Pervasive AI

ECML-PKDD 2023 Tutorial

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

http://pai.di.unipi.it/tutorial-on-continual-learning-in-distributed-and-heterogeneous-systems

Advanced Models

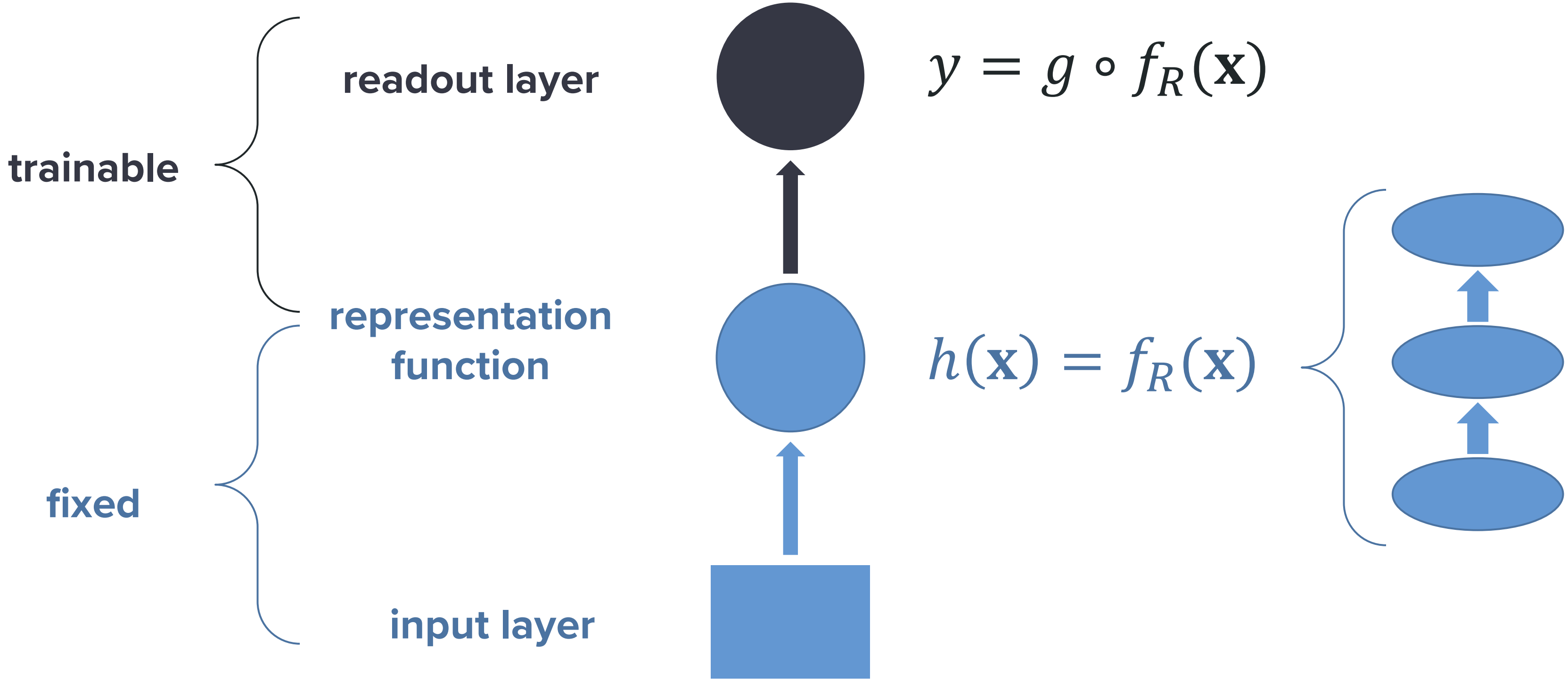
Davide Bacciu, Antonio Carta, Patrizio Dazzi, **Claudio Gallicchio**
University of Pisa, Italy

Outline

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

Deep Randomized Neural Networks

Deep Randomized Architectures



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 image prior

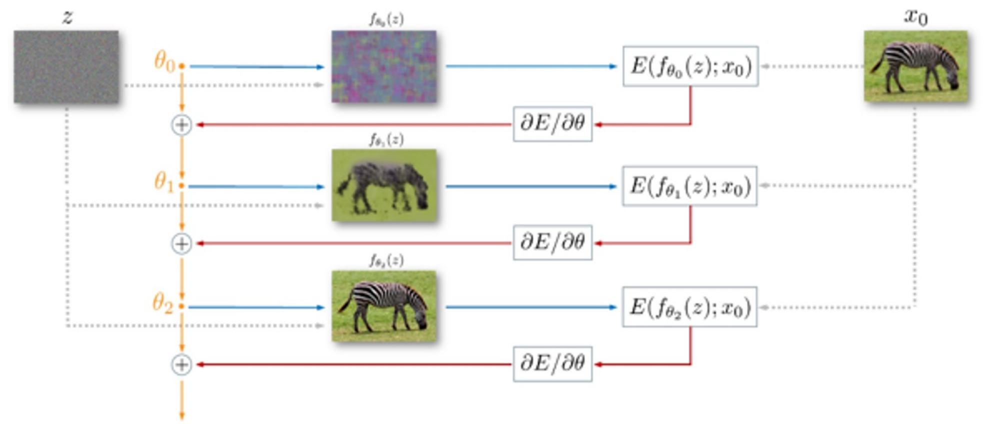
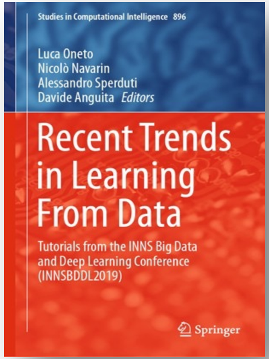


Fig. 2: **Image restoration using the deep image prior.** Starting from a random weights θ_0, we iteratively update them in order to minimize the data term eq. (2). At every iteration the weights θ are mapped to an image $x = f_\theta(z)$, where z is a fixed tensor and the mapping f is a neural network 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).

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.

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.

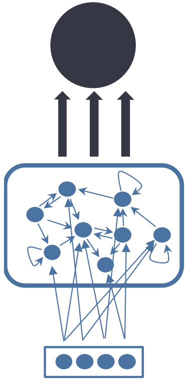
Vanilla Recurrent neural nets

$$\mathbf{h}(t) = \tanh(\boxed{\mathbf{W_h}}\,\mathbf{h}(t-1) + \boxed{\mathbf{W_x}}\,\mathbf{x}(t) + \boxed{\mathbf{b}})$$

trainable

Echo State Networks

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

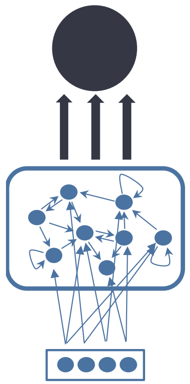


Yildiz, Izzet B., Herbert Jaeger, and Stefan J. Kiebel. "Re-visiting the echo state property." Neural networks 35 (2012): 1-9.

Echo State Networks

fixed weights

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



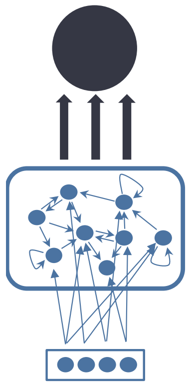
Yildiz, Izzet B., Herbert Jaeger, and Stefan J. Kiebel. "Re-visiting the echo state property." Neural networks 35 (2012): 1-9.

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

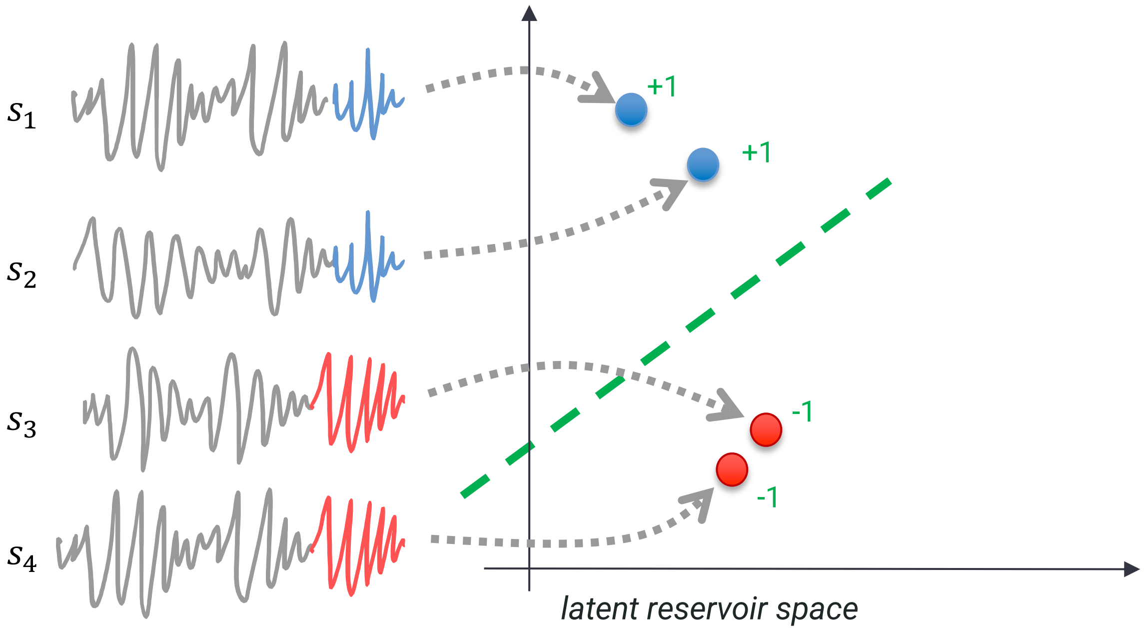
- global asymptotic Lyapunov stability condition
- spectral radius $\rho < 1$

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?

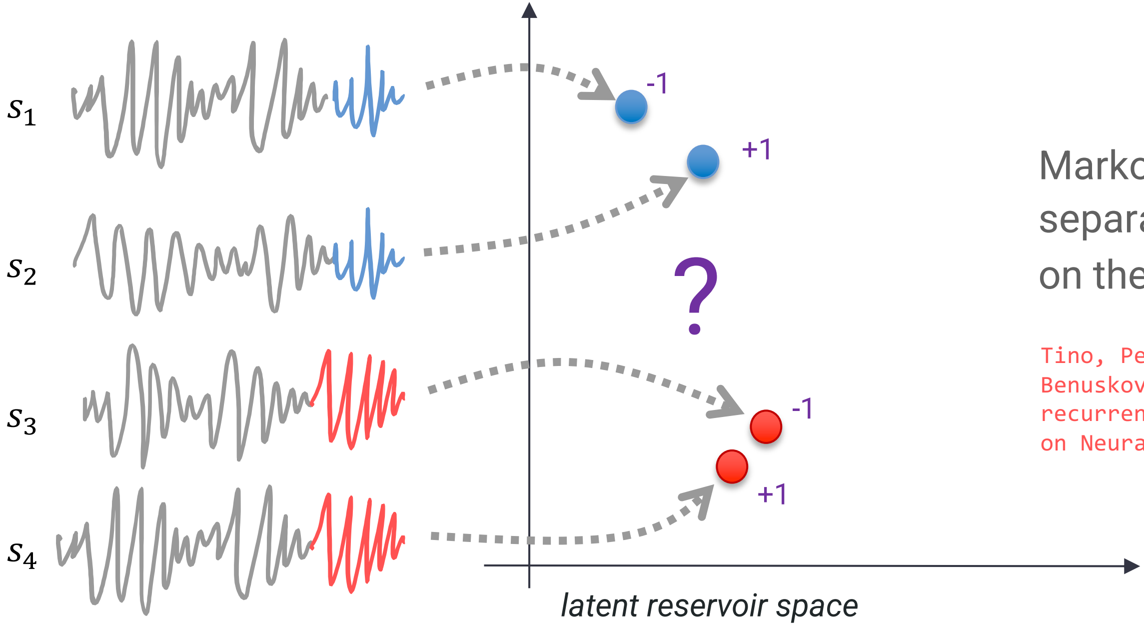
Because of the architectural bias of contracting RNNs



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.

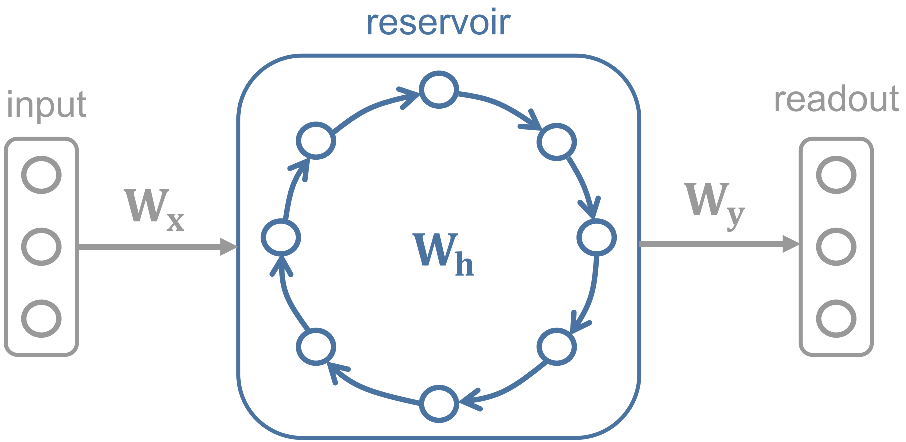
Good reservoirs

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.

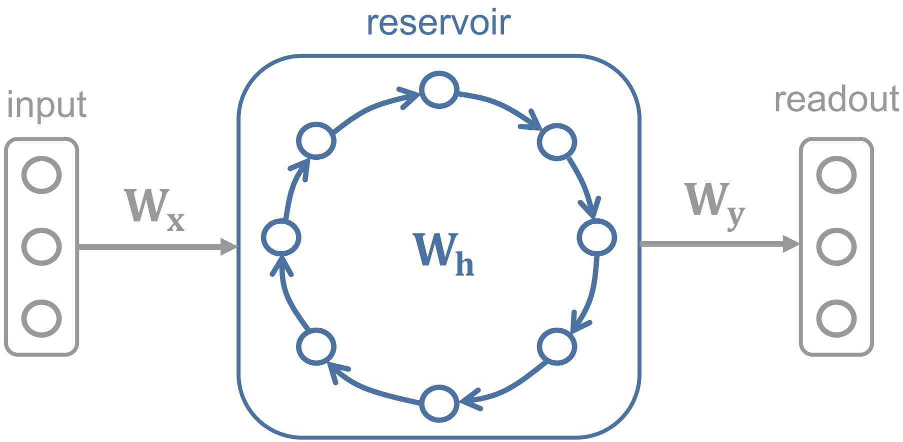


$$\mathbf{W}_\mathbf{h} = \begin{pmatrix} 0 & 0 & \ldots & 0 & \hat{w} \\ \hat{w} & 0 & \ldots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \ldots & 0 & \hat{w} & 0 \end{pmatrix}$$

- 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.



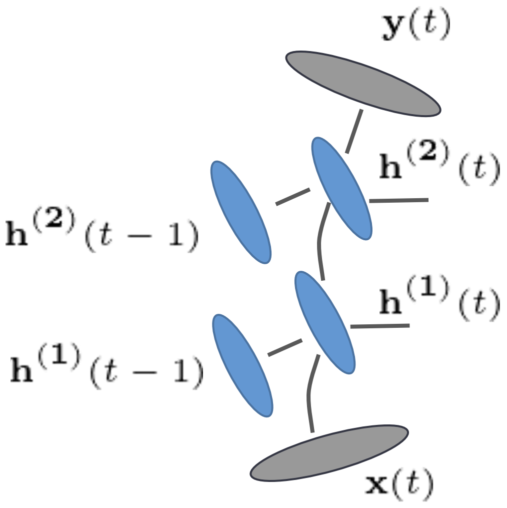
$$\mathbf{W_h} = \begin{pmatrix} 0 & 0 & \dots & 0 & \hat{w} \\ \hat{w} & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & 0 & \hat{w} & 0 \end{pmatrix}$$

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

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

nice eigenstructure

Deep reservoirs



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

Reservoir = set of nested non-linear dynamical systems

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

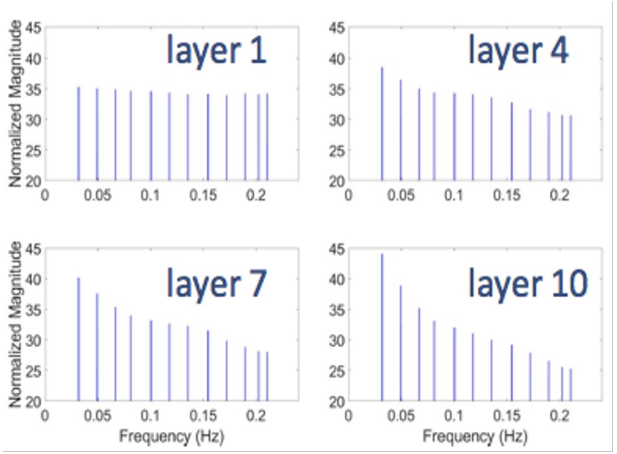
driving input

...

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

- Multiple time-scales
- Multiple frequencies
- Richer dynamics

even without training of the recurrent connections



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

Euler reservoirs for long-range propagation

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

step size

diffusion

$$\mathbf{h}(t) = \mathbf{h}(t-1) + \varepsilon \tanh\left(\boxed{\mathbf{W}_x}\, \mathbf{x}(t) + \left(\boxed{\mathbf{W}_h} - \boxed{\mathbf{W}_h^T} - \gamma \mathbf{I}\right)\mathbf{h}(t-1) + \boxed{\mathbf{b}}\right)$$

untrained

dynamics are arbitrarily close to the edge of chaos

Gallicchio, Claudio. "Euler state networks: Non-dissipative Reservoir Computing." arXiv preprint arXiv:2203.09382 (2022).

Time-series classification

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

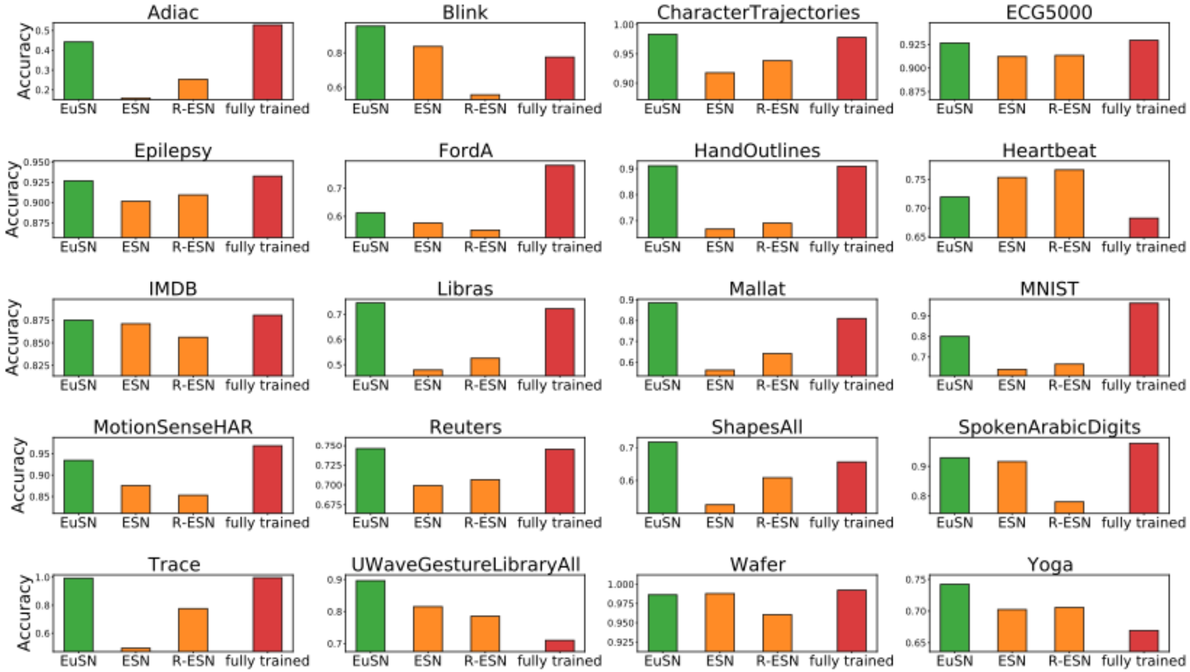


Figure 5: Averaged test set accuracy on the time-series classification benchmarks. The "fully trained" results refer to the trainable model (among RNN, A-RNN and GRU) that achieves the highest accuracy on each task. Further details can be found in Appendix A.

Speed of training

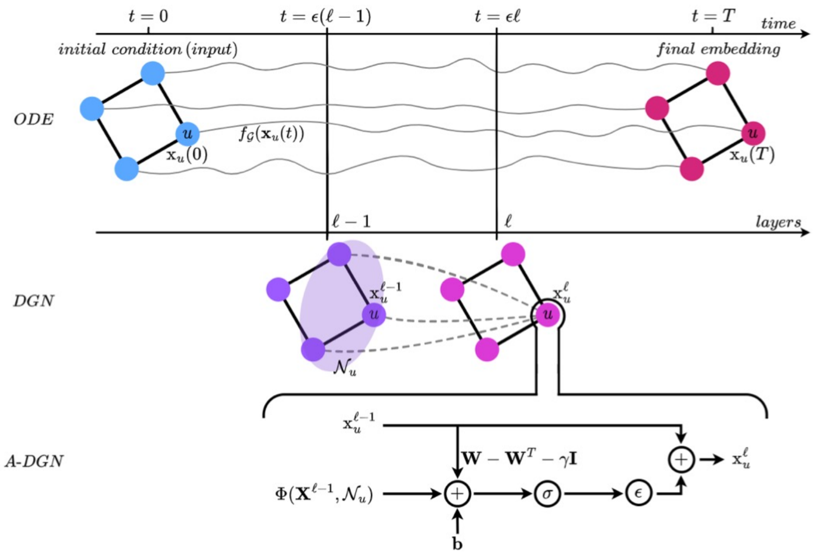
up to 490 x faster

Energy consumption

up to 1750 x cheaper

Antisymmetric Deep Graph Networks

for long-range propagation on graphs



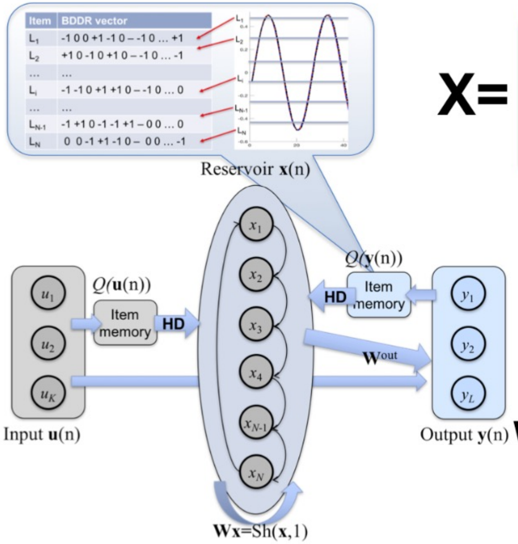
$$\mathbf{x}_u^{\ell} = \mathbf{x}_u^{\ell-1} + \epsilon\sigma\left((\mathbf{W}-\mathbf{W}^T-\gamma\mathbf{I})\mathbf{x}_u^{\ell-1} + \Phi(\mathbf{X}^{\ell-1},\mathcal{N}_u) + \mathbf{b}\right)$$

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

	Diameter	**SSSP**	**Eccentricity**
GCN	0.7424 ± 0.0466	$0.9499\pm9.18\cdot10^{-5}$	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

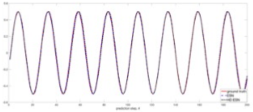
Gravina, Alessio, Davide Bacciu, and Claudio Gallicchio. "Anti-Symmetric DGN: a stable architecture for Deep Graph Networks." ICLR 2023

Integer Echo State Networks



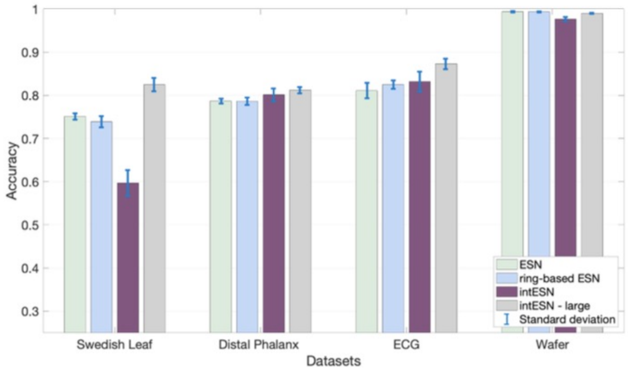
$$X=\begin{pmatrix} 3 & 3 & 3 & -2 & -1 & -2 & -1 \\ 3 & 3 & 2 & 3 & -3 & -2 & -1 \\ 0 & 2 & 3 & 3 & 3 & -2 & -3 \\ -3 & -1 & 1 & 2 & 3 & 2 & -1 \\ 3 & -2 & -2 & 0 & 1 & 2 & 1 \\ 3 & 3 & -1 & -1 & 1 & 2 & 3 \end{pmatrix}$$

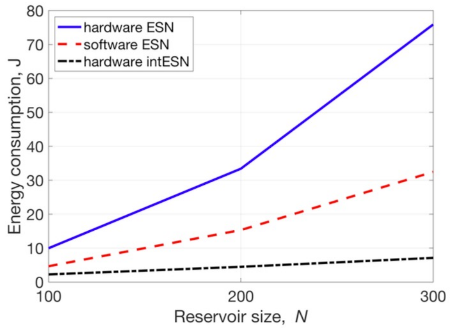
Y=



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

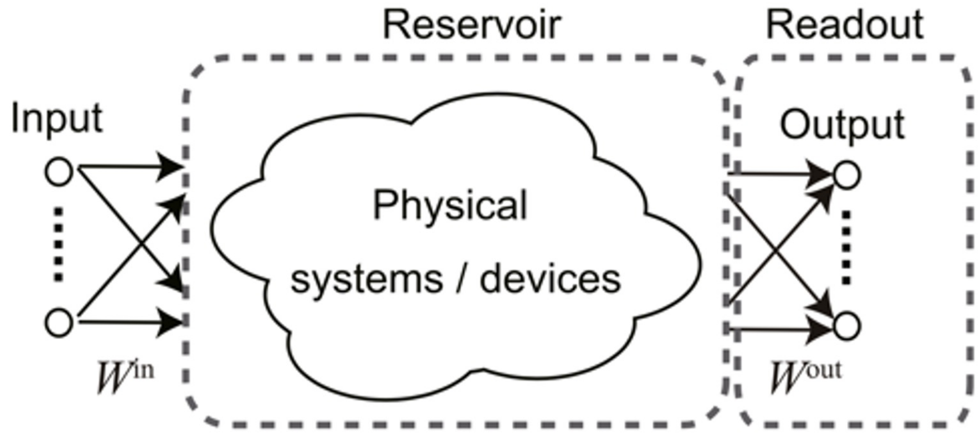
$$x(n) = f_k(Sh(x(n-1),1) + u^{HD}(n) + y^{HD}(n-1))$$

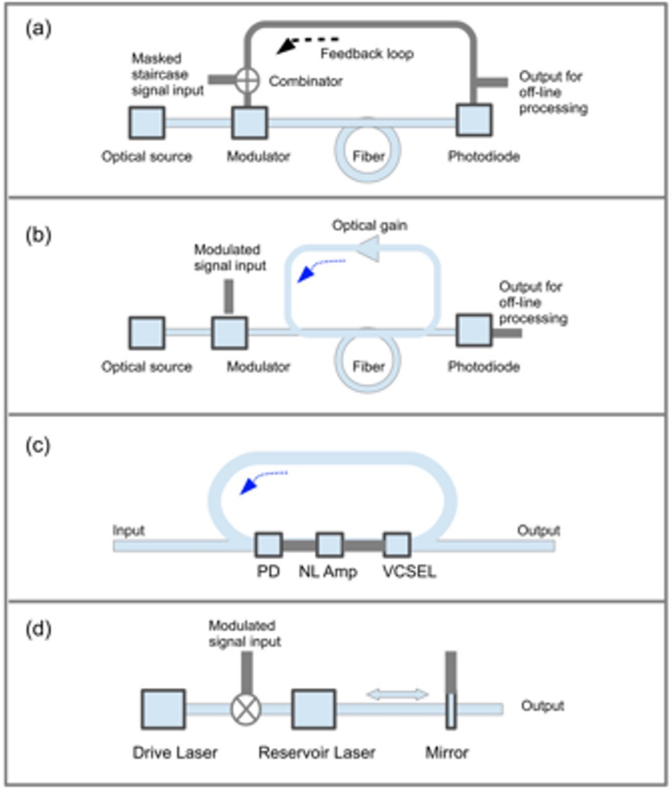




Kleyko, Denis, et al. "Integer echo state networks: efficient reservoir computing for digital hardware." IEEE Transactions on Neural Networks and Learning Systems 33.4 (2020): 1688-1701.

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

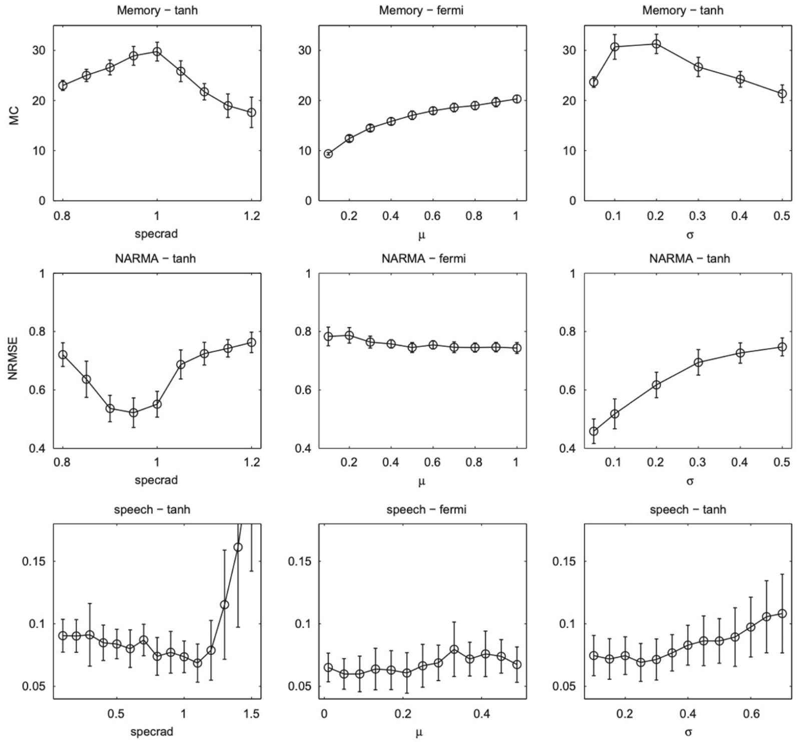


Fig. 4. Results for all three benchmarks for tanh with spectral radius ranging (left column), exponential IP for fermi nodes (middle column), and Gaussian IP for tanh nodes (right column).

- 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

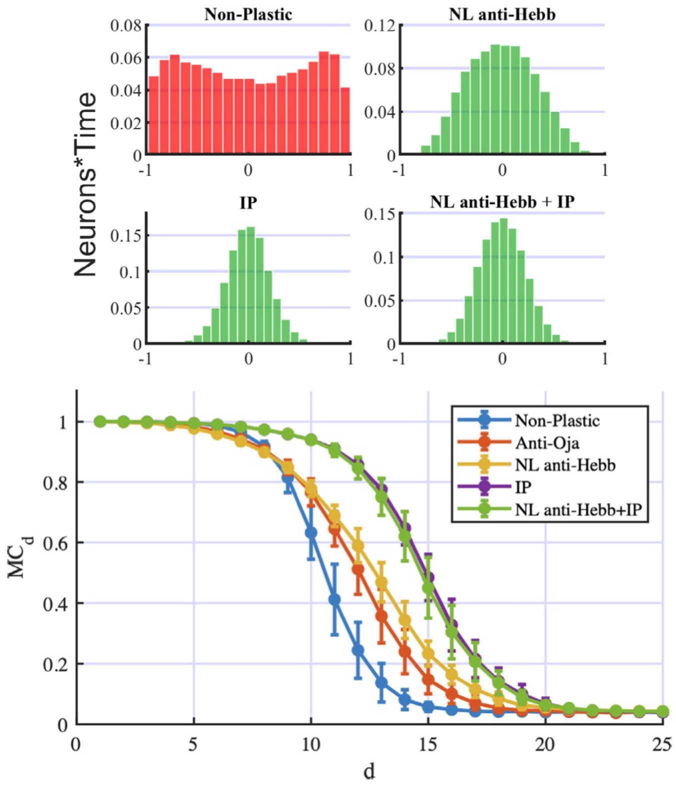
Kullback–Leibler divergence minimization

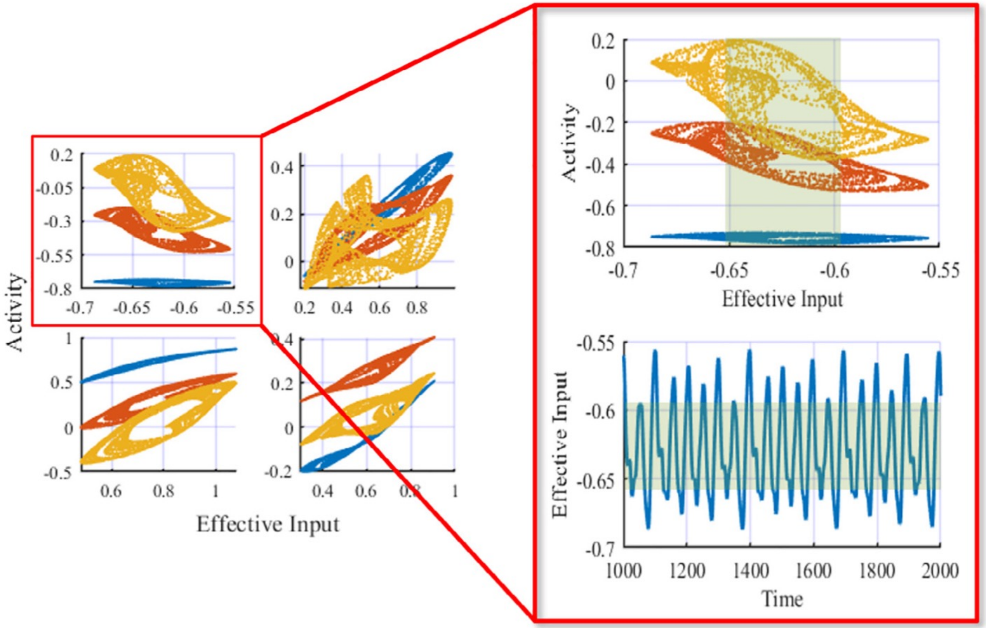
$$\Delta b = -\eta\left(-\frac{\boxed{\mu}}{\boxed{\sigma^2}} + \frac{y}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y)\right),$$

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

hyperparameters

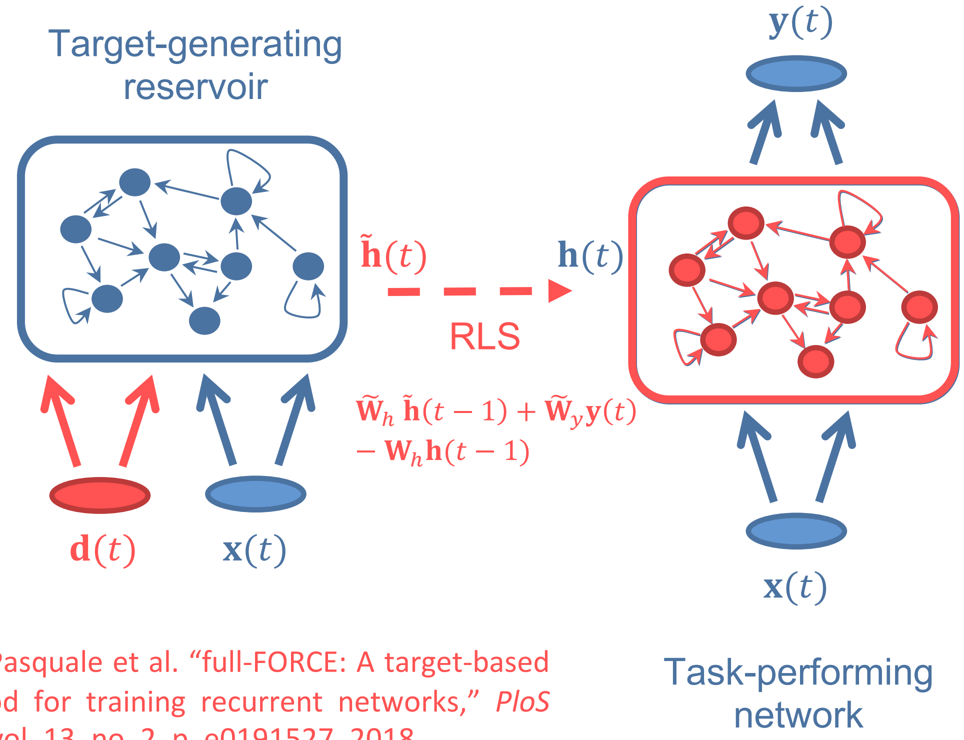
Plasticity improves input separation

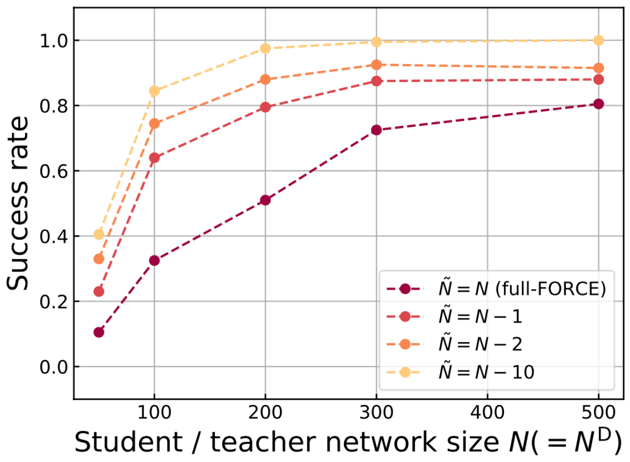




G.B. Morales, C. Mirasso, M.C. Soriano, 2021. Unveiling the role of plasticity rules in reservoir computing. Neurocomputing.

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

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:

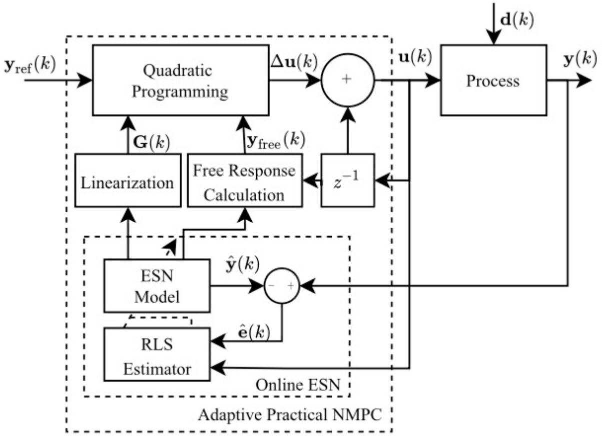
$$\hat{e}_{n-1}(n) = d(n) - \hat{y}_{n-1}(n)$$

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

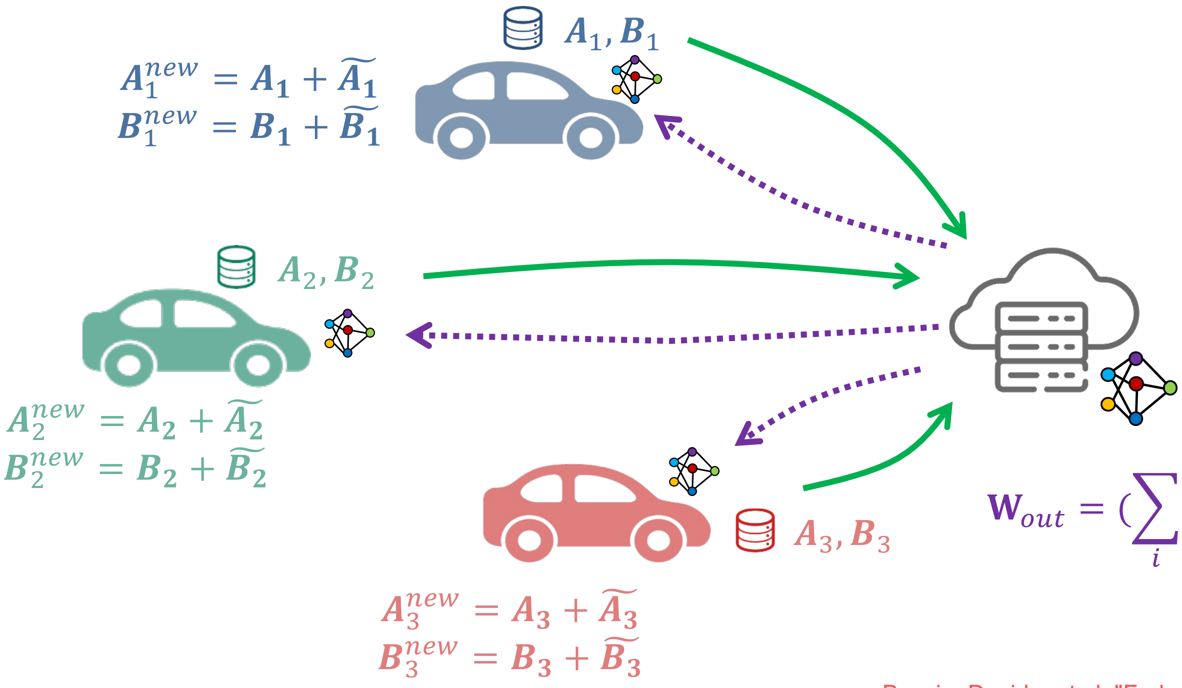
Readout training: offline

- Incremental learning

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

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

Incremental Federated Learning - IncFed



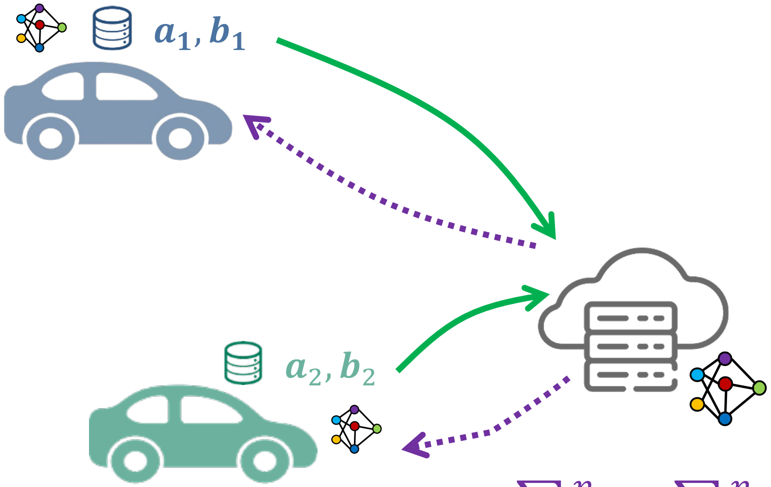
- Training data is not transferred from the clients to the server
- The learned solution is mathematically equivalent to centralized learning

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

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



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

%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

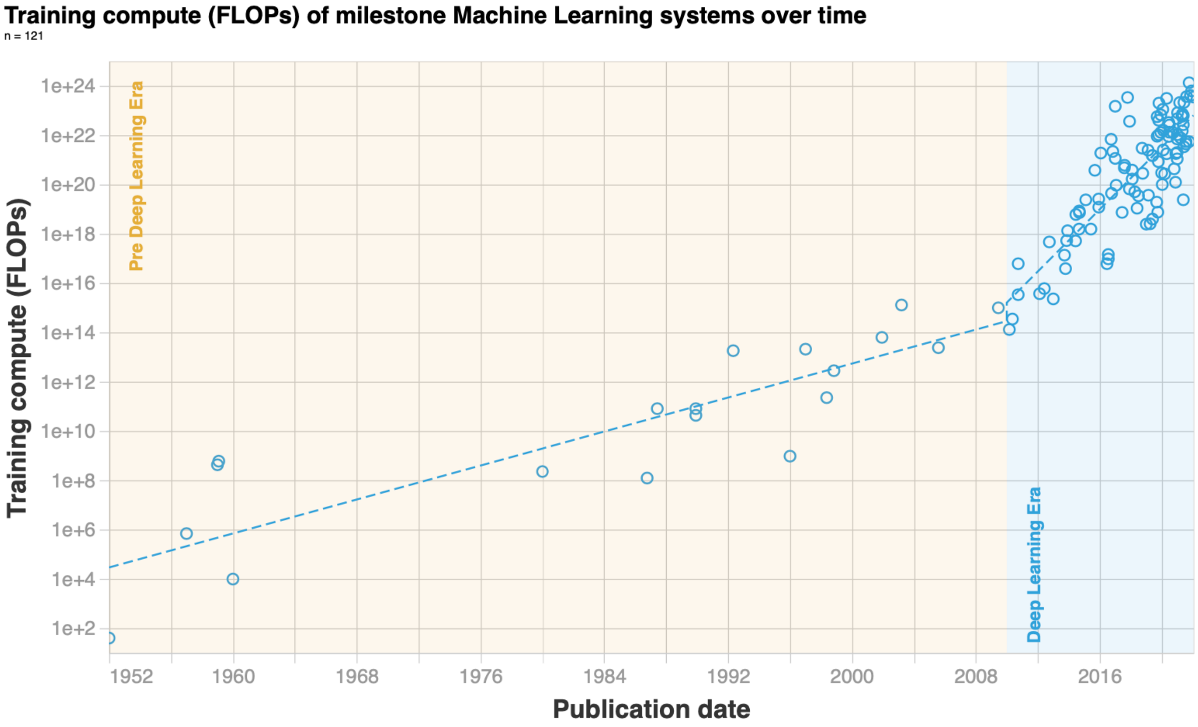
```python
@fedray.remote
class FedESNClient(FedRayNode):
    def build(self, dataset: str, batch_size: int) -> None:
        self.wrapper = VanillaESNWrapper(dataset, self.id, batch_size)

    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



Doubling Time

DL algorithms

$\approx 3\, months$

Moore's law

$\approx 2\, years$

Sevilla, Jaime, et al. "Compute Trends Across Three Eras of Machine Learning." *arXiv e-prints* (2022): arXiv-2202.

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

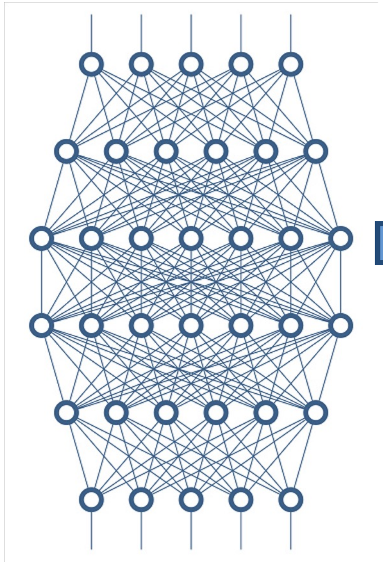
Yang You, Zhao Zhang, James Demmel, Kurt Keutzer, Cho-Jui Hsieh

(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.

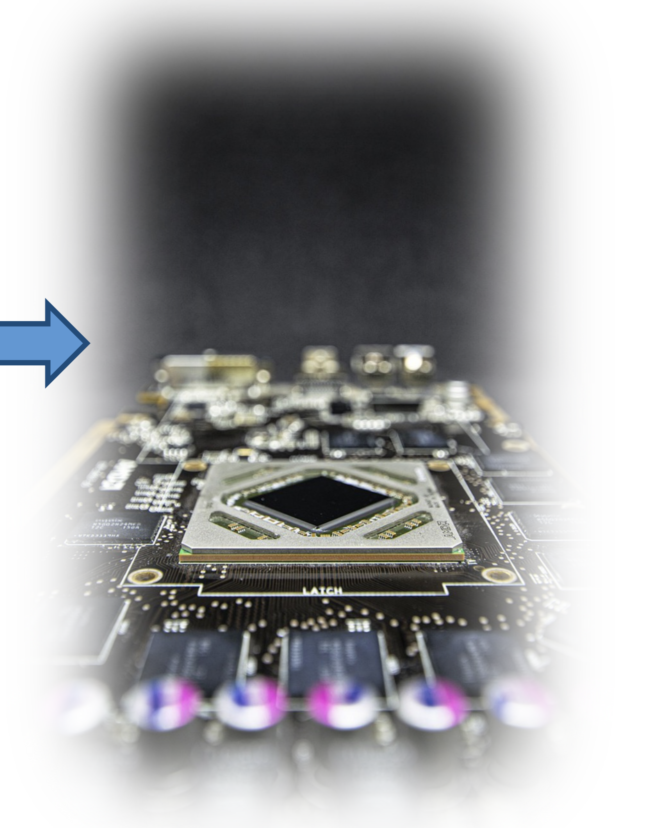
Running DL architectures



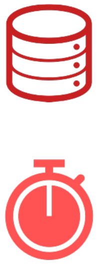
code

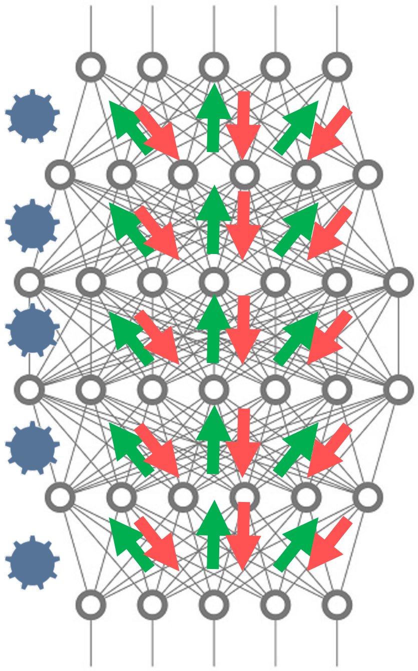
TensorFlow

run



NNs in neuromorphic HW

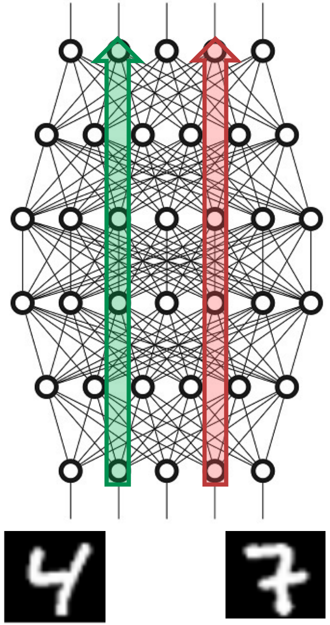




1. circuit for the forward path
2. memory to store neurons' activations
3. circuit for the backward path
4. circuit for adjusting the free parameters
5. time

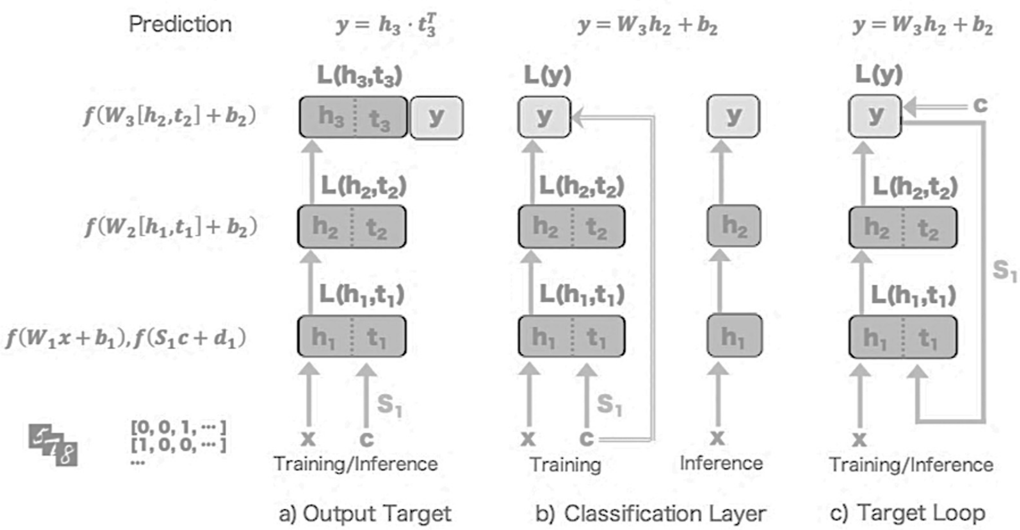
   synchronicity of the layers operations in the forward & backward passes

Forward-forward



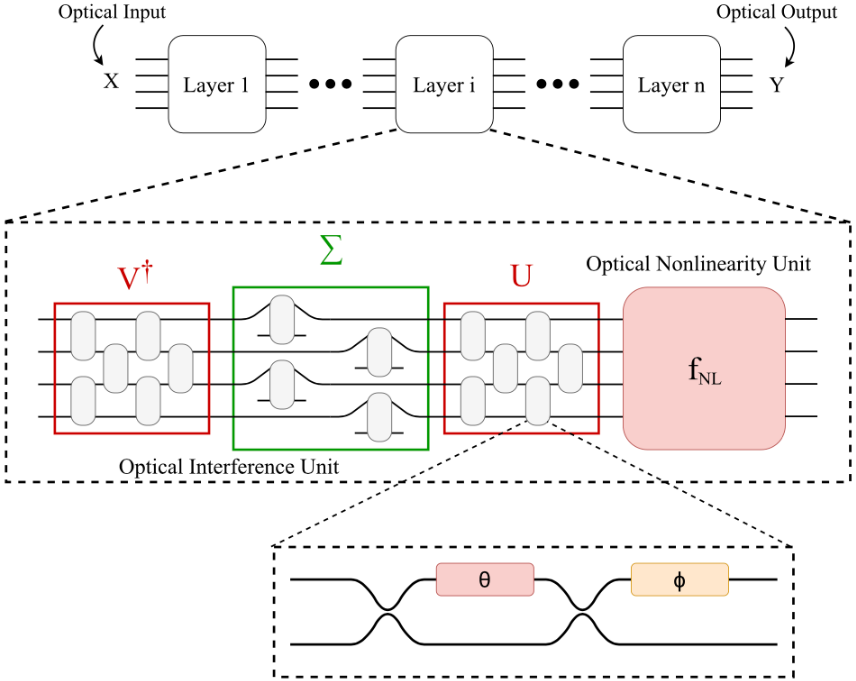
Hinton, Geoffrey. "The forward-forward algorithm: Some preliminary investigations." arXiv preprint arXiv:2212.13345(2022).

Signal Propagation



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

Neuromorphic chip: Photonics

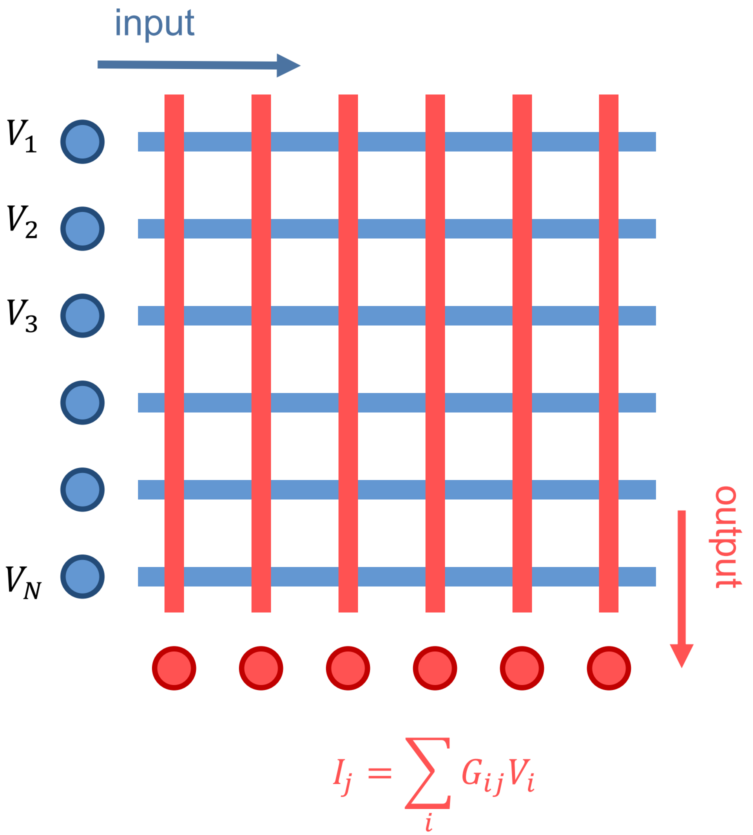


- **the flow of information is light**
- synapses implemented by multiple interferometers or transmission of optical waveguides

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

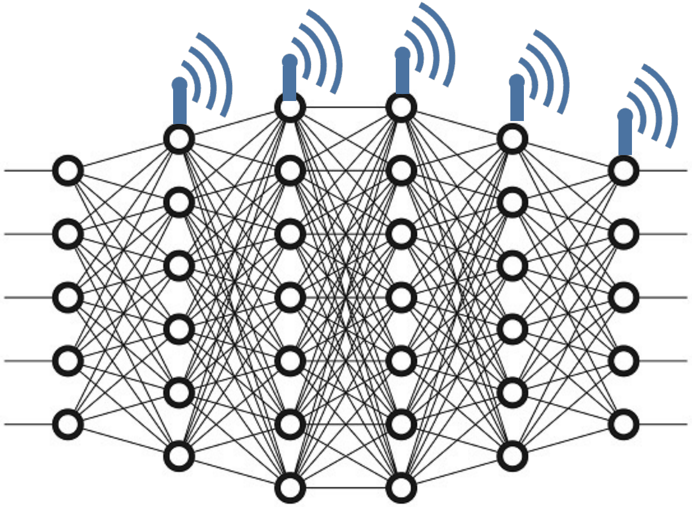
Neuromorphic chip: CMOS with Memristors

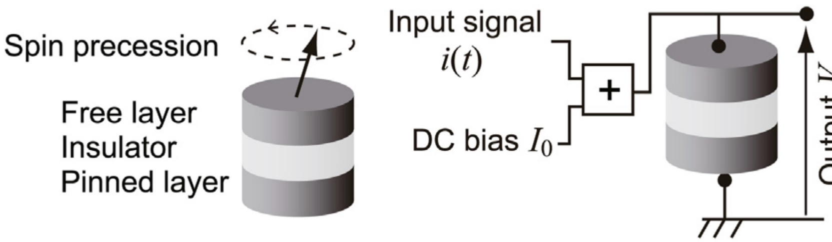
- neurons implemented in CMOS
- **the flowing information is electrical current**
- synapses implemented as memristors
  - nanoscale resistors
  - non-volatile analog conductance states
- synaptic layers can be mapped onto crossbar array blocks



Neuromorphic chip: Spintronics

- magnetic nano-neurons implemented as spin torque oscillators
- synapses implemented as radiowaves



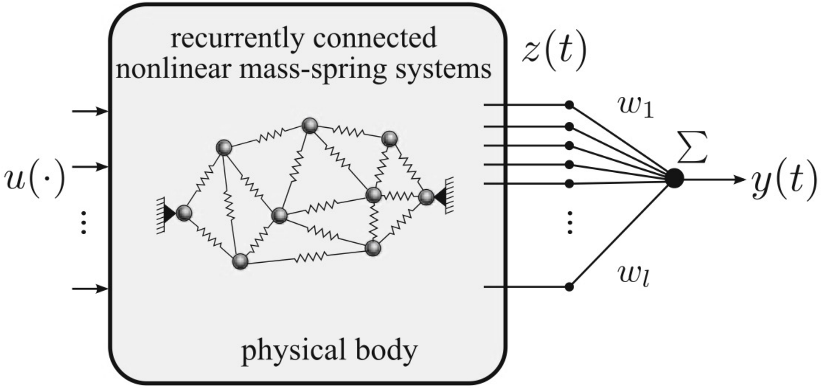


Torrejon, Jacob, et al. "Neuromorphic computing with nanoscale spintronic oscillators." *Nature* 547.7664 (2017): 428-431.

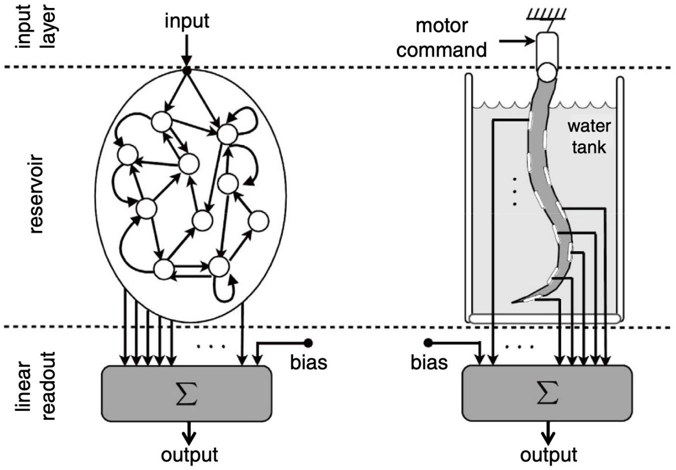
Locatelli, Nicolas, Vincent Cros, and Julie Grollier. "Spin-torque building blocks." *Nature materials* 13.1 (2014): 11-20.

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.

Conclusions

Conclusions

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