Analyzing and Improving Surrogate Gradient Training in Binary Neural Networks Using Dynamical Systems Theory

Rainer Engelken [1] **LF Abbott** [1]

Abstract

Training Binary Recurrent Networks on tasks that span long time horizons is challenging, as the discrete activation function renders the error landscape non-differentiable. Surrogate gradient training replaces the discrete activation function with a differentiable one in the backward pass but still suffers from exploding and vanishing gradients.

We leverage the connection between gradient stability and Lyapunov exponents to address this issue from a dynamical systems perspective, extending our previous work on Lyapunov exponent regularization to non-differentiable systems. We use differentiable linear algebra to regularize *surrogate Lyapunov exponents*, a method we call *surrogate gradient flossing*.

We show that *surrogate gradient flossing* enhances performance on temporally demanding tasks.

1. Introduction

Quantizing neural networks reduces computational cost and memory usage but complicates gradient-based training due to non-differentiable discrete activation functions. Traditional approaches, such as surrogate gradients [1], mitigate this issue by replacing the discrete nonlinearity with a differentiable expression in the backward pass. This technique is widely used for training spiking neural networks [2].

This paper extends our previous work on Lyapunov exponent regularization [3] to non-differentiable models. We introduce *surrogate Lyapunov exponents* to quantify surrogate gradient stability, provide analytical expressions for these exponents, and analyze the impact of the sharpness of the surrogate gradient.

[1]Center for Theoretical Neuroscience, Zuckerman Institute, Columbia University, New York, NY, USA. Correspondence to: Rainer Engelken <re2365@columbia.edu>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

2. Surrogate Lyapunov Exponents

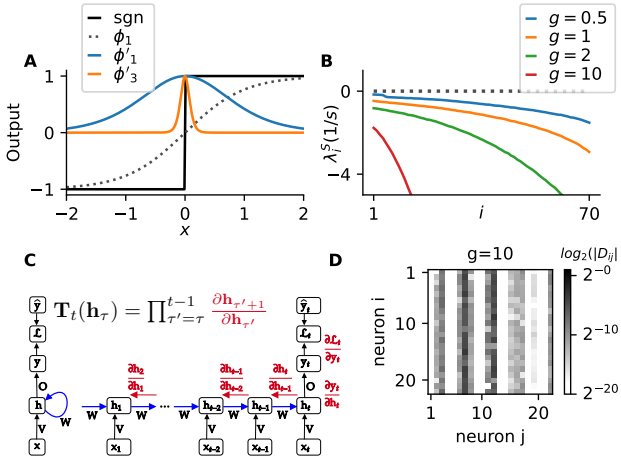

Figure 1. ***Surrogate Lyapunov exponents*** **quantify vanishing and exploding gradients of surrogate gradient training of binary neural networks A)** In binary networks $\mathbf{h}_{s+1} = \mathbf{W}\,\mathrm{sgn}(\mathbf{h}_s) + \mathbf{V}\mathbf{x}_{s+1}$, the step-like activation function leads to discontinuous, non-differentiable dynamics, creating a rugged error landscape in gradient-based learning. Surrogate gradient training replaces the step-like activation function with a continuous function ϕ in the backward pass, implemented here as a normalized derivative for $\phi(x) = \tanh(x)$ with $\phi'_g(x) = \mathrm{sech}(gx)^2$. The subscript denotes the sharpness g of the surrogate gradients. **B)** Surrogate Lyapunov exponents λ_i^S for different values of the surrogate sharpness g. **C)** Vanishing and exploding gradients in discrete recurrent networks arise from the attenuation or amplification of the product of surrogate Jacobians $\prod_{\tau'=\tau}^{t-1} \frac{\partial \mathbf{h}_{\tau'+1}}{\partial \mathbf{h}_{\tau'}}$ **D)** Surrogate Jacobian $\mathbf{D}^{\text{surrogate}} = \mathbf{W} \odot \phi'(\mathbf{h}_s)$ for $g = 10$, which results in a column-sparse Jacobian matrix. Note the logarithmic scale.

We introduce *surrogate Lyapunov exponents* as a tool for analyzing and mitigating vanishing and exploding gradients in discrete recurrent networks, described by

$$\mathbf{h}_{s+1} = \mathbf{W}\,\mathrm{sgn}(\mathbf{h}_s) + \mathbf{V}\mathbf{x}_{s+1}. \tag{1}$$

The core idea is to calculate *surrogate Lyapunov exponents*, which quantify exploding and vanishing gradients, based on the derivative of the surrogate activation function used in the backward pass. This differs from the conventional *Lyapunov exponents* that measure how small perturbations in the forward pass grow or shrink on average [4].

In surrogate gradient training, network parameters θ are gradually changed to reduce a loss $\mathcal{L}_t$ by smoothing the non-differentiable state transitions with a differentiable function for the backward pass (see Fig 1A). For the discretized dynamics of a binary or spiking recurrent neural network, the surrogate gradient of the loss $\mathcal{L}_t$ with respect to parameters θ is computed by unrolling the network dynamics backwards in time and propagating error signals across time steps (See Fig 1C):

$$\frac{\partial \mathcal{L}_t}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t} \sum_{\tau=0}^{\tau=t-1} \left(\prod_{\tau'=\tau}^{t-1} \mathbf{D}_{\tau'}^{\text{surrogate}} \right) \frac{\partial \mathbf{h}_\tau}{\partial \theta}, \qquad (2)$$

where $\mathbf{D}_s^{\text{surrogate}} = \frac{\partial \mathbf{h}_{s+1}}{\partial \mathbf{h}_s}$ is the surrogate Jacobian that describes the propagation of infinitesimal state changes across one time step using a differentiable approximation of the activation function sgn. See Fig 1D for an example surrogate Jacobian of a binary recurrent network with surrogate sharpness $g = 10$ and Appendix E for the derivation of the analytical surrogate Jacobian for a LIF network. The product of surrogate Jacobians $\prod_{\tau'=\tau}^{t-1} \mathbf{D}_{\tau'}^{\text{surrogate}}$ in Eq. 2 reflects the recursive dependence of network states on their past values and tends to give rise to vanishing and exploding gradients, making training across long time horizons challenging in both discrete and continuous recurrent neural networks.

Recent studies have shown that the singular values of the product of Jacobians, which measure the rate of gradient explosion, are mathematically closely related to Lyapunov exponents [4]–[11]. Lyapunov exponents are a core concept in dynamical systems theory, quantifying stability and chaos. The largest Lyapunov exponent λ_1 measures the average rate of divergence or convergence of nearby initial conditions. An N-dimensional system has N Lyapunov exponents that quantify the average growth rates of volume elements in the tangent space. To define *surrogate Lyapunov exponents*, we adapt this concept for surrogate gradients by instead considering the evolution of perturbations near the reference trajectory when evaluating the surrogate activation function (see Fig. 1C). We thus linearize along a reference trajectory of the forward dynamics of the discrete neural network but evaluate the surrogate Jacobians:

$$\lambda_i^{\text{surrogate}} = \lim_{t\to\infty} \frac{1}{t} \log \left[\sigma_i \left(\prod_{\tau'=0}^{t-1} \mathbf{D}_{\tau'}^{\text{surrogate}} \right) \right], \qquad (3)$$

where σ_i denotes the ith singular value of the surrogate Jacobian product. These exponents reflect the average asymptotic growth rates of infinitesimal perturbations in the surrogate tangent space of the forward dynamics, thus constraining signal propagation during BPTT over long time horizons. Specifically, modes with *surrogate Lyapunov exponents* near zero indicate directions along which error signals are, on average, neither strongly attenuated nor amplified, facilitating reliable information propagation across many time steps. We note that the *surrogate Lyapunov exponents* of discretized spiking network dynamics are different from the true Lyapunov exponents of the forward dynamics [12]–[18]. Figure 1B shows the *surrogate Lyapunov exponents* for different values of surrogate sharpness g in a binary RNN. We find that with increasing g, most *surrogate Lyapunov exponents* become very negative, leading to vanishing gradients and an ill-conditioned product of surrogate Jacobians during training (See Appendix G and F for details). Conversely, for small values of g, we find, depending on $\mathbf{W}$, and $\mathbf{V}$, many *surrogate Lyapunov exponents* close to zero, which more easily give rise to oscillations or exploding gradients during task training once the spectral radius of the recurrent weights grows slightly. See supplementary Figures 7 and 8 in Appendix L for an analysis of the convergence of surrogate Lyapunov exponents with simulation time for different values of g.

3. Surrogate Gradient Flossing

Our recent work built upon the mathematical relationship between *Lyapunov exponents* and the challenge of exploding and vanishing gradients in training of recurrent networks, and introduced a novel regularization technique termed *gradient flossing* [3]. This method stabilizes the training process by constraining *Lyapunov exponents* to values close to zero, ensuring more trainable network dynamics through a better-conditioned product of Jacobians. Here, we apply this idea for binary neural networks trained with surrogate gradients. In our approach, we regularize the *surrogate Lyapunov exponents*, extending the concept of *gradient flossing* from conventional *Lyapunov exponents* to the surrogate domain [3]. To mitigate exploding and vanishing gradients, we regularize the mean of the squares of the first k largest *surrogate Lyapunov exponents* $\lambda_1, \lambda_2, \ldots, \lambda_k$, resulting in an additional loss term $\mathcal{L}_{\text{surrogate flossing}} = \frac{1}{k}\sum_{i=1}^{k} \lambda_i^2$. To calculate the *surrogate Lyapunov exponents*, we employ an established iterative orthonormalization method using QR decomposition [4], [8], [19], rather than directly evaluating the product of surrogate Jacobians (Eq.2). This method is numerically robust, avoiding issues with the ill-conditioned long-term surrogate Jacobian (See AppendixA for details and pseudocode). *Surrogate Lyapunov exponents* are obtained by time-averaging the logarithms of the diagonal entries of the $\mathbf{R}^s$ matrices obtained during QR decompositions: $\lambda_i = \lim_{t\to\infty} \frac{1}{t}\sum_{s=1}^{t} \log \mathbf{R}_{ii}^s$. To backpropagate the gradient of the squared *surrogate Lyapunov exponents*, we leverage recent progress in differentiable linear algebra, which provided an analytical expression for the pullback of the QR decomposition (See Appendix B). For clarity, we described *surrogate gradient flossing* in terms of stochastic gradient descent, but we actually implemented it with the

ADAM optimizer [20] using standard hyperparameters η, β_1, and β_2. An example implementation in Julia [21] using Flux [22] is available here. This algorithm is versatile and can be applied to various neuron models and network architectures. The surrogate Jacobian matrix $\mathbf{D}^{\text{surrogate}}$ can be calculated analytically, as done here, or through automatic differentiation.

4. Control of Surrogate Lyapunov Exponents

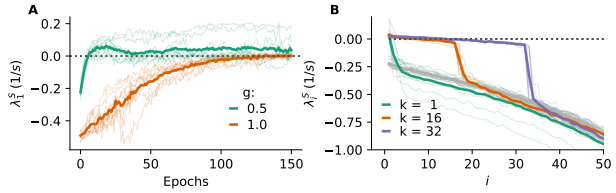

Figure 2. ***Surrogate gradient flossing*** **regularizes** ***surrogate Lyapunov exponents*** **and facilitates gradient signal propagation in binary neural networks A)** The first *surrogate Lyapunov exponent* of a recurrent binary network plotted as a function of training epochs for different surrogate sharpness g. The square of the first *surrogate Lyapunov exponent* is minimized using gradient descent. **B)** *Surrogate Lyapunov spectrum* of a recurrent binary network after different numbers of Lyapunov exponents k have been driven towards zero via *surrogate gradient flossing* for $k \in \{1, 16, 32\}$. The gray lines show the *surrogate Lyapunov spectra* before *surrogate gradient flossing*. Parameters: network size $N = 80$, $g = 1$ for **B**. Input as in Fig. 3. The thin semitransparent lines in **A** and **B** indicate nine network realizations; the full lines are their average.

Figure 2 demonstrates that *surrogate gradient flossing* can set one or several *surrogate Lyapunov exponents* to a target value via gradient descent with the ADAM optimizer in networks of binary neurons for different values of surrogate gradient sharpness g. The initial entries of the recurrent weight matrix $\mathbf{W}$ are independently drawn from a Gaussian distribution with zero mean and variance $1/N$. The recurrent binary network receives a random binary input stream, and the input weights $\mathbf{V}$ are drawn from $\mathcal{N}(0, 1/N_{\text{in}})$. $\mathbf{W}$, g, and $\mathbf{V}$ are all modified by *surrogate gradient flossing*.

Figure 2A shows that for randomly initialized binary RNNs, the *surrogate Lyapunov exponent* can be modified by *surrogate gradient flossing* to approach a desired target value. During *surrogate gradient flossing*, the exponents quickly approached the target value of zero for various surrogate gradient sharpness values g. Notably, networks with large g converge more slowly towards the target value of zero or may not converge at all.

Figure 2B shows *surrogate gradient flossing* for different numbers of flossed surrogate Lyapunov exponents k. During gradient descent, the mean of the squares of 1, 16, or 32 *surrogate Lyapunov exponents* is used as the loss. The resulting *surrogate Lyapunov spectrum* after flossing shows the corresponding number of exponents driven close to zero.

We conclude that *surrogate gradient flossing* can selectively manipulate one, several, or all *surrogate Lyapunov exponents* before or during network training, thereby shaping both the norm and the condition number of the surrogate gradients over long time horizons.

5. *Surrogate Gradient Flossing* Enhances Trainability

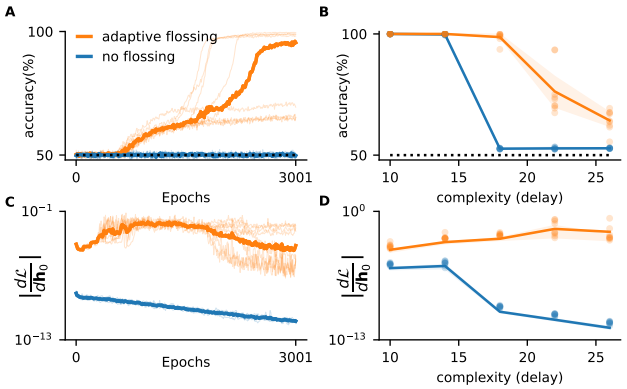

Figure 3. ***Gradient flossing*** **improves binary RNN training A)** Test accuracy for binary RNNs trained on the delayed temporal binary XOR task $y_t = x_{t-d/2} \oplus x_{t-d}$ with *adaptive gradient flossing* during training (orange) and without *gradient flossing* (blue) for $d = 18$. Solid lines are the median across 9 network realizations, and individual network realizations are shown in transparent fine lines. **B)** Mean final test accuracy as a function of task difficulty (delay d) for delayed XOR task. **C)** Gradient norm with respect to initial network state $\mathbf{h}_0$. **D)** Gradient norm with respect to initial network state as a function of temporal task complexity T averaged over training epochs.

We present numerical results on a task with variable temporal complexity, demonstrating that *adaptive surrogate gradient flossing* enhances the trainability of binary RNNs. In *adaptive surrogate gradient flossing*, we initialize the input weights $\mathbf{V}$, recurrent weights $\mathbf{W}$, and output weights randomly, set $g = 1$, and first employ *surrogate gradient flossing* for 500 epochs by minimizing $\mathcal{L}\text{flossing} = \frac{1}{k}\sum i = 1^k \lambda_i^2$. We then alternated between *surrogate gradient flossing* and task training every 100 epochs until task training epoch 500, and then continued with pure task training. Afterwards, we performed one episode of gradient flossing adaptively whenever either $\lambda_1^S > 0.2$ or the gradient norm $|\frac{\partial \mathcal{L}_T}{\partial \mathbf{W}}|$ exceeded 1.

We consider a temporal XOR task that requires the binary RNN to perform a nonlinear input-output computation on a sequential stream of random binary input with a delay of d steps. Specifically, the target output y is $y_t = x_{t-d/2} \oplus x_{t-d}$, where x is a random Bernoulli process $x \in \{0, 1\}$. This task requires retaining d items in memory and performing a nonlinear operation on them. While solving the task is trivial and the correct solution could be implemented manually, RNNs trained on this task quickly encounter exploding or vanishing gradients. We train the network by minimizing

the cross-entropy loss between the one-hot encoded target output and a linear readout. To successfully solve this task, the recurrent binary network must bridge the time horizon of d steps. We change the temporal complexity of the task by scaling the delay d. Figures 3A and B show that *adaptive surrogate gradient flossing* helps train networks for substantially longer delays d, while networks without flossing fail on this temporally challenging task. Figures 3C and D show that binary networks suffer from vanishing gradients, as revealed by measuring the gradient with respect to the initial network state $\mathbf{h}_0$. This issue is mitigated by *adaptive surrogate gradient flossing*.

6. Limitations

When adapting the input weights, recurrent weights, and g during *surrogate gradient flossing*, we observed that g became very small, causing the surrogate function ϕ to become close to linear. In that situation, the binary RNNs were still able to solve linear tasks, such as a delayed copy task, but not XOR. Additionally, binary networks with very small g were sometimes susceptible to oscillations or diverging gradients. We addressed this by using adaptive gradient flossing and successively increasing g whenever the maximum *surrogate Lyapunov exponent* estimate $\lambda_1^S > 0.2$ or the gradient norm $|\frac{\partial \mathcal{L}_T}{\partial \mathbf{W}}|$ exceeded 1. In general, binary RNNs with surrogate gradients seem to have a trade-off between the ability to bridge long time horizons and the ability to solve nonlinear tasks such as XOR.

We focused our analysis on recurrent binary and spiking neural networks because their recurrent interactions make them an interesting and powerful dynamical system. However, the insights from our study could also help address vanishing and exploding gradients in deeply layered feedforward quantized neural networks, where these issues can be linked to transient chaos or transient stability [23].

A limitation of the proposed *surrogate gradient flossing* implementation is that QR decomposition scales with $O(Nk^2)$, where N is the network size and k is the number of *surrogate Lyapunov exponents* being flossed. The number of *surrogate Lyapunov exponents* that need to be pushed towards zero depends on the task. For tasks requiring more than $O(\sqrt{N})$ *surrogate Lyapunov exponents* close to zero, the QR decomposition would become the computational bottleneck. We previously suggested strategies to address this computational bottleneck [3].

7. Discussion

Training discrete networks is challenging because of the non-differentiable error landscapes created by discrete interactions. Surrogate gradient training is commonly used for binary and spiking neural networks, but it often leads to exploding or vanishing gradients, especially in temporally complex tasks. We adopted a dynamical systems perspective to analyze and improve the surrogate gradient training of binary and spiking neural networks.

We introduced *surrogate Lyapunov exponents*, a novel set of indicators based on dynamical systems, to quantify the surrogate gradient flow in the backward pass, which is crucial for addressing exploding and vanishing gradients. *Surrogate Lyapunov exponents* account for the entire trajectory, revealing complex neuronal interactions previously neglected in the analysis of surrogate gradients (see Appendix H for more literature review). We called them *surrogate Lyapunov exponents* because, unlike conventional Lyapunov exponents that stem from a linearization of the forward dynamics, these arise from the surrogate dynamics of the backward pass.

Using differentiable linear algebra, we implement *surrogate gradient flossing* to push *surrogate Lyapunov exponents* towards zero. This method slows collective timescales in the tangent space, facilitating error propagation during training. Unlike previous related work that introduced *gradient flossing* to shape the Lyapunov exponents of the forward dynamics and thus improve the trainability of differentiable dynamical systems [3], we specifically address the surrogate gradients in the backward pass of models that have non-differentiable error landscapes. We demonstrate that *surrogate gradient flossing* prevents the gradient norm from exploding or vanishing during training. Empirically, *surrogate gradient flossing* enhances training on tasks that require bridging long time horizons.

Prior research that addressed vanishing and exploding gradients for surrogate training focused mostly on optimizing the surrogate functions of individual neurons [24], [25], or optimized naturalistic network initializations [26]. In contrast, we studied the collective network state and the tangent dynamics in the backward pass that involves many neurons and many time steps. We find that the sharpness of the surrogate gradient g, which shapes how strongly the activation function is smoothed during the backward pass, plays a crucial role in determining the norm and condition number of the surrogate gradient. This is because g shapes the sparsity of the surrogate Jacobian and thus the shape of the *surrogate Lyapunov spectrum*. Compared to a previous method to control Lyapunov exponents of vanilla RNNs with continuous activation function [3], here we addressed discrete networks and optimized the surrogate gradients relevant to the backward pass of gradient training.

Future studies should evaluate *surrogate gradient flossing* in real-world applications and extend testing to complex architectures, including deep spiking networks, spiking transformers, and quantized neural networks.

References

[1] Y. Bengio, N. Léonard, and A. Courville, *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*, arXiv:1308.3432 [cs], Aug. 2013. DOI: 10.48550/arXiv.1308.3432. [Online]. Available: http://arxiv.org/abs/1308.3432 (visited on 04/13/2024).

[2] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[3] R. Engelken, "Gradient Flossing: Improving Gradient Descent through Dynamic Control of Jacobians," en, May 2023. [Online]. Available: https://openreview.net/forum?id=jEQRoJzDx8&referrer=%5BAuthor%20Console%5D(%2Fgroup%3Fid%3DNeurIPS.cc%2F2023%2FConference%2FAuthors%23your-submissions) (visited on 08/28/2023).

[4] R. Engelken, F. Wolf, and L. F. Abbott, "Lyapunov spectra of chaotic recurrent neural networks," *Physical Review Research*, vol. 5, no. 4, p. 043 044, Oct. 2023. DOI: 10.1103/PhysRevResearch.5.043044. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.5.043044 (visited on 10/19/2023).

[5] R. Engelken, F. Wolf, and L. F. Abbott, "Lyapunov spectra of chaotic recurrent neural networks," *arXiv:2006.02427 [nlin, q-bio]*, Jun. 2020, arXiv: 2006.02427. [Online]. Available: http://arxiv.org/abs/2006.02427 (visited on 06/14/2020).

[6] R. Vogt, M. Puelma Touzel, E. Shlizerman, and G. Lajoie, "On Lyapunov Exponents for RNNs: Understanding Information Propagation Using Dynamical Systems Tools," *Frontiers in Applied Mathematics and Statistics*, vol. 8, 2022, ISSN: 2297-4687. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fams.2022.818799 (visited on 04/08/2023).

[7] J. Mikhaeil, Z. Monfared, and D. Durstewitz, "On the difficulty of learning chaotic dynamics with RNNs," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 11 297–11 312, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/495e55f361708bedbab5d81f92048dcd-Abstract-Conference.html (visited on 04/08/2023).

[8] R. Engelken, "Gradient Flossing: Improving Gradient Descent through Dynamic Control of Jacobians," en, *Advances in Neural Information Processing Systems*, vol. 36, pp. 10 412–10 439, Dec. 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/214ce905bf2072535e34b3cf873cbbc8-Abstract-Conference.html (visited on 03/13/2024).

[9] P. A. Sokół, I. Jordan, E. Kadile, and I. M. Park, "Adjoint Dynamics of Stable Limit Cycle Neural Networks," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, ISSN: 2576-2303, Nov. 2019, pp. 884–887. DOI: 10.1109/IEEECONF44664.2019.9049080.

[10] P. A. Sokół, "Geometry of Learning and Representations in Neural Networks," eng, Ph.D. dissertation, Stony Brook University, May 2023.

[11] I. M. Park, Á. Ságodi, and P. A. Sokół, "Persistent learning signals and working memory without continuous attractors," eng, *ArXiv*, arXiv:2308.12585v1, Aug. 2023, ISSN: 2331-8422.

[12] M. Monteforte and F. Wolf, "Dynamical Entropy Production in Spiking Neuron Networks in the Balanced State," *Physical Review Letters*, vol. 105, no. 26, p. 268 104, Dec. 2010. DOI: 10.1103/PhysRevLett.105.268104. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.105.268104 (visited on 06/02/2012).

[13] M. Monteforte and F. Wolf, "Dynamic Flux Tubes Form Reservoirs of Stability in Neuronal Circuits," *Physical Review X*, vol. 2, no. 4, p. 041 007, Nov. 2012. DOI: 10.1103/PhysRevX.2.041007. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevX.2.041007 (visited on 11/17/2012).

[14] A. Palmigiano, R. Engelken, and F. Wolf, "Boosting of neural circuit chaos at the onset of collective oscillations," en, *eLife*, vol. 12, Nov. 2023. DOI: 10.7554/eLife.90378. [Online]. Available: https://elifesciences.org/reviewed-preprints/90378 (visited on 12/28/2023).

[15] R. Engelken, "Chaotic Neural Circuit Dynamics," deu, Feb. 2018. [Online]. Available: https://ediss.uni-goettingen.de/handle/11858/00-1735-0000-002E-E349-9 (visited on 03/27/2018).

[16] P. Manz, S. Goedeke, and R.-M. Memmesheimer, "Dynamics and computation in mixed networks containing neurons that accelerate towards spiking," eng, *Physical Review. E*, vol. 100, no. 4-1, p. 042 404, Oct. 2019, ISSN: 2470-0053. DOI: 10.1103/PhysRevE.100.042404.

[17] R. Engelken, "SparseProp: Efficient Event-Based Simulation and Training of Sparse Recurrent Spiking Neural Networks," en, May 2023. [Online]. Available: https://openreview.net/forum?id=yzZbwQPkmP&referrer=%5BAuthor%20Console%5D(%2Fgroup%3Fid%3DNeurIPS.cc%2F2023%2FConference%2FAuthors%23your-submissions) (visited on 08/28/2023).

[18] M. Puelma Touzel and F. Wolf, "Statistical mechanics of spike events underlying phase space partitioning and sequence codes in large-scale models of neural circuits," *Physical Review E*, vol. 99, no. 5, p. 052 402, May 2019. DOI: 10.1103/PhysRevE.99.052402. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.99.052402 (visited on 02/15/2022).

[19] G. Benettin, L. Galgani, A. Giorgilli, and J.-M. Strelcyn, "Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; A method for computing all of them. Part 2: Numerical application," en, *Meccanica*, vol. 15, no. 1, pp. 21–30, Mar. 1980, ISSN: 0025-6455, 1572-9648. DOI: 10.1007/BF02128237. [Online]. Available: http://link.springer.com/article/10.1007/BF02128237 (visited on 04/14/2014).

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, vol. 1412, arXiv:1412.6980, Dec. 2014. [Online]. Available: http://adsabs.harvard.edu/abs/2014arXiv1412.6980K (visited on 05/10/2019).

[21] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah, "Julia: A Fresh Approach to Numerical Computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, Jan. 2017, ISSN: 0036-1445. DOI: 10.1137/141000671. [Online]. Available: http://epubs.siam.org/doi/10.1137/141000671 (visited on 02/07/2018).

[22] Innes, "Flux: Elegant machine learning with Julia.," *Journal of Open Source Software*, vol. 3, no. 25, p. 602, May 2018. [Online]. Available: https://doi.org/10.21105/joss.00602.

[23] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," *arXiv:1606.05340 [cond-mat, stat]*, Jun. 2016, arXiv: 1606.05340. [Online]. Available: http://arxiv.org/abs/1606.05340 (visited on 10/20/2016).

[24] Y. Li, Y. Guo, S. Zhang, S. Deng, Y. Hai, and S. Gu, "Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 23 426–23 439. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/c4ca4238a0b923820dcc509a6f75849b-Abstract.html (visited on 04/09/2024).

[25] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural computation*, vol. 33, no. 4, pp. 899–925, 2021, Publisher: MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ... [Online]. Available: https://direct.mit.edu/neco/article-abstract/33/4/899/97482 (visited on 04/09/2024).

[26] J. Rossbroich, J. Gygax, and F. Zenke, "Fluctuation-driven initialization for spiking neural network training," *Neuromorphic Computing and Engineering*, vol. 2, no. 4, p. 044 016, 2022, Publisher: IOP Publishing. [Online]. Available: https://iopscience.iop.org/article/10.1088/2634-4386/ac97bb/meta (visited on 04/09/2024).

[27] S. F. Walter and L. Lehmann, "Algorithmic Differentiation of Linear Algebra Functions with Application in Optimum Experimental Design (Extended Version)," Tech. Rep., Jan. 2010, ADS Bibcode: 2010arXiv1001.1654W Type: article. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2010arXiv1001.1654W (visited on 04/08/2023).

[28] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, "Differentiable Programming Tensor Networks," *Physical Review X*, vol. 9, no. 3, p. 031 041, Sep. 2019, arXiv:1903.09650 [cond-mat, physics:quant-ph], ISSN: 2160-3308. DOI: 10.1103/PhysRevX.9.031041. [Online]. Available: http://arxiv.org/abs/1903.09650 (visited on 03/23/2023).

[29] R. Schreiber and C. Van Loan, "A Storage-Efficient WY Representation for Products of Householder Transformations," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 1, pp. 53–57, Jan. 1989, ISSN: 0196-5204. DOI: 10.1137/0910005. [Online]. Available: https://epubs.siam.org/doi/10.1137/0910005 (visited on 03/23/2023).

[30] M. Seeger, A. Hetzel, Z. Dai, E. Meissner, and N. D. Lawrence, "Auto-Differentiating Linear Algebra," Tech. Rep., Oct. 2017, ADS Bibcode: 2017arXiv171008717S Type: article. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2017arXiv171008717S (visited on 04/08/2023).

[31] F. Farkhooi and W. Stannat, "Complete Mean-Field Theory for Dynamics of Binary Recurrent Networks," *Physical Review Letters*, vol. 119, no. 20, p. 208 301, Nov. 2017. DOI: 10.1103/PhysRevLett.119.208301. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.208301 (visited on 01/24/2018).

[32] H. Sompolinsky, A. Crisanti, and H. J. Sommers, "Chaos in Random Neural Networks," *Physical Review Letters*, vol. 61, no. 3, pp. 259–262, Jul. 1988. DOI: 10.1103/PhysRevLett.61.259. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.61.259 (visited on 01/08/2015).

[33] J. Kadmon and H. Sompolinsky, "Transition to Chaos in Random Neuronal Networks," *Physical Review X*, vol. 5, no. 4, p. 041 030, Nov. 2015. DOI: 10.1103/PhysRevX.5.041030. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevX.5.041030 (visited on 11/08/2016).

[34] L. Molgedey, J. Schuchhardt, and H. G. Schuster, "Suppressing chaos in neural networks by noise," *Physical Review Letters*, vol. 69, no. 26, pp. 3717–3719, Dec. 1992. DOI: 10.1103/PhysRevLett.69.3717. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.69.3717 (visited on 12/23/2014).

[35] J. Schuecker, S. Goedeke, and M. Helias, "Optimal Sequence Memory in Driven Random Networks," *Physical Review X*, vol. 8, no. 4, p. 041 029, Nov. 2018. DOI: 10.1103/PhysRevX.8.041029. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.041029 (visited on 11/20/2018).

[36] A. Crisanti and H. Sompolinsky, "Path integral approach to random neural networks," *Physical Review E*, vol. 98, no. 6, p. 062 120, Dec. 2018. DOI: 10.1103/PhysRevE.98.062120. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.98.062120 (visited on 07/31/2019).

[37] K. Rajan, L. F. Abbott, and H. Sompolinsky, "Stimulus-dependent suppression of chaos in recurrent neural networks," *Physical Review E*, vol. 82, no. 1, p. 011 903, Jul. 2010. DOI: 10.1103/PhysRevE.82.011903. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.82.011903 (visited on 11/09/2012).

[38] R. Engelken, A. Ingrosso, R. Khajeh, S. Goedeke, and L. F. Abbott, "Input correlations impede suppression of chaos and learning in balanced firing-rate networks," en, *PLOS Computational Biology*, vol. 18, no. 12, e1010590, Dec. 2022, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1010590. [Online]. Available: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010590 (visited on 12/23/2022).

[39] C. M. Newman, "The distribution of Lyapanov exponents: Exact results for random matrices," eng, *Communications in mathematical physics*, vol. 103, no. 1, pp. 121–126, 1986, ISSN: 0010-3616. [Online]. Available: http://cat.inist.fr/?aModele=afficheN&cpsidt=7875417 (visited on 10/22/2016).

[40] M. Isopi and C. M. Newman, "The triangle law for Lyapunov exponents of large random matrices," eng, *Communications in mathematical physics*, vol. 143, no. 3, pp. 591–598, 1992, ISSN: 0010-3616. [Online]. Available: http://cat.inist.fr/?aModele=afficheN&cpsidt=5111053 (visited on 10/22/2016).

[41] M. Bauer and W. Martienssen, "Lyapunov exponents and dimensions of chaotic neural networks," en, *Journal of Physics A: Mathematical and General*, vol. 24, no. 19, p. 4557, 1991, ISSN: 0305-4470. DOI: 10.1088/0305-4470/24/19/019. [Online]. Available: http://stacks.iop.org/0305-4470/24/i=19/a=019 (visited on 12/17/2016).

[42] G. Curato and A. Politi, "Onset of chaotic dynamics in neural networks," *Physical Review E*, vol. 88, no. 4, p. 042 908, Oct. 2013. DOI: 10.1103/PhysRevE.88.042908. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.88.042908 (visited on 11/28/2013).

[43] A. Lerchner, C. Ursta, J. Hertz, M. Ahmadi, P. Ruffiot, and S. Enemark, "Response Variability in Balanced Cortical Networks," *Neural Computation*, vol. 18, no. 3, pp. 634–659, Mar. 2006, ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.3.634. [Online]. Available: https://doi.org/10.1162/neco.2006.18.3.634 (visited on 06/20/2023).

[44] R. Engelken, F. Farkhooi, D. Hansel, C. van Vreeswijk, and F. Wolf, "A reanalysis of "Two types of asynchronous activity in networks of excitatory and inhibitory spiking neurons"," en, *F1000Research*, vol. 5, p. 2043, Aug. 2016, ISSN: 2046-1402. DOI: 10.12688/f1000research.9144.1. [Online]. Available: http://f1000research.com/articles/5-2043/v1 (visited on 09/28/2016).

[45] C. van Vreeswijk and H. Sompolinsky, "Chaos in Neuronal Networks with Balanced Excitatory and Inhibitory Activity," *Science*, vol. 274, no. 5293, pp. 1724–1726, Dec. 1996. DOI: 10.1126/science.274.5293.1724. [Online]. Available: http://www.sciencemag.org/content/274/5293/1724.abstract (visited on 12/11/2011).

[46] C. van Vreeswijk and H. Sompolinsky, "Chaotic Balanced State in a Model of Cortical Circuits," *Neural Computation*, vol. 10, no. 6, pp. 1321–1371, 1998, ISSN: 0899-7667. DOI: 10.1162/089976698300017214. [Online]. Available: http://dx.doi.org/10.1162/089976698300017214.

[47] F. Zenke and S. Ganguli, "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks," *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, Jun. 2018, ISSN: 0899-7667. DOI: 10.1162/neco_a_01086. [Online]. Available: https://doi.org/10.1162/neco_a_01086 (visited on 05/13/2023).

[48] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv:1312.6120 [cond-mat, q-bio, stat]*, Dec. 2013, arXiv: 1312.6120. [Online]. Available: http://arxiv.org/abs/1312.6120 (visited on 11/12/2016).

[49] J. Pennington, S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/d9fc0cdb67638d50f411432d0d41d0ba-Abstract.html (visited on 05/17/2023).

[50] J. Pennington, S. S. Schoenholz, and S. Ganguli, "The Emergence of Spectral Universality in Deep Networks," *arXiv:1802.09979 [cs, stat]*, Feb. 2018, arXiv: 1802.09979. [Online]. Available: http://arxiv.org/abs/1802.09979 (visited on 08/02/2019).

[51] E. Cheney and D. Kincaid, *Numerical Mathematics and Computing*, en. Cengage Learning, Aug. 2007, Google-Books-ID: ZUfVZELlrMEC, ISBN: 978-0-495-11475-8.

[52] Y. Bahri, J. Kadmon, J. Pennington, S. S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli, "Statistical Mechanics of Deep Learning," *Annual Review of Condensed Matter Physics*, vol. 11, no. 1, pp. 501–528, 2020. DOI: 10.1146/annurev-conmatphys-031119-050745. [Online]. Available: https://doi.org/10.1146/annurev-conmatphys-031119-050745 (visited on 05/17/2023).

[53] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," en, vol. 15, Apr. 2011, pp. 315–323. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00752497 (visited on 02/23/2019).

[54] M. Chen, J. Pennington, and S. S. Schoenholz, "Dynamical Isometry and a Mean Field Theory of RNNs: Gating Enables Signal Propagation in Recurrent Neural Networks," *arXiv:1806.05394 [cs, stat]*, Aug. 2018, arXiv: 1806.05394. [Online]. Available: http://arxiv.org/abs/1806.05394 (visited on 06/01/2020).

[55] B. Hanin and M. Nica, "Products of Many Large Random Matrices and Gradients in Deep Neural Networks," en, Dec. 2018. [Online]. Available: https://arxiv.org/abs/1812.05994v1 (visited on 03/26/2019).

[56] S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep Information Propagation," Nov. 2016. [Online]. Available: https://openreview.net/forum?id=H1W1UN9gg (visited on 08/02/2019).

[57] B. Hanin and D. Rolnick, "How to Start Training: The Effect of Initialization and Architecture," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/d81f9c1be2e08964bf9f24b15f0e4900-Abstract.html (visited on 10/28/2023).

[58] P. A. Sokol and I. Memming Park, "Information Geometry of Orthogonal Initializations and Training," Tech. Rep., Oct. 2018, ADS Bibcode: 2018arXiv181003785S Type: article. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv181003785S (visited on 05/17/2023).

[59] D. Gilboa, B. Chang, M. Chen, *et al.*, "Dynamical Isometry and a Mean Field Theory of LSTMs and GRUs," en, Jan. 2019. [Online]. Available: https://arxiv.org/abs/1901.08987v1 (visited on 03/26/2019).

[60] T. Can, K. Krishnamurthy, and D. J. Schwab, "Gating creates slow modes and controls phase-space complexity in GRUs and LSTMs," *arXiv:2002.00025 [cond-mat, stat]*, Jan. 2020, arXiv: 2002.00025. [Online]. Available: http://arxiv.org/abs/2002.00025 (visited on 05/27/2020).

[61] D. Doshi, T. He, and A. Gromov, "Critical Initialization of Wide and Deep Neural Networks using Partial Jacobians: General Theory and Applications," en, Nov. 2023. [Online]. Available: https://openreview.net/forum?id=wRJqZRxDEX (visited on 01/16/2024).

[62] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," deu, Ph.D. dissertation, 1991. [Online]. Available: https://www.semanticscholar.org/paper/Untersuchungen-zu-dynamischen-neuronalen-Netzen-Hochreiter/3f3d13e95c25a8f6a753e38dfce88885097cbd43 (visited on 09/23/2021).

[63] S. Hochreiter and Jürgen Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[64] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," Tech. Rep., Sep. 2014, ADS Bibcode: 2014arXiv1409.1259C Type: article. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2014arXiv1409.1259C (visited on 04/08/2023).

[65] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13, Atlanta, GA, USA: JMLR.org, Jun. 2013, pp. III–1310–III–1318. (visited on 06/01/2020).

[66] T. Mikolov, "Statistical language models based on neural networks," Ph.D. thesis, Brno University of Technology, Faculty of Information Technology, Brno, CZ, 2012. [Online]. Available: https://www.fit.vut.cz/study/phd-thesis/283/.

[67] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, en. MIT Press, Nov. 2016, Google-Books-ID: omivDQAAQBAJ, ISBN: 978-0-262-33737-3.

[68] B. Chang, M. Chen, E. Haber, and E. H. Chi, "AntisymmetricRNN: A Dynamical System View on Recurrent Neural Networks," en, International Conference on Learning Representations, Dec. 2018. [Online]. Available: https://openreview.net/forum?id=ryxepo0cFX (visited on 05/12/2023).

[69] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary Evolution Recurrent Neural Networks," en, in *Proceedings of The 33rd International Conference on Machine Learning*, PMLR, Jun. 2016, pp. 1120–1128. [Online]. Available: https://proceedings.mlr.press/v48/arjovsky16.html (visited on 04/08/2023).

[70] K. Helfrich, D. Willmott, and Q. Ye, "Orthogonal Recurrent Neural Networks with Scaled Cayley Transform," en, in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Jul. 2018, pp. 1969–1978. [Online]. Available: https://proceedings.mlr.press/v80/helfrich18a.html (visited on 04/08/2023).

[71] T. Konstantin Rusch and S. Mishra, "Coupled Oscillatory Recurrent Neural Network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies," en, *arXiv e-prints*, arXiv:2010.00951, Oct. 2020. DOI: 10.48550/arXiv.2010.00951. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv201000951K/abstract (visited on 04/08/2023).

[72] N. B. Erichson, O. Azencot, A. Queiruga, L. Hodgkinson, and M. W. Mahoney, "Lipschitz Recurrent Neural Networks," en, International Conference on Learning Representations, Jan. 2021. [Online]. Available: https://openreview.net/forum?id=-N7PBXqOUJZ (visited on 05/12/2023).

[73] A. Gu, K. Goel, and C. Ré, *Efficiently Modeling Long Sequences with Structured State Spaces*, arXiv:2111.00396 [cs], Aug. 2022. DOI: 10.48550/arXiv.2111.00396. [Online]. Available: http://arxiv.org/abs/2111.00396 (visited on 01/16/2024).

[74] J. T. H. Smith, A. Warrington, and S. W. Linderman, *Simplified State Space Layers for Sequence Modeling*, arXiv:2208.04933 [cs], Mar. 2023. DOI: 10.48550/arXiv.2208.04933. [Online]. Available: http://arxiv.org/abs/2208.04933 (visited on 01/16/2024).

[75] A. Orvieto, S. L. Smith, A. Gu, *et al.*, "Resurrecting Recurrent Neural Networks for Long Sequences," Tech. Rep., Mar. 2023, ADS Bibcode: 2023arXiv230306349O Type: article. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2023arXiv230306349O (visited on 04/08/2023).

[76] A. Gu and T. Dao, *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*, arXiv:2312.00752 [cs], Dec. 2023. DOI: 10.48550/arXiv.2312.00752. [Online]. Available: http://arxiv.org/abs/2312.00752 (visited on 01/16/2024).

[77] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34, 2001. [Online]. Available: http://minds.jacobs-university.de/sites/default/files/uploads/papers/EchoStatesTechRep.pdf (visited on 10/30/2016).

[78] M. C. Ozturk, D. Xu, and J. C. Príncipe, "Analysis and Design of Echo State Networks," *Neural Computation*, vol. 19, no. 1, pp. 111–138, Jan. 2007, ISSN: 0899-7667. DOI: 10.1162/neco.2007.19.1.111.

[79] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training Very Deep Networks," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/hash/215a71a12769b056c3c32e7299f1c5ed-Abstract.html (visited on 05/17/2023).

[80] J. G. Zilly, R. K. Srivastava, J. Koutnik, and J. Schmidhuber, "Recurrent Highway Networks," en, in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 4189–4198. [Online]. Available: https://proceedings.mlr.press/v70/zilly17a.html (visited on 05/17/2023).

[81] K. Geist, U. Parlitz, and W. Lauterborn, "Comparison of Different Methods for Computing Lyapunov Exponents," en, *Progress of Theoretical Physics*, vol. 83, no. 5, pp. 875–893, May 1990, ISSN: 0033-068X, 1347-4081. DOI: 10.1143/PTP.83.875. [Online]. Available: http://ptp.oxfordjournals.org/content/83/5/875 (visited on 08/05/2016).

[82] V. I. Oseledets, "A multiplicative ergodic theorem. Characteristic Ljapunov, exponents of dynamical systems," ru, *Trudy Moskovskogo Matematicheskogo Obshchestva*, vol. 19, pp. 179–210, 1968. [Online]. Available: http://mi.mathnet.ru/eng/mmo/v19/p179.

A. Algorithm and Pseudocode for Surrogate Gradient Flossing

We implement *Surrogate Gradient Flossing* by making the established method for calculating Lyapunov exponents differentiable and combining it with analytical surrogate Jacobians. In the established reorthonormalization method, an orthonormal system $\mathbf{Q}_s = [\mathbf{q}_s^1, \mathbf{q}_s^2, \dots, \mathbf{q}_s^k]$ is evolved in the tangent space along the network trajectory using the *surrogate Jacobian* $\mathbf{D}_s^{\text{surrogate}} = \frac{\partial \mathbf{h}_{s+1}}{\partial \mathbf{h}_s}$. This process involves the calculation of: $\widetilde{\mathbf{Q}}_{s+1} = \mathbf{D}_s^{\text{surrogate}} \mathbf{Q}_s$ at each time step. . Subsequently, the exponential growth rates are extracted via QR decomposition: $\widetilde{\mathbf{Q}}_{s+1} = \mathbf{Q}_{s+1}\mathbf{R}^{s+1}$, which decomposes $\widetilde{\mathbf{Q}}_{s+1}$ into an orthonormal matrix $\mathbf{Q}_{s+1}$ and an upper triangular matrix $\mathbf{R}^{s+1}$ with positive diagonal elements.

Surrogate Lyapunov exponents are then determined by the time-averaged logarithms of the diagonal entries of $\mathbf{R}^s$:

The following pseudocode implements the surrogate gradient flossing algorithm, with surrogate gradient function steps in red and conventional forward dynamics in blue.

Algorithm 1 Algorithm for *surrogate gradient flossing* of k surrogate Lyapunov exponents

initialize $\mathbf{h}$, $\mathbf{Q}$
for $e = 1 \to E$ **do**
  for $t = 1 \to T$ **do**
    $\mathbf{h} \leftarrow \mathbf{f}_\theta(\mathbf{h}, \mathbf{x})$
    $\mathbf{D}^{\text{surrogate}} \leftarrow \frac{\mathrm{d}\mathbf{h}_t}{\mathrm{d}\mathbf{h}_{t-1}}$
    $\mathbf{Q} \leftarrow \mathbf{D}^{\text{surrogate}} \cdot \mathbf{Q}$
    if $t \equiv 0 \pmod{t_{\text{ONS}}}$ **then**
      $\mathbf{Q}, \mathbf{R}^{\text{surrogate}} \leftarrow \mathrm{qr}(\mathbf{Q})$
      $\gamma_i^{\text{surrogate}} \mathrel{+}= \log(R_{ii}^{\text{surrogate}})$
    end if
  end for
  $\lambda_i^{\text{surrogate}} = \gamma_i^{\text{surrogate}} / T$
  $\theta_{e+1} \leftarrow \theta_e - \eta \frac{\partial \mathcal{L}_{\text{surrogate flossing}}}{\partial \theta}$
end for

B. Backpropagation Through QR Decomposition

The adjoint of the QR decomposition is given by [8], [27]–[30]

$$\overline{\mathbf{Q}} = \left[\overline{\mathbf{Q}} + \mathbf{Q}\,\mathtt{copyltu}(\mathbf{M})\right]\mathbf{R}^{-T}, \tag{4}$$

where $\mathbf{M} = \mathbf{R}\overline{\mathbf{R}}^T - \overline{\mathbf{Q}}^T\mathbf{Q}$ and the `copyltu` function generates a symmetric matrix by copying the lower triangle of the input matrix to its upper triangle, with the element $[\mathtt{copyltu}(M)]_{ij} = M_{\max(i,j),\min(i,j)}$ [27]–[30]. *Adjoint variable* are written here as $\overline{T} = \partial\mathcal{L}/\partial T$.

Using an analytical pullback is more memory-efficient and less computationally costly than directly performing automatic differentiation through the QR decomposition. We provide implementations of *surrogate gradient flossing* both in Julia (using the Julia package `BackwardsLinalg.jl` by Jinguo Liu available here) and in PyTorch available at https://github.com/RainerEngelken/SurrogateGradientFlossing.

C. Details on *Surrogate Gradient Flossing* for Binary Networks

We consider binary recurrent networks [31], with parallel update of the N units that follows:

$$\mathbf{h}_{s+1} = \mathbf{W}\,\mathrm{sgn}(\mathbf{h}_s) + \mathbf{V}\mathbf{x}_{s+1}. \tag{5}$$

where sgn denotes the sign function. The initial entries of $\mathbf{W}$ are drawn independently from a Gaussian distribution with zero mean and variance $1/N$. In the backward pass of binary networks, we replace the pullback of sgn with the surrogate

function $\phi'_g(x) \coloneqq \mathrm{sech}(gx)^2$. The surrogate Jacobians are given by $\mathbf{D}_s^{\text{surrogate}} = \frac{\partial \mathbf{h}_{s+1}}{\partial \mathbf{h}_s} = \mathbf{W} \odot \phi'(\mathbf{h}_s)$, where $\odot$ denotes the Hadamard product.

For large random networks, the largest Lyapunov exponent $\lambda_{\max}$ can be obtained from dynamic mean-field theory, as described earlier in both the discrete and continuous-time cases with constant input and additive Gaussian white noise drive [32]–[36] or other time-varying inputs [35], [37], [38]. These insights can be leveraged for surrogate gradient training. For large binary networks, not only the first but also the full *surrogate Lyapunov spectrum* can be approximated analytically at initialization for arbitrary g [4] based on results on product of uncorrelated Gaussian matrices, whose eigenvalue distribution approximately follows a triangle law [39], [40]. This results in:

$$\lambda_i = \begin{cases} \lambda_{\max} + \frac{1}{2}\log\left(1 - \frac{i}{pN}\right), & i \leq pN \\ -\infty, & i > pN \end{cases} \tag{6}$$

where $p = \mathrm{erf}\left(\frac{1}{\sqrt{2}g\Delta_0}\right)$ is the average fraction of neurons that are not in the saturated regime of the surrogate gradient. This fraction depends on both the sharpness g and the variance of h, which can be obtained using dynamic mean-field theory [4], [34]. This extends earlier work [4], [41], [42], which used the triangular law for the complete Lyapunov spectrum of the forward dynamics of discrete-time recurrent neural networks, to an analysis of the gradient norm and condition number in the backward pass. We provide code for *surrogate gradient flossing* in binary networks at https://github.com/RainerEngelken/SurrogateGradientFlossing. We find that *surrogate gradient flossing* also helps train binary networks on tasks with many time steps, such as the copy task and the temporally delayed XOR task. A full analytical description of *surrogate gradient flossing* for binary networks would be a promising avenue for future research, as networks with binary dynamics can still have complex nonlinear learning dynamics. Moreover, quantized networks are increasingly of interest because of their memory efficiency. However, this is beyond the scope of the work presented here.

D. Additional Details on Spiking Network Model and Parameter Initialization

In the case of spiking networks, we considered a time-discretized implementation of a network of N leaky integrate-and-fire neurons with exponentially decaying synaptic currents, following the notation of [2], [26]:

$$U_i[n+1] = (\beta U_i[n] + (1-\beta) I_i[n])(1 - S_i^{\text{rec}}[n]) \tag{7}$$

where $\beta \equiv \exp\left(-\frac{\Delta_t}{\tau_{\text{mem}}}\right)$ reflects the decay due to leak, U_i is the membrane potential of neuron i, I_i is the synaptic current. Similarly, the synaptic currents are updated as follows:

$$I_i[n+1] = \alpha I_i[n] + \sum_j V_{ij} S_j^{\text{rec}}[n] + \sum_j W_{ij} S_j^{\text{in}}[n] \tag{8}$$

Unless noted otherwise, we used the default parameters $\Delta_t = 2$ ms, $\tau_{\text{mem}} = 10$ ms, and $\tau_{\text{mem}} = 5$ ms. We chose these time constants for both the recurrent neurons and the readout neurons. Our implementation in Julia [21] using Flux [22] is available at https://github.com/RainerEngelken/SurrogateGradientFlossing. We initialized the recurrent network in the fluctuation-driven regime following [26] by adjusting the input weights W_{ij} and recurrent weights V_{ij} to induce a fluctuation-driven regime at initialization.

We note that during task training, LIF networks can develop unphysiological negative voltage excursions if the input variance becomes large, inducing bursty network activity states with a coefficient of variation of the interspike interval distribution larger than one, indicating super-Poissonian spiking statistics [43], [44]. It will be important to further study surrogate gradient flossing and spiking network training when constraining networks to a biologically plausible activity regime and when imposing further biological constraints like Dale's principle and dynamical balance of excitatory and inhibitory input currents [45], [46].

E. Analytical Surrogate Jacobian of the Spiking Neural Network

To calculate *surrogate Lyapunov exponents*, we first need to obtain the surrogate Jacobian, $\mathbf{D}_n^{\text{surrogate}} = \frac{\partial \mathbf{h}_{n+1}}{\partial \mathbf{h}_n}$, either through automatic differentiation or analytically. The surrogate Jacobians evaluated along the forward trajectory describe how an

infinitesimal perturbation of the network state (e.g., here all membrane potentials and synaptic currents) evolves in one time step.

Here, we give the analytical expression of the $2N \times 2N$ surrogate Jacobian for the discretized dynamics of a recurrent network of leaky integrate-and-fire neurons with exponential synapses. We only consider the dynamics of the recurrent neurons and not of the readout neurons, as these are nonspiking and only passively driven and therefore have trivial *surrogate Lyapunov exponents* of the relaxation time constants $-\frac{1}{\tau_{\text{syn}}}$ and $-\frac{1}{\tau_{\text{mem}}}$.

$$\delta \mathbf{h}_{n+1} = \mathbf{D}_n^{\text{surrogate}} \delta \mathbf{h}_n \tag{9}$$

The surrogate Jacobian matrix, $\mathbf{D}_n^{\text{surrogate}}$, is block-structured, according to the partial derivatives of the membrane potentials and synaptic currents between subsequent time steps:

$$\mathbf{D}_n^{\text{surrogate}} = \begin{pmatrix} \frac{\partial \mathbf{U}[n+1]}{\partial \mathbf{U}[n]} & \frac{\partial \mathbf{U}[n+1]}{\partial \mathbf{I}[n]} \\ \frac{\partial \mathbf{I}[n+1]}{\partial \mathbf{U}[n]} & \frac{\partial \mathbf{I}[n+1]}{\partial \mathbf{I}[n]} \end{pmatrix} \tag{10}$$

Often in surrogate gradient training, the voltage reset term $(1 - S_i)$ is ignored in the backward pass [2]. Here, we provide the surrogate Jacobian entries both including and excluding the voltage reset term. If the voltage reset is ignored, the Jacobian entries are given by:

$$\frac{\partial U_i[n+1]}{\partial U_j[n]} = \beta \delta_{ij} + V_{ij} \phi_g'(U_j[n] - \vartheta) \tag{11}$$

$$\frac{\partial U_i[n+1]}{\partial I_j[n]} = \alpha \delta_{ij} \tag{12}$$

$$\frac{\partial I_i[n+1]}{\partial U_j[n]} = V_{ij} \phi_g'(U_j[n] - \vartheta) \tag{13}$$

$$\frac{\partial I_i[n+1]}{\partial I_j[n]} = \alpha \delta_{ij} \tag{14}$$

$\phi_g'(x)$ is the surrogate gradient function. We use here the derivative of the fast sigmoid $\phi_g'(x) = \frac{1}{(g|x|+1)^2}$ [47]. Note that we absorbed the term $(1 - \beta)$ from Eq 7 into the input weights and recurrent weights.

If the voltage reset term is not ignored in the derivative, the Jacobian entries are:

$$\begin{aligned} \frac{\partial U_i[n+1]}{\partial U_j[n]} = &\left(\delta_{ij}\beta + V_{ij}\phi_g'(U_j[n] - 1)\right)(1 - S_i[n]) \\ &- \left(\beta U_i[n] + \alpha I_i[n] + x_i[n] + I_i^{\text{rec}}[n]\right)\phi_g'(U_i[n] - 1), \end{aligned} \tag{15}$$

$$\frac{\partial U_i[n+1]}{\partial I_j[n]} = \delta_{ij}\alpha\left(1 - S_i[n]\right), \tag{16}$$

$$\frac{\partial I_i[n+1]}{\partial U_j[n]} = V_{ij}\phi_g'(U_j[n] - 1), \tag{17}$$

$$\frac{\partial I_i[n+1]}{\partial I_j[n]} = \delta_{ij}\alpha. \tag{18}$$

where $I_i^{\text{rec}}[n] = \sum_j V_{ij}\phi_g(U_j[n] - \vartheta)$ and $x_i[n] = \sum_j W_{ij} S_j^{\text{in}}[n]$.

To corroborate these analytical expressions, we compare them in the following to expressions of the Jacobian obtained through automatic differentiation:

F. Linking Surrogate Lyapunov Exponents & Gradient Norm

We here show in more detail that *surrogate Lyapunov exponents* give access to the norm and structure of the surrogate gradients for long time horizons, by utilizing a recently discovered link between Lyapunov exponents of RNNs and the exploding and vanishing gradient problem for the case of surrogate gradient training of spiking networks. [4], [6]–[11].

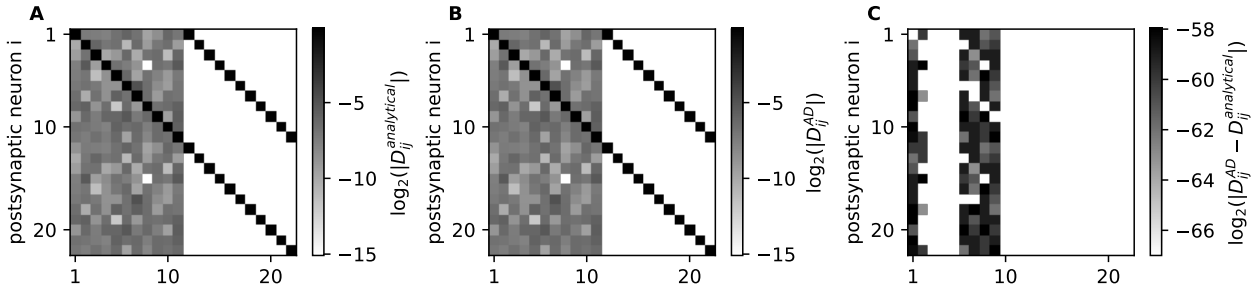

Figure 4. **Comparison of analytical and numerical *surrogate Jacobians* A)** Analytical surrogate Jacobian for $g = 1$. **B)** surrogate Jacobian obtained from automatic differentiation for $g = 10$. **C)** $\log_2$ of absolute value of difference between AD and analytical surrogate Jacobian. The difference is close to machine precision. $\log_2(|D_{ij}^{AD} - D_{ij}^{\text{analytical}}|)$

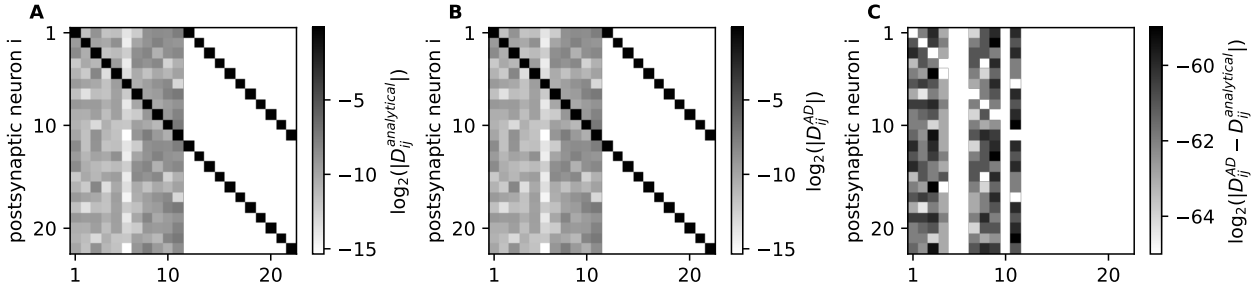

Figure 5. Same as 4 for $g = 10$

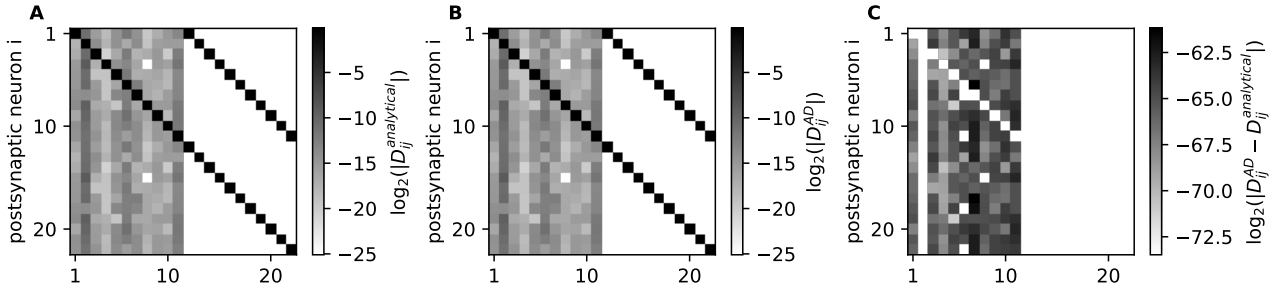

Figure 6. Same as 4 for $g = 100$

Note that the complete error gradient of backpropagation through time in Eq 2 is composed of a summation of products of surrogate Jacobians, reflecting the number of "loops" the error signal traverses through the recurrent spiking dynamics before reaching its target. Consequently, when the leading singular values of the surrogate Jacobians are smaller than 1, the influence of the shorter loops typically dominates the surrogate gradients.

In the case of vanishing surrogate gradients, the gradient norm is dominated by the shorter loops, even though the actual signal in the surrogate gradient originates from the loop of duration T, where T is the maximum temporal delay relevant for solving the temporal credit assignment of the task. To mitigate the contamination of spurious signals from shorter loops and effectively extract the gradient that spans long time horizons, we focus on the gradient at the last time point $\mathcal{L}_T$ with respect to the initial conditions $\mathbf{h}_0$.

$$\frac{\partial \mathcal{L}_T}{\partial \mathbf{h}_0} = \frac{\partial \mathcal{L}_T}{\partial \mathbf{h}_T} \sum_{\tau=0}^{\tau=T-1} \left(\prod_{\tau'=\tau}^{T-1} \frac{\partial \mathbf{h}_{\tau'+1}}{\partial \mathbf{h}_{\tau'}} \right) \frac{\partial \mathbf{h}_\tau}{\partial \mathbf{h}_0} = \frac{\partial \mathcal{L}_T}{\partial \mathbf{h}_T} \prod_{\tau'=0}^{T-1} \frac{\partial \mathbf{h}_{\tau'+1}}{\partial \mathbf{h}_{\tau'}} \tag{19}$$

We note that the sum conveniently drops the only summand that contributes is the product of surrogate Jacobians going from 0 to T. We note that we considered the binary cross entropy loss which makes the derivative $\frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_t}$ trivial. Using the

definition of *surrogate Lyapunov exponents* in Eq 3 and the definition of the spectral norm, we obtain

$$\left|\prod_{\tau'=0}^{T-1}\frac{\partial \mathbf{h}_{\tau'+1}}{\partial \mathbf{h}_{\tau'}}\right|_2 = \sup_{\|\mathbf{x}\|_2=1}\|\prod_{\tau'=0}^{T-1}\frac{\partial \mathbf{h}_{\tau'+1}}{\partial \mathbf{h}_{\tau'}}\mathbf{x}\|_2 = \sigma_1\left(\prod_{\tau'=0}^{T-1}\mathbf{D}_{\tau'}^{\text{surrogate}}\right) \approx \exp(\lambda_1^S T)$$

we can thus approximate the surrogate gradient norm $|\frac{\partial \mathcal{L}_T}{\partial \mathbf{h}_0}| \propto \exp(\lambda_1^S T)$.

G. Linking Surrogate Lyapunov Exponents with Gradient Dimensionality

We pointed out in the main text that a *surrogate Lyapunov exponents* correspond to exponentially exploding gradient modes, while negative *surrogate Lyapunov exponents* correspond to exponentially vanishing surrogate gradient modes.

A well-conditioned surrogate Jacobian is essential for efficient and fast learning [48]–[50]. *Surrogate gradient flossing* improves the condition number of the long-term Jacobian which constrains the error signal propagation across long time horizons in backpropagation [3].The condition number κ_2 of a linear map A measures how close the map is to being singular and is given by the ratio of the largest singular value σ_{max} and the smallest singular values σ_{min}, so $\kappa_2(A) = \frac{\sigma_{\text{max}}(A)}{\sigma_{\text{min}}(A)}$. According to the rule of thumb given in [51], if $\kappa_2(A) = 10^p$, one can anticipate losing at least p digits of precision when solving the equation $Ax = b$. Note that the long-term Jacobian $\mathbf{T}_t$ is composed of a product of surrogate Jacobians, which generically makes it ill-conditioned.

Our theoretical estimate of the condition number κ_2 of an orthonormal system $\mathbf{Q}$ of size $N \times m$ that is temporally evolved by the long-term Jacobian $\mathbf{T}_t$

$$\kappa_2(\widetilde{\mathbf{Q}}_{t+\tau}) = \kappa_2\big(\mathbf{T}_t(\mathbf{h}_\tau)\mathbf{Q}_t\big) = \frac{\sigma_1(\mathbf{T}_t(\mathbf{h}_\tau))}{\sigma_m(\mathbf{T}_t(\mathbf{h}_\tau))} \approx \exp\left((\lambda_1 - \lambda_m)(t-\tau)\right). \tag{20}$$

where $\sigma_1(\mathbf{T}_t(\mathbf{h}_\tau))$ and $\sigma_m(\mathbf{T}_t(\mathbf{h}_\tau))$ are the first and mth singular value of the long-term Jacobian. We note that this theoretical estimate of the condition number follows from the asymptotic definition of Lyapunov exponents and should be exact in the limit of long times. Given that *gradient flossing* reduces the condition number by a factor whose magnitude increases exponentially with T, we can expect that *surrogate gradient flossing* has a stronger effect on problems with a long time horizon to bridge.

Moreover, *surrogate Lyapunov exponents* enable the estimation of the number of gradient dimensions available for the backpropagation of error signals. Generally, the long-term Jacobian is ill-conditioned, however, the Lyapunov spectrum provides for a given number of tangent space dimensions an estimate of the condition number. This indicates how close to singular the gradient signal for a given number of tangent space dimensions is. Given a fixed acceptable condition number—determined, for example, by noise level or floating-point precision—we observe that *gradient flossing* increases the number of usable tangent space dimensions for backpropagation.

We stress that these insights go beyond mean-field methods, which usually only consider uncorrelated i.i.d. weight matrices and become exact only in the large-network limit $N \to \infty$ [52].

H. Further Relation to Previous Literature

Generally, most previous analyses of the notorious vanishing and exploding gradient problem addressed differentiable neural networks and considered different strategies against vanishing and exploding gradients, for instance suitable choice of weights that guarantee well-conditioned Jacobian at initialization [23], [48], [53]–[61], specialized neuron models whose gating interactions shield the latent memory state and which can therefore transport information across long time horizons steps [62]–[64] or pragmatic training tricks like gradient clipping, which re-scales the gradient norm [65] or their individual elements [66] if they become too large [67]. Moreover, specialized network architectures were introduced to tackle gradient issues, for example, antisymmetric networks [68], orthogonal/unitary initializations [48], [69], [70], coupled oscillatory RNNs [71], Lipschitz RNNs [72], state-space models [73]–[76], echo state networks [77], [78], (recurrent) highway networks [79], [80], and stable limit cycle neural networks [9]–[11]. The work presented here can be seen as complementary to the latter.

I. Further Details and Analysis of Surrogate Gradient Flossing

An example implementation of *surrogate gradient flossing* in PyTorch and Flux [22], a machine learning library in Julia [21] is available at https://github.com/RainerEngelken/SurrogateGradientFlossing. We also provide code to directly reproduce Fig 1.

J. Computational Complexity of Surrogate Gradient Flossing

We present here a more in-depth scaling analysis of the computational cost of *surrogate gradient flossing*. There are three main contributors to the computational cost (table 1): First, the forward dynamics of the spiking RNN, which has a computational complexity of $\mathcal{O}\left(N^2\, b\right)$ per time step, where N is the dimension of the recurrent network state (which in the case of leaky integrate-and-fire neurons with exponentially decaying synapses is twice the number of units) and b is the batch size both in the forward and backward pass. Second, the Jacobian step which scales with $\mathcal{O}\left(N^2 k\right)$ per time step, where k is the number of *flossed surrogate Lyapunov exponents*. Third, the QR decomposition, which scales with $\mathcal{O}\left(N\ k^2\right)$, where k is the number of *surrogate Lyapunov exponents* considered.

Together, this results in a total amortized cost of $\mathcal{O}\left(N^2\, b\, T\right)$ per training epoch, where T is the number of training time steps and a total amortized costs per *flossing* epoch of $\mathcal{O}\left(N^2\, T_f(1 + k/t_{\text{ONS}} + k)\right)$ where T_f is the number of *flossing* time steps.

In the case of *surrogate preflossing*, thus, the total computation cost scale with $\mathcal{O}\left(N^2[Eb\, T + E_p\, T_f(1 + k/t_{\text{ONS}} + k)]\right)$, where E is the number of training epochs and E_p is the number of *surrogate preflossing* epochs.

For *surrogate gradient flossing* during training (assuming that there is also *surrogate preflossing* done), the amortized cost scale with $\mathcal{O}\left(N^2[Eb\, T + E_p\, T_p + E_f\, T_f(1 + k/t_{\text{ONS}} + k)]\right)$, where E_f is the total number of *flossing* epochs during training.

We note that as long as $k < \sqrt{N}$ the QR decomposition is not the computational bottleneck.

Moreover, we find empirically that both the number of *surrogate preflossing* epochs E_p and *surrogate flossing* episodes E_f necessary for training success is much smaller than the total number of training epochs E. It remains an important challenge to infer the suitable number of *surrogate flossing* time steps T_f for tasks with unknown temporal correlation structure.

It would also be interesting to investigate how the CPU hours/wall-clock time/flops/Joule/CO2-emission spent on *surrogate gradient flossing* vs on training networks with larger N are trading off against each other. For this, we would suggest first finding the smallest network that on median successfully trains on a binary temporal XOR task for a fixed given delay T and measuring the computational resources involved in training it, e.g. in terms of CPU hours. Then compare it to a network with *surrogate gradient flossing*.

K. Additional Background on Lyapunov Exponents of RNNs

An autonomous dynamical system is usually defined by a set of ordinary differential equations $\mathrm{d}\mathbf{h}/\mathrm{d}t = \mathbf{F}(\mathbf{h}),\ \mathbf{h} \in \mathbb{R}^N$ in the case of continuous-time dynamics, or as a map $\mathbf{h}_{s+1} = \mathbf{f}(\mathbf{h}_s)$ in the case of discrete-time dynamics. In the following, the theory is presented for discrete-time dynamical systems for ease of notation, but everything directly extends to continuous-time systems [81]. Together with an initial condition $\mathbf{h}_0$, the map forms a trajectory. As a natural extension of linear stability analysis, one can ask how an infinitesimal perturbation $\mathbf{h}_0' = \mathbf{h}_0 + \epsilon\mathbf{u}_0$ evolves in time. Chaotic systems are sensitive to initial conditions; almost all infinitesimal perturbations $\epsilon\mathbf{u}_0$ of the initial condition grow exponentially with time $|\epsilon\mathbf{u}_t| \approx \exp(\lambda_1 t)|\epsilon\mathbf{u}_0|$. Finite-size perturbations, therefore, may lead to a drastically different subsequent behavior. The largest Lyapunov exponent λ_1 measures the average rate of exponential divergence or convergence of nearby initial conditions:

$$\lambda_1(\mathbf{h}_0) = \lim_{t\to\infty} \frac{1}{t} \lim_{\epsilon\to 0} \log \frac{||\epsilon\mathbf{u}_t||}{||\epsilon\mathbf{u}_0||} \tag{21}$$

In dynamical systems that are ergodic on the attractor, the Lyapunov exponents do not depend on the initial conditions as long as the initial conditions are in the basins of attraction of the attractor. Note that it is crucial to take the limit $\epsilon \to 0$ first and then $t \to \infty$, as $\lambda_1(\mathbf{h}_0)$ would be trivially zero for a bounded attractor if the limits are exchanged, as $\lim_{t\to\infty} \log \frac{||\epsilon\mathbf{u}_t||}{||\epsilon\mathbf{u}_0||}$ is bounded for finite perturbations even if the system is chaotic. To measure k Lyapunov exponents, one has to study the evolution of k independent infinitesimal perturbations $\mathbf{u}_s$ spanning the tangent space:

	forward pass	**backward pass**
RNN dynamics	$\mathcal{O}\left(N^2\, b\right)$	”
Jacobian step	$\mathcal{O}\left(N^2\, k\right)$	”
QR step	$\mathcal{O}\left(N\, k^2\right)$	”
total amortized costs per training epoch	$\mathcal{O}\left(N^2\, b\, T\right)$	”
total amortized costs per *surrogate gradient flossing* epoch	$\mathcal{O}\left(N^2\, T_f(1 + k/t_{\text{ONS}} + k)\right)$	”
total amortized costs of *surrogate preflossing*	$\mathcal{O}\left(N^2[Eb\, T + E_p\, T_f(1 + k/t_{\text{ONS}} + k)]\right)$	”
total amortized costs *flossing* during training	$\mathcal{O}\left(N^2[Eb\, T + E_p\, T_p + E_f\, T_f(1 + k/t_{\text{ONS}} + k)]\right)$	”

Table 1. **Computational cost for *surrogate gradient flossing* and training of RNNs**
N denotes number of neurons, b is the batch size, T is the number of time steps in forward pass of training, T_f is the number of time steps in forward pass of *flossing*, t_{ONS} is the reorthonormalization interval, k is the number of *flossed* Lyapunov exponents, E is the number of training epochs, E_p is the number of *surrogate preflossing* epochs, E_f is the number of *flossing* epochs during training. Empirically, we find that the necessary number of *surrogate preflossing* epochs E_p and *flossing* episodes E_f is much smaller than both the total number of training epochs E. Moreover, T_p can be smaller than T.

$$\mathbf{u}_{s+1} = \mathbf{D}_s \mathbf{u}_s \tag{22}$$

where the $N \times N$ Jacobian $\mathbf{D}_s(\mathbf{h_s}) = \mathrm{d}\mathbf{f}(\mathbf{h_s})/\mathrm{d}\mathbf{h}$ characterizes the evolution of generic infinitesimal perturbations during one step. Note that this Jacobian along the trajectory is equivalent to a stability matrix only at a fixed point, i.e., when $\mathbf{h}_{s+1} = \mathbf{f}(\mathbf{h}_s) = \mathbf{h}_s$.

We are interested in the asymptotic behavior, and therefore we study the long-term Jacobian

$$\mathbf{T}_t(\mathbf{h}_0) = \mathbf{D}_{t-1}(\mathbf{h}_{t-1}) \dots \mathbf{D}_1(\mathbf{h}_1)\mathbf{D}_0(\mathbf{h}_0). \tag{23}$$

Note that $\mathbf{T}_t(\mathbf{h}_0)$ is a product of generally noncommuting matrices. The Lyapunov exponents $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_N$ are defined as the logarithms of the eigenvalues of the Oseledets matrix

$$\mathbf{\Lambda}(\mathbf{h}_0) = \lim_{t\to\infty} [\mathbf{T}_t(\mathbf{h}_0)^\top \mathbf{T}_t(\mathbf{h}_0)]^{\frac{1}{2t}}, \tag{24}$$

where $\top$ denotes the transpose operation. The expression inside the brackets is the Gram matrix of the long-term Jacobian $\mathbf{T}_t(\mathbf{h}_0)$. Geometrically, the determinant of the Gram matrix is the squared volume of the parallelotope spanned by the columns of $\mathbf{T_t}(\mathbf{h_0})$. Thus, the exponential volume growth rate is given by the sum of the logarithms of its first k (sorted) eigenvalues. Oseledets' multiplicative ergodic theorem guarantees the existence of the Oseledets matrix $\mathbf{\Lambda}(\mathbf{h}_0)$ for almost all initial conditions $\mathbf{h}_0$ [82]. In ergodic systems, the Lyapunov exponents λ_i do not depend on the initial condition $\mathbf{h}_0$. However, for a numerical calculation of the Lyapunov spectrum, Eq 24 cannot be used directly because the long-term Jacobian $T_t(\mathbf{h}_0)$ quickly becomes ill-conditioned, i.e., the ratio between its largest and smallest singular value diverges exponentially with time.

L. Convergence of Surrogate Lyapunov Exponents of discretized Spiking Neural Network

In Fig 7, we demonstrate the convergence of the *surrogate Lyapunov exponents*. We show ing Fig 7A the estimate of the *surrogate Lyapunov exponents* λ_i^S for $i = 1, 20, 60, 80$ for different random seeds for both the network weights, the input weights, the random input, the orthonormal system and the initial conditions of the network state. **B** Shows the same, but for one fixed network weight realization, where the seed of input, the initial conditions, and the orthonormal system are varied. We note that with fixed network parameters, the different estimates of the *surrogate Lyapunov exponent* converge tighter.

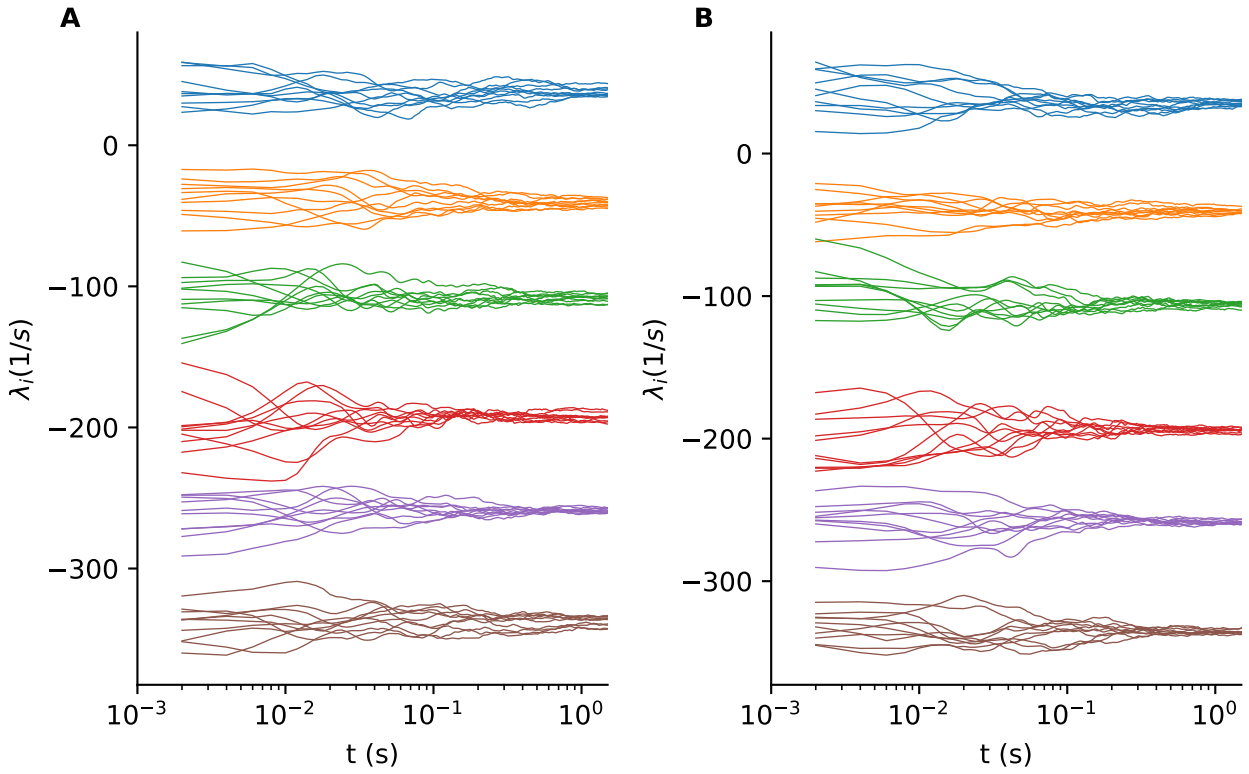

Figure 7. **Convergence of Lyapunov exponents for** $g = 1$ **A** Convergence of selected Lyapunov exponents λ_i for ten identical network realizations with different initial conditions with simulation time ($i = 1, 26, 52, 77, 103, 128$) for $g = 1$. **B** Same as **A** but for fixed network parameters. (Other parameters: $N = 64$, $t_{\text{sim}} = 1.5$ seconds, $t_{\text{ONS}} = 1$).

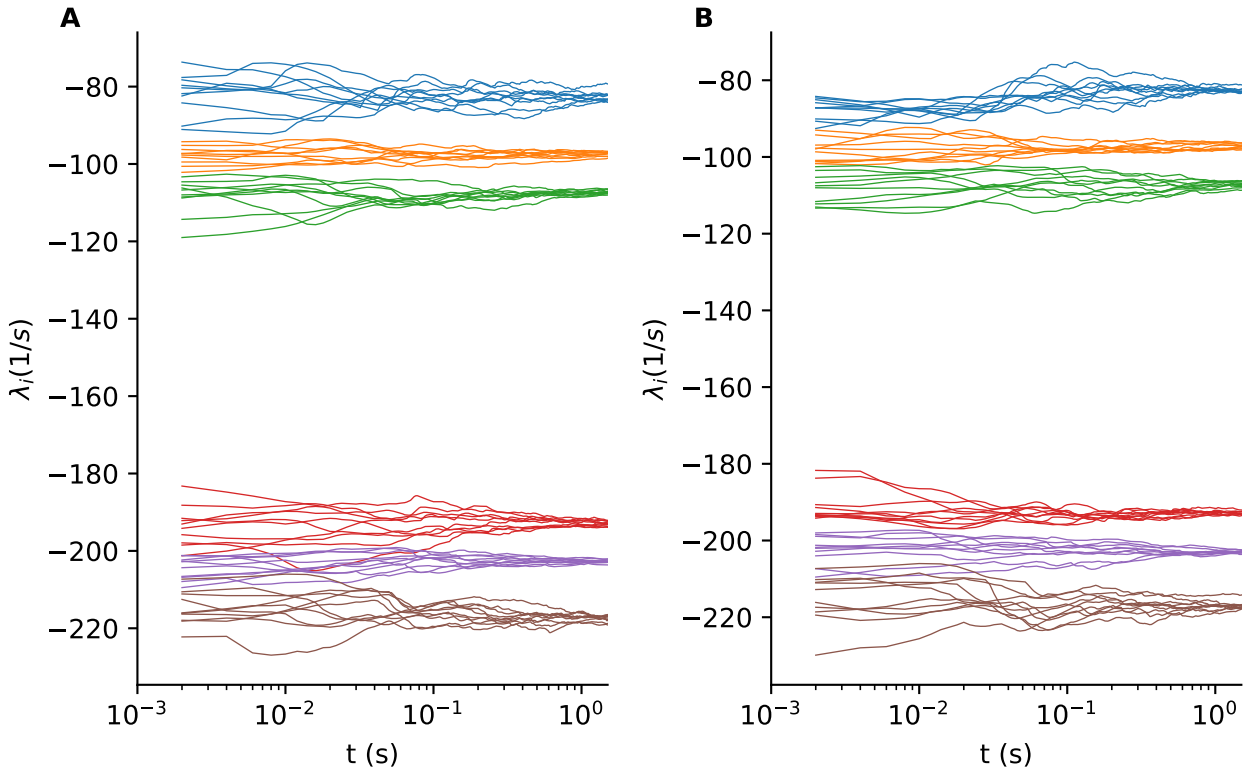

Figure 8. **Convergence of Lyapunov exponents for** $g = 20$ **A** Convergence of selected Lyapunov exponents λ_i for ten identical network realizations with different initial conditions with simulation time ($i = 1, 26, 52, 77, 103, 128$) for $g = 1$. **B** Same as **A** but for fixed network parameters. (Other parameters: $N = 64$, $t_{\text{sim}} = 1.5$ seconds, $t_{\text{ONS}} = 20$).