LETTER **Communicated by Stefano Fusi**

Low-Dimensional Manifolds Support Multiplexed Integrations in Recurrent Neural Networks

Arnaud Fanthomme
arnaud.fanthomme@phys.ens.fr
Rémi Monasson
remi.monasson@phys.ens.fr
Laboratoire de Physique de l'Ecole Normale Supérieure PSLand CNRS UMR 8023, Sorbonne Université, 75005 Paris, France

We study the learning dynamics and the representations emerging in recurrent neural networks (RNNs) trained to integrate one or multiple temporal signals. Combining analytical and numerical investigations, we characterize the conditions under which an RNN with n neurons learns to integrate $D(\ll n)$ scalar signals of arbitrary duration. We show, for linear, ReLU, and sigmoidal neurons, that the internal state lives close to a D-dimensional manifold, whose shape is related to the activation function. Each neuron therefore carries, to various degrees, information about the value of all integrals. We discuss the deep analogy between our results and the concept of mixed selectivity forged by computational neuroscientists to interpret cortical recordings.

1 Introduction

Recurrent neural networks (RNNs) have emerged as a versatile and powerful architecture for supervised learning of complex tasks from examples, involving, in particular, dynamical processing of temporal signals (Chung, Gulcehre, Cho, & Bengio, 2014). Applications of RNNs or their variants designed to capture very long-term dependencies in input sequences through gating mechanisms, such as gated recurrent units (GRU) or long-short term memory (LSTM), are numerous and range from state-of-the-art speech recognition networks (Amodei et al., 2015) to protein sequence analysis (Almagro Armenteros, Sonderby, Sonderby, Nielsen, & Winther, 2017).

How these tasks are actually learned and performed has been extensively studied in the reservoir computing setup where the recurrent part of the dynamics is fixed (see Tanaka et al., 2019, for a review), while the general case of RNNs remains mostly an open question. Understanding those networks would bring valuable advantages to both neuroscience and machine learning, as suggested in Barak (2017) and Richards et al. (2019). Some results have been recently obtained, when the representations and the

https://doi.org/10.1162/neco_a_01366

dynamics are low dimensional (Sussillo & Barak, 2012; Mastrogiuseppe & Ostojic, 2018; Schuessler, Dubreuil, Mastrogiuseppe, Ostojic, & Barak, 2020; Schuessler, Mastrogiuseppe, Dubreuil, Ostojic, & Barak, 2020), a prominent feature of the neural integrators that are the focus of this study. Neural integrators, whose function is to perform integration of time series, have been studied for several decades in neuroscience, both experimentally (Robinson, 1989; Aksay et al., 2007; Wong & Wang, 2006) and theoretically (Elman, 1990; Seung, 1996; Lee, Reis, Seung, & Tank, 1997), and, more recently, numerically, in the context of machine learning (Song, Yang, & Wang, 2016).

The goal of our study is three-fold. First, we study how exactly the task of integration is learned from examples by RNNs. We derive rigorous results for linear RNNs and approximate ones for nonlinear RNNs, which can be compared to numerical simulations. This approach is similar to the one adopted by Saxe, McClelland, and Ganguli (2014) for the case of deep networks, and more recently by Schuessler, Mastrogiuseppe et al. (2020) in the case of recurrent networks. Second, we seek to understand the nature of the internal representations built during learning, an issue of general interest in neural network–based learning (Li, Monroe, & Jurafsky, 2017; Zhang & Zhu, 2018; Montavon, Samek, & Müller, 2018; Olah et al., 2018). It is, in particular, tempting to compare representations emerging in artificial neural networks to their natural counterparts in computational neuroscience. Third, we do not limit ourselves to a single integration, but consider the issue of learning multiple integrators within a unique network. While we do not expect an increase in performance for each individual task, as was observed in the case of natural language processing by Luong, Le, Sutskever, Vinyals, and Kaiser (2016), we are interested in finding representations adequate for parallel computations within a single network, allowing for considerations on the topic of "mixed selectivity" developed in computational neuroscience and studied by Rigotti et al. (2013). The issue of network capacity, the maximum number of tasks that can be performed in parallel, has been previously studied numerically (Collins, Sohl-Dickstein, & Sussillo, 2017), but remains out of the scope of this study, which focuses on a number D of integrals small compared to the network size.

Our letter is organized as follows. We define the RNNs we consider in this work, the integration task, and the training procedure in section 2. The case of linear activation function is studied in detail in section 3. RNNs with nonlinear activation functions are studied in section 4 in the case of a single channel ($D = 1$), and our results for the general situation of multiple channels ($D \geq 2$) are presented in section 5. Conclusions and perspectives are in section 6. The letter is complemented by a series of appendixes containing details about the calculations, simulations and further figures. The source code for the simulations can be found at https://github.com/AFanthomme/ManifoldsSupportRNI.

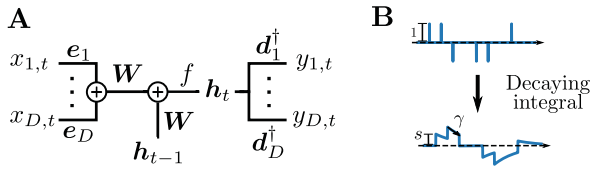

Figure 1: (A) Multiplexed recurrent neural network, with D input channels (left) and the same number of output channels (right). The internal state of the RNN, $\boldsymbol{h}_t$, is a vector of dimension n. The inputs are encoded by the vectors $\boldsymbol{e}_c$ and decoded from the internal state through the decoder weights $\boldsymbol{d}_c$. (B) Illustration of the decaying integral mapping that we want networks to approximate, on a sparse input sequence. At each time step t, if the input time series x_t is nonzero, the integral is increased by $s\,x_t$; then it is multiplied by $\gamma < 1$, which produces the exponential decay in the absence of inputs. In practice, sequences used for experiments were gaussian noise, and these sparse sequences are used only for visualization.

2 Definitions and Model

2.1 Description of the Network. We consider a single-layer RNN of size n, without any gating mechanism; while such refinements are found to improve performance (Lipton, Berkowitz, & Elkan, 2015), we omit them as they are not necessary for such a simple task. The computation diagram presented in Figure 1A can be summed up as follows: at time t, the scalar inputs along all channels $c = 1..D$, denoted $x_{c,t}$, are multiplied by their respective **encoder** vectors $\boldsymbol{e}_c$; these vectors are summed to the previous internal state $\boldsymbol{h}_{t-1}$[1] and multiplied by the weight matrix $\boldsymbol{W}$ before a componentwise activation f is applied to get the new internal state $\boldsymbol{h}_t$. The update equation for $\boldsymbol{h}$ is therefore

$$\boldsymbol{h}_t = f\left(\boldsymbol{v}_t\right), \tag{2.1}$$

where the current[2] $\boldsymbol{v}_t$ is defined through

$$\boldsymbol{v}_t = \boldsymbol{W} \cdot \left[\boldsymbol{h}_{t-1} + \sum_{c=1}^{D} x_{c,t}\,\boldsymbol{e}_c\right]. \tag{2.2}$$

[1] We initialize the internal state before any input to $\boldsymbol{h}_{-1} = \boldsymbol{0}$.

[2] This name is chosen in analogy with computational neuroscience, where W_{ij} is a synaptic weight from neuron j to neuron i, and the image of the activity vector, $v_i = \sum_j W_{ij} h_j$, represents the total current from the recurrent population coming onto neuron i.

The output units are linear: their values $y_{c,t}$ are simply obtained by taking the scalar product of $\boldsymbol{h}_t$ and the **decoder** vectors $\boldsymbol{d}_c$,

$$y_{c,t} = \boldsymbol{d}_c \cdot \boldsymbol{h}_t. \tag{2.3}$$

Most of this study FOCUSES on two activation functions f: the linear activation, which is simply the identity, and the ReLU nonlinearity, which takes component-wise maximum of a vector and zero. Linear activation allows for exact results to be derived on both the learning dynamics and the structure of solutions, while the choice of ReLU will serve as an example of nonlinear activation that can be used to create perfectly generalizing integrators (at least in the $D = 1$ case) and show that the conclusions of the linear network study remain relevant. Finally, we propose a generic procedure to train an RNN with arbitrary nonlinearity f to perform multiplexed integrations, which we illustrate with success in the case of sigmoidal activation.

2.2 Description of the Task. The networks will be trained to map D input time series $(x_{c,t})_{t\in\mathbb{N}}$ to D output ones $(y_{c,t})_{t\in\mathbb{N}}$: for all channels $c = 1, \ldots, D$, the cth output should match the γ_c-discounted sum of the cth channel inputs, times the scale factor s_c:

$$\overline{y}_{c,t} = s_c \sum_{k=0}^{t} \gamma_c^{k+1}\, x_{c,t-k} \tag{2.4}$$

(see Figure 1B). The values of the decay constants γ_c are chosen in [0,1] to restrict memory to recent events and avoid instabilities.[3]

We quantify the performance of the network through the mean square error between the actual and target outputs across the D channels on training epochs of length (duration) T:

$$\mathcal{L} = \left\langle \sum_{c=1}^{D} \sum_{t=0}^{T-1} \left(y_{c,t} - \overline{y}_{c,t}\right)^2 \right\rangle_X . \tag{2.5}$$

2.3 Description of the Learning Procedure. Except when otherwise specified, the encoder **e** and decoder **d** will be considered as randomly fixed at network initialization and forced to be of unit norm. The reason for this hypothesis is two-fold. First, our focus of interest is how the network of

[3] If γ is chosen too close to 1, the network might during training have an effective "decay" larger than 1; in that case, the values of the outputs and the associated gradients become large (in particular when training on long input sequences), which can then be overcompensated and make the training divergent.

connections between neurons evolves during training and the nature of the solutions and representations obtained. The simplified setup allows for deeper mathematical analysis of the dynamics of the $\mathbf{W}$ than the general case, where all parameters of the network evolve simultaneously during training. Second, while the speed of convergence is positively affected by relaxing the constraint of fixing the decoder, numerical experiments indicate that the nature of the $\mathbf{W}$ network is qualitatively unchanged if $\mathbf{e}$ and $\mathbf{d}$ are also trained, in particular, when it comes to the way the integrals are represented.

For theoretical analysis, we train the recurrent weights W using gradient descent (GD) updates at learning rate η:

$$W_{ij}^{(\tau+1)} = W_{ij}^{(\tau)} - \eta \frac{\partial \mathcal{L}}{\partial W_{ij}}(W^{(\tau)}), \tag{2.6}$$

where τ is the discrete learning time. We also performed experiments using the nonlinear Adam optimizer (Kingma & Ba, 2017) to ensure robustness of our results with respect to the specific choice of optimization procedure. Numerical implementations were performed in Python, making extensive use of the Scipy (Virtanen et al., 2020) and Pytorch (Paszke et al., 2019) packages, respectively, for scientific computing and implementation of automatic differentiation and gradient descent optimization.

3 Case of Linear Activation

Throughout this section, we assume that the activation function f is linear. We start with the simplest case of a single channel ($D = 1$) and omit the subscript $c = 1$ below for simplicity; we study the case of multiple channels $D \geq 2$ in section 3.4.

As the network dynamics $\boldsymbol{h}_t \to \boldsymbol{h}_{t+1}$ is linear, the loss, equation 2.5, can be analytically averaged over the input data distribution. The computation is presented in appendix A and yields

$$\mathcal{L}(W) = \sum_{q,p=1}^{T} \boldsymbol{\chi}_{qp}(\mu_q - s\gamma^q)(\mu_p - s\gamma^p), \tag{3.1}$$

where

$$\mu_q = \boldsymbol{d}^\dagger W^q \boldsymbol{e} \tag{3.2}$$

will be hereafter referred to as the qth moment of W, and $\boldsymbol{\chi}$ is a positive-definite matrix, related to the covariance matrix of the inputs x_t.

The average loss implicitly depends on $\gamma, T, s, \boldsymbol{e}, \boldsymbol{d}$ and input correlations $\boldsymbol{\chi}$, which do not evolve during training and are therefore omitted from the argument. Since $\boldsymbol{\chi}$ is positive-definite, the global minimum of the loss is reached when the moments of $\boldsymbol{W}$ fulfill

$$\mu_q = s\gamma^q, \tag{3.3}$$

for all $q = 1, \ldots, T$. The same conditions are obtained for uncorrelated inputs, so we restrict to this case for the following numerical investigations.

The gradient of the averaged loss with respect to the weight matrix $\boldsymbol{W}$ can be computed (see appendix B), with the result

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = 2 \sum_{q,p=1}^{T} \chi_{qp} \, (\mu_q - s\gamma^q) \sum_{m=0}^{p-1} \sum_{\alpha=1}^{n} d_\alpha (W^m)_{\alpha i} \sum_{\beta=1}^{n} (W^{p-1-m})_{j\beta} \, e_\beta. \tag{3.4}$$

We emphasize that while the network update dynamics is linear, the training dynamics over $\boldsymbol{W}$ defined by equations 2.6 and 3.4 is highly nonlinear.

3.1 Conditions for Generalizing Integrators. Conditions 3.3 over the moments μ_q, with $q = 1, \ldots, T$, ensure that the RNN will perfectly integrate input sequences of length up to the epoch duration T. We call **generalizing integrator (GI)** an RNN such that these conditions are satisfied for **all** integer-valued q, ensuring perfect integration of input sequences of arbitrary length.

We now derive a set of sufficient and necessary conditions for a diagonalizable matrix $\boldsymbol{W}$ to be a GI.[4] Let us assume $\boldsymbol{W}$ is diagonalized as $\boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{-1}$, where the spectral matrix $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$ is diagonal and $\boldsymbol{P}$ is invertible, of inverse $\boldsymbol{P}^{-1}$. The moments of $\boldsymbol{W}$ can be expressed from the eigenvalues as

$$\mu_q = \boldsymbol{d}^\dagger \boldsymbol{P} \boldsymbol{\Lambda}^q \boldsymbol{P}^{-1} \boldsymbol{e} = \sum_{i=1}^{n} g_i \lambda_i^q \quad \text{with} \quad g_i = (\boldsymbol{P}^\dagger \boldsymbol{d})_i (\boldsymbol{P}^{-1} \boldsymbol{e})_i.$$

Obviously, a null eigenvalue does not contribute to the above sum; hence, the conditions that we obtain in the following apply only to nonzero eigenvalues. Our condition for null loss is that all of the aforementioned moments μ_q are equal to $s\,\gamma^q$.

[4] As the set of diagonalizable matrices is dense in the space of matrices, any nondiagonalizable matrix $\boldsymbol{W}$ can be made diagonalizable through the addition of an infinitesimal matrix; the moments of the resulting matrix are arbitrarily close to the ones of $\boldsymbol{W}$, which makes our results for diagonalizable matrices directly applicable to $\boldsymbol{W}$. See section 3.3.

The above set of conditions can be rewritten as follows. For any real-valued polynomial $Q(z)$ of degree less than or equal to T in z, such that $Q(0) = 0$, we have

$$\sum_i g_i \, Q(\lambda_i) = s \, Q(\gamma). \tag{3.5}$$

We can evaluate the previous equality for well-chosen polynomials. Let us consider one eigenvalue, say, λ_κ, assumed to be different from γ, and the Lagrange polynomial $Q(z)$ equal to one for $z = \lambda_\kappa$ and zero for $z = 0$, $z = \lambda_i \neq \lambda_\kappa$, and $z = \gamma$. Such a polynomial exists as soon as $T \geq n + 1$ in the general case where all eigenvalues are distinct from each other, 0, γ, and as soon as $T \geq r + 1$ if $n - r$ eigenvalues are equal, for example, to 0. Equality 3.5 gives

$$\sum_i g_i \, \delta_{\lambda_i, \lambda_\kappa} = 0,$$

where $\delta_{\cdot,\cdot}$ denotes the Kronecker delta. Therefore, any eigenvalue different from γ must satisfy an exact cancellation condition for the associated g coefficients, ensuring that it does not contribute to the network output. Similarly, a condition for the γ eigenvalue can be written to ensure that an input of magnitude 1 entails a change of magnitude s in the output.

The necessary and sufficient conditions for a diagonalizable matrix $\boldsymbol{W}$ to be a global minimum of the loss defined with $T \geq n + 1$ therefore read

$$\begin{cases} \sum_i g_i \, \delta_{\lambda_i, \gamma} = s \\ \forall \kappa \text{ s.t. } \lambda_\kappa \notin \{\gamma, 0\}, \quad \sum_i g_i \, \delta_{\lambda_i, \lambda_\kappa} = 0. \end{cases} \tag{3.6}$$

These conditions are in turn enough to guarantee that $\boldsymbol{W}$ is a global minimum of the loss for any value of T; hence, the generalizing integrators and the minima of the losses defined with $T \geq n + 1$ are equal.

Clearly, any global minimum of the averaged loss $\mathcal{L}$ experimentally obtained when using training sequences of length $T \geq n + 1$ is a GI. Networks trained with much shorter epochs can also be GIs if the rank of $\boldsymbol{W}$ remains small enough throughout the training dynamics. More precisely, if we assume we have found a minimum of the loss of rank $r \leq n$, it will be a GI as soon as $T \geq r + 1$. An important illustration is provided by the null initialization of the weights ($\boldsymbol{W}^{(\tau=0)} = \boldsymbol{0}$), which ensures that $\boldsymbol{W}$ remains of rank $r = 2$ at all times τ (see equation 3.4 and the following section).

3.2 Special Case of Null-Weight Initialization. We now assume that the weight matrix $\boldsymbol{W}$ is initially set to zero and characterize all the GIs

accessible through GD, as well as the local convergence to those solutions. A study of the full training dynamics for two special cases ($T = 1$ and $\boldsymbol{e} = \boldsymbol{d}$) is in appendix C.

3.2.1 Low-Rank Parameterization. From the expression of the gradients (see equation 3.4) and the linearity of the weight updates (see equation 2.6), it is clear that starting from $\boldsymbol{W} = \boldsymbol{0}$, the weight matrix will remain at all times τ in the subspace generated by the four rank-1 matrices $\boldsymbol{d}\boldsymbol{d}^\dagger$, $\boldsymbol{d}\boldsymbol{e}^\dagger$, $\boldsymbol{e}\boldsymbol{d}^\dagger$, and $\boldsymbol{e}\boldsymbol{e}^\dagger$. We introduce an orthonormal basis for the $\boldsymbol{v}_1 \equiv \boldsymbol{e}$, $\boldsymbol{v}_2 \equiv \boldsymbol{d}$ space,

$$\overline{\boldsymbol{v}_a} = \sum_{b=1}^{2} (\boldsymbol{\Sigma}^{-1/2})_{ab}\, \boldsymbol{v}_b, \quad \text{with} \quad \boldsymbol{\Sigma}_{ab} = \boldsymbol{v}_a^\dagger \boldsymbol{v}_b, \tag{3.7}$$

and the corresponding parameterization of the subspace spanned by $\boldsymbol{W}$,

$$\boldsymbol{W}^{(\tau)} = \sum_{a,b=1}^{2} \omega_{ab}^{(\tau)}\, \overline{\boldsymbol{v}_a}\, \overline{\boldsymbol{v}_b}^\dagger, \tag{3.8}$$

where $\omega^{(\tau)}$ is a 2×2-matrix.

3.2.2 Generalizing Integrators. Conditions 3.6 for $\boldsymbol{W}$ to be a GI can be turned into conditions over ω (see appendix D). Let us assume that ω is diagonalized through $\omega = \boldsymbol{P}_\omega \boldsymbol{\Lambda}_\omega \boldsymbol{P}_\omega^{-1}$ with $\boldsymbol{\Lambda}_\omega = \mathrm{diag}(\lambda_1, \lambda_2)$, and define $g_i = (\boldsymbol{P}_\omega^\dagger \sqrt{\boldsymbol{\Sigma}})_{i,1} (\boldsymbol{P}_\omega^{-1} \sqrt{\boldsymbol{\Sigma}})_{i,2}$. The conditions for ω to define a GI through (equation 3.8) are ($\lambda_i = \gamma$ for $i = 1$ or 2), ($\sum_i \delta_{\gamma,\lambda_i} g_i = s$), and $g_i = 0$ if $\lambda_i \notin \{0, \gamma\}$. Taking into account the constraint $g_1 + g_2 = \boldsymbol{\Sigma}_{1,2} = \boldsymbol{d}^\dagger \boldsymbol{e}$, we find that the set of GIs is spanned by the following three manifolds in the four-dimensional space of ω matrices (see appendix E for details):

- The first manifold is of dimension 2 and contains rank-1 integrators $\boldsymbol{W}$ at all scales. These weight matrices have one eigenvalue equal to γ and the other to 0 so that one of the g coefficients remains unconstrained:

$$\omega = \frac{\gamma}{\beta - \alpha} \begin{pmatrix} \beta & -1 \\ \alpha\beta & -\alpha \end{pmatrix}, \tag{3.9}$$

  where $(\alpha, \beta) \in \mathbb{R}^2$. Fixing the scale s to any value different from $\boldsymbol{d}^\dagger \boldsymbol{e}$ introduces exactly one relation between α and β, making the set of rank-1 perfect integrators at scale s a one–dimensional manifold (see appendix E).
- The other two manifolds contain rank-2 integrators, operating at the scale $s^* = \boldsymbol{d}^\dagger \boldsymbol{e}$ only. For generic independent encoder and decoder

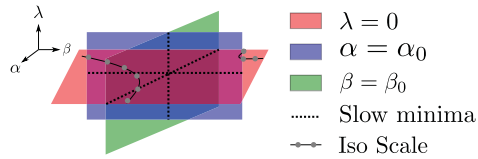

Figure 2: Illustration of the three *GI* manifolds in the space of 2×2 matrices with one eigenvalue equal to γ, the second to λ, and the remaining two degrees of freedom being labeled α and β. In one manifold (red), the second eigenvalue is zero, so that all of those matrices are GIs with decay γ and any scale s. The other two manifolds contain integrators at the particular scale $s^* = \boldsymbol{d}^\dagger \boldsymbol{e}$ only and are of rank 2. The values of α_0 and β_0 are computed in appendix E, where details on the parameterization used here can also be found.

vectors, the scale $s^* = \boldsymbol{d}^\dagger \boldsymbol{e}$ is of the order of $n^{-1/2}$ and vanishes in the large size limit. We discard these solutions and focus on rank-1 solutions given by (equation 3.9) at finite scale s ($\neq \boldsymbol{d}^\dagger \boldsymbol{e}$).

The structure of the GI manifolds is sketched in Figure 2.

In appendix F, we compute the gradient and Hessian of the loss in the null-initialization subspace. In the case of fixed encoder and decoder, the convergence toward a GI is generically exponentially fast; the corresponding decay time can be minimized by appropriately choosing the value s of the scale s (see appendix G). For some specific choices of the scale s, convergence can be much slower and exhibit an algebraic behavior (see appendix H).

3.3 Initialization with Full Rank Connection Matrices. The results above assumed that training started from a null weight matrix in order to constrain the dynamics of $\boldsymbol{W}$ to a very low-dimensional space. Training RNNs on very short epochs ($T = 3$) was then sufficient to obtain rank-1 GIs capable of integrating arbitrary long input sequences.

In practice, we observe that initializing the network with a matrix $\boldsymbol{W}$ of small spectral norm (instead of being strictly equal to zero) does not change the fact that only one of the eigenvalues of $\boldsymbol{W}$ is significantly altered during training, and a GI is obtained as soon as $T \geq 3$. The use of a nonlinear optimization scheme such as Adam rather than GD does not change this observation.

To gain insights into this empirical result, let us consider a perturbation $\boldsymbol{\epsilon} = \sum_i \epsilon_i \boldsymbol{u}_i \boldsymbol{v}_i$, with singular values bounded by one, around a generalizing integrator of rank 1, $\boldsymbol{W} = \sigma \boldsymbol{l} \boldsymbol{r}^\dagger$. Under the assumption that the $\boldsymbol{u}$ and $\boldsymbol{v}$ vectors are drawn randomly on the unit sphere of dimension n, their dot products with $\boldsymbol{e}$, $\boldsymbol{d}$ and each other are realizations of a centered gaussian

distribution of variance $1/n$. We can then consider the image of $\boldsymbol{e}$ by our perturbed matrix:

$$(\boldsymbol{W}+\boldsymbol{\epsilon})\boldsymbol{e} = (\sigma \boldsymbol{r}^\dagger \boldsymbol{e})\boldsymbol{l} + \sum_{i=1}^{n} \epsilon_i (\boldsymbol{v}_i^\dagger \boldsymbol{e})\boldsymbol{u}_i. \tag{3.10}$$

The second term, originating from the perturbation, is a vector whose components are sums of n terms of unfixed signs and magnitudes $1/n$, and is, hence, of the order $1/\sqrt{n}$. Accordingly, the dot product of this perturbation vector with $\boldsymbol{d}$, which is exactly the perturbation to the first moment μ_1, will be of the order of $1/\sqrt{n}$ too. Under a similar hypothesis of independence of gaussian vectors, all moments μ_q will be perturbed by terms of that same order.

Since unstructured eigenvectors do not contribute to the network output at first order, the gradients with respect to those parameters will also be subleading, and this perturbation will remain mostly unchanged during training, in agreement with numerical simulations.

3.4 Case of Multiple Channels. We have seen that GD is generally able to train a linear RNN exponentially fast toward a rank–1 single-channel GI with associated eigenvalue γ and singular vectors tuned to ensure the correct scale of integration. The state of the corresponding network is easily interpretable: it is, at all times, proportional to the output integral. Due to the linearity of the network, this result can be straightforwardly extended to the case of $D > 1$ integration channels, as we show below.

3.4.1 Interpretation of Rank–1 Solutions in the Single Channel Case. We write the rank–1 GI as $\boldsymbol{W} = \sigma \boldsymbol{l}\boldsymbol{r}^\dagger$, where $\boldsymbol{l}$ and $\boldsymbol{r}$ are normalized to 1 and σ is positive. Since $\boldsymbol{W}$ must have γ as its eigenvalue, we need $\sigma \boldsymbol{r}^\dagger \boldsymbol{l} = \gamma$. Additionally, to ensure that the first nonzero input gives the correct output, we require that $\sigma(\boldsymbol{d}^\dagger \boldsymbol{l})(\boldsymbol{r}^\dagger \boldsymbol{e}) = s\gamma$. It is easy to check that these conditions are sufficient to ensure that the state of the network is

$$\boldsymbol{h}_t = a\, \overline{y}_t\, \boldsymbol{l} \qquad \text{with} \qquad a = \frac{1}{\boldsymbol{d}^\dagger \boldsymbol{l}}, \tag{3.11}$$

for all times t, which in turn ensures perfect integration ($y_t = \overline{y}_t$). In other words, rank-1 GIs rely on a linear, one-dimensional representation of the target integral: the internal state is at all times proportional to $\overline{y}_t$.

3.4.2 Representation of Integrals with Multiple Channels. The above discussion of the single-channel case generalizes to multiple channels. Through training, a weight matrix $\boldsymbol{W}$ of rank D is constructed, which has $(\gamma_1, \ldots, \gamma_D)$

as its eigenvalues and singular vectors compatible with the (fixed) encoder and decoder weight vectors. The GI conditions are as follows:

$$\begin{cases} \forall c \in [\![1, D]\!], & \sigma_c \boldsymbol{r}_c^\dagger \boldsymbol{l}_c = \gamma_c \\ \forall c \in [\![1, D]\!], & \sigma_c \, (\boldsymbol{d}_c^\dagger \boldsymbol{l}_c) \, (\boldsymbol{r}_c^\dagger \boldsymbol{e}_c) = s_c \, \gamma_c \\ \forall (c, c') \in [\![1, D]\!]^2, c \neq c', & \boldsymbol{r}_c^\dagger \boldsymbol{e}_{c'} = 0 \\ \forall (c, c') \in [\![1, D]\!]^2, c \neq c', & \boldsymbol{d}_c^\dagger \boldsymbol{l}_{c'} = 0 \\ \forall (c, c') \in [\![1, D]\!]^2, c \neq c', & \boldsymbol{r}_c^\dagger \boldsymbol{l}_{c'} = 0 \end{cases} . \tag{3.12}$$

The two first conditions are exactly the same as in the single channel case, while the last three ensure that the modes of the weight matrix coding for the different integration channels c do not interfere and can independently update the values of their outputs to match the targets $\overline{y}_{c,t}$.

Assuming these conditions are satisfied, the network state is at time t equal to

$$\boldsymbol{h}_t = \sum_{c=1}^{D} a_c \, \overline{y}_{c,t} \, \boldsymbol{l}_c, \tag{3.13}$$

where the a_c's are structural coefficients, which generalizes expression 3.11 to the case $D \geq 2$. The state of any neuron i is therefore a linear combination of the D integrals across the multiple channels. Multiplexing is here possible as long as $D \leq n$ and encoders and decoders each form free families of $\mathbb{R}^n$.

4 Nonlinear Activation: Case of a Single Channel

We now turn to the case of nonlinear activation. The computation of the averaged loss is not analytically feasible any longer. However, by investigating RNNs trained with gradient descent on the mean square error (see equation 2.5) computed on batches of inputs, hereafter referred to as batch SGD, we have identified structural and dynamical properties from which sufficient conditions for generalization can be constructed.

4.1 Empirical Study of Neural Representations in an ReLU Network. We start by considering the case of the ReLU activation, where $f = \lfloor \cdot \rfloor_+ = \max(\cdot, 0)$ is a nonlinear component-wise operator. The simple encoding (see equation 3.11) adopted by linear-activation networks relied on the fact that the activity of each neuron could change sign with $\overline{y}_t$. This is not possible with ReLU activation any more since activities are forced to remain nonnegative, and a novel encoding is obtained after training of the RNNs that we explore below.

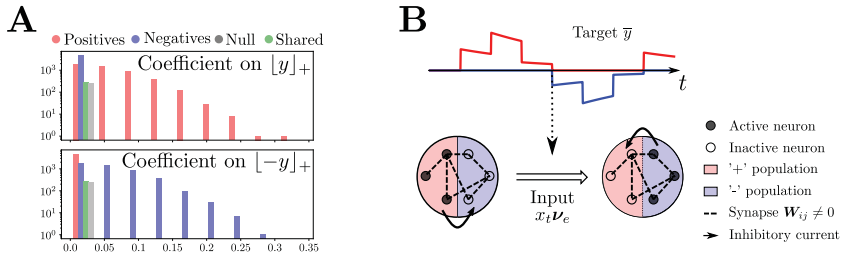

Figure 3: Internal encoding of the integral y_t by a single-channel ReLU network using two populations. (A) Experimentally observed distributions of the components of $\boldsymbol{L}_\pm$, determined by fitting the activity of each neuron with (equation 4.1). Results are aggregated across 10 realizations of batch SGD training $n = 1000$, $s = 1$. (B) Illustration of the activity shift from the $+$ to the $-$ population at arrival of an input that changes the sign of the target. Mutual inhibition between the two subnetworks guarantees that only one can be active at a given time, and an external input is required to perform the shift.

4.1.1 Behavior of Neuron Activities. Based on numerical simulations reported in Figure 3A, we argue that the population activity in ReLU networks depends on two vectors, referred to as $\boldsymbol{L}_+$ and $\boldsymbol{L}_-$, with non-negative components and dot products with $\boldsymbol{d}$ equal to, respectively, $+1$ and -1. More precisely, these vectors determine how the neural activities vary with the integral $\overline{y}_t$, depending on its sign:

$$\boldsymbol{h}_t = \lfloor \overline{y}_t \rfloor_+ \boldsymbol{L}_+ + \lfloor -\overline{y}_t \rfloor_+ \boldsymbol{L}_-. \tag{4.1}$$

Hence, in the space of possible internal states $\mathbb{R}^n_+$, the state $\boldsymbol{h}$ of the RNN lies in the union of the two half lines along $\boldsymbol{L}_+$ and $\boldsymbol{L}_-$, a one-dimensional piecewise linear manifold whose geometry is imposed by the nonlinear activation $\lfloor \cdot \rfloor_+$.

The n components of $\boldsymbol{L}_+, \boldsymbol{L}_-$ define a priori four subpopulations: if $(L_+)_i > 0$ and $(L_-)_i > 0$ neuron i is active at all times t ("shared" population); if $(L_+)_i > 0$ and $(L_-)_i = 0$ (respectively, $(L_+)_i = 0$ and $(L_-)_i > 0$), the neuron is actively when the integral is positive (resp. negative), defining the "+" (resp. "−") population; if $(L_+)_i = (L_-)_i = 0$, the neuron is never active and belongs to the "null" population. In numerical experiments, the shared and null populations account for a small fraction of the neurons (around 5%; see Figure 3A) when training is performed using batch SGD; in addition, shared neurons never have strong activities, and their contributions to the output integral seem irrelevant. We introduce in equation 4.10 a new loss function that allows for training of perfect integrators that do not exhibit any shared or null neurons.

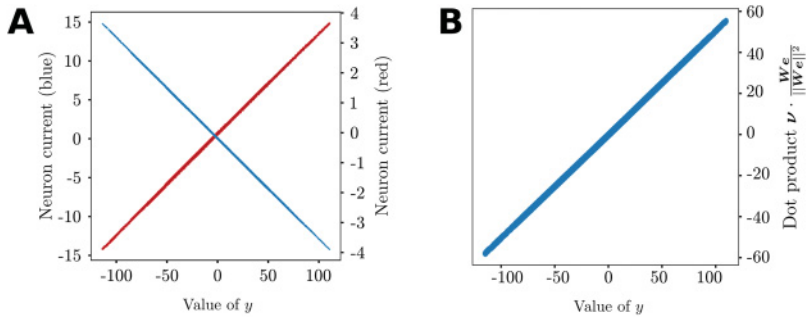

Figure 4: Behavior of currents in an ReLU network trained using the batch SGD loss, $s = 2$, $\gamma = 0.995$. (A) Parametric plot of the currents $(\boldsymbol{v}_t)_i$ incoming on two representative neurons i (red, blue) versus target integral y_t across time t. We observe a linear relation, with a slope that varies in both sign and magnitude from neuron to neuron. (B) Normalized dot product between the vector of currents $\boldsymbol{v}$ and the image of the encoder $\boldsymbol{We}$ versus value of the integral, illustrating equations 4.2 and 4.5.

4.1.2 Behavior of Neuron Currents. Numerical experiments furthermore indicate that the dependence of the current $\boldsymbol{v}_t$ (2.2) on the integral y_t is simpler than the one shown by the activity $\boldsymbol{h}_t$. We observe that the current vector is proportional to the integral,

$$\boldsymbol{v}_t = y_t \, \boldsymbol{L}, \tag{4.2}$$

where the components L_i of the vector $\boldsymbol{L}$ vary from neuron to neuron in both amplitude and sign (see Figure 4A).

The representation of the integral based on two nonoverlapping populations reported above may be seen as a straightforward consequence of the linear encoding at the level of preactivation currents expressed by equation 4.2:

$$\boldsymbol{h}_t = \lfloor \boldsymbol{v}_t \rfloor_+ = \lfloor y_t \, \boldsymbol{L} \rfloor_+ = \lfloor y_t \rfloor_+ \, \lfloor \boldsymbol{L} \rfloor_+ + \lfloor -y_t \rfloor_+ \, \lfloor -\boldsymbol{L} \rfloor_+, \tag{4.3}$$

from which we deduce that the population vectors $\boldsymbol{L}_+$ and $\boldsymbol{L}_-$ defined in equation 4.1 are equal to, respectively, $\lfloor \boldsymbol{L} \rfloor_+$ and $\lfloor -\boldsymbol{L} \rfloor_+$. In other words, neurons i encode positive or negative values of the integral depending on the signs of the components L_i.

Hence, while the neural state $\boldsymbol{h}_t = f(\boldsymbol{v}_t)$ of an ReLU RNN is not proportional to the integral value (see equation 4.1), as was the case for linear RNNs in section 3.3, proportionality is recovered at the level of the preactivation currents $\boldsymbol{v}_t$. We will see below that the linearity of the currents with respect to the integrals extends to the case of multichannel integrators.

4.2 Theoretical Analysis of the ReLU Integrators. We now explain the origin of the linear relationship between current and integral values (see equation 4.2), and how the vector $\boldsymbol{L}$ defining the current direction is related to the connection matrix $\boldsymbol{W}$, the encoder $\boldsymbol{e}$, and the parameters s, γ.

4.2.1 Sufficient Conditions for Integration. Let us first consider the network at time $t = 0$, with all activities set to zero ($\boldsymbol{h}_0 = \boldsymbol{0}$). As the first input x_1 is read by the encoder, the current vector at time $t = 1$ takes value

$$\boldsymbol{v}_1 = \boldsymbol{W}(\boldsymbol{0} + x_1\,\boldsymbol{e}) = x_1\,\boldsymbol{W}\boldsymbol{e} = \frac{\bar{y}_1}{s\gamma}\,\boldsymbol{W}\boldsymbol{e}. \tag{4.4}$$

The above equality agrees with the linear relationship 4.2, provided we have

$$\boldsymbol{L} = \frac{1}{s\gamma}\,\boldsymbol{W}\boldsymbol{e}. \tag{4.5}$$

This identity is in excellent agreement with numerical findings, as shown in Figure 4B.

We now assume that the current linearly expresses the target integral $\bar{y}_t$ at time t and look for sufficient conditions for relationship 4.2 to hold at time $t + 1$ after the new input x_{t+1} is received by the network. The current at time $t + 1$ reads

$$\begin{aligned}\boldsymbol{v}_{t+1} &= \boldsymbol{W}(\boldsymbol{h}_t + x_{t+1}\,\boldsymbol{e}) = \boldsymbol{W}(\lfloor \boldsymbol{v}_t \rfloor_+ + x_{t+1}\,\boldsymbol{e}) \\ &= \boldsymbol{W}\,(\lfloor \bar{y}_t\,\boldsymbol{L} \rfloor_+ + x_{t+1}\,\boldsymbol{e}) = \boldsymbol{W}(\lfloor \bar{y}_t \rfloor_+\,\boldsymbol{L}_+ + \lfloor -\bar{y}_t \rfloor_+\,\boldsymbol{L}_-) + x_{t+1}\,\boldsymbol{W}\boldsymbol{e} \\ &= \lfloor \bar{y}_t \rfloor_+\,\boldsymbol{W}\boldsymbol{L}_+ + \lfloor -\bar{y}_t \rfloor_+\,\boldsymbol{W}\boldsymbol{L}_- + x_{t+1}\,\boldsymbol{W}\boldsymbol{e},\end{aligned} \tag{4.6}$$

and should match

$$\boldsymbol{v}_{t+1} = \frac{\bar{y}_{t+1}}{s\gamma}\,\boldsymbol{W}\boldsymbol{e} = \left(\frac{\bar{y}_t}{s} + x_{t+1}\right)\boldsymbol{W}\boldsymbol{e} \tag{4.7}$$

according to equations 4.2 and 4.5. We deduce that $\boldsymbol{W}\boldsymbol{L}_+$ and $\boldsymbol{W}\boldsymbol{L}_-$ have to be aligned along $\boldsymbol{W}\boldsymbol{e}$ (see equation 4.5). Furthermore, based on the identity $y = \lfloor y \rfloor_+ - \lfloor -y \rfloor_+$, we readily obtain that

$$\boldsymbol{W}\boldsymbol{L}_+ = -\boldsymbol{W}\boldsymbol{L}_- = s^{-1}\,\boldsymbol{W}\boldsymbol{e}. \tag{4.8}$$

These relations are in very good agreement with numerics (see Figure 5).

4.2.2 Proxy Loss for Integration by a Network of ReLU Units. Conditions 4.5 and 4.8, as well as the relations between $\boldsymbol{L}, \boldsymbol{L}_+, \boldsymbol{L}_-$, ensure perfect generalizing integration. They can be summarized in the set of four equalities

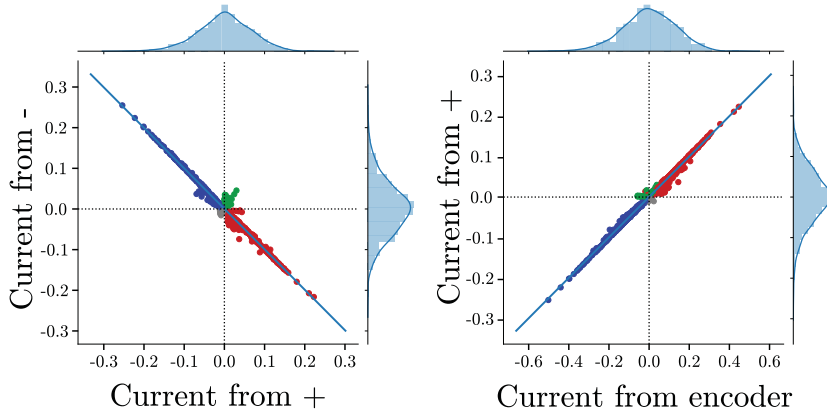

Figure 5: Contributions to the currents in an ReLU integrator trained with batch SGD. (Left) Scatter plot of $\boldsymbol{WL}_-$ versus $\boldsymbol{WL}_+$. (Right) $\boldsymbol{WL}_+$ versus $\boldsymbol{We}$. Colors refer to the neural populations (see Figure 3A). For both panels, we show on the sides the histograms of current components. Results were obtained with $T = 10$, $\gamma = 0.995, s = 2, n = 1000$. Numerical findings confirm that $\boldsymbol{WL}_+ = -\boldsymbol{WL}_-$ and $\boldsymbol{We} = s\,\boldsymbol{WL}_+$.

$$\begin{cases} \boldsymbol{d}^\dagger \lfloor \pm \boldsymbol{We} \rfloor_+ = \pm s\gamma \\ \boldsymbol{W} \lfloor \pm \boldsymbol{We} \rfloor_+ = \pm \gamma \boldsymbol{We} \end{cases} \tag{4.9}$$

linking the matrix of connections, the encoder and decoder vectors, and the scale and decay parameters.

We now introduce a proxy loss for $\boldsymbol{W}$, whose global minimum is achieved when conditions 4.9 are fulfilled,

$$\mathcal{L}^{proxy} = \sum_{z=\pm 1} (\boldsymbol{d}^\dagger \lfloor z\boldsymbol{We} \rfloor_+ - zs\gamma)^2 + \sum_{z=\pm 1} |\boldsymbol{W} \lfloor z\boldsymbol{We} \rfloor_+ - z\gamma \boldsymbol{We}|^2. \tag{4.10}$$

Experimentally, training on this proxy loss is extremely effective and, as expected, leads to perfect integrators satisfying the relations between currents shown in Figure 5. Similar to the linear case, if the encoder and decoder are fixed during training, the convergence time of GD is strongly dependent on s with a preferred scale around $|\boldsymbol{e}||\boldsymbol{d}|$ (see the study of dynamics of learning with $\mathcal{L}^{proxy}$ in appendix I.)

While the batch SGD loss is by definition based on actual computation of the network output for sample input sequences, the proxy loss imposes strict conditions on the dynamical behavior of the network that in turn ensure that the batch SGD loss will be zero. While there is no a priori reason to believe that all global minima of equation 2.5 are global minima of

equation 4.10, we empirically observed that the solutions $\boldsymbol{W}$ found by minimizing the batch SGD seemed to also be approximate minima of the proxy loss (see Figure 5 for the ReLU case).

4.2.3 Properties of the Connection Matrix. Training integrators with either batch SGD or the proxy loss yields solutions with one dominant singular value, of the form

$$\boldsymbol{W} \simeq \sigma\, \boldsymbol{l}\, \boldsymbol{r}^{\dagger}.$$

We report some properties of these solutions in appendix J. In particular, the singular value σ is, in the case of fixed encoder and decoder with unit norms, bounded from below by $2\max(1, s)$, where s is the scale. In practice, except for scales close to 1, this lower bound seems to be tight, that is, $\sigma = 2\max(1, s)$ (see appendix J, Figure 17). We interpret this saturation as a manifestation of the conjecture by Arora, Cohen, Hu, and Luo (2019) that gradient descent implicitly favors solutions with small matrix norm, as rank–1 matrices have a Frobenius norm equal to their singular value.

4.3 Case of Generic Nonlinear Activation. We now turn to the generic case of nonlinear activation function f. To do so, we show how the idea of proxy loss developed in the ReLU case can be naturally extended to any f.

4.3.1 Generic Proxy Loss. We start by writing, for an arbitrary activation function f, the dynamical equation for the current rather than for the activity state:

$$\boldsymbol{v}_{t+1} = \boldsymbol{W}\big(f(\boldsymbol{v}_t) + x_{t+1}\boldsymbol{e}\big). \tag{4.11}$$

At the first time step, since $\boldsymbol{h}_{-1} = \boldsymbol{0}$, the current will be equal to $\boldsymbol{v}_0 = x_0\boldsymbol{W}\boldsymbol{e} = y_0/(s\gamma)\boldsymbol{W}\boldsymbol{e}$. The error will thus vanish if, and only if, for all y in the range of values of the target integral,

$$\boldsymbol{d}^{\dagger} f\left(\frac{y}{s\gamma}\,\boldsymbol{W}\boldsymbol{e}\right) = y. \tag{4.12}$$

These relations generalize the first two conditions in equation 4.9 for ReLU activation. Furthermore, imposing that $\boldsymbol{W}\boldsymbol{e}$ is an "eigenvector" of the nonlinear operator $\boldsymbol{W}f(\cdot)$ with eigenvalue γ,

$$\boldsymbol{W}\cdot f\left(\frac{y}{s\gamma}\,\boldsymbol{W}\boldsymbol{e}\right) = \frac{y}{s}\,\boldsymbol{W}\boldsymbol{e}, \tag{4.13}$$

for any y, will force the current to remain at any time aligned along $\boldsymbol{We}$. A simple inductive proof similar to equation 4.6 shows that in these idealized conditions, the coordinate along that line will evolve proportionally to the output, similar to equation 4.2. Combined with the condition derived for the first input, this is enough to guarantee perfectly generalizing integration.

For arbitrary f, conditions 4.12 and 4.13 can generally not be exactly satisfied for y varying over a continuous domain, that is, for an infinite number of values of y. However, these conditions can be fulfilled for a discrete and finite subset, which will provide sufficient accuracy for good integration in practice, and we observe that the error on the integral of a time series of T inputs to scale as $\epsilon \sim n^{-1/2}$, irrespective of T (as long as the integral values remains below $y_{\max}$).

Based on these considerations, we propose a proxy loss for integration of a single scalar signal using an RNN with arbitrary nonlinearity:

$$\mathcal{L}^{proxy,\,f,D=1}(\boldsymbol{W}) = \int_{z\in Z} \left[\boldsymbol{d}^{\dagger} f(z\boldsymbol{We}) - s\,\gamma\,z\right]^2 + \left[\boldsymbol{W}\cdot f(z\,\boldsymbol{We}) - z\,\gamma\,\boldsymbol{We}\right]^2. \tag{4.14}$$

This integral can be estimated via Monte Carlo, and the choice of $Z = [-z_{\max}, z_{\max}]$ will restrict the maximum value $y_{\max} = s\,\gamma\,z$ of y that can be represented through our network. It is still possible to obtain generalization to an infinite number of integration steps, but the choice of γ has to be tuned so that the integral never exceeds the range the network was trained for.

4.3.2 Application to Sigmoidal Activation. We tested this new loss with a sigmoidal activation function,[5]

$$f : x \to \frac{1}{1+\exp^{-50(x-0.1)}}.$$

Trained with a decay $\gamma = 0.8$, scale $s = 1$, $Z = [-5, 5]$,[6] those networks converge to a solution with a single dominant singular value and manage to

[5] The choice of the slope and bias, here 50 and 0.1 respectively, is not critical to the results. We chose the slope so that the transition from 0 to 1 of the sigmoid happens on a scale of 1/50, close to the expected magnitude of the currents $n^{-1/2} \simeq 1/30$ for $n = 1000$. The bias was then chosen so that $x = 0$ is not in the linear portion of the sigmoid, nor in a fully saturated portion to avoid the null weight-matrix $\boldsymbol{W}^{(0)} = \boldsymbol{0}$ to be a fixed point of the learning dynamics.

[6] For $\gamma = 0.8$, $s = 1$, and inputs of magnitude bounded by 1, the integral evolves in $[-4, 4]$ as $y_{\max}$ is solution of $y_{\max} = \gamma(y_{\max} + s)$, hence $z_{\max} = y_{\max}/(s\gamma) = 5$. In practice, to observe the regimes $|y| \simeq y_{\max}$ more easily, we test the network using sequences alternating between bursts of ± 1 inputs and long periods with no external input, see Figure 9.

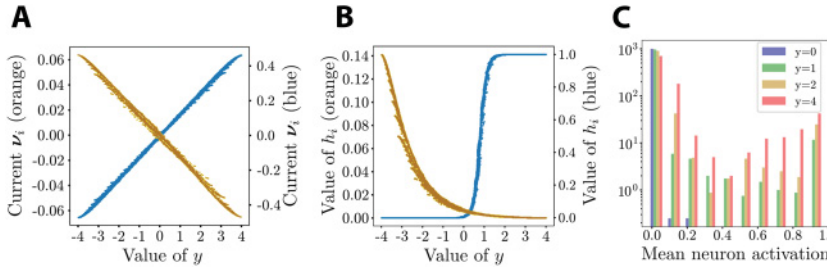

Figure 6: (A) Value of the preactivation current ν_i as a function of the integral for two representative neurons. (B) Activity-integral characteristic curve for the neurons of panel A. One of them (blue, right scale) saturates for low enough values of y_t, while the other (orange, left scale) never saturates. (C) Histogram of the mean activity of neurons for different values of the integrals, aggregated across eight realizations of the training on the proxy loss (see equation 4.14). The range of integral values $[-4, 4]$ was divided in 100 bins to select the time steps in the test sequences that corresponded to the values of y indicated in the legend. As the value of the integral increases, more neurons get strongly activated and eventually saturate. The same evolution could be observed for integrals y_t decreasing below the zero value. Those networks were trained using the batch SGD loss, $\gamma = 0.8$, $s = 1$, $n = 1000$, and the same results are found using the proxy loss.

integrate signals of arbitrary length, despite their inability to generalize to larger values of the integral. We observe that some neurons in the network exhibit a saturated behavior when the integral is above (resp. below) a neuron-specific threshold θ_i, while other neurons never reach that saturation. This results in a behavior where, during monotonous evolution of the integral starting from 0, an increasing number of neurons get activated to support the integral (see Figure 6). While these networks have a very different phenomenology from the ReLU ones in state-space, the integration is still performed through linear currents. We also confirmed that sigmoidal networks could be trained on the batch SGD loss, yielding integrators with a single dominant singular value; training with γ too close to 1 results in poor performance, suggesting that the issues of generalization to large values of y are not entirely due to the choice of proxy loss, but could hint at intrinsic limitations of the network, related to the activation function.

The proxy loss, equation 4.14, will be extended below to the general case $D > 1$. It should be noted that all nonlinear integrators need not be absolute minima of the proxy loss and follow the linear current representation. We only show here that it is one possible representation scheme, which can be adapted to any nonlinearity and could therefore help bridge the gap between idealized ReLU activation and more complex examples (e.g., inspired from real neurons).

5 Nonlinear Activation: Case of Multiple Channels

We now consider the case of a multiplexed integrator with D input-output channels, performing D integrals in parallel. In practice, numerical experiments were carried out for $D = 2, 3, 4$.

5.1 Batch and Proxy Losses for Multiple Integrators. To train our RNN to carry out multiple integrations, we followed two strategies. First, we used the batch loss defined in equation 2.5 from a set of input data, combined with a learning algorithm, *e.g.* SGD.

Second, drawing our inspiration from the detailed analysis of the single-channel case studied in the previous section, we introduced an extension of the proxy loss (see equation 4.14) to an arbitrary number $D > 1$ of input signals,

$$\mathcal{L}^{proxy,f,D}(\boldsymbol{W}) = \int_{z_1 \in Z_1} \cdots \int_{z_D \in Z_D} \left\{ \sum_c \left[\boldsymbol{d}_c^\dagger f\left(\sum_c z_c \boldsymbol{W} \boldsymbol{e}_c \right) - s_c \, \gamma_c \, z_c \right]^2 + \left[\boldsymbol{W} \cdot f\left(\sum_c z_c \boldsymbol{W} \boldsymbol{e}_c \right) - \sum_c \gamma_c \, z_c \boldsymbol{W} \boldsymbol{e}_c \right]^2 \right\}, \tag{5.1}$$

where the integral runs overs the D-dimensional range of values of the integrals, $Z_1 \times Z_2 \times \cdots \times Z_D$. As we shall see, training with this loss allowed us to obtain networks with arbitrary nonlinearity that represent the integral values linearly in the space of currents, as we shall see. Note that different activation functions, varying from neuron to neuron, could be also considered, for example, through the introduction of a distribution of thresholds for the sigmoidal function.

5.2 Characterization of Currents for ReLU Networks. We start with the ReLU case. As in the linear case, training ReLU networks with stochastic gradient descent of the batch loss yields networks that perform multiple integrations with excellent accuracy. Inspection of the connection matrices $\boldsymbol{W}$ reveals that they have D dominant singular values, as illustrated in Figure 7A for $D = 3$ channels. Such a spectral structure, consisting of a large number of "bulk" values and a small number of "outliers" that perform a computational task is reminiscent of the setting investigated in Schuessler, Dubreuil et al. (2020) and Schuessler, Mastrogiuseppe et al. (2020).

The D corresponding left eigenvectors $\boldsymbol{l}_c$ of the $\boldsymbol{W}$ matrix define a D-dimensional linear manifold for the current vector $\boldsymbol{v}_t$,

$$\boldsymbol{v}_t \simeq \sum_{c=1}^{D} \alpha_{c,t} \, \boldsymbol{l}_c, \;\; (\alpha_{1,t}, .., \alpha_{D,t}) \in \mathbb{R}^D, \tag{5.2}$$

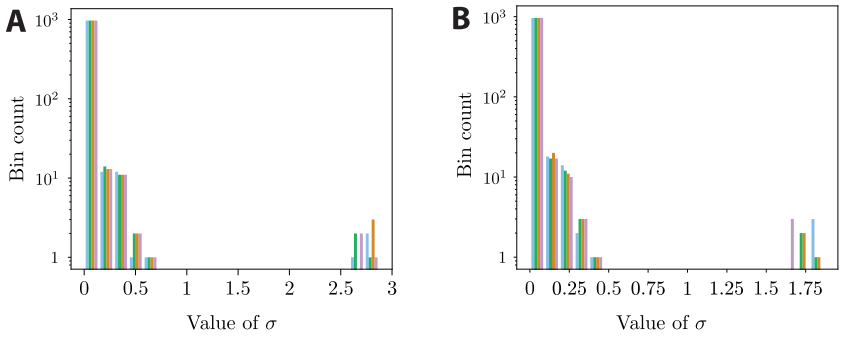

Figure 7: Histograms of the singular values of $\boldsymbol{W}$ in an ReLU (A) and sigmoidal (B) network across four realizations (one color each) of batch SGD with $D = 3$, $T = 10$, $n = 1000$. The ReLU networks were trained with $\gamma_1 = \gamma_2 = .995$, $\gamma_3 = .992$, while the sigmoidal ones were trained with $\gamma_1 = \gamma_2 = .8$, $\gamma_3 = .75$. In both cases, a bulk of eigenvalues are found close to 0, while exactly three of them become substantially larger. A fair amount of variability can be observed in the exact value of those large eigenvalues, even using the same values of the decays.

while the activity state $\boldsymbol{h}_t$ of the network lives on a nonlinear version of this manifold, shaped by the ReLU activation function,

$$\boldsymbol{h}_t = \lfloor \boldsymbol{v}_t \rfloor_+. \tag{5.3}$$

Investigating the relation between the α coordinates in the current manifold and the values of the different integrals $\overline{\boldsymbol{y}}$, we empirically find that they are related by a linear mapping. More precisely, there exists a $D \times D$-matrix $\boldsymbol{R}$ such that the coordinates $\boldsymbol{\alpha}_t$ along the current-manifold can be written at all times as

$$\alpha_{c,t} = \sum_{c'=1}^{D} R_{c,c'}\, \overline{y}_{c',t}. \tag{5.4}$$

In Figure 8, we illustrate this mapping in the $D = 2$ case. The methodology adopted is the following. While the network is performing integration, at each time step t, we infer the $\alpha_{c,t}$ coordinates from the values of the currents through equation 5.2. The panels of Figure 8 show the coordinate α_c (left: $c = 1$; right: $c = 2$; see the color code in the figure) as a function of the two integrals $\overline{y}_1, \overline{y}_2$. Aggregating those results across a large number of long trajectories, we find that the value of the currents as a function of the targets is independent of the exact input sequence and linearly depends on the value of the integrals. Hence, the linear dependence of the current on

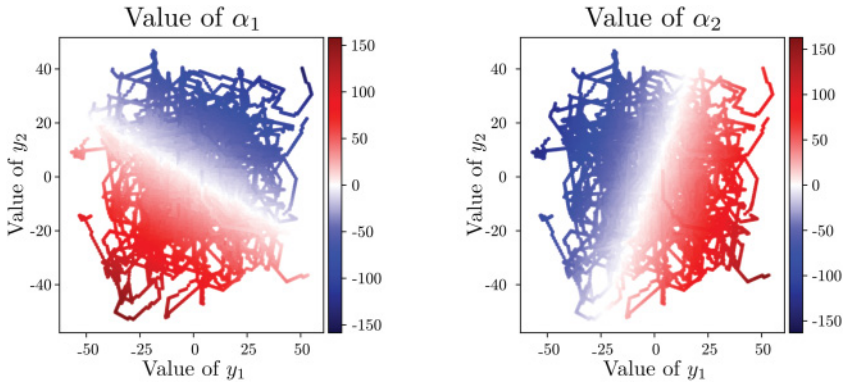

Figure 8: Value of the coordinates α_1 and α_2 in the current manifold as a function of the value of the target outputs $\bar{\boldsymbol{y}}$. Both coordinates depend linearly on the value of the two integrals (y_1, y_2), so that the position in the current manifolds is a linear representation of the integrals. The points were aggregated across 256 trajectories of length $T^{test} = 200$, for networks trained using batch SGD on the mean square error, equation 2.5, with training epochs duration $T^{train} = 10$, $\gamma_1 = 0.995$, $\gamma_2 = 0.992$.

the integrals, empirically found for $D = 1$ in equation 4.2, also holds in the multichannel case.

We emphasize that the presence of a bulk of small, but not negligible, singular values of $\boldsymbol{W}$ (in addition to the D dominant ones) is not in contradiction with the fact that the current lives in a D-dimensional manifold. The corresponding singular vectors may be orthogonal to the encoders and therefore never contribute to the internal state. To illustrate this point, we provide a quantitative evaluation of the distance between the currents $\boldsymbol{v}_t$ and the D-dimensional vector space $\mathcal{D}$ spanned by the D largest singular vectors $\boldsymbol{l}_c$ on the right-hand side of equation 5.2 as follows. After collecting the currents $\boldsymbol{v}$ at all time steps during 128 trajectories of duration $T = 200$, we compute the projection $\boldsymbol{v}_t^{\parallel}$ of those currents on $\mathcal{D}$ using least squares and the orthogonal projection, $\boldsymbol{v}_t^{\perp}$. The ratio of their norms

$$r = \frac{\left\langle |\boldsymbol{v}_t^{\perp}| \right\rangle_t}{\left\langle |\boldsymbol{v}_t^{\parallel}| \right\rangle_t}, \tag{5.5}$$

where $\langle \cdot \rangle_t$ denotes the average over time, estimates how much of the current lies out of the D-dimensional manifold. Results for the ratios are reported in the first line of Table 1 for networks obtained from the batch and the proxy losses and are very small, $r < 0.5$. These values are significantly smaller than what would be expected by chance in a null model in

Table 1: Average Ratios r of the Projections of the Current Outside and Inside the Best D-Dimensional Subspace (see Equation 5.5) for Different Activation Functions and Values of D, and $n = 1000$.

	Batch		Proxy			
$\times 10^{-2}$	D = 1	D = 2	D = 1	D = 2	D = 3	D = 5
ReLU	1.3 ± 0.2	1.3 ± 0.1	1.63 ± 0.1	4.9 ± 0.5	3.3 ± 0.5	2.5 ± 0.1
Sigmoid	5.6 ± 1.5	33.4 ± 1.9	3.22 ± 0.2	11.3 ± 1.4	10.0 ± 0.4	5.9 ± 0.3

Note: Errors were estimated from eight realizations of the training in the same conditions, and all values reported in the table are $10^2 \times r$ for readability.

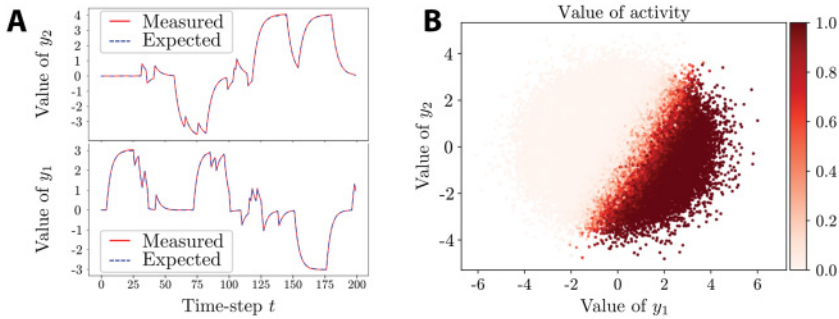

Figure 9: Learning of D-dimensional integrators with sigmoidal networks. (A) Comparison between expected and measured output on structured test sequences, designed to alternate between bursts of ± 1 inputs and long periods with no external input to allow for visual discrimination of the origin of errors between scale and decay. (B) Activity of a representative neuron in the (y_1, y_2) plane, measured on white-noise inputs. The decays are equal to 0.8 and 0.75, $n = 1000$, and the sigmoidal networks were trained using the proxy loss, equation 5.1.

which all directions in the n-dimensional space of currents would be equally significant,

$$r_{null} = \sqrt{\frac{n}{D} - 1}, \tag{5.6}$$

whose value is larger than 20 for $n = 1000$ and $D = 1, 2$.

5.3 Case of Sigmoidal Units. We have repeated the above analysis on networks with sigmoidal units, trained both from the batch and proxy losses. Results for a representative networks trained with the proxy loss to integrate $D = 2$ channels are shown in Figure 9A. We observe an excellent

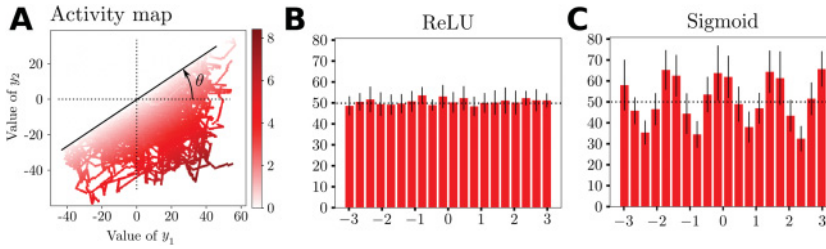

Figure 10: Mixed selectivity in bichannel integrators. (A) Activity h_i of a representative neuron i in a ReLU network as a function of the two integrals, aggregated across 512 epochs of $T^{test} = 200$ time steps. This activity is of the form $\max(\boldsymbol{s}_i^\dagger \boldsymbol{y}, 0)$, meaning that the neuron will only ever be active in half of the (y_1, y_2) plane. (B) Distribution of the angle of the boundary plane between zero and nonzero activity across the $n = 1000$ neurons of a ReLU network. (C) Distribution of the angle in the case of a sigmoidal network. Horizontal dotted lines represent the uniform distribution. Same parameters as in Figure 9; histograms were aggregated across 16 repetitions of the training.

match between the output integrals and their target values. Similar results, albeit less accurate, are obtained with the batch loss.

As in the ReLU case, the connectivity matrix $\boldsymbol{W}$ is characterized by D large singular values and a bulk of smaller ones. This bulk is influenced by several factors, including the initial condition over the matrix W and the choice of the learning algorithm. Despite the presence of these small singular values, the D-dimensional nature of the current can be assessed (see the ratios r reported in Table 1). The values of r are much smaller than what would be expected from a null model and confirm the low dimensionality of the current manifold. Not suprisingly, the values of the ratios for sigmoidal networks are 2 to 10 times larger than for their ReLU counterparts (for the same size n), as expected from the higher difficulty-to-solve conditions, equations 4.12 and 4.13 (see the discussion in section 4.3).

5.4 Nature of Single-Neuron Activity and Mixed Selectivity. The above findings allow us to determine how the state h_i of a neuron depends on the integrals $\bar{\boldsymbol{y}} = (\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_D)$ in an ReLU network:

$$h_i = \lfloor \boldsymbol{s}_i^\dagger \bar{\boldsymbol{y}} \rfloor_+, \text{ with } s_{i,c} = \sum_{c'} R_{c',c}\, \boldsymbol{l}_{c',i}. \tag{5.7}$$

From a geometrical point of view, as illustrated in Figure 10A in the $D = 2$ case, each neuron activity h_i is the image through the ReLU nonlinearity of the dot product between an associated direction $\boldsymbol{s}_i$ and the set of integrals $\bar{\boldsymbol{y}}$. The same feature is encountered for sigmoidal units, as shown in

Figure 9B. We have then characterized the distribution of the angular direction of s_i across the n neurons and find that it is equally distributed on $[0, 2\pi]$ when the network activation is ReLU, while it shows clear peaks for multiples of $\pi/2$ in the case of sigmoidal activation (see Figures 10B and 10C).

In the solutions empirically obtained through gradient descent, either on the batch loss or the proxy loss and for any number D of channels, we found that the network jointly encodes information about all integrals in the state of all neurons, a phenomenon similar to the one of mixed selectivity used to interpret cortical recordings in the field of computational neuroscience (Rigotti et al., 2013) and closely related to the issue of class selectivity in computer vision (see Leavitt & Morcos, 2020a).

Mixed selectivity can be seen here as being deeply connected to the choice of the input and output layers of the network. In our experiments, all encoders and decoders have nonzero components on all neurons of the internal state. Therefore, during training, the connectivity matrix will be optimized in such a way that each of those neurons will extract and represent information about all integrals. If we instead constrain the encoder and decoder for each channel to have the same support, spanning only n/D neurons and nonoverlapping with the support for any other channel, we find that the obtained solutions do not exhibit mixed selectivity anymore; the connection matrices $\boldsymbol{W}$ are block-diagonal, indicating that the network subdivided into D independent populations, each responsible for the coding of one integral. Relaxing the support constraint on either the encoders or the decoders causes mixed specificity to reappear. Last of all, allowing the support of the channels to overlap causes the corresponding neurons to exhibit mixed selectivity, while the rest of the network remains simply selective. Those findings are illustrated in Figure 11.

We interpret this difference in behavior by the fact that the heavy constraints imposed between the encoders and decoders through their supports are enough to modify the energy landscape in such a way that the entropically favored connectivity matrices do not exhibit mixed selectivity anymore. None of these support constraints have a significant impact on the final performance or the learning dynamics, and only the topology of the connectivity matrix is affected. Finally, we note that the choice of activation function also influences the distribution of selectivity angle, a fact that cannot easily be understood from entropic considerations and could potentially be related to learning.

5.5 Learning with Sign-Constrained Connections. So far, the only biological constraint we have considered regarded the states of neurons, which were forced to remain positive through the use of the ReLU activation function in order to represent firing rates. We now introduce a constraint on the weight matrix $\boldsymbol{W}$ itself, corresponding to the observed division between

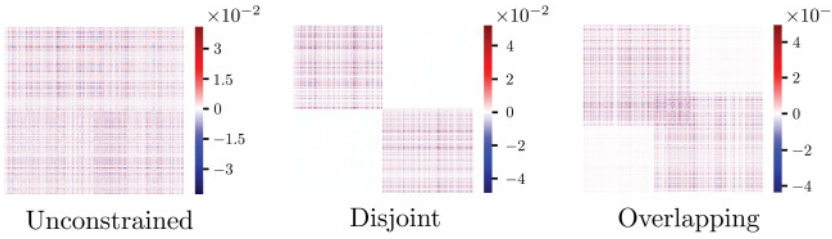

Figure 11: Visualization of the elements of the weight matrix W after training a ReLU network to integrate $D = 2$ signals through batch SGD in three cases of initialization: (left) the encoders and decoders are independent gaussian vectors without any restriction; (middle) the population is divided in two: the first half of the neurons have nonzero encoder and decoder only on channel 1, and similarly the other half on channel 2; and (right) starting from the nonoverlapping case, we allow a small fraction of the neurons (middle portion) to have nonzero components on all $\boldsymbol{e}, \boldsymbol{d}$ vectors. We find that the use of disjoint supports produces block-diagonal solutions where one population is in charge of one integral and isolated from the others, thus exhibiting single selectivity.

excitatory and inhibitory neurons known as Dale's law (Squire et al., 2012): at initialization, we fix a certain fraction of the columns of W, corresponding to the outgoing connections from a subpopulation of neurons, to have only negative entries, while the rest of the columns will have only positive entries. In order to maintain these constraints satisfied during training, after each step of optimization, we fix to 0 all the elements of W that changed sign.

At the end of the training, the weight matrices exhibit one additional relevant singular value compared to their unconstrained counterparts:

$$W \simeq \sigma_0 \, l_0 \, r_0^\dagger + \sum_{c=1}^{D} \sigma_c \, l_c \, r_c^\dagger.$$

The rank-1 contribution coming from this additional mode has the correct signs to satisfy Dale's constraint, as illustrated in Figure 12. Additionally, the left singular vector l_0 is almost orthogonal to all decoding vectors $\boldsymbol{d}_c$, suggesting that this mode is not used for the computation of the integrals, only as a way to satisfy the sign constraints over W. It should be noted that our empirical result does not rule out the existence of networks of rank D performing D multiplexed integrals while satisfying Dale's law. However, such solutions, if they exist, are not obtained through a simple gradient descent procedure from a zero or small W.

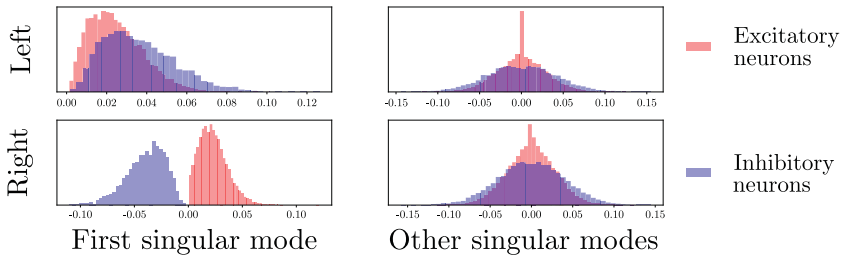

Figure 12: Distribution of the components of the left and right singular vectors for the largest singular value (left) and the following D ones (right). These histograms were obtained with 16 realizations of the batch SGD training, using $n = 1000$, $D = 2$, and 25% of inhibitory neurons. While the signs of the components of the second and third singular vectors appear random, they have a particular structure in the first singular vector: the left singular vector is always positive, while the right is positive (resp. negative) if the neuron is in the excitatory (resp. inhibitory) population; the corresponding rank-1 matrix has columns of fixed signs corresponding to the ones of Dale's constraints.

6 Conclusion and Perspectives

6.1 Summary of Results and Open Questions. We have studied in this work how an RNN with n neurons learns to perform one or more integrations of temporal inputs; each integration was characterized by the target values of the scale factor s and the decay coefficient γ (generally, slightly below 1).

In the case of an RNN with linear activation performing a single integral, we have precisely characterized the length of the temporal input necessary for perfect generalization (integration of any temporal signal), the optimal learning rate, and the convergence time of the training procedure when the weight matrix is initially set to zero (or is small enough in norm). The coding of the integral is realized in a simple way: the activity vector of the entire neural population varies along a one-dimensional direction in the n-dimensional space, with a proportionality factor equal to the integral.

In the case of ReLU activation, accurate integration was obtained at the end of the training too. While a full mathematical analysis seemed much harder than for linear activation, we showed empirical evidence for the fact that the activity vector belongs to a piecewise one-dimensional manifold. Coding of the positive and negative values of the integrals is done by two essentially nonoverlapping populations of neurons, switching on and off when the integral value crosses zero. Remarkably, the preactivation current of the ReLU units shows a simple behavior: it is proportional to the integral. We have derived sufficient conditions over the weight matrix for such

a coding to take place and characterized the nature (directions of left and right eigenvectors, amplitude of singular value as a function of s, γ) of the corresponding rank-1 integrator.

In the case of a multiplexed network with D input and output channels, we have found that the weight matrix is of rank D; this statement is exact for linear activation and approximately true for ReLU activation RNNs, whose weight matrix has D large singular values compared to the $n - D$ remaining ones. Consequently, the network activity is restricted to a D-dimensional manifold in $\mathbb{R}^n$, whose geometry is imposed by the activation function of the neurons. For ReLU activation, as in the single-integral case, strong empirical evidence suggests that the preactivation currents are linear combinations of the D integrals and span a D-dimensional linear subspace.

It is important to stress that the above results are not mere consequences of the threshold-linear nature of ReLU units. We have repeated our analysis with saturating units, obeying a sigmoidal activation function, with essentially the same results. Interestingly, some units never saturate for all possible values of the integral(s), other do, and all participate to produce the right outputs. To elucidate the reason for the D-dimensional nature of the coding of integrals by the currents, we have introduced a proxy loss reflecting sufficient conditions for such a coding. The networks trained from data (and the batch loss) behave similar to the networks minimizing this proxy loss from the point of view of both performance and representations.

From a purely machine-learning point of view, our work shows the versality of RNNs to achieve simultaneously several computational tasks. The variety of representations supporting these computations could then be harnessed for transfer learning (see Pan & Yang, 2010, for a review) by using our trained RNN as a (possibly fixed) feature extractor. One example of such a task is the one of context-dependent integration, studied in the prefrontal cortex of monkeys by Mante, Sussillo, Shenoy, and Newsome (2013), and which we adapt to our setup in appendix K. The proxy loss we derived could also a priori be used as part of a full-task loss, following a similar reasoning to Haviv, Rivkind, and Barak (2019), where one term in the loss is used to encourage internal dynamics that are known to be relevant for the task at hand and facilitate training.

Empirical analysis shows that very accurate multi-integrators with nontrivial activation functions can be obtained through gradient descent, and the representation scheme they adapt is linear in the space of currents. Three main limitations in this observation have to be noted. First, we do not show that this is the only representation scheme possible, and different solutions could possibly be found from pure mathematical reasoning. Second, rigorous analysis of the proxy loss remains necessary to understand in which conditions these representations are achievable, and to which accuracy. Finally, our study has focused on the case where the number of integrals D is small and the number of neurons n is large, and the question of how the optimal computational capacity (maximal sustainable value of D) precisely

increases with n remains to be understood in the case of RNNs with nonlinear activation.

6.2 Nature of Representations and Connection with Computational Neuroscience. While scalar integration using a single-layer recurrent network is far from state-of-the-art machine learning, the abundance of studies in the field of neuroscience (often motivated by the oculomotor system in fish) and the absence of a comprehensive theory of representation in such networks make it a worthwhile case study. Our theoretical analysis provides new evidence for the relevance of low-dimensional representations, and this result is robust to changes in the training method, the initial conditions of the weight matrix, the choice of activation function. Our work therefore provides additional motivation for the theoretical study of the properties of RNNs with low-rank coupling matrix initiated in the contexts of statistical physics (Mastrogiuseppe & Ostojic, 2018) and computational neuroscience (see Barak, 2017, for a review).

As far as neuroscience is concerned, we believe that our result about the encoding of multiple integrals by each neuron, expressed by equation 5.7, is of particular interest. There is, indeed, a very striking analogy between our findings and the concept of mixed selectivity used to interpret cortical recordings in the field of computational neuroscience (Rigotti et al., 2013). For a long time, neuroscientists have focused on neurons whose activities depended on a single sensory relevant variable, such as the orientation of a bar in the visual cortex area V1 or the animal's head direction in the subiculum (in our case, the value of one particular integral y_c). Such neurons are obviously easier to identify from activity recordings. However, there is growing recognition that most cells display mixed sensitivity, that is, have activities varying nonlinearly with several relevant variables, and that the relative degree of importance of each variable in determining the activity may vary considerably from neuron to neuron (as we find in Figure 10). Such mixed representations could be useful for decision making based on multisensorial streams of information, a possibility sometimes put forward to explain their relevance in neuroscience. It is, from this point of view, remarkable that mixed representations spontaneously emerge in our study, where the RNN lacks any explicit incentive to exploit them simply because they are much more likely than pure representations when the encoders and decoders have no intrinsic structure (see Figure 10). The computational advantages of such mixed representations have been studied in Leavitt and Morcos (2020a, 2020b) and suggest that they could improve both generalization and the robustness of the performed computations. Other studies have focused on the emergence of disentangled representations, which have been shown to be relevant in both natural language processing (Radford, Jozefowicz, & Sutskever, 2017) and computer vision (Denton & Birodkar, 2017; Lee, Tseng, Huang, Singh, & Yang, 2018), suggesting that the optimal type of representation might depend on the specific task it supports. Studying the

representations of computational tasks in artificial neural networks could therefore be a valuable tool to understand their biological counterparts, an approach already proposed in the domain of spatial navigation (Banino et al., 2018).

Appendix A: Fully Averaged Loss for Linear Single-Channel Integrators

For an arbitrary input sequence $(x_t)_{0\le t\le T-1}$, we compute through induction the value of the output at any time t as

$$\forall t \in \mathbb{N},\ y_t = \sum_{q=0}^{t} x_{t-q} \boldsymbol{d}^T \boldsymbol{W}^{q+1} \boldsymbol{e} := \sum_{q=0}^{t} x_{t-q} \mu_{q+1}.$$

The target output is $\overline{y}_t = s \sum_{q=0}^{t} x_{t-q} \gamma^{q+1}$, so that the square error is

$$\begin{aligned}
\epsilon_t^2 &= (y_t - \overline{y}_t)^2 \\
&= \left[\sum_{q=0}^{t} x_{t-q} (\mu_{q+1} - s\gamma^{q+1}) \right]^2 \\
&= \sum_{q,p=0}^{t} x_{t-q} x_{t-p} (\mu_{q+1} - s\gamma^{q+1})(\mu_{p+1} - s\gamma^{p+1}).
\end{aligned}$$

The loss to minimize is the average of the sum of those errors along input sequences of length T:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{W}) &= \left\langle \sum_{t=0}^{T-1} \epsilon_t^2 \right\rangle = \left\langle \sum_{t=0}^{T-1} \sum_{p,q=0}^{t} x_{t-q} x_{t-p} (\mu_{q+1} - s\gamma^{q+1})(\mu_{p+1} - s\gamma^{p+1}) \right\rangle \\
&= \left\langle \sum_{t=0}^{T-1} \sum_{p,q=0}^{T-1} x_{t-q} x_{t-p} \mathbf{1}_{q\le t} \mathbf{1}_{p\le t} \right\rangle (\mu_{q+1} - s\gamma^{q+1})(\mu_{p+1} - s\gamma^{p+1}) \\
&= \sum_{p,q=0}^{T-1} \left\langle \sum_{t=0}^{T-1} x_{t-q} x_{t-p} \mathbf{1}_{q\le t} \mathbf{1}_{p\le t} \right\rangle (\mu_{q+1} - s\gamma^{q+1})(\mu_{p+1} - s\gamma^{p+1}) \\
&:= \sum_{p,q=1}^{T} \boldsymbol{\chi}_{qp} (\mu_q - s\gamma^q)(\mu_p - s\gamma^p), \qquad \text{(A.1)}
\end{aligned}$$

where we introduced the time-integrated correlation matrix $\boldsymbol{\chi}$.

$\boldsymbol{\chi}$ is symmetric, and it is easily shown that

$$\forall \boldsymbol{v} \in \mathbb{R}^T, \boldsymbol{v}^\dagger \boldsymbol{\chi} \boldsymbol{v} = \sum_{p,q=0}^{T-1} v_q \chi_{qp} v_p = \sum_{p,q=0}^{T-1} \left\langle \sum_{t=0}^{T-1} v_q x_{t-q} x_{t-p} \mathbf{1}_{q\le t} \mathbf{1}_{p\le t} v_p \right\rangle$$
$$= \left\langle \sum_{t=0}^{T-1} \left(\sum_{q=0}^{T-1} v_q x_{t-q} \mathbf{1}_{q\le t} \right) \left(\sum_{p=0}^{T-1} x_{t-p} \mathbf{1}_{p\le t} v_p \right) \right\rangle$$
$$= \sum_{t=0}^{T-1} \left\langle \left(\sum_{p=0}^{t} x_{t-p}\, v_p \right)^2 \right\rangle .$$

Therefore, $\boldsymbol{\chi}$ is nonnegative. Assuming now that $\boldsymbol{v}$ is such that the quadratic form above vanishes, the term corresponding to $t = 0$, equal to $v_0^2 \langle x_0^2 \rangle$, vanishes, entailing that $v_0 = 0$ as soon as the input is assumed to have positive probability to be nonzero at this first time step. Then the $t = 1$ contribution, $\langle (x_0\, v_1 + x_1\, v_0)^2 \rangle = \langle (x_0\, v_1)^2 \rangle$ also vanishes, which implies that $v_1 = 0$. By recursion over t, all the components of $\boldsymbol{v}$ must vanish, which shows that $\boldsymbol{\chi}$ is definite positive.

Appendix B: Gradient and Hessian of the Linear Single Channel Loss

We have

$$\nabla_{W_{ij}} \mathcal{L} = \sum_{q,p=1}^{T} \chi_{qp} \left[(\mu_q - s\gamma^q) \frac{\partial \mu_p}{\partial W_{ij}} + (\mu_p - s\gamma^p) \frac{\partial \mu_q}{\partial W_{ij}} \right]$$
$$= 2 \sum_{q,p=1}^{T} \chi_{qp} (\mu_q - s\gamma^q) \frac{\partial \mu_p}{\partial W_{ij}}. \tag{B.1}$$

We compute through induction:

$$\frac{\partial W_{ij}^p}{\partial W_{kl}} = \sum_{m=0}^{p-1} W_{ik}^m W_{lj}^{p-1-m} \quad \text{hence} \quad \frac{\partial \mu_p}{\partial W_{kl}} = \sum_{i,j=1}^{n} \sum_{m=0}^{p-1} d_i W_{ik}^m W_{lj}^{p-1-m} e_j,$$

so that the gradient of $\mathcal{L}$ with respect to $\boldsymbol{W}$ is

$$\nabla_{W_{ij}} \mathcal{L} = 2 \sum_{q,p=1}^{T} \chi_{qp} (\mu_q - s\gamma^q) \sum_{m=0}^{p-1} \sum_{\alpha,\beta} (d_\alpha W_{\alpha i}^m)(W_{j\beta}^{p-1-m} e_\beta). \tag{B.2}$$

We now want to compute the Hessian $\mathcal{H}$ of this loss:

$$\mathcal{H}_{ij,kl} = \frac{\partial \mathcal{L}}{\partial W_{ij} \partial W_{kl}} = 2 \sum_{q,p=1}^{T} \chi_{qp} (\mu_q - s\gamma^q) \frac{\partial}{\partial W_{kl}} \left[\sum_{m=0}^{p-1} \sum_{\alpha,\beta} (d_\alpha W_{\alpha i}^m)(W_{j\beta}^{p-1-m} e_\beta) \right]$$
$$+ 2 \sum_{q,p=1}^{T} \chi_{qp} \frac{\partial \mu_q}{\partial W_{kl}} \sum_{m=0}^{p-1} \sum_{\alpha,\beta} (d_\alpha W_{\alpha i}^m)(W_{j\beta}^{p-1-m} e_\beta).$$

We will only be interested in the value of the Hessian at global minima of $\mathcal{L}$, so that the first term in this equation will not contribute. In that case, we find

$$\mathcal{H}_{ij,kl} = 2 \sum_{q,p=1}^{T} \chi_{qp} \left[\sum_{m=0}^{p-1} \sum_{\alpha,\beta} (d_\alpha W_{\alpha i}^m)(W_{j\beta}^{p-1-m} e_\beta) \right] \left[\sum_{\tilde{\alpha},\tilde{\beta}=1}^{n} \sum_{\tilde{m}=0}^{q-1} d_{\tilde{\alpha}} W_{\tilde{\alpha}k}^{\tilde{m}} W_{l\tilde{\beta}}^{q-1-\tilde{m}} e_{\tilde{\beta}} \right]. \tag{B.3}$$

Appendix C: Two Special Cases of Null Initialization

C.1 Exact Solution for $T = 1$. We begin by the simplest case possible, when the epochs are of length 1. The gradient descent updates become in that case

$$\Delta W_{ij} = -2\eta (\boldsymbol{d}^\dagger \boldsymbol{W} \boldsymbol{e} - s\gamma) d_i e_j.$$

Hence, we have at any time $\boldsymbol{W} = \omega \boldsymbol{d}\boldsymbol{e}^T$, so that we can study the optimization dynamics on the scalar ω only:

$$\Delta \omega = -2\eta (\omega ||\boldsymbol{e}||^2 ||\boldsymbol{d}||^2 - s\gamma).$$

Therefore, after τ steps of optimization, the coefficient ω is equal to

$$\omega^{(\tau)} = \frac{s\gamma}{||\boldsymbol{e}||^2 ||\boldsymbol{d}||^2} [1 - (1 - 2\eta ||\boldsymbol{e}||^2 ||\boldsymbol{d}||^2)^\tau].$$

This dynamics is stable if and only if $\eta < ||\boldsymbol{e}||^{-2} ||\boldsymbol{d}||^{-2}$. When it is, the network converges exponentially fast to $\boldsymbol{W} = \frac{\gamma s}{||\boldsymbol{e}||^2 ||\boldsymbol{d}||^2} \boldsymbol{d}\boldsymbol{e}^T$, which gives the following moments:

$$\forall k \geq 1, \mu_k = \gamma s \left(\frac{\gamma s \, \boldsymbol{e} \cdot \boldsymbol{d}}{||\boldsymbol{e}||^2 ||\boldsymbol{d}||^2} \right)^{k-1}.$$

Therefore, in that case, we converge toward a solution that achieves the desired scaling s, but has a decay constant that is fixed by the initial choice of s, $\boldsymbol{e}$, and $\boldsymbol{d}$.

C.2 Same Encoder and Decoder, $T > 1$, Uncorrelated Inputs. For this part, we assume that $\boldsymbol{e} = \boldsymbol{d}$. Considering that the first update in that case is proportional to $\boldsymbol{e}\boldsymbol{e}^T$, all subsequent ones are too, so we know that $\boldsymbol{W}^{(t)} = \omega(t)\boldsymbol{e}\boldsymbol{e}^\dagger$ and the dynamics can be studied on the scalar ω only.

Because of this, all moments are given by $\mu_k = \omega^k \|\boldsymbol{e}\|^{2k+2} = \|\boldsymbol{e}\|^2(\omega\|\boldsymbol{e}\|^2)^k$. The corresponding scale and decay are, respectively, $\|\boldsymbol{e}\|^2$ and $\omega\|\boldsymbol{e}\|^2$, and because of this, it is only possible to obtain a generalizing integrator at scale $s = \|\boldsymbol{e}\|^2$.

The fixed points of the gradient descent dynamics are the real roots of the following polynomial P:

$$\frac{\Delta\omega}{\eta} = 2\sum_{k=1}^{T} k(T+1-k)(\omega^k\|\boldsymbol{e}\|^{2k+2} - s\gamma^k)\omega^{k-1}\|\boldsymbol{e}\|^{2k-2} := P(\omega).$$

Choosing $s = \|\boldsymbol{e}\|^2$, we can check with Mathematica that for any value of T larger than 2, this polynomial has a single real root at $\omega = \gamma/\|\boldsymbol{e}\|^2$, which is a generalizable minimum. Since the leading-order term in this polynomial is of odd degree, we know that $\lim_{w\to\pm\infty} P(\omega) = \pm\infty$, and therefore the derivative of P at $\omega = \gamma/\|\boldsymbol{e}\|^2$ is positive, so that this value of ω is an attractive fixed point of the dynamics. As before, the dynamics is convergent only if the learning rate is smaller than $P'(\gamma/\|\boldsymbol{e}\|^2)^{-1}$.

Appendix D: Expression of the Moments in the Low Rank Parameterization

We have defined ω so that

$$\boldsymbol{W} = \sum_{a,b=1}^{2} \omega_{a,b}\overline{\boldsymbol{v}_a}\,\overline{\boldsymbol{v}_b}^\dagger.$$

Because of the orthogonality conditions $\overline{\boldsymbol{v}_a}^\dagger\overline{\boldsymbol{v}_b} = \delta_{a,b}$, we can easily compute the powers of $\boldsymbol{W}$ as

$$\boldsymbol{W}^k = \sum_{a,b=1}^{2} (\omega^k)_{a,b}\overline{\boldsymbol{v}_a}\,\overline{\boldsymbol{v}_b}^\dagger,$$

which yields the following moments:

$$\mu_k = \boldsymbol{d}^\dagger \boldsymbol{W}^k \boldsymbol{e} = \boldsymbol{d}^\dagger \sum_{a,b=1}^{2} \omega_{a,b}^k \overline{\boldsymbol{v}_a}\, \overline{\boldsymbol{v}_b}^\dagger \boldsymbol{e}$$

$$= \sum_{a,b=1}^{2} \sqrt{\boldsymbol{\Sigma}}_{1,a} \omega_{a,b}^k \sqrt{\boldsymbol{\Sigma}}_{b,2} = (\sqrt{\boldsymbol{\Sigma}} \omega^k \sqrt{\boldsymbol{\Sigma}})_{1,2}.$$

We now assume ω to be diagonalizable as $\omega = \boldsymbol{P}_\omega \boldsymbol{\Lambda}_\omega \boldsymbol{P}_\omega^{-1}$, so that

$$\begin{aligned}\mu_k &= (\sqrt{\boldsymbol{\Sigma}} \omega^k \sqrt{\boldsymbol{\Sigma}})_{1,2} = (\sqrt{\boldsymbol{\Sigma}} \boldsymbol{P}_\omega \boldsymbol{\Lambda}_\omega^k \boldsymbol{P}_\omega^{-1} \sqrt{\boldsymbol{\Sigma}})_{1,2} \\ &= \sum_{i=1}^{2} \lambda_i^k (\sqrt{\boldsymbol{\Sigma}} \boldsymbol{P}_\omega)_{1,i} (\boldsymbol{P}_\omega^{-1} \sqrt{\boldsymbol{\Sigma}})_{i,2} = \sum_{i=1}^{2} \lambda_i^k (\boldsymbol{P}_\omega^\dagger \sqrt{\boldsymbol{\Sigma}})_{i,1} (\boldsymbol{P}_\omega^{-1} \sqrt{\boldsymbol{\Sigma}})_{i,2} \\ &:= \sum_{i=1}^{2} g_i \lambda_i^k.\end{aligned}$$

Using a reasoning similar to the one of section 3.1, we find the same conditions (see equation 3.6) for generalizing integrators, but with a new expression of the g coefficients.

Appendix E: Generalizing Integrators in the Null Initialization Subspace

In this section, we seek to determine all matrices ω that correspond to generalizing integrators at decay γ. A first, obvious condition is that at least one of the eigenvalues of ω has to be equal to γ. Without loss of generality, we will consider this eigenvalue to be the first one, and we will denote the other λ.

The matrix $\boldsymbol{P}_\omega$ that diagonalizes ω can be parameterized as

$$\boldsymbol{P}_\omega = \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \end{pmatrix}.$$

Each column of $\boldsymbol{P}_\omega$ can be independently multiplied by a nonzero scalar and yield the same ω. There are therefore three cases: either u_1 or v_1 is null (but not both, since $\boldsymbol{P}$ would then not be invertible), or both are nonzero.

We also recall that

$$g_1 + g_2 = \sum_{i=1}^{2} [(\sqrt{\Sigma} \boldsymbol{P}_\omega)_{1,i} (\boldsymbol{P}_\omega^{-1} \sqrt{\Sigma})_{i,2}] = \Sigma_{1,2} = \boldsymbol{d}^\dagger \boldsymbol{e}. \tag{E.1}$$

E.1 Case $u_1 \neq 0$ and $v_1 \neq 0$. In that case, the modal matrix $\boldsymbol{P}_\omega$ of $\omega^\perp$ will be parameterized as follows, with $\alpha \neq \beta$ to ensure invertibility:

$$\boldsymbol{P}_\omega = \begin{pmatrix} 1 & 1 \\ \alpha & \beta \end{pmatrix}.$$

This results in the following parameterization of the space E_γ of 2×2 matrices with at least one eigenvalue equal to γ:

$$\begin{aligned} E_\gamma &= \left\{ \Gamma(\lambda, \alpha, \beta) = \begin{pmatrix} 1 & 1 \\ \alpha & \beta \end{pmatrix} \begin{pmatrix} \gamma & 0 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} 1 & 1 \\ \alpha & \beta \end{pmatrix}^{-1} ; (\lambda, \alpha, \beta) \in \mathbb{R}^3, \alpha \neq \beta \right\} \\ &= \left\{ \frac{1}{\alpha - \beta} \begin{pmatrix} \alpha\lambda - \beta\gamma & \gamma - \lambda \\ \alpha\beta(\lambda - \gamma) & \alpha\gamma - \beta\lambda \end{pmatrix} ; (\lambda, \alpha, \beta) \in \mathbb{R}^3, \alpha \neq \beta \right\}. \end{aligned} \tag{E.2}$$

E.1.1 If $\lambda = \gamma$. All values of α and β correspond to $\gamma \mathbf{1}$, a perfect integrator at scale $s^* = \boldsymbol{d}^\dagger \boldsymbol{e}$.

E.1.2 If $\lambda \neq \gamma$ and $\lambda \neq 0$. In that case, we need to impose $g_2(\alpha, \beta) = 0$; otherwise the second eigenvalue will contribute to the output and the system will not be a generalizing integrator. This will in turn impose that $g_1 = \boldsymbol{d}^\dagger \boldsymbol{e}$, and hence these will be integrators at scale $s = \boldsymbol{d}^\dagger \boldsymbol{e}/\gamma$. We find that

$$g_2[\alpha, \beta] = Z \frac{(\alpha - \beta_0)(\beta - \alpha_0)}{\alpha - \beta},$$

where

$$\begin{cases} \alpha_0 = -\dfrac{(\kappa - l_-)r_- + (\kappa + l_-)r_+}{2\boldsymbol{d}^\dagger \boldsymbol{e}(r_+ - r_-)} \\ \beta_0 = \dfrac{(\kappa + l_-)r_- + (\kappa - l_-)r_+}{2\boldsymbol{d}^\dagger \boldsymbol{e}(r_+ - r_-)} \\ Z = \dfrac{[\boldsymbol{d}^\dagger \boldsymbol{e}(r_+ - r_-)]^2}{2\kappa^2} \\ l_\pm = \|\boldsymbol{d}\|^2 \pm \|\boldsymbol{e}\|^2 \\ \kappa = \sqrt{l_-^2 + 4(\boldsymbol{d}^\dagger \boldsymbol{e})^2} \in]0, l_+[\\ r_\pm = \sqrt{l_+ \pm \kappa}. \end{cases}$$

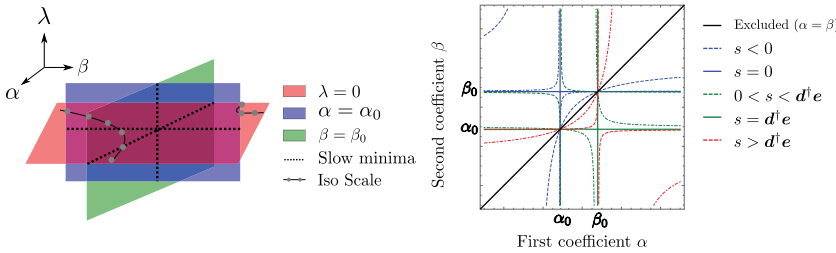

Figure 13: Structure of the minima in the null initialization subspace. On the left, we present the structure of the manifolds of 2×2 matrices with exactly one eigenvalue equal to γ and both eigenvectors with nonzero first components. The coefficients α and β parameterize the eigenvectors, and λ is the second eigenvalue. The slow minima, which are located at the intersections of our manifolds, are the ones toward which convergence of gradient descent can be algebraically slow (see appendix H). On the right, we detail the structure of the iso-scale manifolds in the $\lambda = 0$ submanifold and show that they are indeed one-dimensional as long as $s \notin \{0, \boldsymbol{d}^\dagger \boldsymbol{e}\}$. While the lines appear to cross, they do so only on the $\alpha = \beta$ line, which is a singularity of our parameterization and therefore nonphysical.

Hence, there are two manifolds of points satisfying $g_2(\alpha, \beta) = 0$:

$$\begin{cases} \mathcal{M}_\alpha = \{\Gamma(\lambda, \alpha, \alpha_0), (\lambda, \alpha) \in \mathbb{R}^2\}, \\ \mathcal{M}_\beta = \{\Gamma(\lambda, \beta_0, \beta), (\lambda, \beta) \in \mathbb{R}^2\}, \end{cases} \tag{E.3}$$

where Γ is defined in equation E.2. These two manifolds intersect along a one-Dimensional manifold:

$$\mathcal{M}_\alpha \cap \mathcal{M}_\beta = \{\Gamma(\lambda, \beta_0, \alpha_0), \lambda \in \mathbb{R}\}.$$

E.1.3 If $\lambda = 0$. The system will always be a perfect integrator at decay γ, since no other eigenvalue can contribute to the output. Its scale is determined by

$$s = g_1(\alpha, \beta) = Z \frac{(\alpha - \alpha_0)(\beta - \beta_0)}{\beta - \alpha}. \tag{E.4}$$

From this result, we deduce the following:

- For any given value of $s \notin \{0, \boldsymbol{d}^\dagger \boldsymbol{e}\}$, the set of values of α, β that give $c_1[\alpha, \beta] = s$ is a one-dimensional manifold, as can be seen in Figure 13. A parameterization of this manifold can be obtained by inverting equation E.4 as the set $\Gamma(0, \alpha, \beta_s(\alpha))$ where Γ is defined in

equation E.2 and

$$\beta_s(\alpha) = \frac{Z(\alpha-\alpha_0)\beta_0 - \alpha s}{Z(\alpha-\alpha_0) - s}.$$

- When $s = 0$, the two solutions are $\alpha = \alpha_0$ or $\beta = \beta_0$, no matter the value of the other parameter.
- When $s = \boldsymbol{d}^\dagger \boldsymbol{e}$, equation E.4 is not invertible and the condition $g_1 = s$ is satisfied if and only if $\beta = \alpha_0$ or $\alpha = \beta_0$, which are exactly the intersection of the $\mathcal{M}_\alpha$ (resp. $\mathcal{M}_\beta$) manifolds of equation E.3 with the $\lambda = 0$ subspace.

E.2 From ω to W. We have shown that in the generic case $s \notin \{0, \boldsymbol{d}^\dagger \boldsymbol{e}\}$, the generalizing integrators at scale s correspond to ω of rank 1 and form a one-dimensional manifold.

Such matrices ω can be parameterized as

$$\begin{aligned}\mathcal{M}_{\lambda=0} &= \left\{ \frac{\gamma}{\beta-\alpha} \begin{pmatrix} \beta & -1 \\ \alpha\beta & -\alpha \end{pmatrix}; (\alpha,\beta) \in \mathbb{R}^2, \alpha \neq \beta \right\} \\ &= \left\{ \sigma_\omega \boldsymbol{x}\boldsymbol{y}^\dagger; \sigma_\omega = \frac{\gamma\sqrt{(\alpha^2+1)(\beta^2+1)}}{\beta-\alpha}, \boldsymbol{x} = \frac{1}{\sqrt{\alpha^2+1}}(1,\alpha), \right. \\ &\qquad \left. \boldsymbol{y} = \frac{1}{\sqrt{\beta^2+1}}(\beta,-1), (\alpha,\beta) \in \mathbb{R}^2, \alpha \neq \beta \right\},\end{aligned}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are, respectively, the left and right eigenvector of the corresponding rank 1 matrix. When ω is of that form, we have

$$\boldsymbol{W} = \sigma_\omega \sum_{a,b} \boldsymbol{x}_a \boldsymbol{y}_b \overline{\boldsymbol{v}_a}\, \overline{\boldsymbol{v}_b}^\dagger = \sigma_\omega \left(\sum_a \boldsymbol{x}_a \overline{\boldsymbol{v}_a} \right) \left(\sum_b \boldsymbol{y}_b \overline{\boldsymbol{v}_b}^\dagger \right) := \sigma \boldsymbol{l}\boldsymbol{r}^\dagger,$$

so that $\boldsymbol{W}(\omega)$ is of rank 1 too.

E.3 Case $u_1 = 0$. The matrix $\boldsymbol{P}_\omega$ that diagonalizes ω will be parameterized as

$$\boldsymbol{P}_\omega = \begin{pmatrix} 0 & 1 \\ 1 & \upsilon \end{pmatrix}.$$

and is always invertible no matter the value of $\upsilon \neq 0$.

Now there are two degrees of freedom λ and υ, and the corresponding matrices are

$$\omega = \begin{pmatrix} \lambda & 0 \\ (\gamma - \lambda)\upsilon & \gamma \end{pmatrix}.$$

As before, the scale s is determined by c_γ, which depends linearly on υ. Hence, the choice of scale fixes υ, and λ has to be chosen either equal to 0 when s is different from $\boldsymbol{d}^\dagger \boldsymbol{e}$ (yielding a single solution) or it remains free if the choice of scale imposes $c_\lambda = 0$ (yielding a $1D$ manifold).

A perfectly analogous reasoning can be applied when $\upsilon_1 = 0$ and $u_1 \neq 0$, and in both cases, the manifolds of solution are of lower dimension than their counterparts, which have nonzero u_1 and u_2; we discard those solutions as we expect them to be smooth limits of the generic case.

Appendix F: Gradients and Hessian in the Low-Rank Parameterization

We now explicitly compute the derivative of the loss with respect to the 2×2 matrix ω. We previously found that

$$\begin{aligned} \mu_q &= \boldsymbol{d}^\dagger \boldsymbol{W}^q \boldsymbol{e} = \left(\sqrt{\boldsymbol{\Sigma}}\, \omega^q \sqrt{\boldsymbol{\Sigma}}\right)_{12} \\ &= \sum_{a,b=1}^{2} \sqrt{\boldsymbol{\Sigma}}_{1a} \omega^q_{ab} \sqrt{\boldsymbol{\Sigma}}_{b2}, \end{aligned}$$

and we have that

$$\frac{\partial \omega^q_{a,b}}{\partial \omega_{ij}} = \sum_{m=0}^{q-1} \omega^m_{ai} \omega^{q-1-m}_{jb},$$

so that

$$\frac{\partial \mu_q}{\partial \omega_{ij}} = \sum_{m=0}^{q-1} \left(\sqrt{\boldsymbol{\Sigma}} \omega^m\right)_{1i} \left(\omega^{q-1-m} \sqrt{\boldsymbol{\Sigma}}\right)_{j2},$$

which allows us to compute the gradient of the loss with respect to the coefficients of ω through

$$\frac{\partial \mathcal{L}}{\partial \omega_{ij}} = 2 \sum_{q,p=1}^{T} \chi_{q,p} (\mu_q - s\gamma^q) \frac{\partial \mu_p}{\partial \omega_{ij}}.$$

We can now compute the Hessian of the loss, which will allow us to derive formulas for stability and speed of convergence of gradient descent. This Hessian can be decomposed as

$$\mathcal{H}_{ij,kl} = \frac{\partial \mathcal{L}}{\partial \omega_{ij} \partial \omega_{kl}} = \sum_{q,p=1}^{T} \chi_{q,p} \left[\frac{\partial \mu_q}{\partial \omega_{ij}} \frac{\partial \mu_p}{\partial \omega_{kl}} + (\mu_q - s\gamma^q) \frac{\partial \mu_p}{\partial \omega_{ij} \partial \omega_{kl}} \right].$$

We want to estimate this Hessian at rank 1 generalizing integrators, so that the second part will always be zero. Therefore, the Hessian is simply

$$\mathcal{H}_{ij,kl} = \sum_{q,p=1}^{T} \chi_{q,p} \left[\sum_{m=0}^{q-1} (\sqrt{\mathbf{\Sigma}} \omega^m)_{1i} (\omega^{q-1-m} \sqrt{\mathbf{\Sigma}})_{j2} \right] \left[\sum_{\tilde{m}=0}^{p-1} (\sqrt{\mathbf{\Sigma}} \omega^{\tilde{m}})_{1k} (\omega^{p-1-\tilde{m}} \sqrt{\mathbf{\Sigma}})_{l2} \right]. \tag{F.1}$$

Appendix G: Minimum Convergence Time

We are now interested in studying the dynamics of convergence toward the GIs $\boldsymbol{W}^*$ in the null initialization subspace. To do so, we use a Taylor expansion of the loss around one of its minima:

$$\mathcal{L}(\boldsymbol{W}^* + \delta \boldsymbol{W}) = \sum_{i,j,k,l=1}^{n} \delta \boldsymbol{W}_{ij} \, \mathcal{H}_{ij,kl}(\boldsymbol{W}^*) \, \delta \boldsymbol{W}_{kl}.$$

Seeing $\delta \boldsymbol{W}$ as a vector and $\mathcal{H}$ as a (symmetric) matrix, we can diagonalize $\mathcal{H}$ with real eigenvalues λ_I and normalized eigenvectors $\boldsymbol{u}_I$, and express $\delta \boldsymbol{W} = \sum_{I=1}^{n^2} \delta_I \boldsymbol{u}_I$ in that basis so that

$$\mathcal{L}(\boldsymbol{W}^* + \delta \boldsymbol{W}) = \sum_{I=1}^{n^2} \lambda_I \, \delta_I^2. \tag{G.1}$$

Since our loss is positive for any weight-matrix $\boldsymbol{W}$, we expect that all eigenvalues of the Hessian computed at a GI be positive. We also expect that (in the generic case $s \neq \boldsymbol{d}^\dagger \boldsymbol{e}$) one of them is zero, corresponding to the local tangent to the one-dimensional manifold of GIs.

Writing the *GD* dynamics on the perturbation $\boldsymbol{\delta}$, we find that

$$\delta_I^{(\tau+1)} = (1 - \eta \lambda_I) \delta_I^{(\tau)};$$

hence, $\boldsymbol{\delta}$ will see each of its components either be conserved (if it corresponds to a null eigenvalue) or evolve exponentially. This exponential

evolution is convergent as long as $|1 - \eta\lambda_I| < 1$ and monotonic as long as $\eta < 1/\lambda_I$. Choosing the optimal learning rate for the full system $\eta^* = 1/\lambda_{\max}$, the slowest component of $\boldsymbol{\delta}$ evolves as

$$(1 - \eta^*\lambda_{\min})^\tau = (1 - \lambda_{\min}/\lambda_{\max})^\tau \simeq e^{\tau \ln(1-\mathcal{C}^{-1})} \simeq e^{-\tau/\mathcal{C}},$$

hence the characteristic convergence time will be $\mathcal{C} = \lambda_{\max}/\lambda_{\min}$.[7]

Since in the null initialization case, the weights are parameterized by a 2×2 matrix, the Hessian is 4×4 and its spectrum can easily be computed numerically by using equation F.1. We therefore performed the following study: fixing the L_2 norm of the encoder and decoder as well as their dot product,[8] we evaluate the spectrum of $\mathcal{H}$ and deduce from it the condition number $\mathcal{C}$ along each manifold of rank 1 s-scaled GIs; we find that a minimum exists for α, β (see appendix E) of order 1, and this value will be a lower bound for the convergence time to **any** GI at scale s using gradient descent. We plot the value of this bound as a function of s for different initial choices of i/o–vectors in Figure 14.

Appendix H: Algebraic Convergence for Specific Scale Value

From the previous analysis, it seems that when $s = \boldsymbol{d}^\dagger\boldsymbol{e}$, the $\lambda = 0$ manifold of solutions is hard to reach. Numerically, we see that gradient descent converges to a solution that lies in the union of the two $2D$ manifolds described earlier, corresponding to $\boldsymbol{W}$ of rank a priori 2. If we initialize with a random, nonzero, $\boldsymbol{W}$ in the null initialization subspace, we converge exponentially; if we start from $\boldsymbol{W} = 0$, the convergence is algebraic as a power law τ^{-2} instead of exponential (see Figure 15).

To understand this phenomenon, we consider the continuous-time, nonlinear differential equation on the coefficients of ω: $\dot{\omega} = -\partial_\omega\mathcal{L}$. We also introduce the two manifolds of GIs at scale $s = \boldsymbol{d}^\dagger\boldsymbol{e}$:

$$\mathcal{M}_{\alpha_0}(\alpha, \lambda) = \frac{1}{\alpha - \alpha_0}\begin{pmatrix} \alpha\lambda - \alpha_0\gamma & \gamma - \lambda \\ \alpha\alpha_0(\lambda - \gamma) & \alpha\gamma - \alpha_0\lambda \end{pmatrix},$$

$$\mathcal{M}_{\beta_0}(\beta, \lambda) = \frac{1}{\beta - \beta_0}\begin{pmatrix} \beta\gamma - \beta_0\lambda & \lambda - \gamma \\ \beta\beta_0(\gamma - \lambda) & \beta\lambda - \alpha_0\gamma \end{pmatrix}.$$

[7] $\lambda_{\min}$ is the minimum nonzero eigenvalue.

[8] In order to generate $\boldsymbol{e}$ and $\boldsymbol{d}$ with macroscopic overlaps (larger than $n^{-1/2}$), we first draw them independently, normalize them to 1, then modify the decoder as $\boldsymbol{d} = o\boldsymbol{e} + (1 - o)\boldsymbol{d}$ where o is the overlap; for large enough n, the dot product $\boldsymbol{d}^\dagger\boldsymbol{e}$ will be close to this overlap. We then independently rescale them to the desired norm.

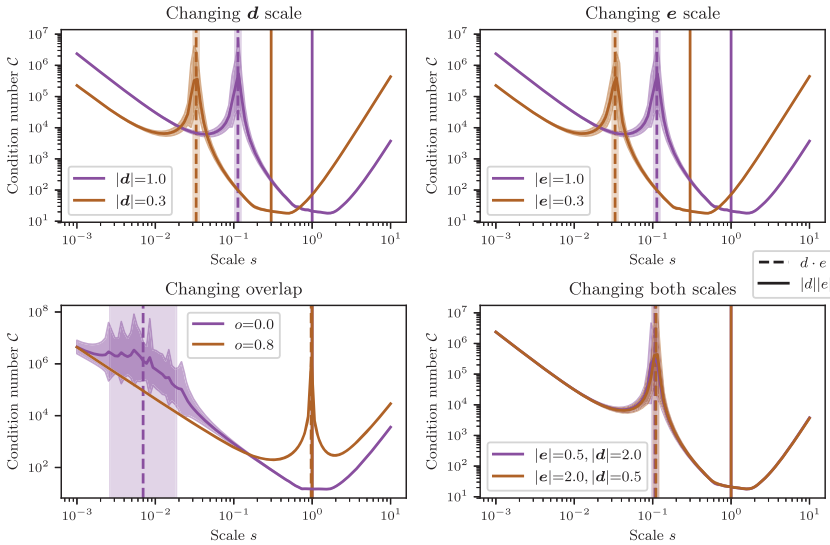

Figure 14: Evolution of the minimum convergence time as a function of the scale.

In the following, we refer respectively to the union and the intersection of those manifolds as $\mathcal{U}$ and $\mathcal{I}$. If we consider any GI $\boldsymbol{M}$ in $\mathcal{U} \setminus \mathcal{I}$, numerical experiments show that the Hessian has exactly two null and two strictly positive eigenvalues. The two null directions, which give us the linearized center space E_c around $\boldsymbol{M}$ in which convergence is at most as a power law, correspond to the local tangent of the manifold of minima and are therefore not relevant; convergence of the loss is exponential.

On the other hand, if $\boldsymbol{M} \in \mathcal{I}$, the Hessian exhibits three null eigenvalues (because the manifolds of minima intersect nontangentially), so that the center space E_c is now of dimension 3. Since the GIs are only two $2D$ planes, there exists an invariant manifold along which convergence is not exponential. Denoting as x the coordinate along that slow direction, the center manifold theorem ensures that the evolution of x is given by

$$\dot{x} = g(x),$$

where g is a polynomial of order at least 2 with neither constant nor first-order term.

Assuming that the order 2 term is nonzero, we get that locally, for x close to 0, $\dot{x} = ax^2$. Integrating over time, we get that x evolves as τ^{-1}. We then look at the value of the loss when ω is equal to a generalizing minimum $\boldsymbol{M}$

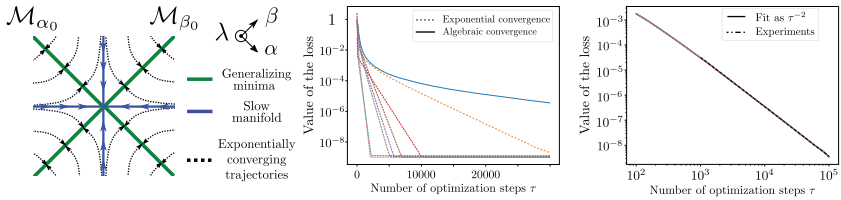

Figure 15: Explanation of the dynamics of convergence toward a minimum at $s = \boldsymbol{d}^\dagger \boldsymbol{e}$. (Left) Informal representation of GD trajectories. Two types of trajectories converging to GIs exist: starting from a random $\boldsymbol{W}$ in the null initialization subspace (middle), we usually converge outside the intersection of the two manifolds exponentially fast, with a fairly wide range of times of convergence depending on the precise starting point. In rare cases, that random starting point lies on (or close enough to) the slow manifold on which convergence is as a power law τ^{-2}. We illustrate this algebraic convergence by starting from $\boldsymbol{W} = 0$ (right), which is experimentally found to be on the slow manifold. Both experimental curves show eight different realizations of the training, with the same learning rate, norms of vectors, and overlap (but scale chosen exactly to $\boldsymbol{d}^\dagger \boldsymbol{e}$ after $\boldsymbol{e}$ and $\boldsymbol{d}$ have been drawn for that particular realization of the experiment).

plus a small perturbation $\boldsymbol{X}$ of order x:

$$\begin{aligned}
\mathcal{L} &= \sum_{q,p=1}^{T} \chi_{qp}(\mu_q - s\gamma^q)(\mu_p - s\gamma^p) \\
&= \sum_{q,p=1}^{T} \chi_{qp}[\sqrt{\boldsymbol{\Sigma}}\,(\boldsymbol{M}+\boldsymbol{X})^q\,\sqrt{\boldsymbol{\Sigma}})_{12} - s\gamma^q][\sqrt{\boldsymbol{\Sigma}}\,(\boldsymbol{M}+\boldsymbol{X})^p\,\sqrt{\boldsymbol{\Sigma}})_{12} - s\gamma^p] \\
&= \sum_{q,p=1}^{T} \chi_{qp}[(\sqrt{\boldsymbol{\Sigma}}\,\boldsymbol{M}^q\,\sqrt{\boldsymbol{\Sigma}})_{12} - s\gamma^q + \mathcal{O}(x)][(\sqrt{\boldsymbol{\Sigma}}\,\boldsymbol{M}^p\,\sqrt{\boldsymbol{\Sigma}})_{12} - s\gamma^p + \mathcal{O}(x)] \\
&= \mathcal{O}(x^2).
\end{aligned}$$

Therefore, if the quadratic term of g is nonzero, x scales as τ^{-1} and the loss as τ^{-2} when τ is large, as is observed experimentally. It should be noted that this result does not depend on the value of T or on the choice of $\boldsymbol{e}$ and $\boldsymbol{d}$, as observed experimentally too.

Therefore, algebraic convergence is observed only when very strict conditions are met:

- The gradient descent starts from a very specific subspace, the preimage of the intersection, which we will refer to as a slow manifold.

It is of lower dimension than the initial space of weight matrices, so that random initial conditions will almost never satisfy this criterion.
- The system always remains on the trajectory of GD. In particular, this means that η and the noise on the computed updates need to be small enough that we don't accidentally leave the slow manifold, which would then lead to exponential convergence.

Appendix I: Single-Channel ReLU Proxy Loss Gradients and Hessian

We showed in section 4.2.2 that the two following pairs of conditions are enough to guarantee perfect integration of arbitrary signals:

$$\begin{cases} \boldsymbol{d}^\dagger \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ = \pm s\gamma \\ \boldsymbol{W} \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ = \pm \gamma \boldsymbol{W}\boldsymbol{e}. \end{cases} \tag{I.1}$$

We define the **proxy** loss as the sum of four terms corresponding to the residuals of those equalities:

$$\mathcal{L}^{proxy} = \mathcal{L}^1_+ + \mathcal{L}^1_- + \mathcal{L}^2_+ + \mathcal{L}^2_-$$

where $\mathcal{L}^1_\pm = (\boldsymbol{d}^\dagger \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ - \pm s\gamma)^2$ and $\mathcal{L}^2_\pm = |\boldsymbol{W} \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ - \pm \gamma \boldsymbol{W}\boldsymbol{e}|^2$.

The gradients of these quantities are computed as

$$\begin{cases} \dfrac{\partial \mathcal{L}^1_\pm}{\partial \boldsymbol{W}_{ij}} = 2(\boldsymbol{d}^\dagger \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ - \pm s\gamma)(\pm \boldsymbol{d}_i \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e})_i \boldsymbol{e}_j) \\ \dfrac{\partial \mathcal{L}^2_\pm}{\partial \boldsymbol{W}_{ij}} = 2(\boldsymbol{W} \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ - \pm \gamma \boldsymbol{W}\boldsymbol{e})|_i (\lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ - \pm s\gamma)|_j \\ \qquad\qquad \pm 2\boldsymbol{e}_j \sum_a (\boldsymbol{W} \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ - \pm \gamma \boldsymbol{W}\boldsymbol{e})|_a \boldsymbol{W}_{ai} \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e})|_i \end{cases}$$

where $\boldsymbol{H}$ is the componentwise Heaviside function that takes a vector as input and returns a vector whose kth component is 0 if the kth component of the input was strictly negative, 1 if it was strictly positive, and .5 if it is exactly 0.

The Hessian of the $\mathcal{L}^1$ terms is readily computed as

$$\begin{aligned} \frac{\partial \mathcal{L}^1_\pm}{\partial \boldsymbol{W}_{ij} \partial \boldsymbol{W}_{kl}} &= 2\boldsymbol{d}_i \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e})|_i \boldsymbol{e}_j \boldsymbol{d}_k \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e})|_k \boldsymbol{e}_l) \\ &\quad + 2(\boldsymbol{d}^\dagger \lfloor \pm \boldsymbol{W}\boldsymbol{e} \rfloor_+ - \pm s\gamma)\delta_{ik} d_i e_j e_l \boldsymbol{\delta}(\pm \boldsymbol{W}\boldsymbol{e})|_i, \end{aligned}$$

where δ is the discrete Dirac distribution δ_{ik}, which is one if $i = k$ and 0 otherwise, and $\boldsymbol{\delta}$ the componentwise Dirac distribution such that $\boldsymbol{\delta}(\boldsymbol{v})$ is a vector of the same shape as $\boldsymbol{v}$ whose components are 1 if the corresponding composant of $\boldsymbol{v}$ is 0, and 0 otherwise. This part of the Hessian is indeed symmetric by exchange of ij and kl because of the δ_{ik} in the second term.

The Hessian of the $\mathcal{L}^2$ terms is more complicated but can be found to be

$$\begin{aligned}\frac{1}{2}\frac{\partial \mathcal{L}^2_{\pm}}{\partial \boldsymbol{W}_{ij}\partial \boldsymbol{W}_{kl}} = &\sum_a \boldsymbol{W}_{ai}\boldsymbol{W}_{ak}e_j e_l \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e})_i \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e})_k \\ &+ \sum_a (\boldsymbol{W}\lfloor \pm \boldsymbol{W}\boldsymbol{e}\rfloor_+ - \pm\gamma \boldsymbol{W}\boldsymbol{e})|_a \boldsymbol{W}_{ai}\delta_{ik}\boldsymbol{\delta}(\pm \boldsymbol{W}\boldsymbol{e})|_i e_j e_l \\ &+ \delta_{ik}(\lfloor \pm \boldsymbol{W}\boldsymbol{e}\rfloor_+ - \pm\gamma \boldsymbol{e})|_j\ (\lfloor \pm \boldsymbol{W}\boldsymbol{e}\rfloor_+ - \pm\gamma \boldsymbol{e})|_l \\ &+ \pm [\delta_{jk}(\boldsymbol{W}\lfloor \pm \boldsymbol{W}\boldsymbol{e}\rfloor_+ - \pm\gamma \boldsymbol{W}\boldsymbol{e})|_i e_l \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e}) + ij \Leftrightarrow kl] \\ &+ \pm [(\lfloor \pm \boldsymbol{W}\boldsymbol{e}\rfloor_+ - \pm\gamma \boldsymbol{e})|_j \boldsymbol{W}_{ik} e_l \boldsymbol{H}(\pm \boldsymbol{W}\boldsymbol{e})|_k + ij \Leftrightarrow kl].\end{aligned}$$

Combining those four terms, we get the Hessian of our full proxy loss:

$$\begin{aligned}\frac{1}{2}\frac{\partial \mathcal{L}^{proxy}_{\pm}}{\partial \boldsymbol{W}_{ij}\partial \boldsymbol{W}_{kl}} = &\, d_i e_j d_k e_l [\boldsymbol{H}(\boldsymbol{W}\boldsymbol{e})|_i \boldsymbol{H}(\boldsymbol{W}\boldsymbol{e})|_k + \boldsymbol{H}(-\boldsymbol{W}\boldsymbol{e})|_i \boldsymbol{H}(-\boldsymbol{W}\boldsymbol{e})|_k] \\ &+ e_j e_l \sum_a \boldsymbol{W}_{ai}\boldsymbol{W}_{ak}[\boldsymbol{H}(\boldsymbol{W}\boldsymbol{e})|_i \boldsymbol{H}(\boldsymbol{W}\boldsymbol{e})|_k + \boldsymbol{H}(-\boldsymbol{W}\boldsymbol{e})|_i \boldsymbol{H}(-\boldsymbol{W}\boldsymbol{e})|_k] \\ &+ \delta_{ik}[(\lfloor \boldsymbol{W}\boldsymbol{e}\rfloor_+ - \gamma \boldsymbol{e})|_j (\lfloor \boldsymbol{W}\boldsymbol{e}\rfloor_+ - \gamma \boldsymbol{e})|_l \\ &+ (\lfloor -\boldsymbol{W}\boldsymbol{e}\rfloor_+ + \gamma \boldsymbol{e})|_j (\lfloor -\boldsymbol{W}\boldsymbol{e}\rfloor_+ + \gamma \boldsymbol{e})|_j] \\ &+ [\delta_{jk}(\boldsymbol{W}^2\boldsymbol{e} - 2\gamma \boldsymbol{W}\boldsymbol{e})|_i e_l + ij \Leftrightarrow kl] \\ &+ [\boldsymbol{W}_{ik}(\boldsymbol{W}\boldsymbol{e} - 2\gamma \boldsymbol{e})|_j e_l + ij \Leftrightarrow kl] \\ &+ \delta_{ik} d_i e_j e_l \boldsymbol{d}^{\dagger} |\boldsymbol{W}\boldsymbol{e}| \boldsymbol{\delta}(\boldsymbol{W}\boldsymbol{e})|i \\ &+ \delta_{ik} e_j e_l \boldsymbol{d}^{\dagger} |\boldsymbol{W}\boldsymbol{e}| \sum_a \boldsymbol{W}_{ai}(\boldsymbol{W}|\boldsymbol{W}\boldsymbol{e}|)|_a \boldsymbol{\delta}(\boldsymbol{W}\boldsymbol{e})|i.\end{aligned}$$

Contrary to the linear null initialization case where the Hessian was a 4×4 matrix, we have no way to a priori reduce the number of degrees of freedom and $\mathcal{H}$ is a $n^2 \times n^2$–matrix. We are therefore restricted to a very low number of neurons (around 50 in our case) for the diagonalization to remain computationally tractable. Another major obstacle is that we do not have an analytical expression of the GI manifolds at which we want to evaluate the Hessian. We adopted the following methodology. First, we train networks on the proxy loss using gradient descent at a low learning rate and wait for convergence; we evaluate the largest eigenvalue λ_+ of $\mathcal{H}$ at the obtained

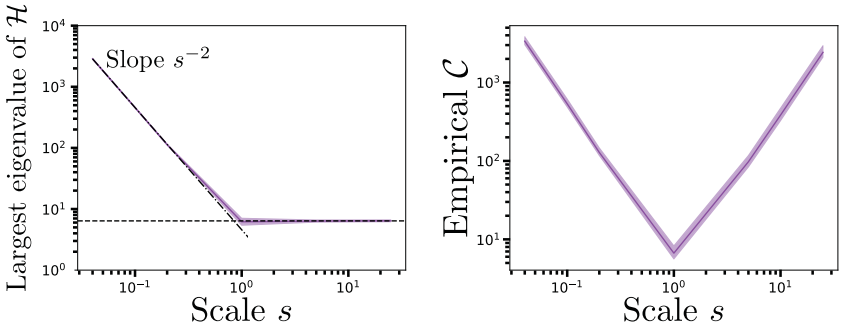

Figure 16: Experimentally determined values of the highest eigenvalue of the Hessian around the GI manifold, determining the optimal stable learning rate for GD and of the empirical convergence time as a function of the scale. Numerical experiments carried out with $n = 50$ neurons, independent encoder, and decoder of norm 1.

weight matrix, but do not compute the lowest ones as they are both prone to numerical errors and not necessarily positive, as some small negative eigenvalues will exist when we are only close to a GI. We compute an effective lowest eigenvalue by fitting the decay of the loss during GD at learning rate $\eta < 1/\lambda_+$ and deduce the corresponding minimum convergence time. The results of these numerical simulations are presented in Figure 16. We performed tests on larger networks to verify if the inferred maximum stable learning rate remained valid, as well as the order of magnitude of the convergence time, yielding the expected results.

Appendix J: Analysis of Rank-1 ReLU Generalizing Integrators

Training of the ReLU RNNs, on either real data or on the proxy loss (see equation 4.10), leads to GIs exhibiting one dominant singular value. As in the linear case, we write $\boldsymbol{W} = \sigma \boldsymbol{l}\boldsymbol{r}^\dagger$; with no loss of generality, we can impose $\boldsymbol{r}^\dagger \boldsymbol{e} > 0$ by multiplying $\boldsymbol{r}$ and $\boldsymbol{l}$ by -1. Conditions 4.9 become

$$\begin{cases} \sigma\, \boldsymbol{r}^\dagger \boldsymbol{l}_\pm & = \pm\gamma \\ \sigma\, (\boldsymbol{r}^\dagger \boldsymbol{e})(\boldsymbol{d}^\dagger \boldsymbol{l}_\pm) & = \pm s\gamma \end{cases} \quad \text{where } \boldsymbol{l}_\pm = \lfloor \pm \boldsymbol{l} \rfloor_+.$$

Using a Cauchy-Schwarz inequality and denoting as $\mathbf{1}_\pm$ the vector whose component i is equal to 1 if $\boldsymbol{l}_i$ is of the corresponding sign and 0 otherwise, we have

$$|\boldsymbol{d}^\dagger \boldsymbol{l}_\pm| = |(\boldsymbol{d}\mathbf{1}_\pm)^\dagger (\pm \boldsymbol{l}\mathbf{1}_\pm)| \leq |\boldsymbol{d}\mathbf{1}_\pm||\boldsymbol{l}\mathbf{1}_\pm|.$$

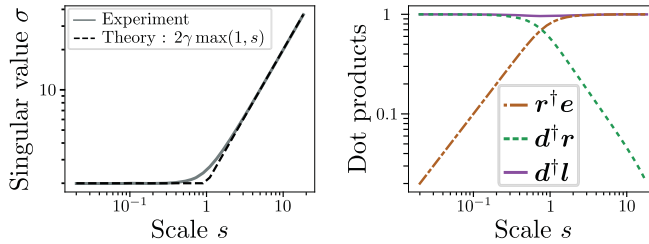

Figure 17: Behavior of the singular value σ and dot products of the singular vectors $\boldsymbol{l}, \boldsymbol{r}$ with $\boldsymbol{e}, \boldsymbol{d}$ as functions of s. The figure was obtained with $n = 1000$, independently drawn encoder and decoder, and aggregated across six realizations of GD on the proxy loss. Error bars are not reported as they are not distinguishable from the line width.

Assuming half of the components of $\boldsymbol{l}$ are positive and half are negative, as confirmed by numerical studies and given that $|\boldsymbol{d}| = |\boldsymbol{l}| = 1$, both norms on the right-hand side are equal to $2^{-1/2}$ in the large n limit, yielding $|\boldsymbol{d}^\dagger \boldsymbol{l}_\pm| \leq 1/2$. Since $0 \leq \boldsymbol{r}^\dagger \boldsymbol{e} \leq 1$, we conclude that $\sigma \geq 2s\gamma$. Similarly, we have that $|\boldsymbol{r}^\dagger \boldsymbol{l}_\pm| \leq 1/2$, implying $\sigma \geq 2\gamma$. These conditions can then be summarized into $\sigma \geq 2\gamma \max(s, 1)$.

Experimentally, we find that this lower bound is closely followed when s is either large or small. Since $\boldsymbol{W}$ is of rank 1, its Frobenius norm is equal to σ, and we argue that the saturation of this lower bound on σ is a manifestation of the conjecture of Arora et al. (2019) that gradient descent implicitly favors solutions with small matrix norm. Therefore, we have for any scale s significantly different from 1:

$$\sigma = 2\gamma \max(s, 1). \tag{J.1}$$

Numerical experiments show that for a wide range of scales, $\boldsymbol{l}$ and $\boldsymbol{d}$ are almost equal. Hence, $\boldsymbol{d}^\dagger \boldsymbol{l}_\pm = \pm |\boldsymbol{d}\lfloor \pm \boldsymbol{d}\rfloor_+|^2 \simeq \pm 1/2$, entailing $\boldsymbol{r}^\dagger \boldsymbol{e} \simeq \min(s, 1)$. For $s \ll 1$, $\boldsymbol{r}$ is almost aligned with $\boldsymbol{d}$, while for $s > 1$, we have $\boldsymbol{r} \simeq \boldsymbol{e}$ (this statement holds for uncorrelated encoder and decoder). Our theoretical predictions are in very good agreement with numerical experiments, as shown in Figure 17. Notice that the change of direction of $\boldsymbol{r}$ with s has consequences on the signs of the couplings: W_{ij} is positive for pairs of neurons within the $+$ and $-$ populations and negative in between at small s, but is essentially random at large s (see Figure 18).

Appendix K: Example of Transfer Learning: Context-Dependent Selectivity

In order to illustrate the versatility of the current-linear representations described in the main text, we implement a simple example of transfer

Figure 18: Weight matrix $\boldsymbol{W}$ of a single-channel ReLU network visualized as a discrete heat map. Neurons were reordered so that the indices of the + population are from 0 to $n/2$, while the − population goes from $n/2$ to n. When $s = 0.1$, the sign of W_{ij} is fully determined by whether i and j are in the same cluster; when $s = 10$, this observation is no longer true. We also note that as expected, the coefficients of $\boldsymbol{W}$ are larger when s increases as the norm of $\boldsymbol{W}$ scales as $\max(1, s)$. This figure was obtained for independent encoder and decoder of scale 1, $n = 1000$.

learning to context-dependent selectivity, inspired by Mante et al. (2013). The idea is the following: a pretrained three-channel integrator is used to integrate three time series $x_0,\ x_1,\ x_2$ (respectively, the motion evidence, color evidence, and contextual cue in the experiment described by Mante et al. (2013)) into their decaying integrals $y_0,\ y_1,\ y_2$, potentially with different decay constants. The cue integral y_2 is used to determine to which integral, y_0 or y_1, the network must be sensitive: when y_2 is negative, the network must output 0 if $y_0 < 0$ and 1 if $y_0 > 0$; when the integral y_2 is positive, the network must output 0 if $y_1 < 0$, and 1 if $y_1 > 0$.

To train this network, we first train the three-channel integrator using any of the methods described in the main text. We then use it as a fixed input transformer, mapping a three-dimensional time series to an n–dimensional one (the state $\boldsymbol{h}_t$ at any time step). For each time step, the value of the expected output is determined using the aforementioned rules on $\boldsymbol{y}$, and the classification output is obtained as $out_t = (1 + \exp^{-50(\boldsymbol{u}^\dagger \boldsymbol{h}_t - 0.1)})^{-1}$. The trainable parameter of this new decoding layer is the vector $\boldsymbol{u}$. It is easy to learn the value of $\boldsymbol{u}$ through batch SGD using the supervised learning procedure described here, and the resulting networks behave as shown in Figure 19.

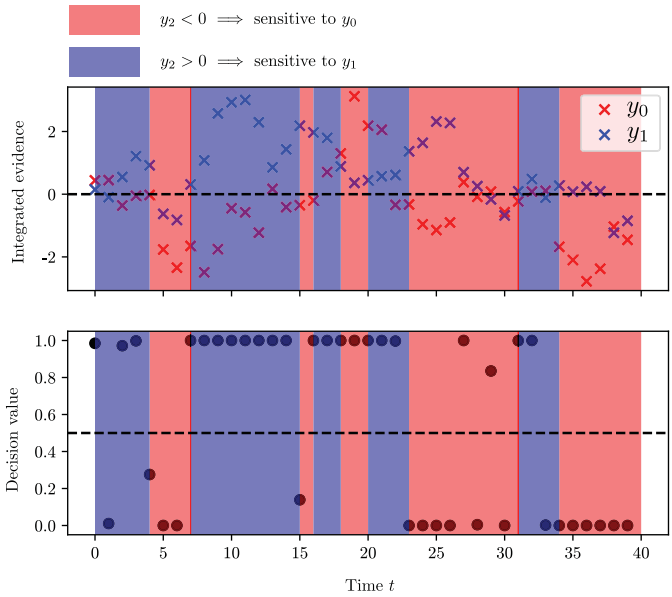

Figure 19: Output of an online context-dependent classifier. The task is the following: The network receives $D = 3$ input channels. When the integral y_2 is negative, the network must output 0 if $y_0 < 0$ and 1 if $y_0 > 0$; when the integral y_2 is positive, the network must output 0 if $y_1 < 0$, and 1 if $y_1 > 0$. This result is obtained by training a sigmoidal decoding layer on the internal states of a fixed sigmoidal network pretrained through batch SGD.

Acknowledgments

We are grateful to the referees for useful suggestions, in particular, the connection with (Mante, Sussillo, Shenoy, & Newsome, 2013); see appendix K, as well as the study of selectivity with sigmoidal networks, which shows preferred orientations see Figure 10C. We benefited from the support of NVIDIA Corporation with the donation of a Tesla K40 GPU card.

References

Aksay, E., Olasagasti, I., Mensh, B. D., Baker, R., Goldman, M. S., & Tank, D. W. (2007). Functional dissection of circuitry in a neural integrator. *Nature Neuroscience*, *10*(4), 494–504.

Almagro Armenteros, J. J., Sonderby, C. K., Sonderby, S. K., Nielsen, H., & Winther, O. (2017). DeepLoc: Prediction of protein subcellular localization using deep learning. *Bioinformatics*, *33*(21), 3387–3395.

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., . . . Zhu, Z. (2015). *Deep Speech 2: End-to-end speech recognition in English and Mandarin*. arXiv:1512.02595.

Arora, S., Cohen, N., Hu, W., & Luo, Y. (2019). *Implicit regularization in deep matrix factorization.* CoRR, abs/1905.13655.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., . . . Kumaran, D. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, *557*(7705), 429.

Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, *46*, 1–6.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical evaluation of gated recurrent neural networks on sequence modeling.* arXiv:1412.3555.

Collins, J., Sohl-Dickstein, J., & Sussillo, D. (2017). *Capacity and trainability in recurrent neural networks.* arXiv:1611.09913.

Denton, E. L., & Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. In I. Guyon, Y. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*, *30* (pp. 4414–4423). Red Hook, NY: Curran.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Haviv, D., Rivkind, A., & Barak, O. (2019). Understanding and controlling memory in recurrent neural networks. In *Proceedings of the International Conference on Machine Learning* (pp. 2663–2671).

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization*. arXiv:1412.6980.

Leavitt, M. L., & Morcos, A. (2020a). *Selectivity considered harmful: Evaluating the causal impact of class selectivity in DNNs*. arXiv:2003.01262.

Leavitt, M. L., & Morcos, A. S. (2020b). *On the relationship between class selectivity, dimensionality, and robustness.* arXiv:2007.04440.

Lee, D. D., Reis, B. Y., Seung, H. S., & Tank, D. W. (1997). Nonlinear network models of the oculomotor integrator. In J. M. Bower (Ed.), *Computational neuroscience* (pp. 371–377). Boston: Springer.

Lee, H.-Y., Tseng, H.-Y., Huang, J.-B., Singh, M., & Yang, M.-H. (2018). Diverse image-to-image translation via disentangled representations. In *Proceedings of the European Conference on Computer Vision* (pp. 35–51). Berlin: Springer.

Li, J., Monroe, W., & Jurafsky, D. (2017). *Understanding neural networks through representation erasure*. arXiv:1612.08220.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). *A critical review of recurrent neural networks for sequence learning.* arXiv:1506.00019.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., & Kaiser, L. (2016). *Multitask sequence to sequence learning.* arXiv:1511.06114.

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*(7474), 78–84.

Mastrogiuseppe, F., & Ostojic, S. (2018). Linking connectivity, dynamics and computations in low-rank recurrent neural networks. *Neuron*, *99*(3), 609–623.e29.

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, *73*, 1–15.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., & Mordvintsev, A. (2018). The building blocks of interpretability. *Distill*, *3*(3), e10.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*, *32* (pp. 8024–8035). Red Hook, NY: Curran.

Radford, A., Jozefowicz, R., & Sutskever, I. (2017). *Learning to generate reviews and discovering sentiment*. arXiv:1704.01444.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Chris tensen, A., . . . Kording, K. P. (2019). A deep learning framework for neuroscience. *Nature Neuroscience*, *22*(11), 1761–1770.

Rigotti, M., Barak, O., Warden, M. R., Wang, X.-J., Daw, N. D., Miller, E. K., & Fusi, S. (2013). The importance of mixed selectivity in complex cognitive asks. *Nature*, *497*(7451), 585–590.

Robinson, D. A. (1989). Integrating with neurons. *Annual Review of Neuroscience*, *12*(1), 33–45.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2014). *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*. arXiv:1312.6120.

Schuessler, F., Dubreuil, A., Mastrogiuseppe, F., Ostojic, S., & Barak, O. (2020). Dynamics of random recurrent networks with correlated low-rank structure. *Physical Review Research*, *2*(1):013111. arXiv:1909.04358.

Schuessler, F., Mastrogiuseppe, F., Dubreuil, A., Ostojic, S., & Barak, O. (2020). *The interplay between randomness and structure during learning in RNNs*. arXiv: 2006.11036.

Seung, H. S. (1996). How the brain keeps the eyes still. In *Proceedings of the National Academy of Sciences*, *93*(23), 13339–13344.

Song, H. F., Yang, G. R., & Wang, X.-J. (2016). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLOS Computational Biology*, *12*(2), 1–30.

Squire, L., Berg, D., Bloom, F., Lac, S., Ghosh, A., & Spitzer, N. (2012). *Fundamental neuroscience* (4th ed.). Amsterdam: Elsevier.

Sussillo, D., & Barak, O. (2012). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, *25*(3), 626–649.

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., . . . Hirose, A. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, *115*, 100–123.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., . . . van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for computing in Python. *Nature Methods*, *17*(3), 261–272.

Wong, K.-F., & Wang, X.-J. (2006). A recurrent network mechanism of time integration in perceptual decisions. *Journal of Neuroscience*, *26*(4), 1314–1328.
Zhang, Q.-s., & Zhu, S.-c. (2018). Visual interpretability for deep learning: A survey. *Frontiers of Information Technology and Electronic Engineering*, *19*(1), 27–39.

Received June 17, 2020; accepted November 4, 2020.