Working memory facilitates reward-modulated Hebbian learning in recurrent neural networks

Roman Pogodin[1], Dane Corneil[2], Alexander Seeholzer[2], Joseph Heng[2], Wulfram Gerstner[2]

[1] Gatsby Unit, University College London, London, [2] School of Computer and Communication Sciences and School of Life Sciences, Brain Mind Institute, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Introduction

How do we learn new movements?

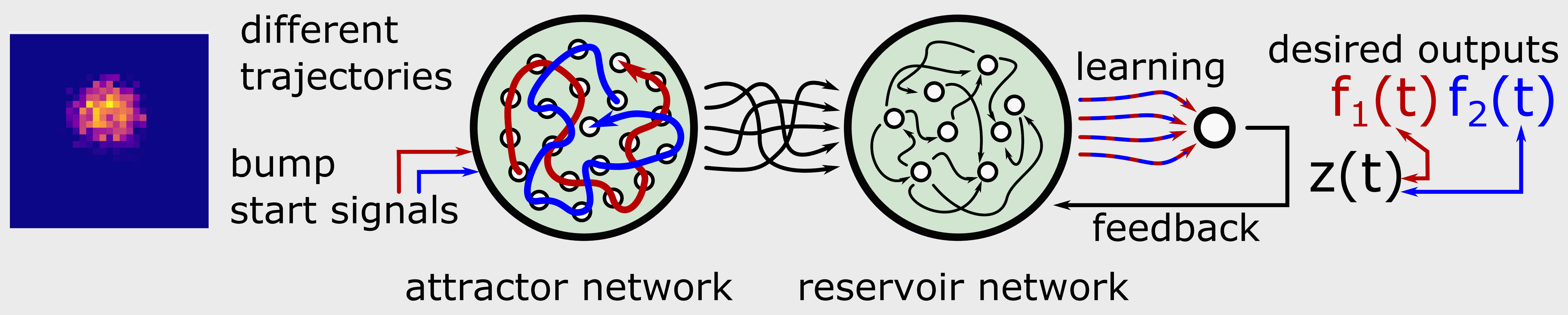
- reservoir computing model does that with a pool of randomly connected (cortical) neurons and plastic readout weights [1]
- training the readout with FORCE [1] is efficient but implausible
- training with plausible learning rules such as reward-modulated Hebbian learning [2] only works for short and simple tasks
- the key problem is that plausible rules are too unstable

Our model:

- we extend the reservoir with dynamic working memory
- the memory provides a temporally stable signal to make reward-modulated Hebbian learning more robust
- it is implemented with a 2D dynamic attractor [3]

Model

Attractor network (left) creates a task-specific trajectory and feeds it to the **reservoir network** (right), which learns the task:



Attractor network [3]. Rate neurons $\mathbf{x}$ with input $\mathbf{e}$ and adaptation $\mathbf{h}$ of strength s:

$$\tau_{\mathrm{m}}\dot{\mathbf{x}} = -\mathbf{x} + [\mathbf{Jx} + \mathbf{e} - \mathbf{h}]_+$$
$$\tau_{\mathrm{a}}\dot{\mathbf{h}} = -\mathbf{h} + s\,\mathbf{x}$$

- weights $\mathbf{J}$ create a "bump" solution
- adaptation $\mathbf{h}$ moves the bump
- this movement projects to the reservoir

Reservoir network [1, 2]. Rate neurons $\mathbf{u}$ with input $\mathbf{W}_{\mathrm{attr}}\mathbf{x}$, readout $\mathbf{z}$ and noise $\boldsymbol{\mu}$:

$$\tau\dot{\mathbf{u}} = -\mathbf{u} + \mathbf{W}_{\mathrm{rec}}\tanh(\mathbf{u}) + \mathbf{W}_{\mathrm{fb}}\mathbf{z} + \mathbf{W}_{\mathrm{attr}}\mathbf{x}$$
$$\mathbf{z} = \mathbf{W}_{\mathrm{ro}}\tanh(\mathbf{u}) + \boldsymbol{\eta}$$

- the task is to approximate $\mathbf{f}(t)$ with $\mathbf{z}(t)$
- attractor input is task-specific
- only readout weights $\mathbf{W}_{\mathrm{ro}}$ are plastic

Reward-modulated Hebbian rule [2]:

$$\Delta\mathbf{W}_{\mathrm{ro}}(t) = \eta(t)M(t)(\mathbf{z}(t) - \bar{\mathbf{z}}(t))\mathbf{r}^{\top}(t), \quad \eta(t) = \eta_0/(1 + t/\tau_\eta)$$

where $\bar{x}$ denotes low-pass filtered x ($\Rightarrow \mathbf{z}(t) - \bar{\mathbf{z}}(t) \approx \boldsymbol{\eta}(t)$). Reward modulation $M(t)$ is **binary** and tracks performance $P(t)$ with the Heaviside step function Θ as:

$$M(t) = \Theta(P(t) - \bar{P}(t)), \quad P(t) = -\|\mathbf{f}(t) - \mathbf{z}(t)\|^2$$

Conclusions

Key results

- attractor input makes reward-modulated Hebbian rule faster and more robust
- it works on harder and longer tasks, when the pure reservoir network fails completely
- it also works when weight updates are applied at the end of each training period
- attractor network exhibits a set of stable, task-specific trajectories without learning
- those trajectories are diverse enough to learn multiple tasks with the same readout weights

Possible extensions

- attractor network can be combined with a transient input signal to decouple the "elapsed time" input (attractor) and the task-specific input

Experimental setup

Short and long tasks:

- 1D functions sampled from a Gaussian Process (matching the complexity of the few hand-picked tasks from [1, 2])
- 50 different tasks for each length, each for a new reservoir network (examples below)
- performance is measured with normalized cross-correlation (1 is the perfect match)

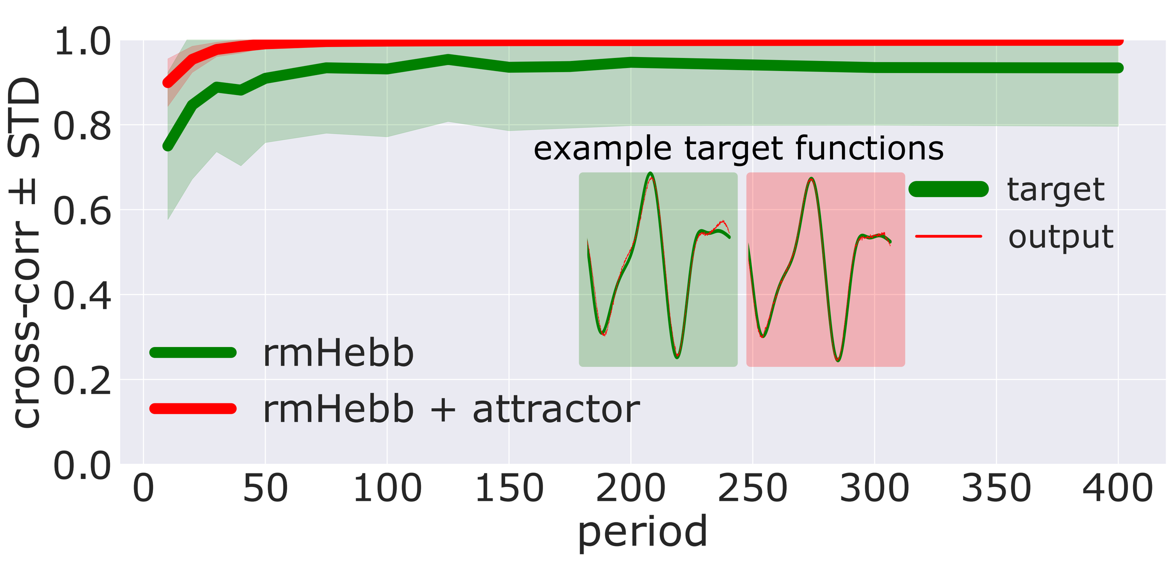
Digits task:

- 2D 1 second drawings of digits
- the attractor produces trajectories based on 50 similar inputs for each digit on train, and 50 on test

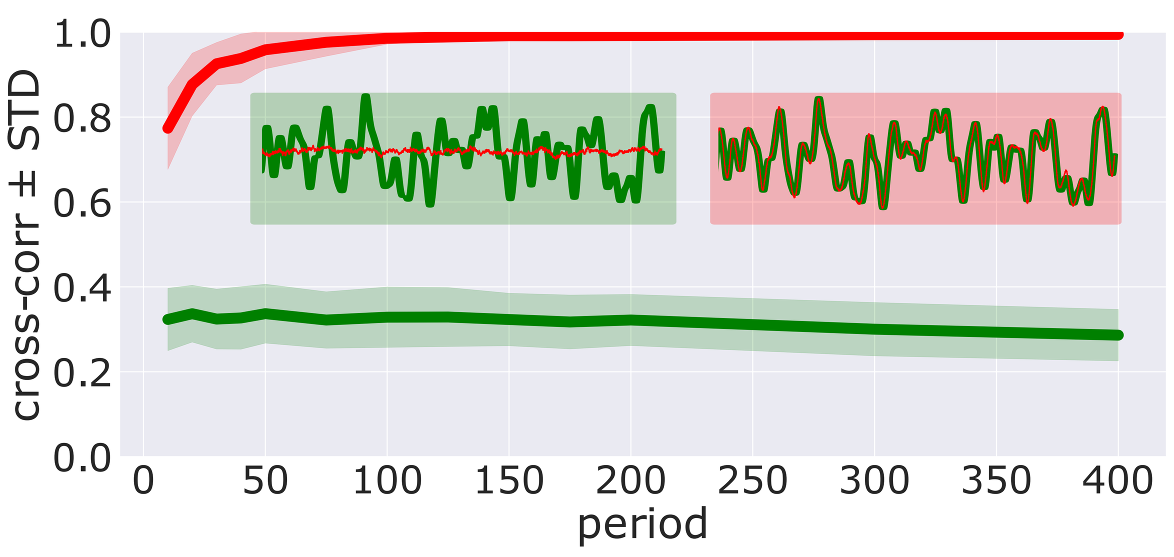
Simulation details:

- 2500 attractor neurons, 1000 reservoir neurons
- attractor networks is similar to [3], reservoir network matches [2]
- reservoir receives the same attractor input on each period (expect for the digits task)

Performance on short and long tasks



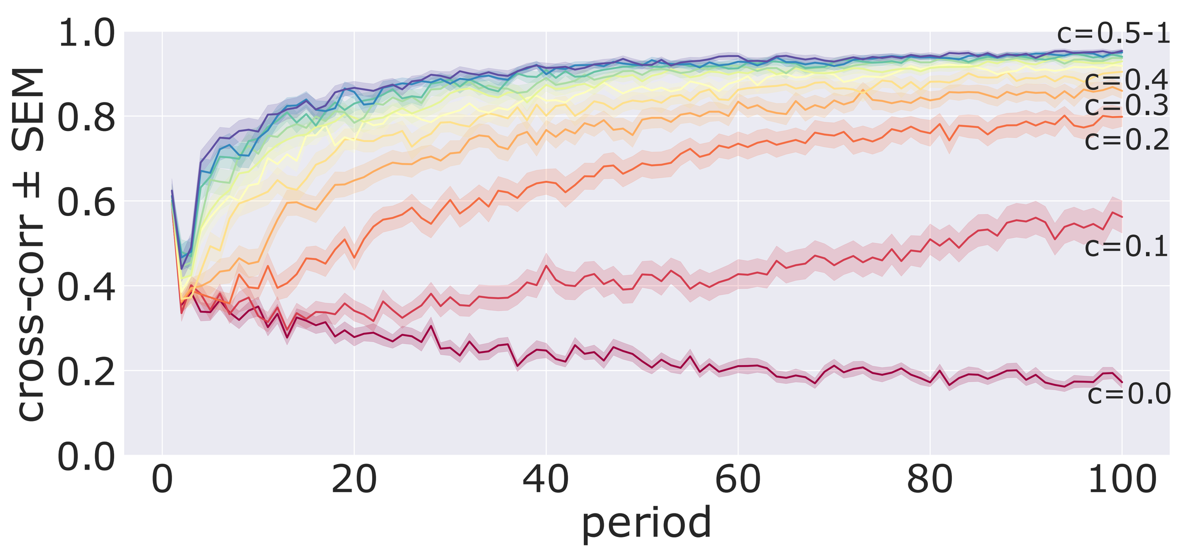
Mean performance on **1 second** tasks and an example task (insets).



Mean performance on **10 seconds** tasks and an example task (insets).

- on short **1 second** tasks attractor input makes the reward-modulated Hebbian rule (**rmHebb**) learn faster and better, reaching nearly perfect performance
- on longer **10 seconds** tasks, the reservoir alone fails completely, while our model just takes more trials to master each task
- same performance on the hand-picked functions from [1, 2]; perfect performance with FORCE

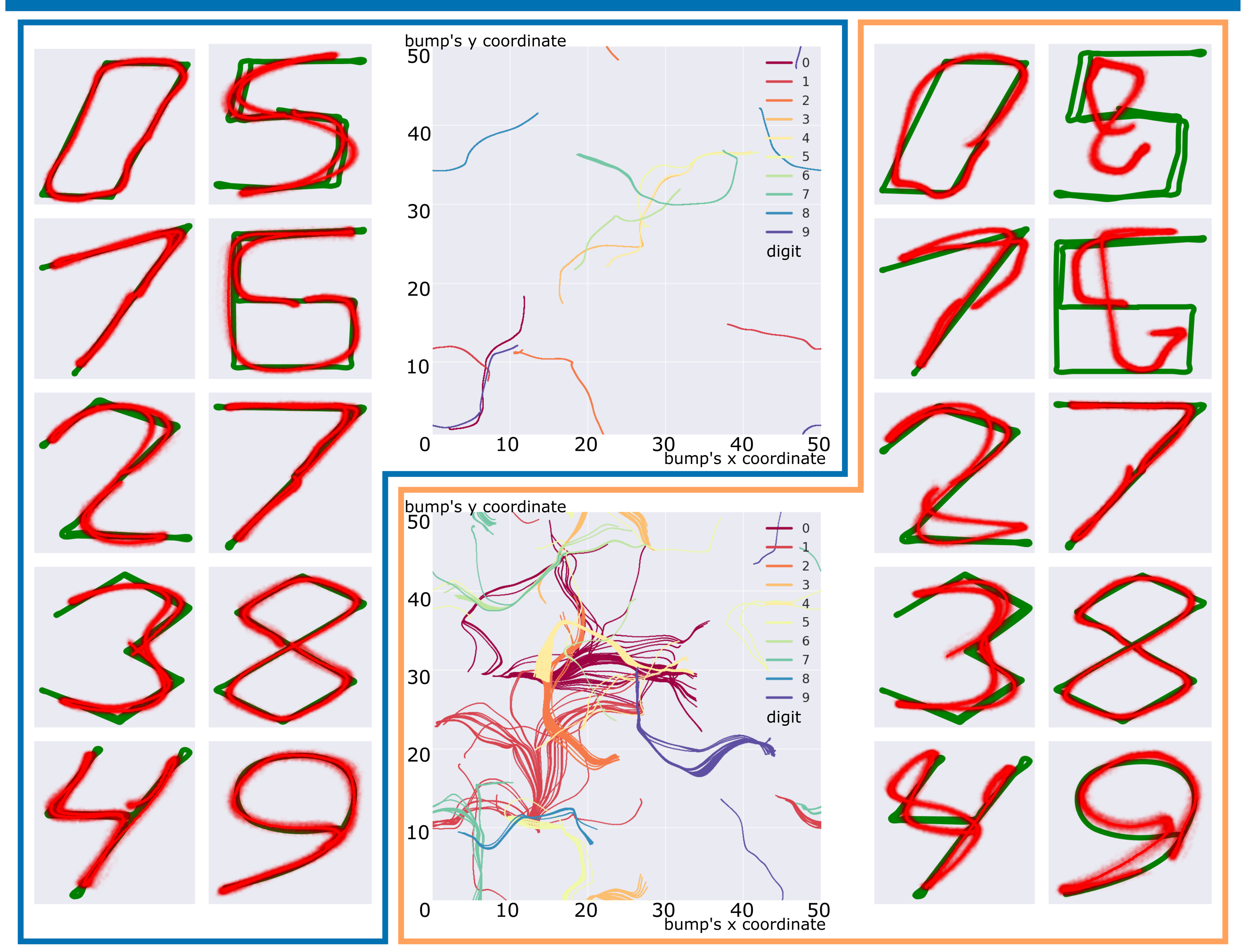
Delayed updates on short tasks



Mean performance with delayed updates on 1 second tasks for different attractor coupling $c = [0, 1]$.

- weight updates are summed in the background and only applied after each 1 second period
- for strong coupling with the attractor c (the reservoir receives $c\mathbf{W}_{\mathrm{attr}}\mathbf{x}$), performance is still high

Learning of multiple signals (the digits task)



Targets (green) and outputs (red) with **noiseless** (left) and **noisy** (right) attractor trajectories, as learned by a single set of readout weights. FORCE with noisy input performed as well as the reward-modulated Hebbian rule

Contact information & acknowledgments

Email: roman.pogodin.17@ucl.ac.uk

This research was supported by the Gatsby Charitable Foundation, Swiss National Science Foundation (no. 200020 - 184615) and by the European Union Horizon 2020 Framework Program under grant agreement no. 785907 (HumanBrain Project, SGA2). The authors thank Moritz Deger for an earlier version of the reservoir code and Jorge Aurelio Menendez for useful comments on the manuscript.

References

[1] D. Sussillo and L.F. Abbott. *Neuron*, 63:544–557, 2009.
[2] G.M. Hoerzer, R. Legenstein, and W. Maass. *Cerebral cortex*, 24:677–90, 2014.
[3] V. Itskov, C. Curto, E. Pastalkova, and G. Buzsaki. *Journal of Neuroscience*, 31:2828–2834, 2011.