymbol{W}_{xh} + \boldsymbol{h}_{t-1}\boldsymbol{W}_{hh}) \quad \boldsymbol{ES^2N}$$

Ceni, Andrea, and Claudio Gallicchio. "Edge of stability echo state networks." *arXiv preprint arXiv:2308.02902* (2023).

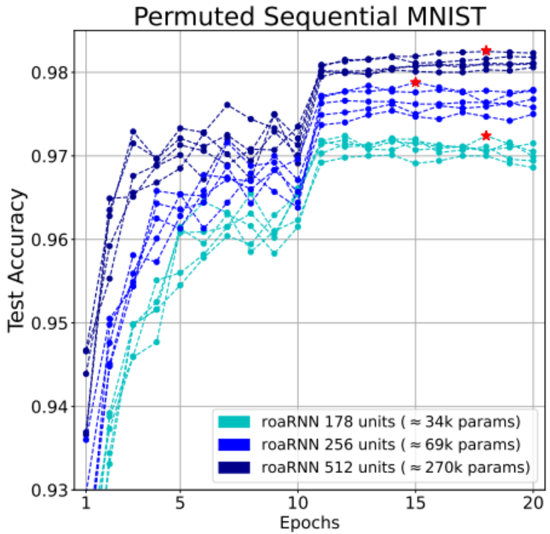




RoaRNN: fully trained ESESN

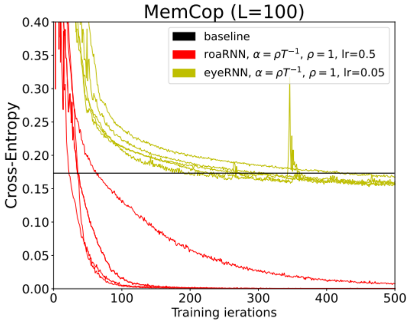
A simple solution to the V/E gradient issue!

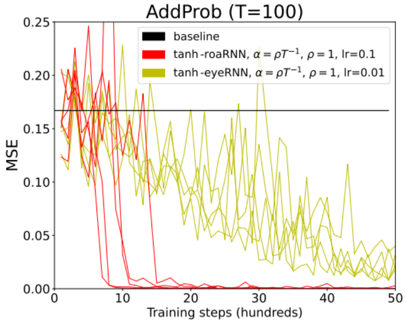
Ceni, Andrea. "Random orthogonal additive filters: a solution to the vanishing/exploding gradient of deep neural networks." *arXiv preprint arXiv:2210.01245* (2022).

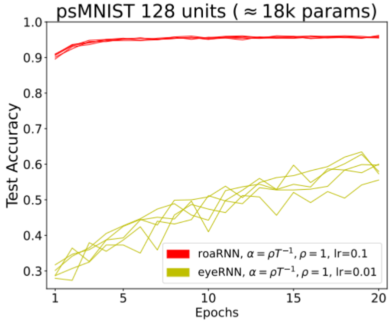


Model	Parameters	Test accuracy
Vanilla RNN [46]	≈68k	71.6%
LSTM [31]	≈270k	92.9%
GRU [46]	≈200k	94.1%
Dilated CNN [46]	≈46k	96.7%
FC uRNN [29]	≈270k	94.1%
BN LSTM [57]	–	95.4%
expRNN [30]	≈137k	96.6%
Lipschitz RNN [42]	≈ 34k	96.3%
res-IndRNN [56]	–	97.02%
dense-IndRNN [56]	–	97.2%
Shuffling RNN [40]	≈50k	96.43%
NRU [45]	≈165k	95.38%
LMU [44]	≈102k	97.15%
coRNN [43]	≈134k	97.3%
roaRNN	≈ 34k	97.24%
roaRNN	≈ 69k	97.88%
roaRNN	≈270k	98.25%

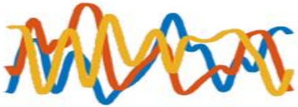
The identity in place of a random orthogonal matrix doesn't give same results!







STATE SPACE MODELS



Input sequence

S4 layer

nonlinearity

S4 layer

nonlinearity

S4 layer

Output sequence

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$
$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)$$

$u_k = u(k\Delta)$

discretisation

$$x_k = \overline{\boldsymbol{A}}x_{k-1} + \overline{\boldsymbol{B}}u_k$$
$$y_k = \overline{\boldsymbol{C}}x_k$$

Unfold in time

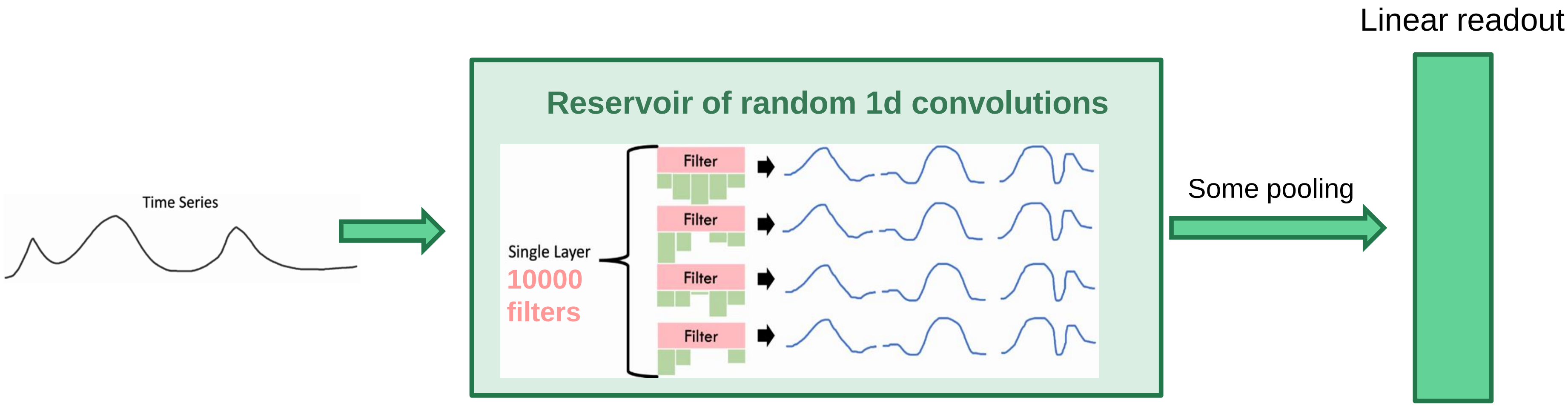
Gu, Albert, Karan Goel, and Christopher Ré. "Efficiently modeling long sequences with structured state spaces." *arXiv preprint arXiv:2111.00396* (2021).

$$y_k = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{k-1}\overline{\boldsymbol{B}}u_1 + \cdots + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}}u_k$$
$$y = \overline{\boldsymbol{K}} * u.$$

Convolutional representation

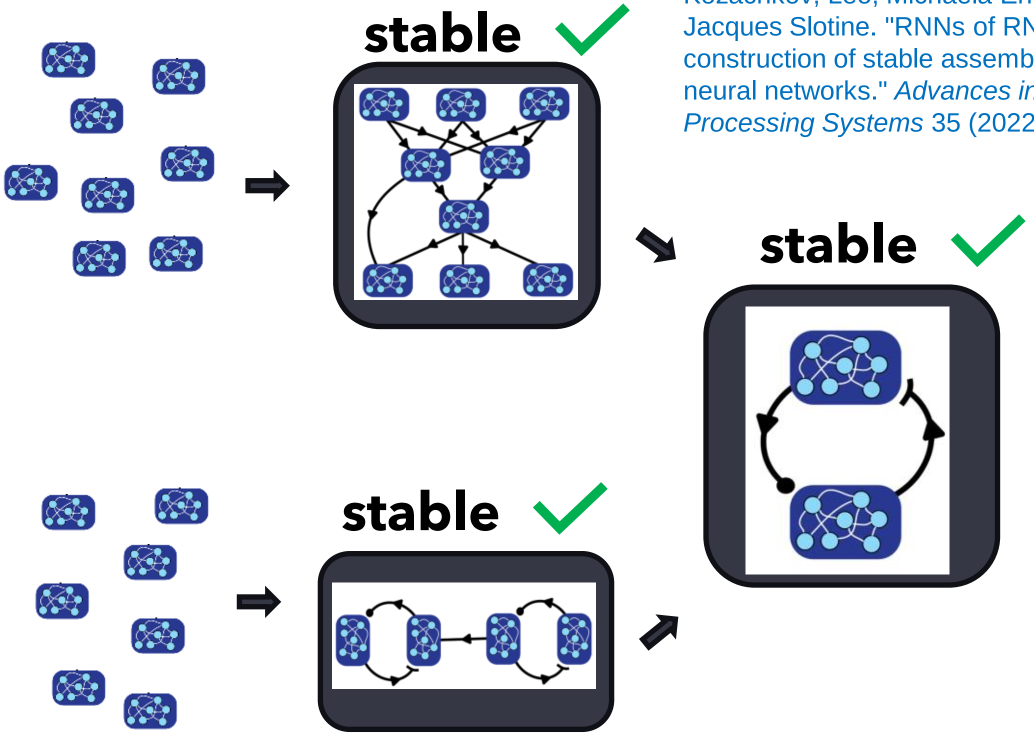
$$\overline{\boldsymbol{K}} \in \mathbb{R}^L := \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}) := \left(\overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^i\overline{\boldsymbol{B}}\right)_{i\in[L]} = (\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}).$$

ROCKET

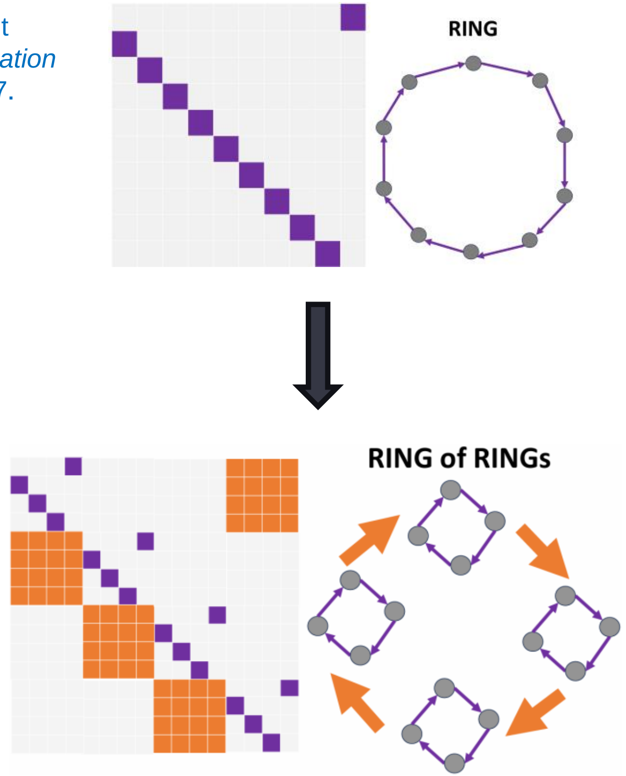


Dempster, Angus, François Petitjean, and Geoffrey I. Webb. "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels." *Data Mining and Knowledge Discovery* 34.5 (2020): 1454-1495.

MODULAR COMPOSITION of RNNs

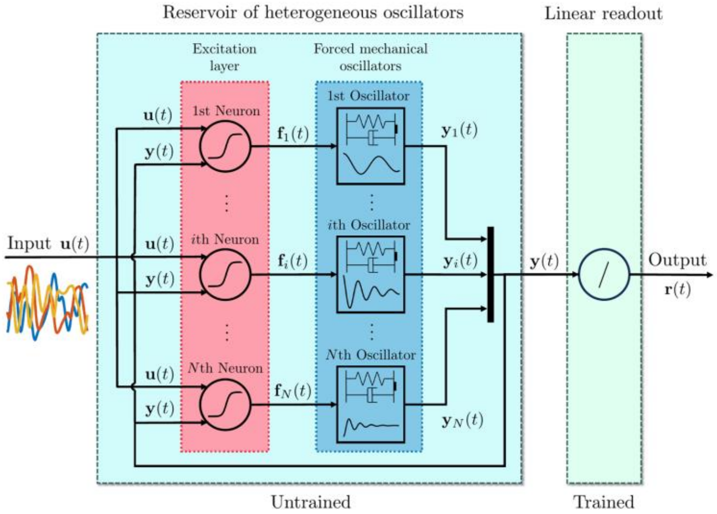


Kozachkov, Leo, Michaela Ennis, and Jean-Jacques Slotine. "RNNs of RNNs: Recursive construction of stable assemblies of recurrent neural networks." *Advances in Neural Information Processing Systems* 35 (2022): 30512-30527.



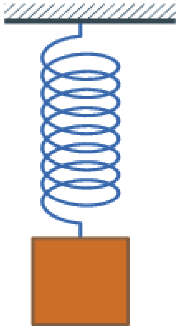
RANDOM OSCILLATORS NETWORK

Ceni, Andrea, et al. "Random Oscillators Network for Time Series Processing." *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024.



Reservoir of oscillators equation

$$\ddot{\mathbf{y}} = \tanh(\mathbf{W}\mathbf{y} + \mathbf{V}\mathbf{u}(t) + \mathbf{b}) - \gamma \odot \mathbf{y} - \varepsilon \odot \dot{\mathbf{y}}$$



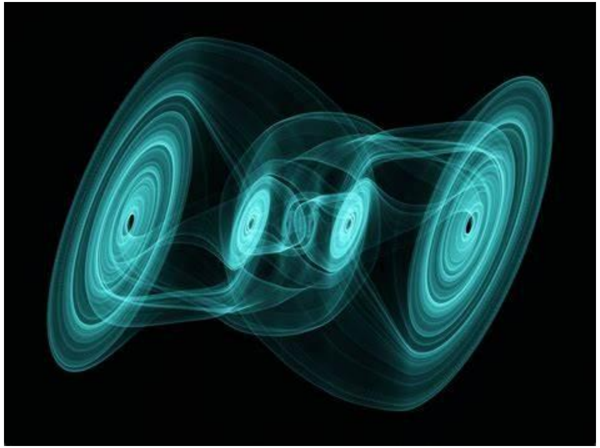
Linear readout:

$$\mathbf{r} = \mathbf{W}_o\mathbf{y} + \mathbf{b}_o$$

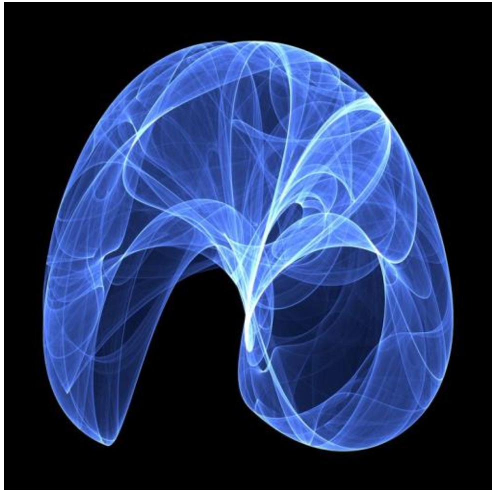
Model	Fully-trained			Randomised	
	LSTM	coRNN	hcoRNN (our)	Leaky ESN	RON (our)
sMNIST ↑	$0.9860_{0.0017}$	**$0.9921_{0.0002}$**	$0.9871_{0.0011}$	$0.9211_{0.0020}$	**$0.9780_{0.0006}$**
psMNIST ↑	$0.8761_{0.0390}$	$0.9435_{0.0224}$	**$0.9635_{0.0048}$**	$0.8503_{0.0150}$	**$0.9301_{0.0054}$**
npCIFAR-10 ↑	$0.1000_{0.0000}$	**$0.5841_{0.0033}$**	$0.5548_{0.0031}$	$0.2060_{0.0016}$	**$0.4158_{0.0101}$**
FordA ↑	$0.5803_{0.0432}$	$0.7003_{0.1535}$	**$0.7944_{0.0859}$**	$0.5461_{0.0320}$	**$0.6885_{0.0385}$**
Adiac ↑	$0.4793_{0.0187}$	$0.4517_{0.0252}$	**$0.5586_{0.0706}$**	$0.6928_{0.0116}$	**$0.7313_{0.0050}$**
Lorenz96 ↓	$2.4 \times 10^{-1}_{3.6\times10^{-2}}$	**$2.1 \times 10^{-1}_{5.2\times10^{-2}}$**	$2.6 \times 10^{-1}_{2.5\times10^{-2}}$	$2.0 \times 10^{-3}_{2.0\times10^{-4}}$	**$1.6 \times 10^{-3}_{1.7\times10^{-4}}$**
Mackey-Glass ↓	**$3.4 \times 10^{-2}_{3.2\times10^{-3}}$**	$6.2 \times 10^{-2}_{1.5\times10^{-2}}$	$5.4 \times 10^{-2}_{4.9\times10^{-3}}$	$3.0 \times 10^{-2}_{1.4\times10^{-3}}$	**$1.8 \times 10^{-2}_{6.5\times10^{-3}}$**

ATTRACTORS & AWARENESS

Meta-cognition awareness attractor



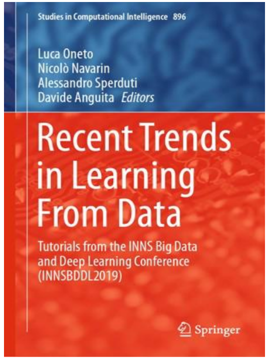
Spatial awareness attractor



Can we map dimensions of awareness into attractor’s geometrical properties?

Resources

Deep Randomized Neural Networks



Gallicchio, C. and Scardapane, S., 2020. Deep Randomized Neural Networks. In *Recent Trends in Learning From Data* (pp. 43-68). Springer, Cham.

arXiv.org https://arxiv.org/pdf/2002.12287.pdf

AAAI-21 tutorial website:

https://sites.google.com/site/cgallicch/resources/tutorial_DRNN

Reservoir Computing NNs

IJCNN 2021 Tutorial: Reservoir Computing: Randomized Recurrent Neural Networks

You Tube https://www.youtube.com/watch?v=XJg7VdN7g-0

SSIE | SUMMER PHD SCHOOL

SSIE Summer PhD School of Information Engineering: Reservoir Recurrent Neural Networks

You Tube https://www.youtube.com/watch?v=1K7oJCtTzKQ

IEEE Task Force on Reservoir Computing

https://sites.google.com/view/reservoir-computing-tf/

Promote and stimulate the development of Reservoir Computing research under both theoretical and application perspectives.

SCAN ME

Summary

- Reservoir Computing: paradigm for designing and training RNNs
  - the dynamical reservoir is initialized to be stable (ESP) and left untrained
  - the readout is trained to solve the learning task
- Fast (& simple) training compared to standard RNNs
- The intrinsic state space organization explains the good results on tasks featured by Markovian characterizations
  - Good for sensor data
- Very active area of research...
  - Deep Reservoir Computing
  - Embedded applications
  - Neuromorphic AI
  - Stable RNN architectures
  - Graph Neural Networks

Some arguments for thesis

- **Oscillators-like networks**
- CNNs for sequential processing (ROCKET-like)
- Attractors and Awareness
- Modular compositions of RNNs
- Structured State Space Models
- Improve Edge of Stability RNNs

Reservoir Computing

Andrea Ceni
andrea.ceni@di.unipi.it