ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Coupled Oscillatory Recurrent Neural Network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies

T. K. Rusch and S. Mishra

Research Report No. 2020-63
October 2020

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Funding ERC: 770880 COMANFLO

Coupled Oscillatory Recurrent Neural Network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies

T. Konstantin Rusch * Siddhartha Mishra *

October 2, 2020

Abstract

Circuits of biological neurons, such as in the functional parts of the brain can be modeled as networks of coupled oscillators. Inspired by the ability of these systems to express a rich set of outputs while keeping (gradients of) state variables bounded, we propose a novel architecture for recurrent neural networks. Our proposed RNN is based on a time-discretization of a system of second-order ordinary differential equations, modeling networks of controlled nonlinear oscillators. We prove precise bounds on the gradients of the hidden states, leading to the mitigation of the exploding and vanishing gradient problem for this RNN. Experiments show that the proposed RNN is comparable in performance to the state of the art on a variety of benchmarks, demonstrating the potential of this architecture to provide stable and accurate RNNs for processing complex sequential data.

1 Introduction

Recurrent neural networks (RNNs) have achieved tremendous success in a variety of tasks involving sequential (time series) inputs and outputs, ranging from speech recognition to computer vision and natural language processing, among others. However, it is well known that training RNNs to process inputs over long time scales (input sequences) is notoriously hard on account of the so-called *exploding and vanishing gradient problem (EVGP)* [33], which stems from the fact that the well-established BPTT algorithm for training RNNs requires computing products of gradients (Jacobians) of the underlying hidden states over very long time scales. Consequently, the overall gradient can grow (to infinity) or decay (to zero) exponentially fast with respect to the number of recurrent interactions.

A variety of approaches have been suggested to mitigate the exploding and vanishing gradient problem. These include adding *gating mechanisms* to the RNN in order to control the flow of information in the network, leading to architectures such as *long short-term memory* (LSTM) [21] and *gated recurring units* (GRU) [10], that can overcome the vanishing gradient problem on account of the underlying additive structure. However, the gradients might still explode and learning very long term dependencies remains a challenge [30]. Another popular approach for handling the EVGP is to *constrain* the structure of underlying recurrent weight matrices by requiring them to be orthogonal (unitary), leading to the so-called *orthogonal RNNs* [20, 2, 42, 24] and references therein. By construction, the resulting Jacobians have eigen- and singular-spectra with unit norm, alleviating the EVGP. However as pointed out in [24], imposing such constraints on the recurrent matrices may lead to a significant loss of *expressivity* of the RNN resulting in inadequate performance on realistic tasks.

In this article, we adopt a different approach, based on observation that *coupled networks of controlled non-linear forced and damped oscillators*, that arise in many physical, engineering and biological systems such as networks of biological neurons, do seem to ensure expressive representations while constraining the dynamics of state variables and their gradients. This motivates us to propose a novel architecture for RNNs, based on time-discretizations of second-order systems of non-linear ordinary differential equations

*Seminar for Applied Mathematics (SAM), D-MATH
ETH Zürich, Rämistrasse 101, Zürich-8092, Switzerland

(ODEs) (1) that model coupled oscillators. For these RNNs, we are able to *rigorously prove precise bounds on the hidden states and their gradients, enabling the solution of the exploding and vanishing gradient problem*, while demonstrated through benchmark numerical experiments, that the resulting system still retains sufficient expressivity, with a performance comparable to the state of the art on a variety of sequential learning tasks.

2 The proposed RNN

Our proposed RNN is based on the following second-order system of ODEs,

$$\mathbf{y}'' = \sigma\left(\mathbf{W}\mathbf{y} + \mathcal{W}\mathbf{y}' + \mathbf{V}\mathbf{u} + \mathbf{b}\right) - \gamma\mathbf{y} - \epsilon\mathbf{y}'. \tag{1}$$

Here, $t \in [0,1]$ is the (continuous) time variable, $\mathbf{u} = \mathbf{u}(t) \in \mathbb{R}^d$ is the time-dependent *input signal*, $\mathbf{y} = \mathbf{y}(t) \in \mathbb{R}^m$ is the *hidden state* of the RNN with $\mathbf{W}, \mathcal{W} \in \mathbb{R}^{m\times m}$, $\mathbf{V} \in \mathbb{R}^{m\times d}$ are weight matrices, $\mathbf{b} \in \mathbb{R}^m$ is the bias vector and $0 < \epsilon, \gamma$ are parameters. $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is the *activation function*, set to $\sigma(u) = \tanh(u)$ here. By introducing the so-called *velocity* variable $\mathbf{z} = \mathbf{y}'(t) \in \mathbb{R}^m$, we rewrite (1) as the first-order system:

$$\mathbf{y}' = \mathbf{z}, \quad \mathbf{z}' = \sigma\left(\mathbf{W}\mathbf{y} + \mathcal{W}\mathbf{z} + \mathbf{V}\mathbf{u} + \mathbf{b}\right) - \gamma\mathbf{y} - \epsilon\mathbf{z}. \tag{2}$$

We fix a timestep $0 < \Delta t < 1$ and define our proposed RNN hidden states at time $t_n = n\Delta t \in [0,1]$ (while omitting the affine output state) as the following IMEX (implicit-explicit) discretization of the first order system (2):

$$\begin{aligned} \mathbf{y}_n &= \mathbf{y}_{n-1} + \Delta t\mathbf{z}_n, \\ \mathbf{z}_n &= \mathbf{z}_{n-1} + \Delta t\sigma\left(\mathbf{W}\mathbf{y}_{n-1} + \mathcal{W}\mathbf{z}_{n-1} + \mathbf{V}\mathbf{u}_n + \mathbf{b}\right) - \Delta t\gamma\mathbf{y}_{n-1} - \Delta t\epsilon\mathbf{z}_n. \end{aligned} \tag{3}$$

Motivation and background. We term the RNN (3) as *coupled oscillatory Recurrent Neural Network* (coRNN), because each neuron is a *controlled forced, damped nonlinear oscillator* [18], with diagonal entries of $\mathbf{W}$ and $\mathcal{W}$ controlling the frequency and amount of damping of the oscillation, respectively, whereas the non-diagonal entries of these matrices modulate interactions between neurons in the network. The parameters $\mathbf{V}, \mathbf{b}$ modulate the effect of the driving force proportional to the input signal $\mathbf{u}(t)$ and the tanh activation mediates a non-linear response. We provide heuristics for the dynamics of oscillator networks in **SM**§B, where we demonstrate with simple examples that a network of (forced, driven) oscillators can access a very rich set of output states, in particular oscillatory input signals can yield non-oscillatory outputs. This ability of such systems to express a variety of output states indicating the possibility of *high expressivity* for the proposed RNN.

Such oscillator networks are ubiquitous in nature and in engineering systems [18, 38] with canonical examples being pendulums (classical mechanics), business cycles (economics), heartbeat (biology) for single oscillators and electrical circuits for networks of oscillators. Our motivating examples arises in neurobiology, where individual biological neurons can be viewed as oscillators with periodic spiking and firing of the action potential. Moreover, functional circuits of the brain, such as cortical columns and prefrontal-striatal-hippocampal circuits, are being increasingly interpreted by networks of oscillatory neurons, see [37] for an overview and [17] for modeling specific brain functions such as interval timing and working memory as oscillatory neural networks. Following well-established paths in machine learning such as for convolutional neural networks [29], our focus here is to abstract the essence of functional brain circuits being networks of oscillators and design an RNN based on much simpler mechanistic systems such as those modeled by (2), while ignoring the complicated biological details of neural function.

Related work. While there are many examples of ODE and dynamical systems inspired RNN architectures, these approaches can roughly be distinguished into two branches, namely RNNs based on discretized ODEs and continuous-time RNNs. Examples of continuous-time approaches include neural ODEs [8] with ODE-RNNs [35] as its recurrent extension as well as [13] and references therein, to name just a few. We focus, however, in this article on an ODE-inspired discrete-time RNN, as the proposed coRNN is derived from a discretized ODE. A prominent example for discrete-time ODE-based RNNs is the so-called *anti-symmetric* RNN of [6], where the RNN architecture is based on a stable ODE using a

skew-symmetric hidden weight matrix. We also mention hybrid methods, which use a discretization of an ODE (in particular a Hamiltonian system) in order to learn the continuous representation of the data, see for instance [15, 9]. Our approach here differs from these papers in our explicit use of networks of oscillators, with the underlying biological motivation.

3 Rigorous analysis of the proposed RNN

For simplicity of exposition, we set $\mathbf{y}_0 = \mathbf{z}_0 = 0, \epsilon = \gamma = 1$ in (3) leading to,

$$\begin{aligned} \mathbf{y}_n &= \mathbf{y}_{n-1} + \Delta t \mathbf{z}_n, \\ \mathbf{z}_n &= \tfrac{\mathbf{z}_{n-1}}{1+\Delta t} + \tfrac{\Delta t}{1+\Delta t}\sigma(\mathbf{A}_{n-1}) - \tfrac{\Delta t}{1+\Delta t}\mathbf{y}_{n-1}, \ \mathbf{A}_{n-1} := \mathbf{W}\mathbf{y}_{n-1} + \mathcal{W}\mathbf{z}_{n-1} + \mathbf{V}\mathbf{u}_n + \mathbf{b}. \end{aligned} \tag{4}$$

Bounds on the hidden states. As the hidden states in the RNN (3) are the outputs of a network of driven, damped oscillators, with a bounded (tanh) nonlinearity, it is straightforward to obtain,

Proposition 3.1. *Let $\mathbf{y}_n, \mathbf{z}_n$ be the hidden states of the RNN (4) for $1 \le n \le N$, then the hidden states satisfy the following (energy) bounds:*

$$\mathbf{y}_n^\top \mathbf{y}_n + \mathbf{z}_n^\top \mathbf{z}_n \le nm\Delta t = mt_n \le m. \tag{5}$$

The proof of the *energy* bound (5) is provided in **SM**§C.1 and a straightforward variant of the proof (see **SM**§C.2) yields an estimate on the sensitivity of the hidden states to changing inputs. In particular, this bound *rules out chaotic behavior of hidden states.*

Bounds on hidden state gradients. We train the RNN (3) to minimize the loss function,

$$\mathcal{E} := \frac{1}{N}\sum_{n=1}^{N} \mathcal{E}_n, \quad \mathcal{E}_n = \frac{1}{2}\|\mathbf{y}_n - \bar{\mathbf{y}}_n\|_2^2, \tag{6}$$

with $\bar{\mathbf{y}}$ being the underlying ground truth (training data). During training, we compute gradients of the loss function (6) with respect to the weights and biases $\mathbf{\Theta} = [\mathbf{W}, \mathcal{W}, \mathbf{V}, \mathbf{b}]$, i.e.

$$\frac{\partial \mathcal{E}}{\partial \theta} = \frac{1}{N}\sum_{n=1}^{N} \frac{\partial \mathcal{E}_n}{\partial \theta}, \quad \forall\ \theta \in \mathbf{\Theta}. \tag{7}$$

Proposition 3.2. *Let $\mathbf{y}_n, \mathbf{z}_n$ be the hidden states generated by the RNN (4). We assume that the time step Δt can be chosen such that,*

$$\max\left\{\frac{\Delta t(1+\|\mathbf{W}\|_\infty)}{1+\Delta t}, \frac{\Delta t\|\mathcal{W}\|_\infty}{1+\Delta t}\right\} = \eta \le \Delta t^r, \quad \frac{1}{2} \le r \le 1 \tag{8}$$

Denoting $\delta = \frac{1}{1+\Delta t}$, the gradient of the loss function $\mathcal{E}$ (6) with respect to any parameter $\theta \in \mathbf{\Theta}$ is bounded as,

$$\left|\frac{\partial \mathcal{E}}{\partial \theta}\right| \le Cm\delta, \tag{9}$$

for some constant C, independent of the parameters of (4).

Sketch of the proof. Denoting $\mathbf{X}_n = [\mathbf{y}_n, \mathbf{z}_n]$, we can apply the chain rule repeatedly (for instance as in [33]) to obtain,

$$\frac{\partial \mathcal{E}_n}{\partial \theta} = \sum_{1 \le k \le n} \underbrace{\frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n}\frac{\partial \mathbf{X}_n}{\partial \mathbf{X}_k}\frac{\partial^+ \mathbf{X}_k}{\partial \theta}}_{\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta}}. \tag{10}$$

Here, the notation $\frac{\partial^+ \mathbf{X}_k}{\partial \theta}$ refers to taking the partial derivative of $\mathbf{X}_k$ with respect to the parameter θ, while keeping the other arguments constant. This quantity can be readily calculated from the structure of the RNN (4) and is presented in the detailed proof provided in **SM**§C.3. From (6), we can directly compute that $\frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n} = [\mathbf{y}_n - \bar{\mathbf{y}}_n, 0]$.

Repeated application of the chain rule and a direct calculation with (4) yields,

$$\frac{\partial \mathbf{X}_n}{\partial \mathbf{X}_k} = \prod_{k<i\leq n} \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_{i-1}}, \quad \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_{i-1}} = \begin{bmatrix} \mathrm{I} + \Delta t \mathbf{B}_{i-1} & \Delta t \mathbf{C}_{i-1} \\ \mathbf{B}_{i-1} & \mathbf{C}_{i-1} \end{bmatrix}, \tag{11}$$

where I is the identity matrix and

$$\mathbf{B}_{i-1} = \delta \Delta t \left(\mathrm{diag}(\sigma'(\mathbf{A}_{i-1}))\mathbf{W} - \mathrm{I}\right), \quad \mathbf{C}_{i-1} = \delta \left(\mathrm{I} + \Delta t \, \mathrm{diag}(\sigma'(\mathbf{A}_{i-1}))\mathcal{W}\right). \tag{12}$$

It is straightforward to calculate using the assumption (8) that $\|\mathbf{B}_{i-1}\|_\infty < \eta$ and $\|\mathbf{C}_{i-1}\|_\infty \leq \eta + \delta$. Using the definitions of matrix norms and (8), we obtain:

$$\begin{aligned} \left\| \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_{i-1}} \right\|_\infty &\leq \max\left(1 + \Delta t(\|\mathbf{B}_{i-1}\|_\infty + \|\mathbf{C}_{i-1}\|_\infty), \|\mathbf{B}_{i-1}\|_\infty + \|\mathbf{C}_{i-1}\|_\infty\right) \\ &\leq \max\left(1 + \Delta t(\delta + 2\eta), \delta + 2\eta\right) \leq 1 + 3\Delta t^r. \end{aligned} \tag{13}$$

Therefore, using (11), we have

$$\left\| \frac{\partial \mathbf{X}_n}{\partial \mathbf{X}_k} \right\|_\infty \leq \prod_{k<i\leq n} \left\| \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_{i-1}} \right\|_\infty \leq (1 + 3\Delta t^r)^{n-k} \approx 1 + 3(n-k)\Delta t^r. \tag{14}$$

Note that we have used an expansion around Δt and neglected terms of $O(\Delta t^{2r})$ as $\Delta t << 1$. We remark that the bound (13) is the *crux of our argument* about gradient control as we see from the structure of the RNN that the recurrent matrices have close to unit norm. In order to complete the proof, one has to substitute the bound (14) in (10) and estimate the product (and the sum) carefully to obtain the desired bound (9). This is done in the detailed proof, presented in **SM**§C.3. As the entire gradient of the loss function (6), with respect to the weights and biases of the network, is bounded above in (9), the *exploding gradient problem* is mitigated for this RNN.

On the vanishing gradient problem. The vanishing gradient problem [33] arises if $\left|\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta}\right|$, defined in (10), $\to 0$ exponentially fast in k, for $k << n$ (long-term dependencies). In that case, the RNN does not have long-term memory, as the contribution of the k-th hidden state to error at time step t_n is infinitesimally small. We already see from (14) that $\left\|\frac{\partial \mathbf{X}_n}{\partial \mathbf{X}_k}\right\|_\infty \approx 1$ (independently of k). Thus, we should not expect the products in (10) to decay fast. In fact, we will provide a much more precise characterization of this gradient. To this end, we introduce the following *order*-notation,

$$\begin{aligned} &\beta = O(\alpha), \text{for } \alpha, \beta \in \mathbb{R}_+ \quad \text{if there exists constants } \overline{C}, \underline{C} \text{ such that } \underline{C}\alpha \leq \beta \leq \overline{C}\alpha. \\ &\mathbf{M} = O(\alpha), \text{for } \mathbf{M} \in \mathbb{R}^{d_1 \times d_2}, \alpha \in \mathbb{R}_+ \quad \text{if there exists constant } \overline{C} \text{ such that } \|\mathbf{M}\| \leq \overline{C}\alpha. \end{aligned} \tag{15}$$

For simplicity of notation, we will also set $\bar{\mathbf{y}}_n = \mathbf{u}_n \equiv 0$, for all n, $\mathbf{b} = 0$ and $r = 1$ in (8) and we will only consider $\theta = \mathbf{W}_{i,j}$ for some $1 \leq i, j \leq m$ in the following proposition.

Proposition 3.3. *Let* $\mathbf{y}_n$ *be the hidden states generated by the RNN* (4)*. Under the assumption that* $\mathbf{y}_n^i = O(\sqrt{t_n})$*, for all* $1 \leq i \leq m$ *and* (8)*, the gradient for long-term dependencies satisfies,*

$$\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta} = O\left(\hat{c}\delta\Delta t^{\frac{3}{2}}\right) + O\left(\hat{c}\delta\Delta t^{\frac{5}{2}}\right) + O(\Delta t^3), \text{ with } \hat{c} = sech^2\left(\sqrt{k\Delta t}(1+\Delta t)\right) \quad k << n, \tag{16}$$

This precise bound (16) on the gradient shows that although the gradient can be small i.e $O(\Delta t^{\frac{3}{2}})$, it is in fact *independent of* k, ensuring that long-term dependencies contribute to gradients at much later

steps and mitigating the vanishing gradient problem.
Sketch of proof. By an induction argument, detailed in **SM**§C.5, we can prove the following representation formula for products of Jacobians in (14):

$$\frac{\partial \mathbf{X}_n}{\partial \mathbf{X}_k} = \prod_{k<i\leq n} \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_{i-1}} = \begin{bmatrix} \mathrm{I} & \Delta t \sum_{j=k}^{n-1} \prod_{i=j}^{k} \mathbf{C}_i \\ \mathbf{B}_{n-1} + \sum_{j=n-2}^{k} \left(\prod_{i=n-1}^{j+1} \mathbf{C}_i \right) \mathbf{B}_j & \prod_{i=n-1}^{k} \mathbf{C}_i \end{bmatrix} + O(\Delta t) \tag{17}$$

Applying this representation formula in (4) results in,

$$\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta} = \mathbf{y}_n^\top \Delta t^2 \delta \mathbf{Z}_{m,d}^{i,j}(\mathbf{A}_{k-1}) \mathbf{y}_{k-1} + \mathbf{y}_n^\top \Delta t^2 \delta \mathbf{C}^* \mathbf{Z}_{m,d}^{i,j}(\mathbf{A}_{k-1}) \mathbf{y}_{k-1} + O(\Delta t^3), \tag{18}$$

with matrix $\mathbf{C}^*$ defined as,

$$\mathbf{C}^* := \sum_{j=k}^{n-1} \prod_{i=j}^{k} \mathbf{C}_i,$$

and $\mathbf{Z}_{m,d}^{i,j}(\mathbf{A}_{k-1}) \in \mathbb{R}^{m\times d}$ is a matrix with all elements are zero except for the (i,j)-th entry which is set to $\sigma'(\mathbf{A}_{k-1})_i$, i.e. the i-th entry of $\sigma'(\mathbf{A}_{k-1})$. It is straightforward to verify the bound (16) using definitions (12), assumption (8) and elementary but tedious calculations, detailed in **SM**§C.5.

4 Experiments

We test our proposed RNN architecture on a variety of different learning tasks, ranging from pure synthetic tasks designed to learn long-term dependencies (LTD) to more realistic tasks which also require high expressivity of the network. Details of the training procedure for each experiment can be found in **SM**§A. We wish to clarify here that we use a straightforward hyperparameter tuning protocol and do not use additional performance enhancing tools such as dropout [36], gradient clipping [33] or batch normalization [22], which might further improve the performance of coRNNs. Code to replicate the experiments can be found at *https://github.com/tk-rusch/coRNN*.

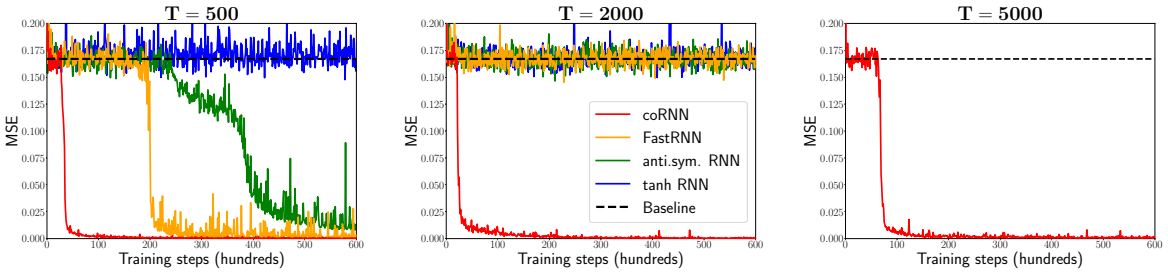

Figure 1: Results of the adding problem for the coRNN, FastRNN, anti.sym. RNN and tanh RNN based on three different sequence lengths T, i.e. $T = 500$, $T = 2000$ and $T = 5000$.

Adding problem. We start with the well-known adding problem, first proposed in [21], to test the ability of an RNN to learn (very) long-term dependencies. The input is a two-dimensional sequence of length T, with the first dimension consisting of random numbers drawn from $\mathcal{U}([0,1])$ and with two non-zero entries in the second dimension, both set to 1 and chosen at random, but one each in both halves of the sequence. The output is the sum of two numbers of the first dimension at the positions which are indicated by the two 1 entries in the second dimension. Thus the goal of this experiment is to beat the baseline output of 1, whose mean square error (MSE) equals the variance of 0.167. We compare our coRNN to two recently proposed RNNs, which were explicitly designed to learn LTDs, namely the

FastRNN [26] and the antisymmetric (anti.sym.) RNN [6]. To emphasize the challenging nature of this experiment, we also show the results of a plain-vanilla RNN with tanh activation. All methods have 128 hidden units as well as the same training protocol is used in all cases. Fig. 1 shows the results for different lengths T of the input sequences. We can see that while the tanh RNN is not able to beat the baseline for any sequence length, the FastRNN as well as the anti.sym. RNN successfully learn the adding task for $T = 500$. However, in this case, the coRNN converges significantly faster and reaches a lower test MSE than other tested methods. When setting the length to $T = 2000$, the difficulty of solving the adding problem increases considerably. In fact, most of recent publications only consider lengths of $T \leq 1000$. We can see that in this case i.e. $T = 2000$, only the coRNN solves the problem within a reasonable number of training steps. In order to further demonstrate the superiority of coRNN over recently proposed RNN architectures for learning LTDs, we consider the adding problem for $T = 5000$. Since all other methods failed for $T = 2000$, we only train the coRNN on this task. We can see that even in this case, the coRNN converges very quickly. We thus conclude that the coRNN mitigates the vanishing/exploding gradient problem for this example, even for very long sequences.

Table 1: Test accuracies on sMNIST and psMNIST (we provide our own psMNIST result for the FastGRNN, as no official result for this task has been published so far)

Model	sMNIST	psMNIST	# units	# params
uRNN [2]	95.1%	91.4%	512	9k
LSTM [11]	98.9%	90.2%	100	41k
GRU [7]	99.1%	94.1%	256	200k
anti.sym. RNN [6]	98.0 %	95.8%	128	10k
DTRIV∞ [5]	99.0%	96.8%	512	137k
FastGRNN [26]	98.7%	94.8%	128	18k
coRNN (128 units)	99.3%	96.6%	128	34k
coRNN (256 units)	**99.4%**	**97.34%**	256	134k

Sequential (permuted) MNIST. Sequential MNIST (sMNIST) [27] is a benchmark for RNNs, in which the model is required to classify an MNIST [28] digit one pixel at a time leading to a classification task with a sequence length of $T = 784$. In permuted sequential MNIST (psMNIST), a fixed random permutation is applied in order to increase the time-delay between interdependent pixels and to make the problem harder. In Table 1, we compare the test accuracy of the coRNN on sMNIST and psMNIST with recently published results for other recurrent models which were explicitly designed to solve long-term dependencies together with baselines corresponding to gated and unitary RNNs. To the best of our knowledge the proposed coRNN outperforms all single-layer recurrent architectures, published in the literature, for both the sMNIST and psMNIST. Moreover in Fig. 2, we present the performance (with respect to number of epochs) of different RNN architectures for psMNIST with the same fixed random perturbation and the same number of hidden units, i.e. 128. As seen from this figure, coRNN clearly outperforms the other architectures, some of which were explicitly designed to learn LTDs, handily for this perturbation.

Noise padded CIFAR10 Another challenging test problem for learning LTDs is the recently proposed noise-padded CIFAR10 experiment [6], in which CIFAR10 data points [25] are fed to the RNN row-wise and flattened along the channels resulting in sequences of length 32. To test the long term memory, entries of uniform random numbers are added such that the resulting sequences have a length of 1000, i.e. the last 968 entries of each sequences are only noise to distract the network. Table 2 shows the result for the coRNN together with other recently published results. We observe that coRNN readily outperforms the state-of-the-art on this benchmark, while requiring only 128 hidden units.

Our theoretical guarantees regarding the non-exploding/non-vanishing gradient depend on the weight assumptions (8), which need to be fulfilled throughout the whole training procedure. We check this

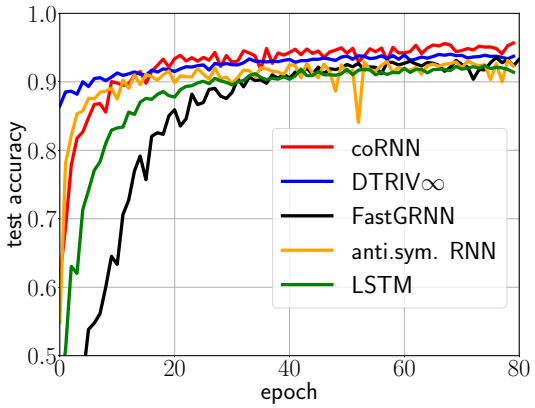

Figure 2: Performance on psMNIST for different models, all with 128 hidden units and the same fixed random permutation.

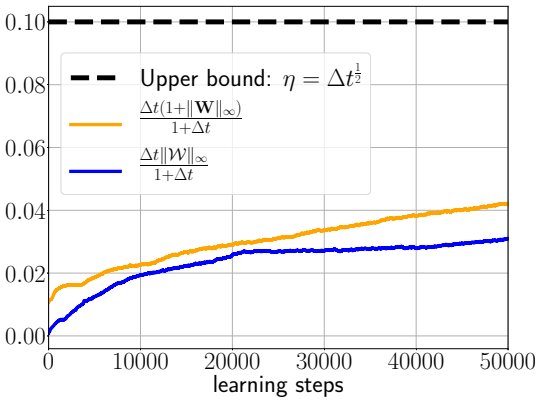

Figure 3: Weight assumptions (8) evaluated during training for the noise padded CIFAR10 experiment.

assumption for the noisy CIFAR10 experiment in Fig. 3, where we plot the relevant quantities on both sides of the inequality (8), with $r = \frac{1}{2}$. As seen from the figure, although the norms grow slightly during training, they are well below the needed bound, thus verifying (8) for this example. We also provide a theoretical argument for why this assumption (8) can be satisfied during training in **SM**§C.4.

Table 2: Test accuracies on noise padded CIFAR10

Model	test accuracy	# units	# params
LSTM [23]	11.6%	128	64k
Incremental RNN [23]	54.5%	128	12k
FastRNN [23]	45.8%	128	16k
anti.sym. RNN [6]	48.3%	256	36k
Gated anti.sym. RNN [6]	54.7 %	256	37k
Lipschitz RNN [14]	55.2%	256	134k
coRNN	**58.2**%	128	46k

Human activity recognition. This experiment is based on the human activity recognition data set [1]. The data set is a collection of tracked human activities, which were measured by an accelerometer and gyroscope on a Samsung Galaxy S3 smartphone. Six activities were binarized to obtain two merged classes {Sitting, Laying, Walking_Upstairs} and {Standing, Walking, Walking_Downstairs}, leading to the HAR-2 data set, which was first proposed in [26]. Table 3 shows the result of the coRNN together with other very recently published results on the same data set. We can see that the coRNN readily outperforms all other methods. We also ran this experiment on a *tiny coRNN* with very few parameters, i.e. only 1k. We can see that even in this case, the tiny coRNN beats all baselines. We thus conclude that the coRNN can efficiently be used on resource-constrained IoT micro-controllers.

IMDB sentiment analysis. The IMDB data set [31] is a collection of 50k movie reviews where 25k reviews are used for training (with 7.5k of these reviews used for validating) and 25k reviews are used for testing. The aim of this binary sentiment classification task is to decide whether a movie review is positive or negative. We use a dictionary size of 25k words and follow the standard procedure by initializing the word embedding with pretrained 100d GloVe [34] vectors. Table 4 shows the results for the coRNN and

Table 3: Test accuracies on HAR-2

Model	test accuracy	# units	# params
GRU [26]	93.6%	75	19k
LSTM [23]	93.7%	64	16k
FastRNN [26]	94.5%	80	7k
FastGRNN [26]	95.6%	80	7k
antisymmetric RNN [23]	93.2%	120	8k
incremental RNN [23]	96.3%	64	4k
coRNN	**97.2**%	64	9k
tiny coRNN	96.5%	20	1k

other recently published models which are trained similarly and have the same number of hidden units, i.e. 128. We can see that the coRNN compares favorable with gated baselines (which are known to perform very well on this task) while at the same time requiring significantly less parameters.

Table 4: Test accuracies on IMDB

Model	test accuracy	# units	# params
LSTM [4]	86.8%	128	220k
Skip LSTM[4]	86.6%	128	220k
GRU [4]	86.2%	128	164k
Skip GRU [4]	86.6%	128	164k
ReLU GRU [12]	84.8%	128	99k
coRNN	**87.4**%	128	46k

5 Discussion

Inspired by many models in physics, biology and engineering, particularly by circuits of biological neurons [37, 17], we proposed a novel RNN architecture (3) based on a model (1) of *coupled controlled forced and damped oscillators*. For this RNN, we rigorously showed that the hidden states are bounded (5) and obtained precise bounds on the gradients (Jacobians) of the hidden states, (9) and (16). Thus by design, this architecture is proved to mitigate the exploding and vanishing gradient problem (EVGP) and this is also verified in a series of numerical experiments. Furthermore, these experiments also demonstrate that the proposed RNN shows sufficient expressivity for performing complex tasks. In particular, the results showed that the proposed RNN was comparable to (or better than) other state of the art RNNs for a variety of tasks including sequential image classification, activity recognition and sentiment analysis. Moreover, the proposed RNN was able to show comparable performance to other RNNs with significantly fewer tuning parameters. Thus, we provide a novel and promising strategy for designing RNN architectures that are motivated by the functioning of biological neural networks, have rigorous bounds on hidden state gradients and are robust, accurate, straightforward to train and cheap to evaluate.

This work can be extended in many different directions. Our main theoretical focus in this paper was to demonstrate the mitigation of the exploding and vanishing gradient problem. On the other hand, we only provided some heuristics and numerical evidence on why the proposed RNN still has sufficient expressivity. A priori, it is natural to think that the proposed RNN architecture will introduce a strong bias towards oscillatory functions. However, we argue in **SM**§B, the proposed coRNN can be significantly more expressive as the damping, forcing and coupling of several oscillators modulates nonlinear response

to yield a very rich and diverse set of output states. This is also evidenced by the ability of the coRNN to be comparable to (and better than) the state of the art for all the presented numerical experiments, which do not have an explicit oscillatory structure. To further investigate the issue of expressivity, we aim to prove expressivity rigorously in the future by showing some sort of *universality* of the proposed coRNN architecture, as in the case of echo state networks in [16]. One possible approach would be to leverage the ability of the proposed RNN to convert general inputs into a rich set of superpositions of harmonics (oscillatory wave forms). One might then adapt the approach of Barron [3], where expressing functions in terms of superpositions of oscillatory functions (Fourier basis) was the key to universality results, to the context of the proposed RNN. Results on global dynamics of networks of oscillators, reviewed in [39] might be useful.

The proposed RNN was based on the simplest model of coupled oscillators (1). Much more detailed models of oscillators are available, particularly those that arise in the modeling of biological neurons, [37] and references therein. An interesting variant of our proposed RNN would be to base the RNN architecture on these more elaborate models, resulting in analogues of the spiking neurons model [32] for RNNs. These models might result in better expressivity than the proposed RNN, while still keeping the gradients under control. Another avenue of extension is to add gates to the proposed coRNN architecture, possibly improving expressivity further. Using first principles derivations of gated dynamics [40] would be instrumental for this task.

Acknowledgements.

The research of SM and TKR was partially supported by European Research Council Consolidator grant ERCCoG 770880: COMANFLO.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International Workshop on Ambient Assisted Living*, pages 216–223. Springer, 2012.

[2] M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

[3] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transcations on Information Theory*, 39, 1993.

[4] V. Campos, B. Jou, X. Giró-i-Nieto, J. Torres, and S. Chang. Skip RNN: learning to skip state updates in recurrent neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[5] M. L. Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems*, pages 9154–9164, 2019.

[6] B. Chang, M. Chen, E. Haber, and E. H. Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[7] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 77–87, 2017.

[8] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[9] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic recurrent neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[10] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[11] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. C. Courville. Recurrent batch normalization. In *5th International Conference on Learning Representations, ICLR*, 2017.

[12] R. Dey and F. M. Salemt. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[13] W. E. A proposal on machine learning via dynamical systems. *Commun. Math. Stat*, 5:1–11, 2017.

[14] N. B. Erichson, O. Azencot, A. Queiruga, and M. W. Mahoney. Lipschitz recurrent neural networks. *arXiv preprint arXiv:2006.12070*, 2020.

[15] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15379–15389, 2019.

[16] L. Grigoryeva and J.-P. Ortega. Echo state networks are universal. *Neural Networks*, 108:495 – 508, 2018. ISSN 0893-6080.

[17] B.-M. Gu, H. vanRijn, and W. K. Meck. Oscillatory multiplexing of neural population codes for interval timing and working memory. *Neuroscience and Behaviorial reviews,*, 48:160–185, 2015.

[18] J. Guckenheimer and P. Holmes. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*. Springer Verlag, New York, 1990.

[19] S. S. H. Sakaguchi and Y. Kuramoto. Local and global self-entrainment in oscillator lattices. *Progress of Theoretical Physics*, 77:1005–1010, 1987.

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

[21] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[23] A. Kag, Z. Zhang, and V. Saligrama. Rnns incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[24] G. Kerg, K. Goyette, M. P. Touzel, G. Gidel, E. Vorontsov, Y. Bengio, and G. Lajoie. Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics. In *Advances in Neural Information Processing Systems*, pages 13591–13601, 2019.

[25] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*, pages 9017–9028, 2018.

[27] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[30] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2018.

[31] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 142–150. Association for Computational Linguistics, 2011.

[32] W. Maass. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9:279–304, 2001.

[33] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28 of *ICML'13*, page III–1310–III–1318. JMLR.org, 2013.

[34] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[35] Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems 32*, pages 5320–5330. 2019.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

[37] K. M. Stiefel and G. B. Ermentrout. Neurons as oscillators. *Journal of Neurophysiology*, 116: 2950–2960, 2016.

[38] S. Strogatz. *Nonlinear Dynamics and Chaos*. Westview, Boulder CO, 2015.

[39] S. H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2001.

[40] C. Tallec and Y. Ollivier. Can recurrent networks warp time. In *International Conference on Learning Representations, ICLR*, 2018.

[41] A. T. Winfree. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of Theoretical Biology*, 16:15–42, 1967.

[42] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 4880–4888, 2016.

Supplementary Material for:
Coupled Oscillatory Recurrent Neural Network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies

A Training details

The IMDB task was conducted on an NVIDIA GeForce GTX 1080 Ti GPU, while all other experiments were run on a Intel Xeon E3-1585Lv5 CPU. The weights and biases of the coRNN are randomly initialized according to $\mathcal{U}(-\frac{1}{\sqrt{n_{in}}}, \frac{1}{\sqrt{n_{in}}})$, where n_{in} denotes the input dimension of each affine transformation. While the $\mathbf{z}_n$ equation in (3) can be solved implicitly or explicitly given the $\mathbf{z}_n$ control term, i.e. using $\Delta t \epsilon \mathbf{z}_n$ or $\Delta t \epsilon \mathbf{z}_{n-1}$, the presented results are based on the explicit form. However, we point out that no major difference in the results was obtained when using the implicit form instead of the explicit form. Additionally, instead of treating the parameters $\Delta t, \gamma$ and ϵ as fixed hyperparameters, we can also treat them as trainable network parameters by constraining Δt to $[0, 1]$ by using a sigmoidal activation function and $\epsilon, \gamma > 0$ by the use of ReLU for instance. However, also in this case no major performance difference is obtained. The hyperparameters are optimized with a random search algorithm, where the results of the best performing coRNN (based on the validation set) are reported. The ranges of the hyperparameters for the random search algorithm are provided in Table 5. Table 6 shows the rounded hyperparameters of the best performing coRNN architecture resulting from the random search algorithm for each learning task. We used 100 training epochs for the sMNIST and psMNIST problem with additional 20 epochs in which the learning rate was reduced by a factor of 10. Additionally, we used 100 epochs for the IMDB task and 250 epochs for all other experiments.

Table 5: Setting for the hyperparameter optimization of the coRNN. Intervals denote ranges of the corresponding hyperparameter for the grid search algorithm, while fixed numbers mean that no hyperparameter optimization was done in this case.

task	learning rate	batch size	Δt	γ	ϵ
Adding	2×10^{-2}	50	$[10^{-2}, 10^{-1}]$	$[1, 100]$	$[1, 100]$
sMNIST ($n_{hid} = 128$)	$[10^{-4}, 10^{-1}]$	120	$[10^{-2}, 10^{-1}]$	$[10^{-1}, 10]$	$[10^{-1}, 10]$
sMNIST ($n_{hid} = 256$)	$[10^{-4}, 10^{-1}]$	120	$[10^{-2}, 10^{-1}]$	$[10^{-1}, 10]$	$[10^{-1}, 10]$
psMNIST ($n_{hid} = 128$)	$[10^{-4}, 10^{-1}]$	120	$[10^{-2}, 10^{-1}]$	$[10^{-1}, 10]$	$[10^{-1}, 10]$
psMNIST ($n_{hid} = 256$)	$[10^{-4}, 10^{-1}]$	120	$[10^{-2}, 10^{-1}]$	$[10^{-1}, 10]$	$[10^{-1}, 10]$
Noisy CIFAR10	$[10^{-4}, 10^{-1}]$	100	$[10^{-2}, 10^{-1}]$	$[1, 100]$	$[1, 100]$
HAR-2	$[10^{-4}, 10^{-1}]$	64	$[10^{-2}, 10^{-1}]$	$[10^{-1}, 10]$	$[10^{-1}, 10]$
IMDB	$[10^{-4}, 10^{-1}]$	64	$[10^{-2}, 10^{-1}]$	$[10^{-1}, 10]$	$[10^{-1}, 10]$

Table 6: Rounded hyperparameters of the best performing coRNN architecture.

task	learning rate	batch size	Δt	γ	ϵ
Adding ($T = 5000$)	2×10^{-2}	50	1.6×10^{-2}	94.5	9.5
sMNIST ($n_{hid} = 128$)	3.5×10^{-3}	120	5.3×10^{-2}	1.7	4
sMNIST ($n_{hid} = 256$)	2.1×10^{-3}	120	4.2×10^{-2}	2.7	4.7
psMNIST ($n_{hid} = 128$)	3.7×10^{-3}	120	8.3×10^{-2}	1.3×10^{-1}	4.1
psMNIST ($n_{hid} = 256$)	5.4×10^{-3}	120	7.6×10^{-2}	4×10^{-1}	8.0
Noisy CIFAR10	7.5×10^{-3}	100	3.4×10^{-2}	1.3	12.7
HAR-2	1.7×10^{-2}	64	10^{-1}	2×10^{-1}	6.4
IMDB	6.0×10^{-4}	64	5.4×10^{-2}	4.9	4.8

B Heuristics of network function

To see that the RNN (3) models a *coupled network of controlled forced, damped nonlinear oscillators*, we start with the single neuron (scalar) case by setting $d = m = 1$ in (1) and assume an identity activation function $\sigma(x) = x$. Setting $\mathbf{W} = \mathcal{W} = \mathbf{V} = \mathbf{b} = \epsilon = 0$ leads to the simple ODE, $\mathbf{y}'' + \gamma\mathbf{y} = 0$, which exactly models *simple harmonic motion*, for instance that of a mass attached to a spring [18]. Letting $\epsilon > 0$ in (1) adds *damping* or friction to the system [18]. Then, by introducing non-zero $\mathbf{V}$ in (1), we drive the system with a driving force proportional to the input signal $\mathbf{u}(t)$. The parameters $\mathbf{V}, \mathbf{b}$ modulate the effect of the driving force, $\mathbf{W}$ controls the frequency of oscillations and $\mathcal{W}$ the amount of damping in the system. Finally, the tanh activation mediates a non-linear response in the oscillator. This picture can be readily generalized, when the full network is considered. Then, each neuron updates its hidden state based on the input signal as well as information from other neurons. The diagonal entries of $\mathbf{W}$ and $\mathcal{W}$ control the frequency and amount of damping for each neuron, respectively, whereas the non-diagonal entries of these matrices modulate interactions between neurons.

At the level of a single neuron, the dynamics of the RNN is relatively straightforward. We start with the scalar case i.e. $m = d = 1$ and illustrate different hidden states $\mathbf{y}$ as a function of time, for different input signals, in Fig. 4. In this figure, we consider two different input signals, one oscillatory signal given by $\mathbf{u}(t) = \cos(4t)$ and another is a combination of step functions. First, we plot the solution $\mathbf{y}(t)$ of (1), with the parameters $\mathbf{V}, \mathbf{b}, \mathbf{W}, \mathcal{W}, \epsilon = 0$ and $\gamma = 1$. This simply corresponds to the case of a simple harmonic oscillator (SHO) and the solution is described by a sine wave with the natural frequency of the oscillator. Next, we introduce forcing by the input signal by setting $\mathbf{V} = 1$ and the activation function is the identity $\sigma(x) = x$, leading to a forced damped oscillator (FDO). As seen from Fig. 4, in the case of an oscillatory signal, this leads to a very minor change over the SHO, whereas for the step function, the change is only in the amplitude of the wave. Next, we add damping by setting $\epsilon = 0.25$ and see that the resulting forced damped oscillator (FDO), merely damps the amplitude of the waves, without changing their frequency. Then, we consider the case of *controlled oscillator* (CFDO) by setting $\mathbf{W} = -2, \mathbf{V} = 2, \mathbf{b} = 0.25, \mathcal{W} = 0.75$. As seen from Fig. 4, this leads to a significant change in the wave form in both cases. For the oscillatory input, the output is now a superposition of many different forms, with different amplitudes and frequencies (phases) whereas for the step function input, the phase is shifted. Already, we can see that for a linear controlled oscillator, the output can be very complicated with the superposition of different waves. This holds true when the activation function is set to $\sigma(x) = \tanh(x)$ (which is our proposed coRNN). For both inputs, the output is a modulated version of the one generated by CFDO, expressed as a superposition of waves. On the other hand, we also plot the solution with a Duffing type oscillator (DUFF) by setting the activation function as,

$$\sigma(x) = x - \frac{x^3}{3}. \tag{19}$$

In this case, the solution is very different from the CFDO and coRNN solutions and is heavily damped (either in the output or its derivative). On the other hand, given the chaotic nature of the dynamical system in this case, a slight change in the parameters led to the output blowing up. Thus, a bounded nonlinearity seems essential in this context.

Coupling neurons together further accentuates this generation of superpositions of different wave-forms, as seen even with the simplest case of a network with two neurons, shown in Fig. 4 (Bottom row). For this figure, we consider two neurons i.e $m = 2$ and two different network topologies. For the first, we only allow the first neuron to influence the second one and not vice versa. This is enforced with the weight matrices,

$$\mathbf{W} = \begin{bmatrix} -2 & 0 \\ 3 & -2 \end{bmatrix}, \quad \mathcal{W} = \begin{bmatrix} 0.75 & 0 \\ -1 & 0.75 \end{bmatrix}.$$

We also set $\mathbf{V} = [2, 2]^\top, \mathbf{b} = [0.25, 0.25]^\top$. Note that in this case (we name as ORD (for ordered connections)), the output of the first neuron should be exactly as the same as in the uncoupled (UC) case, whereas there is a distinct change in the output of the second neuron and we see that the first neuron has modulated a sharp change in the resulting output wave form. It is well illustrated by the emergence of an approximation to the step function (Bottom Right of Fig. 4), even though the input signal is oscillatory.

Next, we consider the case of fully connected (FC) neurons by setting the weight matrices as,

$$\mathbf{W} = \begin{bmatrix} -2 & 1 \\ 3 & -2 \end{bmatrix}, \quad \mathcal{W} = \begin{bmatrix} 0.75 & 0.3 \\ -1 & 0.75 \end{bmatrix}$$

The resulting outputs for the first neuron are now slightly different from the uncoupled case. On the the other hand, the approximation of step function output for the second neuron is further accentuated.

Even these simple examples illustrate the functioning of a network of controlled oscillators well. The input signal is converted into a superposition of waves with different frequencies and amplitudes, with these quantities being controlled by the weights and biases in (1). Thus, very complicated outputs can be generated by modulating the number, frequencies and amplitudes of the waves. In practice, a network of a large number of neurons is used and can lead to extremely rich global dynamics, along the lines of emergence of synchronization or bistable heterogeneous behavior seen in systems of idealized oscillators and explained by their mean field limit, see [19, 41, 39]. Thus, we argue that the ability of the network of (forced, driven) oscillators to access a very rich set of output states can lead to high expressivity of the system. The training process selects the weights that modulate frequencies, phases and amplitudes of individual neurons and their interaction to guide the system to its target output.

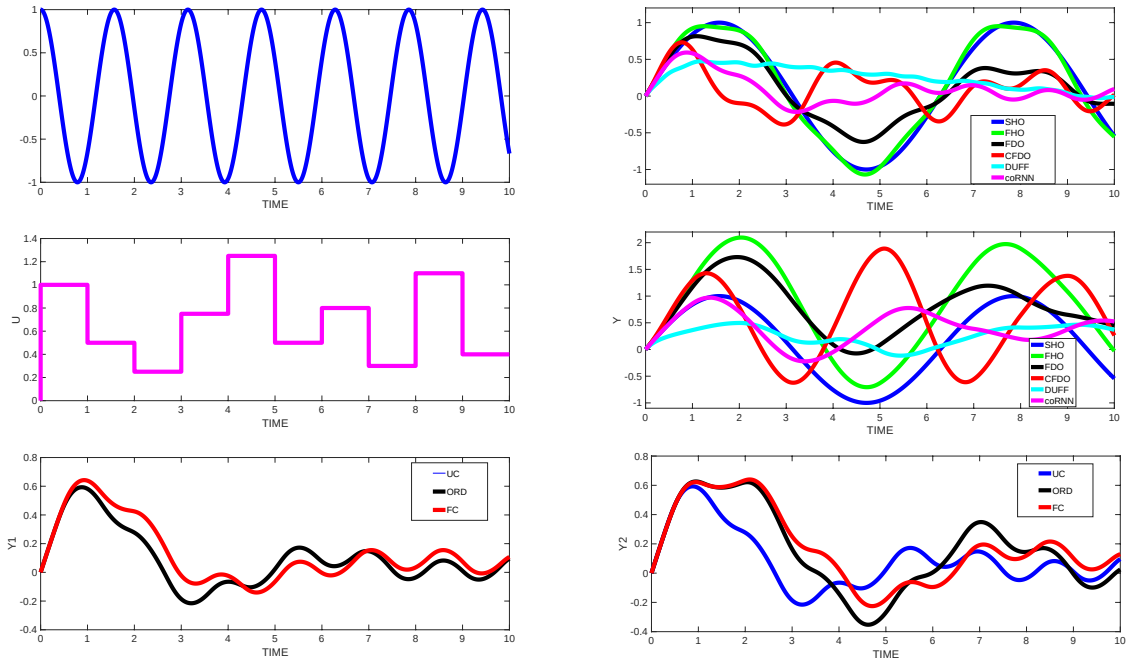

Figure 4: Illustration of the hidden state $\mathbf{y}$ of the coRNN (3) with a scalar input signal $\mathbf{u}$ (Top, Middle, Left) with one neuron with state $\mathbf{y}$ (Top and Middle, Right) and two neurons with states $\mathbf{y}_1$ (Bottom left), and $\mathbf{y}_2$ (Bottom right), corresponding to scalar input signal, shown in Top Left. Legend is SHO (simple harmonic oscillator), FHO (forced oscillator), FDO (forced and damped oscillator), CFDO (controlled forced and damped oscillator), DUFF (Duffing type) UC (Uncoupled), Ord (ordered coupling) and FC (fully coupled). Legend explained in the text

C Supplement to the rigorous analysis of the coRNN

In this section, we supplement the section on rigorous analysis of the proposed RNN (4). We start with

C.1 Proof of Proposition 3.1

We multiply $(\mathbf{y}_{n-1}^\top, \mathbf{z}_n^\top)$ to (3) and use the elementary identities,

$$\mathbf{a}^\top(\mathbf{a}-\mathbf{b}) = \frac{\mathbf{a}^\top\mathbf{a}}{2} - \frac{\mathbf{b}^\top\mathbf{b}}{2} + \frac{1}{2}(\mathbf{a}-\mathbf{b})^\top(\mathbf{a}-\mathbf{b}), \quad \mathbf{b}^\top(\mathbf{a}-\mathbf{b}) = \frac{\mathbf{a}^\top\mathbf{a}}{2} - \frac{\mathbf{b}^\top\mathbf{b}}{2} - \frac{1}{2}(\mathbf{a}-\mathbf{b})^\top(\mathbf{a}-\mathbf{b})$$

to obtain the following,

$$\begin{aligned}
\frac{\mathbf{y}_n^\top\mathbf{y}_n + \mathbf{z}_n^\top\mathbf{z}_n}{2} &= \frac{\mathbf{y}_{n-1}^\top\mathbf{y}_{n-1} + \mathbf{z}_{n-1}^\top\mathbf{z}_{n-1}}{2} + \frac{(\mathbf{y}_n - \mathbf{y}_{n-1})^\top(\mathbf{y}_n - \mathbf{y}_{n-1})}{2} \\
&\quad - \frac{(\mathbf{z}_n - \mathbf{z}_{n-1})^\top(\mathbf{z}_n - \mathbf{z}_{n-1})}{2} + \Delta t\mathbf{z}_n^\top\sigma(\mathbf{A}_{n-1}) - \Delta t\mathbf{z}_n^\top\mathbf{z}_n \\
&\leq \frac{\mathbf{y}_{n-1}^\top\mathbf{y}_{n-1} + \mathbf{z}_{n-1}^\top\mathbf{z}_{n-1}}{2} + \Delta t\,(1/2 + \Delta t/2 - 1)\,\mathbf{z}_n^\top\mathbf{z}_n + \frac{\Delta t}{2}\sigma^\top(\mathbf{A}_{n-1})\sigma(\mathbf{A}_{n-1}) \\
&\leq \frac{\mathbf{y}_{n-1}^\top\mathbf{y}_{n-1} + \mathbf{z}_{n-1}^\top\mathbf{z}_{n-1}}{2} + \frac{m\Delta t}{2} \quad \text{as } \sigma^2 \leq 1 \text{ and } \epsilon > \Delta t << 1.
\end{aligned}$$

Iterating the above inequality n leads to the energy bound,

$$\mathbf{y}_n^\top\mathbf{y}_n + \mathbf{z}_n^\top\mathbf{z}_n \leq \mathbf{y}_0^\top\mathbf{y}_0 + \mathbf{z}_0^\top\mathbf{z}_0 + nm\Delta t = mt_n, \tag{20}$$

as $\mathbf{y}_0 = \mathbf{z}_0 = 0$.

C.2 Sensitivity to inputs

Next, we examine how changes in the input signal $\mathbf{u}$ affect the dynamics, we have the following proposition:

Proposition C.1. *Let $\mathbf{y}_n, \mathbf{z}_n$ be the hidden states of the trained RNN (4) with respect to the input $\mathbf{u} = \{\mathbf{u}_n\}_{n=1}^N$ and let $\overline{\mathbf{y}}_n, \overline{\mathbf{z}}_n$ be the hidden states of the same RNN (4), but with respect to the input $\overline{\mathbf{u}} = \{\overline{\mathbf{u}}_n\}_{n=1}^N$, then the differences in the hidden states are bounded by,*

$$(\mathbf{y}_n - \overline{\mathbf{y}}_n)^\top(\mathbf{y}_n - \overline{\mathbf{y}}_n) + (\mathbf{z}_n - \overline{\mathbf{z}}_n)^\top(\mathbf{z}_n - \overline{\mathbf{z}}_n) \leq 2mt_n. \tag{21}$$

The proof of this proposition is completely analogous to the proof of proposition 3.1, we subtract

$$\begin{aligned}
\overline{\mathbf{y}}_n &= \overline{\mathbf{y}}_{n-1} + \Delta t\overline{\mathbf{z}}_n, \\
\overline{\mathbf{z}}_n &= \tfrac{\overline{\mathbf{z}}_{n-1}}{1+\Delta t} + \tfrac{\Delta t}{1+\Delta t}\sigma(\overline{\mathbf{A}}_{n-1}) - \tfrac{\Delta t}{1+\Delta t}\overline{\mathbf{y}}_{n-1}, \ \overline{\mathbf{A}}_{n-1} := \mathbf{W}\overline{\mathbf{y}}_{n-1} + \mathcal{W}\overline{\mathbf{z}}_{n-1} + \mathbf{V}\overline{\mathbf{u}}_n + \mathbf{b}.
\end{aligned} \tag{22}$$

from (4) and multiply $\left((\mathbf{y}_n - \overline{\mathbf{y}}_n)^\top, (\mathbf{z}_n - \overline{\mathbf{z}}_n)^\top\right)$ to the difference. The estimate (21) follows identically to the proof of (5) (presented above) by realizing that $\sigma(\mathbf{A}_{n-1}) - \sigma(\overline{\mathbf{A}}_{n-1}) \leq 2$.

Note that the bound (21) ensures that the hidden states can only separate linearly in time for changing in input. Thus, chaotic behavior, such as for Duffing type oscillators, characterized by at least exponential separation of trajectories, is ruled out for this proposed RNN, showing that it is stable with respect to changes in input. This is largely on account of the fact that the activation function σ in (3) is globally bounded.

C.3 Proof of Proposition 3.2

From (6), we readily calculate that,

$$\frac{\partial\mathcal{E}_n}{\partial\mathbf{X}_n} = [\mathbf{y}_n - \bar{\mathbf{y}}_n, 0]\,. \tag{23}$$

Similarly from (3), we calculate,

$$\frac{\partial^+ \mathbf{X}_k}{\partial \theta} = \begin{cases} \left[\left(\frac{\Delta t^2}{1+\epsilon\Delta t}\mathbf{Z}^{i,j}_{m,m}(\mathbf{A}_{k-1})\mathbf{y}_{k-1}\right)^\perp, \left(\frac{\Delta t}{1+\epsilon\Delta t}\mathbf{Z}^{i,j}_{m,m}(\mathbf{A}_{k-1})\mathbf{y}_{k-1}\right)^\perp\right]^\perp & \text{if} \quad \theta = (i,j)\text{−th entry of } \mathbf{W}, \\ \left[\left(\frac{\Delta t^2}{1+\epsilon\Delta t}\mathbf{Z}^{i,j}_{m,m}(\mathbf{A}_{k-1})\mathbf{z}_{k-1}\right)^\perp, \left(\frac{\Delta t}{1+\epsilon\Delta t}\mathbf{Z}^{i,j}_{m,m}(\mathbf{A}_{k-1})\mathbf{z}_{k-1}\right)^\perp\right]^\perp & \text{if} \quad \theta = (i,j)\text{−th entry of } \mathcal{W}, \\ \left[\left(\frac{\Delta t^2}{1+\epsilon\Delta t}\mathbf{Z}^{i,j}_{m,d}(\mathbf{A}_{k-1})\mathbf{u}_k\right)^\perp, \left(\frac{\Delta t}{1+\epsilon\Delta t}\mathbf{Z}^{i,j}_{m,d}(\mathbf{A}_{k-1})\mathbf{u}_k\right)^\perp\right]^\perp & \text{if} \quad \theta = (i,j)\text{−th entry of } \mathbf{V}, \\ \left[\left(\frac{\Delta t^2}{1+\epsilon\Delta t}\mathbf{Z}^{i,1}_{m,1}(\mathbf{A}_{k-1})\right)^\perp, \left(\frac{\Delta t}{1+\epsilon\Delta t}\mathbf{Z}^{i,1}_{m,1}(\mathbf{A}_{k-1})\right)^\perp\right]^\perp & \text{if} \quad \theta = i\text{−th entry of } \mathbf{b}, \end{cases} \tag{24}$$

where $\mathbf{Z}^{i,j}_{m,d}(\mathbf{A}_{k-1}) \in \mathbb{R}^{m\times d}$ is a matrix with all elements are zero except for the (i,j)-th entry which is set to $\sigma'(\mathbf{A}_{k-1})_i$, i.e. the i-th entry of $\sigma'(\mathbf{A}_{k-1})$. We easily see that $\|\mathbf{Z}^{i,j}_{m,d}(\mathbf{A}_{k-1})\|_\infty \le 1$ for all i,j,d,m and all choices of $\mathbf{A}_{k-1}$.

Now, using definitions of matrix and vector norms and applying (14) in (10), together with (23) and (24), we obtain the following estimate on the norm:

$$\left|\frac{\partial \mathcal{E}^{(k)}_n}{\partial \theta}\right| \le \begin{cases} (\|\mathbf{y}_n\|_\infty + \|\bar{\mathbf{y}}_n\|_\infty)(1+3(n-k)\Delta t^r)\delta\Delta t\|\mathbf{y}_{k-1}\|_\infty, & \text{if} \quad \theta \text{ is entry of } \mathbf{W}, \\ (\|\mathbf{y}_n\|_\infty + \|\bar{\mathbf{y}}_n\|_\infty)(1+3(n-k)\Delta t^r)\delta\Delta t\|\mathbf{z}_{k-1}\|_\infty, & \text{if} \quad \theta \text{ is entry of } \mathcal{W}, \\ (\|\mathbf{y}_n\|_\infty + \|\bar{\mathbf{y}}_n\|_\infty)(1+3(n-k)\Delta t^r)\delta\Delta t\|\mathbf{u}_k\|_\infty, & \text{if} \quad \theta \text{ is entry of } \mathbf{V}, \\ (\|\mathbf{y}_n\|_\infty + \|\bar{\mathbf{y}}_n\|_\infty)(1+3(n-k)\Delta t^r)\delta\Delta t, & \text{if} \quad \theta \text{ is entry of } \mathbf{b}, \end{cases} \tag{25}$$

We will estimate the above term, just for the case of θ is an entry of $\mathbf{W}$, the rest of the terms are very similar to estimate.

As $\bar{\mathbf{y}}$ is the ground truth, we assume that it is bounded and can neglect it in the estimate. Also for simplicity of notation, we let $k-1 \approx k$ and aim to estimate the term,

$$\begin{aligned} \left|\frac{\partial \mathcal{E}^{(k)}_n}{\partial \theta}\right| &\le \|\mathbf{y}_n\|_\infty \|\mathbf{y}_k\|_\infty (1+3(n-k)\Delta t^r)\delta\Delta t \\ &\le m\sqrt{nk}\Delta t(1+3(n-k)\Delta t^r)\delta\Delta t \quad \text{by (5)} \\ &\le m\sqrt{nk}\delta\Delta t^2 + 3m\sqrt{nk}(n-k)\delta\Delta t^{r+2}. \end{aligned} \tag{26}$$

To further analyze the above estimate, we assume that $n\Delta t = t_n \le 1$ and consider two different regimes. Let us start by considering *short-term dependencies* by letting $k \approx n$, i.e $n-k=c$ with constant $c \sim O(1)$, independent of n,k . In this case, a straightforward application of the above assumptions in the bound (26) yields,

$$\begin{aligned} \left|\frac{\partial \mathcal{E}^{(k)}_n}{\partial \theta}\right| &\le m\sqrt{nk}\delta\Delta t^2 + 3m\sqrt{nk}(n-k)\delta\Delta t^{r+2} \\ &\le mt_n\delta\Delta t + mct_n\delta\Delta t^{r+1} \\ &\le Ct_nm\delta\Delta t \quad (\text{as } r \ge 1/2) \\ &\le Cm\delta\Delta t, \end{aligned} \tag{27}$$

for a constant C independent of n,k.

Next, we consider *long-term dependencies* by setting $k << n$ and estimating,

$$\begin{aligned} \left|\frac{\partial \mathcal{E}^{(k)}_n}{\partial \theta}\right| &\le m\sqrt{nk}\delta\Delta t^2 + 3m\sqrt{nk}(n-k)\delta\Delta t^{r+2} \\ &\le m\sqrt{t_n}\delta\Delta t^{\frac{3}{2}} + 3mt_n^{\frac{3}{2}}\delta\Delta t^{r+\frac{1}{2}} \\ &\le m\delta\Delta t^{\frac{3}{2}} + 3m\delta\Delta t^{r+\frac{1}{2}} \quad (\text{as } t_n < 1) \\ &\le Cm\delta\Delta t^{r+\frac{1}{2}} \quad (\text{as } r \le 1). \end{aligned} \tag{28}$$

Thus, in all case, we have that,

$$\left|\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta}\right| \leq Cm\delta\Delta t \quad (\text{as } r \geq 1/2). \tag{29}$$

Applying the above estimate in (10) allows us to bound the gradient by,

$$\left|\frac{\partial \mathcal{E}_n}{\partial \theta}\right| \leq \sum_{1\leq k\leq n}\left|\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta}\right| \leq Cnm\delta\Delta t = Cm\delta t_n. \tag{30}$$

Therefore, the gradient of the loss function (6) can be bounded as,

$$\left|\frac{\partial \mathcal{E}}{\partial \theta}\right| \leq \frac{1}{N}\sum_n\left|\frac{\partial \mathcal{E}_n}{\partial \theta}\right| \leq \frac{Cm\delta}{N}\sum_{n=1}^{N} t_n = \frac{m\delta\Delta t}{N}\sum_{n=1}^{N} n \leq Cm\delta N\Delta t = Cm\delta, \tag{31}$$

which is the desired estimate (9).

C.4 On the assumption (8) and training

Note that all the estimates were based on the fact that we were able to choose a time step Δt in (3) that enforces the condition (8). For any fixed weights $\mathbf{W}, \mathcal{W}$, we can indeed choose such a value of ϵ to satisfy (8). However, we *train* the RNN to find the weights that minimize the loss function (6). Can we find a hyperparameter Δt such that (8) is satisfied at every step of the stochastic gradient descent method for training?

To investigate this issue, we consider a simple gradient descent method of the form:

$$\theta_{\ell+1} = \theta_\ell - \zeta\frac{\partial \mathcal{E}}{\partial \theta}(\theta_\ell). \tag{32}$$

Note that ζ is the constant (non-adapted) learning rate. We assume for simplicity that $\theta_0 = 0$ (other choices lead to the addition of a constant). Then, a straightforward estimate on the weight is given by,

$$\begin{aligned} |\theta_{\ell+1}| &\leq |\theta_\ell| + \zeta\left|\frac{\partial \mathcal{E}}{\partial \theta}(\theta_\ell)\right| \\ &\leq |\theta_\ell| + C\zeta m\delta \quad \text{by (31)}, \\ &\leq |\theta_0| + C\ell\zeta m\delta = C\ell\zeta m\delta. \end{aligned} \tag{33}$$

In order to calculate the minimum number of steps L in the gradient descent method (33) such that the condition (8), we set $\ell = L$ in (33) and applying it to the condition (8) leads to the straightforward estimate,

$$L \geq \frac{1}{C\zeta m^2\Delta t^{1-r}\delta^2} \tag{34}$$

Note that the constant $C \sim O(1)$ and the parameter $\delta < 1$, while in general, the learning rate $\zeta << 1$. Thus, as long as $r \leq 1$, we see that the assumption (8) holds for a very large number of steps of the gradient descent method. We remark that the above estimate (34) is a large underestimate on L. In the experiments presented in this article, we are able to take a very large number of training steps, while the gradients remain within a range (see Fig. 3).

C.5 Proof of Proposition 3.3

We start with the following decomposition of the recurrent matrices:

$$\begin{aligned} &\frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_{i-1}} = M_{i-1} + \Delta t\tilde{M}_{i-1}, \\ &M_{i-1} := \begin{bmatrix} \mathrm{I} & \Delta t\mathbf{C}_{i-1} \\ \mathbf{B}_{i-1} & \mathbf{C}_{i-1} \end{bmatrix}, \quad \tilde{M}_{i-1} := \begin{bmatrix} \mathbf{B}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \end{aligned}$$

with $\mathbf{B}, \mathbf{C}$ defined in (12). By the assumption (8), one can readily check that $\|\tilde{M}_{i-1}\|_\infty \le \Delta t$, for all $k \le i \le n-1$.

We will use an induction argument to show the representation formula (17). We start by the outermost product and calculate,

$$\begin{aligned}\frac{\partial \mathbf{X}_n}{\partial \mathbf{X}_{n-1}}\frac{\partial \mathbf{X}_{n-1}}{\partial \mathbf{X}_{n-2}} &= \left(M_{n-1} + \Delta t \tilde{M}_{n-1}\right)\left(M_{n-2} + \Delta t \tilde{M}_{n-2}\right) \\ &= M_{n-1}M_{n-2} + \Delta t(\tilde{M}_{n-1}M_{n-2} + M_{n-1}\tilde{M}_{n-2}) + O(\Delta t^2).\end{aligned}$$

By direct multiplication in (12), we obtain,

$$\begin{aligned}M_{n-1}M_{n-2} = &\begin{bmatrix} \mathrm{I} & \Delta t\,(\mathbf{C}_{n-2} + \mathbf{C}_{n-1}\mathbf{C}_{n-2}) \\ \mathbf{B}_{n-1} + \mathbf{C}_{n-1}\mathbf{B}_{n-2} & \mathbf{C}_{n-1}\mathbf{C}_{n-2}\end{bmatrix} \\ &+ \Delta t \begin{bmatrix} \mathbf{C}_{n-1}\mathbf{B}_{n-2} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{n-1}\mathbf{C}_{n-2}\end{bmatrix}\end{aligned}$$

Using the definitions in (12) and (8), we can easily see that

$$\begin{bmatrix} \mathbf{C}_{n-1}\mathbf{B}_{n-2} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{n-1}\mathbf{C}_{n-2}\end{bmatrix} = O(\Delta t).$$

Similarly, it is easy to show that

$$\tilde{M}_{n-1}M_{n-2}, M_{n-1}\tilde{M}_{n-2} \sim O(\Delta t).$$

Plugging all the above estimates yields,

$$\frac{\partial \mathbf{X}_n}{\partial \mathbf{X}_{n-1}}\frac{\partial \mathbf{X}_{n-1}}{\partial \mathbf{X}_{n-2}} = \begin{bmatrix} \mathrm{I} & \Delta t\,(\mathbf{C}_{n-2} + \mathbf{C}_{n-1}\mathbf{C}_{n-2}) \\ \mathbf{B}_{n-1} + \mathbf{C}_{n-1}\mathbf{B}_{n-2} & \mathbf{C}_{n-1}\mathbf{C}_{n-2}\end{bmatrix} + O(\Delta t^2),$$

which is exactly the form of the leading term (17).

Iterating the above calculations $(n-k)$ times and realizing that $(n-k)\Delta t^2 \approx n\Delta t^2 = t_n \Delta t$ yields the formula (17).

Recall that we have set $\theta = \mathbf{W}_{i,j}$, for some $1 \le i, j \le m$ in proposition 3.3. Directly calculating with (23), (24) and the representation formula (17) yields the formula (18), which can be explicitly written as,

$$\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta} = \delta \Delta t^2 \sigma'(a_{k-1}^i)\mathbf{y}_n^j \mathbf{y}_{k-1}^j + \delta \Delta t^2 \sigma'(a_{k-1}^i)\sum_{\ell=1}^m \mathbf{C}_{\ell j}^* \mathbf{y}_n^j \mathbf{y}_{k-1}^j + O(\Delta t^3), \tag{35}$$

with $\mathbf{y}_n^j$ denoting the j-th element of vector $\mathbf{y}_n$, the matrix $\mathbf{C}^*$, defined in (18) and

$$a_{k-1}^i := \sum_{\ell=1}^m \mathbf{W}_{i\ell}\mathbf{y}_{k-1}^\ell + \sum_{\ell=1}^m \mathcal{W}_{i\ell}\mathbf{z}_{k-1}^\ell \tag{36}$$

By the assumption (8), we can readily see that

$$\|\mathbf{W}\|_\infty, \|\mathcal{W}\|_\infty \le 1 + \Delta t$$

Therefore by the fact that $\sigma' = sech^2$, the assumption $\mathbf{y}_k^i = O(\sqrt{t_k})$ and (36), we obtain,

$$\hat{c} = sech^2(\sqrt{k\Delta t}(1+\Delta t) \le \sigma'(a_i^{k-1}) \le 1. \tag{37}$$

Using (37) in (35), we obtain,

$$\delta \Delta t^2 \sigma'(a_{k-1}^i)\mathbf{y}_n^j \mathbf{y}_{k-1}^j = O\left(\hat{c}\delta \Delta t^{\frac{5}{2}}\right) \tag{38}$$

By definition,

$$\mathbf{C}^* := \sum_{j=k}^{n-1}\prod_{i=j}^{k}\mathbf{C}_i.$$

It is easy to see from the definition of $\mathbf{C}_i$ (12) that

$$\prod_{i=j}^{k} \mathbf{C}_i = \delta^{j-k+1}\mathrm{I} + O(\delta^{j-k+1}\Delta t^{j-k+1}).$$

Summing over i and using the fact that $k << n$, we obtain that

$$\mathbf{C}^* = \frac{\delta}{1-\delta}\mathrm{I} + O(\Delta t) = \frac{1}{\Delta t}\mathrm{I} + O(\Delta t). \tag{39}$$

Plugging (39) and (37) into (35) leads to,

$$\delta\Delta t^2 \sigma'(a_{k-1}^i) \sum_{\ell=1}^{m} \mathbf{C}_{\ell j}^* \mathbf{y}_n^j \mathbf{y}_{k-1}^j = O\left(\hat{c}\delta\Delta t^{\frac{3}{2}}\right) + O(\Delta t^3) \tag{40}$$

Combining (38) and (40) yields the desired estimate (16).