Pervasive AI

Tutorial of the 37th AAAI Conference on Artificial Intelligence
February 7, 2023 – Washington DC, USA

Davide Bacciu, Antonio Carta, Patrizio Dazzi, Claudio Gallicchio
University of Pisa, Italy

http://pai.di.unipi.it/aaai-2023-tutorial-on-pervasive-ai/

Advanced Models

Davide Bacciu, Antonio Carta, Patrizio Dazzi, **Claudio Gallicchio**
University of Pisa, Italy

Outline

- Deep randomized NNs
- Reservoir Computing architectures
- Training reservoirs
- Neuromorphic computing & training beyond backpropagation

Deep Randomized Neural Networks

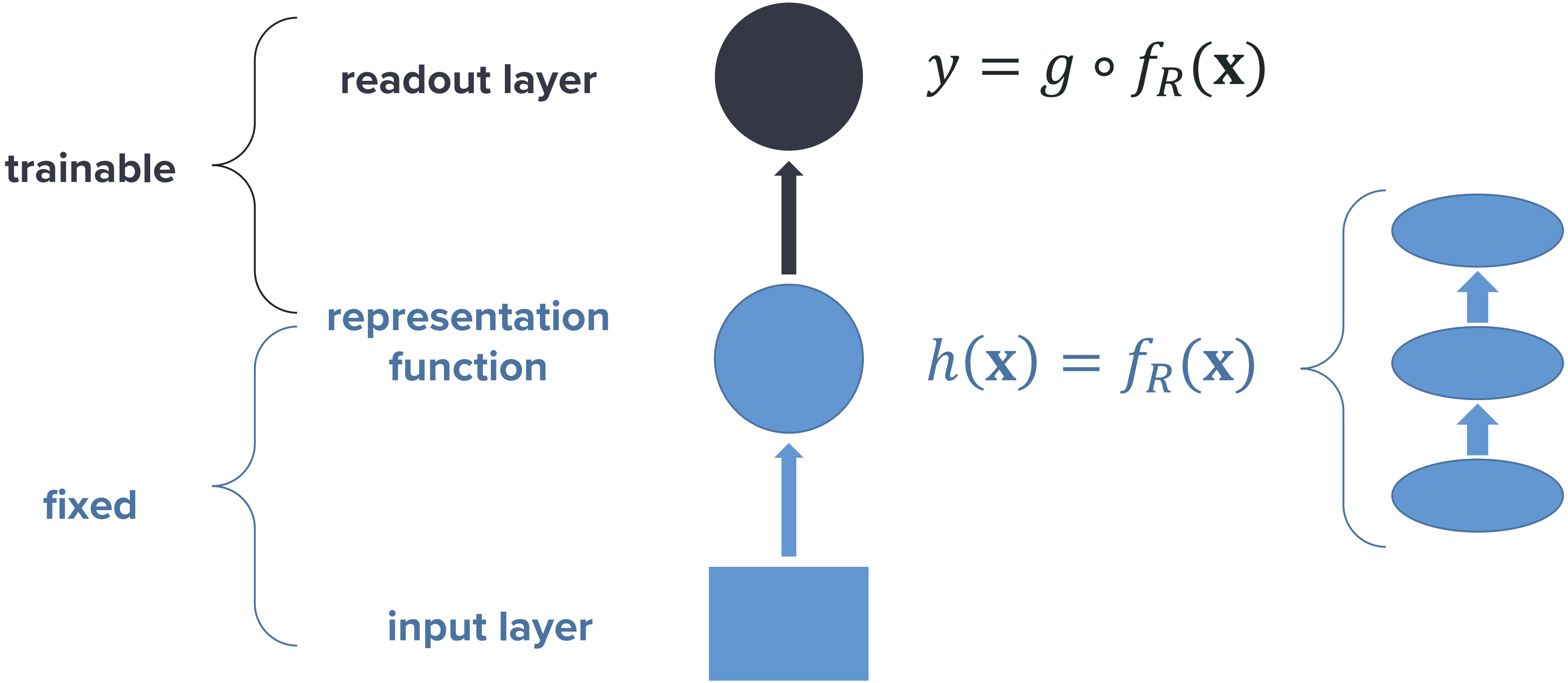
The Philosophy

"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.

Deep Randomized Architectures



Deep image prior

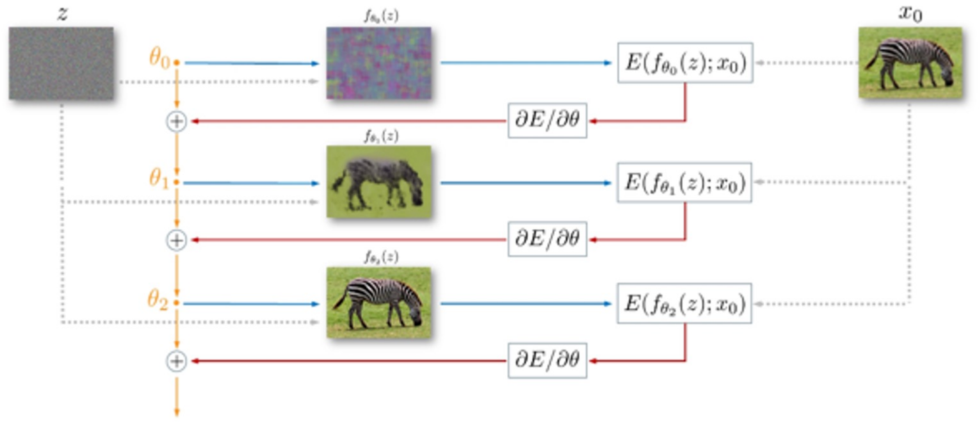


Fig. 2: **Image restoration using the deep image prior.** Starting from a random weights θ_0, we iteratively update them in order to minimize the data term eq. (2). At every iteration the weights θ are mapped to an image $x = f_\theta(z)$, where z is a fixed tensor and the mapping f is a neural network with parameters θ. The image x is used to compute the task-dependent loss $E(x, x_0)$. The gradient of the loss w.r.t. the weights θ is then computed and used to update the parameters.

a randomly initialized CNN contains enough structural information to act as an efficient prior in many image processing problems

Ulyanov, D., Vedaldi, A. and Lempitsky, V., 2018. Deep image prior. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 9446-9454).

Reservoir Transformers

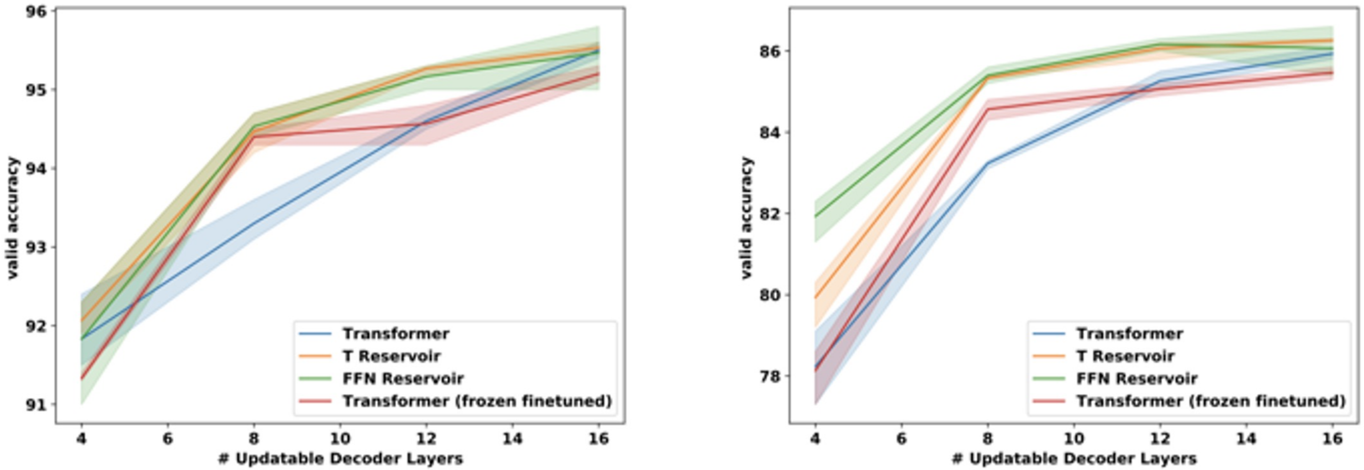


Figure 4: Downstream RoBERTa performance on SST-2 (left) and MultiNLI-matched (right).

Shen, S., Baevski, A., Morcos, A.S., Keutzer, K., Auli, M. and Kiela, D., 2020. Reservoir Transformer. arXiv preprint arXiv:2012.15045.

Performers

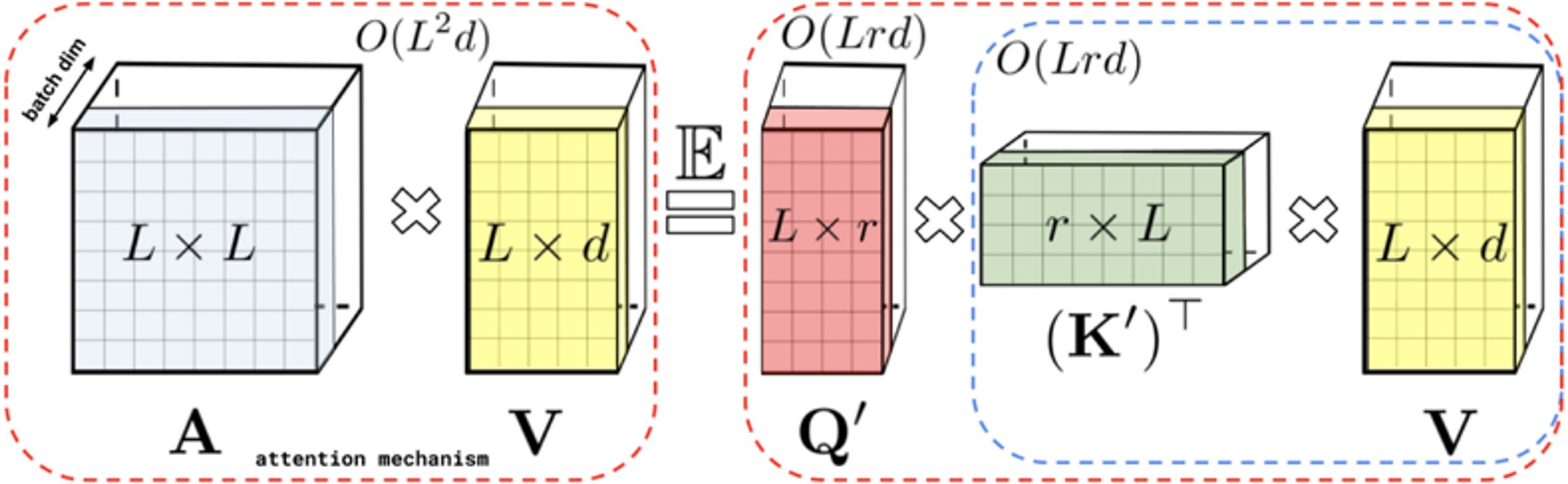
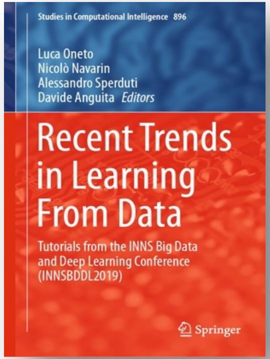


Figure 1: Approximation of the regular attention mechanism **AV** (before $\mathbf{D}^{-1}$-renormalization) via (random) feature maps. Dashed-blocks indicate order of computation with corresponding time complexities attached.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L. and Belanger, D., 2020. Rethinking attention with performers. arXiv preprint arXiv:2009.14794.

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

A deeper dive into Reservoir Computing

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.

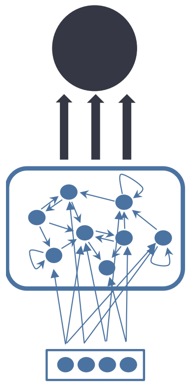
Vanilla Recurrent neural nets

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

trainable

Echo State Networks

$$\mathbf{h}(t) = \tanh(\rho \mathbf{W_h} \mathbf{h}(t-1) + \omega_x \mathbf{W_x} \mathbf{x}(t) + \omega_b \mathbf{b})$$

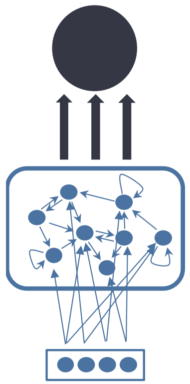


Yildiz, Izzet B., Herbert Jaeger, and Stefan J. Kiebel. "Re-visiting the echo state property." Neural networks 35 (2012): 1-9.

Echo State Networks

fixed weights

$$\mathbf{h}(t) = \tanh(\rho \mathbf{W_h} \mathbf{h}(t-1) + \omega_x \mathbf{W_x} \mathbf{x}(t) + \omega_b \mathbf{b})$$



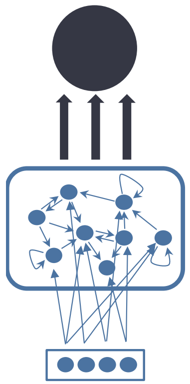
Yildiz, Izzet B., Herbert Jaeger, and Stefan J. Kiebel. "Re-visiting the echo state property." Neural networks 35 (2012): 1-9.

Echo State Networks

fixed weights

$$\mathbf{h}(t) = \tanh(\rho \mathbf{W_h} \mathbf{h}(t-1) + \omega_x \mathbf{W_x} \mathbf{x}(t) + \omega_b \mathbf{b})$$

scaling hyper-parameters



How to scale the weight matrices?

Fulfill the "echo state" property

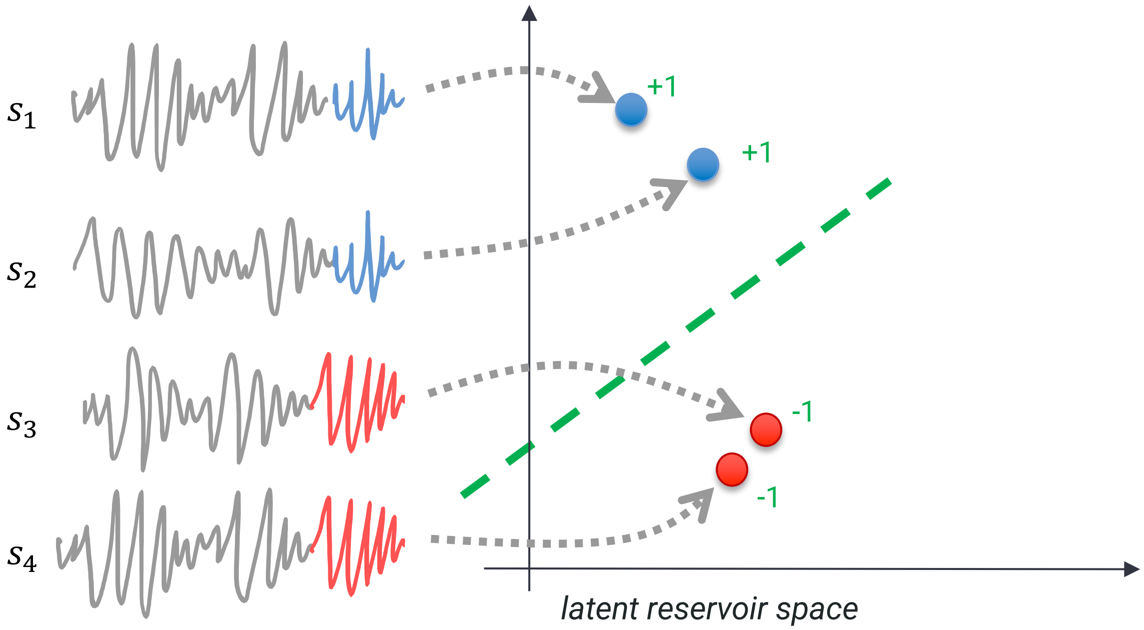
- global asymptotic Lyapunov stability condition
- spectral radius $\rho < 1$

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

Gallicchio, Claudio, and Alessio Micheli, Neural Networks 24.5 (2011): 440-456.

Why does it work?

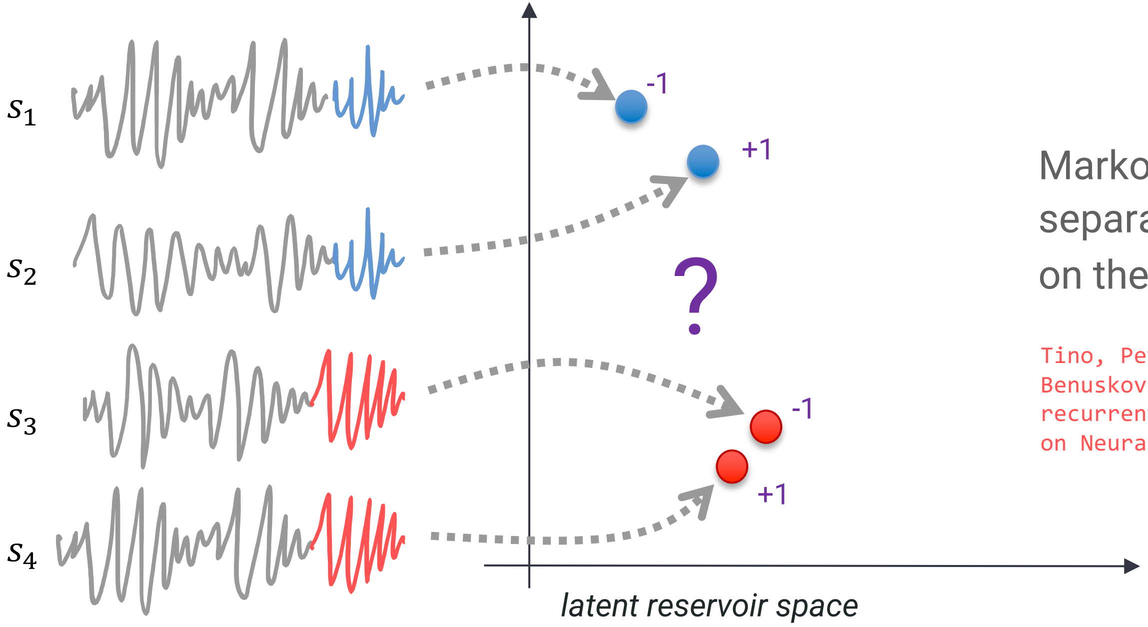
Because of the architectural bias of contracting RNNs



Gallicchio, Claudio, and Alessio Micheli, Neural Networks 24.5 (2011): 440-456.

Why does it work?

Because of the architectural bias of contracting RNNs



Markovian bias of RNNs separate input sequences based on the suffix even prior to learning

Tino, Peter, Michal Cernansky, and Lubica Benuskova. "Markovian architectural bias of recurrent neural networks." IEEE Transactions on Neural Networks 15.1 (2004): 6-15.

Leaky Integrator – Echo State Network

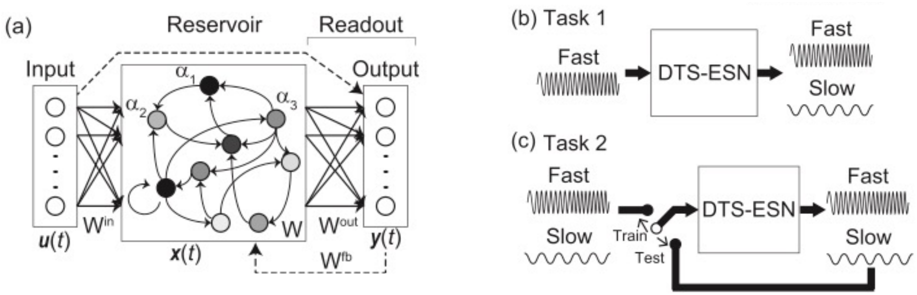
Use leaky integrators reservoir neurons:

$$\mathbf{h}(t) = (1-\alpha)\mathbf{h}(t-1) + \alpha \tanh(\mathbf{W}_x \mathbf{x}(t) + \mathbf{W}_\mathbf{h} \mathbf{h}(t-1) + \mathbf{b})$$

leaking rate hyper-parameter

$\alpha \in (0,1]$

- smaller values for reservoirs that react more slowly to the input

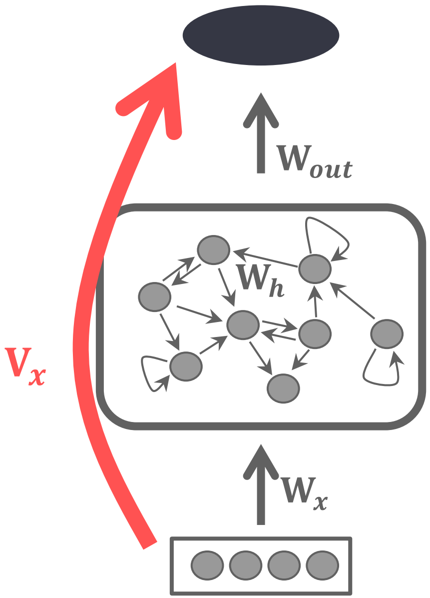


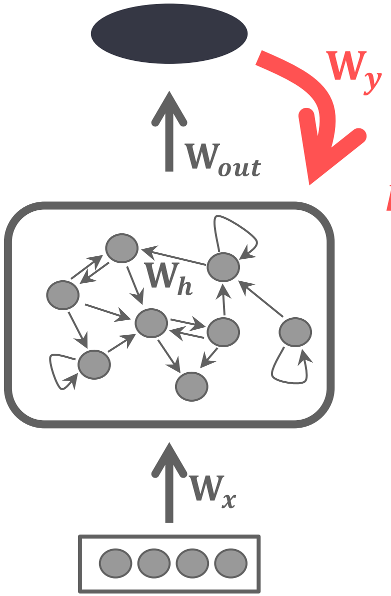
Jaeger, Herbert, et al. "Optimization and applications of echo state networks with leaky-integrator neurons." *Neural networks*20.3 (2007): 335-352.

Tanaka, Gouhei, et al. "Reservoir computing with diverse timescales for prediction of multiscale dynamics." Physical Review Research 4.3 (2022): L032014.

Input-output and output-feedback connections

$$y(t) = W_x h(t) + V_x x(t)$$





$$h(t) = \sigma(W_x x(t) + W_h h(t-1) + W_y y(t-1))$$

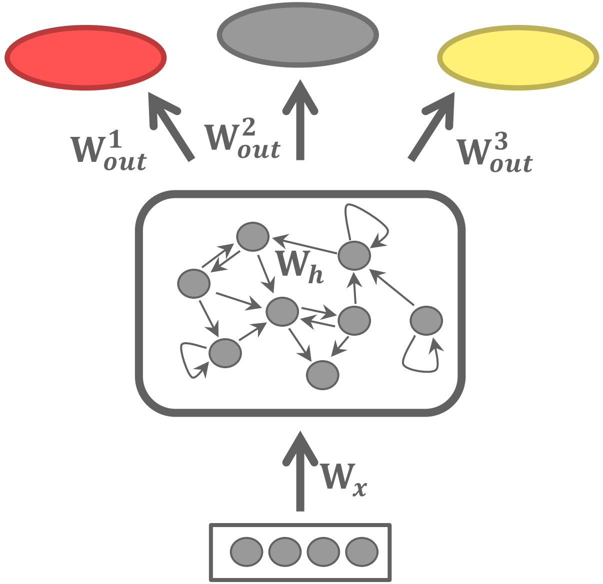
$$h(t) = \sigma(W_x x(t) + (W_h + W_y W_{out}) h(t-1))$$

trained

FORCE: W_{out} connections trained by RLS

Sussillo D, Abbott LF. Generating coherent patterns of activity from chaotic neural networks. Neuron. 2009; 63(4): 544–557

Multiple readouts



- The reservoir is operating in a purely unsupervised mode
- If multiple tasks involve the same input time-series (but different targets) the same reservoir could be used

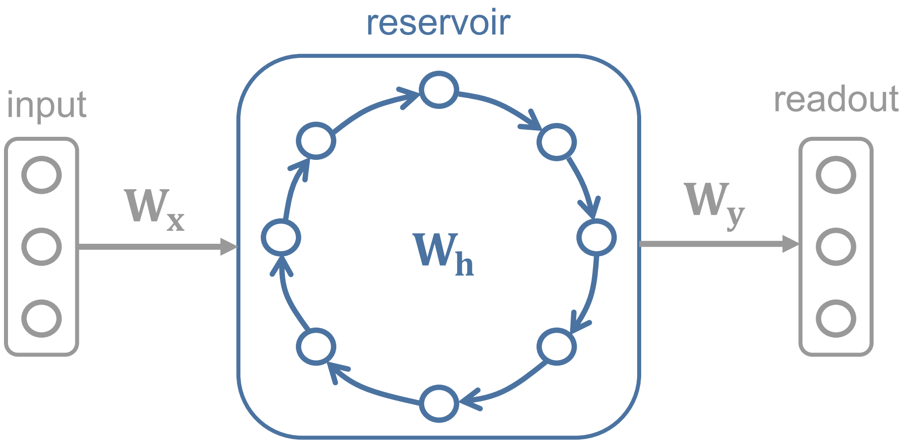
Good reservoirs

Can we find a better reservoir than just a random one?

- High entropy of neurons activations
  - diversify the temporal response of the reservoir neurons
- Long short-term memory capacity
  - latch input information effectively
- Close to the edge of 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

Cycle reservoirs

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

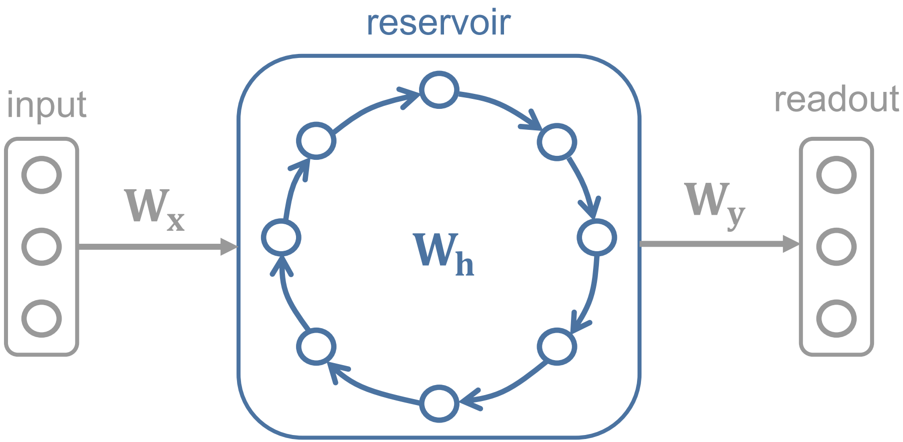


$$\mathbf{W_h} = \begin{pmatrix} 0 & 0 & \cdots & 0 & \hat{w} \\ \hat{w} & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & \hat{w} & 0 \end{pmatrix}$$

- The architecture is further simplified: $O(1)$ rather than $O(N^2)$
- Matrix multiplications simplify to shift operations

Cycle reservoirs

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



$$\mathbf{W_h} = \begin{pmatrix} 0 & 0 & \dots & 0 & \hat{w} \\ \hat{w} & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & \hat{w} & 0 \end{pmatrix}$$

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

$$J(t) = \mathbf{D}(t)\,\mathbf{P}$$

nice eigenstructure

Deep reservoirs

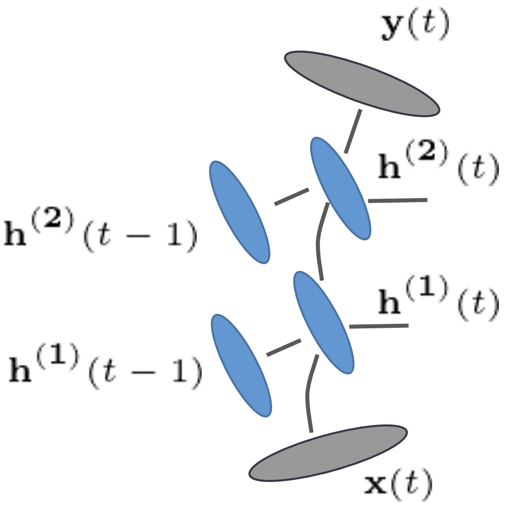
Reservoir = set of nested non-linear dynamical systems

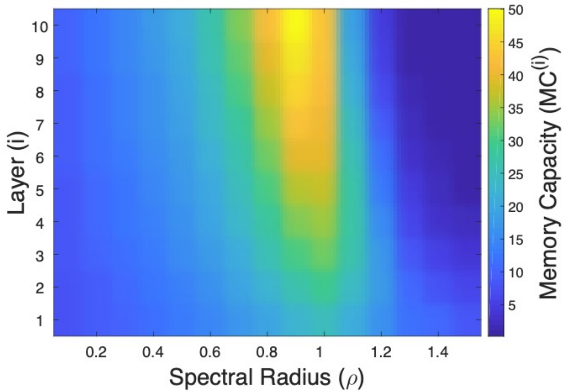
$$\mathbf{h}^{(i)}(t) = \tanh(\mathbf{W}_h^{(i)}\mathbf{h}^{(i)}(t-1) + \mathbf{W}_x^{(i)}\mathbf{h}^{(i-1)}(t) + \mathbf{b}^{(i)})$$

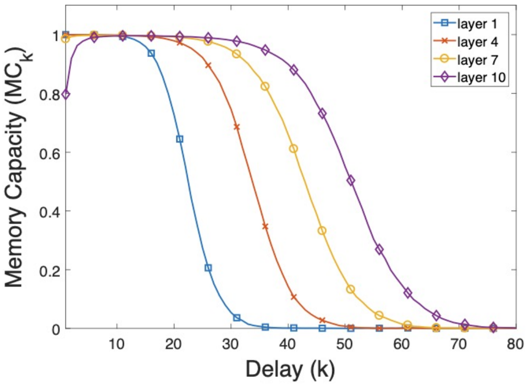
driving input

...

$$\mathbf{h}^{(1)}(t) = \tanh(\mathbf{W}_h^{(1)}\mathbf{h}^{(1)}(t-1) + \mathbf{W}_x^{(1)}\mathbf{x}(t) + \mathbf{b}^{(1)})$$



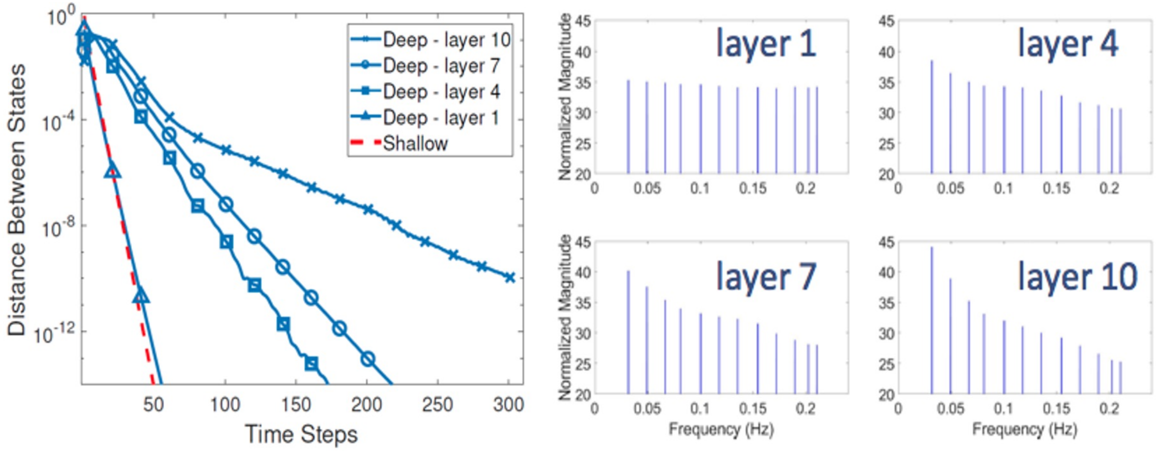




Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli. "Deep reservoir computing: A critical experimental analysis." Neurocomputing 268 (2017): 87-99

Architectural bias of depth in Recurrent Neural Nets

- Multiple time-scales
- Multiple frequencies
- Develop richer dynamics even without training of the recurrent connections

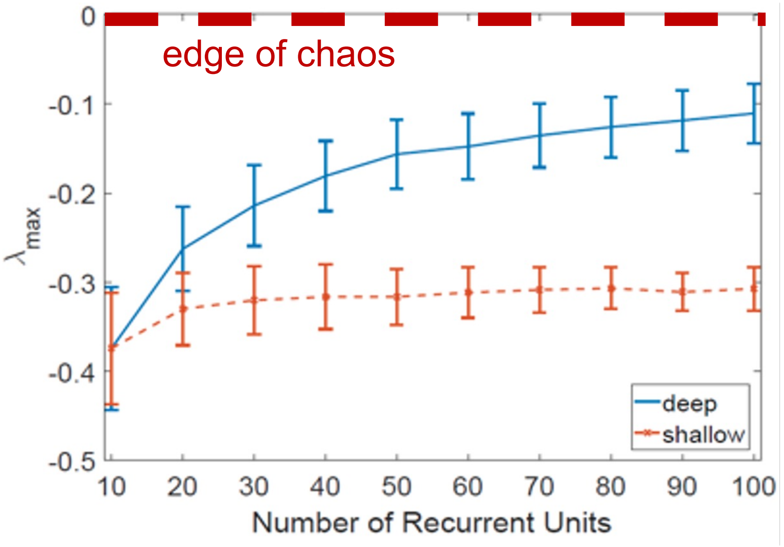


Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli. "Deep reservoir computing: A critical experimental analysis." Neurocomputing 268 (2017): 87-99

Gallicchio, C., Micheli, A. and Pedrelli, L., 2018. Design of deep echo state networks. Neural Networks, 108, pp.33-47.

Architectural bias of depth in Recurrent Neural Nets

- Multiple time-scales
- Multiple frequencies
- Develop richer dynamics even without training of the recurrent connections



local Lyapunov exponents

$$\lambda_{max} = \max_k \frac{1}{T} \sum_{t=1,\dots,T} \ln |\lambda_k(J(t))|$$

state dynamics are closer to the edge of chaos

$$\lambda_{max} \approx 0$$

Gallicchio, Claudio, and Alessio Micheli. "Deep reservoir computing." Reservoir Computing (2021): 77-95.

Gallicchio, Claudio. "Euler state networks." arXiv preprint arXiv:2203.09382 (2022).

Euler reservoirs

non-dissipative stable dynamics by design

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

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

$$\mathbf{h}(t) = \mathbf{h}(t-1) + \underbrace{\varepsilon}_{\text{step size}} \tanh\left(\boxed{\mathbf{W}_\mathbf{x}}\, \mathbf{x}(t) + \left(\boxed{\mathbf{W}_h} - \boxed{\mathbf{W}_h^T} - \underbrace{\gamma}_{\text{diffusion}} \mathbf{I}\right)\mathbf{h}(t-1) + \boxed{\mathbf{b}}\right)$$

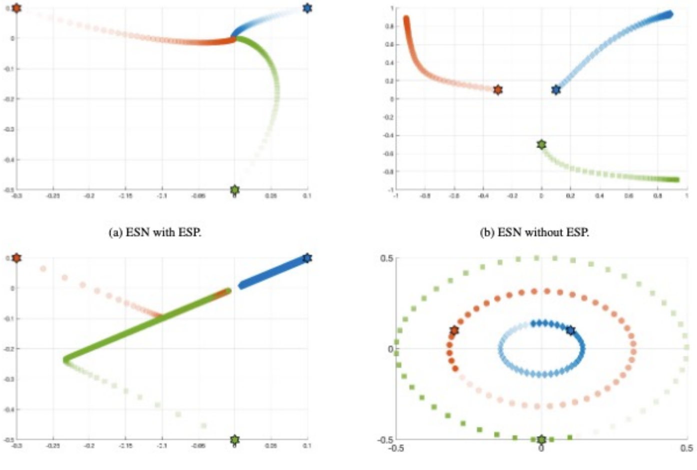
untrained

dynamics are arbitrarily close to the edge of chaos

Euler reservoirs

Gallicchio, Claudio. "Euler state networks." arXiv preprint arXiv:2203.09382 (2022).

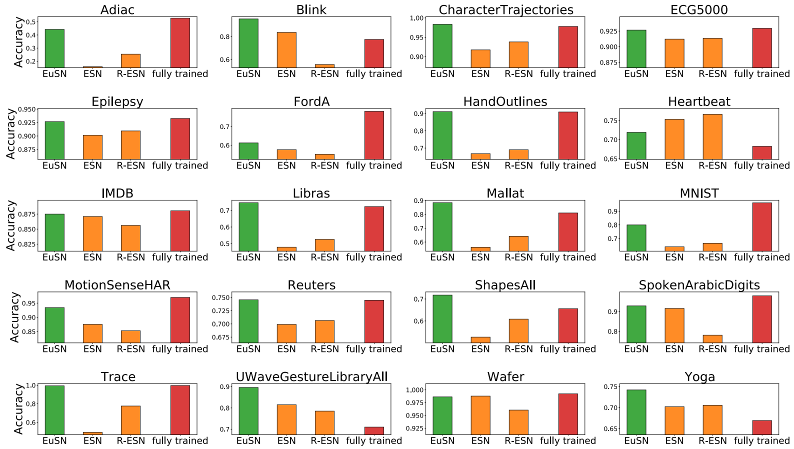
The input signal is preserved without exploding nor dying



(a) ESN with ESP. (b) ESN without ESP.

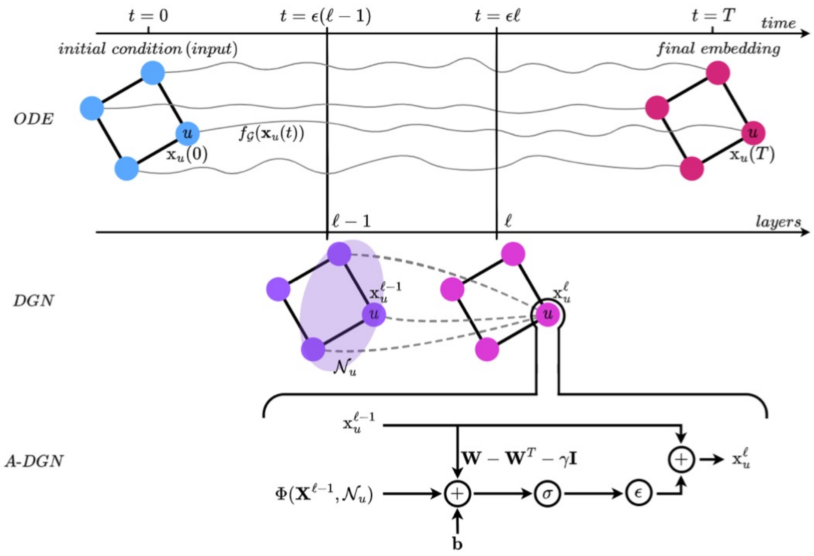
(c) R-ESN. (d) EuSN.

bridge the accuracy gap with fully trainable models



up to x100 times more efficient than fully trainable RNNs

Antisymmetric Deep Graph Networks



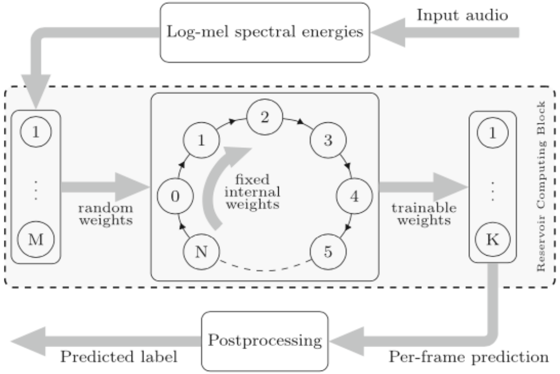
$$\mathbf{x}_u^{\ell} = \mathbf{x}_u^{\ell-1} + \epsilon\sigma\left((\mathbf{W}-\mathbf{W}^T-\gamma\mathbf{I})\mathbf{x}_u^{\ell-1} + \Phi(\mathbf{X}^{\ell-1},\mathcal{N}_u) + \mathbf{b}\right)$$

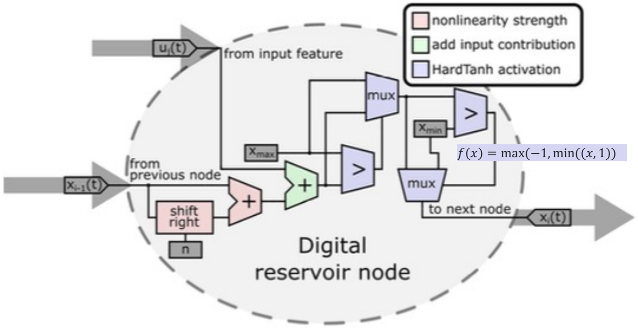
- Long range information between nodes
- No gradient vanishing/exploding
- Sensible performance improvement in applications

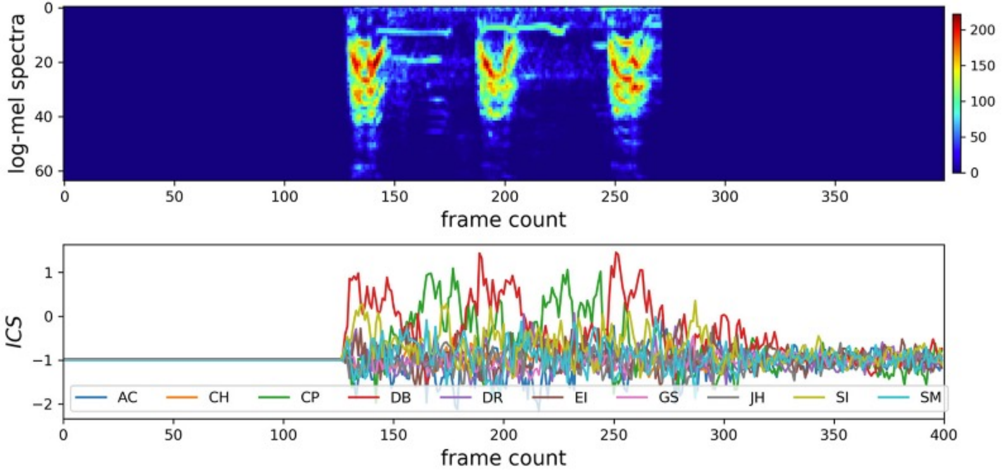
	Diameter	**SSSP**	**Eccentricity**
GCN	0.7424±0.0466	$0.9499 \pm 9.18 \cdot 10^{-5}$	0.8468±0.0028
GAT	0.8221±0.0752	0.6951±0.1499	0.7909±0.0222
GraphSAGE	0.8645±0.0401	0.2863±0.1843	0.7863±0.0207
GIN	0.6131±0.0990	-0.5408±0.4193	0.9504±0.0007
Our	**-0.5188±0.1812**	**-3.2417±0.0751**	**0.4296±0.1003**
Our(GCN)	0.2646±0402	-1.3659±0.0702	0.7177±0.0345

Gravina, Alessio, Davide Bacciu, and Claudio Gallicchio. "Anti-Symmetric DGN: a stable architecture for Deep Graph Networks." ICLR 2023

Hardware-optimized RC

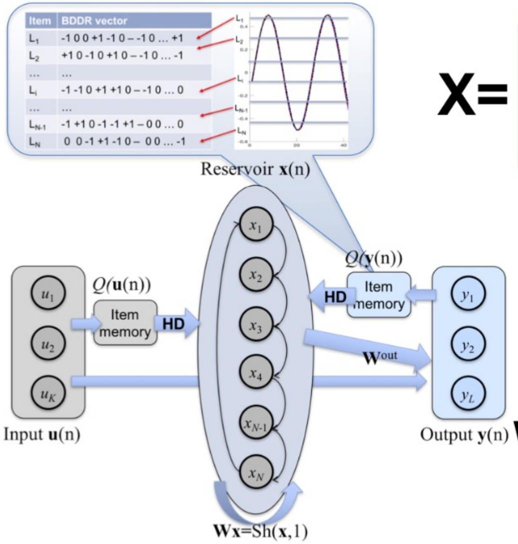






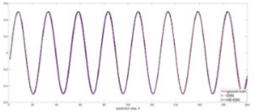
Morán, Alejandro, et al. "Hardware-optimized reservoir computing system for edge intelligence applications." *Cognitive Computation* (2021): 1-9.

Integer Echo State Networks



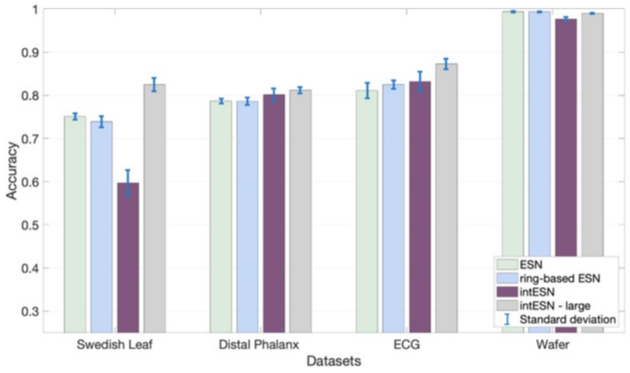
$$X=\begin{pmatrix} 3 & 3 & 3 & -2 & -1 & -2 & -1 \\ 3 & 3 & 2 & 3 & -3 & -2 & -1 \\ 0 & 2 & 3 & 3 & 3 & -2 & -3 \\ -3 & -1 & 1 & 2 & 3 & 2 & -1 \\ 3 & -2 & -2 & 0 & 1 & 2 & 1 \\ 3 & 3 & -1 & -1 & 1 & 2 & 3 \end{pmatrix}$$

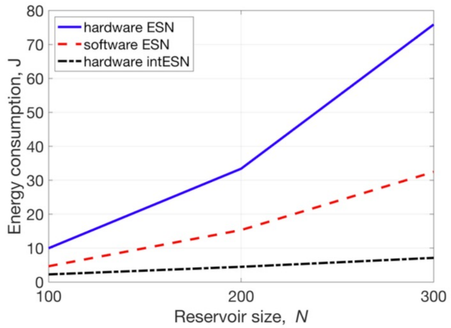
Y=



$$W=\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

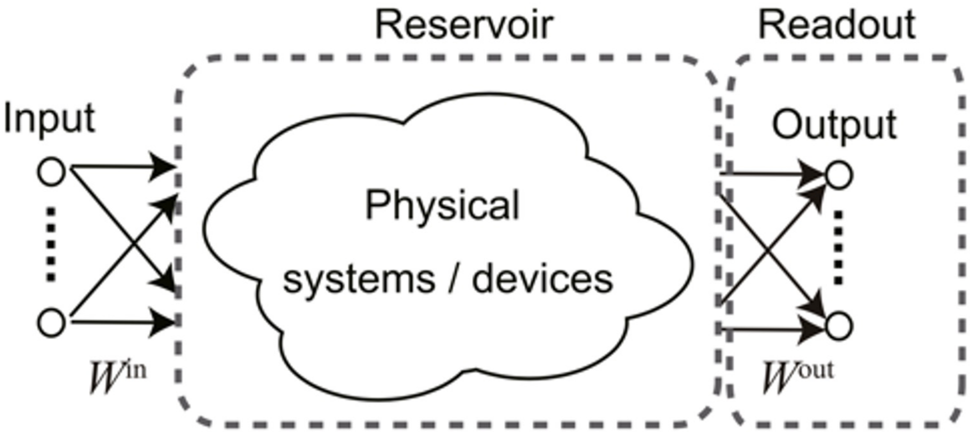
$$x(n) = f_k(Sh(x(n-1),1) + u^{HD}(n) + y^{HD}(n-1))$$

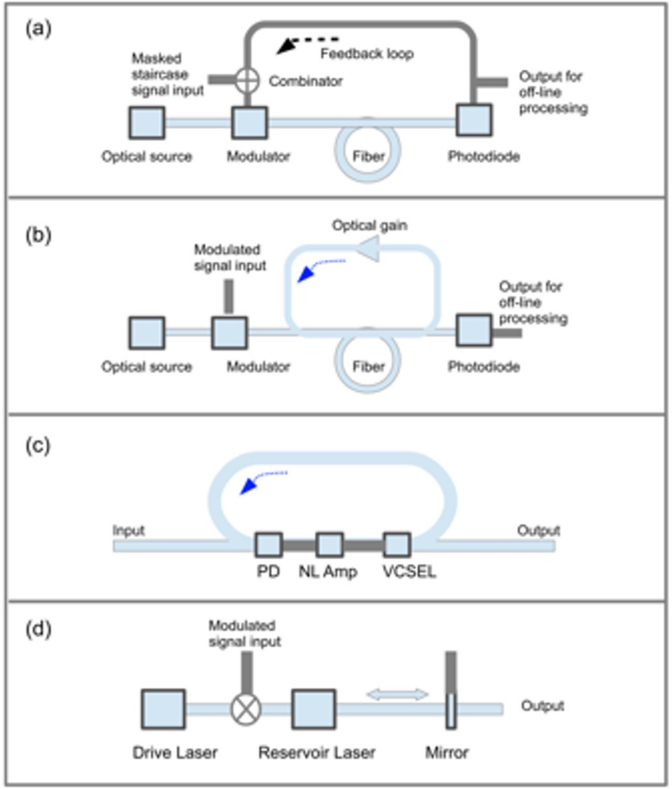




Kleyko, Denis, et al. "Integer echo state networks: efficient reservoir computing for digital hardware." IEEE Transactions on Neural Networks and Learning Systems 33.4 (2020): 1688-1701.

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.

Training Reservoirs

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.

Intrinsic Plasticity

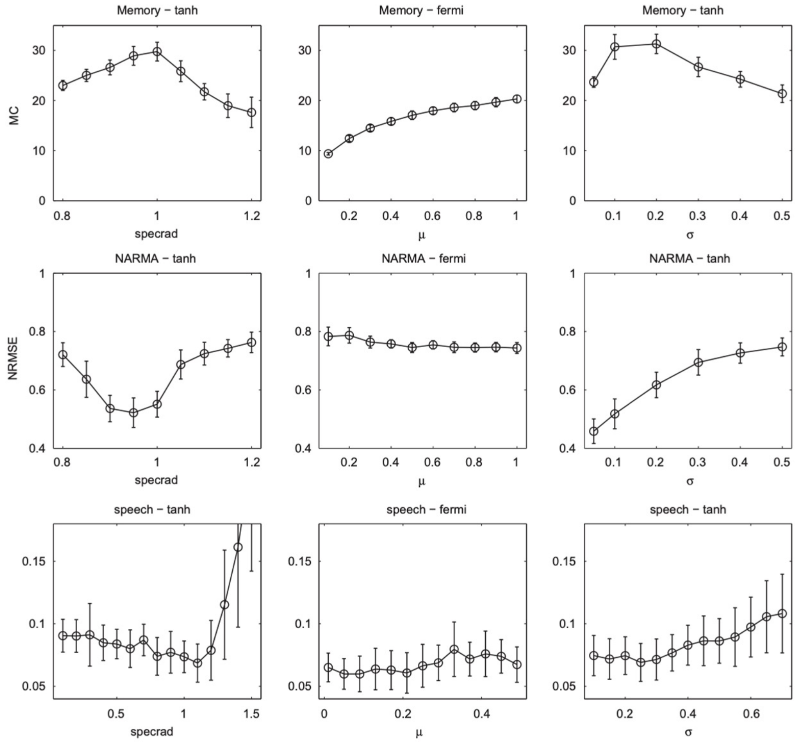


Fig. 4. Results for all three benchmarks for tanh with spectral radius ranging (left column), exponential IP for fermi nodes (middle column), and Gaussian IP for tanh nodes (right column).

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

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

gain

bias

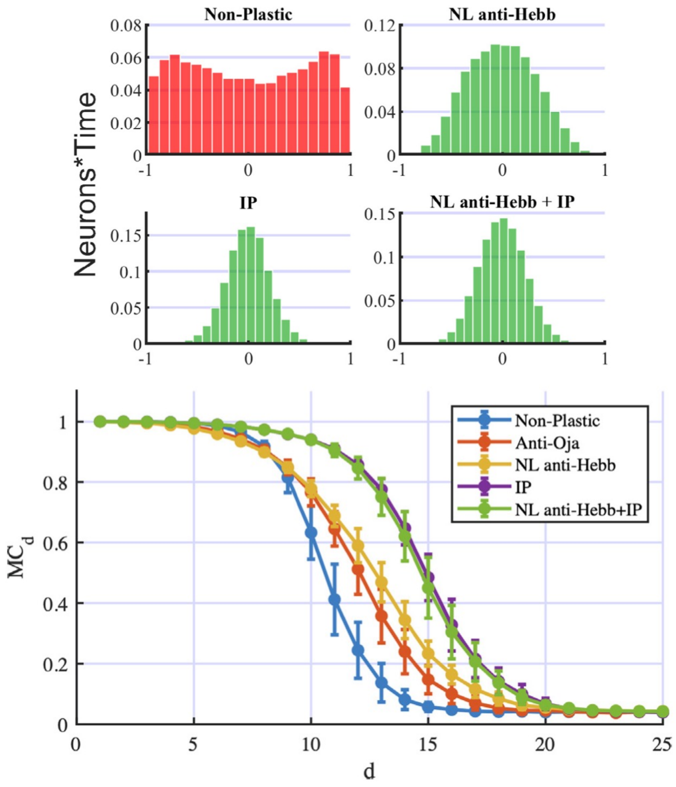
Kullback–Leibler divergence minimization

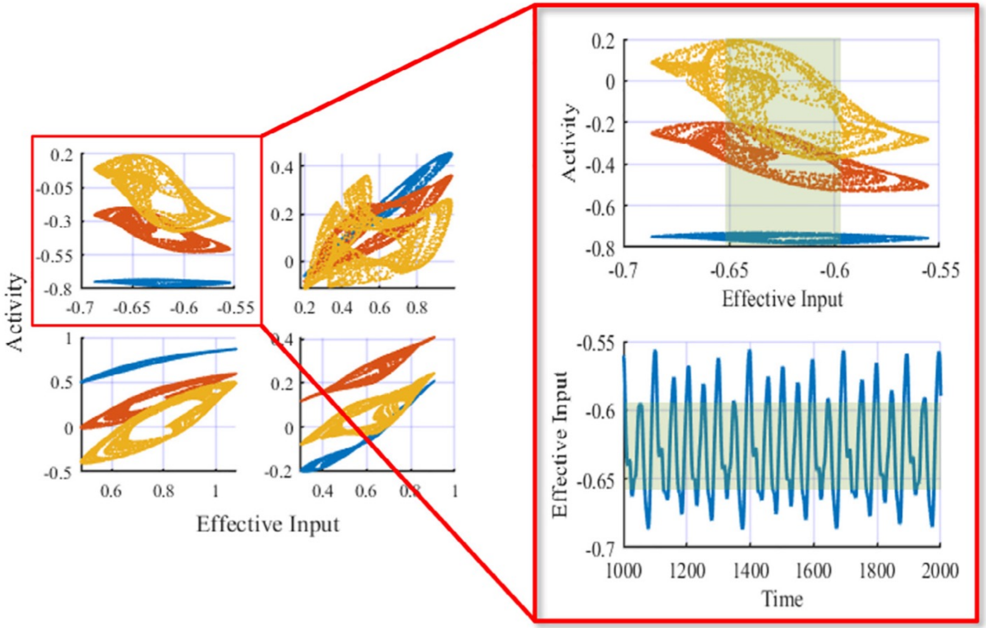
$$\Delta b = -\eta\left(-\frac{\boxed{\mu}}{\boxed{\sigma^2}} + \frac{y}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y)\right),$$

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

hyperparameters

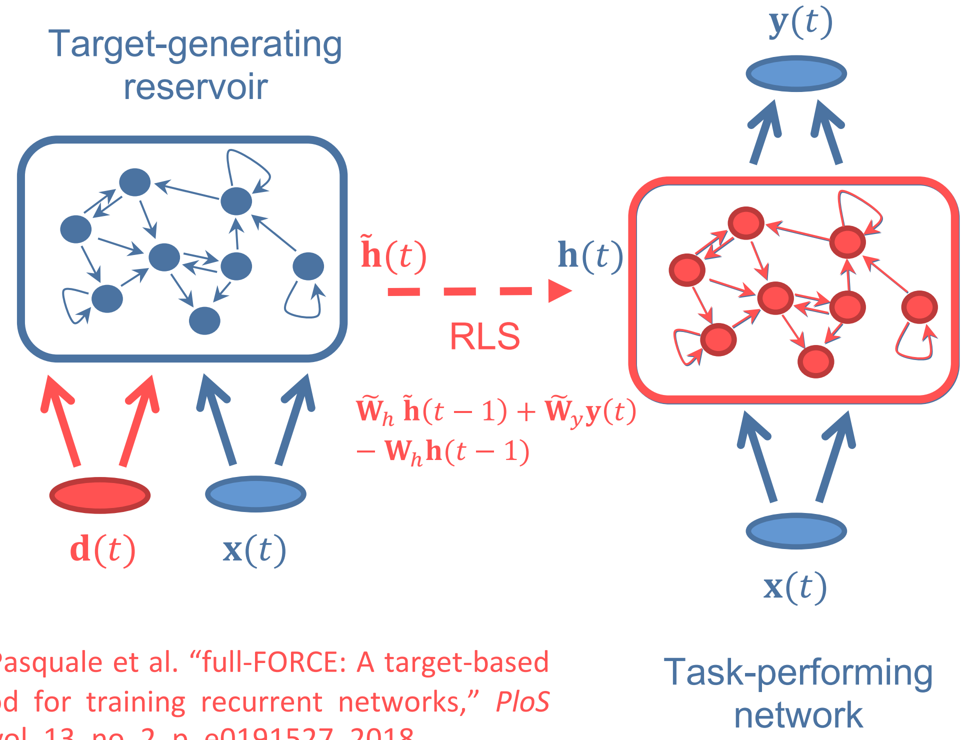
Plasticity improves input separation

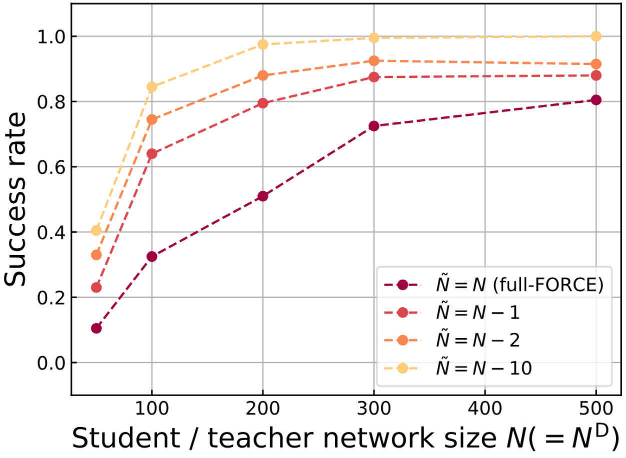




G.B. Morales, C. Mirasso, M.C. Soriano, 2021. Unveiling the role of plasticity rules in reservoir computing. Neurocomputing.

Full-FORCE





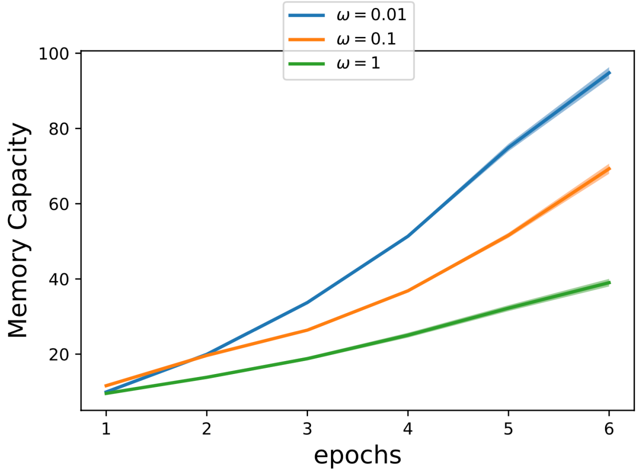
B. DePasquale et al. "full-FORCE: A target-based method for training recurrent networks," *PloS ONE*, vol. 13, no. 2, p. e0191527, 2018.

H. Tamura, G. Tanaka. "partial-FORCE: a fast and robust online training method for recurrent neural networks". IJCNN 2021

Phase Transition Adaptation

Train echo state networks to the edge of chaos

$$\mathbf{h}(t) = \tanh(\boldsymbol{a} \odot (\mathbf{W_h}\mathbf{h}(t-1) + \mathbf{W_x}\mathbf{x}(t)) + \mathbf{b})$$

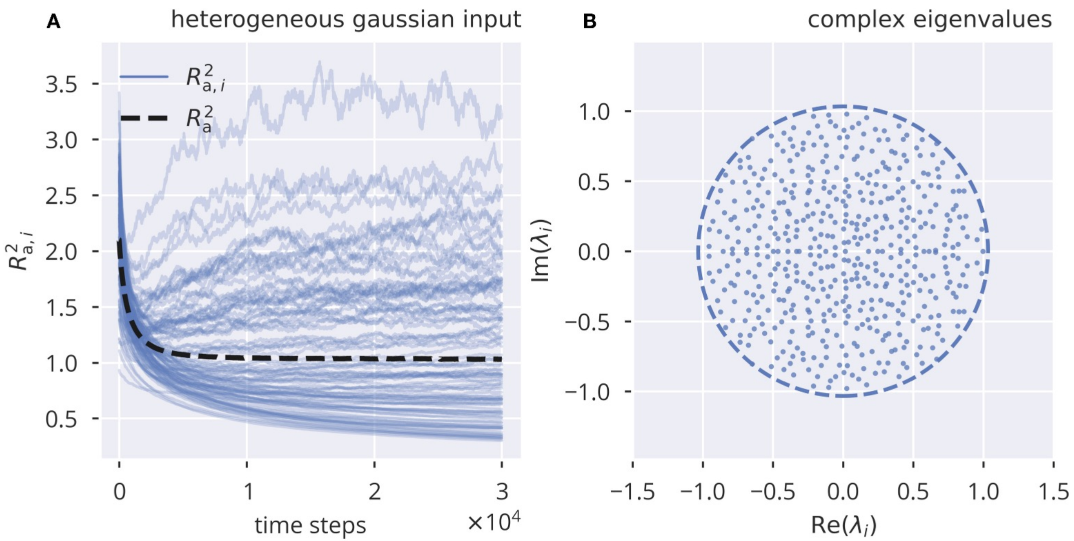


- local learning of gain and bias
- cycle reservoirs: the eigenstructure is easily adapted

C. Gallicchio, A. Micheli, L. Silvestri. “Phase Transition Adaptation”. IJCNN 2021

Homeostatic regulation

autonomous adaptation of the spectral radius during external input stimulation



F. Schubert, C. Gros. "Local homeostatic regulation of the spectral radius of echo-state networks". Frontiers in Computational Neuroscience, 2021

Federated Reservoir Computing

Readout training: online

- Least Mean Squares (LMS) is not practically used due to high eigenvalue spread of $\mathbf{H}\mathbf{H}^T$
- Recursive Least Squares (RLS) algorithm

1. Computation of the gain vector:

$$\mathbf{u}(n) = \mathbf{\Psi}_\lambda^{-1}(n-1)\mathbf{x}(n)$$

$$\mathbf{k}(n) = \frac{1}{\lambda + \mathbf{x}^{\mathrm{T}}(n)\mathbf{u}(n)}\mathbf{u}(n)$$

2. Filtering:

$$\hat{y}_{n-1}(n) = \hat{\mathbf{w}}^{\mathrm{T}}(n-1)\mathbf{x}(n)$$

3. Error estimation:

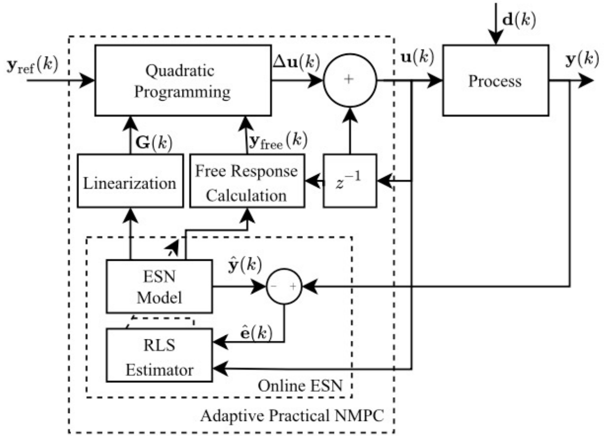
$$\hat{e}_{n-1}(n) = d(n) - \hat{y}_{n-1}(n)$$

4. Tap-weight vector adaptation:

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\hat{e}_{n-1}(n)$$

5. $\mathbf{\Psi}_\lambda^{-1}(n)$ update:

$$\mathbf{\Psi}_\lambda^{-1}(n) = \mathrm{Tri}\,\{\lambda^{-1}(\mathbf{\Psi}_\lambda^{-1}(n-1) - \mathbf{k}(n)\mathbf{u}^{\mathrm{T}}(n))\}$$



Farhang-Boroujeny, Behrouz. Adaptive filters: theory and applications. John Wiley & Sons, 2013.

Schwedersky et al."Adaptive practical nonlinear model predictive control for echo state network models." IEEE Transactions on Neural Networks and Learning Systems 33.6 (2021): 2605-2614.

Readout training: offline

- Closed form solution

$$\mathbf{H} = \begin{bmatrix} | & & | \\ \mathbf{h}(1) & \dots & \mathbf{h}(\mathrm{T}) \\ | & & | \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} | & & | \\ \mathbf{d}(1) & \dots & \mathbf{d}(\mathrm{T}) \\ | & & | \end{bmatrix}$$

states targets

  - Moore-Penrose pseudo-inversion

$$\mathbf{W}_{out} = \mathbf{D}\,\mathbf{H}^{+} = \mathbf{D}\,\mathbf{H}^{T}(\mathbf{H}\mathbf{H}^{T})^{-1}$$

  - Ridge-regression

$$\mathbf{W}_{out} = \mathbf{D}\,\mathbf{H}^{T}(\mathbf{H}\mathbf{H}^{T} + \lambda\,\mathbf{I})^{-1}$$

    - λ is a Tikhonov regularization coefficient

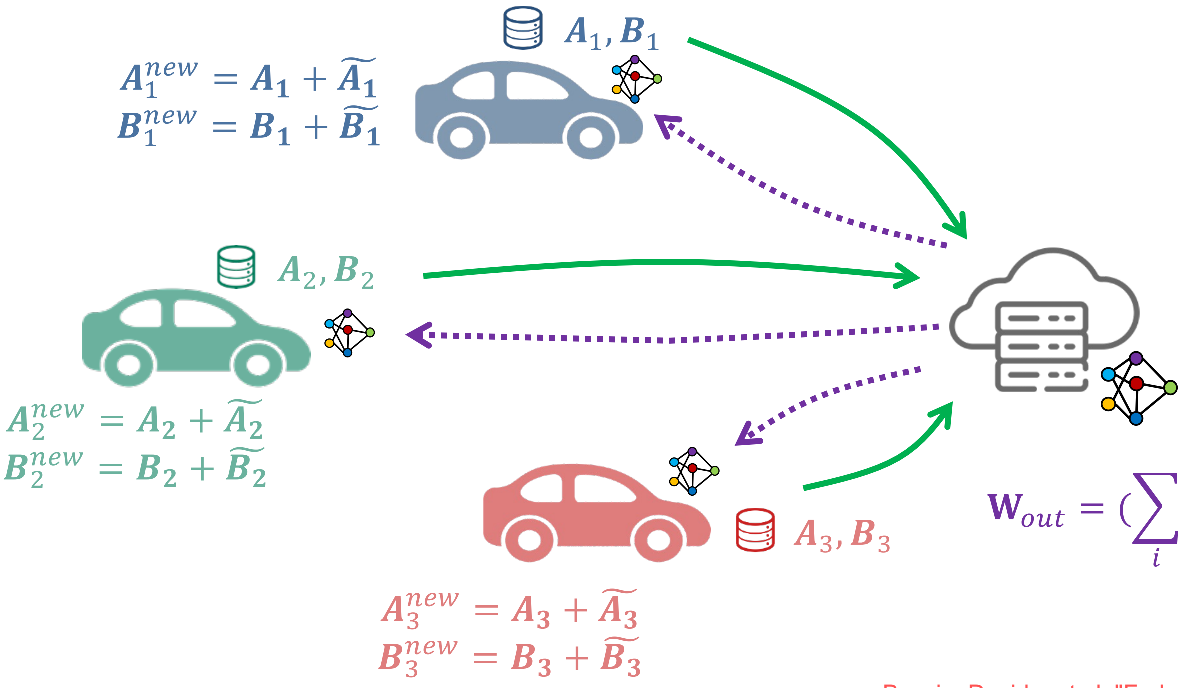
Readout training: offline

- Incremental learning

$$\mathbf{W}_{out} = \underbrace{\mathbf{D}\,\mathbf{H}^T}_{\sum_{i\in C} D_i H_i^T = \sum_{i\in C} A_i}(\underbrace{\mathbf{H}\mathbf{H}^T}_{\sum_{i\in C} H_i H_i^T = \sum_{i\in C} B_i} + \lambda\,\mathbf{I})^{-1}$$

$$\mathbf{W}_{out} = (\sum_i \boldsymbol{A}_i)\left(\sum_i \boldsymbol{B}_i + \lambda\,\mathbf{I}\right)^{-1}$$

Incremental Federated Learning - IncFed



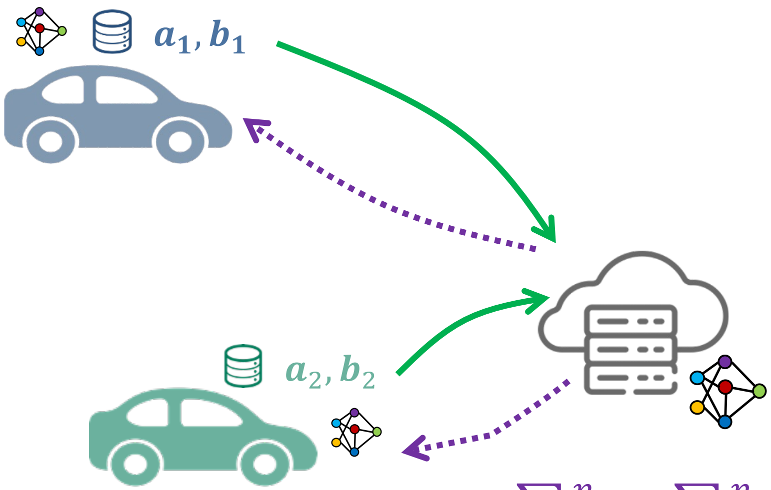
- Training data is not transferred from the clients to the server
- The learned solution is mathematically equivalent to centralized learning

$$\mathbf{W}_{out} = \left(\sum_i \boldsymbol{A}_i\right)\left(\sum_i \boldsymbol{B}_i + \boldsymbol{\lambda}\,\mathbf{I}\right)^{-1}$$

Bacciu, Davide, et al. "Federated reservoir computing neural networks." 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021.

Federated Intrinsic Plasticity - FedIP

Variant of the IP learning rule for federated scenarios



$$\boldsymbol{a}, \boldsymbol{b} = \sum_{i \in S^t} \frac{n_c}{n} \boldsymbol{a}_i \,, \sum_{i \in S^t} \frac{n_c}{n} \boldsymbol{b}_i$$

%TR	WESAD		HHAR	
	w/o FedIP	w/ FedIP	w/o FedIP	w/ FedIP
25%	72.09 ±0.59	**78.68 ±0.12**	57.08 ±3.11	**69.83 ±0.64**
50%	72.04 ±1.03	**77.43 ±0.19**	**63.88 ±6.02**	57.74 ±0.19
75%	76.53 ±1.08	**77.97 ±0.41**	**71.09 ±0.56**	**71.08 ±0.69**
100%	77.78 ±0.58	**79.42 ±0.39**	70.29 ±0.99	**71.38 ±0.43**

V. De Caro, C. Gallicchio, D. Bacciu. “Federated adaptation of reservoirs via intrinsic plasticity” ESANN 2022

Fedray

Torch-ESN

FEDRAY

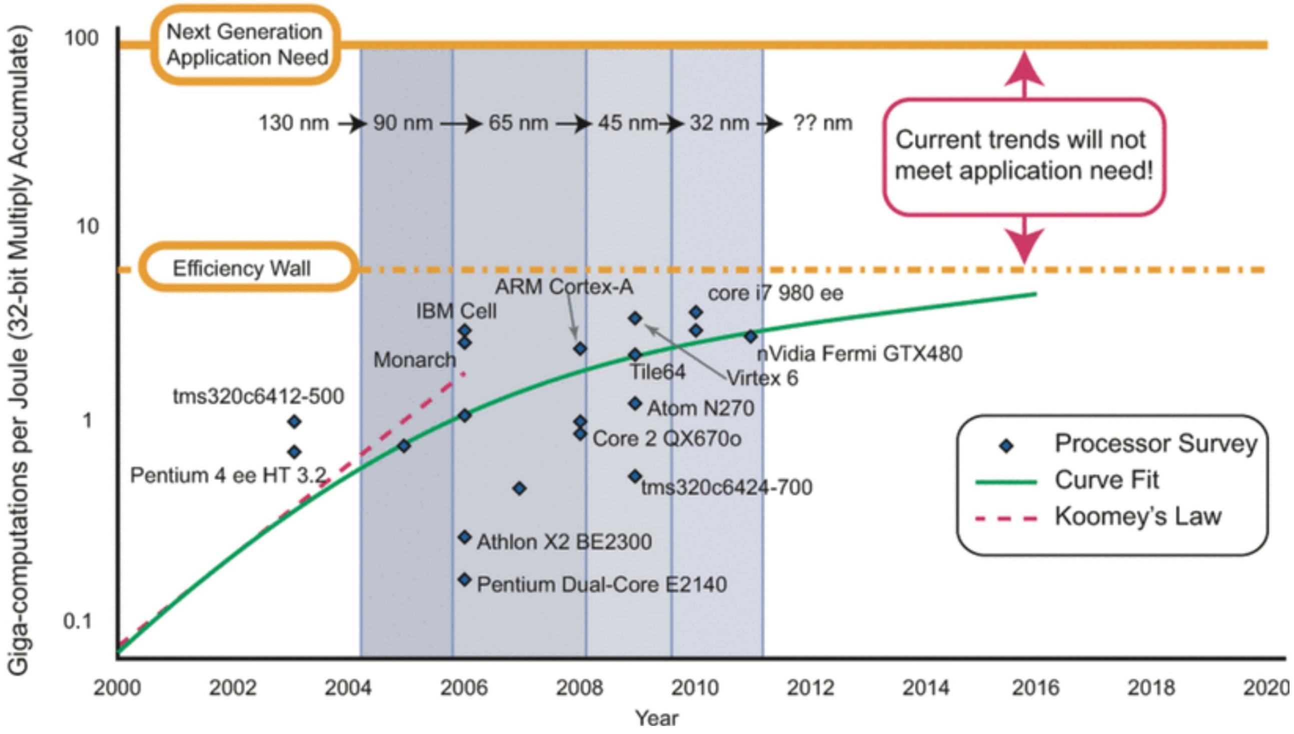
```python
@fedray.remote
class FedESNClient(FedRayNode):
    def build(self, dataset: str, batch_size: int) -> None:
        self.wrapper = VanillaESNWrapper(dataset, self.id, batch_size)

    def train(self, mu: float, sigma: float, eta: float, epochs: int):
        while True:
            reservoir: Reservoir = self.receive().body["model"]
            reservoir = self.wrapper.ip_step(
                reservoir=reservoir, mu=mu, sigma=sigma, eta=eta, epochs=epochs
            )
            state_dict = reservoir.state_dict()
            n_samples = self.wrapper.get_dataset_size()
            self.send(
                header="ip_update",
                body={
                    "net_a": state_dict["net_a"].cpu(),
                    "net_b": state_dict["net_b"].cpu(),
                    "n_samples": n_samples,
                },
            )
```

```python
@fedray.remote
class FedESNServer(FedRayNode):
    def train(self, reservoir: Reservoir):
        ip_aggregator = FedAvgAggregator()
        while True:
            self.send("reservoir", {"model": reservoir})
            ip_aggregator.setup(self.neighbors)
            while not ip_aggregator.ready:
                ip_aggregator(self.receive())
            ip_params = ip_aggregator.compute()
            reservoir.load_state_dict(ip_params, strict=False)
            self.update_version(reservoir=reservoir)
```

```python
federation = ESNFederation(
    dataset=dataset,
    batch_size=batch_size,
    n_clients_or_ids=10,
    roles=["train" for _ in range(10)],
)
reservoir = Reservoir(**reservoir_params)
federation.ip_train(reservoir, mu, sigma, eta, epochs)
for r in range(rounds):
    model = federation.pull_version()["model"]
federation.stop()
```

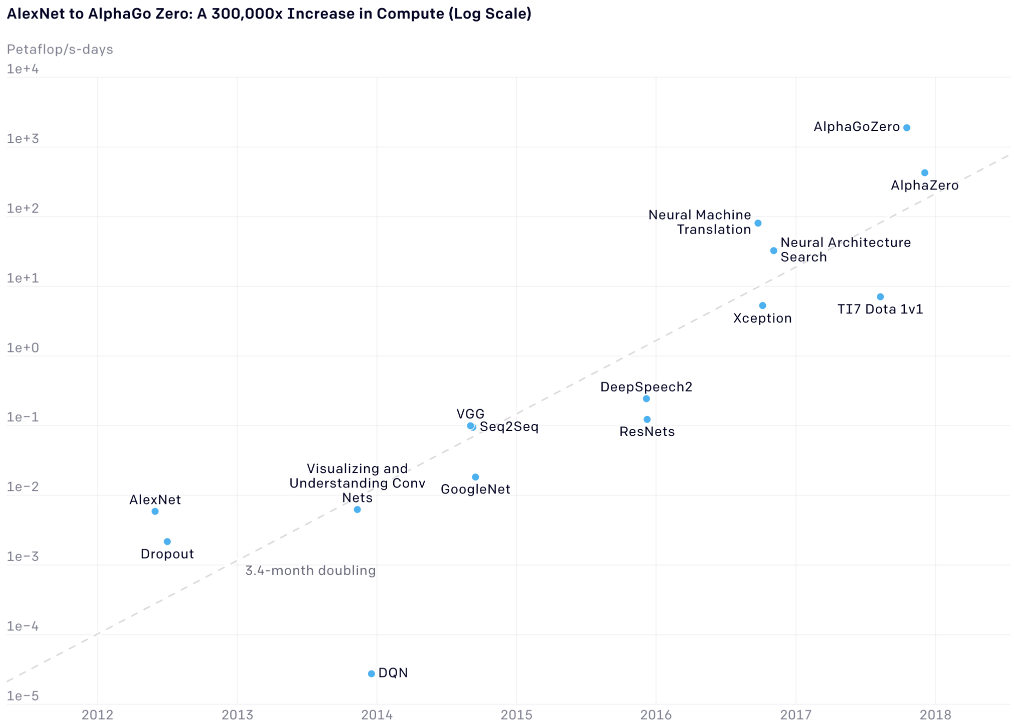
Further Advances

Energy efficiency



Marr, Bo, et al. "Scaling energy per operation via an asynchronous pipeline." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 21.1 (2012): 147-151.

Energy consumption matters!



- 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

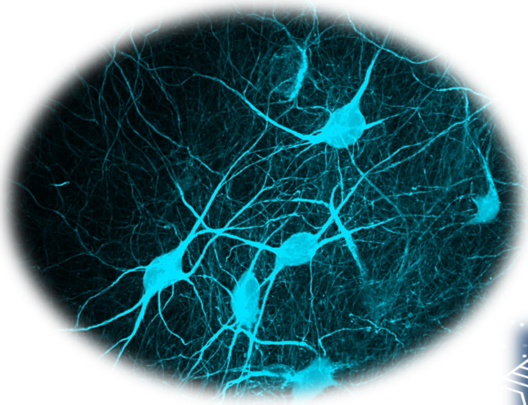
Yang You, Zhao Zhang, James Demmel, Kurt Keutzer, Cho-Jui Hsieh

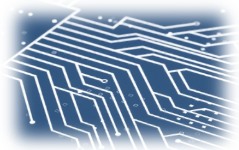
(Submitted on 14 Sep 2017)

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.

vs the Brain...





≈30 PFlops

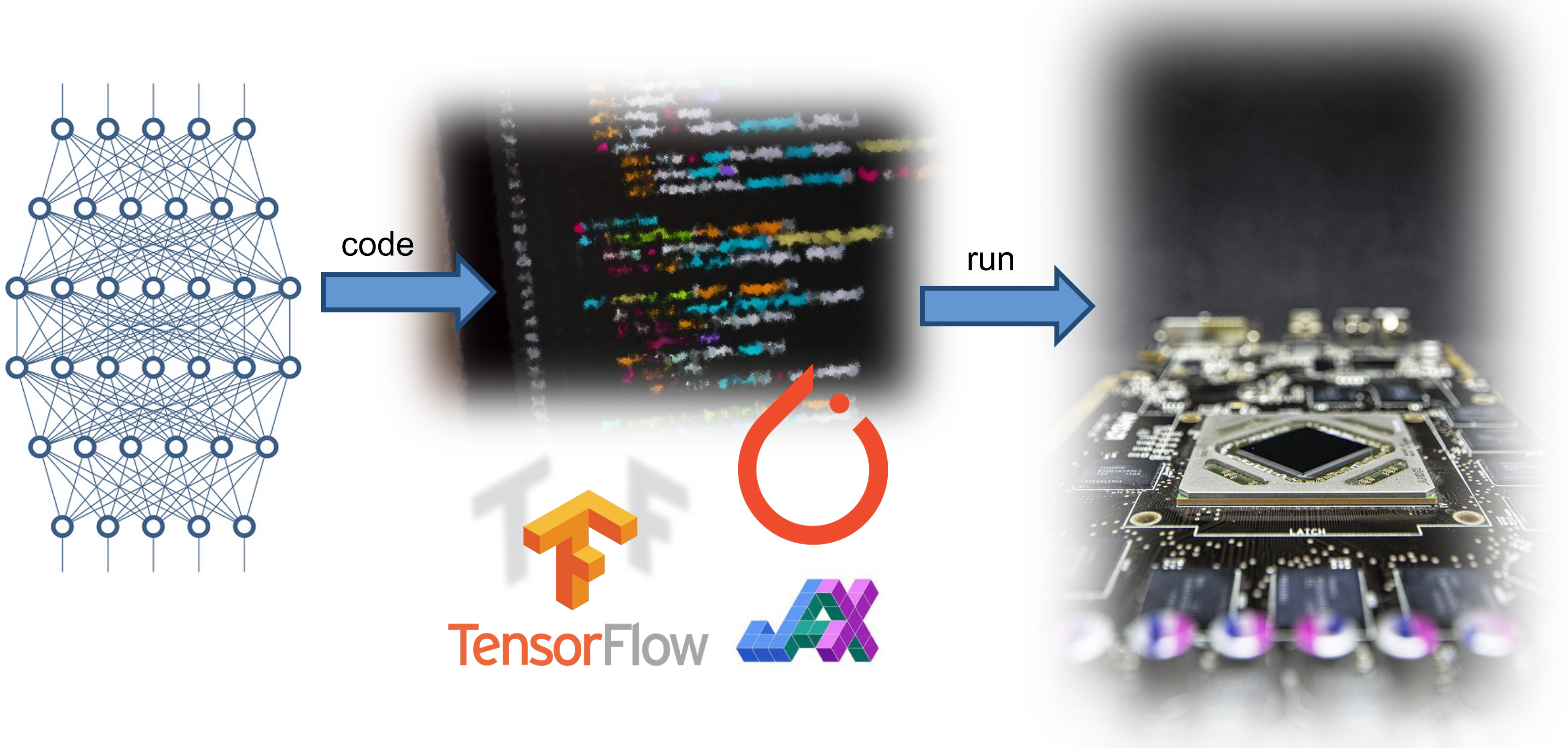
10 MW vs 20 W

memory and computing are co-located

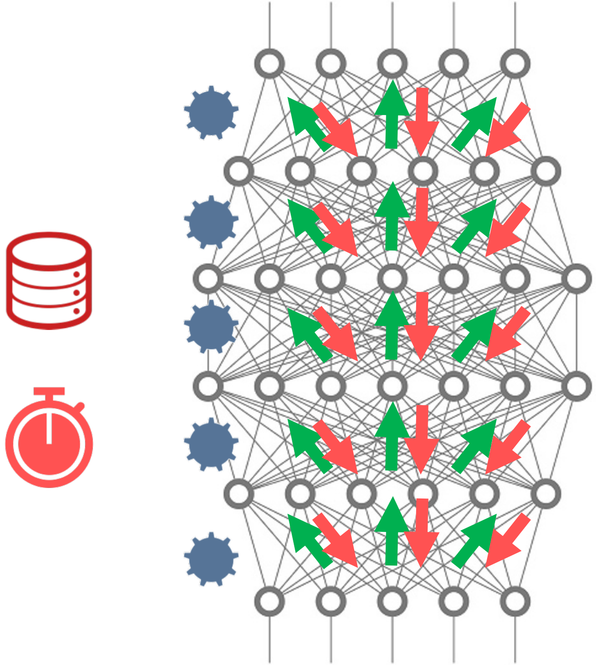
10^{11} neurons, 10^{15} synapses

10000 synapses/neuron

Running DL architectures



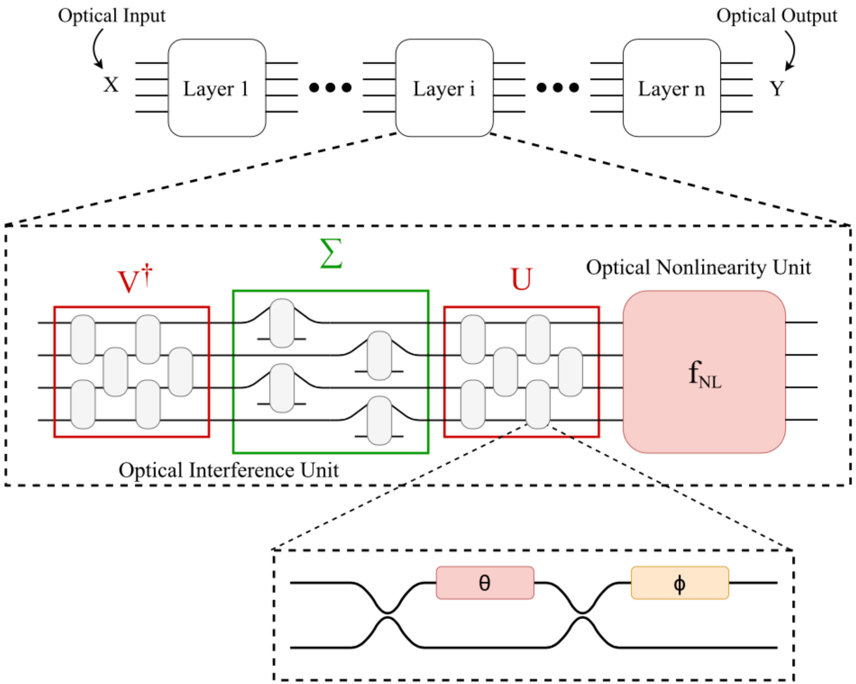
NNs in neuromorphic HW



1. circuit for the forward path
2. memory to store neurons' activations
3. circuit for the backward path
4. circuit for adjusting the free parameters
5. time

   synchronicity of the layers operations in the forward & backward passes

Neuromorphic chip: Photonics

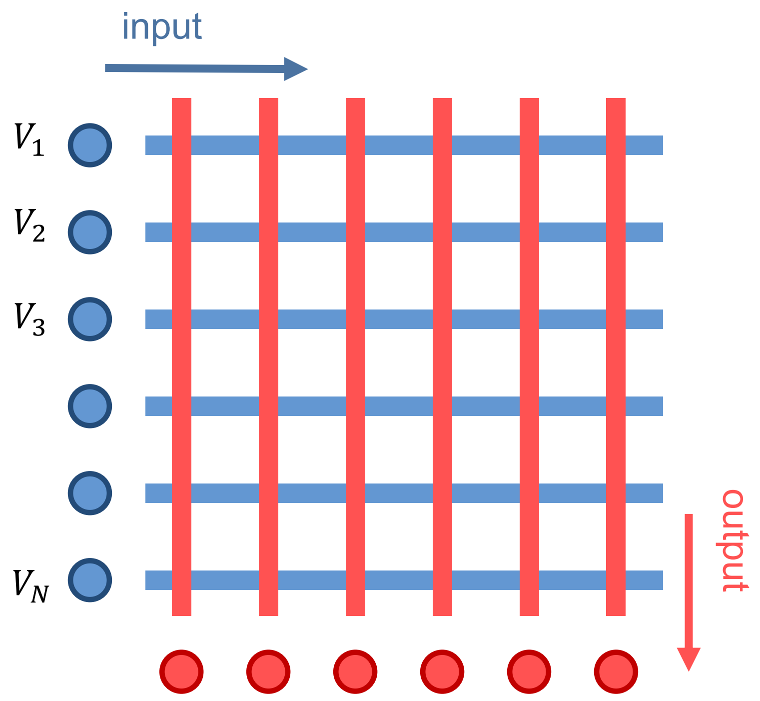


- neurons implemented by optical resonators
- the flow of information is light
- synapses implemented by multiple interferometers or transmission of optical waveguides

De Marinis, Lorenzo, et al. "Photonic neural networks: a survey." *IEEE Access* 7 (2019): 175827-175841.

Neuromorphic chip: CMOS with Memristors

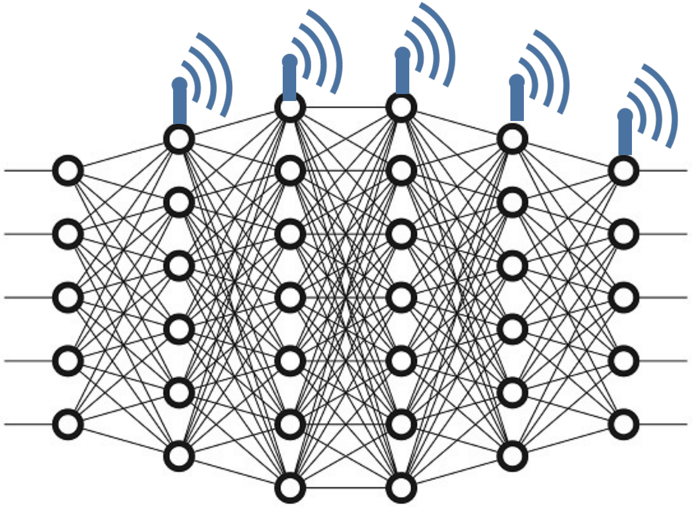
- neurons implemented in CMOS
- the flowing information is electrical current
- synapses implemented as memristors
  - nanoscale resistors
  - non-volatile analog conductance states

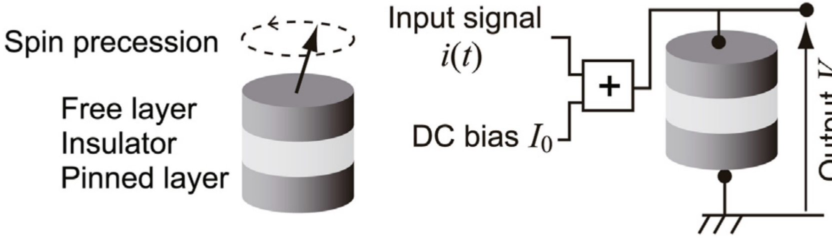


$$I_j = \sum_i G_{ij} V_i$$

Neuromorphic chip: Spintronics

- magnetic nano-neurons
- synapses implemented as radiowaves



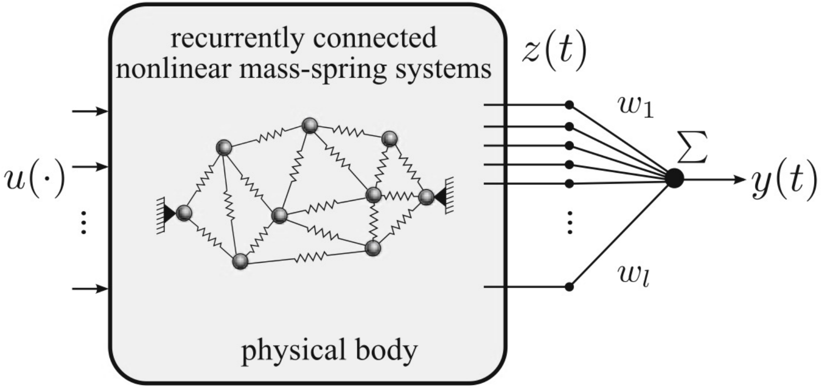


Torrejon, Jacob, et al. "Neuromorphic computing with nanoscale spintronic oscillators." *Nature* 547.7664 (2017): 428-431.

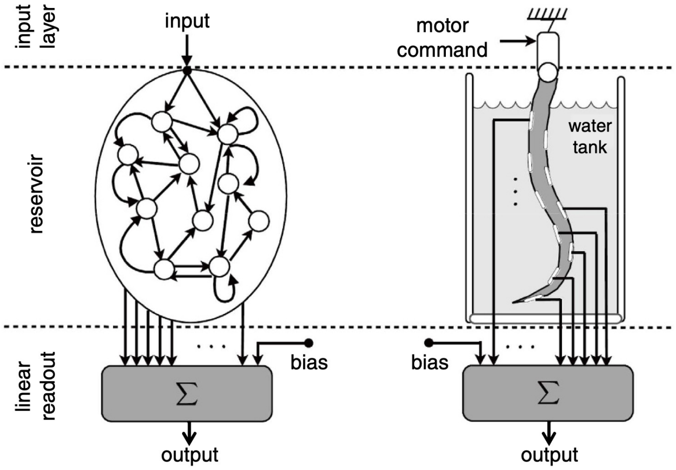
Locatelli, Nicolas, Vincent Cros, and Julie Grollier. "Spin-torque building blocks." *Nature materials* 13.1 (2014): 11-20.

Mechanical systems

- Neural Networks implemented by physical bodies or soft robots



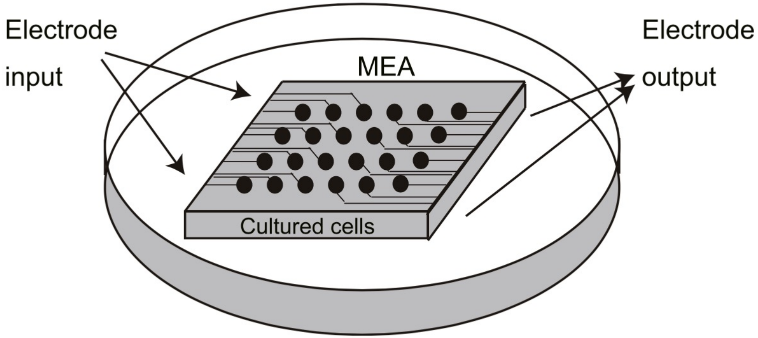
Hauser, Helmut, et al. "Towards a theoretical foundation for morphological computation with compliant bodies." *Biological cybernetics* 105.5 (2011): 355-370.



Nakajima, Kohei, et al. "Information processing via physical soft body." Scientific reports 5.1 (2015): 1-11.

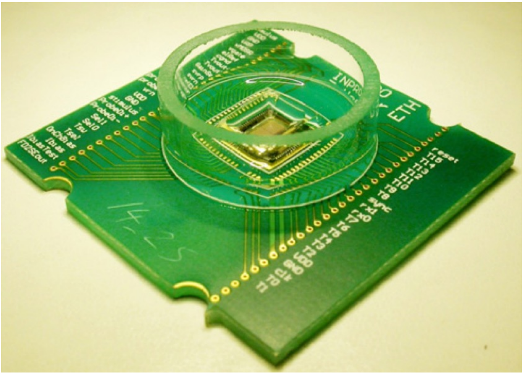
Biological systems

- Neural Networks implemented on in vitro biological components



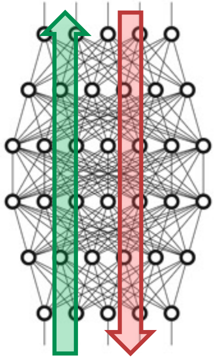
Tanaka, Gouhei, et al. "Recent advances in physical reservoir computing: A review." *Neural Networks* 115 (2019): 100-123.

Obien, Marie Engelene J., et al. "Revealing neuronal function through microelectrode array recordings." *Frontiers in neuroscience* 8 (2015): 423.



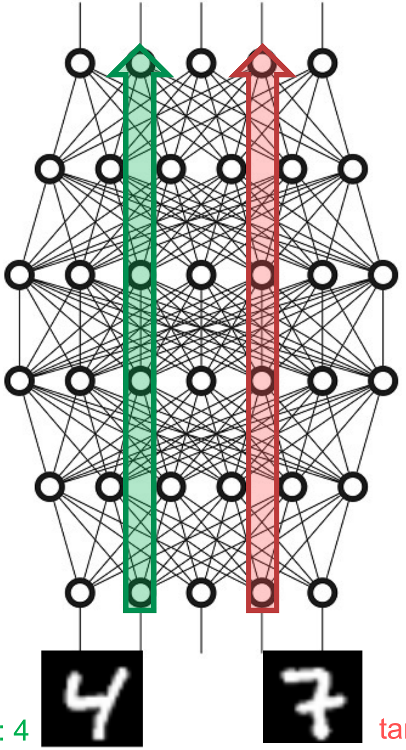
Hafizovic, Sadik, et al. "A CMOS-based microelectrode array for interaction with neuronal cultures." *Journal of neuroscience methods* 164.1 (2007): 93-106.

Beyond backpropagation



- Memory overhead for storing the neural activations
- Sequentiality and synchronicity of forward & backward passes
- Weight transport problem: forward-backward weight symmetry

- Biological implausible
    - no evidence for derivatives propagation
    - no evidence for storing neural activations
    - no evidence for mirrored connections (top-down $\neq$ bottom-up)

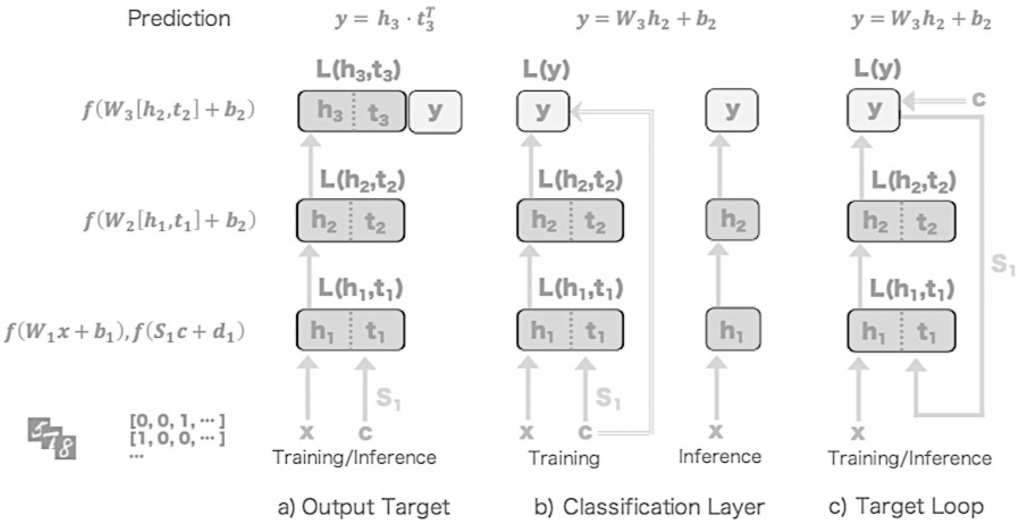


Forward Forward

- Local goodness $\sum_j h_j^2$
- Local loss function $\mathcal{L} = \sigma(\pm(\sum_j h_j^2 - \theta))$
- "positive" forward pass: increase the goodness of each layer for real data
- "negative" forward pass: reduce the goodness of each layer for wrong data

Hinton, Geoffrey. "The forward-forward algorithm: Some preliminary investigations." arXiv preprint arXiv:2212.13345(2022).

SigProp



- every layer processes both input and learning signal
  - h_i - output of the layer
  - t_i - target for the layer

$$h_1, t_1 = f(W_1 x + b_1), f(S_1 c_m + d_1)$$
$$[h_2, t_2] = f(W_2[h_1, t_1] + b_2)$$
$$[h_3, t_3] = f(W_3[h_2, t_2] + b_2)$$

- local loss

$$\mathcal{L}(h_i, t_i) = CE(y_i^*, O_{dot}(h_i, t_i))$$

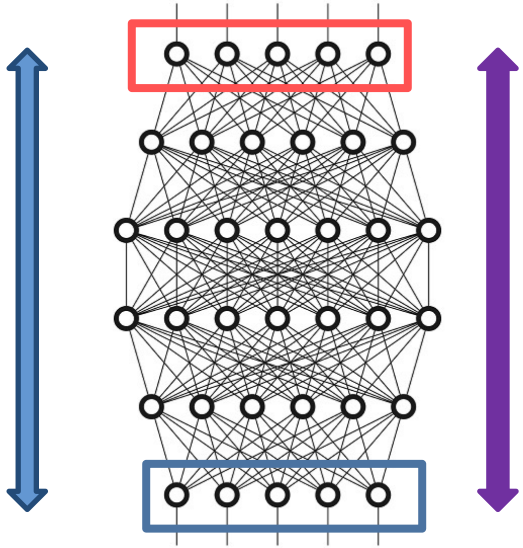
Kohan, Adam, Edward A. Rietman, and Hava T. Siegelmann. "Forward Signal Propagation Learning." arXiv preprint arXiv:2204.01723 (2022).

Equilibrium Propagation

$$E(u) := \frac{1}{2}\sum_i u_i^2 - \frac{1}{2}\sum_{i \neq j} W_{ij}\rho(u_i)\rho(u_j) - \sum_i b_i\rho(u_i).$$

$$C := \frac{1}{2}\|y - d\|^2.$$

$$F := E + \beta C$$



$$\Delta W_{ij} \propto \frac{1}{\beta}\left(\rho\left(u_i^{\beta}\right)\rho\left(u_j^{\beta}\right) - \rho\left(u_i^{0}\right)\rho\left(u_j^{0}\right)\right)$$

Scellier, Benjamin, and Yoshua Bengio. "Equilibrium propagation: Bridging the gap between energy-based models and backpropagation." *Frontiers in computational neuroscience* 11 (2017): 24.

Laydevant, Jérémie, et al. "Training Dynamical Binary Neural Networks with Equilibrium Propagation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

Conclusions

Conclusions

- Leverage principled architectural biases of dynamical systems for fast computation in sequential data
- Hardware-Software co-design
    - activation functions, weight quantization, architecture & topology
    - cyclic, deep, Euler reservoirs
- Simplified training algorithms & learning beyond backprop
    - Local adaptation, Federated learning
    - Intrinsic Plasticity, FORCE, Forward-forward, SigProp, ...