Goal-driven optimization of single-neuron properties in artificial networks reveals regularization role of neural diversity and adaptation

Victor Geadah [1,2] **Stefan Horoi** [2,3] **Giancarlo Kerg** [2,4]

Guy Wolf [2,3,5] **Guillaume Lajoie** [2,3,5,*]

Abstract

Individual neurons, and the circuits they collectively form in the brain, have been subject to joint evolutionary pressure to produce system-level functions. Considerable effort has been invested in understanding the impact of single-neuron input-output mechanisms, such as diversity in f-I curves and spike frequency adaptation, on network computations. Yet, how goal-driven requirements at the network level influence single-neuron coding properties remains largely unexplored. Toward addressing this, we systematically investigate single-neuron input-output adaptive mechanisms, optimized in an end-to-end fashion in artificial recurrent neural networks. This is achieved by interconnected *Adaptive Recurrent Units* (ARU), which perform online control of a novel two-parameter family of activation functions mimicking the diversity of f-I curves found in common neural types in the brain. Our network of ARUs shows much-improved robustness to noise and changes in input statistics. Importantly, we find that ARUs recover precise biological coding strategies such as gain scaling and fractional order differentiation. Using tools from dynamical systems theory, we elucidate the role of these emergent single neuron properties and argue that neural diversity and adaption likely play an active regularization role that enables neural circuits to optimally propagate information across time. In doing so, we discuss how goal-driven optimization approaches, while not biologically plausible themselves, reveal neural mechanisms that are consistent with evolutionary pressures on the brain.

1 Introduction

Biological neurons show an outstanding range of input response diversity and adaptive behavior (Gjorgjieva et al., 2016; Weber et al., 2019). How the rich dynamics of biological neurons combine with network interactions to support complex tasks, such as sensory integration and behavior, remains largely unresolved. While the past decades have seen considerable work aimed at elucidating single neuron coding properties, most efforts have been “bottom up", modeling mechanistic features observed in biology and analyzing their computational impact. We argue that to shed light on the system-level role of single neuron properties, a “top-down" approach is needed. One way to achieve this is with deep-learning optimization, where “goal-driven" models aim to solve system-level objectives, and emergent neuron properties are studied. In recent years, this method has been extremely successful in capturing single neuron static tuning properties, such as that in the visual system (Yamins and DiCarlo, 2016). In this work, we use a goal-driven approach to investigate adaptive

[1]Program in Applied and Computational Mathematics, Princeton University, Princeton, U.S.A. [2]Mila - Quebec Artificial Intelligence Institute, Montréal, Canada. [3]Département de Mathématiques et Statistiques, Université de Montréal, Montréal, Canada. [4]Département d’Informatique et Recherche Opérationelle, Université de Montréal, Montréal, Canada. [5]Canada CIFAR AI Chair. [*]Correspondence should be addressed to G.L. (g.lajoie@umontreal.ca).

Preprint. Under review.

input-output properties of neurons that emerge from end-to-end optimization of recurrent neural networks, and shed light on their role in biological systems.

A central dynamic component of single-neuron coding is the transformation of input currents into output firing rates neuron execute, as measured by so called *f-I curves*, or *activation functions* (AF). These are both adaptive and diverse across neurons. At the heart of this modularity lies the efficient coding hypothesis, a theoretical paradigm by which neurons aim to be maximally informative about the inputs they encode. Supported by this principle, neurons are known to effectively modulate their f-I curve in response to constant step-like stimulus, in a process know as *spike frequency adaptation* (SFA) (Benda and Herz, 2003). It has been shown that SFA and other adaptive mechanisms in single neurons enable faithful encoding of input signals regardless of baseline, a crucial feature for animals subject to changing environments (Fairhall et al., 2001a; Peron and Gabbiani, 2009; Gjorgjieva et al., 2016). SFA also facilitates information integration over long timescales (Pozzorini et al., 2015), and provides robustness to rapid variation and noise (Lundstrom et al., 2008). At the network level, adaptive neural responses have been shown to support efficient coding with metabolic advantages (Gutierrez and Denève, 2019), facilitate computations over long timescales (Bellec et al., 2018; Fitz et al., 2020; Salaj et al., 2021), and even enable forms of Bayesian inference (Deneve, 2008; Kilpatrick and Ermentrout, 2011). Recent work also shows robustness gains from learned modulated neural dynamics (Vecoven et al., 2020) and with diverse and dynamics synapses and IF curves (Burnham et al., 2021; Winston et al., 2022a), While a number of coding advantages of diverse and dynamic single neuron responses are now established, it is still unknown how these mechanisms have come to bear, and how they influence learning and configuration of larger neural networks that support system-level tasks such as perception or prediction.

In parallel, modern artificial neural networks used in artificial intelligence (AI) loosely mimic neural responses with simple AFs (also called *nonlinearities*) which transform summed inputs to an artificial neuron into a scalar state value, akin to a firing rate. While different shapes of activation functions have been used, and even optimized (Hayou et al., 2019), the prevailing sentiment in AI is that a simple AF such as the *rectified linear unit* (ReLU) is enough for large networks to implement almost any transformation. In fact, this is mathematically guaranteed by the universal function approximation theorem, stating that large enough nonlinear neural networks can implement any function (Cybenko, 1989; Hornik et al., 1989). Reconciling the diverse and dynamic nature of biological neurons' input-output properties with the computational function of the large networks in the mammalian brain, for example, is therefore a tricky exercise. The prevalent hypothesis is that the single neuron input-output richness found in the brain has evolved and been optimized to guide network-level function such as stable population dynamics, and coordinated learning.

In this work, we propose a step towards complementing these longstanding mechanistic investigative efforts into IF-response diversity and adaptation, through the lens of goal-driven optimization. Using simple artificial neural networks and deep learning, we ask: given the possibility to implement a wide range of single neuron input-output properties, including rapid adaptive mechanisms, do networks optimized end-to-end to perform systems-level goals develop biologically realistic solutions at the single neuron level? If so, can we reconcile single-neuron properties with network-level mechanisms? To address this, we concentrate on the problem of perception on sequential stimuli, such as visual input streams. Our goal is to prescribe the simplest system possible recurrent neural network (RNN) that has enough flexibility to develop optimal solutions for it's units' AFs. As such, we propose a two-parameter family of AFs mimicking the diversity of f-I curves that can be implemented by known neural types, and interpolating between often used nonlinearities in AI. In addition, we implement a dynamic controller that modulates AFs in real time, acting locally and independently at each neuron. This controller, implemented with a distinct and smaller RNN, models the genetically encoded adaptation strategy that would have been refined by evolution. We then train this system end-to-end on sequential classification tasks. We call our novel adaptive artificial neuron *Adaptive Recurrent Unit* (ARU).

Our findings can be summarized in three points. First, we find that both diverse and adaptive AFs help the main RNN learn tasks, and provide surprising robustness to noise and distractors. Second, we investigate the learned solutions obtained by the optimization procedure and find that surprisingly, a number of biologically realistic strategies are implemented. Indeed, optimal AFs take on biologically plausible configurations (i.e. not simple sigmoid or ReLU), diversity of AFs is an important and necessary feature for robustness, and crucially, the adaption controller implements gain scaling and fractional differentiation, just like several neocortical neurons. Finally, we analyze the optimization mechanism that led to these solutions and find that diversity and adaptation acts as a dynamic regularizer, enabling the main RNN to remain in a dynamic regime close to the edge of chaos where information transmission and error gradients propagate optimally.

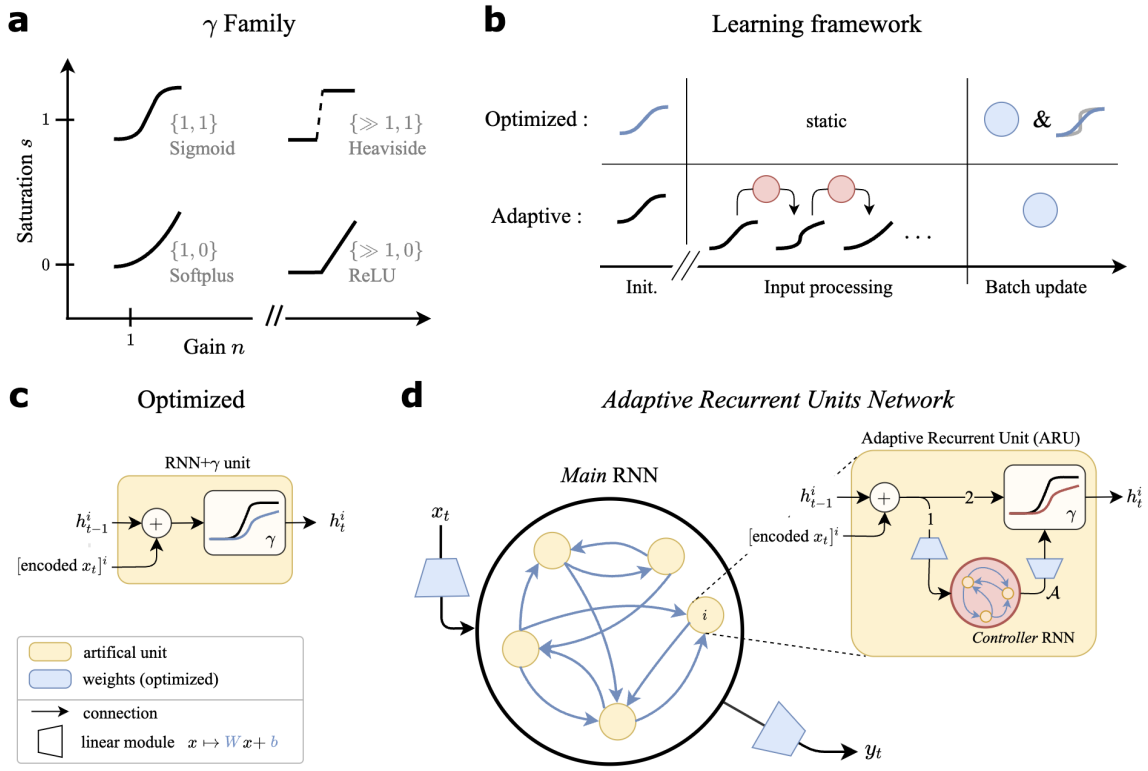

Figure 1: *Model details*. (**a**) Various shapes of the γ activation function (AF), represented in the parameter space $\{n, s\}$. We attain commonly used nonlinearities. (**b**) Different settings considered for modulation of the AF. This modulation is either done alongside training (*Optimized*) or online (*Adaptive*). Blue indicates learned parameters ; notice the blue activation function in the *Optimized* setting, updated alongside weights. (**c**) Artificial unit i of a standard RNN with γ activation function. Legend at the bottom applies to (**c**-**d**). (**d**) Graphical depiction of the Adaptive Recurrent Units (ARU) and associated recurrent network model. Numbers $\{1, 2\}$ on the arrows in the ARU represent the order of processing. Removing the adaptation mechanism $\mathcal{A}$, we recover the RNN+γ model in **c**. Each neuron has a private copy of the controller RNN (in red).

2 Results

2.1 Static and adaptive activation functions in recurrent neural network models

We propose a novel, differentiable family of activation functions defined by

$$\gamma(x; n, s) = (1 - s)\frac{\log(1 + e^{nx})}{n} + s\frac{e^{nx}}{1 + e^{nx}} \tag{1}$$

for $x \in \mathbb{R}$ with two parameters controlling its shape: the *degree of saturation* s and *neuronal gain* n[1]. This is a s-modulated convex sum of two $C^\infty(\mathbb{R})$ functions: the non-saturating softplus ($s = 0$), and the saturating sigmoid ($s = 1$), while n rescales the domain and controls response sharpness, or gain (Sompolinsky et al., 1988). Figure 1**a** shows the graph of γ for different values of (n, s), interpolating between well-known activation functions in deep learning. We note γ is differentiable in both s and n, and include these parameters in the optimization scheme in several experiments described below. We refer the reader to Appendix §A for error gradient derivations that include these parameters. Finally, the activation function can either be shared by all neurons (*homogeneous*) or vary neuron-to-neuron (*heterogeneous*). We incorporate this diversity by setting scalar (n, s) parameters in the homogeneous case, and by setting vectors $\boldsymbol{n}, \boldsymbol{s} \in \mathbb{R}^N$ in the heterogeneous case for N neurons, such that the activation function of neuron i is set by $\boldsymbol{n}^i, \boldsymbol{s}^i$. This flexible parametric family

Code available at: https://github.com/vgeadah/NonlinMod

[1]We often shorten these to *saturation* and *gain* and collectively refer to them as the *activation parameters*

allows to capture some key properties of f-I curve shapes present in different neuronal types. For instance, Type I neurons show gradual frequency increase with increasing applied current (i.e. γ with low n, $s \geq 0$), whereas Type II neurons show sharp firing onset at non-zero frequencies (i.e. approximated by γ with low n, $s > 0$). While there is no control of the AF threshold, the combined effect of recurrence and multiplicative scaling allows sufficient expressivity (Krishnamurthy et al., 2022).

In line with the goal of isolating the role of activation functions, we elect to use a simple "vanilla" RNN model for experiments. The vector equation for the recurrent unit activation $\boldsymbol{h}_t \in \mathbb{R}^{N_h}$ in response to input $\boldsymbol{x}_t \in \mathbb{R}^{N_x}$, $t \in \{0, \dots, T\}$ is given by

$$\boldsymbol{h}_t = \gamma(W_{hh}\boldsymbol{h}_{t-1} + W_{xh}\boldsymbol{x}_t + \boldsymbol{b}_h \,; \boldsymbol{n}, \boldsymbol{s}) \tag{2}$$

where the output $\boldsymbol{y}_t \in \mathbb{R}^{N_y}$ is generated by a linear readout $\boldsymbol{y}_t = W_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y$. Weight matrices $W_{(\cdot)}$ and biases $b_{(\cdot)}$ are optimized in all experiments. In the rest of this paper, we explore different ways in which learning and computations are influenced by the shape of γ, which operates point-wise on its inputs. To flesh out computational properties of activation function diversity, and of their use in real-time adaptation, we consider two main learning frameworks for analysis: (1) *static activation functions*, and (2) *adaptive activation functions* (see Figure 1**b** for a schematic).

The goal of the first *static* category of scenarios is to study the sole impact of activation shape on computational properties of RNNs. Specifically, we consider the activation function as a stationary property, not changing at inference or during input processing (Figure 1**b**). The activation function can either be imposed a priori and remain *fixed* throughout training, or be *optimized* by including the activation parameters tuple in the optimization process. We performed a grid hyperparameter search over the parameters $\{n, s\}$ in this fixed setting to set a prior for initialization (further details in Methods). We settled on $n_{\text{init}} \sim \mathcal{N}(5, 2^2)$ and $s_{\text{init}} = 0$, and further used this prior for the optimized but static setting, focusing on this setting for the rest of this work.

In the second optimization category, we investigate recurrent models with *adaptive* activation functions, allowing the activation parameters $\{n_t, s_t\}$ to vary during input processing (Figure 1**b,d**). Our goal is to allow as much flexibility for an adaptation strategy to emerge from end-to-end optimization. As such, we propose to use communicating standard RNN modules to control these activation parameters. We introduce the network of *Adaptive Recurrent Units* (ARUs) composed of parallel recurrent modules: a *main* RNN processes inputs, and parallel *adaptation controller* RNNs act locally to modulate the activation function of each neuron of the main RNN during input processing. See Figure 1**c** for a schematic representation of the architecture. The overall neuron-wise equations in response to input $x_t \in \mathbb{R}^{N_x}$ are given by:

$$a_t^i = W_{xh}^{i,:}x_t + W_{hh}^{i,:}h_{t-1} + b_h \tag{3}$$

$$g_t^{(i)} = \tanh\big(W_{ag}a_t^i + W_{gg}g_{t-1}^{(i)} + b_g\big) \tag{4}$$

$$n_t^i = W_{gn}g_t^{(i)} + b_n, \quad s_t^i = W_{gs}g_t^{(i)} + b_s \tag{5}$$

$$h_t^i = \gamma\left(a_t^i;\ n_t^i, s_t^i\right) \tag{6}$$

for each neuron $i = 1, \dots, N_h$ (superscripts denoting indices, subscripts denoting time). The weights $W_{\alpha\beta} \in \mathbb{R}^{N_\beta \times N_\alpha}$ and biases $b_\beta \in \mathbb{R}^{N_\beta}$ are updated via gradient descent using backpropagation through time (see Methods). Equations (3) and (6) define the main RNN with hidden-states $h_t \in \mathbb{R}^{N_h}$ similarly to (2), only now with time-varying shape signals $n_t,\ s_t \in \mathbb{R}^{N_h}$. The signals are dictated by the composition of (4) and (5), yielding the conceptual *adaptation mechanism* $\mathcal{A} : \boldsymbol{a}_t | \Theta_\mathcal{A} \mapsto \{\boldsymbol{n}_t, \boldsymbol{s}_t\}$, with weights $\Theta_\mathcal{A} = \{W_{ag}, W_{gg}, W_{gc}, b_g, b_c\}$ shared across all neurons. The adaptation mechanism maps the pre-activation a_t^i to a nonlinear activation function $\gamma(\cdot\,; n_t^i, s_t^i)$, akin to nonlinearity adaptation in cortical networks. Importantly, we construct $\mathcal{A}$ so that, given the neuron-dependent pre-activation a_t^i, it is independent of the specific neuron i.

2.2 Neural adaptation and diversity improves RNN performance and robustness to input perturbations

We use basic classification tasks to explore static and adaptive AF optimization, considering tasks complex enough for non-trivial solutions to emerge from end-to-end learning but simple enough to reduce confounders and augment interpretability. To this end, we conduct experiments on two synthetic tasks of sequential classification. In our numerical analysis of information propagation metrics during learning and associated emergent phenomena, we focus primarily on the task of classifying MNIST (Le et al., 2015) digits from a permuted sequential sequence of pixels (**psMNIST**). The second task, a grayscaled and sequential version of

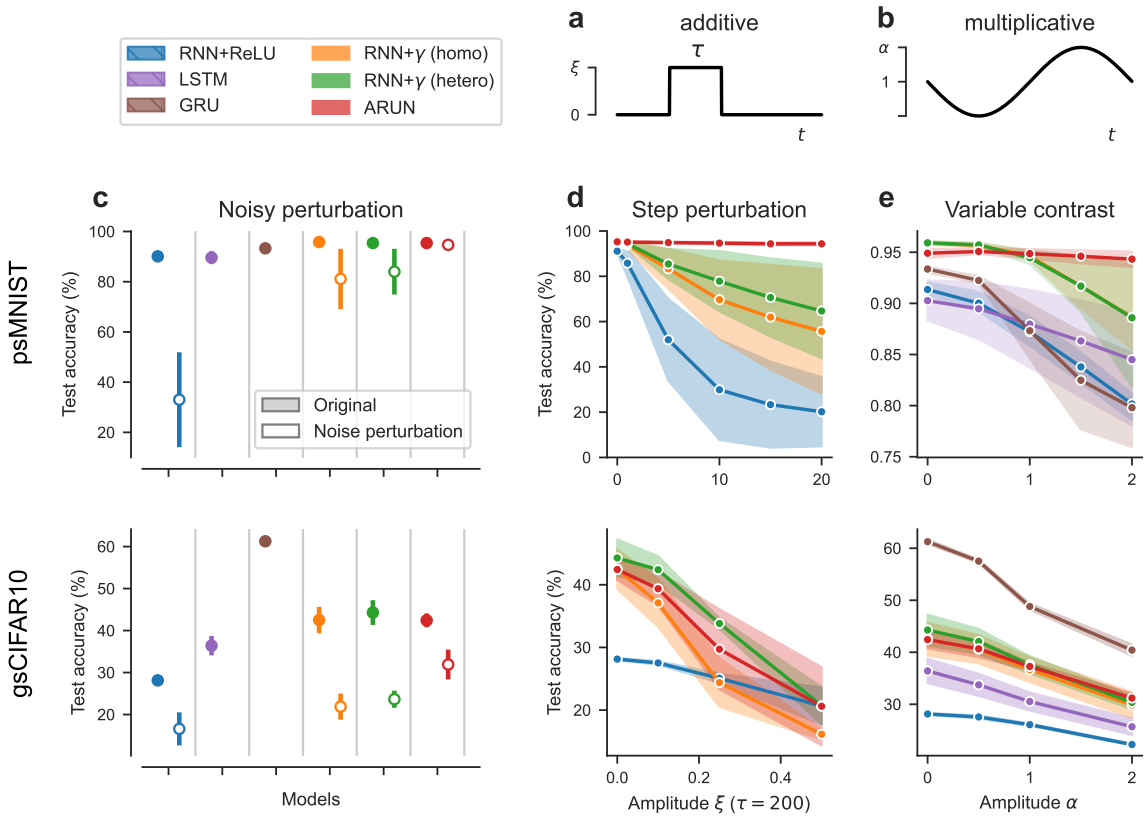

Figure 2: *Performance and robustness of RNN architectures on sequential classification tasks.* (**a, b**) Perturbations considered, either an additive step perturbation of amplitude $\xi > 0$ for τ time-steps, or a multiplicative change in contrast applied to the pixels inputs. (**c**–**e**) Results on the permuted sequential MNIST (top) and grayscaled sequential CIFAR10 (bottom) classification tasks. (**c**) Performance on the original unperturbed setting, and under an additive step noisy perturbation ($\xi_t \sim \mathcal{N}(5, 2^2)$ for psMNIST, $\xi_t \sim \mathcal{N}(0.5, 2^2)$ for gsCIFAR10) applied for $\tau = 200$ time-steps starting at $t = 200$. See Appendix Table 2 for the numerical values. (**d**) Sensitivity analysis under the additive step perturbation, for varying amplitude $\xi > 0$ for fixed duration $\tau = 200$ time-steps starting at $t = 200$. (**e**) Sensitivity analysis under variable contrast, for varying amplitude $\alpha > 0$ for fixed phase π.

the CIFAR10 classification task (**gsCIFAR10**), is further used as a more computationally demanding task to explore more complex regimes. It should be noted that performance on gsCIFAR10 is far from perfect, mostly due to the small network size we use. However this tasks serves the purpose of verifying that the emergent solutions at the single neuron level are consistent with the similar but more manageable task of psMNIST. See Methods for further details on both tasks. As baselines where appropriate, we focus on the RNN+ReLU network for comparison with a efficient but non-gated architecture (Glorot et al., 2011), and consider gating through the LSTM and GRU architectures. We chose the later as they are known to be more efficient that RNNs at learning long time-dependencies and are robust to transient perturbations, however we note that they rely on non-local, biologically unrealistic gating mechanisms.

Learning diverse, static activation functions We start by considering the static setting as a stepping stone towards AF adaptation on transient timescales. First, we find that the introduction of homogeneous $\gamma(\cdot; n, s)$ AF learning provides a considerable increase in performance compared to baselines. On the psMNIST task, RNN with γ outperform both ReLU and gated (LSTM, GRU) baselines. Moreover, learned combinations of (n, s) activation parameters differ from conventional nonlinearities, converging to the unit-norm manifold $\{(n, s) : \|\gamma'(x; n, s)\| = 1\}$ (see Appendix B.1). Similar results are obtained on the gsCIFAR10 task. While the GRU offers the highest performance by far, homogeneous optimization of RNN+γ achieves greater classification accuracy and provides a significant improvement over the RNN+ReLU (see Figure 2**c**). Then we turned to heterogeneous optimization, providing a more in-depth portrait of AF modulation in RNNs, in

a manner closer to the diversity of activations displayed in cortical networks. In terms of performance for the psMNIST task, we found that learning heterogeneous activation did not provide a significant advantages over the already well performing optimization settings, outperforming the fixed setting but not necessarily the homogeneous setting. On the gsCIFAR10 task the same conclusions hold, the heterogeneous RNN+γ performs as well if not slightly better than the homogeneous RNN+γ (see Figure 2), however the increase in performance is again not statistically significant.

Adaptive units improve performance and robustness to changes in environment Perturbations of both neuron- and network-level inputs are an integral part of the literature on neural adaptation. We introduced in §2.1 the network of ARUs (ARUN), a general RNN-type model that offers modulation of the AF on a transient scale, during online input processing. How do these adaptive AFs compare in terms of performance, and upon facing these perturbations? We consider two classes of perturbations. First, we draw inspiration from optogenetic stimulation and inject a scalar external drive $\xi > 0$ directly to the neuron before the AF is applied during τ time-steps (Fig. 2**d**). Second, we transform the network-level inputs x_t by applying a sinusoidal change in contrast (Fig. 2**e**), thus altering the input statistics directly. More details in Methods §5.4. We report our results on the psMNIST and gsCIFAR10 tasks with these variations in Figure 2.

We observe that networks with adaptive nonlinearities exceed or equl performance on the initial tasks (within error bars). Furthermore, their real advantage emerge when considering their ability to mitigate the changes in input statistics in psMNIST from the transformed digits or in response to an added stimulus. In psMNIST, the ARUN outperforms other architectures on the noisy drive (Fig. 2**c**), as such offering the lowest difference between the original setting and this step perturbation. Furthermore, it shows high robustness to variable drive amplitude (Fig. 2**e**). Performance experiments on the gsCIFAR10 task are less conclusive, as expected, attributable in part to the overall lower performance for all networks. We can see however that the ARUN is significantly more robust to the noisy perturbation when compared to the other RNNs or LSTMs (Fig. 2**c**). ARUs also seem to be more robust in the extreme cases of step perturbations or variable contrast since it retains more of its performance in both experiments at high amplitudes.

On top of the results presented in Figure 2, we conducted a sensitivity analysis with respect to the various parameters of the transformations (see Appendix Figure 12). First, we varied the phase and amplitude of the sinusoidal transformation applied on inputs, and we observe that the ARUN presents the best robustness. Second, we varied the amplitude and length of the step-drive applied on neurons. In this driven case, the ARUN presents a test loss of an order of magnitude lower than the other RNN models while varying the parameters. In all, endowing networks with adaptive nonlinearities present significant advantages in mitigating changes in input statistics.

2.3 Top-down optimization of adaptive RNNs recovers biological dynamic coding mechanisms of single neurons

When trained on temporal perception tasks (sequential MNIST/CIFAR10), we demonstrated that our network of ARUs shows improved robustness to noise and changes in input statistics. Remarkably, we find that in doing so, ARUs implement precise SF mechanisms from biological neurons, including gain scaling and specifically fractional input differentiation (Laughlin, 1981; Fairhall et al., 2001b; Lundstrom et al., 2008). This suggests that even in simplified models, environmental pressures and objective-based optimization are enough for sophisticated single neuron mechanisms to emerge. Below, we investigate these mechanisms arising from goal-oriented optimization, and find that they implement a number of observed properties of biological neurons. We reiterate that our system does not have any constraints apart from the AF parametrization, and that it could have, in principle, chosen any adaptive of AF strategy, biologically realistic or not.

Heterogeneity Heterogeneous activation functions already provide a setting more reminiscent of the diversity of activations in cortical networks. In Section §2.2 we demonstrated that heterogeneity was beneficial to task performance and robustness (albeit not significantly). Furthermore, we observe that when the activation function γ is initialized homogeneously, the optimization procedure leads to heterogeneity in the activation functions across the network (Fig.3**a** top). See Winston et al. (2022a) for similar results when AFs are parametrized following known relations between ionic currents and f-I curves. Further experiments (details included in Appendix §B.2) consider trained RNN+γ networks reading psMNIST digits rotated by $\pi/4$ rad. As opposed to the perturbation experiments highlighted previously, this also changes the temporal order in which the inputs are fed. In this new setting, we observed an increase in $\{n, s\}$ heterogeneity upon changes in task temporal statistics (Fig.3**a** bottom), when all other parameters are kept fixed. Simply allowing the AFs to

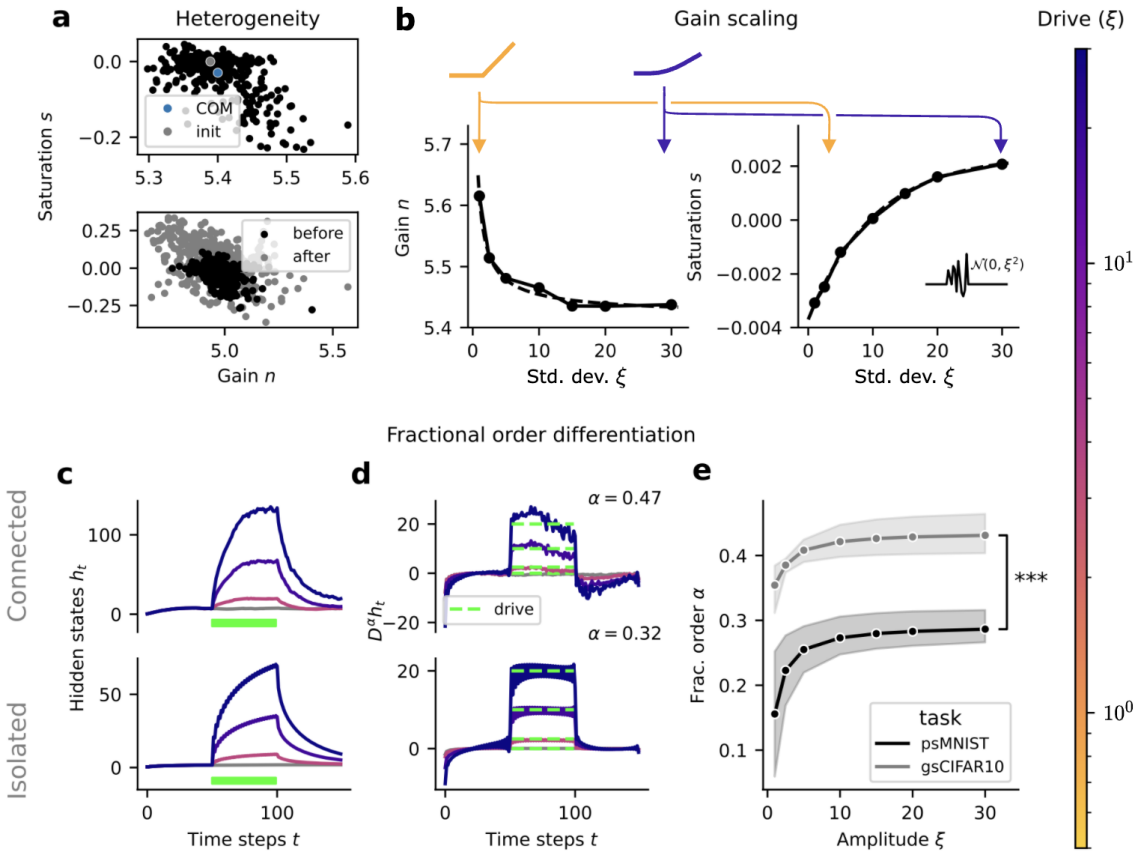

Figure 3: *Biological dynamic coding mechanisms recovered by ARUs*. Colorbar applies to whole Figure. **(a)** Neuron-to-neuron heterogeneity in AFs. (Top) Learned activation parameterss in RNN+γ (het.), from homogeneous initialization. (Bottom) Increase in heterogeneity before and after a 45° rotation is applied to the psMNIST digit, learned through backpropagation. **(b)** As a response to a noisy external drive of varying variance, the gain of the ARUs displays power-like decay (fit in dashed) as a function of the std.-dev. ξ, and the saturation displays exponential ramp-up (fit in dashed). Averaged over $N = 3$ seeds. Depictions of the associated γ AFs are included above, exagerated for visualisation purposes (color follows colorbar). **(c,d)** Experiments on original fully connected recurrent dynamics (top), or with neurons in isolation (diagonal recurrent weight matrix) (bottom). Gray indicates undriven activity. **(c)** Mean hidden-states h_t under varying step external drive ξ, applied during the green bar period. **(d)** Mean fractional-order-differentiated hidden-states $D^\alpha h_t$, of order α under varying step external drive ξ. Dashed green step-constant lines indicate the original additive step-perturbation applied to the networks. **(e)** Fractional integration order established by minimizing the MSE between the fractional order differentiated signal of isolated ARUs activity (**d** bottom) and the original step drive (**d** bottom dashed green), as a function of the drive and the task.

be modulated could recover over a quarter of lost performance (over 25%) in this altered task. These results show that heterogeneity is both beneficial and is learned through optimization.

Adaptation implements gain scaling As a first indication of ARUs implementing optimal coding mechanisms akin to biological neurons, we find the general gain adaptation behavior following general gain scaling principles of cortical neurons (Laughlin, 1981; Fairhall et al., 2001b). We subjected ARUs to a low-noise signal $\mathcal{N}(0, 0.01^2)$ for $t = 200$ time steps, followed by a i.i.d samples from $\mathcal{N}(0, \xi^2)$ for varying $xi > 0$ during another $t = 200$ time steps, before returning to the original low noise (see inlet Fig. 3**b**). We observe the gain $n_\infty(\xi)$ to display a power-law dependence on ξ (Fig. 3 left) (Lundstrom et al., 2008). As for the saturation, we observed a exponential dependence on ξ (Fig. 3 right). These have the combined effect of following Laughlin's original assertion (Laughlin, 1981); we found ARUs to allocate their effective AF range

proportionally to the input statistics, thereby mitigating variations in the output distribution. Further details about the role of this mean response value on the stability of network dynamics are presented in Section 2.4.

Adaptation implements fractional differentiation We find that ARUs learn to produce dynamics that implement fractional differentiation of input signals, a fundamental mechanism for efficient information processing in cortical neurons intimately related to gain scaling (Lundstrom et al., 2008; Weber et al., 2019). Fractional order differentiation can be understood by a filtering operation in Fourier domain $H(f) = (2i\pi f)^{\alpha}$, where $\alpha \in [-1, 1]$ is the order of fractional differentiation. Conceptually, differentiation pertains to $\alpha > 0$, and a negative order α corresponds to the inverse process of fractional order *integration*. Both are mathematically justified, observed in retinal neural populations (Kastner and Baccus, 2011, 2013), and carry meaning as a convolutional filter mechanism.

In response to a step-constant drive, we observe ARU activity ramping up to a different regime during stimulation, followed by a return to a regime almost identical to un-stimulated dynamics (Fig. 3**a**)—an expected mechanism coherent with firing rate dynamics under optogenetic stimulation (O'Shea et al., 2018). Turning to the activation parameters, we observe prototypical onset $\{n_0, s_0\}$ with exponential decay to steady-state values in the signals (Fig. 3**b**). In doing so, the controller network effectively implements fractional order filtering. Indeed, fractional order integration of these latter signals reveals step-linear-increase signals (see Appendix Fig. 8).

To further elucidate this finding, we consider the same step-constant drives, now applied to single neurons in isolation, with trained weights. This is a setting conceptually closer to the original experiment by Lundstrom et al. (2008), performed on slices of neocortical pyramidal neurons. Population averaged hidden states of these non-interacting ARUs show inversely exponential ramp up from steady state (Fig. 3**c**), similar for varying amplitudes (ξ) of step perturbation. Fractional order differentiation of this signal reveals that we can recover nearly exactly the original step perturbation (Fig. 3**d**). We determine the fractional order precisely by finding the order that minimizes the mean square error (MSE) between the fractionally differentiated signal and the original step perturbation (Fig. 9). We find this minimum to be sharp, and generally, this methodology yielded persistent results on all three random seeds and both tasks (see examples in Fig. 9-11). This indicates that the ARU controller effectively employs the precise fractional differentiation filter in Fourier domain observed in cortical neurons (Weber et al., 2019), with specific order values akin to sensitizing retina cells (Kastner and Baccus, 2011).

Taking a closer look at the exact fractional orders, we find that they depend on the task considered. See Fig. 3**e**. For all ξ tested, we found the average fractional orders α to be statistically different between the two tasks ($p < 0.001$ for $\xi \geq 1$, $N = 5$ seeds, independent two-sample t-test). Furthermore, we did find the fractional order to depend ξ ($p < 0.05$, $N = 5$, related two-sample t-test between $\xi = 1$ and $\xi = 30$), increasing for low values to a plateau for higher values. Given that the same random seeds were used, the ARUN networks were initialized to the same parameters, and thus this difference in order of fractional integration stems from the tasks' input statistics only.

2.4 Neural adaptation as a local regularizer that improves global network information propagation

So far, we have established that diversity and adaptive tuning of single neuron AFs improve an RNN's performance on perceptual tasks, and considerably improves robustness to noise. The level of improvements is on par with advantages afforded by gating architectures (e.g. LSTM, GRU), which are not biologically realistic. Furthermore, we showed that optimized adaptive dynamics are not only biologically more plausible, as they are implemented locally at each neuron, but they also implement dynamic coding mechanisms observed in real neurons (gain scaling, fractional differentiation). In this section, we analyze the solution implemented by the ARU controller network, and reveal the mechanisms that led to their optimization during gradient descent, and the advantages they afford.

Adaptive activation reduces noise variance in single neurons To shed some light onto the mechanisms observed, we now quantify the impact of the activation function γ on noise integration. As a response to a general perturbation scalar $\eta \sim \mathcal{N}(\mu, \sigma^2)$, linearization of the hidden-states dynamics about the perturbation means yields better understanding of the role of the parameters $\{n, s\}$ in amplifying or reducing this noise.

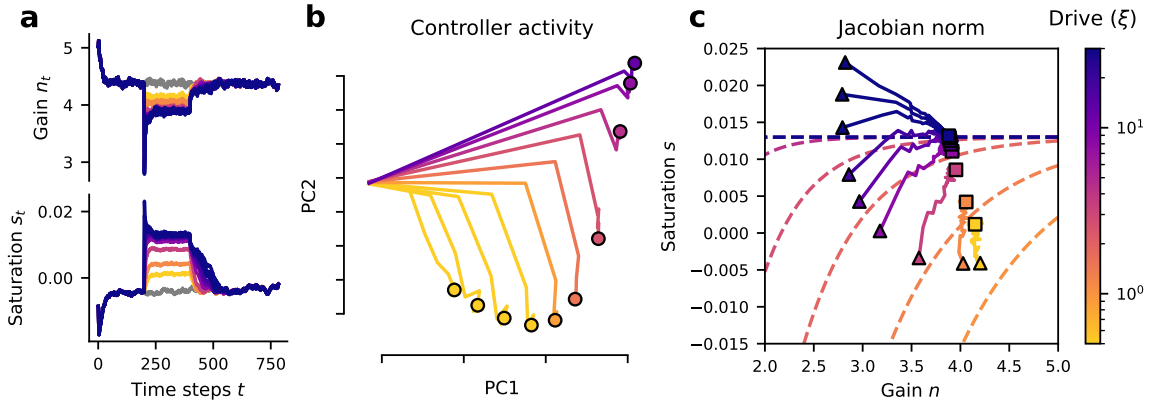

Figure 4: *Controller activity.* The gray color is used throughout to indicate the original perturbation-less setting. All networks are taken with trained parameters, on the psMNIST task. **(a)** Under drives of varying amplitude ξ, the gain (top) and saturation (bottom) signals during the processing of a input digit. **(b)** Limit points in controller activity from constant input ξ. **(c)** $\{n_t, s_t\}$ trajectories during stimulation, with onset (triangle) and steady state (square), for each external drive ξ. Overlayed over jacobian $\gamma'(\xi; n, s) = 1 - \epsilon$ level curves in phase space ($\epsilon = 0.01$).

Proposition 1. *For unitary W_{hh} weight initialization, the variance explained along a vector $u \in \mathbb{R}^{N_h}$ as a response to a perturbation $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 I)$ decays if and only if the parameters $\{\boldsymbol{n}_t, \boldsymbol{s}_t\}$ satisfy*

$$\sigma^2 \left[\frac{d}{dx} \gamma \left(\mu^i \, ; n_t^i, s_t^i \right) \right]^2 < 1 + \mathcal{O}\left(\sigma^3\right) \tag{7}$$

for $i \in \{1, \ldots, N_h\}$.

Proof. See proof in Supplementary Materials. (Appendix §C.2.1) □

Hence, noise robustness is equivalent to setting the LHS in (7) smaller than one so that noise is non-amplified by the dynamics. For example: in a linear setting with a linearity of slope a, the above condition reads $\sigma \leq 1/a$. This proposition reformulates known conditions on the jacobian of the dynamics in a matter especially suited for our experimental settings. For conciseness, consider a specific neuron i, and drop superscripts. By the implicit function theorem, setting in Proposition 1 an equality of the form $\sigma^2 \left[\frac{d}{dx} \gamma \left(\xi \, ; n_t, s_t \right) \right]^2 = 1 - \epsilon$ defines a manifold in $\{n, s\}$-parameter space for any $\epsilon > 0$. This manifold is a function of the amplitude ξ of the drive η. Given an initial condition $\{n_0, s_0\}$, one can solve the above system to obtain a path $\{n(\xi), s(\xi)\}$ in parameter space as a function of ξ (assuming continuous dependence on ξ, see Appendix §C for further details). This path corresponds to the expected variation in activation parameters $\{n, s\}$ as a function of ξ to absorb, through the hidden-state dynamics within a linearization about the mean, the injected noise by a distance ϵ.

We observe that in the absence of an external perturbation, the un-perturbed shape signals $\{n_t, s_t\}$ in Fig. 4**a** show transient behavior before settling into a stable value $\{n_\infty, s_\infty\}$. As a stimulation ξ is injected into the system, we observe an onset value $\{n_0(\xi), s_0(\xi)\}$ decreasing (or increasing) with an exponential time-constant to a steady-state value $\{n_\infty(\xi), s_\infty(\xi)\}$. Note that as the activation parameters $\{n_t^i, s_t^i\}$ are an affine transformation of the controller hidden-states $g_t^{(i)}$ at each time-step, this behavior indicates similar controller RNN dynamics. We found the controller RNN to implement varying fixed points of activity as a response to constant input drive to the main RNN (Fig. 4**b**). This shape parameter control ties back to the gain scaling and fractional order differentiation results presented in §2.3.

These values during the stimulation period are precisely the subject of Proposition 1. As observed in Fig. 4**c**, we find that the learned $\{n_t, s_t\}$ trajectories as a function of varying external drive ξ support the expected behavior detailed above. We observe the onset values (triangle) falling in the region of decaying Jacobian norm, with the steady-state values (square) approaching its boundary from within. This indicates that the activation parameters are adapted such that the linearized dynamics enforce the decay of external noise. This mechanism prescribes the precise value of the steady-state shift of AF parameters due to adaptation, which

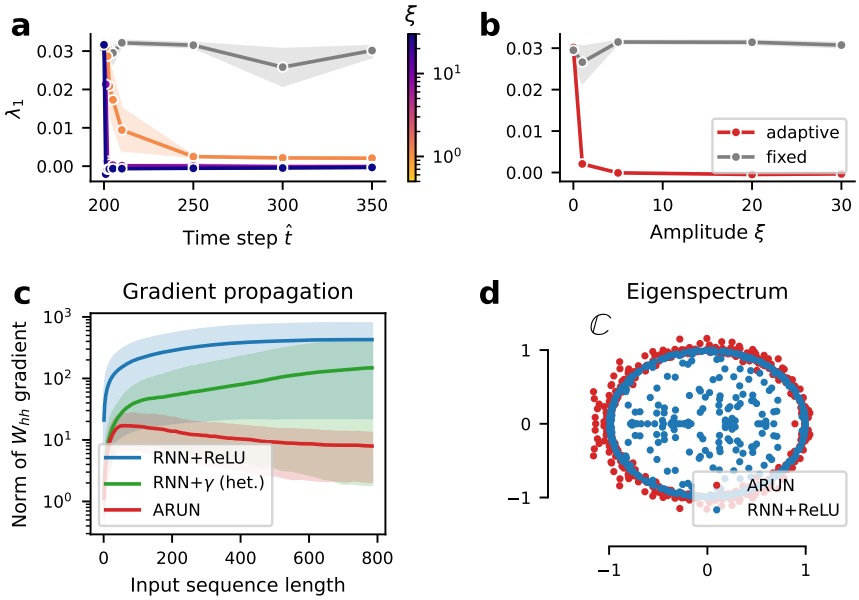

Figure 5: *Dynamic regularization by ARU controllers.* **(a-b)** Maximum Lyapunov Exponent (λ_1) of an ARUN's *main* RNN, under varying settings, averaged over $n = 10$ draws from the used stationary distribution (mean and 95% c.i. shown). (**a**) λ_1 as a function of the chosen time-step $\hat{t}$ for varying ξ. Here, we take $\{n_{\hat{t}}(\xi), s_{\hat{t}}(\xi)\}$ set by an ARU and treat them as unvarying in time (equivalent to RNN+γ het.). (**b**) λ_1 as a function of the step-drive amplitude ξ, with letting the activation parameters $\{n_\infty(\xi), s_\infty(\xi)\}$ be "adapted", or remain "fixed". In a perfect ergodic system, the fixed line would be horizontal. **(c)** Frobenius norm of the hidden-to-hidden weight matrix gradients as a function of input sequence length (mean and 95% c.i. shown, $n = 3$ seeds). For sequences of length n, the last n pixels of the original input sequences are used. **(d)** Eigenspectra of main RNN connectivity matrices of trained ARUN and RNN+ReLU networks on psMNIST. Both networks were initialized with Henaff orthogonal matrices, whose eigenvalues lie on the complex plane's unit circle.

corresponds to the gain-scaling operation described in Section 2.3. The theory prediction is not perfect for low input amplitudes as the main network can compensate by itself, but our theoretical prediction improves for high input ranges, where neural adaptation is most needed. Now, to provide a more thorough picture of the compounded nonlinear effect of the dynamics, we turn our attention to the metric of Lyapunov exponents.

Adaptation regularizes network dynamics at the edge of chaos We find that our optimized adaptation implements a form of dynamic regularization that aims to keep global network dynamics in an optimal regime. To better quantify this, we leverage Lyapunov Exponents, a measurement of average expansion and contraction of state space by a dynamical system, as well as notions of gradient propagation, a proxy for information transfer quality.

We consider the maximum Lyapunov exponent (λ_1) of the autonomous dynamics of the main network in trained ARUNs (see Appendix D for a primer on the topic). The λ_1 computation relies on first drawing an initial condition $h_0 \sim \pi$, and then calculating forward steps based on (3-6) with parameters $\{W_{hh}, n, s\}$ (see Vogt et al. (2022) for details). In all cases, we take $\pi(\xi)$ to be the stationary distribution of the ARUs during a drive of amplitude ξ (see further details in Methods §5.4). We find this distribution to be Gaussian, of mean and standard deviation depending linearly on ξ (see Appendix C). We consider both the *adaptive* case with activation parameters $\{n_\infty(\xi), s_\infty(\xi)\}$ actively set by the ARUs, and with activation parameters $\{n_\infty(0), s_\infty(0)\}$ taken from the undriven scenario and thus not adapted (*fixed*). We find that for constant drives of distinct amplitude $\xi > 0$, the activation parameters are effectively adapted to push the networks to have a λ_1 several orders of magnitude closer to 0 in comparison to non-adaptive RNNs (see Fig. 4**b**. In other words adaptation regularizes the main network dynamics, otherwise chaotic, to be at the edge of chaos under external drive. This process is typically achieved by globally modulating connectivity strength, but is implemented here thanks to a local mechanism at each neuron.

Adaptation promotes good gradient propagation The well documented vanishing and exploding gradients problems of RNNs prohibit effective training over long timescales. In particular, the gradient of the loss computed with the output of the network at time-step $t + \delta$ with respect to the hidden states of the network at time-step t either decays or increases exponentially with δ. In the vanishing case this makes the learning of long term dependencies impossible, while in the exploding case the entire training procedure is compromised. This phenomenon is linked to dynamic regimes in which an RNN operates, and is thus related to the leading Lyapunov exponent measurement described above (see Vogt et al. (2022); Poole et al. (2016) for more details).

We quantify the effects of learned neural adaptation on gradient propagation in RNNs by computing the gradient norms of the hidden-to-hidden weight matrix (W_{hh} in equations 2 and 3) on the psMNIST training set starting the gradient accumulation at different points in the input sequence. This was done for trained networks to take into consideration the learned adaptive behavior when considering the gradient propagation. In RNN+ReLU networks and to some lesser extend in RNN+γ heterogeneous networks the gradient norm increases monotonously with sequence length, as shown in logarithmic scale in Fig. 5**c**. The earlier the accumulation of the gradients is started for these two network types the larger their norm is at the end of the input sequence when the loss is computed. In ARU networks however, after an initial increase the norms of the gradients actually decrease with sequence length. When the gradients are computed using the entire input sequences, the norm of the W_{hh} gradients in ARU networks is an order of magnitude smaller than in RNN+ReLU or in RNN+γ heterogeneous networks which promotes trainability and the stability of the gradient propagation during the training procedure. We also note that in ARU networks, elements (pixels) which are at the beginning of the input sequence and further away from the moment the loss is computed actually contribute more to the gradient of the weights when compared to later inputs. This is in stark contrast with gradient contribution in RNN+ReLU networks where the gradient contribution is monotonously increasing with the element's position in the input sequence, early inputs contributing much less than later inputs to the gradient. See Appendix C.4 for more details on the gradient contribution results.

3 Discussion

Optimal information propagation

Recurrent neural networks, whether biological or artificial, must balance *expressivity* and *stability* to implement complex computations. A network whose dynamics quickly converge to a fixed point, for example, is quite stable but not expressive since very little mathematical transformations of inputs can take place (e.g. integration, amplification). In contrast, a network operating in a rich, chaotic regime for example, is expressive but unstable: its dynamics implement very rich computations but tiny perturbations lead to widely contrasting outcomes. The balance between these two requirements has been identified in several contexts, and is often referred to as the "edge of chaos" (Bertschinger and Natschläger, 2004). Dynamics close to the transition point between stable and chaotic dynamics offer optimal information propagation over long timescales, as well as rich transformations (Bertschinger and Natschläger, 2004; Legenstein and Maass, 2007; Boedecker et al., 2011). This regime has also been shown to be important in deep and recurrent artificial neural networks (Poole et al., 2016). Indeed, a rich theory of how large networks learn and implement computations shows that expressivity is maximized when dynamics are close to chaotic. This finding is closely linked to the backpropagation of errors in gradient descent optimization.

Several strategies have been developed to ensure efficient training of artificial neural networks. Much of these strategies rely on global knowledge of networks (Pascanu et al., 2012), and global interventions on connectivity (Arjovsky et al., 2016; Le et al., 2015; Henaff et al., 2016; Lezcano-Casado and Martínez-Rubio, 2019; Kerg et al., 2019). For instance during training, batch normalization has proven incredibly efficient at enforcing certain dynamical regimes. However, this process is inherently non-local, requiring network-level knowledge. As for online input processing, gating has been the predominant view in forward-processing recurrent neural networks. Recent work by Krishnamurthy et al. (2022) appeals to the question of multiplicative gating in RNNs, and its impact on input-driven shifts in dynamical behavior. Here, in subsections §2.3 and §2.4, we report the learned adaptation mechanisms of trained ARUs. We observe that the ARUs' controller RNN implement denoising via fractional order differentiation, and that the input-modulated steady states $\{n_\infty(\xi), s_\infty(\xi)\}$ effectively bring the main RNN dynamics to the edge-of-chaos. In doing so, we propose that ARU controller implements a (conditionally purely) local, more biologically plausible form of dynamic regularization.

Importantly, this adaptive strategy offers additional advantages: it allows connectivity matrices to efficiently implement transformations away from orthogonality, normally leading to chaotic regimes. In a standard RNN, connectivity matrices with eigenvalues whose magnitude lies beyond the complex plane unit circle typically

lead to chaotic dynamics. Such dynamics are computationally rich, but their inherent instabilities hinder information propagation over long timescales as well as network trainability. Therefore, it is typical to see optimized RNNs with connectivity matrix eigenspectra lying on the unit circle. In contrast, ARUs allow optimized networks to have connectivity spectra well beyond the unit circle (Fig. 5**d**), thus allowing more expressivity. In addition, in an externally driven setting for which the network was not directly trained, we found networks to be dynamically stabilized by adaptive neurons. this suggests that adaptation at the single neuron level enables added expressivity while maintaining stability across a range of dynamic regimes.

Deep learning and backpropagation as a framework to uncover biologically realistic optimal solutions

It is unlikely that the brain makes use of backpropagation, as it is generally implemented for artificial network optimization. Nevertheless, we argue that the solutions found by back-propagation in RNNs are likely consistent, in some key regards, with those found by whatever learning and evolutionary mechanism that unfold in the brain. More precisely, the requirement of stable and expressive information propagation is shared between artificial and biological networks. For artificial RNNs subject to backpropagation, this manifests in parameter configuration incentives that help gradients propagate further back in time. Yet, backward gradient propagation and forward information propagation are two sides of the same coin. Indeed in the process of backpropagation through time (BPTT), a recurrent neural network's ability to accomplish a task dependents entirely on its ability to efficiently propagate information back through the multiple internal transformations. As highlighted by Pascanu et al. (2012), recurrent neural networks are subject to the vanishing and exploding gradient problem. The Jacobian of the recurrent dynamics is a leading culprit to this behavior, and our investigation of the linearized dynamics in §2.4 specifically translates its dependence on the activation parameters $\{n, s\}$. We found the Jacobian to provide significant insights into the integration of external noise, and more generally metrics on nonlinear composition such as the Maximum Lyapunov Exponent demonstrated the intricate "edge of chaos" regime neared by the optimal solutions. This is exemplified by gradient propagation improvements afforded by ARUs and illustrated in Fig. 5**c**.

Hence while BPTT is regarded as not itself biologically plausible, its requirements on the backward gradient propagation translate into the same requirements for forward stability in networks (Bertschinger and Natschläger, 2004). We further find that indeed, optimal solutions under BPTT have the same characteristics as optimal solutions from a more biologically realistic optimization process (§2.3). Taken together, this highlights how BPTT is a powerful tool for goal-driven investigation in an artificial setting of biological dynamic processes. Not because of its process, which is biologically implausible, but because it operates under the same optimization constraints as biological networks, and finds solutions that are consistent with these requirements.

Therefore, this work focuses on the characterization of optimal solutions, and not on the modeling of learning itself. This is shared approach that yields impressive results across varied settings. Notably, Winston et al. (2022b) similarly optimize static neuron properties in RNNs and find advantages of diversity, and the line of work spawned by Yamins and DiCarlo (2016) keeps reveling stimulus tuning properties of neurons in the visual pathway and beyond.

Finally, we note that we did not focus on the issue of learning timescales. In comparison to neuron dynamics and neural circuits in the brain, ARU controller would have been optimized over evolutionary timescales, while the main RNN parameters, representing synaptic connections, over the lifespan of an animal. We did try a limited number of experiments, for example by fixing one while learning the other and vice versa (see Appendix §C.3), and did not see any significant differences in results. A different methodology, borrowing from deep learning frameworks like meta-learning, could allow for a more adequate consideration of adaptive mechanisms as a product of evolution-like pressures. Such a more thorough investigation of the impact of learning timescales on solutions is outside of the scope of this paper, but is a fascinating direction of future work to disentangle evolution and learning pressures.

4 Conclusion

In this work, we sought to investigate goal-driven learning pressures from the system-level onto dynamic coding mechanisms at the single-neuron level. We do so by introducing *adaptive recurrent units*, allowing for only AF control from a novel parametric family. Our main findings are threefold: (1) Diverse and adaptive activation functions considerably improve computational performance of networks while also helping mitigate previously unobserved changes in input statistics during a task, thus improving out-of-distribution generalization. (2) System-level learning pressures drive biologically plausible adaptation strategies, namely

activation function having biologically realistic configurations and more importantly, the implementation of biological SFA mechanisms such as gain scaling and fractional differentiation. (3) Finally we find that adaptation acts as a dynamic regularizer allowing recurrent neural networks to remain in a dynamic regime closer to the edge of chaos where forward and backward information propagation is optimal. These findings are supported by detailed numerical experiments and analytically derived bounds for information propagation in networks. We discuss how ARU adaptation can effectively implement a number of methods often used in deep learning to ensure good network expressivity and stability, including regularization and normalization. In contrast to these methods which require global, biologically unrealistic network quantities, ARU adaptation is local to each neuron and is consistent with known physiology. Taken together, our results support that neural diversity and adaptation serves a crucial role in goal-oriented network optimization, which suggests a coordinated and consistent optimality across scales linking brain circuits and single neurons.

Acknowledgments

We are grateful for scholarship support from NSERC [V.G., S.H]; FRQNT [V.G., S.H]; IVADO [V.G., G.W., S.H.]; and UNIQUE [G.K.], and well as support from NIH grant R01GM135929 [G.W.]; NSERC Discovery Grant (RGPIN-2018-04821), FRQNT Young Investigator Startup Program (2019- NC-253251), FRQS Research Scholar Award, Junior 1 (LAJGU0401-253188) [G.L.]; and the Canada CIFAR AI Chair Program [G.W., G.L.]. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies

5 Methods

5.1 Tasks

psMNIST : This task focusses on classifying MNIST (Le et al., 2015) digits from a permuted sequential sequence of pixels (**psMNIST**) and requires an accumulation of information over long timescales. More precisely, a fixed random permutation is applied to pixels of the popular hand-written digits MNIST dataset, and the model reads them sequentially. Correct digit class needs to be outputed at the end.

gsCIFAR10 : Grayscaled and sequential version of the CIFAR10 image classification task (Krizhevsky et al., 2009). The network is shown images of real world objects one pixel at the time and has to determine to which one of the 10 classes the image belongs. Because the images are of objects in a variety of settings and not simply of digits, this constitutes a significantly harder task than psMNIST.

5.2 Task setup and training

The vector of all trainable parameters is denoted Θ, and the parameters are updated via gradient descent using backpropagation through time (BPTT), with the matrix W_{rec} initialized using a random orthogonal scheme (Henaff et al., 2016). Independently of the task, we used Cross-entropy loss as our loss function and the Adam (Kingma and Ba, 2015) optimizer. We experimented with the RMSprop optimizer (introduced in Hinton et al. (2012), first used in Graves (2013)) with smoothing constant $\alpha = 0.99$ and no weight decay, which yielded similar results. (more details on initialization schemes an be found in Appendix §A). We trained the networks for 100 epochs. We investigated different learning rates (LR $\in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$), and settled on those in the Table 1 for each task.

activation parameters The initialization grid for the activation parameters used throughout this work is $N \times S$, where $N = \{1.0\} \cup \{1.25k \ : \ 1 \leq k \leq 16\}$ and $S = \{0.0,\ 0.25,\ 0.5,\ 0.75,\ 1.0\}$ such that $|N| = 17$ and $|S| = 5$. In the heterogeneous adaptation scenario, both $\boldsymbol{n}$ and $\boldsymbol{s}$ vectors are initialized with the same value for each component.

Pytorch autograd implementation of gamma We implement $\gamma(x; n, s)$ as a Pytorch autograd Function with corresponding Pytorch Module. See zipped supplementary code `gamma_function.py`.

To allow for activation function adaptation, we further include the activation parameters in the backpropagation algorithm. We do so by defining the gradient of γ with respect to the input and parameters. We can rewrite $\gamma(x; n, s)$ as :

$$\gamma(x; n, s) = \frac{(1-s)}{n}\gamma_1(nx) + s\sigma(nx) \tag{8}$$

where $\sigma(x)$ is the sigmoid activation function. With this notation, the partial derivatives of γ with respect to x (or total derivative), n and s are

$$\gamma'(x; n, s) = \frac{\partial}{\partial x}\gamma(x; n, s) = (1-s)\sigma(nx) + ns\sigma(nx)(1-\sigma(nx)) \tag{9}$$

$$\frac{\partial}{\partial n}\gamma(x; n, s) = \frac{1-s}{n}(x\sigma(nx) - \frac{\gamma_1(nx)}{n}) + sx\sigma(nx)(1-\sigma(nx)) \tag{10}$$

$$\frac{\partial}{\partial s}\gamma(x; n, s) = \sigma(nx) - \frac{\gamma_1(nx)}{n} \tag{11}$$

5.3 Evaluation methods

To assess how the activation gain and saturation influence on signal propagation, we use three quantities:

Jacobian norm The main mechanism leading to the well studied problem of exploding & vanishing gradients in backpropagation and BPTT happens when products of Jacobian matrices explode or vanish (Pascanu et al., 2012; Bengio et al., 1994). We average the L^2-norm of the derivative of Eq. (2) with respect to $\boldsymbol{h_{t-1}} \sim \mathcal{U}(-5, 5)$. A mean Jacobian norm (JN) that is greater/less than one leads to exploding/vanishing gradients, respectively. An issue with this approximation is that the true mean depends on $\boldsymbol{h_t}$'s invariant distribution, which changes with (n, s).

Lyapunov Exponents Developed in Dynamical Systems theory, Lyapunov exponents measure the exponential rate of expansion/contraction of state space along iterates. Let us define $F : X \to X$ to be a continuously differentiable function, and consider the discrete dynamical system (F, X, T) defined by

$$x_{t+1} = F(x_t) \tag{12}$$

for all $t \in T$, where X is the phase space, and T the time range. Let $x_0, w \in X$, define

$$\lambda(x_0, w) \stackrel{\text{def}}{=} \lim_{m\to\infty} \frac{1}{m} \ln \prod_{t=1}^{m} \frac{\|DF^t(x_0) \cdot w\|}{\|w\|} \tag{13}$$

$$= \lim_{m\to\infty} \frac{1}{m} \sum_{t=1}^{m} \ln \frac{\|DF^t(x_0) \cdot w\|}{\|w\|} \tag{14}$$

Note that once x_0 and w have been fixed, the quantity $\lambda(x_0, w)$ is intrinsic to the discrete dynamical system defined by $x_{t+1} = F(x_t)$. We call $\lambda(x_0, w)$ a **Lyapunov exponent** of the mentioned dynamical system. Intuitively, Lyapunov exponents are topological quantities intrinsic to the dynamical system that describe the average amount of instability along infinite time horizons. Now, a randomly chosen vector, has a non-zero projection in the direction of the Maximal Lyapunov exponent (MLE) with probability 1, and thus over time the effect of the other Lyapunov exponents will become negligible. This motivates taking the MLE as a way of measuring the overall amount of stability or instability of a dynamical system. (see Appendix §D for a LE primer). As an asymptotic quantity, the MLE has been used to quantify ANN stability and expressivity (Pennington et al., 2018; Poole et al., 2016). We numerically compute it for our system using a QR algorithm (as motivated in Appendix D.3). The MLE gives a better measurement of stability than the Jacobian norm above, although it requires more effort to approximate. A positive MLE indicates chaotic dynamics and can lead to exploding gradients, while a negative MLE leads to vanishing ones.

5.4 Network perturbations and task variations

To test the adaptive capabilities of our model and to compare it with conventional RNNs, we consider two different ways in which these external inputs may be perturbed:

1. **Variable contrast:** transforming the inputs x_t ($Wx_t + b$). A brightness factor from a randomly sampled sinusoidal curve may multiplies the x_t input at each time-step t (Figure 2). These transformed inputs are then encoded by the same linear module W_{xh}.
2. **Perturbed:** applying an external drive directly to the neurons ($Wx_t + b$). Taking inspiration from optogenetic stimulation, we inject a scalar external drive $\xi \in \mathbb{R}$ directly to the neuron (e.g. for ARUN, we add $+\xi$ in equation (3)), before the activation function is applied. This perturbation may either be a non-random scalar (Figure 2) or noisy.

References

Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1120–1128. JMLR.org.

Arnold, L. (1998). *Random Dynamical Systems*. Springer.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. *32nd Conference on Neural Information Processing Systems*, abs/1803.09574.

Benda, J. and Herz, A. V. M. (2003). A Universal Model for Spike-Frequency Adaptation. *Neural Computation*, 15(11):2523–2564.

Benettin, G., Galgani, L., Giorgilli, A., and Strelcyn, M. (1980). Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: theory. *Meccanica*, 15:9–20.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bertschinger, N. and Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436.

Boedecker, J., Obst, O., Lizier, J. T., Mayer, N. M., and Asada, M. (2011). Information processing in echo state networks at the edge of chaos. *Theory Biosci.*, 131(3):205–213.

Burnham, D., Shea-Brown, E., and Mihalas, S. (2021). Learning to predict in networks with heterogeneous and dynamic synapses. *bioRxiv*.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control Signal Systems 2*, pages 303–314.

Deneve, S. (2008). Bayesian Spiking Neurons I: Inference. *Neural Computation*, 20(1):91–117.

Dieci, L. and Vleck, E. S. V. (1995). Computation of a few lyapunov exponents for continuous and discrete dynamical systems. *Applied Numerical Mathematics*, 17(3):275 – 291. Special Issue on Numerical Methods for Ordinary Differential Equations.

Fairhall, A. L., Lewen, G. D., Bialek, W., and de Ruyter van Steveninck, R. R. (2001a). Efficiency and ambiguity in an adaptive neural code. *Nature Publishing Group*, 412(6849):787–792.

Fairhall, A. L., Lewen, G. D., Bialek, W., and de Ruyter van Steveninck, R. R. (2001b). Efficiency and ambiguity in an adaptive neural code. *Nature*, 412(6849):787–792.

Fitz, H., Uhlmann, M., van den Broek, D., Duarte, R., Hagoort, P., and Petersson, K. M. (2020). Neuronal spike-rate adaptation supports working memory in language processing. *Proceedings of the National Academy of Sciences*, 117(34):20881–20889.

Gjorgjieva, J., Drion, G., and Marder, E. (2016). ScienceDirect Computational implications of biophysical diversity and multiple timescales in neurons and synapses for circuit performance. *Current Opinion in Neurobiology*, 37:44–52.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Gutierrez, G. J. and Denève, S. (2019). Population adaptation in efficient balanced networks. *eLife*, 8:e46926.

Hayou, S., Doucet, A., and Rousseau, J. (2019). On the impact of the activation function on deep neural networks training. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2672–2680. PMLR.

Henaff, M., Szlam, A., and LeCun, Y. (2016). Recurrent orthogonal networks and long-memory tasks. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2034–2042, New York, New York, USA. PMLR.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Kastner, D. B. and Baccus, S. A. (2011). Coordinated dynamic encoding in the retina using opposing forms of plasticity. *Nat Neurosci*, 14(10):1317–1322.

Kastner, D. B. and Baccus, S. A. (2013). Spatial segregation of adaptation and predictive sensitization in retinal ganglion cells. *Neuron*, 79(3):541–554.

Kerg, G., Goyette, K., Touzel, M. P., Gidel, G., Vorontsov, E., Bengio, Y., and Lajoie, G. (2019). Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics. *NeurIPS*.

Kilpatrick, Z. P. and Ermentrout, B. (2011). Sparse gamma rhythms arising through clustering in adapting neuronal networks. *PLOS Computational Biology*, 7(11):1–17.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Krishnamurthy, K., Can, T., and Schwab, D. J. (2022). Theory of gating in recurrent neural networks. *Phys. Rev. X*, 12(1).

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, MIT & NYU.

Laughlin, S. B. (1981). A simple coding procedure enhances a neuron's information capacity. *Zeitschrift für Naturforschung C*, 36:910 – 912.

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv e-prints*, page arXiv:1504.00941.

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941.

Legenstein, R. and Maass, W. (2007). *What makes a dynamical system computationally powerful?*, pages 127–154. MIT Press, 1 edition.

Lezcano-Casado, M. and Martínez-Rubio, D. (2019). Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group. *ICML*.

Lundstrom, B. N., Higgs, M. H., Spain, W. J., and Fairhall, A. L. (2008). Fractional differentiation by neocortical pyramidal neurons. *Nature Neuroscience*, 11(11):1335–1342.

O'Shea, D. J., Kalanithi, P., Ferenczi, E. A., Hsueh, B., Chandrasekaran, C., Goo, W., Diester, I., Ramakrishnan, C., Kaufman, M. T., Ryu, S. I., Yeom, K. W., Deisseroth, K., and Shenoy, K. V. (2018). Development of an optogenetic toolkit for neural circuit dissection in squirrel monkeys. *Sci Rep*, 8(1).

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). On the difficulty of training Recurrent Neural Networks. *arXiv e-prints*, page arXiv:1211.5063.

Pennington, J., Schoenholz, S. S., and Ganguli, S. (2018). The Emergence of Spectral Universality in Deep Networks. *arXiv.org*.

Peron, S. and Gabbiani, F. (2009). Spike frequency adaptation mediates looming stimulus selectivity in a collision-detecting neuron. *Nature neuroscience*, 12:318–26.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. (2016). Exponential expressivity in deep neural networks through transient chaos. *arXiv e-prints*.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. (2016). Exponential expressivity in deep neural networks through transient chaos. *arXiv.org*.

Pozzorini, C., Mensi, S., Hagens, O., Naud, R., Koch, C., and Gerstner, W. (2015). Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLOS Computational Biology*, 11(6):1–29.

Salaj, D., Subramoney, A., Kraisnikovic, C., Bellec, G., Legenstein, R., and Maass, W. (2021). Spike frequency adaptation supports network computations on temporally dispersed information. *eLife*, 10:e65459.

Sompolinsky, H., Crisanti, A., and Sommers, H. J. (1988). Chaos in random neural networks. *Phys. Rev. Lett.*, 61:259–262.

Vecoven, N., Ernst, D., Wehenkel, A., and Drion, G. (2020). Introducing neuromodulation in deep neural networks to learn adaptive behaviours. *PLOS ONE*, 15(1):1–13.

Vogt, R., Puelma Touzel, M., Shlizerman, E., and Lajoie, G. (2022). On lyapunov exponents for rnns: Understanding information propagation using dynamical systems tools. *Frontiers in Applied Mathematics and Statistics*, 8.

Weber, A. I., Krishnamurthy, K., and Fairhall, A. L. (2019). Coding principles in adaptation. *Annual Review of Vision Science*, 5(1):427–449. PMID: 31283447.

Winston, C. N., Mastrovito, D., Shea-Brown, E., and Mihalas, S. (2022a). Heterogeneity in neuronal dynamics is learned by gradient descent for temporal processing tasks. *bioRxiv*.

Winston, C. N., Mastrovito, D., Shea-Brown, E., and Mihalas, S. (2022b). Heterogeneity in neuronal dynamics is learned by gradient descent for temporal processing tasks. *bioRxiv*.

Yamins, D. L. K. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19:356 – 365.

Supplementary Material for:
Goal-driven optimization of single neuron properties in artificial networks reveals regularization role of neural diversity and adaptation

A Experimental details

Task	LR	hid	LR-scheduler	Rec. init.	In init.
copy	10^{-4}	128		Henaff	Glorot normal
psMNIST	10^{-4}	400		Henaff	Kaiming
PTB	10^{-4}	600	ReduceLROnPlateau	Henaff	Kaiming

Table 1: *Task-dependent hyperparameters*, where "hid" is hidden state size, "LR" is learning rate, "Rec. init." and "In init." are the initialization scheme for respectively the state transition matrix and the input weights.

The initialization schemes in Table 1 for the recurrent weights and input weights refer to :

Activation parameters The initialization grid for the activation parameters used throughout this work is $N \times S$, where $N = \{1.0\} \cup \{1.25k \ : \ 1 \leq k \leq 16\}$ and $S = \{0.0,\ 0.25,\ 0.5,\ 0.75,\ 1.0\}$ such that $|N| = 17$ and $|S| = 5$. In the heterogeneous adaptation scenario, both $\boldsymbol{n}$ and $\boldsymbol{s}$ vectors are initialized with the same value for each component.

Task independent stability metrics Figure 6 shows the task-independent stability metrics of JN and MLE for a range of (n,s) values (fixed across neurons). Clearly, activation shape influences Jacobian norms and will play an important role during training. Consistent with the average gradient norm, the MLE reports distinct (n, s)-regions of stability for random networks. In some cases, expansion and contractions can be useful for computations, and we further use these measurements to study training dynamics.

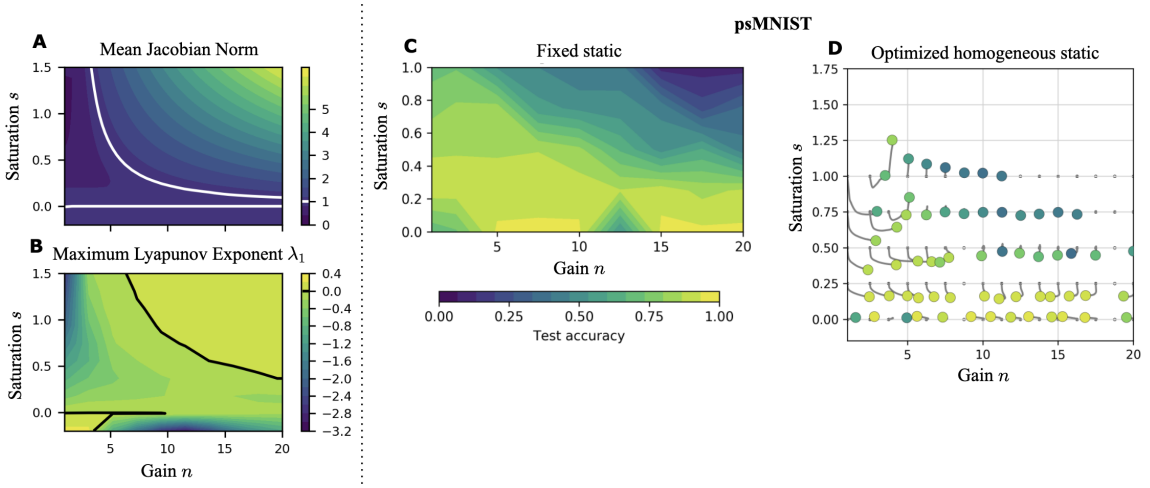

Figure 6: **A-B** Task independent stability metrics in activation parameter space. **C-D** Test accuracy in activation parameter space for the psMNIST task under two different learning scenarios.

B Performance: supplemental

B.1 Further details on learning differences and performance in the static setting

As expected, we find a strong correlation between the norm of the Jacobian in parameter space which is task-independent (Fig. 6**A**) and the performance landscapes for each task (see Fig. 6**C**). Interestingly, regions

Model	psMNIST		gsCIFAR10	
	Original	Noise pert.	Original	Noise pert.
RNN+ReLU	90.1 ± 0.2	33.0 ± 18.9	28.1 ± 0.2	16.6 ± 3.9
LSTM	89.6 ± 2.6	90.2 ± 2.8	36.4 ± 2.3	10.4 ± 0.7
GRU	93.3 ± 0.2	91.6 ± 0.7	$\mathbf{61.3 \pm 0.4}$	$\mathbf{60.4 \pm 0.1}$
RNN+γ (homo.)	95.8 ± 0.3	81.1 ± 12.0	42.5 ± 3.1	21.9 ± 3.1
RNN+γ (hetero.)	95.4 ± 0.5	84.0 ± 9.1	44.3 ± 2.9	23.7 ± 2.0
ARUN	95.4 ± 0.2	$\mathbf{94.7 \pm 0.4}$	42.4 ± 1.7	31.9 ± 3.5

Table 2: Bold indicates significant superior performance ($p < .01$, two sample t-test on pairwise comparisons).

in space (n, s) with poor performance are all associated with an exploding gradient, not a vanishing gradient. Networks whose activation functions have activation parameters in a neighborhood of $\{(n, s) : \|\gamma'(x; n, s)\| = 1\}$ present optimal performance, on all the tasks. On the one hand, this further emphasizes the performance of ReLU (see (Glorot et al., 2011)) as part of this (n, s)-neighborhood. However, as we show in Fig. 6**C**-**D**, traditional nonlinearities (including ReLU) are outperformed by the considerably different activation functions arising in the different scenarios of end-to-end learning. This result highlights that non trivial combinations of parameters may also achieve optimal performance while allowing for much more complex nonlinear transformations than ReLU.

B.2 Learned adaptation offers transfer learning advantages

In neuroscience, the term adaptation is mostly used to describe processes that occur on short timescales and at a neuron level which have been shown to account for changes in stimulus statistics (Weber et al., 2019). This mechanism is naturally linked to the concept of transfer learning in AI where one seeks systems where minimal changes in parameters allow adaptation from learned tasks to novel ones. To see if changes in single neurons activation could offer transfer advantages in ANNs, we design a novel task using the psMNIST test data set where the images are rotated by 45°. The goal is for a trained network to adapt to this change in input structure by only changing its activation function parameters. To evaluate this, we split rotated images into training and test sets, each containing approximately 5k images and the same number of images per digit. We then briefly retrain heterogeneous activation parameters (n_i, s_i) on this rotated data set using the heterogeneous adaptation scenario. For initialization, we take the parameters (including the (n_i, s_i)'s) that resulted from training with normal images, also under the heterogeneous adaptation scenario. Before retraining, the networks achieved an accuracy of 94% on the original data set, this fell to 42% after rotation. Retraining (n, s) allowed the networks to recover classification accuracy up to 56%. This shows that simply allowing the activation functions to adapt can recover over a quarter of lost performance (over 25%). An example of the variation of (n, s) trajectories after retraining is showed in Fig. 3**a** (bottom). Like in Fig. 3**a** (top), the cloud of (n, s) parameters expands with respect to its initialization, suggesting that a diversification in activation function shapes is needed to adapt to the change in task.

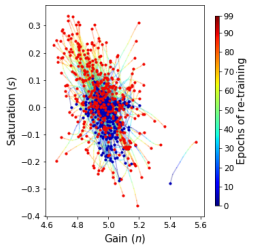

Figure 7: Trajectories of the activation parameters during retraining on the modified MNIST images.

Allowing for small changes in the activation functions of individual neurons helps to mitigate the loss in performance caused by drastic changes in network inputs. The following question naturally arises from these results: is it possible to leverage the advantages brought by adaptation in an online manner instead of relying on retraining a part of the network? Such a "dynamic" adaptation process, which allows the network's activation functions to instantly change when presented with inputs of different statistics, would not only be less computationally expensive and faster but would also be more alike its natural counterpart. We further explore the idea of implementing rapid adaptation protocols for ANNs in the next section.

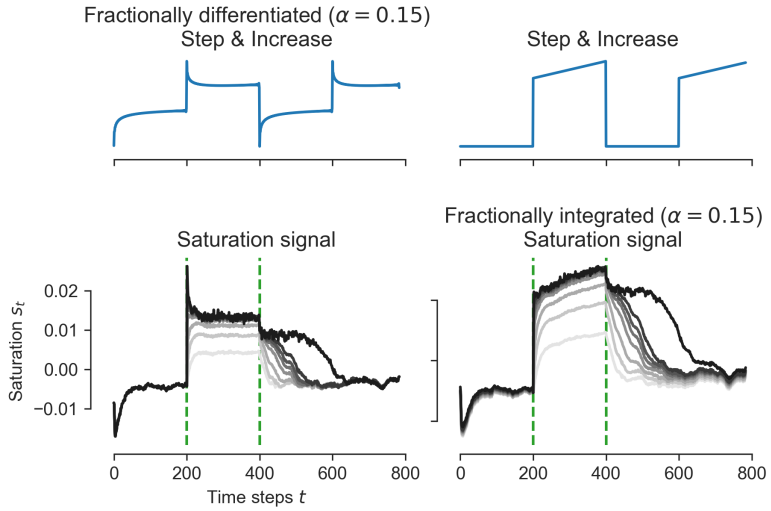

Figure 8: (**Top**) Graph of a step to linear-increase function (right), then fractional order ($\alpha = 0.15$) differentiated (left). (**Bottom**) Saturation s_t as a function of the time (left), for varying external drives $\xi \in [0, 30]$ with the usual range applied during a stimulation period framed by the two dashed green lines. See next Figure 9 for colorbar. (right) The saturation signals s_t fractionally integrated with $\alpha = 0.15$ reveal step to linear increase signals during the stimulation period.

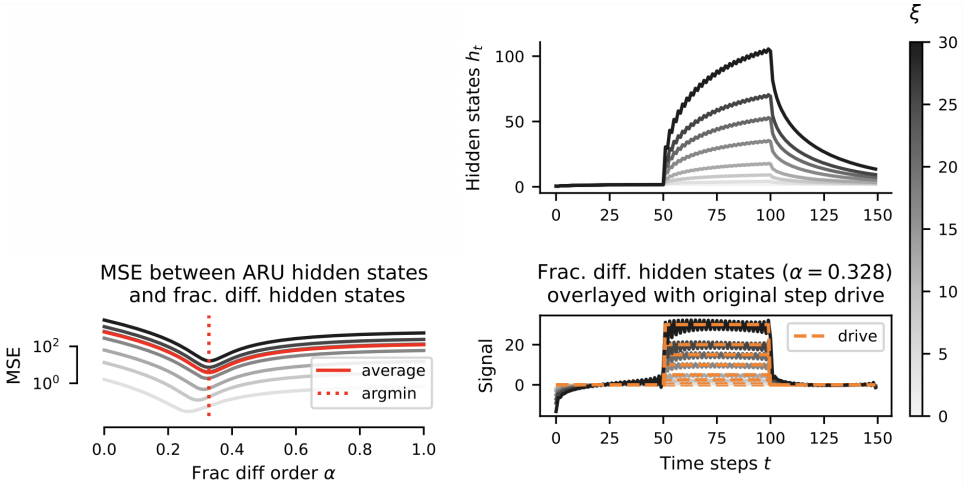

Figure 9: Task: psMNIST. Random seed #: 400. Colorbar applies to whole figure. (**top-right**) mean ARU hidden-states for non-interacting ARUs, just as main text's setting. For other panels, see respective titles.

C Adaptation: supplemental

C.1 Fractional differentiation

Activation function parameters Further details on fractional order differentiation of the activation parameter signals, as opposed to the resulting hidden-states h_t, is included in Figure 8.

Determination of fractional order The order α of fractional order differentiation was determined as the arg-min (over α) of the mean square error between the fractional α-order integrated signal and the precise step inputs that drove the network. See Figure 9. We observe that this minimum is sharp, and observe close correspondence between the fractional order integrated signal and the original step-drive. This analysis was consistent across tasks and random seeds (see examples in Figures 9, 10, 11).

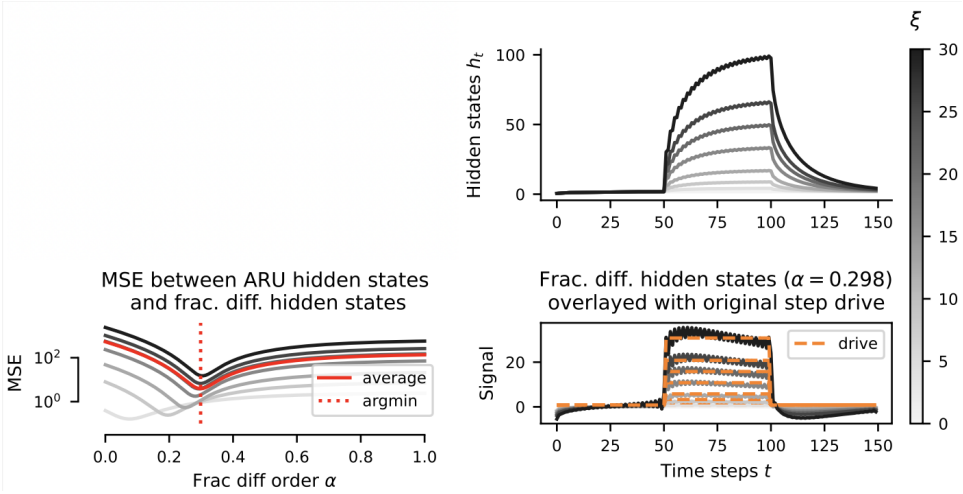

Figure 10: Task: psMNIST. Random seed #: 500.

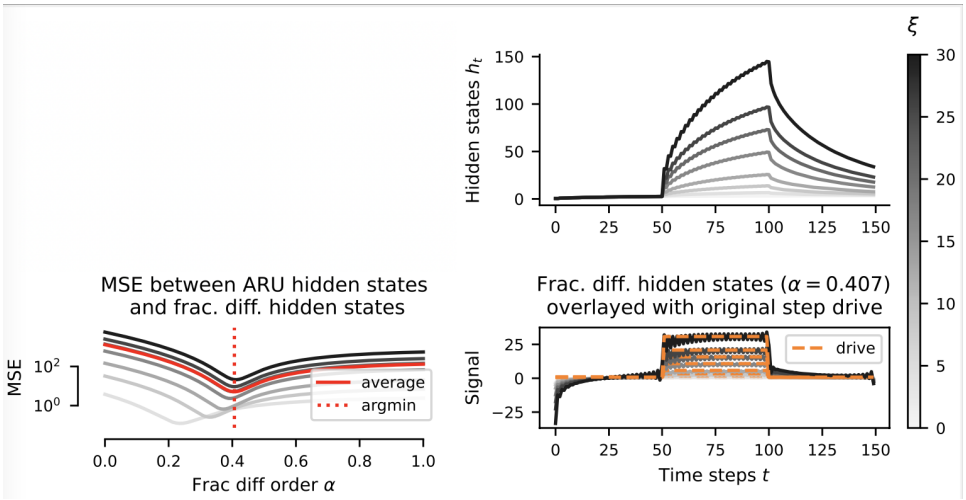

Figure 11: Task: gsCIFAR10. Random seed #: 403.

C.2 Further details on dynamic regularization

C.2.1 Proof of Proposition

Proof. Consider some multivariate Gaussian noise $\eta \sim \mathcal{N}(\mu, \sigma^2 I)$, injected in the dynamics

$$h_t = \gamma\left(W_{hh} h_{t-1} + \eta\,; n_t, s_t\right)$$

Now, the variance of this noise along a given vector $u \in \mathbb{R}^{N_h}$ as it propagates through the dynamics is given by:

$$\mathrm{Var}\left[u^\top W_{hh} \gamma(\eta\,; n_t, s_t)\right] = u^\top W_{hh} \mathrm{Var}\left[\gamma(\eta\,; n_t, s_t)\right] W_{hh}^\top u$$

after one iteration. Since η is chosen such that η_i is independent of η_j for $i \neq j$, i.e. $\mathrm{Cov}[\eta_i, \eta_j] = \sigma^2 \delta_{ij}$, and $\gamma(\cdot)$ acts element-wise, we have that $\mathrm{Cov}\left[\gamma(\eta_i), \gamma(\eta_j)\right] = 0$. As such,

$$\mathrm{Var}\left[\gamma(\eta\,; n_t, s_t)\right] = \mathrm{diag}\left\{\mathrm{Var}\left[\gamma(\eta_i; n_t^i, s_t^i)\right]\right\}_{1 \leq i \leq n} =: D_{n,s}$$

and

$$\mathrm{Var}\left[u^\top W_{hh} \gamma(\eta\,; n, s)\right] = u^\top W_{hh} D_{n,s} W_{hh}^\top u \tag{15}$$

Using a first order Taylor expansion of $\gamma(x\,; n_t, s_t)$ about the mean of η, we obtain

$$[D_{n,s}]_{i,i} = \text{Var}\left[\gamma(\eta_i; n_t^i, s_t^i)\right] = \sigma^2 \left[\frac{d}{dx}\gamma\left(\mathbb{E}[\eta_i]; n_t^i, s_t^i\right)\right]^2 + O\left(\mathbb{E}\left[(\eta_i - \mathbb{E}[\mu_i])^3\right]\right) \tag{16}$$

$$= \sigma^2 \left[\frac{d}{dx}\gamma\left(\mathbb{E}[\eta_i]; n_t^i, s_t^i\right)\right]^2 + O\left(\sigma^3\right) \tag{17}$$

where the first term of the RHS can easily be evaluated directly (see SM equation (10) for a closed form expression of $\frac{d}{dx}\gamma$). Also, under the initialisation schemes considered in our experiments, W_{hh} is unitary and as such $W_{hh} D_{n,s} W_{hh}^\top$ defines a normal matrix with eigenvalues exactly given by the entries of diagonal matrix $D_{n,s}$. This gives the result. □

C.2.2 Noise integration

Let us for a moment restrict our attention to a single neuron, thus removing subscripts i and assuming scalar quantities. We note in passing that σ is non-zero even for scalar ξ as our formalization accounts for the linearly scaled inputs x_t, which are distributed under the task input statistics. Now, consider the level set

$$\Lambda(\xi) := \left\{(n, s) : \frac{\partial}{\partial x}\gamma\left(\mu + \xi\,; n, s\right) = \frac{1}{|\sigma|}\right\} \tag{18}$$

consisting of (n, s) values at the boundary of the region derived from Proposition 1 for a noise shifted by an external drive $\xi \geq 0$ (un-perturbed if $\xi = 0$). As mentioned earlier this set corresponds to a manifold in (n, s) space, one that shifts as a function of ξ (see Fig. 3**c** for a visualization of these curves). Take $\{\hat{n}, \hat{s}\}$ satisfying Prop. 1, and assume that there exists $\epsilon > 0$ for which $d(\{\hat{n}, \hat{s}\}, \Lambda(0)) = \epsilon$. For noise robustness to be maintained in stimulated regimes, we have that the activation parameters $\{n(\xi), s(\xi)\}$ should shift to stay within the region highlighted by Prop 1, i.e. $d(\{\hat{n}, \hat{s}\}, \Lambda(\xi)) > 0$ for all $\xi \geq 0$. This is what we observe, see Fig. 3**c**.

Still, this does not account for particular behavior observed of an onset value $\{n_0, s_0\}$ decreasing or increasing with an exponential time-constant to a steady-state value $\{n_\infty, s_\infty\}$, in a matter akin to spike frequency adaptation. Both onsets and steady-states satisfy the observations previously highlighted, but their–distinct–existence is unaccounted for. This sets a rich ground for future work.

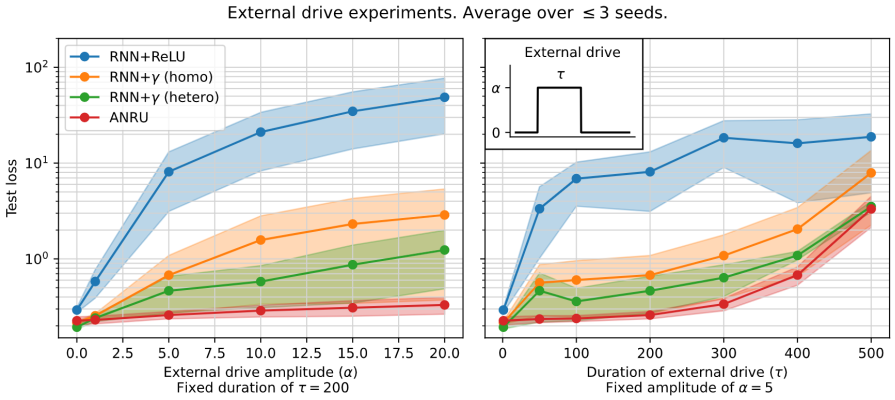

Figure 12: Sensitivity analysis for the step drive experiment. Lower is better. ARUN performs the best.

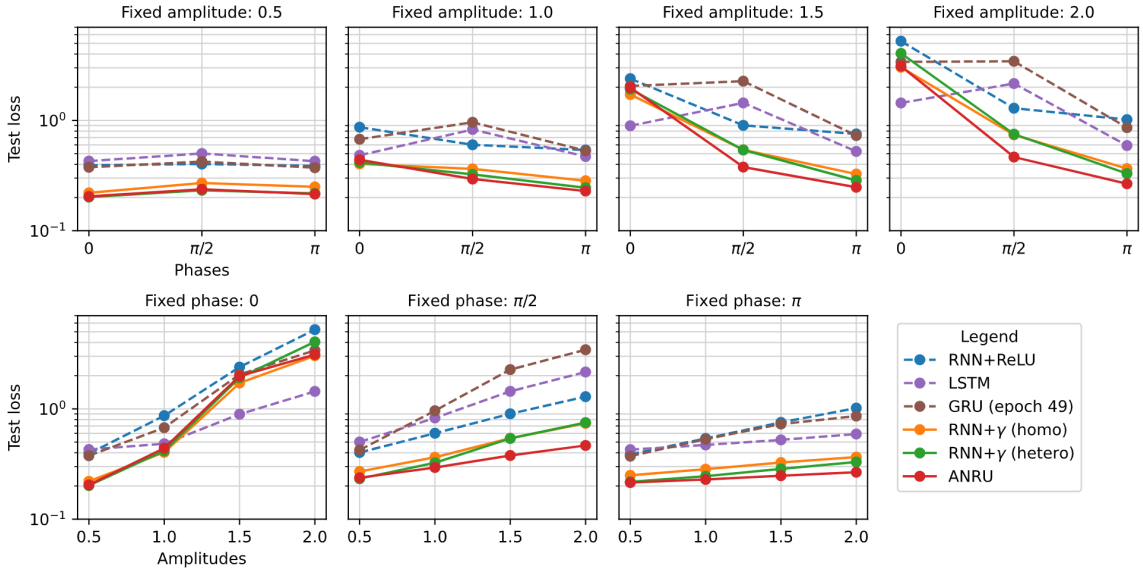

Figure 13: Sensitivity analysis for the Sinusoidal transformation on inputs, varying phase and amplitude alternatively. Lower is better. ARUN performs the best on average.

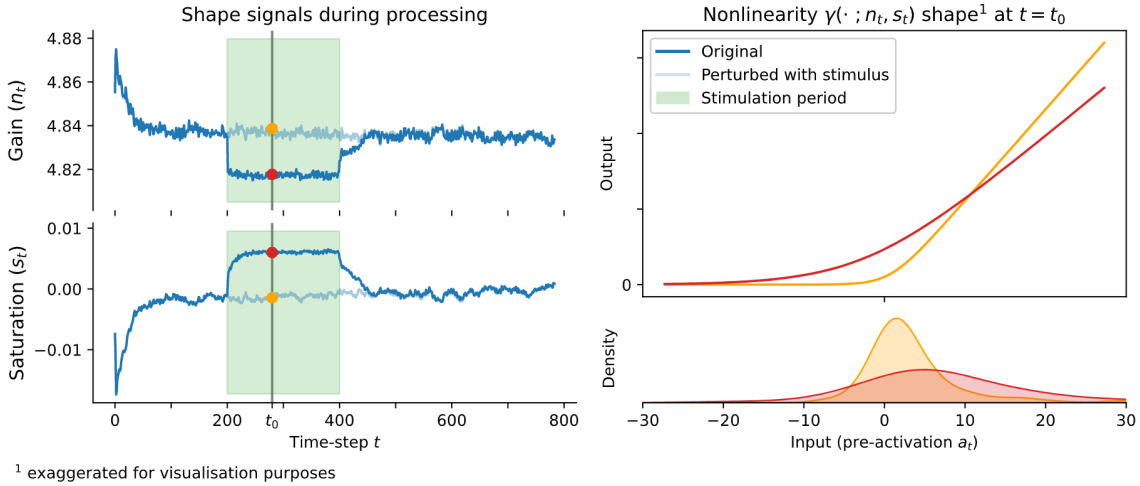

Figure 14: (Left) Modulation of gain n_t and saturation s_t during input processing, averaged over samples in a given batch. (Right, top) Shape of the activation function γ as dictated by $\{n_t^i.s_t^i\}$ at a given time-step, for the original and perturbed signals. (Right, bottom) Pre-activation density, i.e. an empirical distribution over neurons of what is received by the adaptation sub-network.

C.3 Testing the evolutionary plausibility of our adaptive units

The performance and robustness results presented in the main paper were obtained by randomly initializing and then simultaneosly training both the main RNN networks as well as the adaptive sub-networks. For our adaptive units to adequately model adaptation in biological neurons, the adaptive sub-units of each network should in principle be fixed when training the main RNN network. Indeed, in the brain, single neuron adaptation mechanisms have been developed over evolutionary timescales and are passed down through genetic information.

In this section we test our AURNs in a more biologically plausible setting, and see if the structure of the adaptive sub-network can be efficiently passed down from a network to another without affecting the network's

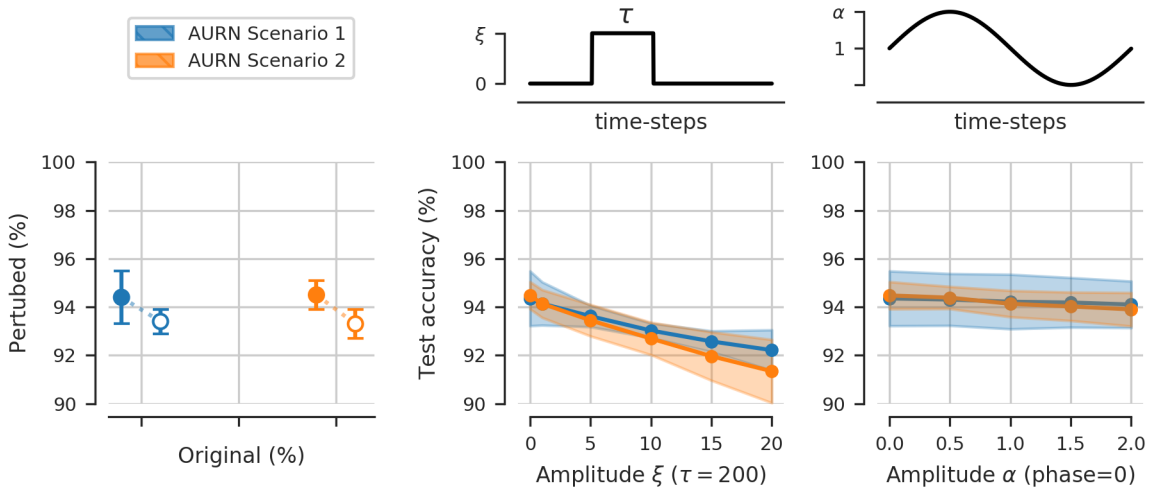

Figure 15: Performance on the psMNIST classification task and robustness to the noise perturbed, step drive transformed and sine transformed inputs. The mean and standard deviation across three different initializations are shown.

performance or robustness. To verify this we have tested the performance and robustness to the noise, step and sine data transformations of AURNs generated using two distinct initializing and training scenarios:

- **Scenario 1:** Both the main and the adaptive RNNs are randomly initialized, using a specified random seed (here denoted $seed_1$), and trained from scratch as previously described. All results from the paper are obtained with AURNs generated using this scenario.
- **Scenario 2:** The main RNN is randomly initialized using a specified random seed ($seed_2$) while the adaptive sub-network is taken from a scenario 1 trained AURN with $seed_1 \neq seed_2$. The main RNN is then trained using the same training procedure as in scenario 1 but the adaptive sub-network's parameters are kept constant.

This was done for multiple random seeds of both the main RNN and the trained adaptive RNN, the results are shown in Fig. 15. We can see that the adaptation mechanisms previously learned with a specific main network can be used, as efficiently, by another main network without needing any re-training of the sub-network. The performance and robustness to different perturbations are, for all practical purposes, the same in both the setting where the main and the adaptive networks were trained simultaneously (scenario 1) and the setting where the adaptive sub-network was imported from a previously trained network and only the main network was trained (scenario 2).

C.4 Gradient contribution according to position in input sequence

The vanishing (resp. exploding) gradient problem in RNNs can be characterized by inputs at time-step t, which are further away from the moment the loss is computed (time-step $t + \delta$), contributing exponentially less (resp. more) to the gradient than inputs closer to $t + \delta$ as δ increases. To see the effects of neural adaptation on gradient propagation in RNNs we have computed the gradient contribution to the hidden-to-hidden weight matrix gradients of the psMNIST training set pixels with respect to their position in the input sequence. We plot the results in Fig.16. We see that in ReLU RNNs, the gradient contribution is almost perfectly monotonously increasing with respect to the element's (the pixel's) position in the input sequence, i.e. pixels early on in the input sequence contribute significantly less to the gradient than pixels at the end of the sequence. In ARUNs however, and to some extent even in RNN γ hetero, the gradient contribution is more uniform across the different pixel positions with almost all pixels contributing equally to the gradients regardless of their position in the input sequence. This shows that the adaptive capabilities of ARUNs help improve information (gradient) propagation in the model's training process.

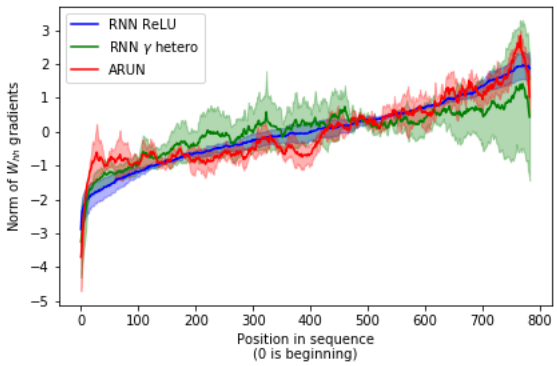

Figure 16: Frobenius norm of the hidden-to-hidden weight matrix W_{hh} gradient contribution of a given input element, or pixel, as a function of that element's position in the input sequence. The sequences are of length 784 and elements closer to position 0 are closer to the beginning of the input sequences. The gradients are computed in trained RNN+ReLU, RNN+γ heterogeneous and ARU networks on the psMNIST training set. The values were **standardized** for comparison purposes, mean and standard deviation across three random initialization are shown.

D A primer on Lyapunov exponents

In this section we are first going to give a bit of theoretical background on Lyapunov exponents. Exponential explosion and vanishing of long products of Jacobian matrices is a long studied topic in dynamical systems theory, where an extensive amount of tools have been developed in order to understand these products. Thus one can hope to leverage these tools in order to better understand the exploding and vanishing gradient problem in the context of RNNs.

D.1 Definition of Lyapunov exponents

Let $F : X \to X$ be a continuously differentiable function, and consider the discrete dynamical system (F, X, T) defined by

$$x_{t+1} = F(x_t) \tag{19}$$

for all $t \in T$, where X is the phase space, and T the time range. We would like to gain an intuition for how trajectories of the mentioned dynamical system behave under small perturbations.

Let x_t and x'_t be two trajectories with initial conditions x_0 and x'_0, such that $|x_0 - x'_0|$ is sufficiently small.

Defining $\epsilon_t = x'_t - x_t$, we get by the first order Taylor expansion

$$\begin{aligned} x'_{t+1} &= F(x'_t) && (20)\\ &= F(x_t + \epsilon_t) && (21)\\ &= F(x_t) + DF(x_t) \cdot \epsilon_t + O(|\epsilon_t|^2) && (22)\\ &= x_{t+1} + DF(x_t) \cdot \epsilon_t + O(|\epsilon_t|^2) && (23) \end{aligned}$$

Substracting x_{t+1} both sides we get the variational equation

$$\begin{aligned} \epsilon_{t+1} &= DF(x_t) \cdot \epsilon_t + O(|\epsilon_t|^2) && (24)\\ &\approx \prod_{k=0}^{t} DF(x_k) \cdot \epsilon_0 && (25)\\ &= DF^{t+1}(x_0) \cdot \epsilon_0 && (26) \end{aligned}$$

(Here $DF^{t+1}(x_0)$ is an abuse of notation for the Jacobian of the $(t+1)$-th iterate of F, evaluated at x_0). Intuitively the ratio $\frac{\|\epsilon_t\|}{\|\epsilon_0\|} = \frac{\|DF^t(x_0)\cdot\epsilon_0\|}{\|\epsilon_0\|}$ describes the expansion/contraction rate after t time steps if our initial perturbation was ϵ_0, which motivates the following definition:

Let $x_0, w \in X$, define

$$\begin{aligned} \lambda(x_0, w) &\overset{\text{def}}{=} \lim_{m\to\infty} \frac{1}{m} \ln \prod_{t=1}^{m} \frac{\|DF^t(x_0)\cdot w\|}{\|w\|} && (27)\\ &= \lim_{m\to\infty} \frac{1}{m} \sum_{t=1}^{m} \ln \frac{\|DF^t(x_0)\cdot w\|}{\|w\|} && (28) \end{aligned}$$

Thus $\lambda(x_0, w)$ measures the average rate of expansion/contraction over an infinite time horizon of the trajectory starting at x_0, if it has been given an initial perturbation w. Note that once x_0 and w have been fixed, the quantity $\lambda(x_0, w)$ is intrinsic to the discrete dynamical system defined by $x_{t+1} = F(x_t)$. We call $\lambda(x_0, w)$ a **Lyapunov exponent** of the mentioned dynamical system.

Since the Lyapunov exponents describe the the average rate of expansion/contraction for long products of Jacobian matrices, it doesn't sound too surprising that they might provide an interesting perspective to study the exploding and vanishing gradient problem in RNNs. To give a complete picture of the analogy to RNNs, one can think of x_t as the hidden state at time t, and F can be seen as the function defined in the RNN cell. The only difference is that in RNNs we have inputs at every time steps, and thus the function F changes at every time step. This is the distinction between autonomous and non-autonomous dynamical systems, which is explained in more detail in the upcoming subsection D.4.

Finally, let us remark that the expression in the above definition of Lyapunov exponents is not always well defined. This will be the topic of the next subsection D.2, where we are presenting Oseledets theorem which gives exact conditions for when the above expression in well-defined.

D.2 Oseledets theorem

As already stated, we bypassed the fact that the limit in the definition of $\lambda(x_0, w)$ might not actually exists. In fact this is the result of the well-known *Oseledets theorem*, but before stating the theorem let us point out a definition.

Definition. A *cocycle* of an autonomous dynamical system (F, X, T) is a map $C : X \times T \to \mathbb{R}^{n\times n}$ satisfying:

- $C(x_0, 0) = \mathrm{Id}$
- $C(x_0, t + s) = C(x_t, s)C(x_0, t)$ for all $x_0 \in X$ and $s, t \in T$

Oseledets theorem. (sometimes referred to as Oseledets *multiplicative ergodic theorem*) Let μ be an ergodic invariant measure on X, and let C be a cocycle of a dynamical system (F, X, T) such that for each $t \in T$, the maps $x \mapsto \log \|C(x, t)\|$ and $x \mapsto \log \|C(x, t)^{-1}\|$ are L^1-integrable with respect to μ. Then for μ-almost all x and each non-zero vector $w \in \mathbb{R}^n$ the limit

$$\lambda(x, w) = \lim_{t\to\infty} \frac{1}{t} \ln \frac{\|C(x, t)w\|}{\|w\|} \tag{29}$$

exists and assumes, depending on w but not on x, up to n different values, called the Lyapunov exponents (giving rise to a more general definition)

One can prove that the following matrix limit

$$\Lambda = \lim_{t\to\infty} [C(x, t)^T C(x, t)]^{1/2t} \tag{30}$$

exists, is symmetric positive-definite and its log-eigenvalues are the Lyapunov exponents. We call Λ the *Oseledets matrix*.

In order to make this definition a little bit more intuitive, let us come back to our original situation, and note that the terms $\prod_{k=0}^{t} DF(x_k) = DF^{t+1}(x_0)$ define a cocycle verifying the conditions of the theorem. Thus, in this case, the Lyapunov exponents are not only well defined, but there are up to n distinct ones of them, and they are the log-eigenvalues of the following Oseledets matrix:

$$\Lambda = \lim_{t\to\infty} [DF^t(x_0)^T \cdot DF^t(x_0)]^{1/2t} \tag{31}$$

Let us now consider the singular value decomposition of $DF^t(x_0)$,

$$DF^t(x_0)V(x_0, t) = U(x_0, t)\Sigma(x_0, t) \tag{32}$$

where $\Sigma(x_0, t)$ is a diagonal matrix composed of the singular values $\sigma_1(x_0, t) \geq \ldots \geq \sigma_n(x_0, t) \geq 0$, and $U(x_0, t)$ as well as $V(x_0, t)$ are orthogonal matrices, composed column-wise of the left and right singular vectors respectively. Then

$$\Lambda = \lim_{t\to\infty} V(x_0,t)^T \Sigma(x_0,t)^{1/t} V(x_0,t) \tag{33}$$

Thus, for large t, the log-eigenvalues of Λ can be approximated by $\frac{1}{t}\ln\sigma_i(x_o,t)$'s, which can be thought of as the average singular value along an infinite time horizon. It turns out that for ergodic systems, the Lyapunov exponents are independent of initial conditions x_0. Thus, intuitively, Lyapunov exponents are topological quantities intrinsic to the dynamical system that describe the average amount of instability along infinite time horizons.

In order to understand how this instability manifests along each direction, let us further look what we can say about the vectors associated with the individual Lyapunov exponents. If we denote $\lambda^{(1)} \geq \lambda^{(2)} \geq \ldots \geq \lambda^{(s)}$ the *distinct* Lyapunov exponents, and $v_i(x_0)$ the corresponding vector of the matrix $\lim_{t\to\infty} V(x_0,t)$, then let us define the nested subspaces

$$S_j(x_0) = \text{span}\{v_i(x_0) | i = j, j+1, \ldots, s\} \tag{34}$$

for all $j = 1, 2, \ldots, s$, and take a vector $w_j(x_0) \in S_j(x_0) \setminus S_{j+1}(x_0)$. Then $w_j(x_0)$ is orthogonal to all $v_i(x_0)$ with $i < j$, and has a non-zero projection onto $v_j(x_0)$ since $v_j(x_0) \notin S_{j+1}(x_0)$, and thus

$$\|DF^t(x_0)\cdot w_j(x_0)\| \sim e^{\lambda^{(j)}t} \tag{35}$$

In particular, since $S_1(x_0)$ is the whole phase space X, and $S_2(x_0)$ is only a hyperplane in X (a subset of Lebesgue measure zero), we have that for "almost all" $w \in X$:

$$\|DF^t(x_0)\cdot w\| \sim e^{\lambda^{(1)}t} \tag{36}$$

hence aligning with the direction of maximum Lyapunov exponent (MLE). In other words a randomly chosen vector, has a non-zero projection in the direction of the MLE with probability 1, and thus over time the effect of the other exponents will become negligible. This motivates taking the MLE as a way of measuring the overall amount of stability or instability of a dynamical system. One typically distinguishes the cases, where the MLE is negative, zero and positive.

Thus computing MLEs, LEs and their corresponding subspaces can be a useful tool to understand the average expansion/ contraction rate as well as the corresponding directions of gradients in recurrent neural networks.

D.3 The QR algorithm

It is generally not advised to calculate the Lyapunov exponents and the associated vectors using $DF^t(x_0)$ as this matrix becomes increasingly ill-conditioned. There is a known algorithm that in most cases allows to provide good estimates, called the *QR algorithm*.

As a preliminary remark, let us emphasize that the right singular vectors of $DF(x_{t+1})$ do not necessarily match the left singular vectors of $DF(x_t)$, thus simply applying the singular value decomposition in order to calculate the Lyapunov exponents does not work.

Let us denote $J_t = DF(x_t)$ for each time step $t = 0, 1, 2, \ldots$, then lets us pick an orthogonal matrix Q_0, and compute $Z_0 = J_0 Q_0$. Then let us perform the QR decomposition $Z_0 = Q_1 R_1$. Let us further assume that J_0 is invertible and we are imposing that the diagonal elements of R_1 are non-negative (which we can), thus making the QR decomposition unique.

In the next step, we compute $Z_1 = J_1 Q_1$ and perform the QR decomposition $Z_1 = Q_2 R_2$, where again we are imposing the diagonal elements of R_2 to be non-negative.

Continuing in this fashion at each time step k, we then have the identity $J_k = Q_{k+1} R_{k+1} Q_k^T$, and thus

$$DF^{t+1}(x_0) = \prod_{k=0}^{t} J_k \tag{37}$$

$$= Q_{t+1}(R_t \cdot \ldots \cdot R_1) Q_0^T \tag{38}$$

It turns out that, as long as the dynamical system is "regular", we can then compute the i-th Lyapunov exponent via

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \sum_{k=1}^{t} \ln(R_k)_{ii} \tag{39}$$

where the Lyapunov exponents are ordered $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ as explained in Benettin et al. (1980) and Dieci and Vleck (1995).

D.4 Link to RNNs

Recalling the update equation of an RNN:

$$h_{t+1} = \phi(Vh_t + Ux_{t+1} + b) \tag{40}$$

for $t = 0, 1, \ldots$, and by denoting $F(h, x) = \phi(Vh + Ux + b)$, we can see that

$$\tilde{h}_{t+1} = F(\tilde{h}_t, 0) \tag{41}$$

defines an autonomous discrete dynamical system (DS1), while

$$h_{t+1} = F(h_t, x_{t+1}) \tag{42}$$

defines a non-autonomous discrete dynamical system (DS2).

For (DS1), the machinery that we have developed over the last subsections is directly applicable, as we are in the autonomous case. For instance, we can compute the Lyapunov exponents of recurrent neural network over the course of training using the QR algorithm, and in particular observe the evolution of the maximum Lyapunov exponent (MLE), as a means to measure the amount of instability or chaos in the network. For example in the case of a linear RNN with a unitary or orthogonal connectivity matrix, all LEs are equal to zero, and thus no expansion nor contraction is happening. If all LEs are negative, we are in the contracting regime, where every point eventually will approach an attractor, thus producing a vanishing gradient. For instance, Bengio et al. (1994) showed that storing information in a fixed-size state vector (as is the case in a vanilla RNN) over sufficiently long time horizon in a stable way necessarily leads to vanishing gradients when back-propagating through time (here stable means insensitive to small input perturbations).

The natural question arises whether and to what extent the machinery will stay valid for (DS2). It turns out that one can use the theory of Random Dynamical Systems Theory, where Oseledet's multiplicative ergodic theorem holds under some stationarity assumption of the underlying distribution generating the inputs x_t as stated in Arnold (1998). However in this paper we are just making use of the machinery developed for (DS1), by computing Lyapunov exponents for trained RNNs but computed without inputs ($x_t = 0$ for all t).

Supplementary References

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Lecture 6e, rmsprop: Divide the gradient by a running average of its recent magnitude.