Reservoir Computing

Claudio Gallicchio
University of Pisa, Italy
Contact Information

Research focus:
dynamical recurrent neural models, reservoir computing, stable deep neural networks architectures

MSc Thesis supervision on:
Recurrent Neural Networks, Reservoir Computing, Stable Neural Architectures, Neural ODEs, ...

Claudio Gallicchio, Ph.D.
Assistant Professor
web: www.di.unipi.it/~gallicch
email: gallicch@di.unipi.it
tel.:+39 050 2213145
Contact Information

Claudio Gallicchio
Assistant Professor at the Department of Computer Science, University of Pisa
Email verificata su di unipi.it - Home page
Deep Learning, Neural Networks, Recurrent Neural Networks, Reservoir Computing, Echo State Networks

TITOLO | CITATA DA | ANNO
--- | --- | ---
Deep reservoir computing: A critical experimental analysis | 235 | 2017
C Gallicchio, A Micheli, L Pedrelli
Neurocomputing 268, 87-96

Architectural and markovian factors of echo state networks | 147 | 2011
C Gallicchio, A Micheli
Neural Networks 24 (5), 440-456

Human activity recognition using multisensor data fusion based on reservoir computing | 126 | 2016
F Palumbo, C Gallicchio, R Pucci, A Micheli
Journal of Ambient Intelligence and Smart Environments 8 (2), 87-107

Graph echo state networks | 114 | 2010
C Gallicchio, A Micheli
Neural Networks (IJCNN), The 2010 International Joint Conference on, 1-8

https://twitter.com/claudiogallicc1

https://www.linkedin.com/in/claudio-gallicchio-05a47038/
Reservoir Computing

= Extremely efficient way of designing and training RNNs
Recurrent Neural Networks

- State update:
  \[ h_t = \tanh(x_t W_{xh} + h_{t-1} W_{hh}) \]

- Output function:
  \[ y_t = h_t W_{hy} \]
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
Approaches

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

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

\[ x_t \]

\[ c_t \]

\[ h_t \]

\[ y_t \]

\[ c_{t-1} \]

\[ h_{t-1} \]

\text{forget gate}

\text{input gate}

\text{output gate}

standard RNN layer
Long-term dependencies
LSTM equations

vanilla RNN

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

extra computation

- \( \) extra parameters

training is slow
(computationally intensive)
Randomization in Deep Neural Networks
Energy consumption matters!

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


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.
Quantifying the carbon emissions of ML


https://mlco2.github.io/impact/
Energy consumption matters!

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

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 $2 \times 10^{17}$ 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.

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 outsized environmental impact.
vs the Brain...

≈30 PFlops
10 MW vs 20 W

memory and computing are co-located
$10^{11}$ neurons, $10^{15}$ synapses
10000 synapses/neuron

...Neuromorphic Computing
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://bitalino.com/en/freestyle-kit-bt

Source: https://www.eenewseembedded.com/news/raspberry-pi-3-now-compute-module-format
Deep Neural Networks

Powerful representations by applying multiple non-linear levels of transformation

Deep Learning = Architectural Biases + Learning Algorithms
Complexity / Accuracy Tradeoff

- Deep Randomized NNs
- SVMs-like
- Deep NNs
- Linear models
The Philosophy

“Randomization is computationally cheaper than optimization”


Randomized Recurrent Neural Networks

\[ y_t = g \circ f_R(x_t, h_{t-1}) \]

\[ h_t = f_R(x_t, h_{t-1}) \]
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

- 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)

Reservoir Computing
Reservoir Computing: focus on the dynamical system

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

Randomly initialized under stability conditions on the dynamical system

Stable dynamics - Echo State Property

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

PETER TÑO
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
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

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
  \[ h_t = \tanh(x_t W_{xh} + h_{t-1} W_{hh}) \]
  - large layer of recurrent units
  - sparsely connected
  - randomly initialized under the ESP
  - untrained

- Input layer
- Reservoir
- Readout
**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

\[ F: \mathbb{R}^{N_x} \times \mathbb{R}^{N_H} \to \mathbb{R}^{N_H} \]

\[ h_t = F(x_t, h_{t-1}) \]

\[ = \tanh(x_t W_{xh} + h_{t-1} W_{hh}) \]
Reservoir: State Transition Function

Iterated version of state transition function

- given an input sequence $s = [x_1, x_2, \ldots, x_t]$
- return the final state $h_t$

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

input sequence $s$

initial state $h_0$

final state $h_t$
Iterated version of state transition function

\[ \hat{F}(s, h_0) = \begin{cases} 
    h_0 & \text{if } s = [\ ] \\
    F(x_t, \hat{F}([x_1, ..., x_{t-1}], h_0)) & \text{if } s = [x_1, ..., x_t] 
\end{cases} \]

iterated application:
the state after seeing all but 1 input
Echo State Property (ESP)

A valid ESN should satisfy the “Echo State Property”

- (DEF) An ESN satisfies the ESP whenever:
  \[
  \forall s \in (\mathbb{R}^{N_x})^N, \forall h_0, z_0 \in \mathbb{R}^{N_H}:
  \| \hat{F}(s, h_0) - \hat{F}(s, z_0) \| \to 0,
  \text{ as } N \to \infty
  \]
  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 $W_{hh}$
- **Necessary Condition**, involving the control of the maximum eigenvalue in modulus of $W_{hh}$
- Active area of intense mathematical research: link it to the input
Sufficient Condition for the ESP

- **Theorem.** If the maximum singular value of $W_{hh}$ 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
  \[ \sigma_{\text{max}}(W_{hh}) = \|W_{hh}\|_2 < 1 \]
Necessary Condition for the ESP

- Theorem. If the spectral radius of $W_{hh}$ 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
  \[ \rho(W_{hh}) = \max(\text{abs}(\text{eig}(W_{hh}))) < 1 \]
Relation between the ESP conditions

- Known linear algebra fact: $\rho(W_{hh}) \leq \|W_{hh}\|_n$
- 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 $W_{hh}$:

1) Generate a random matrix $W$
   e.g. from a uniform distribution on [-1,1]
2) Scale by the desired spectral radius

$$W_{hh} \leftarrow \rho_{desired} \frac{W}{\rho(W)}$$

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

Initialization of $W_{xh}$:

1) Generate a random matrix $W_{xh}$, whose elements are drawn e.g. from a uniform distribution on $[-1,1]$

2) Scale by an input scaling parameter $\omega_{in}$
   - by range:$W_{xh} \leftarrow \omega_{in} W_{xh}$ (now weights are in $[-\omega_{in}, \omega_{in}]$
   - by norm:$W_{xh} \leftarrow \omega_{in} \frac{W_{xh}}{||W_{xh}||_2}$ (now the 2-norm of $W_{xh}$ equals $\omega_{in}$)

- 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
Stability in Practice

Stable dynamics
- orbits synchronize after a transient

Unstable dynamics
- orbits are sensitive to initial conditions

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

Given a training set \( \{(x_t, d_t)\}_{t=1}^{N} \)

1. Run the reservoir on the input sequence & collect the states
   \[ H = [h_1, \ldots, h_N] \]
2. Washout the initial transient. \( H \leftarrow H(N_w: N, :) \)
3. Collect the target data similarly into a matrix
   \[ D = [d_{N_w}, \ldots, d_N] \]
4. Solve the linear regression problem for the readout
   \[
   \min_{W_{hy}} \| HW_{hy} - D \|_2^2
   \]
Readout Training

- Typically training performed off-line in closed-form
  - pseudo-inverse \( W_{hy} = (H^T H)^{-1} H^T D \)
  - ridge-regression \( W_{hy} = (H^T H + \lambda I)^{-1} H^T D \)

- 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

Tikhonov regularizer
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 $\rho$
- input scaling $\omega_{in}$
- readout regularization $\lambda$
- ...


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 $\rho$ 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
Why does it work?

Suffix-based Markovian organization of the state space of contraction reservoir mappings even prior to learning.

Markovianity in practice

1st PC: last input symbol
2nd PC: next-to-last input symbol

\( \sigma = 0.3 \)
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

\[ h_t = \tanh(x_t W_{xh} + h_{t-1} W_{hh}) \]
\[ y_t = [h_t; x_t] W_{hy} \]

useful, e.g., when instantaneous (i.e., non-temporal) I/O transformations can be useful
Architectural Variants: Output feedback

Feedback connections from the readout

\[
y_t = h_t W_{hy}
\]

\[
h_t = \tanh(x_t W_{xh} + h_{t-1} W_{hh} + y_{t-1} W_{yh})
\]

Might impact on the stability of the reservoir dynamics.
Note: in this case the reservoir dynamics are adapted...

\[
h_t = \tanh(x_t W_{xh} + h_{t-1}(W_{hh} + W_{hy} W_{yh}))
\]
Leaky Integrator ESN (LI-ESN)

Use leaky integrators reservoir neurons:

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

leaking rate hyper-parameter

$$\alpha \in (0,1]$$
A few examples of RC applications
Reservoir Computing in practice

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.
Human Activity Monitoring

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
Robot localization in critical environments

Human Activity Recognition

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

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

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

Brugada Syndrome


ECG leads in input → processed by ESNs → Brugada Type 1 syndrome diagnosis ≈ 80% accurate
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.
**RC in Autonomous Vehicles**

- Automatic detection of physiological, emotional, cognitive state of the human \(\rightarrow\) **Human-centric personalization**
- **Good performance in human state monitoring + efficiency**

<table>
<thead>
<tr>
<th></th>
<th>WESAD</th>
<th>HHAR</th>
<th>PAMAP2</th>
<th>OPPORTUN.</th>
<th>ASCERTAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Std</td>
<td>Avg</td>
<td>Std</td>
<td>Avg</td>
</tr>
<tr>
<td>RNN</td>
<td>94.62</td>
<td>2.84</td>
<td>78.54</td>
<td>2.04</td>
<td>96.00</td>
</tr>
<tr>
<td>ESN</td>
<td>94.96</td>
<td>2.60</td>
<td>89.79</td>
<td>3.81</td>
<td>97.50</td>
</tr>
<tr>
<td>LSTM</td>
<td>95.48</td>
<td>1.17</td>
<td>92.71</td>
<td>2.72</td>
<td>96.50</td>
</tr>
<tr>
<td>GRU</td>
<td>98.13</td>
<td>1.16</td>
<td>98.54</td>
<td>0.83</td>
<td>98.50</td>
</tr>
</tbody>
</table>

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
Distributed, embeddable and federated learning

Driving-Style Personalization Based on Driver Stress


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

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

\[ f_{\text{gen}}(x) = f(ax + b) \]
\[ \Delta b = -\eta \left( -\frac{\mu}{\sigma^2} + \frac{y}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y) \right), \]
\[ \Delta a = \frac{\eta}{a} + \Delta bx. \]

**Edge of chaos**

**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

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

\[ \hat{W} = \begin{pmatrix} 0 & 0 & \cdots & 0 & \hat{w} \\ \hat{w} & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \hat{w} \end{pmatrix} \]

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)


Physical Reservoir Computing

Depth in RNNs

Deep Echo State Networks

Deep Echo State Networks

Deep reservoir = nested set of dynamical systems

\[
\begin{align*}
    h^{(L)}(t) &= \tanh(W^{(L)}h^{(L-1)}(t) + W_R^{(L)}h^{(L)}(t-1)) \\
    h^{(2)}(t) &= \tanh(W^{(2)}h^{(1)}(t) + W_R^{(2)}h^{(2)}(t-1)) \\
    h^{(1)}(t) &= \tanh(W^{(1)}x(t) + W_R^{(1)}h^{(1)}(t-1))
\end{align*}
\]

Multiple time-scales

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


How many layers?

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].
Richer dynamics: stability regime

**Fig. 4.** Averaged values of $\lambda_{\text{max}}$ 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 $scale_{II} = 0.5$. 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.

Neural networks for graphs
Vertex-wise graph encoding

\[ h(v) = \tanh (W_x(v) + \sum_{v' \in N(v)} W_R h(v')) \]

- time-step $\rightarrow$ vertex
- previous time step $\rightarrow$ neighborhood

embedding (state)
input features
embeddings of neighbors
Reservoir Computing for graphs

\[ H = \tanh \left( WX + W_R H \tilde{A} \right) \]

- **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., \( \rho(W_R) < 1 \)
Reservoir Layer for graphs

\[ H[t] = \tanh(WX + W_R H[t - 1]\tilde{\Delta}) \]

- 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

Fast and Deep Graph Neural Networks (FDGNN)

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

\[
H^{(L-1)*}
\]

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

\[
H^{(1)*}
\]

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

### It’s accurate

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

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

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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUTAG</td>
<td>0.56&quot;±0.33</td>
<td>0.06&quot;±0.04</td>
</tr>
<tr>
<td>PTC</td>
<td>0.16&quot;±0.03</td>
<td>0.02&quot;±0.00</td>
</tr>
<tr>
<td>COX2</td>
<td>1.36&quot;±0.42</td>
<td>0.15&quot;±0.05</td>
</tr>
<tr>
<td>PROTEINS</td>
<td>2.16&quot;±0.47</td>
<td>0.24&quot;±0.04</td>
</tr>
<tr>
<td>NCI1</td>
<td>2.00&quot;±0.45</td>
<td>13.36&quot;±3.02</td>
</tr>
<tr>
<td>IMDB-b</td>
<td>7.46&quot;±3.14</td>
<td>0.83&quot;±0.35</td>
</tr>
<tr>
<td>IMDB-m</td>
<td>8.68&quot;±1.73</td>
<td>0.98&quot;±0.22</td>
</tr>
<tr>
<td>REDDIT</td>
<td>2.47&quot;±0.01</td>
<td>16.49&quot;±0.28</td>
</tr>
<tr>
<td>COLLAB</td>
<td>22.86&quot;±4.70</td>
<td>2.54&quot;±0.52</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>FDGNN</th>
<th>GNN</th>
<th>GIN</th>
<th>WL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.56&quot;±0.33</td>
<td>202.28&quot;±166.87</td>
<td>499.24&quot;±2.25</td>
<td>1.16&quot;±0.03</td>
</tr>
</tbody>
</table>
Reservoir Computing by discretizing ODEs

- 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

\[ h_t = h_{t-1} + \epsilon \tanh(W_x x_t + (W_h - W_h^T - \gamma I) h_{t-1} + b) \]

- step size
- diffusion coefficient
The input signal is preserved without exploding nor dying.

High accuracy vs state-of-the-art fully trainable models & ESNs
Extremely more efficient (up to 100x) than fully trainable models
Deep Randomized Neural Networks


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

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

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

https://www.youtube.com/watch?v=1K7oJCTzKQ
Deep Reservoir Computing - TensorFlow Library

TF implementation of the functionalities of Deep Reservoir Computing. Import the content of DeepRC.py and use the SimpleDeepReservoirLayer as the dynamical component of your deep network for time-series processing. In the code you can find an example on how to use SimpleDeepReservoirLayer to define a SimpleDeepESNClassifier.

If you use this code in your work, please cite the following paper, in which the concept of Deep Reservoir Computing has been introduced.


EulerStateNetworks

This repository contains the TensorFlow 2.0 / Keras implementation of Euler State Networks (EuSN), as described in the paper C. Gallicchio, "Euler State Networks", Submitted to Journal (2022)  
[https://arxiv.org/abs/2203.09382]
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.
IEEE Task Force on Randomization-based Neural Networks and Learning Systems

https://sites.google.com/view/randnn-tf/

Promote the research and applications of deep rand. neural networks and learning systems, to demonstrate the competitive performance of randomization-based algorithms in diverse scenarios, to educate the research community about the randomization-based learning methods and their relationships,
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
Reservoir Computing

Claudio Gallicchio

gallicch@di.unipi.it
https://www.linkedin.com/in/claudio-gallicchio-05a47038/
https://twitter.com/claudiogallicc1