SPARSE RESERVOIR COMPUTING

Eleni Vasilaki, University of Sheffield

WORK WITH

Luca Manneschi

Andrew Lin

Google DeepMind

EPSRC
Engineering and Physical Sciences Research Council

Manneschi et al (2021) IEEE Trans. Neural Networks & Learning Systems https://arxiv.org/abs/1907.08600,

LEARNING ASSOCIATIONS: THE MUSHROOM BODY

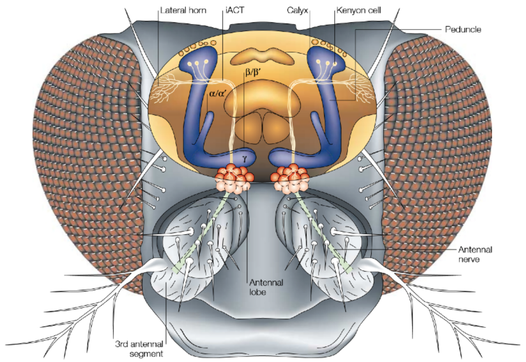


Image from M. Heisenberg, Nat. Rev. Neurosci., 2003

LEARNING ASSOCIATIONS: THE MUSHROOM BODY

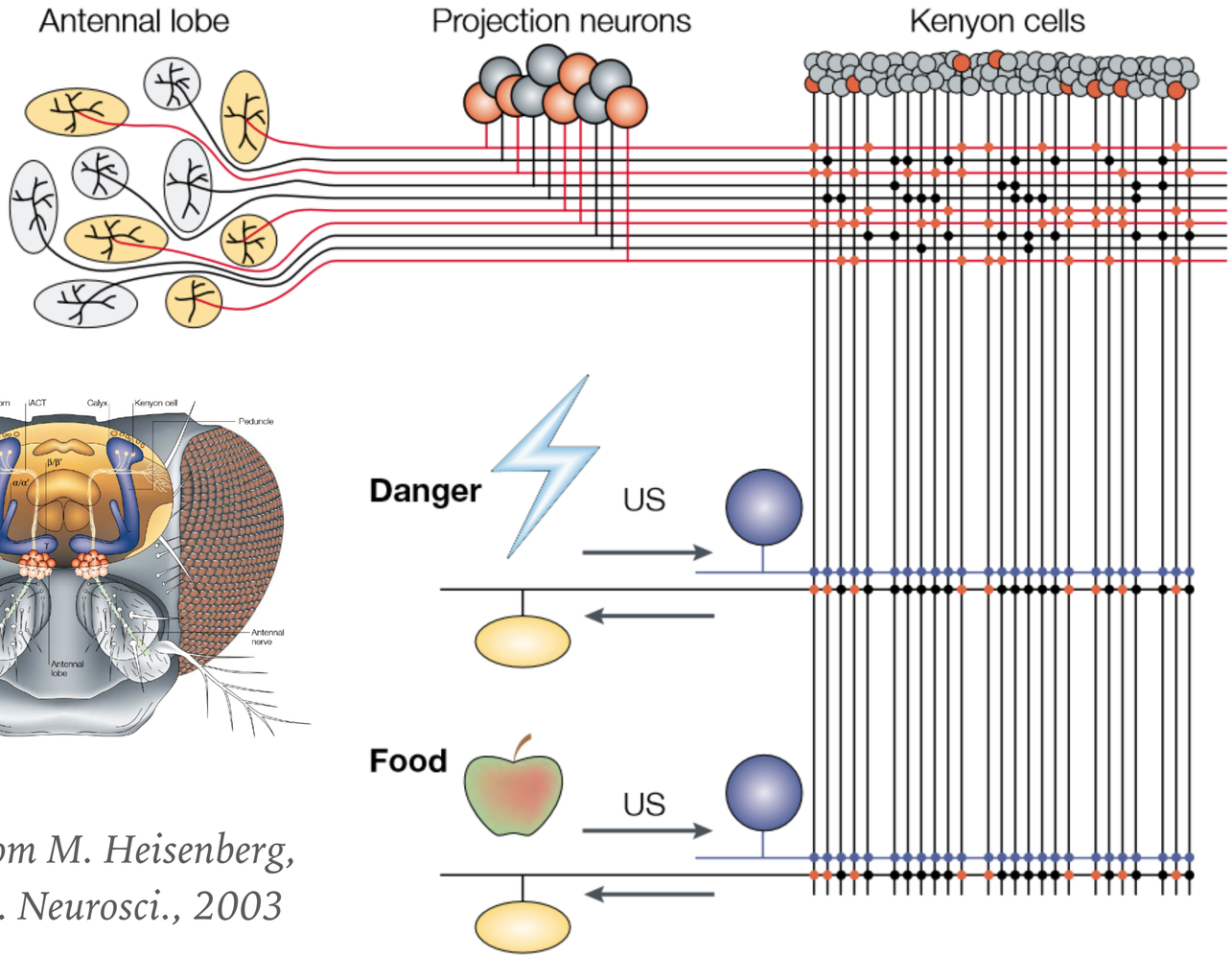
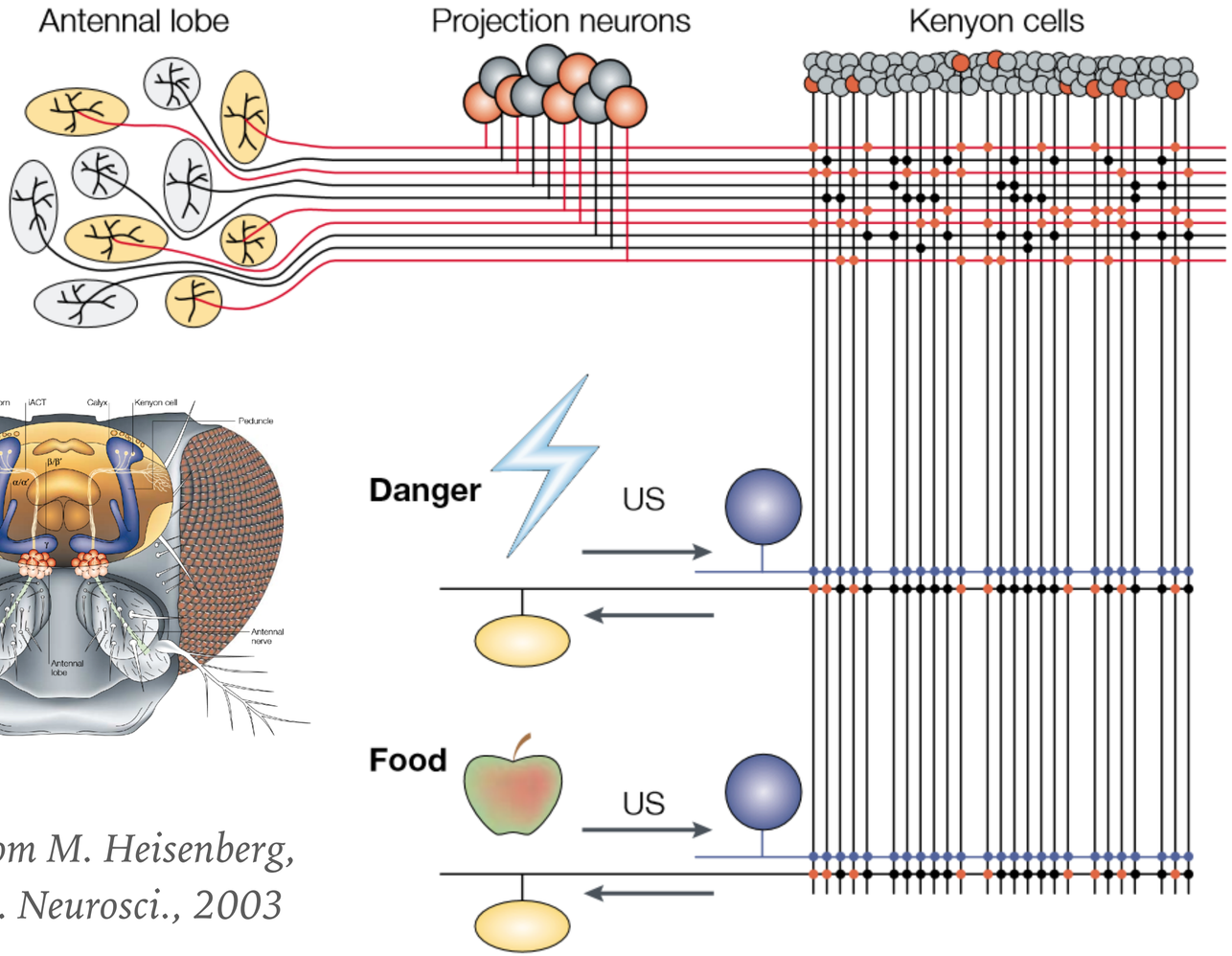


Image from M. Heisenberg, Nat. Rev. Neurosci., 2003

LEARNING ASSOCIATIONS: THE MUSHROOM BODY



Sparse representation

Image from M. Heisenberg, Nat. Rev. Neurosci., 2003

LEARNING ASSOCIATIONS: THE MUSHROOM BODY

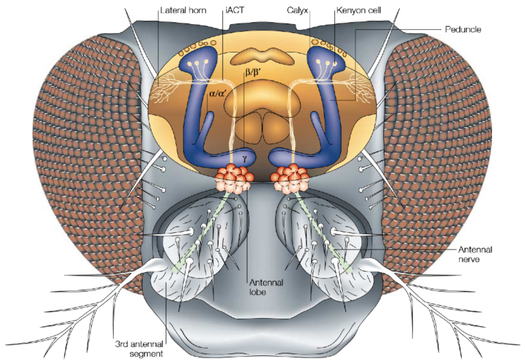
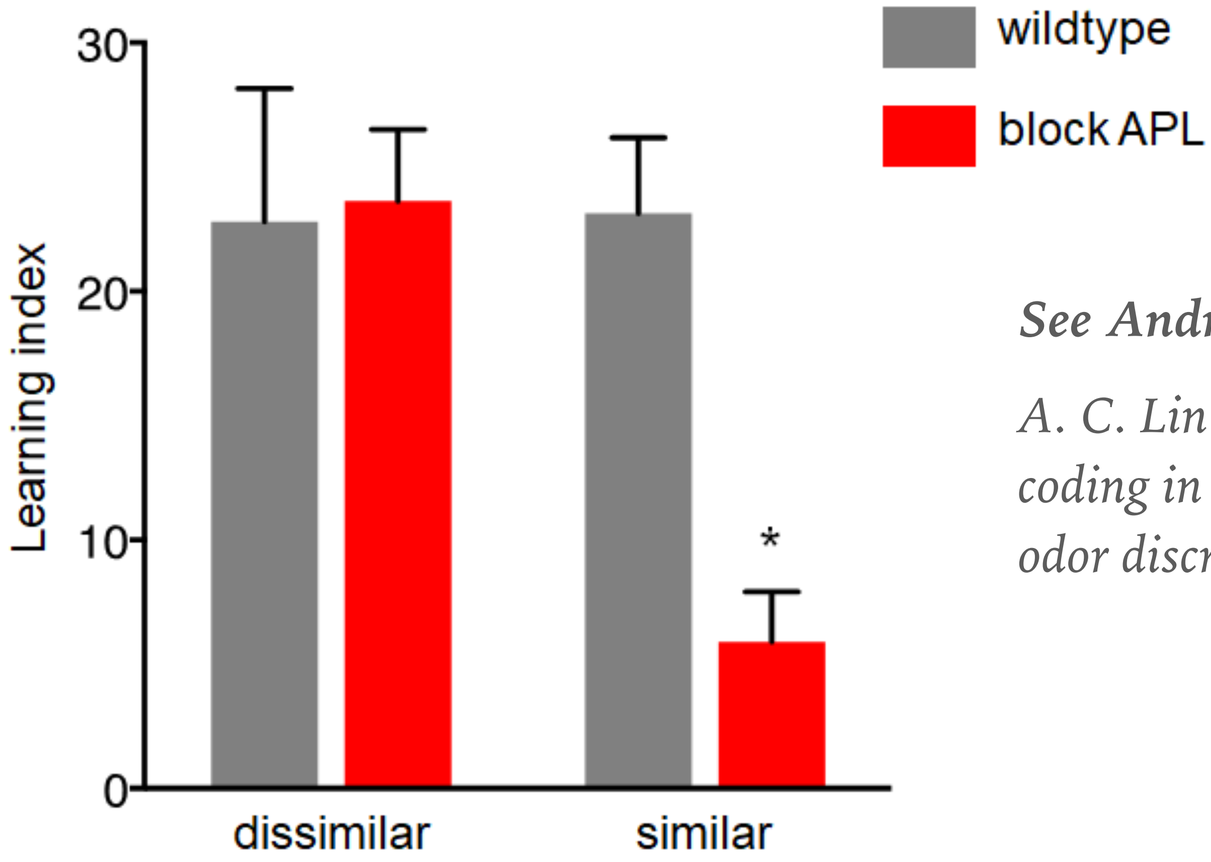


Image from M. Heisenberg, Nat. Rev. Neurosci., 2003

Sparse representation

APL

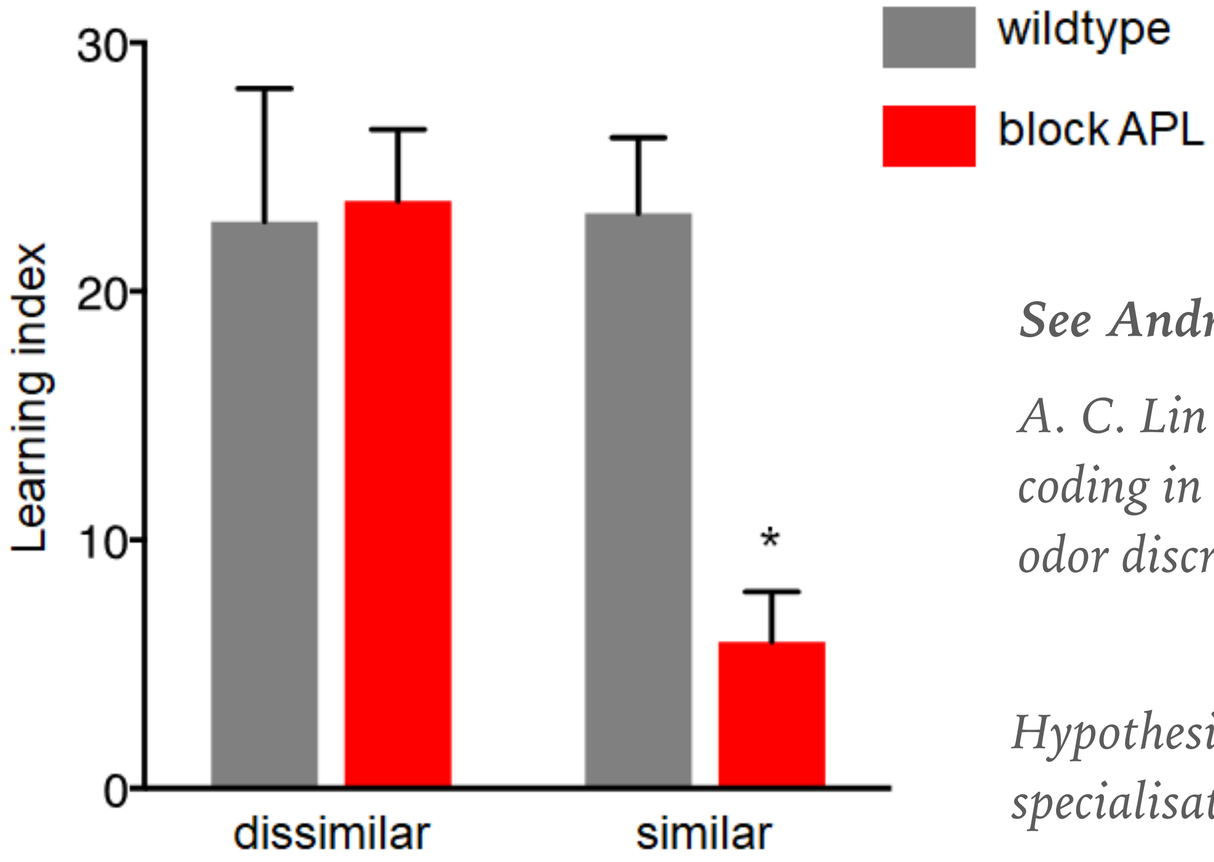
BLOCKING APL PREVENTS LEARNED DISCRIMINATION OF SIMILAR ODOURS BUT NOT DISSIMILAR ODOURS



See Andrew's work:

A. C. Lin et al "Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination," Nature Neurosci.,2014.

BLOCKING APL PREVENTS LEARNED DISCRIMINATION OF SIMILAR ODOURS BUT NOT DISSIMILAR ODOURS

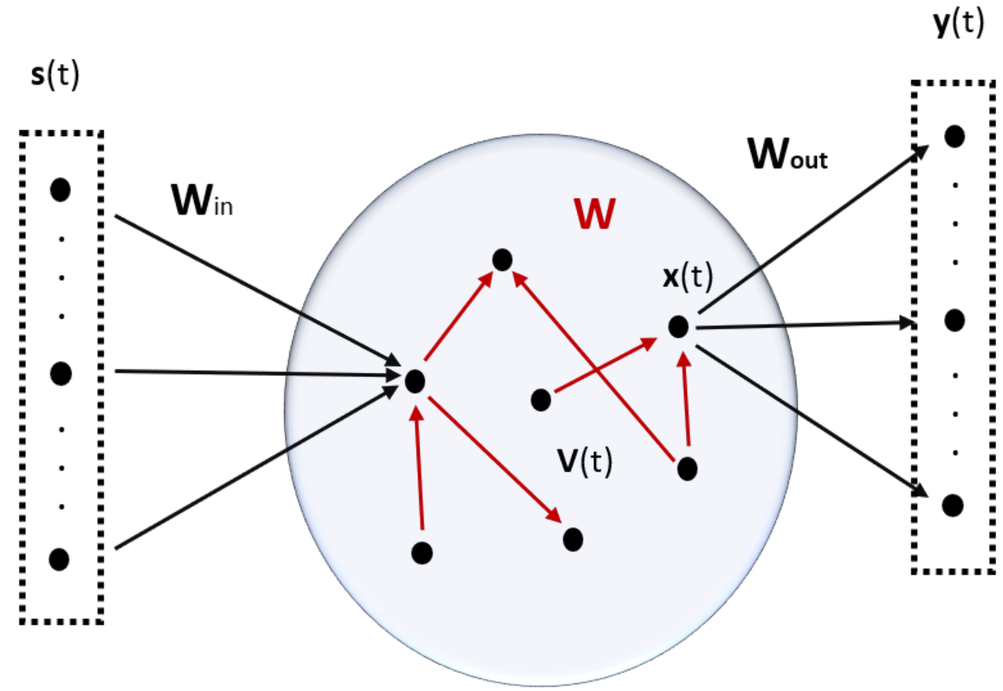


See Andrew's work:

A. C. Lin et al "Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination," Nature Neurosci.,2014.

Hypothesis: Sparcity introduces neuronal specialisation

THE MUSHROOM BODY AS RESERVOIR

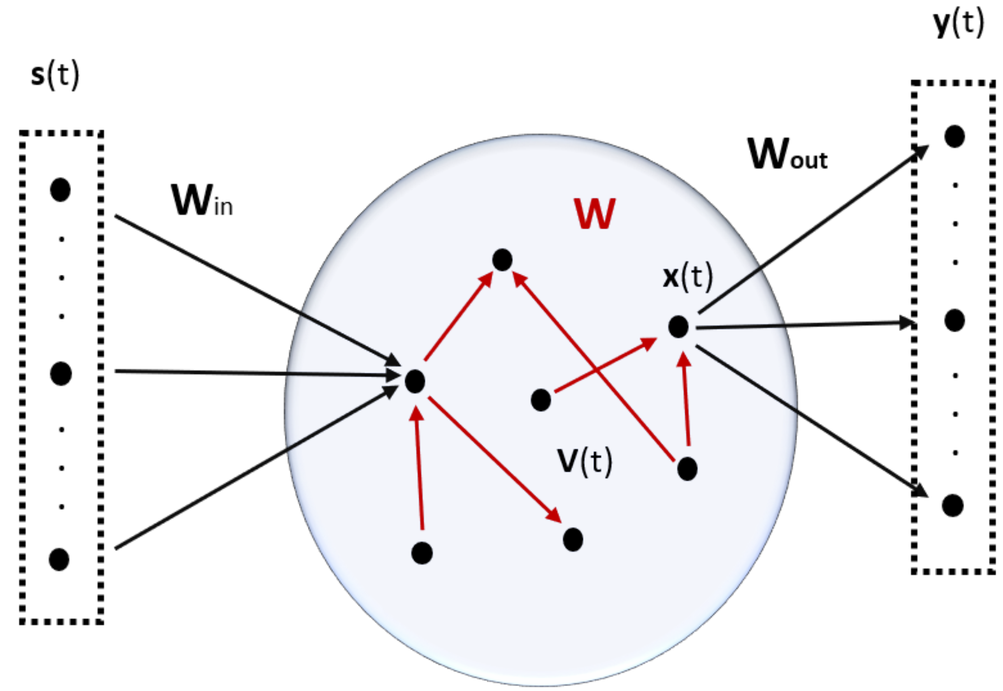


input *random sparse matrix* W *output*

H. Jaeger, GMD-Report (2001), Neural Networks (2007)

W. Maass et al, Neural Comput. (2002) W. Maass & H. Markram, JCSS (2004)

THE MUSHROOM BODY AS RESERVOIR



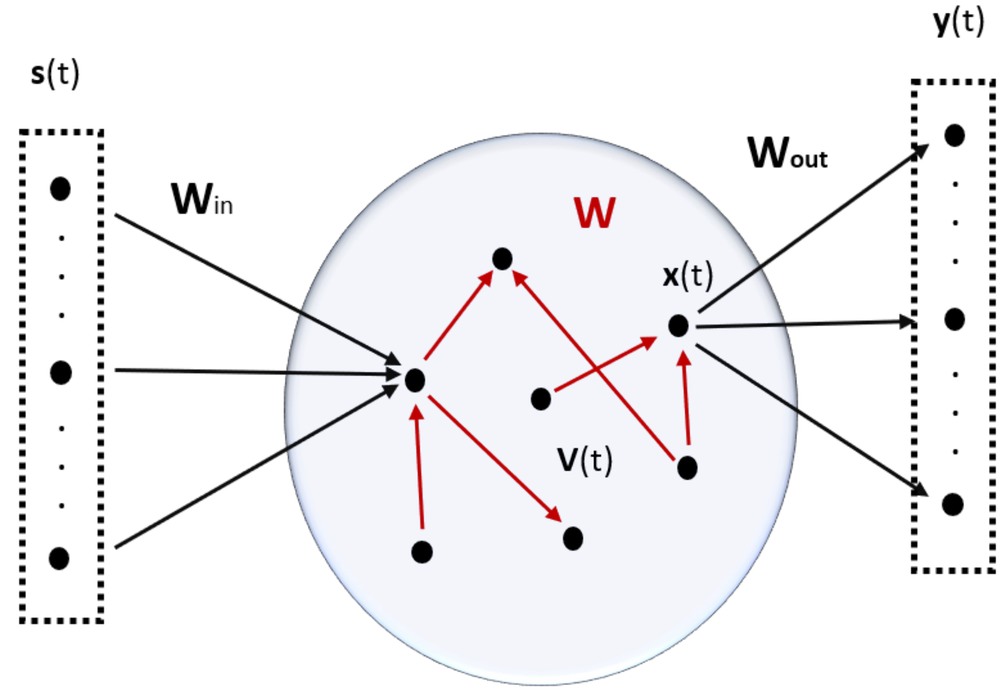
input *random sparse matrix* W *output*

Exploit randomness to achieve a rich representation

H. Jaeger, GMD-Report (2001), Neural Networks (2007)

W. Maass et al, Neural Comput. (2002) W. Maass & H. Markram, JCSS (2004)

THE MUSHROOM BODY AS RESERVOIR



input *random sparse matrix* W *output*

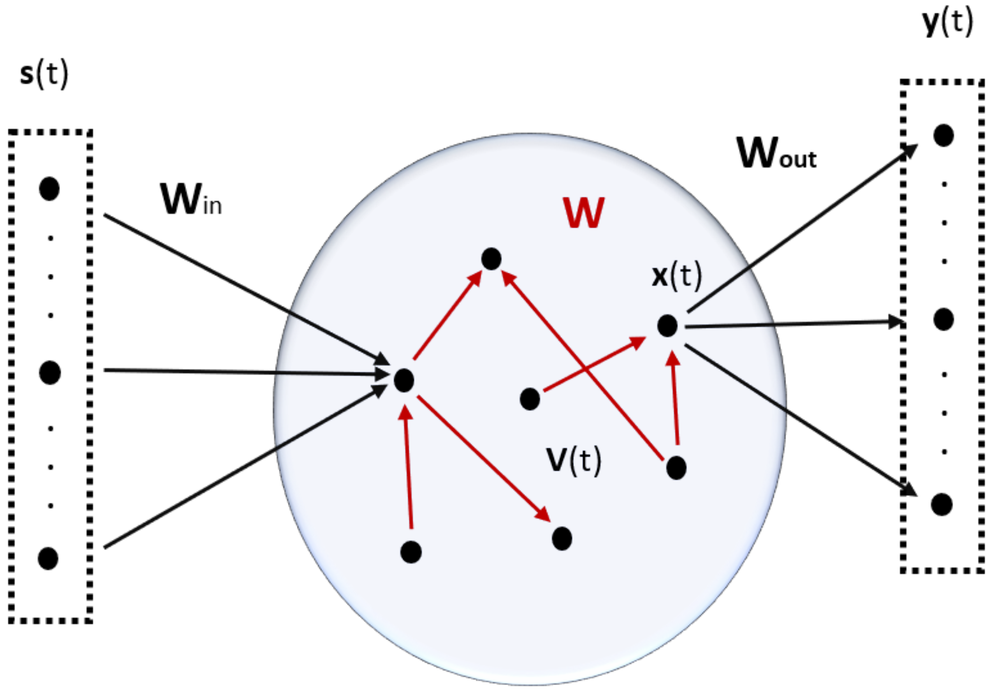
Exploit randomness to achieve a rich representation

Learn only output matrix W_{out}

H. Jaeger, GMD-Report (2001), Neural Networks (2007)

W. Maass et al, Neural Comput. (2002) W. Maass & H. Markram, JCSS (2004)

RESERVOIR COMPUTING



$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left(\gamma \mathbf{W}_{\mathbf{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right)$$

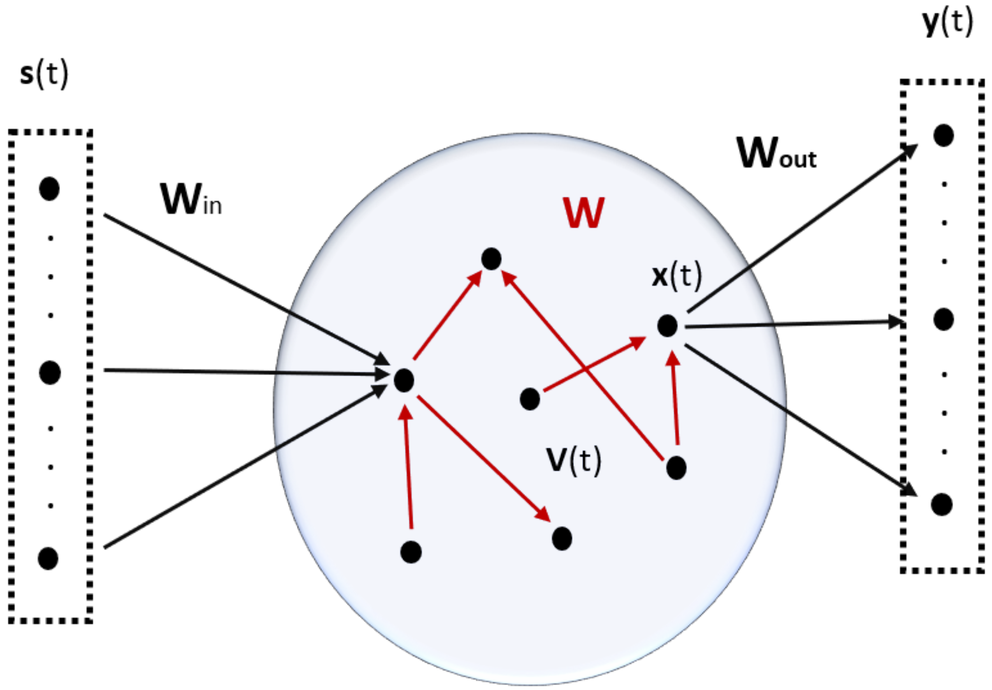
$$\alpha = \frac{\delta t}{\tau}$$

γ : *gain factor*

$$0 < \rho < 1$$

Manneschi et al (2021) Frontiers in Applied Mathematics and Statistics

RESERVOIR COMPUTING



$$\alpha = \frac{\delta t}{\tau}$$

γ : *gain factor*

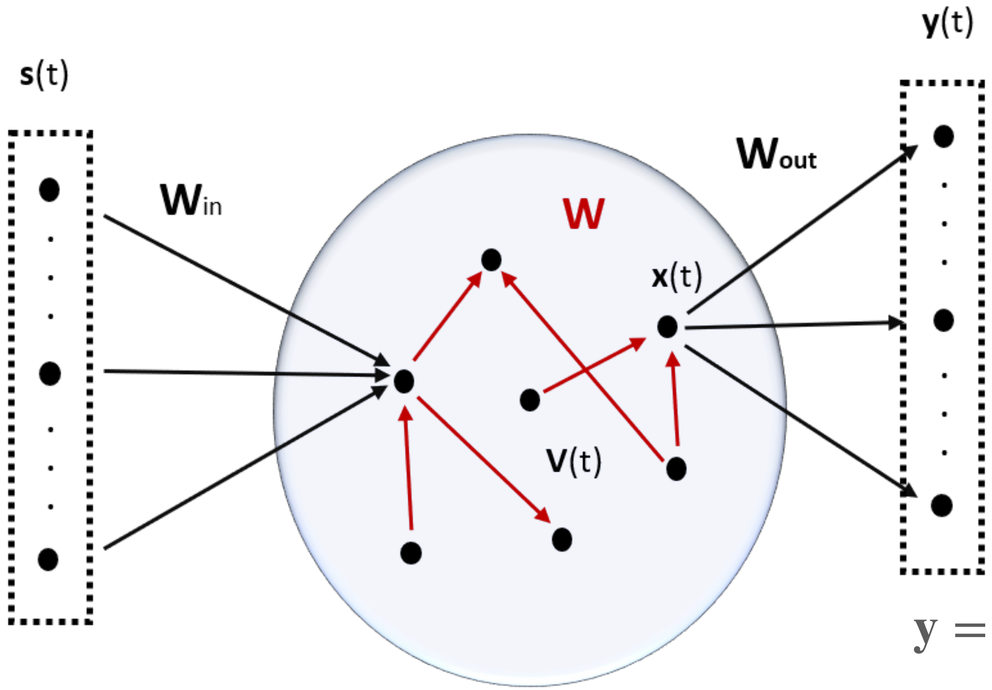
$$0 < \rho < 1$$

$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left(\gamma \mathbf{W}_{\text{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right)$$

$$\mathbf{x}(t) = \mathbf{V}(\mathbf{t})$$

Manneschi et al (2021) Frontiers in Applied Mathematics and Statistics

RESERVOIR COMPUTING



$$\mathbf{y} = \mathbf{W}_{out}^{T}\mathbf{x}$$

$$\alpha = \frac{\delta t}{\tau}$$

γ : *gain factor*

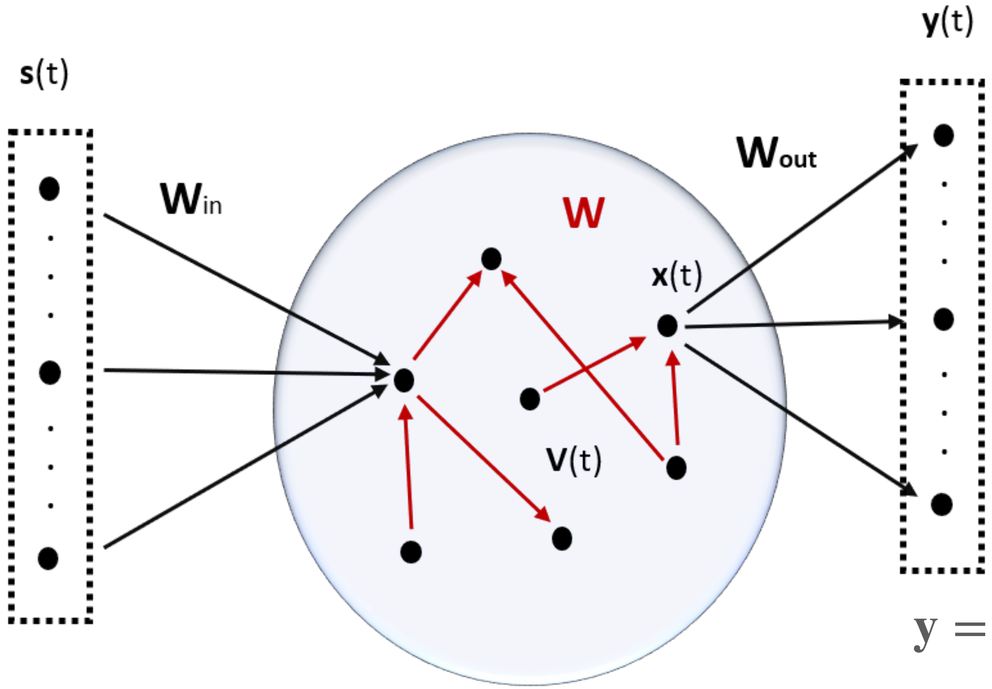
$$0 < \rho < 1$$

$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left(\gamma \mathbf{W}_{\mathbf{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right)$$

$$\mathbf{x}(t) = \mathbf{V}(\mathbf{t})$$

Manneschi et al (2021) Frontiers in Applied Mathematics and Statistics

RESERVOIR COMPUTING



$$\mathbf{y} = \mathbf{W}_{out}^{T}\mathbf{x}$$

Echo state property:
Eigenvalues of $\mathbf{W}$
inside unit circle

$$\alpha = \frac{\delta t}{\tau}$$

γ : *gain factor*

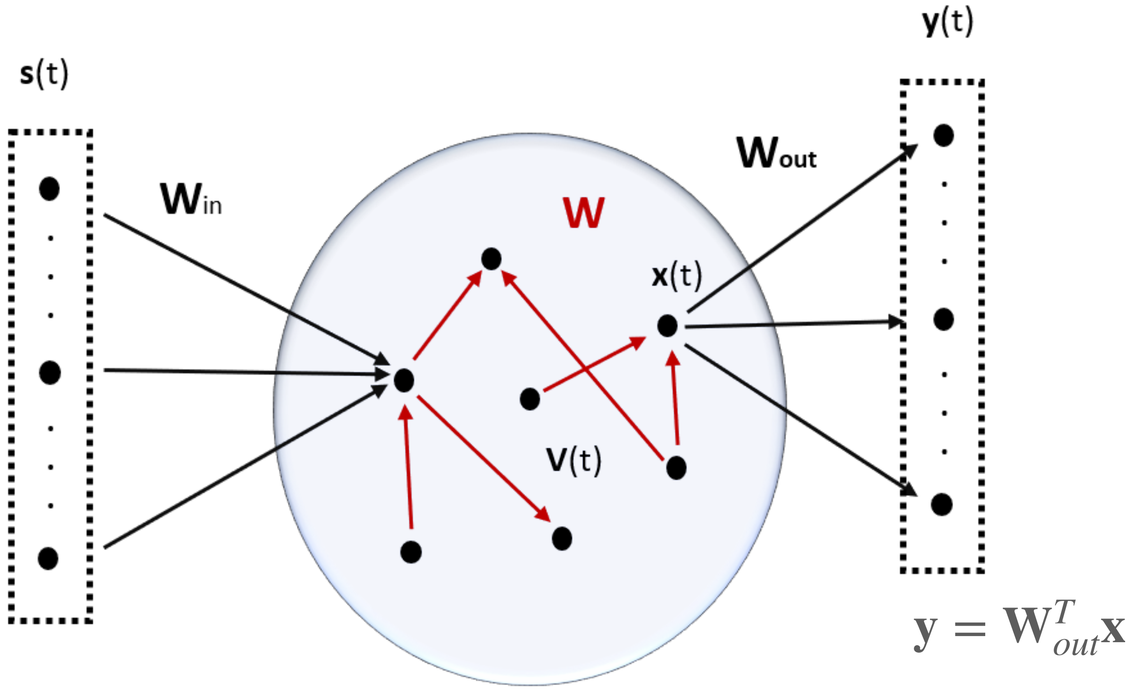
$$0 < \rho < 1$$

$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left(\gamma \mathbf{W}_{\mathbf{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right)$$

$$\mathbf{x}(t) = \mathbf{V}(\mathbf{t})$$

Manneschi et al (2021) Frontiers in Applied Mathematics and Statistics

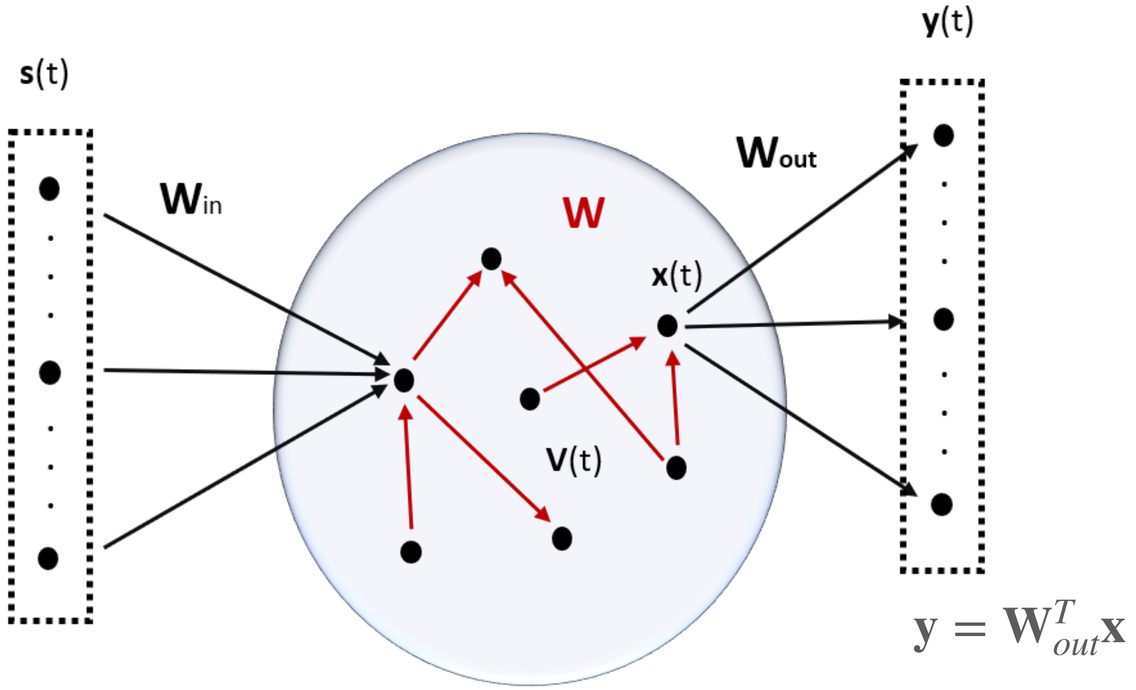
SPARSE RESERVOIR COMPUTING (SPARCE)



$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left(\gamma \mathbf{W}_{\text{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right)$$

Manneschi et al (2021) IEEE TNNLS
https://arxiv.org/abs/1907.08600

SPARSE RESERVOIR COMPUTING (SPARCE)

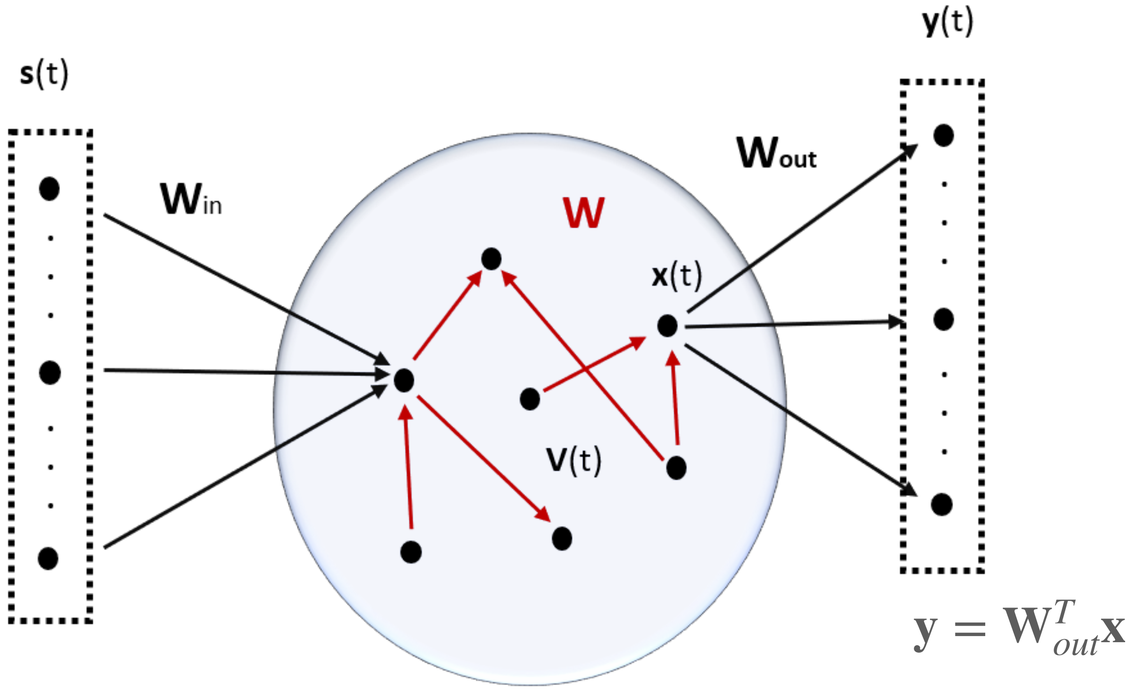


$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left(\gamma \mathbf{W}_{\text{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right)$$

$$x_i(t) = sign\left(V_i(t)\right) relu\left[\,|\,V_i(t)\,| - \theta_i\right]$$

Manneschi et al (2021) IEEE TNNLS
https://arxiv.org/abs/1907.08600

SPARSE RESERVOIR COMPUTING (SPARCE)



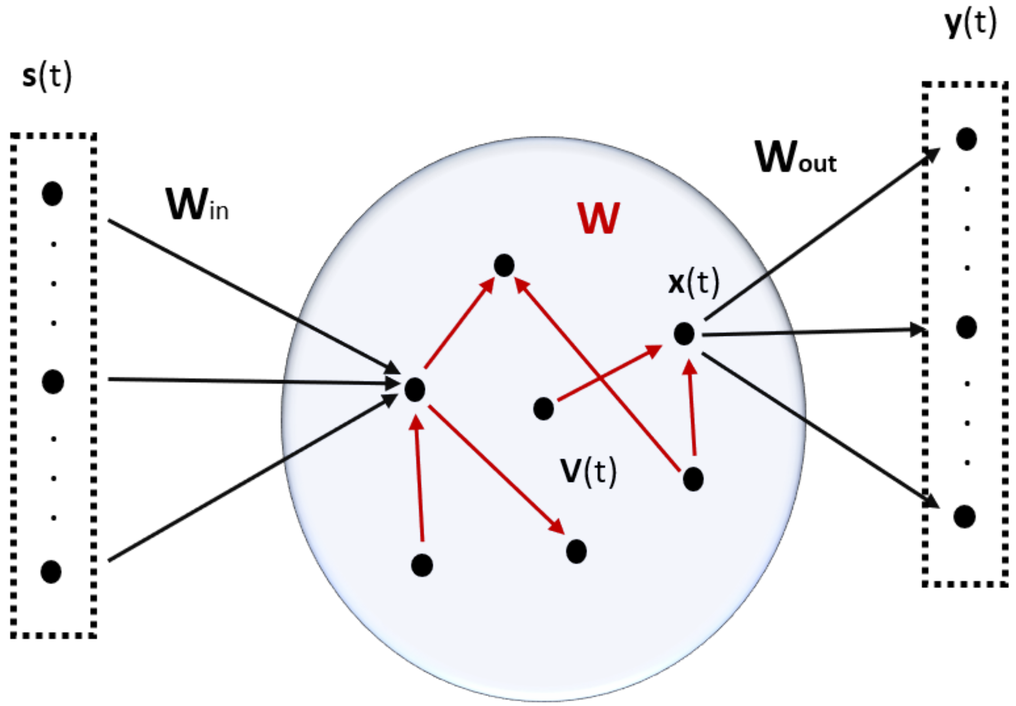
$$relu(x) = max(0,x)$$

$$\mathbf{V}(t+\delta t) = (1-\alpha)\mathbf{V}(t) + \alpha f\left(\gamma \mathbf{W}_{\text{in}}\mathbf{s}(t) + \rho \mathbf{W}\mathbf{V}(t)\right)$$

$$x_i(t) = sign\left(V_i(t)\right) relu\left[\,|\,V_i(t)\,| - \theta_i\right]$$

Manneschi et al (2021) IEEE TNNLS
https://arxiv.org/abs/1907.08600

READ-OUT WEIGHTS UPDATE RULE



$$E = \frac{1}{2}\sum_{j}\left[y_j^t - y_j\right]^2$$

$$\Delta W_{ji}^{out} = -\eta_W \frac{\partial E}{\partial W_{ji}^{out}} = \eta_w \left[y_j^t - y_j\right] x_i(t)$$

On line setting: (more) biologically plausible, avoid inversion of large matrices.

LEARNING THE READ-OUT THRESHOLDS

$$x_i = sign\left(V_i\right) relu\Big[\,|V_i| - \theta_i\Big]$$

Gradient descent, local thresholds θ_i

LEARNING THE READ-OUT THRESHOLDS

$$x_i = sign\left(V_i\right) relu\Big[\,|V_i| - \theta_i\Big]$$

Gradient descent, local thresholds θ_i

$$E = \frac{1}{2}\sum_j \left[y_j^t - \sum_l W_{jl}^{out} sign\left(V_l\right) relu\Big[\,|V_l| - \theta_l\Big]\right]^2$$

LEARNING THE READ-OUT THRESHOLDS

$$x_i = sign\left(V_i\right) relu\Big[\,|V_i| - \theta_i\Big]$$

Gradient descent, local thresholds θ_i

$$E = \frac{1}{2}\sum_j \Big[\underline{y_j^t} - \sum_l W_{jl}^{out} sign\left(V_l\right) relu\Big[\,|V_l| - \theta_l\Big]\Big]^2$$

Target

LEARNING THE READ-OUT THRESHOLDS

$$x_i = sign\left(V_i\right) relu\left[\left|V_i\right| - \theta_i\right]$$

Gradient descent, local thresholds θ_i

$$E = \frac{1}{2}\sum_j \left[y_j^t - \sum_l W_{jl}^{out} sign\left(V_l\right) relu\left[\left|V_l\right| - \theta_l\right]\right]^2$$

Target

Output y_j

LEARNING THE READ-OUT THRESHOLDS

$$x_i = sign\left(V_i\right) relu\left[\,|V_i| - \theta_i\right]$$

Gradient descent, local thresholds θ_i

$$E = \frac{1}{2}\sum_j \left[y_j^t - \sum_l W_{jl}^{out} sign\left(V_l\right) relu\left[\,|V_l| - \theta_l\right]\right]^2$$

Target

Output y_j

$$\Delta\theta_i = -\eta\frac{\partial E}{\partial \theta_i}$$

LEARNING THE READ-OUT THRESHOLDS

$$x_i = sign\left(V_i\right) relu\Big[\,|V_i| - \theta_i\Big]$$

Gradient descent, local thresholds θ_i

$$E = \frac{1}{2}\sum_j \Big[y_j^t - \sum_l W_{jl}^{out} sign\left(V_l\right) relu\big[\,|V_l| - \theta_l\big]\Big]^2$$

Target

Output y_j

$$\Delta\theta_i = -\eta\frac{\partial E}{\partial \theta_i} = -\eta\sum_j \left[y_j^t - y_j\right] W_{ji}^{out} sign\left(V_i\right) H\left(|V_i| - \theta_i\right)$$

LEARNING THE READ-OUT THRESHOLDS

$$x_i = sign\left(V_i\right) relu\Big[\,|V_i| - \theta_i\Big]$$

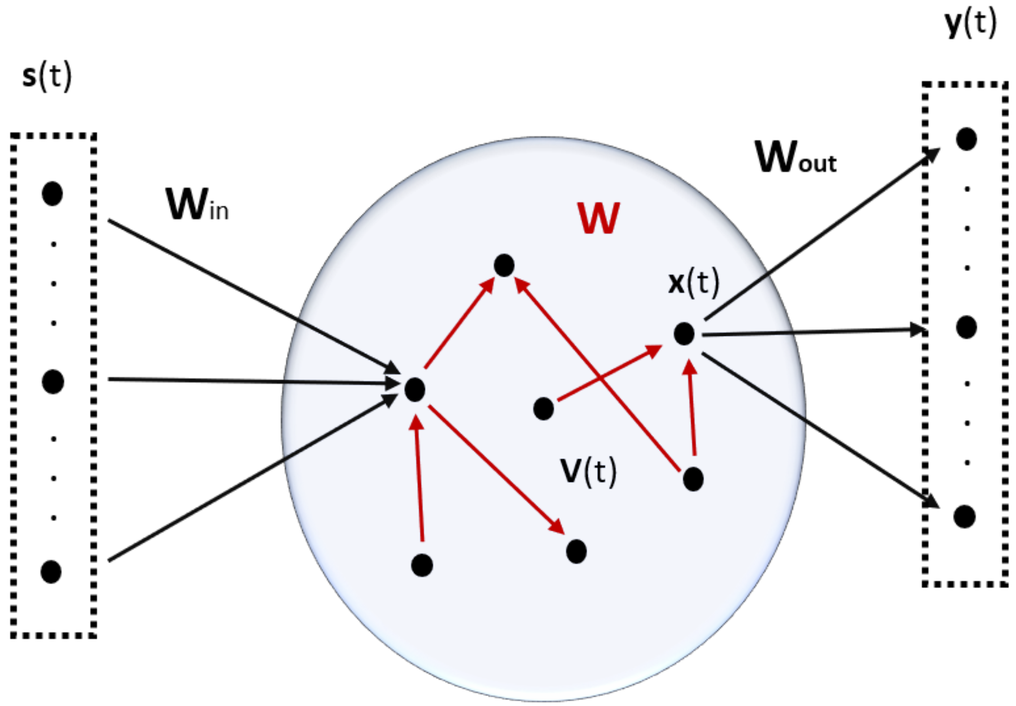
Gradient descent, local thresholds θ_i

$$E = \frac{1}{2}\sum_j \Big[\underbrace{y_j^t}_{\text{Target}} - \underbrace{\sum_l W_{jl}^{out} sign\left(V_l\right) relu\Big[\,|V_l| - \theta_l\Big]}_{\text{Output } y_j}\Big]^2$$

$$\Delta\theta_i = -\eta\frac{\partial E}{\partial\theta_i} = -\eta\sum_j \Big[y_j^t - y_j\Big] W_{ji}^{out} sign\left(V_i\right) H\big(\,|V_i| - \theta_i\big)$$

$$= -\eta\sum_j \Big[y_j^t - y_j\Big] W_{ji}^{out} sign\left(x_i\right)$$

THRESHOLDS UPDATE RULE

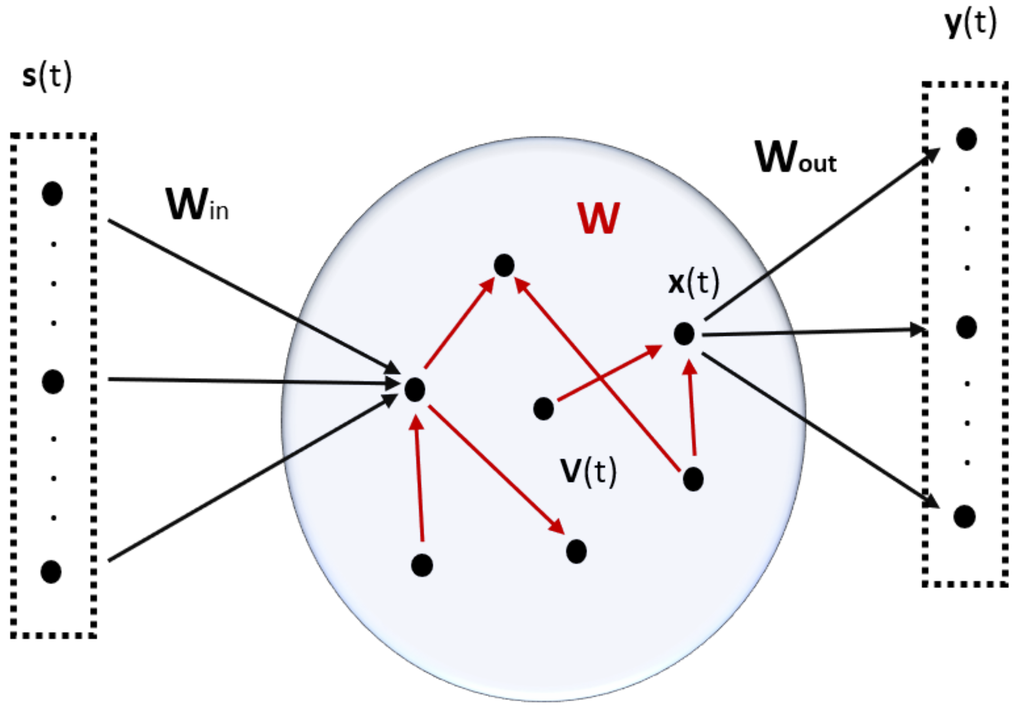


$$E = \frac{1}{2}\sum_j \left[y_j^t - y_j\right]^2$$

$$\Delta\theta_i = -\eta_\theta \sum_j \left[y_j^t - y_j\right] W_{ji}^{out} sign\left(x_i\right)$$

On line setting: (more) biologically plausible, avoid inversion of large matrices.

THRESHOLDS UPDATE RULE



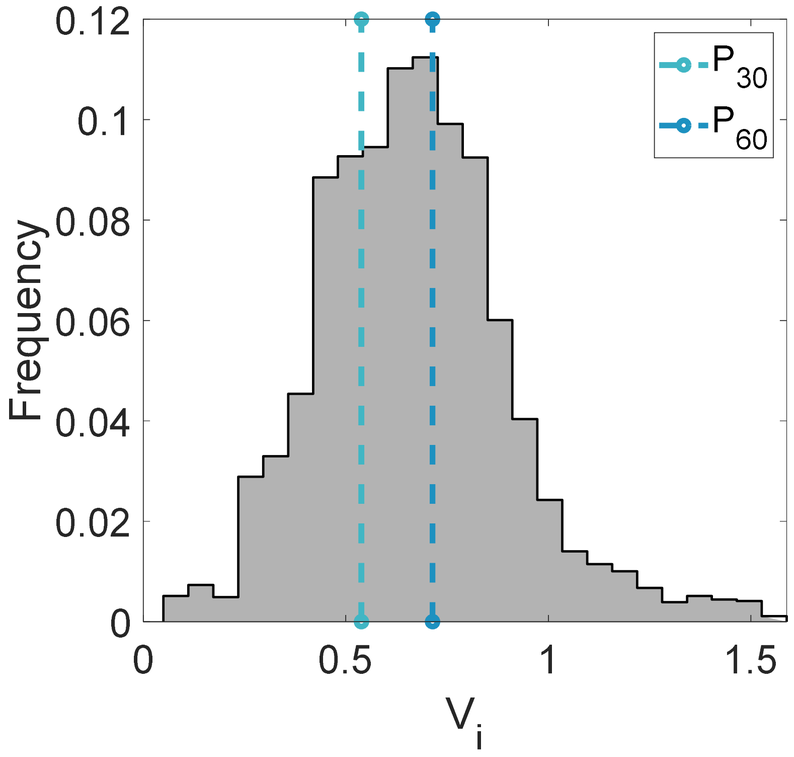
$$E = \frac{1}{2}\sum_j \left[y_j^t - y_j\right]^2$$

$$\Delta\theta_i = -\eta_\theta \sum_j \left[y_j^t - y_j\right] W_{ji}^{out} sign\left(x_i\right)$$

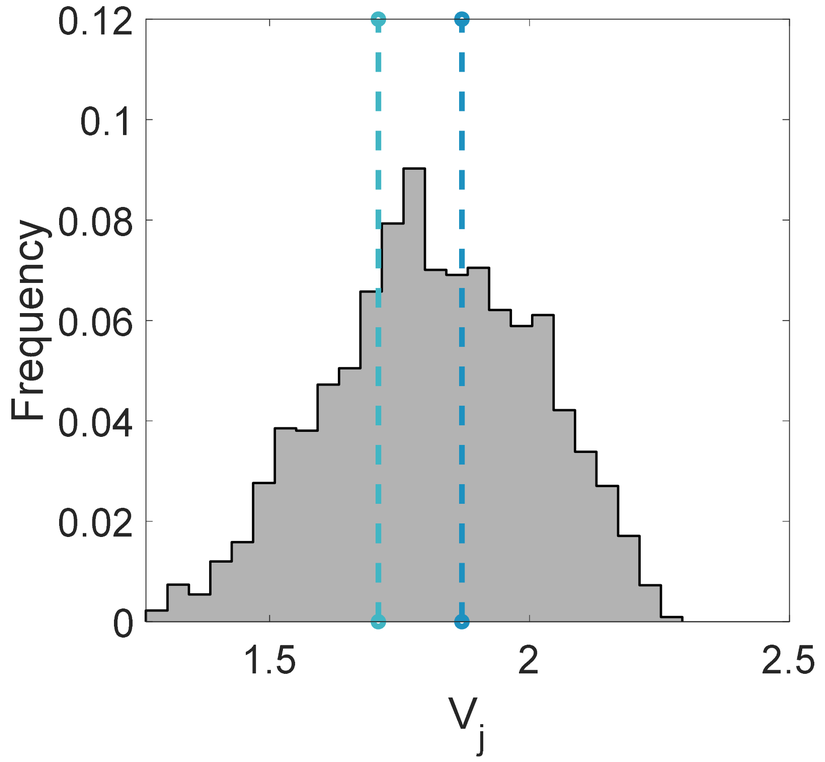
On line setting: (more) biologically plausible, avoid inversion of large matrices.

“DEMOCRATIC” INITIALISATION OF THRESHOLDS

$\theta_i(0)$



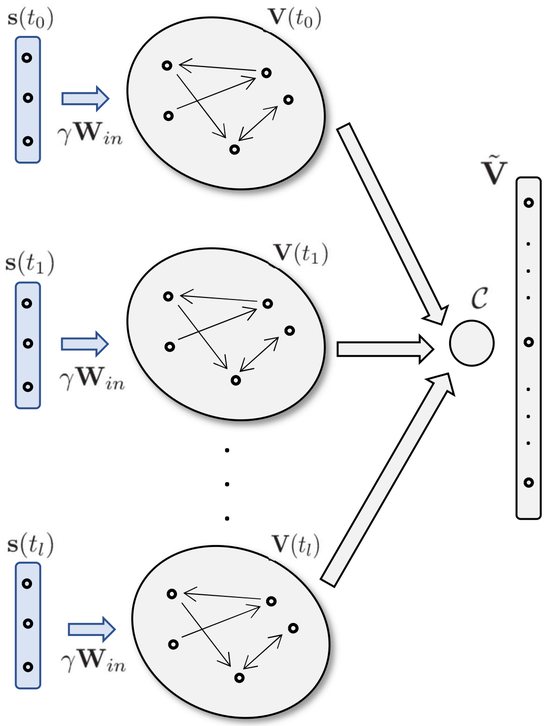
$\theta_j(0)$



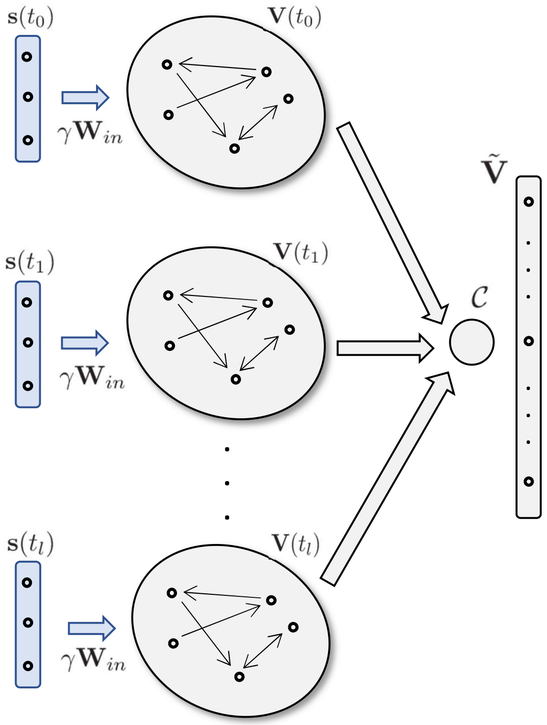
Define the initial sparsity level by the distribution of the neuron activity.

Can be learned on-line.

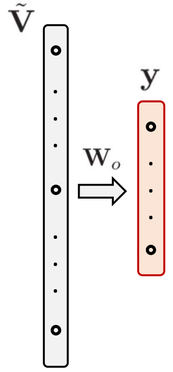
SPARCE SCHEME



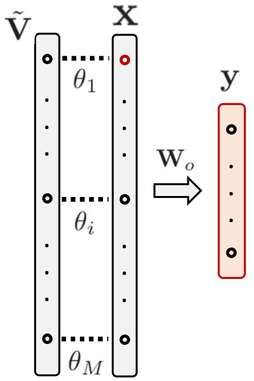
SPARCE SCHEME



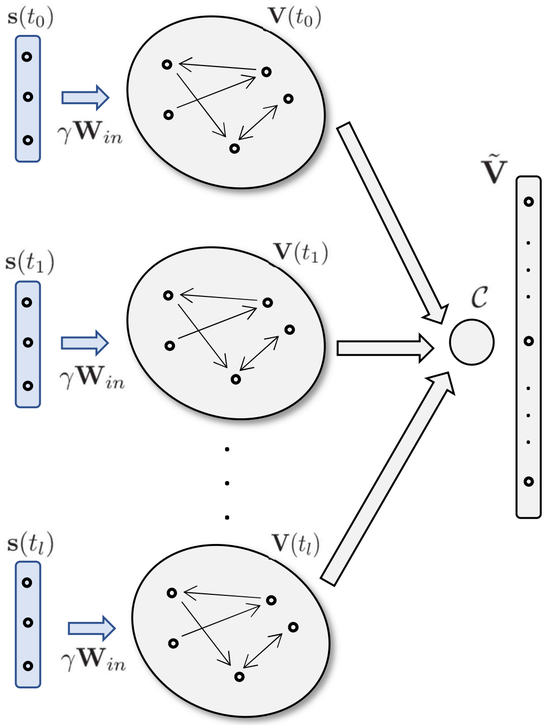
ESN

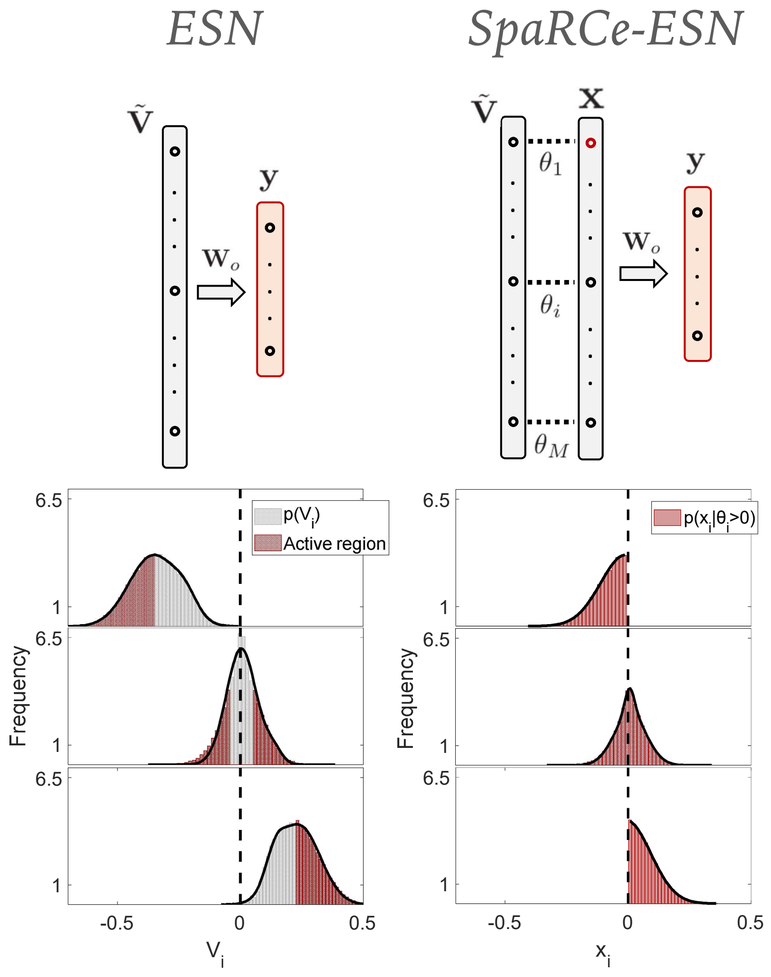


SpaRCe-ESN



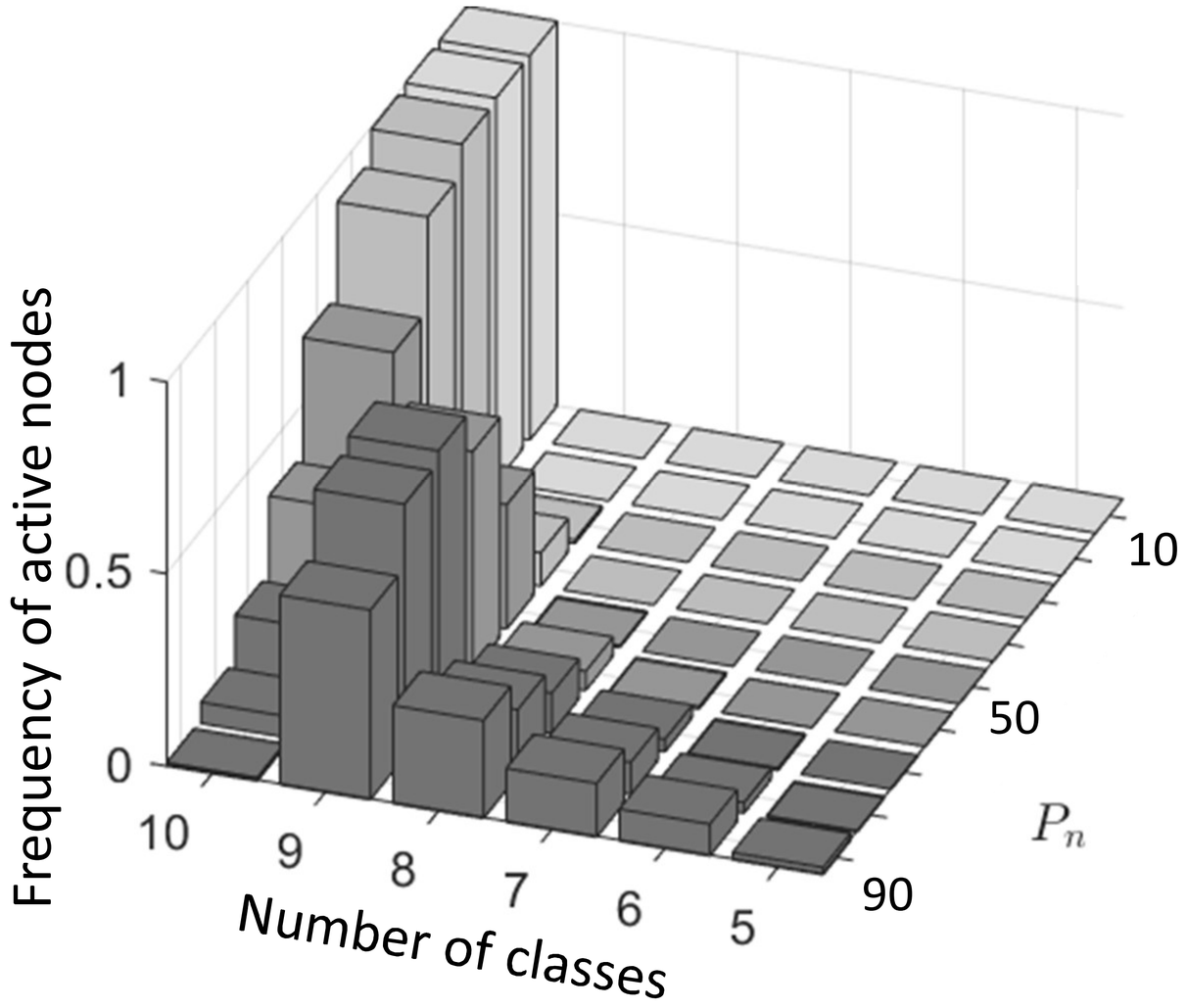
SPARCE SCHEME





$$x_i(t) = sign\left(V_i(t)\right) relu\left[\,|\,V_i(t)\,| - \theta_i\right]$$

SPARSITY INTRODUCES NEURONAL SPECIALISATION



Example: column by column MNIST

ANALYSIS OF THRESHOLD LEARNING

$$\Delta\theta_i = -\eta\frac{\partial E}{\partial \theta_i} = -\eta\sum_j \left[y_j^t - y_j\right] W_{ji}^{out} sign\left(x_i\right)$$

ANALYSIS OF THRESHOLD LEARNING

$$\Delta\theta_i = -\eta\frac{\partial E}{\partial \theta_i} = -\eta\sum_j \left[y_j^t - y_j\right] W_{ji}^{out} sign\left(x_i\right)$$

$$= -\eta\sum_j \left[y_j^t - \sum_l W_{jl}^{out} x_l\right] W_{ji}^{out} sign\left(x_i\right)$$

ANALYSIS OF THRESHOLD LEARNING

$$\Delta\theta_i = -\eta\frac{\partial E}{\partial\theta_i} = -\eta\sum_j\left[y_j^t - y_j\right]W_{ji}^{out}sign\left(x_i\right)$$

$$= -\eta\sum_j\left[y_j^t - \sum_l W_{jl}^{out}x_l\right]W_{ji}^{out}sign\left(x_i\right)$$

$$= -\eta\sum_j y_j^t W_{ji}^{out}sign\left(x_i\right) + \eta\sum_j\sum_l W_{jl}^{out}x_l W_{ji}^{out}sign\left(x_i\right)$$

ANALYSIS OF THRESHOLD LEARNING

$$\Delta\theta_i = -\eta\frac{\partial E}{\partial \theta_i} = -\eta \sum_j \left[y_j^t - y_j\right] W_{ji}^{out} sign\left(x_i\right)$$

$$= -\eta \sum_j \left[y_j^t - \sum_l W_{jl}^{out} x_l\right] W_{ji}^{out} sign\left(x_i\right)$$

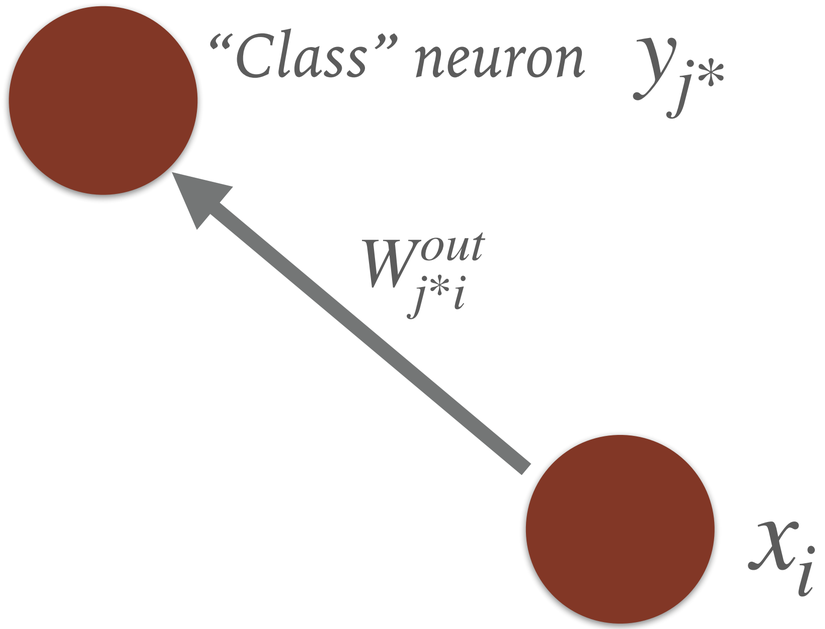
$$= -\eta \sum_j y_j^t W_{ji}^{out} sign\left(x_i\right) + \eta \sum_j \sum_l W_{jl}^{out} x_l W_{ji}^{out} sign\left(x_i\right)$$

one-hot encoding

$$= -\eta\ W_{j^*i}^{out} sign\left(x_i\right) + \eta \sum_j \sum_l W_{jl}^{out} x_l W_{ji}^{out} sign\left(x_i\right)$$

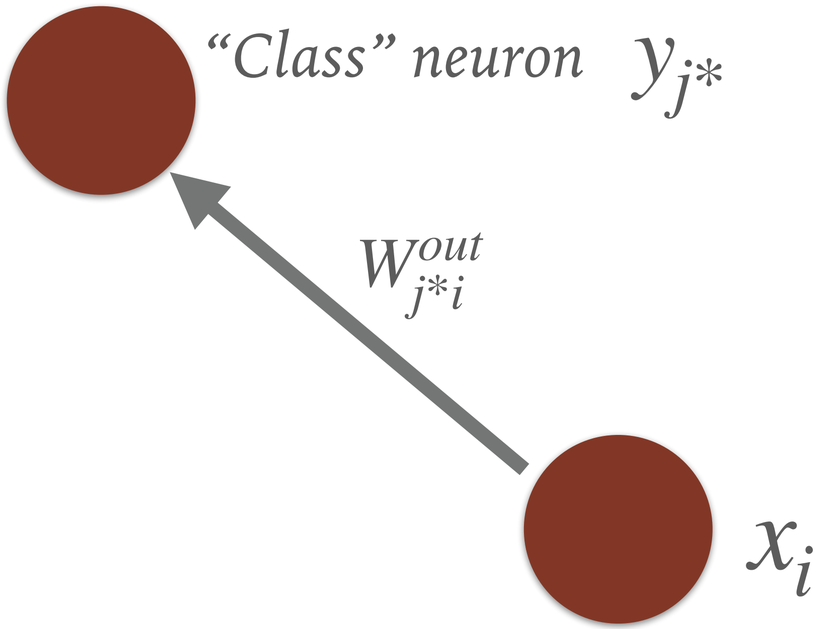
TWO FORCES DRIVE THRESHOLD LEARNING

$$\Delta\theta_i = -\eta_\theta \left(W^{out}_{j^*i} sign(x_i) \right) + \eta_\theta \sum_j \sum_l \left(W^{out}_{jl} x_l \right) \left(W^{out}_{ji} sign(x_i) \right)$$



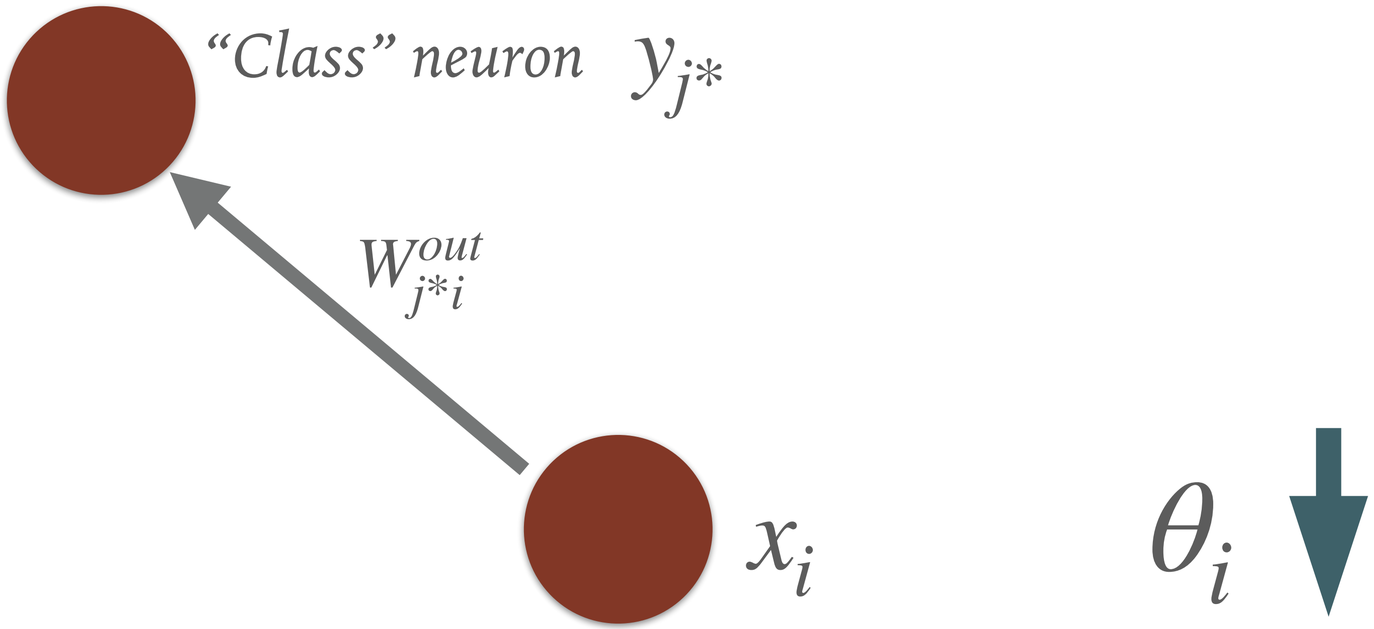
TWO FORCES DRIVE THRESHOLD LEARNING

$$\Delta\theta_i = -\eta_\theta \left(W^{out}_{j^*i} sign(x_i) \right) + \eta_\theta \sum_j \sum_l \left(W^{out}_{jl} x_l \right) \left(W^{out}_{ji} sign(x_i) \right)$$



TWO FORCES DRIVE THRESHOLD LEARNING

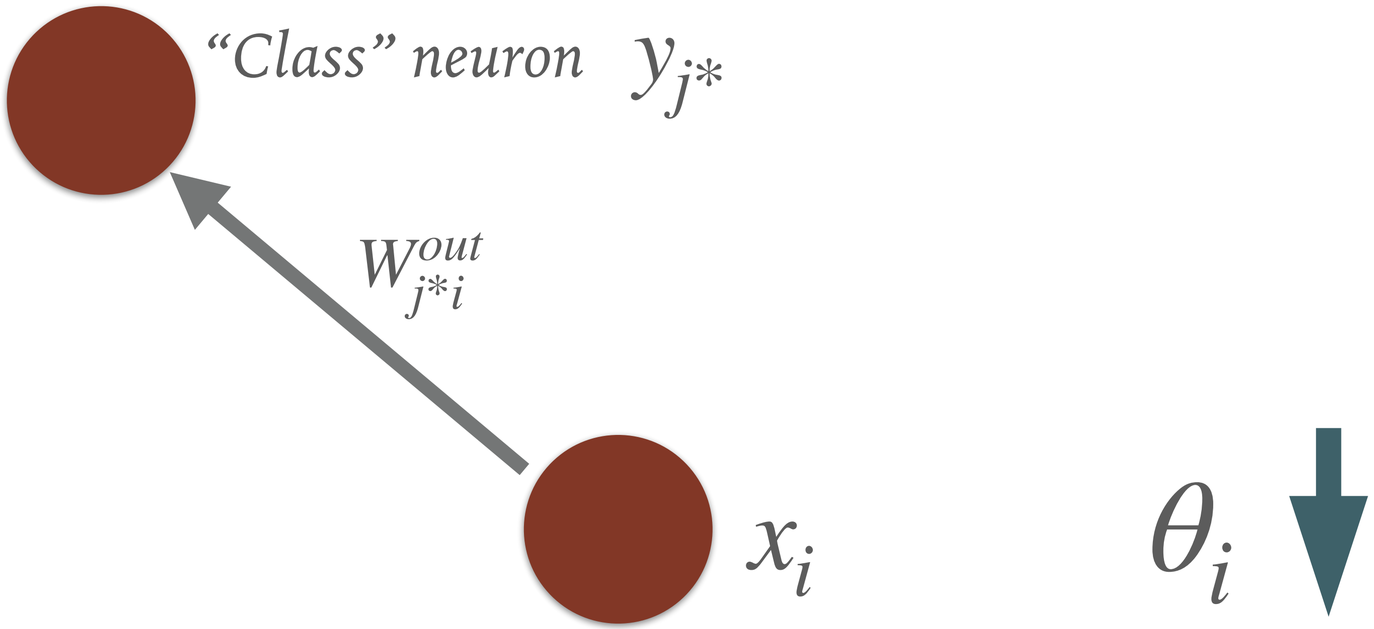
$$\Delta\theta_i = -\eta_\theta \left(W^{out}_{j^*i} sign(x_i) \right) + \eta_\theta \sum_j \sum_l \left(W^{out}_{jl} x_l \right) \left(W^{out}_{ji} sign(x_i) \right)$$



θ_i

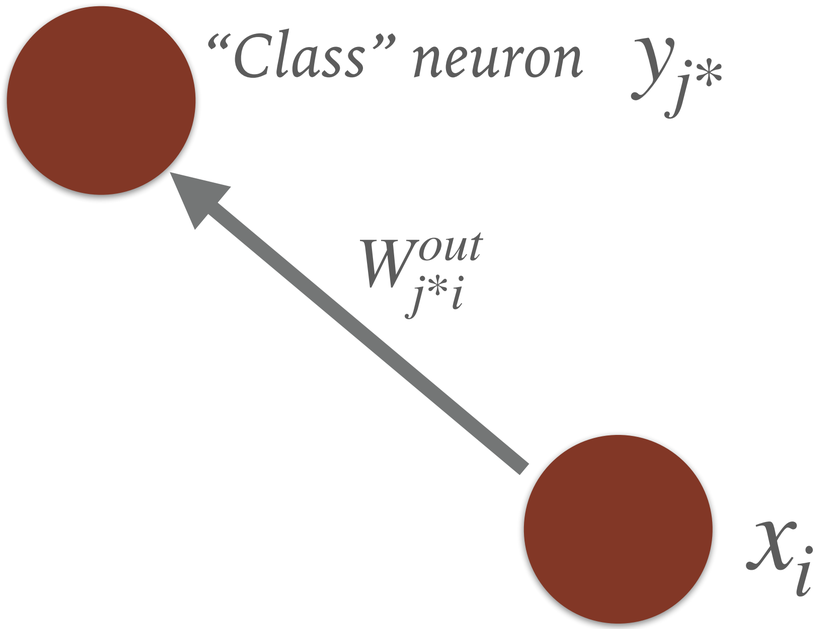
TWO FORCES DRIVE THRESHOLD LEARNING

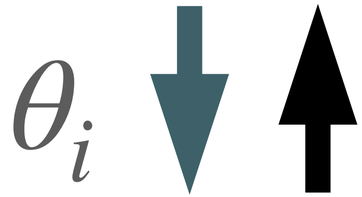
$$\Delta\theta_i = -\eta_\theta \left(W^{out}_{j^*i} sign(x_i) \right) + \eta_\theta \sum_j \sum_l \left(W^{out}_{jl} x_l \right) \left(W^{out}_{ji} sign(x_i) \right)$$



TWO FORCES DRIVE THRESHOLD LEARNING

$$\Delta\theta_i = -\eta_\theta \left(W^{out}_{j^*i} sign(x_i) \right) + \eta_\theta \sum_j \sum_l \left(W^{out}_{jl} x_l \right) \left(W^{out}_{ji} sign(x_i) \right)$$

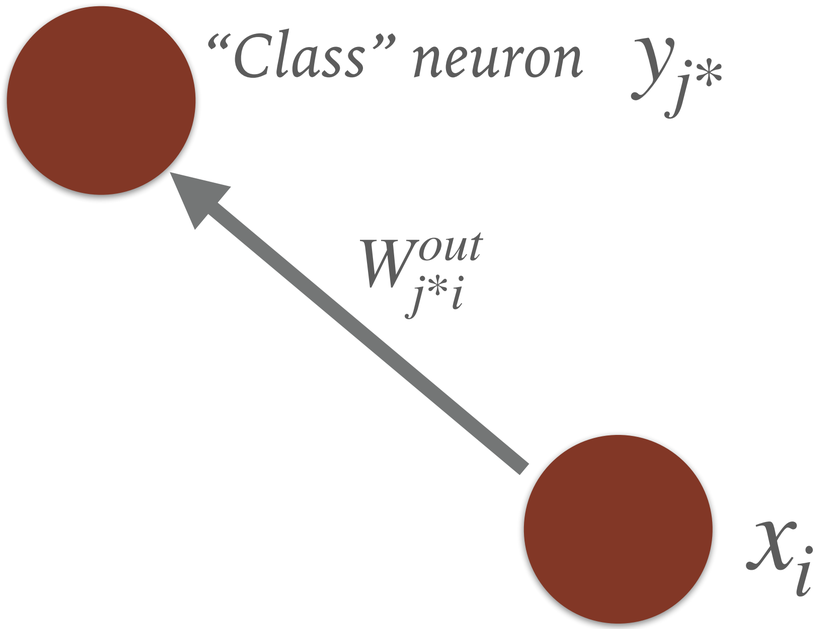


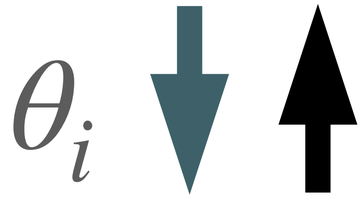


TWO FORCES DRIVE THRESHOLD LEARNING

$$\Delta\theta_i = -\eta_\theta \left(W^{out}_{j^*i} sign(x_i) \right) + \eta_\theta \sum_j \sum_l \left(W^{out}_{jl} x_l \right) \left(W^{out}_{ji} sign(x_i) \right)$$

reduces correlations

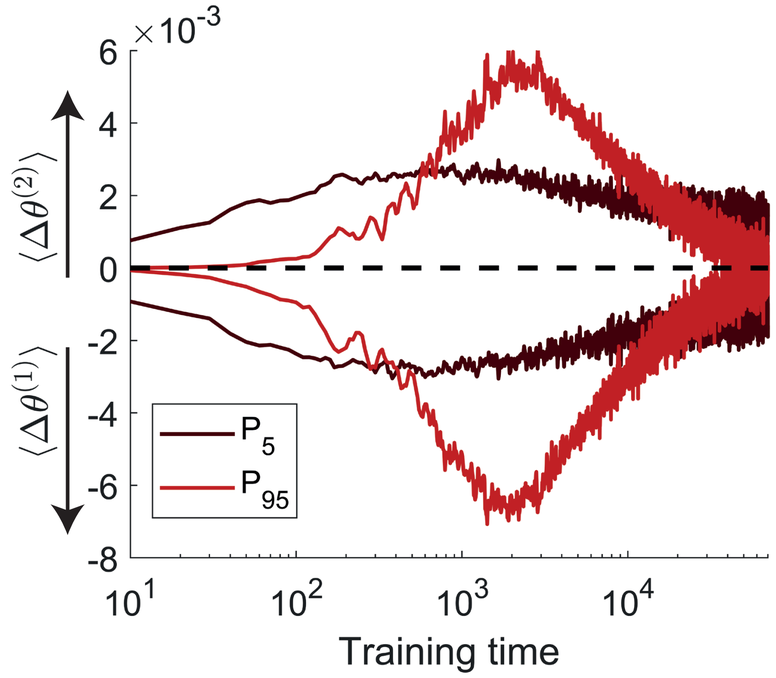




TWO FORCES DRIVE THRESHOLD LEARNING

$$\Delta\theta_i/\eta_\theta = -\,y^t_{j*}\left(W^{out}_{j*i}\,sign(x_i)\right) + \sum_j \sum_l \left(W^{out}_{jl}\,x_l\right)\left(W^{out}_{ji}\,sign(x_i)\right)$$

$\Delta\theta^{(2)}$ $\Delta\theta^{(1)}$

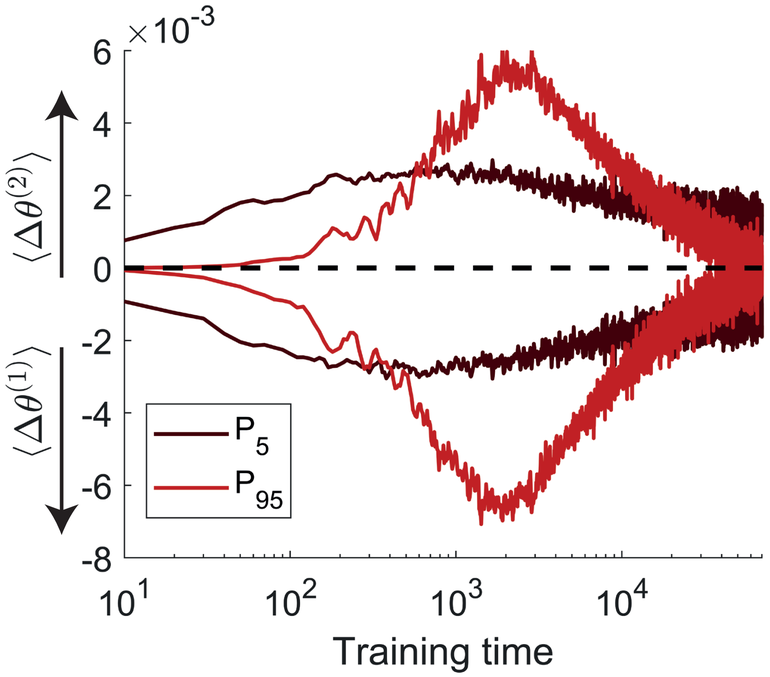


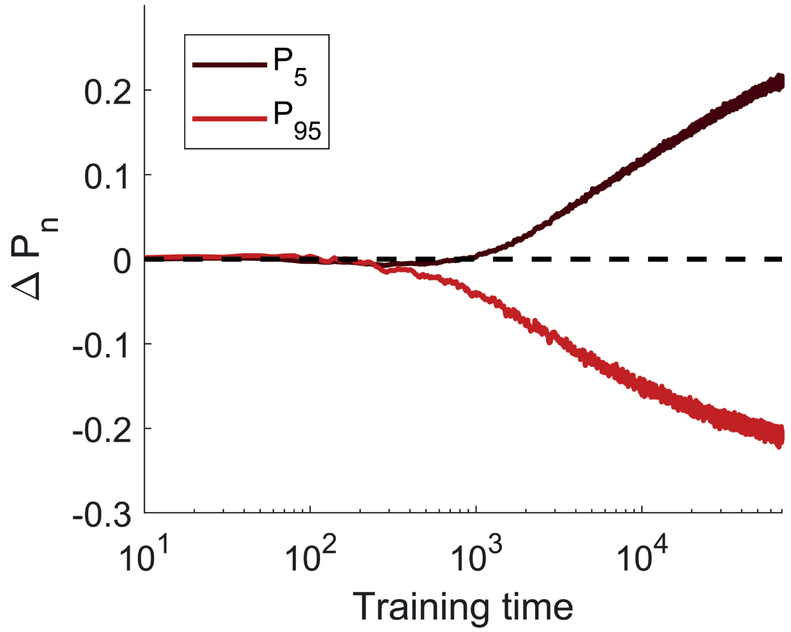
Example: column by column MNIST

TWO FORCES DRIVE THRESHOLD LEARNING

$$\Delta\theta_i/\eta_\theta = -\,y^t_{j*}\left(W^{out}_{j*i}\,sign(x_i)\right) + \sum_j \sum_l \left(W^{out}_{jl}\,x_l\right)\left(W^{out}_{ji}\,sign(x_i)\right)$$

$\Delta\theta^{(2)}$ $\Delta\theta^{(1)}$



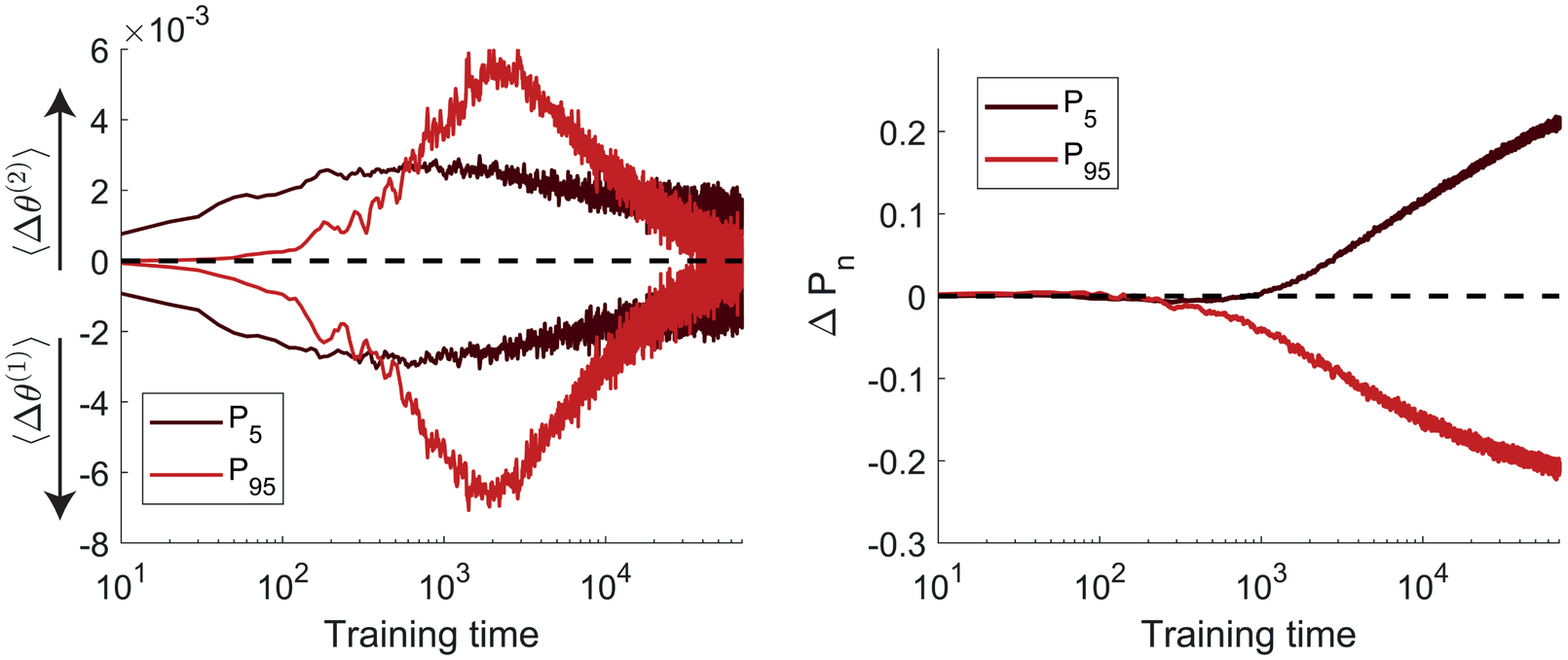


Example: column by column MNIST

TWO FORCES DRIVE THRESHOLD LEARNING

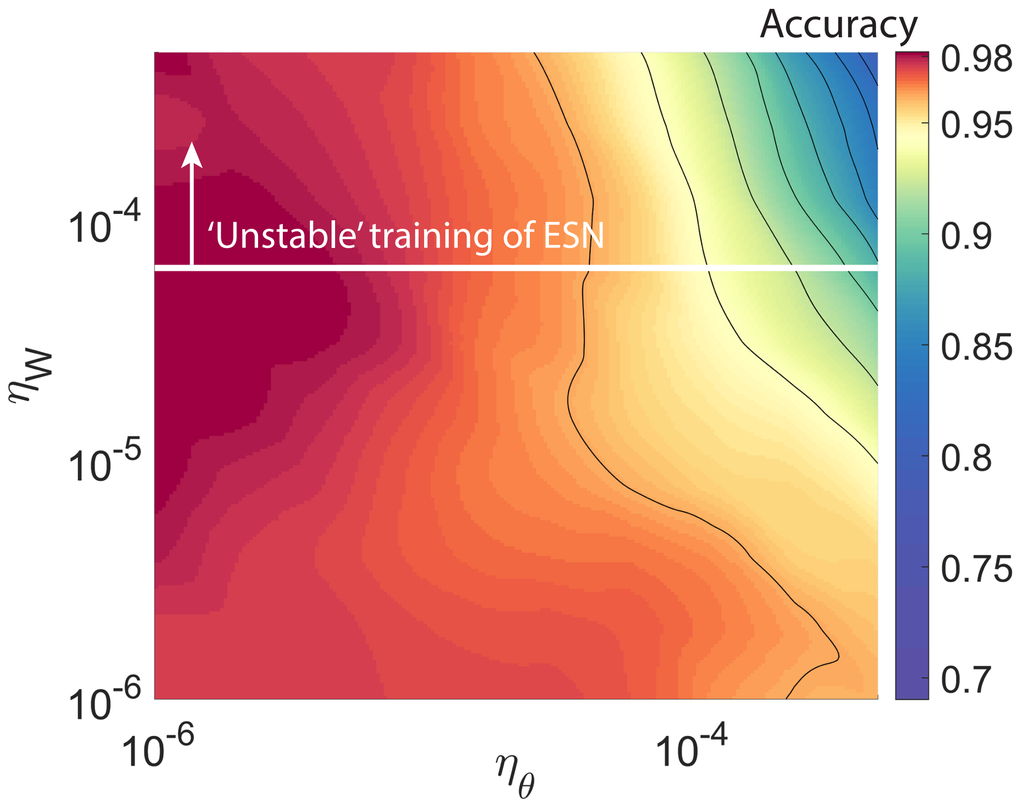
$$\Delta\theta_i/\eta_\theta = \underbrace{-\,y^t_{j*}\left(W^{out}_{j*i}\,sign(x_i)\right)}_{\Delta\theta^{(2)}} + \underbrace{\sum_j \sum_l \left(W^{out}_{jl}\,x_l\right)\left(W^{out}_{ji}\,sign(x_i)\right)}_{\Delta\theta^{(1)}}$$

Example: column by column MNIST

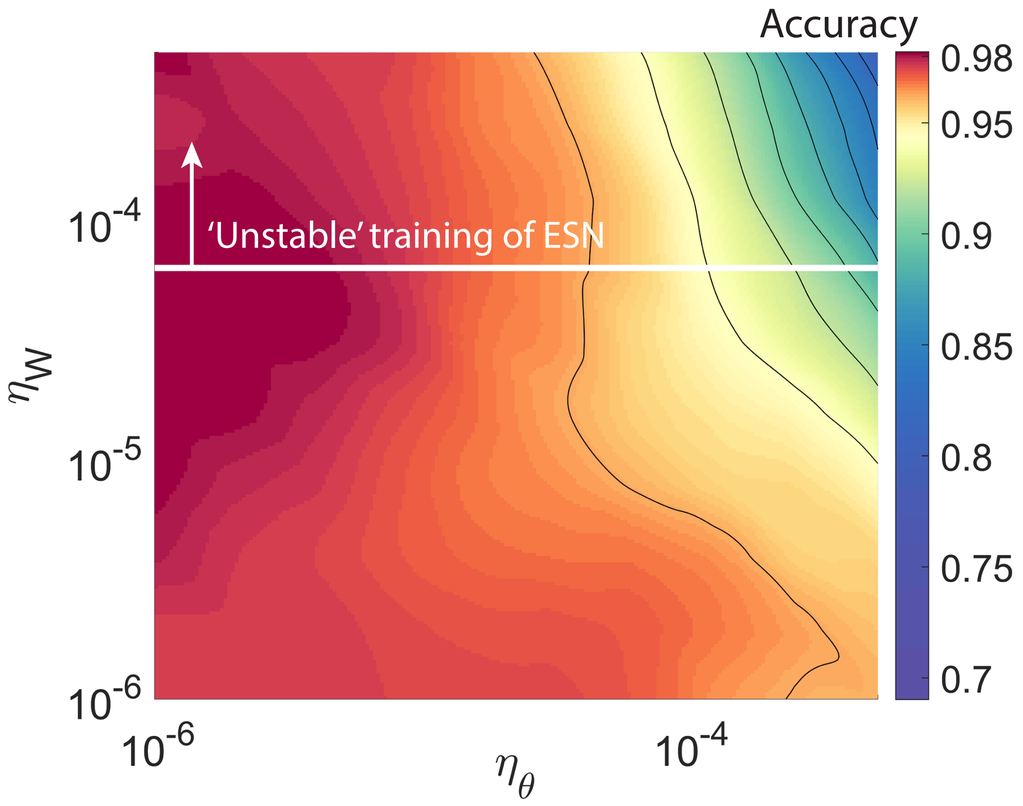


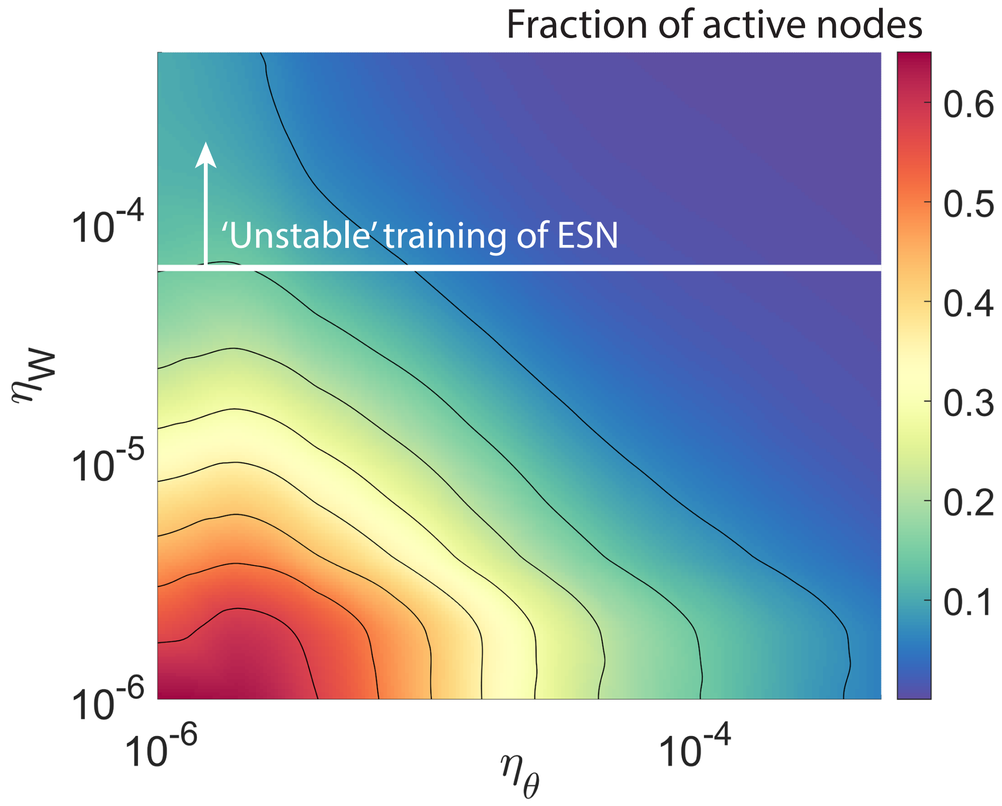
Conclusions also hold for cross-entropy error function

THRESHOLDS ALLOW FOR HIGHER LEARNING RATES

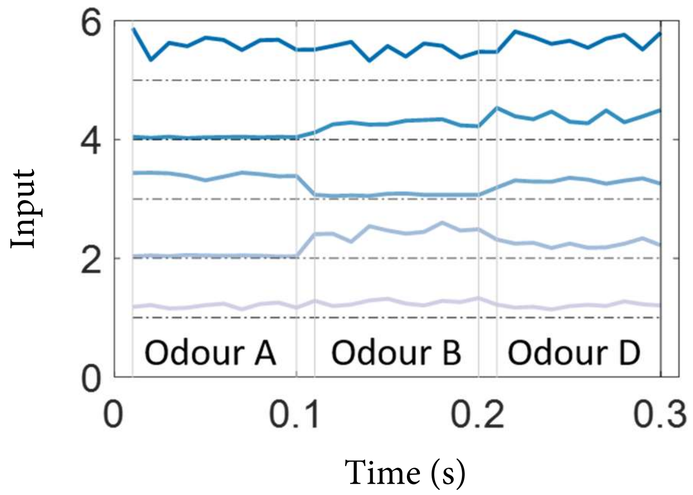


THRESHOLDS ALLOW FOR HIGHER LEARNING RATES



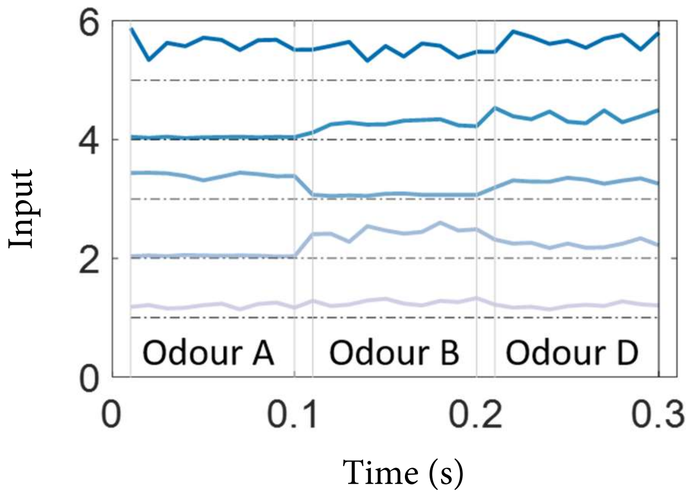


BENCHMARK: ODOURS SEQUENCE



Elissa A Hallem and John R Carlson. Coding of odors by a receptor repertoire. *Cell*, 125(1):143–160, 2006.

BENCHMARK: ODOURS SEQUENCE

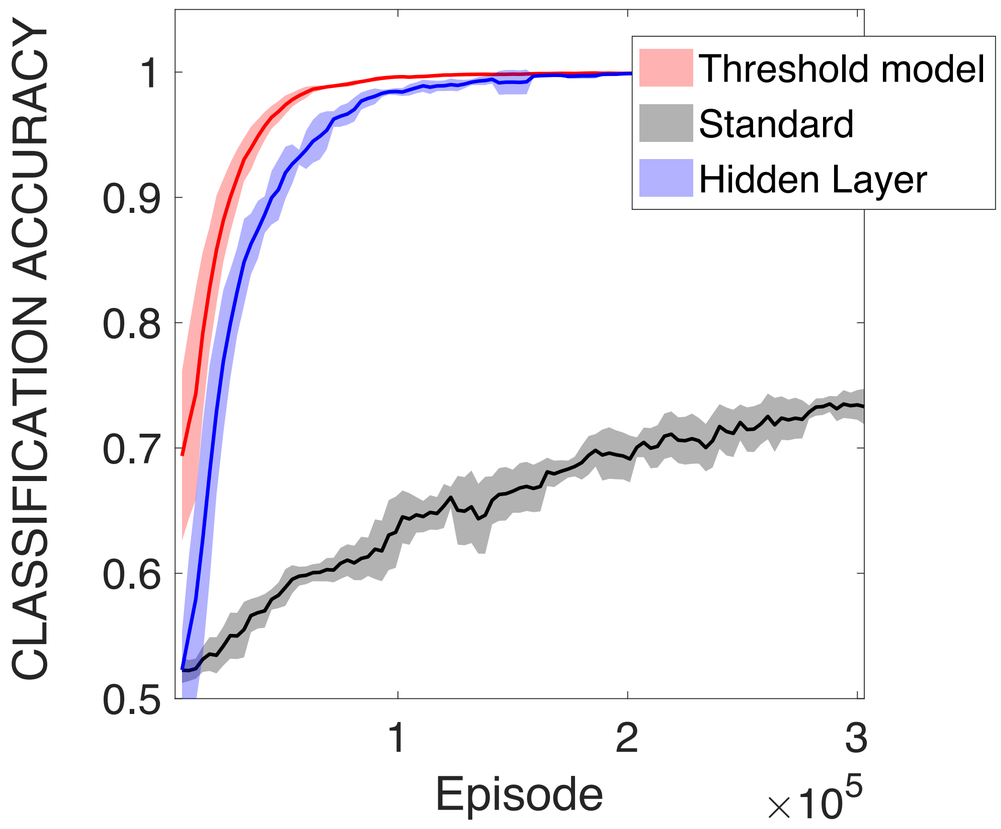


Elissa A Hallem and John R Carlson. Coding of odors by a receptor repertoire. *Cell*, 125(1):143–160, 2006.

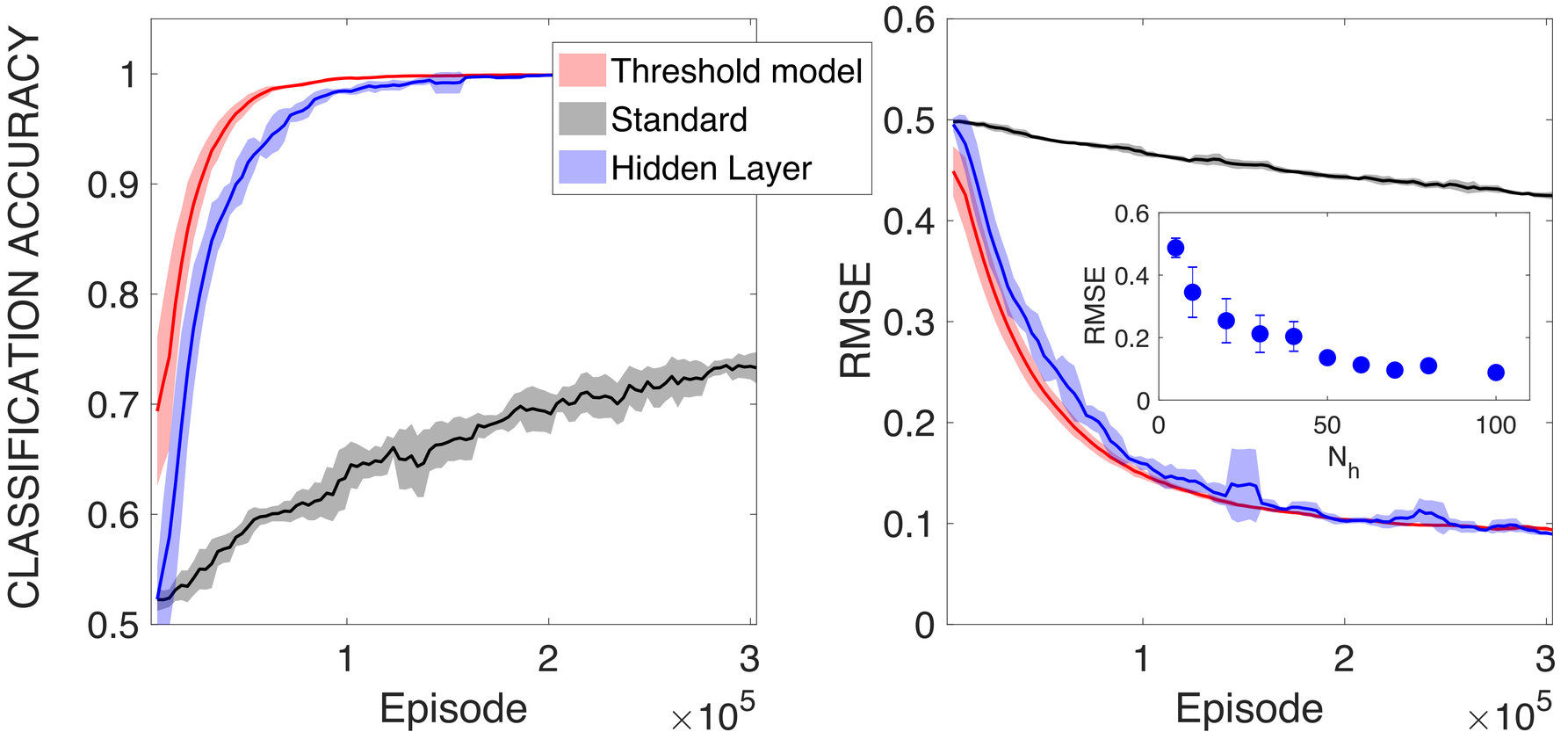
			Class
A	B	D	1
A	B	E	2
A	F	C	1
A	G	C	2
H	B	C	1
I	B	C	2

			Class
L	M	D	2
L	M	E	1
L	F	N	2
L	G	N	1
H	M	N	2
I	M	N	1

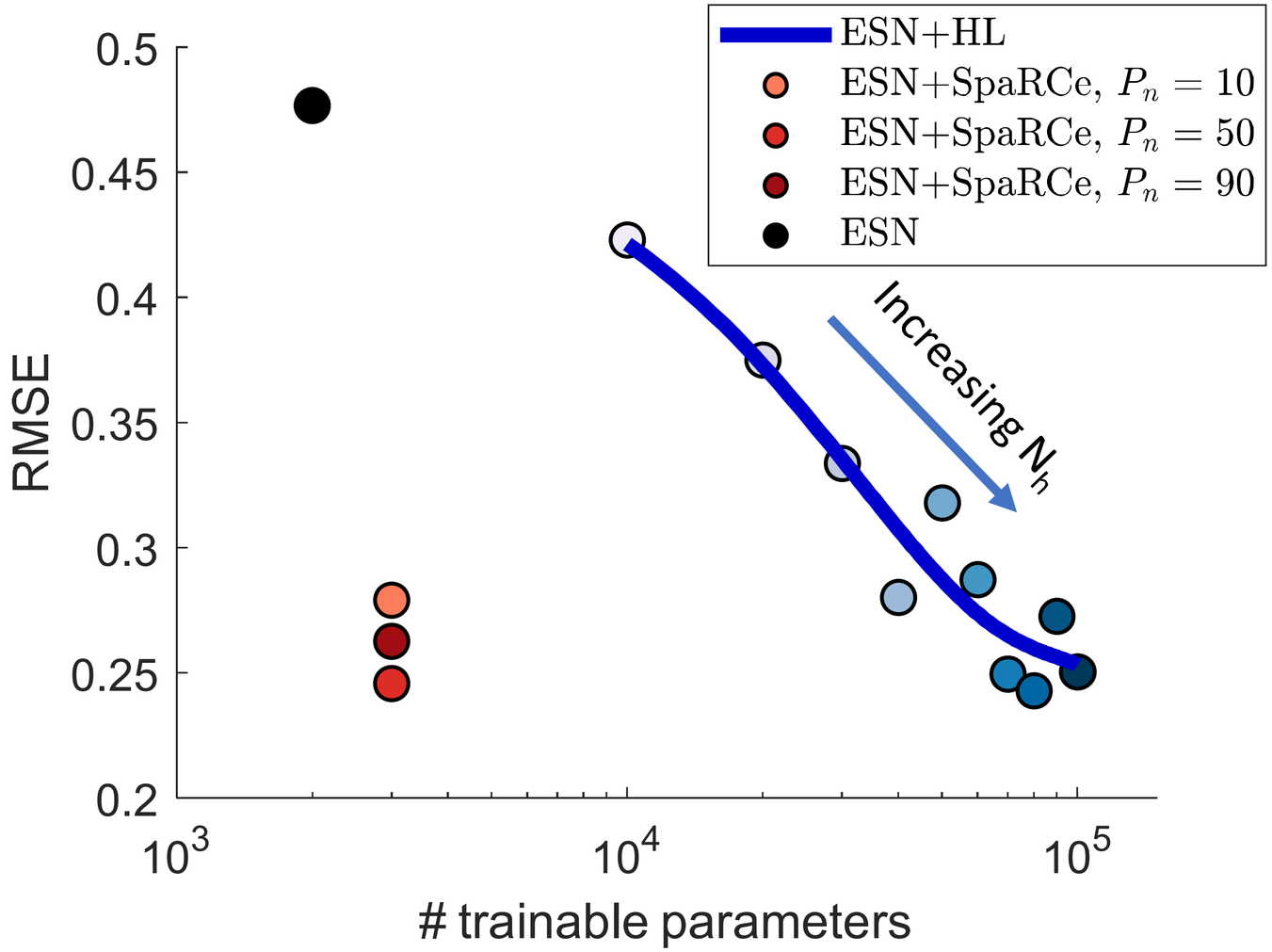
PERFORMANCE: ODOURS SEQUENCE



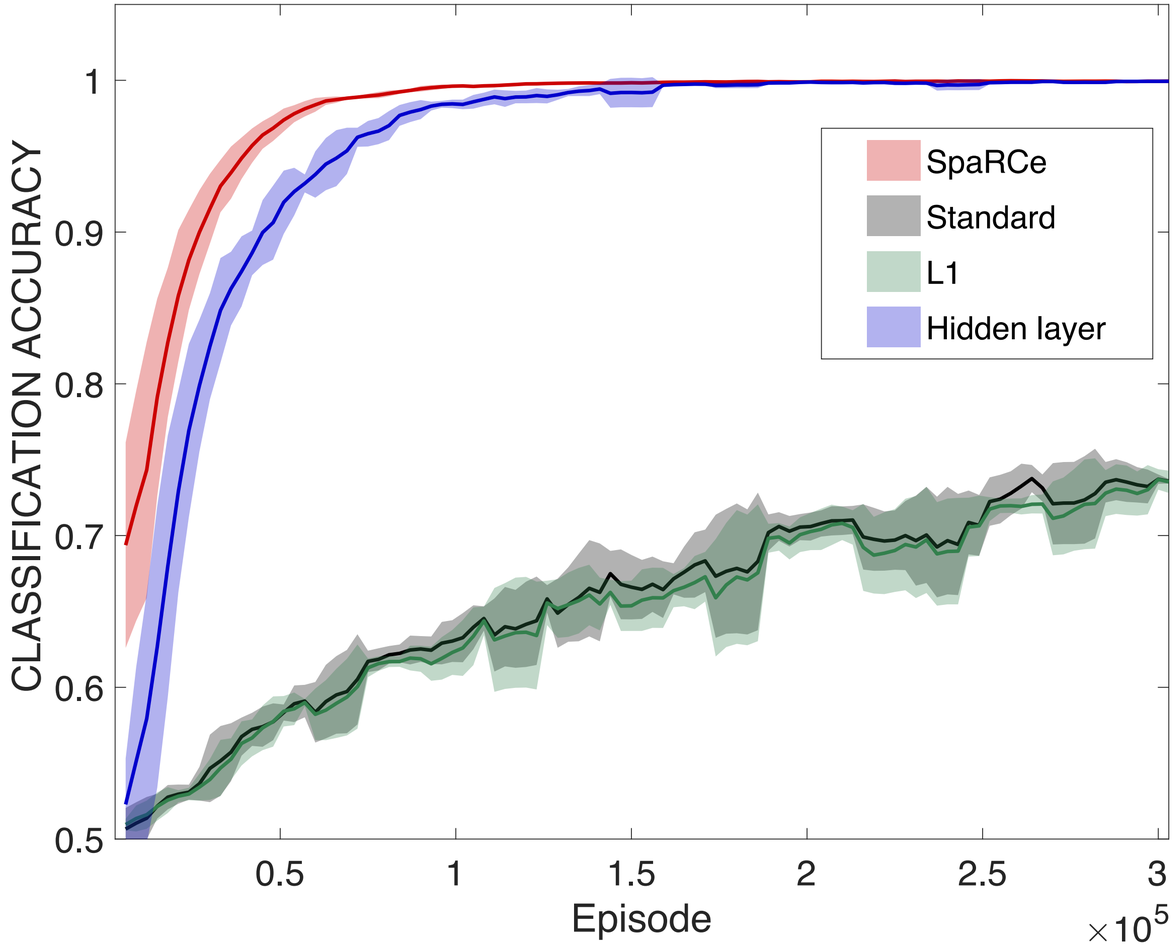
PERFORMANCE: ODOURS SEQUENCE



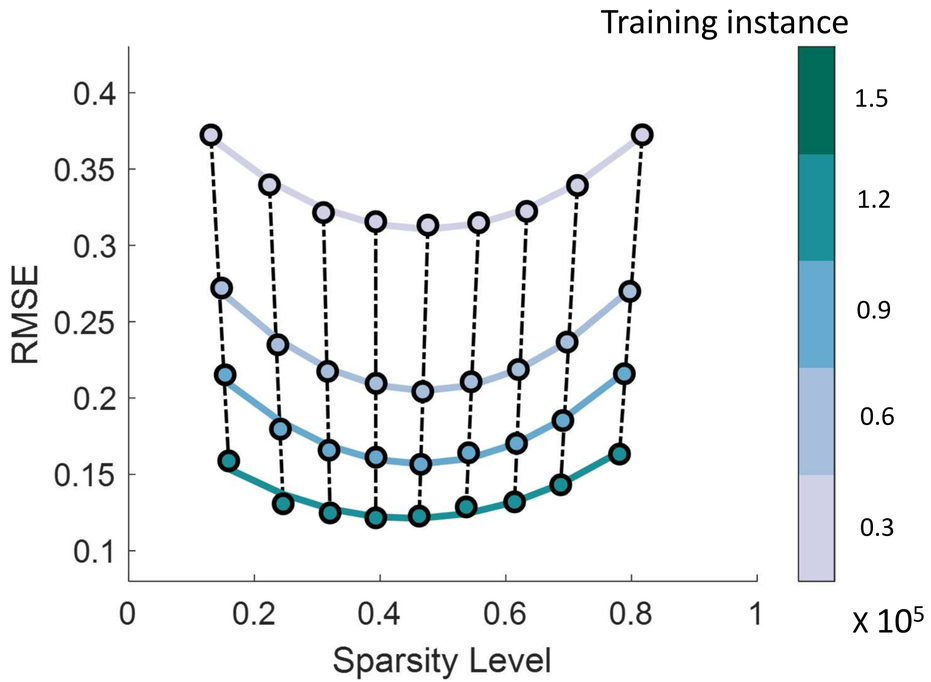
NUMBER OF LEARNABLE PARAMETERS



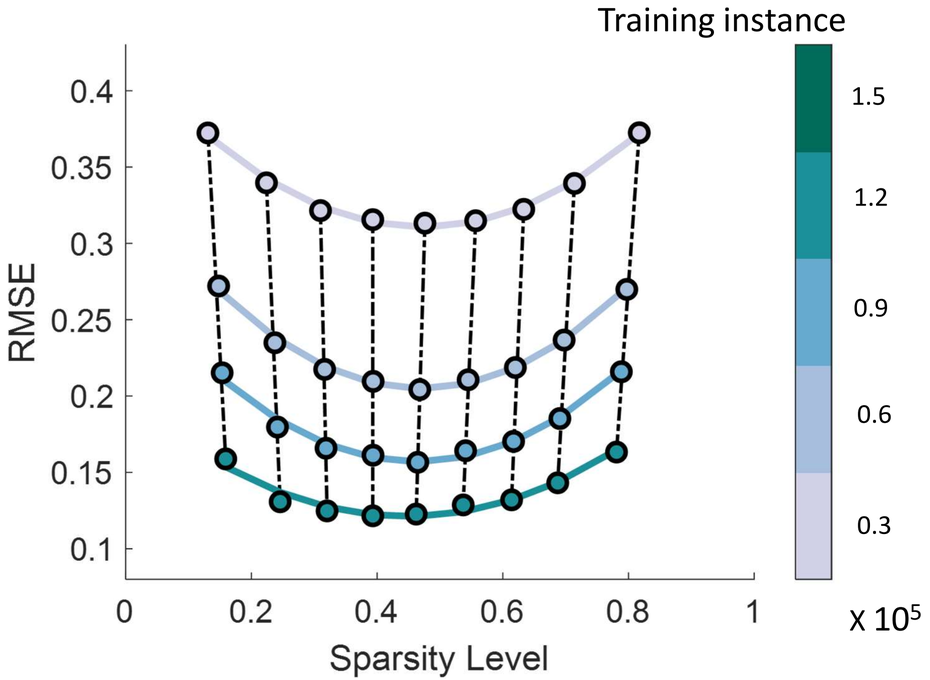
L1 DOES NOT IMPROVE PERFORMANCE

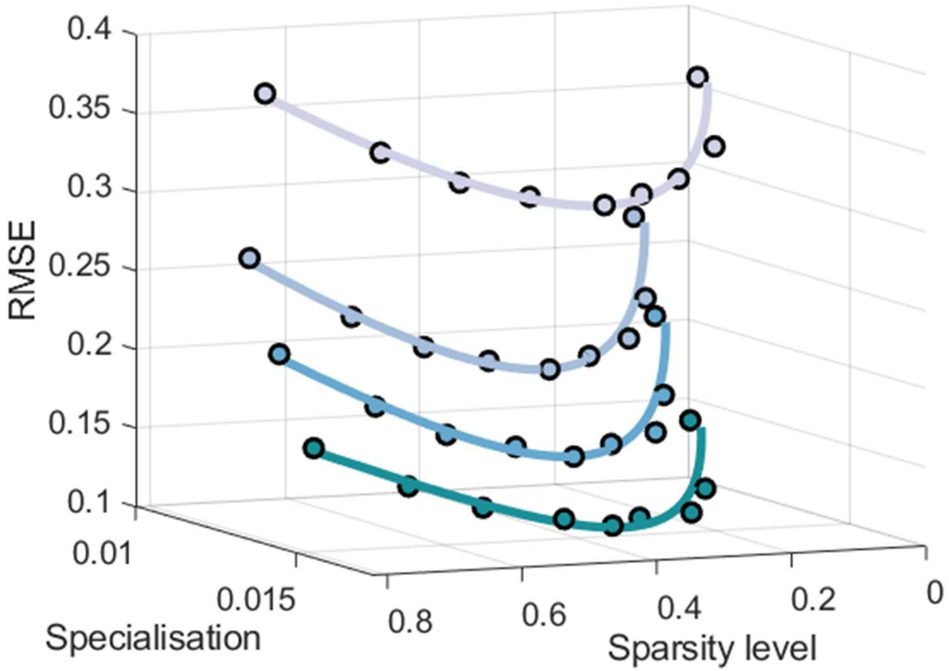


MINIMISING RMSE MODULATES THE NETWORK SPARSITY

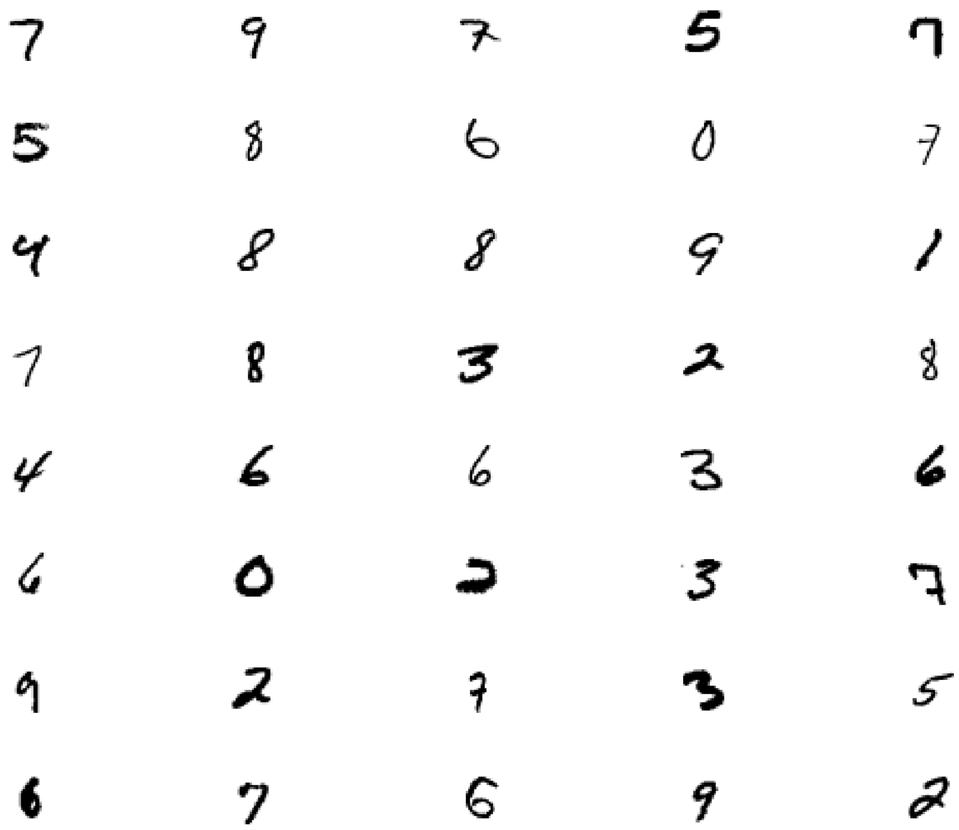


MINIMISING RMSE MODULATES THE NETWORK SPARSITY



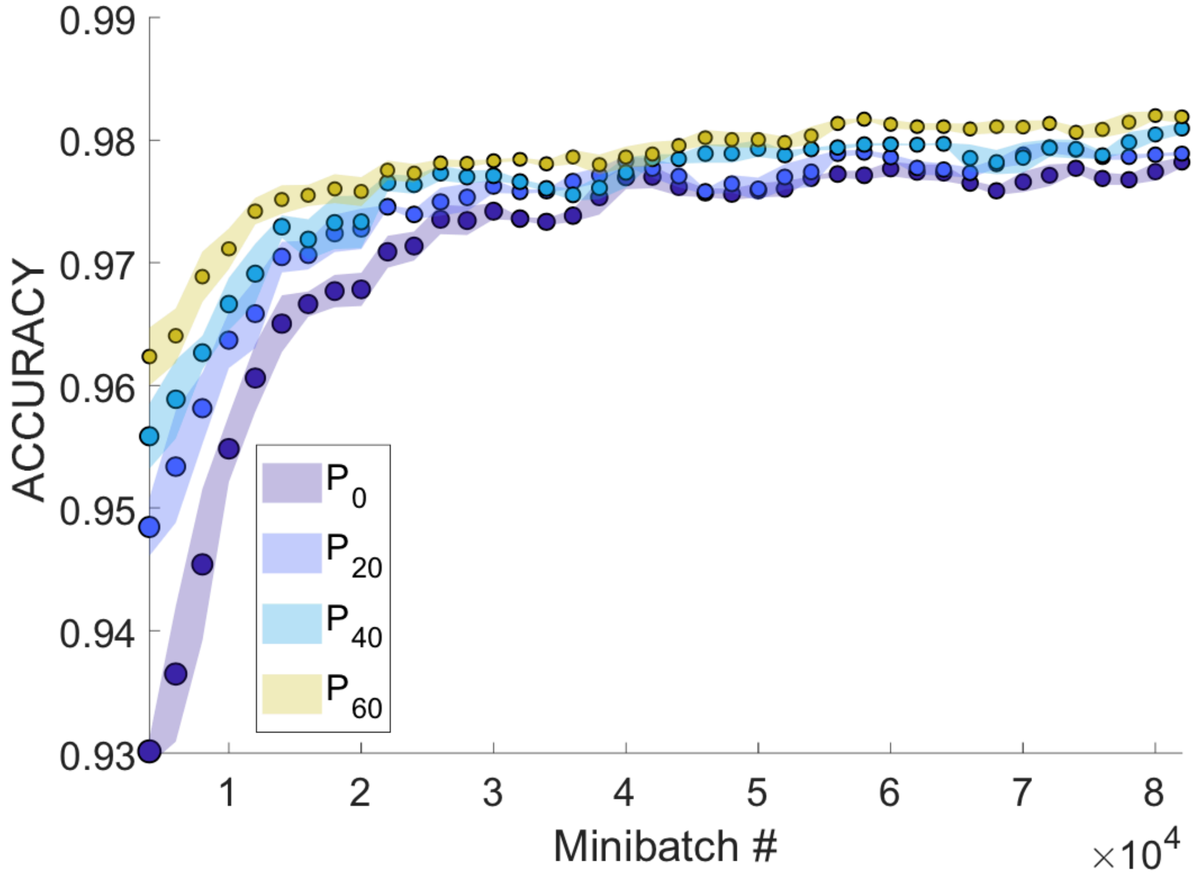


BENCHMARK: SEQUENTIAL (COLUMN BY COLUMN) MNIST

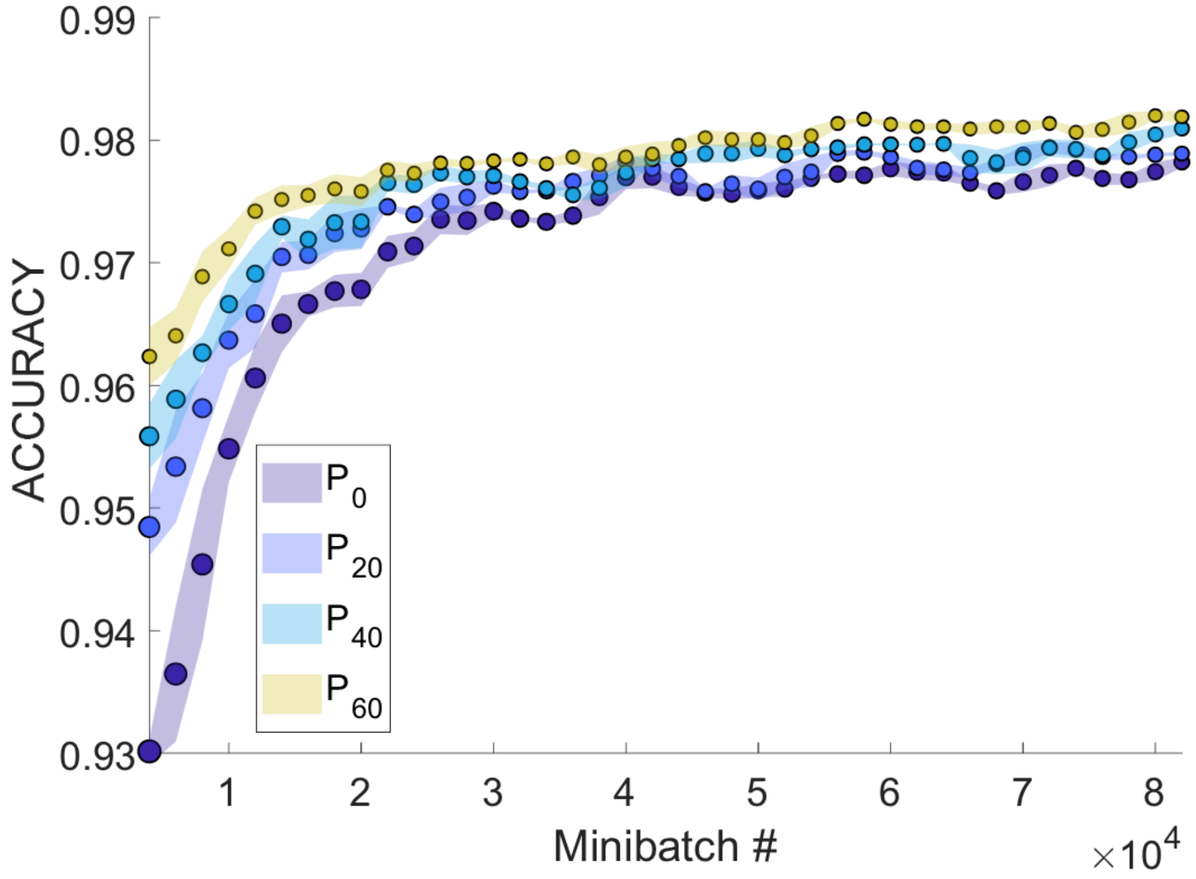


MNIST database - LeCun et al., Proceedings of the IEEE, 1998

PERFORMANCE: SEQUENTIAL (COLUMN BY COLUMN) MNIST

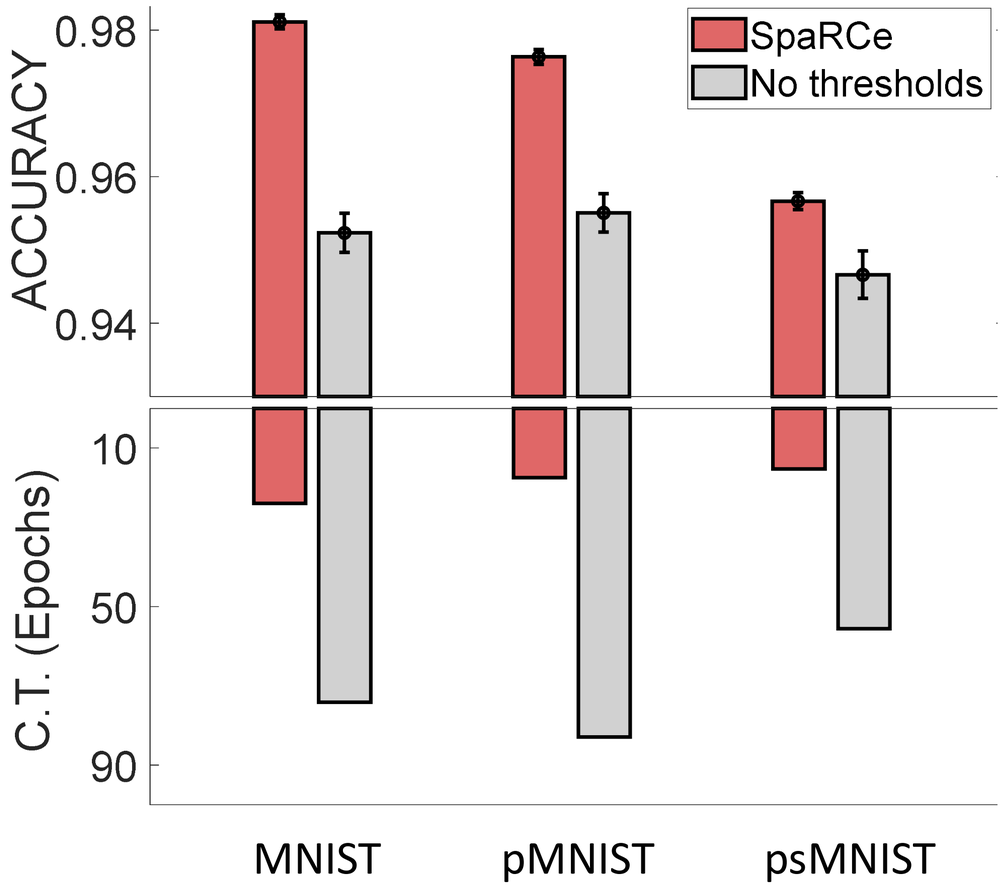


PERFORMANCE: SEQUENTIAL (COLUMN BY COLUMN) MNIST

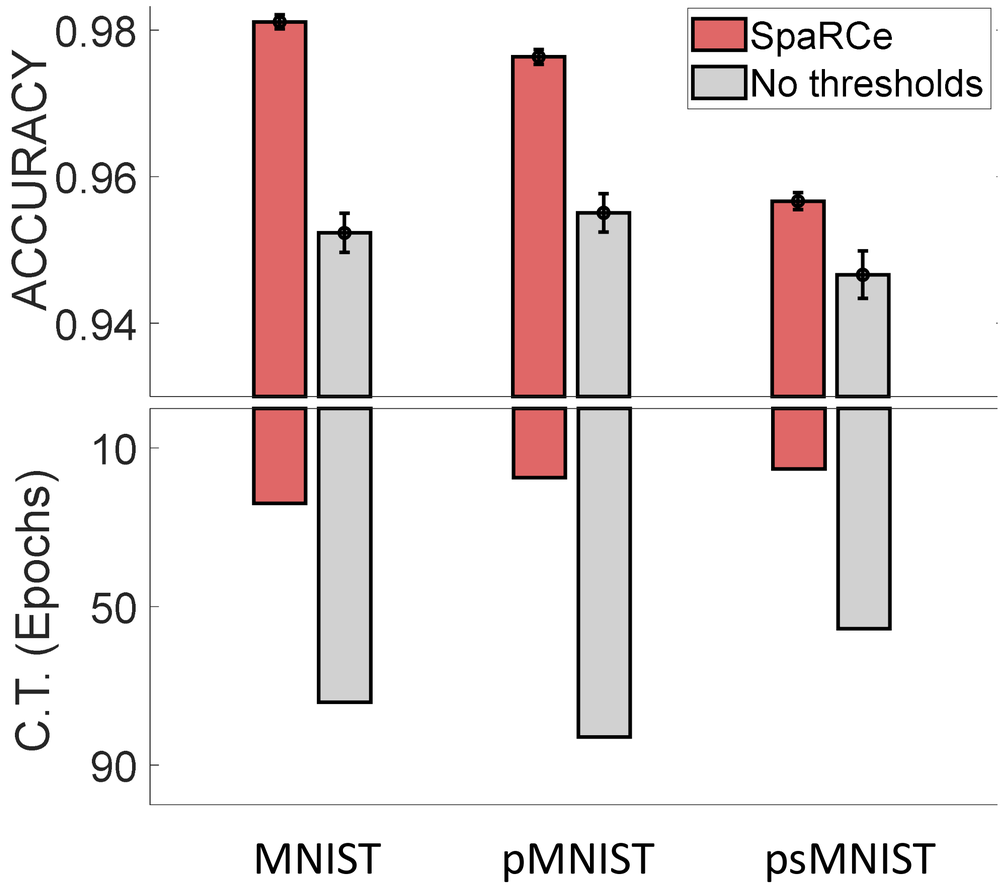


In practice we can start at P50 and adjust learning rate to reach maximum performance.

PERFORMANCE COMPARISON ON MNIST VARIADS



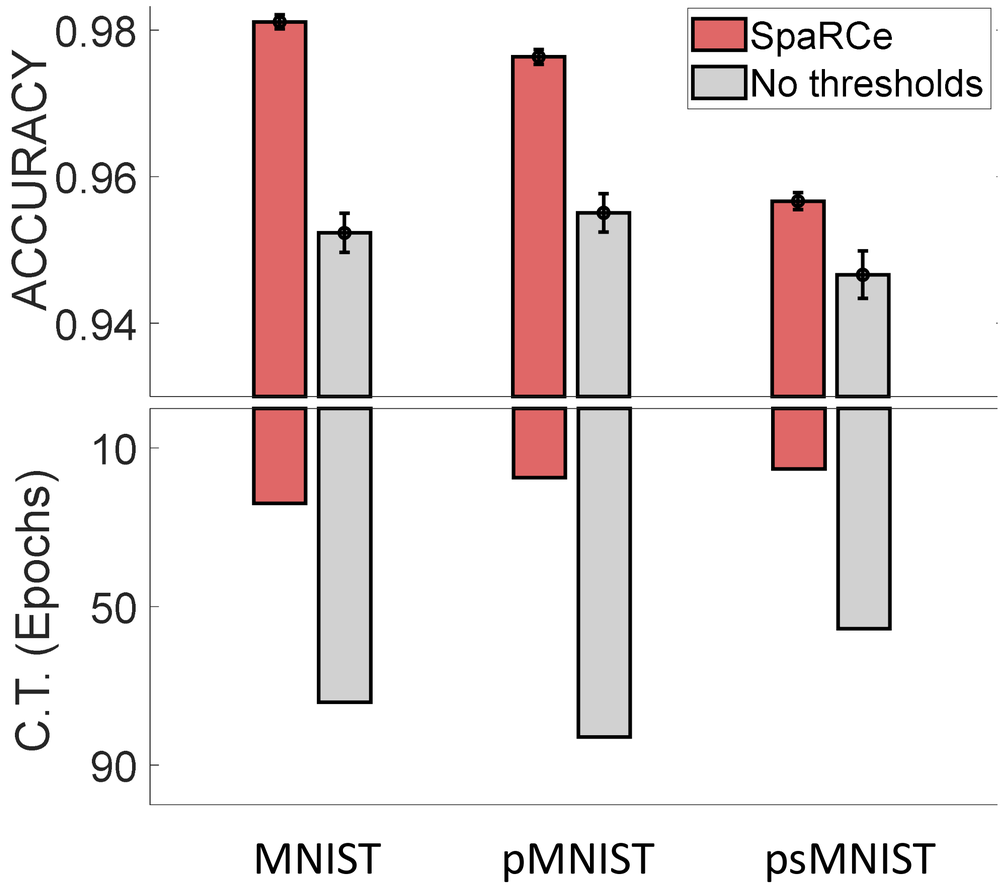
PERFORMANCE COMPARISON ON MNIST VARIADS



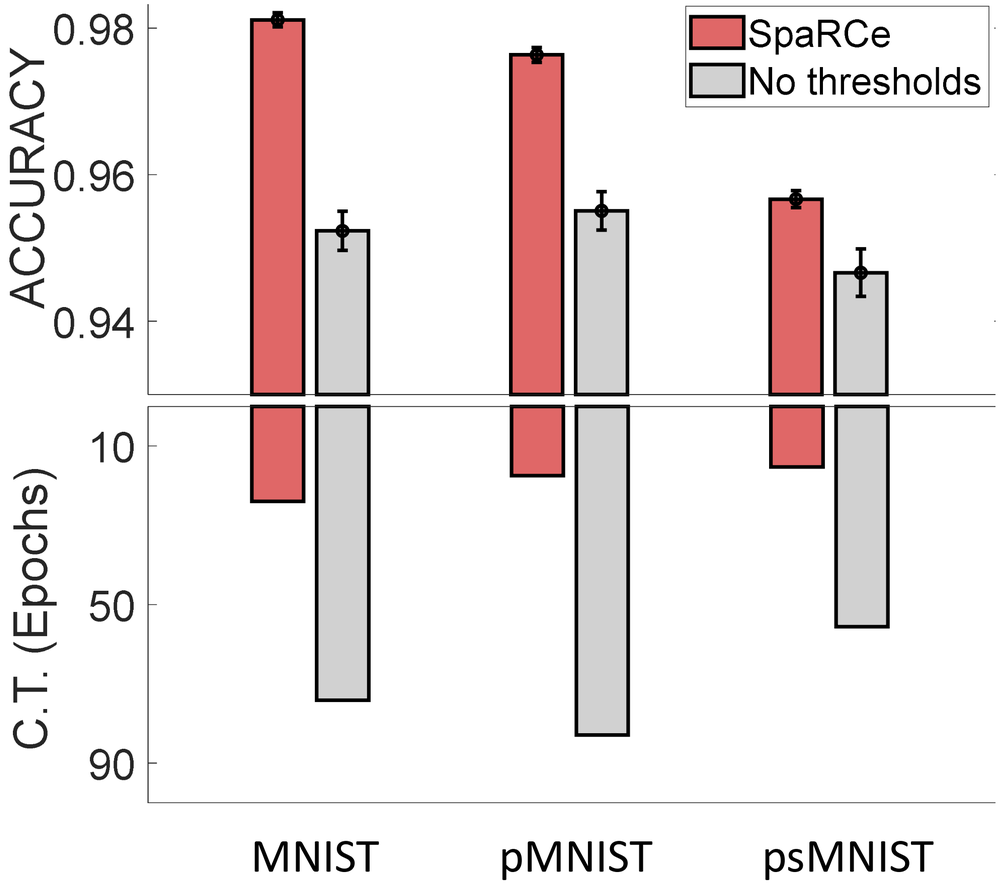
	MNIST
ESN	95.2
SpaRCe	98.1
MLP	97.0|98.5*
Conv.	98.3|99.6*

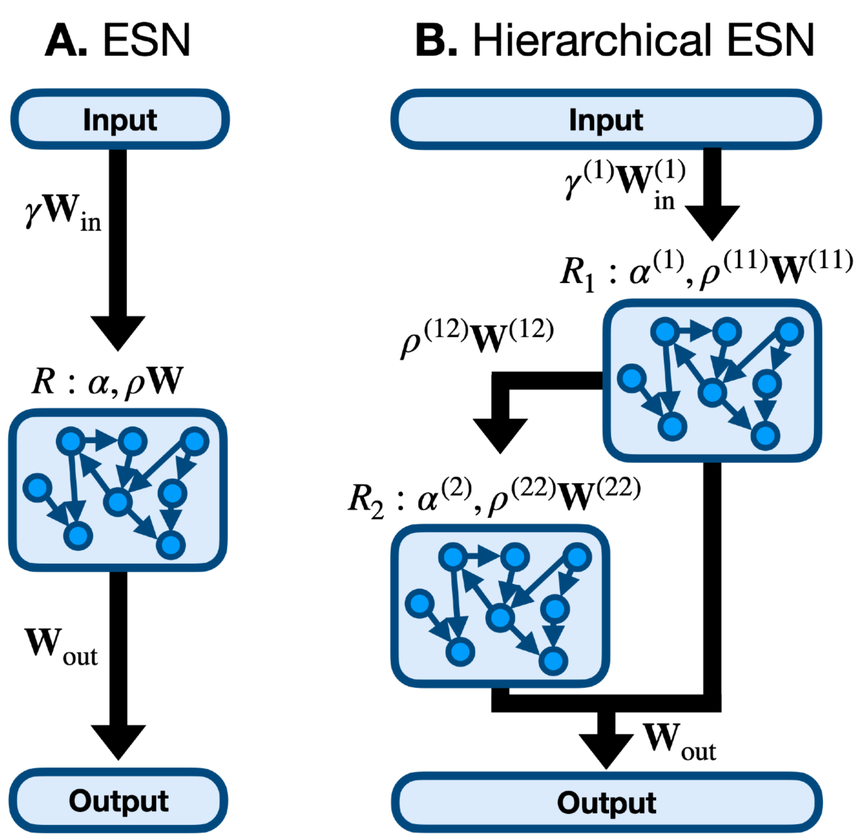
**Deng (2012) IEEE Signal Processing Magazine*
*** Chandar et al (2019) AAAI*

PERFORMANCE COMPARISON ON MNIST VARIADS



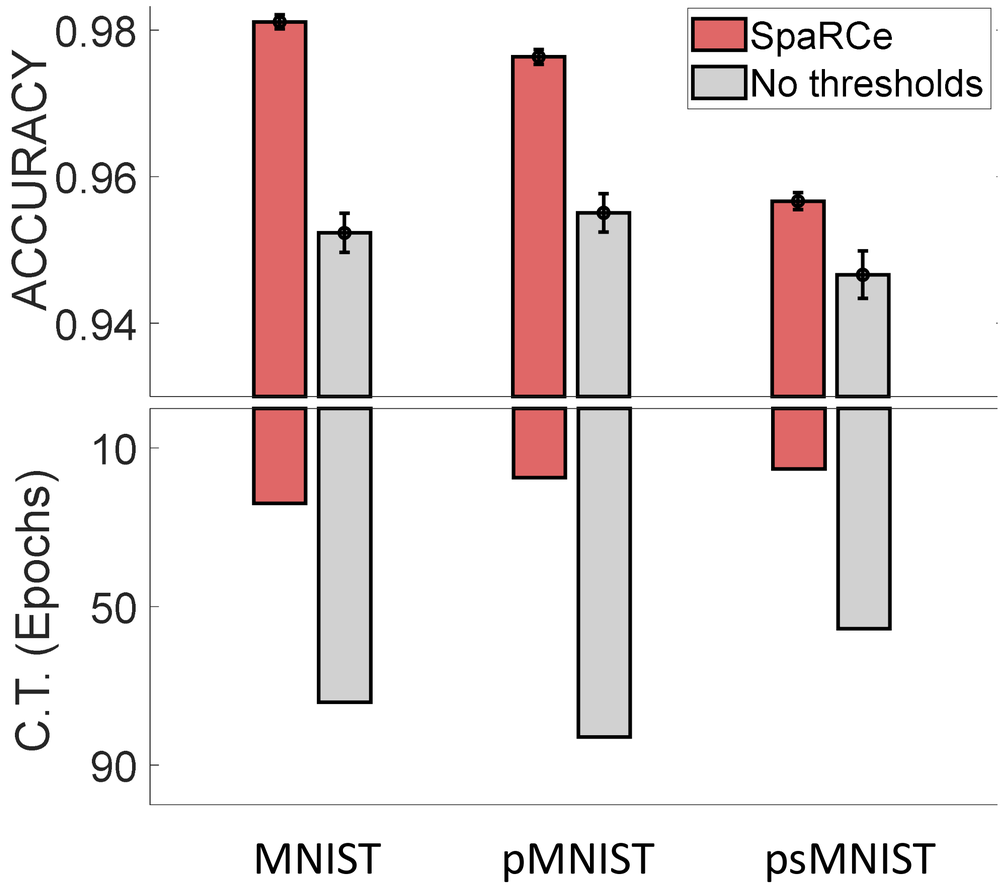
PERFORMANCE COMPARISON ON MNIST VARIADS





Manneschi et al, Frontiers in Applied Mathematics and Statistics (2021). doi.org/10.3389/fams.2020.616658 https://arxiv.org/abs/2101.04223

PERFORMANCE COMPARISON ON MNIST VARIADS

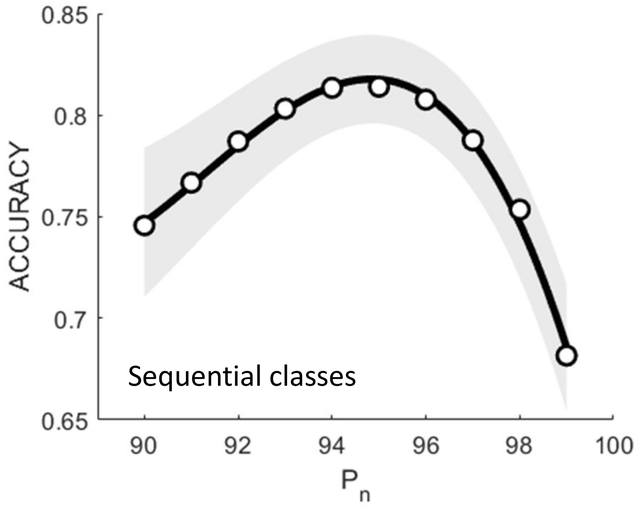


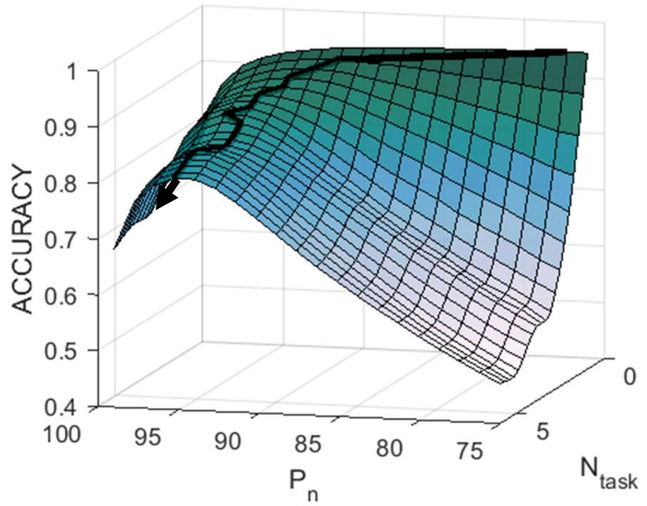
	psMNIST		
Reservoir size	500	1000	1500
ESN^2	95.6	95.9	96.3
$SpaRCe^2$	96.2	96.6	96.9
$LSTM$	89.9**		
NRU	95.4**		

**Deng (2012) IEEE Signal Processing Magazine*

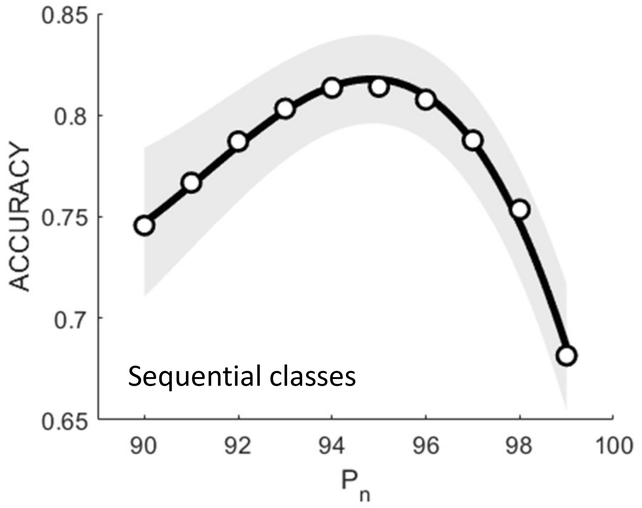
*** Chandar et al (2019) AAAI*

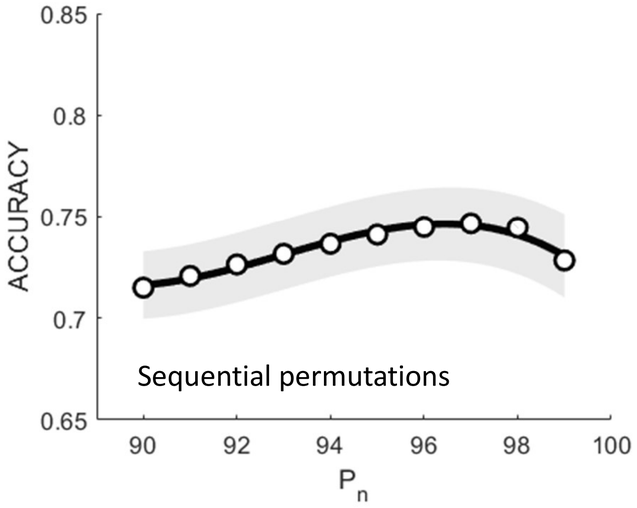
CATASTROPHIC FORGETTING

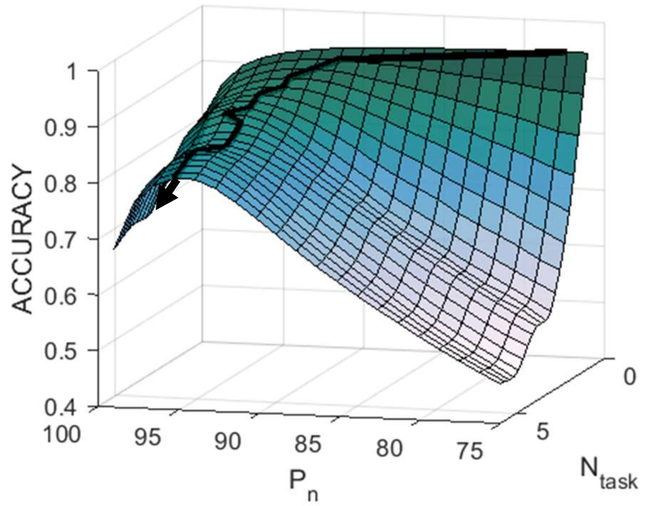


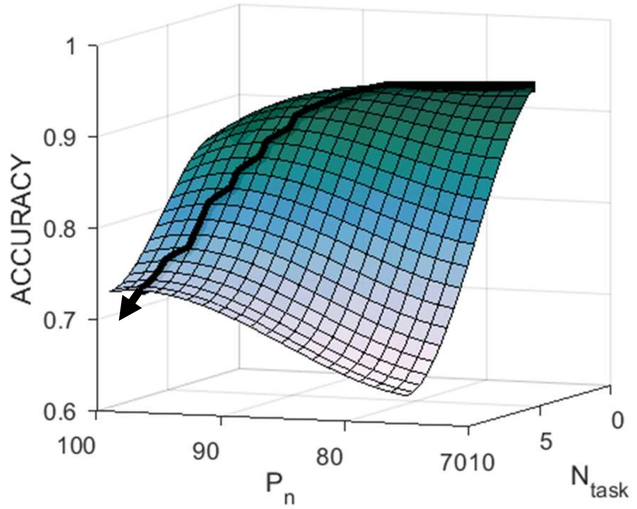


CATASTROPHIC FORGETTING

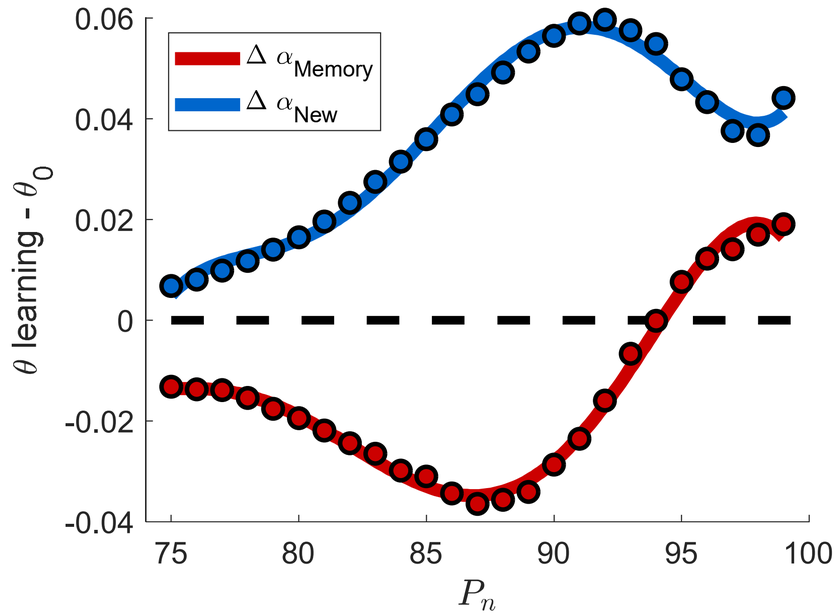








THRESHOLD INITIALISATION VS THRESHOLD LEARNING

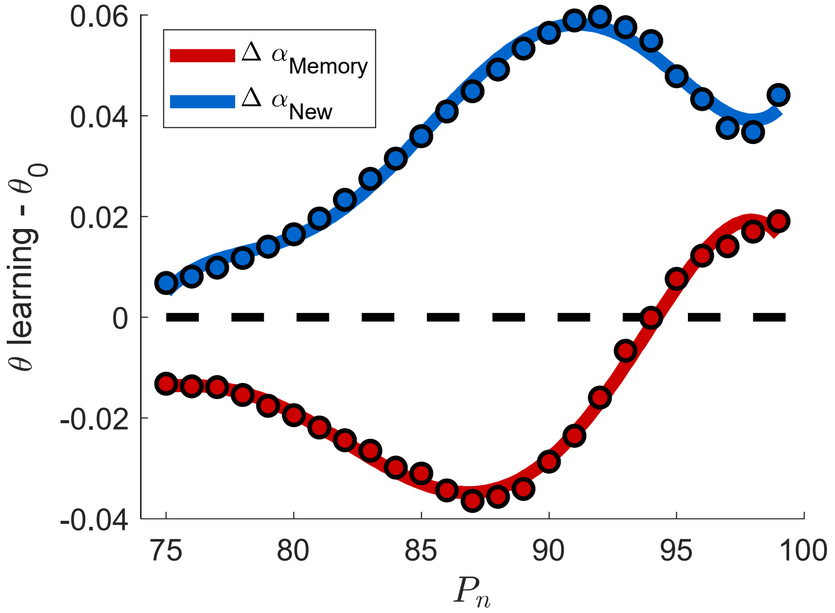


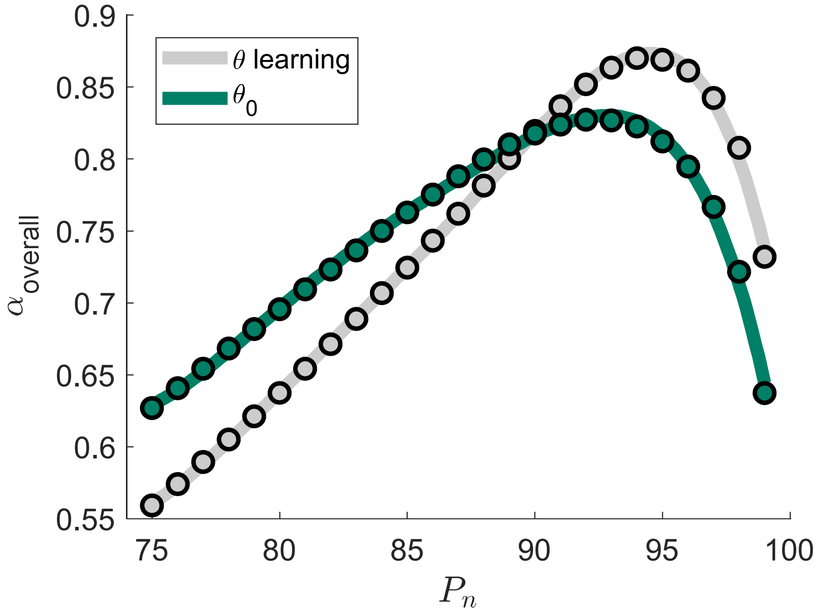
$$\alpha_{Memory} = \frac{1}{N_{task}} \sum_{n=1}^{N_{task}} [acc(n, N_{task}) - acc(n, n)]$$

$$\alpha_{New} = \frac{1}{N_{task}} \sum_{n=1}^{N_{task}} acc(n, n)$$

Metrics from: R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, Thirty-second AAAI conference on artificial intelligence, 2018.

THRESHOLD INITIALISATION VS THRESHOLD LEARNING





$$\alpha_{Memory} = \frac{1}{N_{task}} \sum_{n=1}^{N_{task}} [acc(n, N_{task}) - acc(n, n)]$$

$$\alpha_{Overall} = \frac{1}{N_{task} - 1} \sum_{n=2}^{N_{task}} \frac{acc_n}{acc(1,1)}$$

$$\alpha_{New} = \frac{1}{N_{task}} \sum_{n=1}^{N_{task}} acc(n, n)$$

Metrics from: R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, Thirty-second AAAI conference on artificial intelligence, 2018.

CATASTROPHIC FORGETTING

Results, $\alpha_{overall}$			
Sequential classes		Sequential permutations	
ESN	0.1	ESN	0.1
$SpaRCe$	0.870	$SpaRCe$	0.897
EWC	0.133^{+}	EWC	0.746^{+}
FEL	0.439^{+}	FEL	0.279^{+}
$GeppNet$	0.922^{+}	$GeppNet$	0.364^{+}

+R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, Thirty- second AAAI conference on artificial intelligence, 2018.

https://arxiv.org/abs/2110.05572v2

EchoVPR: Echo State Networks for Visual Place Recognition

Anıl Özdemir[1,†], Mark Scerri[1,†], Andrew B. Barron[2], Andrew Philippides[3], Michael Mangan[1,*], Eleni Vasilaki[1,4,*], and Luca Manneschi[1,*]

***Abstract*— Recognising previously visited locations is an important, but unsolved, task in autonomous navigation. Current visual place recognition (VPR) benchmarks typically challenge models to recover the position of a query image (or images) from sequential datasets that include both spatial and temporal components. Recently, Echo State Network (ESN) varieties have proven particularly powerful at solving machine learning tasks that require spatio-temporal modelling. These networks are simple, yet powerful neural architectures that—exhibiting memory over multiple time-scales and non-linear high-dimensional representations—can discover temporal relations in the data while still maintaining linearity in the learning. In this paper, we present a series of ESNs and analyse their applicability to the VPR problem. We report that the addition of ESNs to pre-processed convolutional neural networks led to a dramatic boost in performance in comparison to non-recurrent networks in five out of six standard benchmarks (GardensPoint, SPEDTest, ESSEX3IN1, Oxford RobotCar, and Nordland) demonstrating that ESNs are able to capture the temporal structure inherent in VPR problems. Moreover, we show that models that include ESNs can outperform class-leading VPR models which also exploit the sequential dynamics of the data. Finally, our results demonstrate that ESNs also improve generalisation abilities, robustness, and accuracy further supporting their suitability to VPR applications.**

change (e.g. DenseVLAD [4], NetVLAD [5], AMOSNet [6], SuperGlue [7], DELG [8], Patch-NetVLAD [9], and HEAPUtil [10]). While matching single images is successful in many scenarios, it can suffer from the effects of aliasing, individual image corruption, or sampling mismatches between datasets (e.g. it is challenging to ensure that images sampled along the same route precisely overlap).

Milford and Wyeth [11] were the first to demonstrate that such issues could be overcome by matching sequences of images using a global search to overcome individual image mismatches. This has led to a family of models that improve VPR performance by exploiting the temporal relationships inherent in images sampled along routes [11], [12], [13], [14], [15], [16], [17]. While achieving state-of-the-art performance on challenging real-world datasets (e.g. Oxford RobotCar [18], Extended CMU Seasons [19], Pittsburgh [20], Tokyo24/7 [4], Nordland [21], Mapillary Street Level Sequences [22]), such models often include an explicit encoding of non-visual information to limit the image search space, require a cache of input images for global searches, and can be computationally expensive in

CHALLENGING TASKS

Visual Place Recognition: The Norland Test

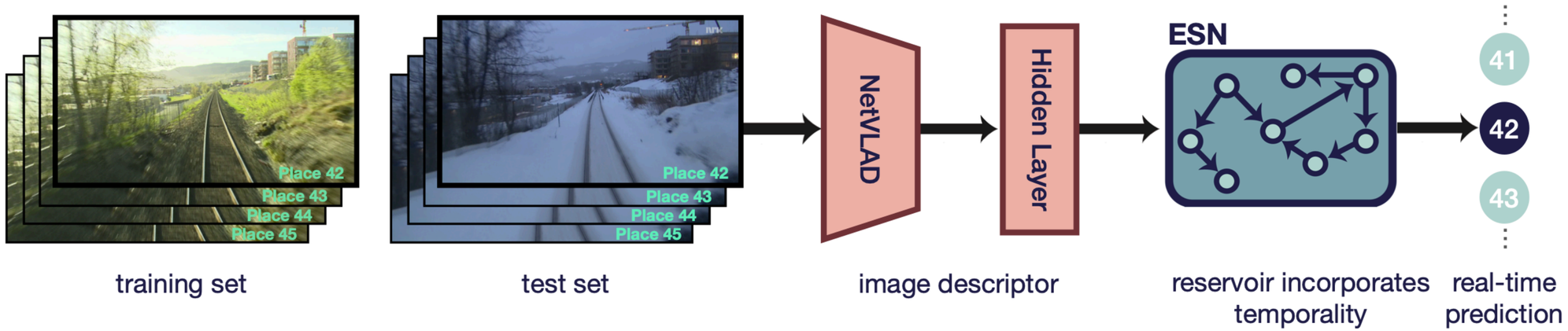
Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters

CHALLENGING TASKS

Visual Place Recognition: The Norland Test

Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters

ECHO-VPR FRAMEWORK

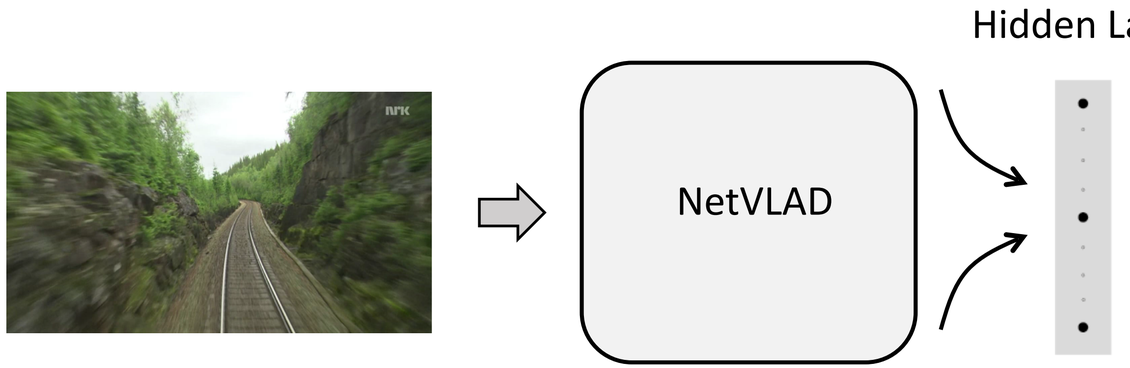


We used a small subset removed from the test set to tune hyper-parameters without overfitting.

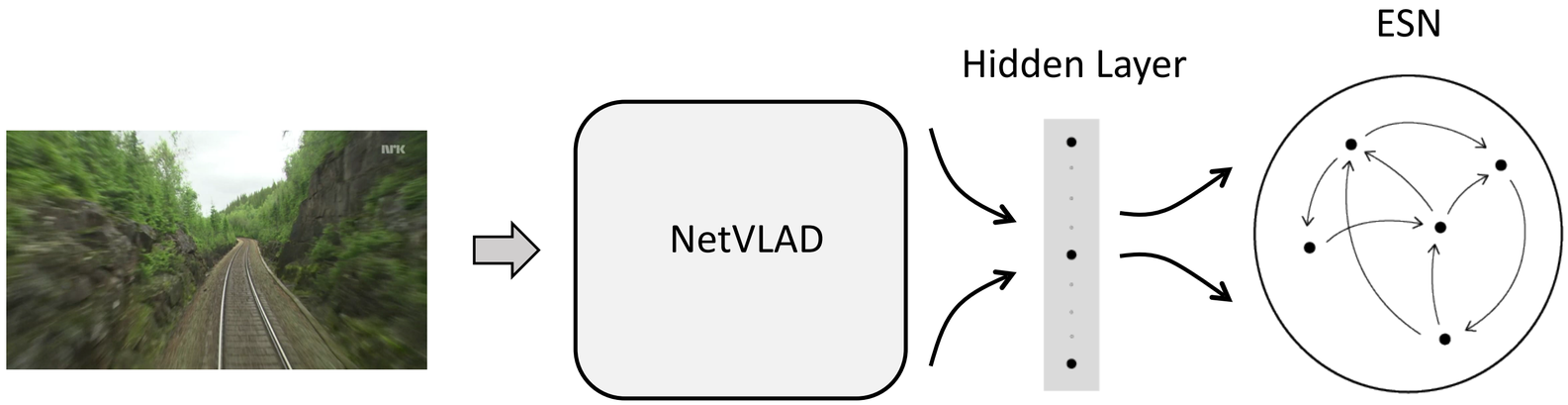
Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters

NetVLAD

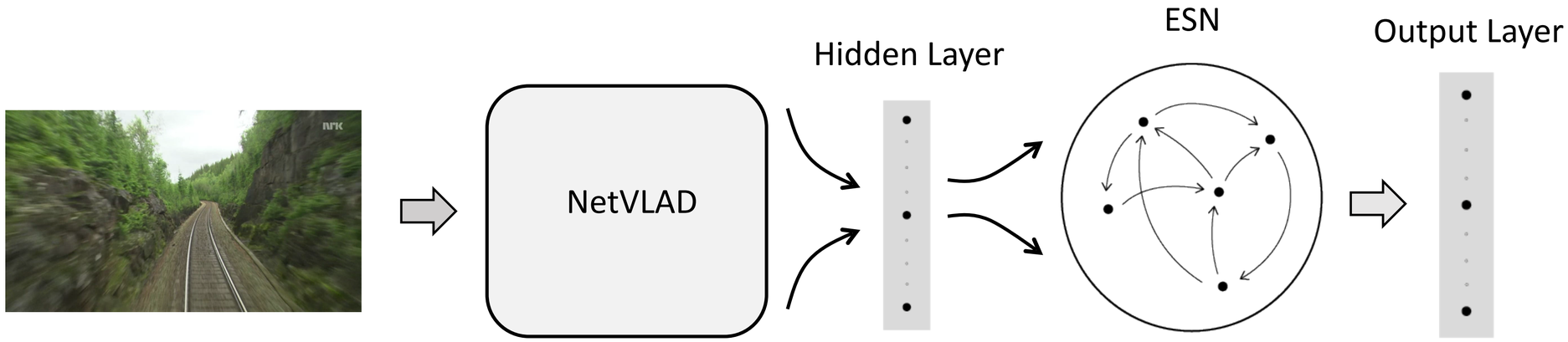
Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters



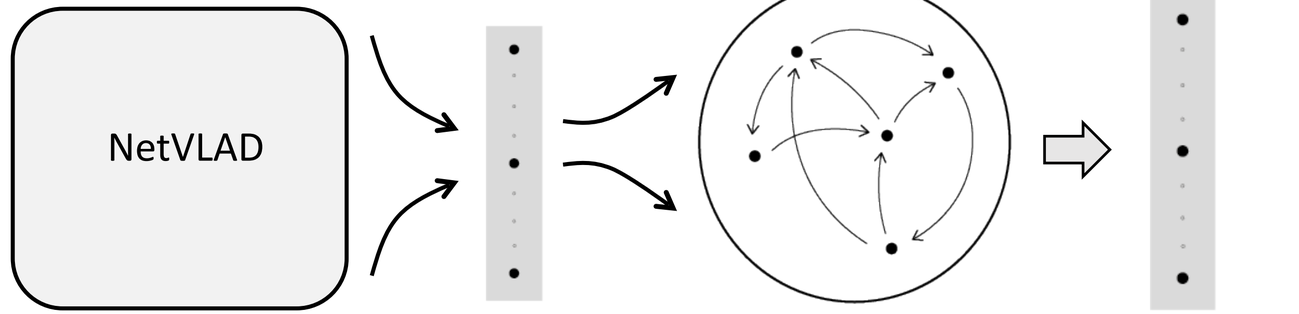
Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters



Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters



Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters

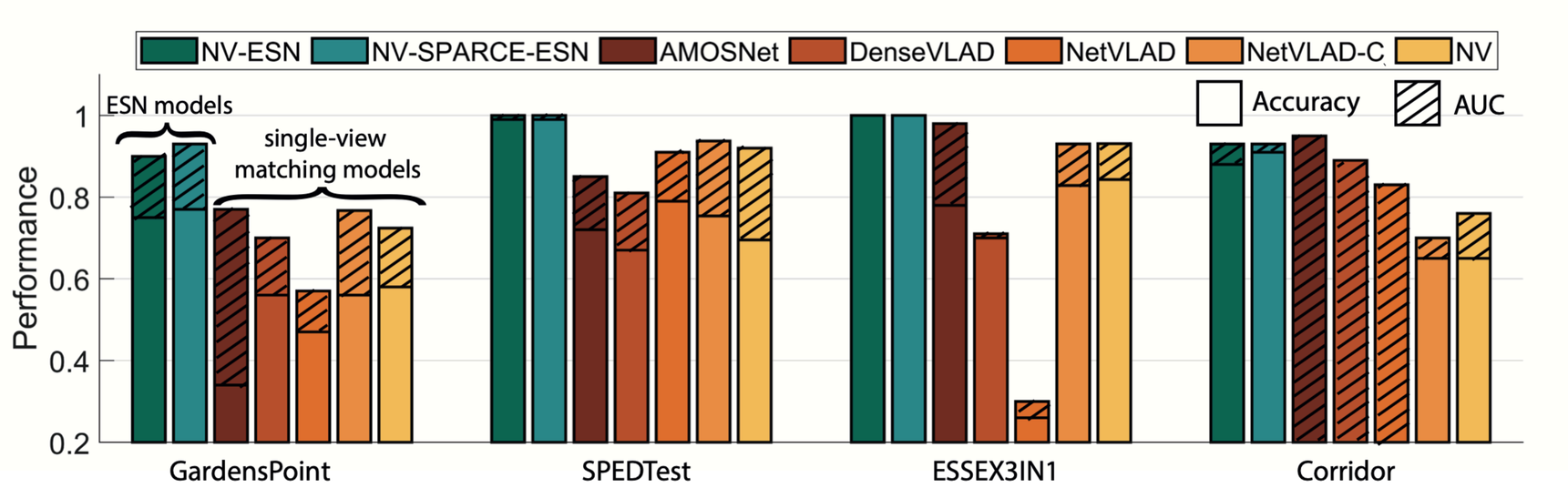


ESN Activities example

Output Layer example

Ozdemir, Scerri et al (2022), IEEE Robotics and Automation Letters

RESULTS IN "SMALL" DATASETS (200–600 IMAGES)



RESULTS IN "LARGE" DATASETS (~30,000 IMAGES)

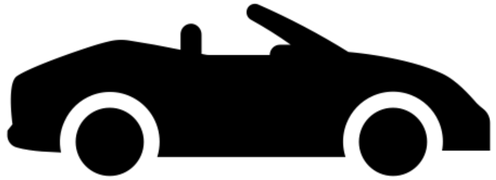
Method	Nordland Summer vs Winter			Oxford RobotCar Day vs Night		
	R@1	R@5	R@10	R@1	R@5	R@10
NetVLAD [5]	13.0	20.6	25.0	31.2	48.6	58.3
SuperGlue [7]	29.1	33.4	35.0	–	–	–
DELG global [8]	23.4	35.4	41.7	–	–	–
DELG local [8]	60.1	63.5	64.6	–	–	–
Patch-NetVLAD [9]	44.9	50.2	52.2	80.7	84.8	86.4
NV-ESN	47.5	61.0	66.4	44.0	58.1	65.5
NV-SPARCE-ESN	46.7	60.3	65.9	80.4	87.9	91.5
PatchL-ESN	**66.4**	**76.0**	**78.7**	75.9	85.2	87.7
PatchL-SPARCE-ESN	66.3	75.7	78.6	**88.8**	**97.0**	**98.2**

WHY DO WE CARE ABOUT EFFICIENT ML MODELS?

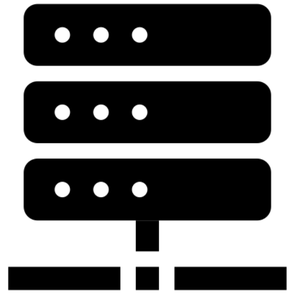
2019

Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell **Ananya Ganesh** **Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, aganesh, mccallum}@cs.umass.edu



VS



WHY DO WE CARE ABOUT EFFICIENT ML MODELS?

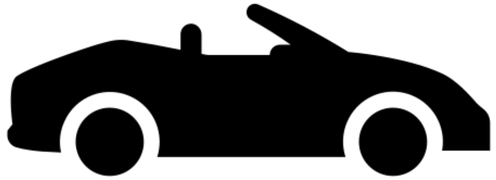
2019

Energy and Policy Considerations for Deep Learning in NLP

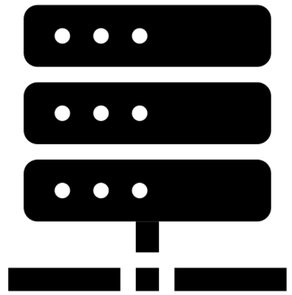
Emma Strubell **Ananya Ganesh** **Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, aganesh, mccallum}@cs.umass.edu

Consumption	**CO_2e (lbs)**
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]



VS



SUMMARY

SUMMARY

- SPARCe: SPArse Reservoir Computing, inspired by the insect brain.

SUMMARY

- SPARCe: SPArse Reservoir Computing, inspired by the insect brain.
- Key ingredients: threshold learning and “democratic” threshold initialisation.

SUMMARY

- SPARCe: SPArse Reservoir Computing, inspired by the insect brain.
- Key ingredients: threshold learning and “democratic” threshold initialisation.
- Sparsity leads to neuronal specialisation and alleviates catastrophic forgetting.

SUMMARY

- SPARCe: SPArse Reservoir Computing, inspired by the insect brain.
- Key ingredients: threshold learning and “democratic” threshold initialisation.
- Sparsity leads to neuronal specialisation and alleviates catastrophic forgetting.
- Threshold learning“kills” correlated neurons in the reservoir and helps increase neuronal specialisation.

SUMMARY

- SPARCe: SPArse Reservoir Computing, inspired by the insect brain.
- Key ingredients: threshold learning and “democratic” threshold initialisation.
- Sparsity leads to neuronal specialisation and alleviates catastrophic forgetting.
- Threshold learning“kills” correlated neurons in the reservoir and helps increase neuronal specialisation.
- It helps increase performance in problems where memory is required, such as the VPR.

Thank you!