Supplementary Material for ”UnICORNN: A recurrent model for learning *very* long time dependencies”

A. Training details

All experiments were run on GPU, namely NVIDIA GeForce GTX 1080 Ti and NVIDIA GeForce RTX 2080 Ti. The hidden weights $\mathbf{w}$ of the UnICORNN are initialized according to $\mathcal{U}(0, 1)$, while all biases are set to zero. The trained vector $\mathbf{c}$ is initialized according to $\mathcal{U}(-0.1, 0.1)$. The input weights $\mathbf{V}$ are initialized according to the Kaiming uniform initialization (He et al., 2015) based on the input dimension mode and the negative slope of the rectifier set to $a = 8$.

The hyperparameters of the UnICORNN are selected using a random search algorithm based on a validation set. The hyperparameters of the best performing UnICORNN can be seen in Table 1. The value for Δt and α is shared across all layers, except for the IMDB task and EigenWorms task, where we use a different Δt value for the first layer and the corresponding Δt value in Table 1 for all subsequent layers, i.e. we use $\Delta t = 6.6 \times 10^{-3}$ for IMDB and $\Delta t = 2.81 \times 10^{-5}$ for EigenWorms in the first layer. Additionally, the dropout column corresponds to variational dropout (Gal & Ghahramani, 2016), which is applied after each consecutive layer. Note that for the IMDB task also an embedding dropout with $p = 0.65$ is used.

We train the UnICORNN for a total of 50 epochs on the IMDB task and for a total of 250 epochs on the EigenWorms task. Moreover, we train UnICORNN for 650 epochs on psMNIST, after which we decrease the learning rate by a factor of 10 and proceed training for 3 times the amount of epochs used before reducing the learning rate. On all other tasks, UnICORNN is trained for 250 epochs, after which we decrease the learning rate by a factor of 10 and proceed training for additional 250 epochs. The resulting best performing networks are selected *based on a validation set*.

Table 1. Hyperparameters of the best performing UnICORNN architecture (based on a validation set) for each experiment.

Experiment	learning rate	dropout	batch size	Δt	α
noise padded CIFAR-10	3.14×10^{-2}	1.0×10^{-1}	30	1.26×10^{-1}	13.0
psMNIST (#units = 128)	1.14×10^{-3}	1.0×10^{-1}	64	4.82×10^{-1}	12.53
psMNIST (#units = 256)	2.51×10^{-3}	1.0×10^{-1}	32	1.9×10^{-1}	30.65
IMDB	1.67×10^{-4}	6.1×10^{-1}	32	2.05×10^{-1}	0.0
EigenWorms	8.59×10^{-3}	0.0	8	3.43×10^{-2}	0.0
Healthcare: RR	3.98×10^{-3}	1.0×10^{-1}	32	1.1×10^{-2}	9.0
Healthcare: HR	2.88×10^{-3}	1.0×10^{-1}	32	4.6×10^{-2}	10.0

B. Implementation details

As already described in the implementation details of the main paper, we can speed up the computation of the forward and backward pass, by parallelizing the input transformation and computing the recurrent part for each independent dimension in an independent CUDA thread. While the forward/backward pass for the input transformation is simply that of an affine transformation, we discuss only the recurrent part. Since we compute the gradients of each dimension of the UnICORNN independently and add them up afterwards to get the full gradient, we will simplify to the following one-dimensional system:

$$
\begin{aligned}
z_n &= z_{n-1} - \Delta t \hat{\sigma}(c)[\sigma(w y_{n-1} + x_n) + \alpha y_{n-1}], \\
y_n &= y_{n-1} + \Delta t \hat{\sigma}(c) z_n,
\end{aligned}
$$

where $x_n = (\mathbf{V}\mathbf{u}_n)_j$ is the transformed input corresponding to the respective dimension $j = 1, \dots, m$.

Since we wish to train the UnICORNN on some given objective

$$\mathcal{E} := \sum_{n=1}^{N} \tilde{\mathcal{E}}(y_n), \tag{1}$$

where $\tilde{\mathcal{E}}$ is some loss function taking the hidden states y_n as inputs, for instance mean-square distance of (possibly transformed) hidden states y_n to some ground truth. During training, we compute gradients of the loss function (1) with respect to the following quantities $\mathbf{\Theta} = [w, \Delta t, x_n]$, i.e.

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{n=1}^{N} \frac{\partial \tilde{\mathcal{E}}(y_n)}{\partial \theta}, \quad \forall\, \theta \in \mathbf{\Theta}. \tag{2}$$

We can work out a recursion formula to compute the gradients in (2). We will exemplarily provide the formula for the gradient with respect to the hidden weight w. The computation of the gradients with respect to the other quantities follow similarly. Thus

$$\delta_k^z = \delta_{k-1}^z + \delta_{k-1}^y \Delta t \hat{\sigma}(c), \tag{3}$$

$$\delta_k^y = \delta_{k-1}^y - \delta_k^z \Delta t \hat{\sigma}(c)[\sigma'(w y_{N-k} + x_{N-k+1}) w + \alpha] + \frac{\partial \tilde{\mathcal{E}}}{\partial y_{N-k}}, \tag{4}$$

with initial values $\delta_0^y = \frac{\partial \tilde{\mathcal{E}}}{\partial y_N}$ and $\delta_0^z = 0$. The gradient can then be computed as

$$\frac{\partial \mathcal{E}}{\partial w} = \sum_{k=1}^{N} a_k, \qquad \text{with } a_k = -\delta_k^z \Delta t \hat{\sigma}(c) \sigma'(w y_{N-k} + x_{N-k+1}) y_{N-k}. \tag{5}$$

Note that this recursive formula is a direct formulation of the back-propagation through time algorithm (Werbos, 1990) for the UnICORNN.

We can verify formula (3)-(5) by explicitly calculating the gradient in (2):

$$\begin{aligned}
\frac{\partial \mathcal{E}}{\partial w} &= \sum_{n=1}^{N} \frac{\partial \tilde{\mathcal{E}}(y_n)}{\partial w} = \sum_{n=1}^{N-1} \frac{\partial \tilde{\mathcal{E}}(y_n)}{\partial w} + \frac{\partial \tilde{\mathcal{E}}}{\partial y_N} \left[\frac{\partial y_{N-1}}{\partial w} + \Delta t \hat{\sigma}(c) \left(\frac{\partial z_{N-1}}{\partial w} - \Delta t \hat{\sigma}(c) (\sigma'(w y_{N-1} + x_N) \right.\right. \\
&\quad \left.\left. (y_{N-1} + w \frac{\partial y_{N-1}}{\partial w}) + \alpha \frac{\partial y_{N-1}}{\partial w} \right) \right] = \sum_{n=1}^{N-2} \frac{\partial \tilde{\mathcal{E}}(y_n)}{\partial w} + a_1 + \delta_1^z \frac{\partial z_{N-1}}{\partial w} + \delta_1^y \frac{\partial y_{N-1}}{\partial w} \\
&= \sum_{n=1}^{N-2} \frac{\partial \tilde{\mathcal{E}}(y_n)}{\partial w} + a_1 + \delta_1^y \frac{\partial y_{N-2}}{\partial w} + (\delta_1^y \Delta t \hat{\sigma}(c) + \delta_1^z) \left(\frac{\partial z_{N-2}}{\partial w} - \Delta t \hat{\sigma}(c) (\sigma'(w y_{N-2} + x_{N-1}) \right. \\
&\quad \left. (y_{N-2} + w \frac{\partial y_{N-2}}{\partial w}) + \alpha \frac{\partial y_{N-2}}{\partial w}) \right) = \sum_{n=1}^{N-3} \frac{\partial \tilde{\mathcal{E}}(y_n)}{\partial w} + \sum_{k=1}^{2} a_k + \delta_2^z \frac{\partial z_{N-2}}{\partial w} + \delta_2^y \frac{\partial y_{N-2}}{\partial w}.
\end{aligned}$$

Iterating the same reformulation yields the desired formula (3)-(5).

C. Rigorous bounds on UnICORNN

We rewrite UnICORNN (Eqn. (6) in the main text) in the following form: for all $1 \leq \ell \leq L$ and for all $1 \leq i \leq m$

$$\begin{aligned}
\mathbf{y}_n^{\ell,i} &= \mathbf{y}_{n-1}^{\ell,i} + \Delta t \hat{\sigma}(\mathbf{c}^{\ell,i}) \mathbf{z}_n^{\ell,i}, \\
\mathbf{z}_n^{\ell,i} &= \mathbf{z}_{n-1}^{\ell,i} - \Delta t \hat{\sigma}(\mathbf{c}^{\ell,i}) \sigma(\mathbf{A}_{n-1}^{\ell,i}) - \alpha \Delta t \hat{\sigma}(\mathbf{c}^{\ell,i}) \mathbf{y}_{n-1}^{\ell,i}, \\
\mathbf{A}_{n-1}^{\ell,i} &= \mathbf{w}^{\ell,i} \mathbf{y}_{n-1}^{\ell,i} + \left(\mathbf{V}^{\ell} \mathbf{y}_n^{\ell-1}\right)^i + \mathbf{b}^{\ell,i}.
\end{aligned} \tag{6}$$

Here, we have denoted the i-th component of a vector $\mathbf{x}$ as $\mathbf{x}^i$.

We follow standard practice and set $\mathbf{y}_0^\ell = \mathbf{z}_0^\ell \equiv 0$, for all $1 \leq \ell \leq L$. Moreover for simplicity of exposition, we set $\alpha > 0$ in the following.

C.1. Pointwise bounds on hidden states.

We have the following bounds on the discrete hidden states,

Proposition C.1. *Let $\mathbf{y}_n^\ell, \mathbf{z}_n^\ell$ be the hidden states at the n-th time level t_n for UnICORNN (6), then under the assumption that the time step $\Delta t << 1$ is sufficiently small, these hidden states are bounded as,*

$$\max_{1\leq i\leq m} |\mathbf{y}_n^{\ell,i}| \leq \sqrt{\frac{2}{\alpha}\left(1+2\beta t_n\right)}, \quad \max_{1\leq i\leq m} |\mathbf{z}_n^{\ell,i}| \leq \sqrt{2\left(1+2\beta t_n\right)} \quad \forall n, \forall\, 1\leq \ell \leq L, \tag{7}$$

with the constant

$$\beta = \max\{1+2\alpha, 4\alpha^2\}.$$

Proof. We fix ℓ, n and multiply the first equation in (6) with $\alpha \mathbf{y}_{n-1}^{\ell,i}$ and use the elementary identity

$$b(a-b) = \frac{a^2}{2} - \frac{b^2}{2} - \frac{1}{2}(a-b)^2,$$

to obtain

$$\begin{aligned}
\frac{\alpha(\mathbf{y}_n^{\ell,i})^2}{2} &= \frac{\alpha(\mathbf{y}_{n-1}^{\ell,i})^2}{2} + \frac{\alpha}{2}(\mathbf{y}_n^{\ell,i} - \mathbf{y}_{n-1}^{\ell,i})^2 + \alpha\Delta t \hat{\sigma}(\mathbf{c}^{\ell,i})\mathbf{y}_{n-1}^{\ell,i}\mathbf{z}_n^{\ell,i}, \\
&= \frac{\alpha(\mathbf{y}_{n-1}^{\ell,i})^2}{2} + \frac{\alpha\Delta t^2}{2}(\hat{\sigma}(\mathbf{c}^{\ell,i}))^2(\mathbf{z}_n^{\ell,i})^2 + \alpha\Delta t \hat{\sigma}(\mathbf{c}^{\ell,i})\mathbf{y}_{n-1}^{\ell,i}\mathbf{z}_n^{\ell,i}.
\end{aligned} \tag{8}$$

Next, we multiply the second equation in (6) with $\mathbf{z}_n^{\ell,i}$ and use the elementary identity

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

to obtain

$$\begin{aligned}
\frac{(\mathbf{z}_n^{\ell,i})^2}{2} &= \frac{(\mathbf{z}_{n-1}^{\ell,i})^2}{2} - \frac{1}{2}(\mathbf{z}_n^{\ell,i} - \mathbf{z}_{n-1}^{\ell,i})^2 - \Delta t \hat{\sigma}(\mathbf{c}^{\ell,i})\sigma(\mathbf{A}_{n-1}^{\ell,i})\left(\mathbf{z}_n^{\ell,i} - \mathbf{z}_{n-1}^{\ell,i}\right) \\
&\quad - \Delta t \hat{\sigma}(\mathbf{c}^{\ell,i})\sigma(\mathbf{A}_{n-1}^{\ell,i})\mathbf{z}_{n-1}^{\ell,i} - \alpha\Delta t \hat{\sigma}(\mathbf{c}^{\ell,i})\mathbf{y}_{n-1}^{\ell,i}\mathbf{z}_n^{\ell,i}.
\end{aligned} \tag{9}$$

Adding (8) and (9) and using Cauchy's inequality yields,

$$\begin{aligned}
\frac{\alpha(\mathbf{y}_n^{\ell,i})^2}{2} + \frac{(\mathbf{z}_n^{\ell,i})^2}{2} &\leq \frac{\alpha(\mathbf{y}_{n-1}^{\ell,i})^2}{2} + \frac{(1+\Delta t)(\mathbf{z}_{n-1}^{\ell,i})^2}{2} + \frac{\alpha\Delta t^2}{2}(\hat{\sigma}(\mathbf{c}^{\ell,i}))^2(\mathbf{z}_n^{\ell,i})^2 \\
&\quad + (\hat{\sigma}(\mathbf{c}^{\ell,i}))^2(\sigma(\mathbf{A}_{n-1}^{\ell,i}))^2\Delta t + \frac{\Delta t - 1}{2}(\mathbf{z}_n^{\ell,i} - \mathbf{z}_{n-1}^{\ell,i})^2 \\
\Rightarrow \alpha(\mathbf{y}_n^{\ell,i})^2 + (\mathbf{z}_n^{\ell,i})^2 &\leq \alpha(\mathbf{y}_{n-1}^{\ell,i})^2 + (1+\Delta t)(\mathbf{z}_{n-1}^{\ell,i})^2 + 2\Delta t + \alpha\Delta t^2(\mathbf{z}_n^{\ell,i})^2,
\end{aligned}$$

where the last inequality follows from the fact that $|\sigma|, |\hat{\sigma}| \leq 1$ and $\Delta t < 1$. Using the elementary inequality,

$$(a+b+c)^2 \leq 4a^2 + 4b^2 + 2c^2,$$

and substituting for $\mathbf{z}_n^{\ell,i}$ from the second equation of (6) in the last inequality leads to,

$$\alpha(\mathbf{y}_n^{\ell,i})^2 + (\mathbf{z}_n^{\ell,i})^2 \leq (1+4\alpha^2\Delta t^4)\alpha(\mathbf{y}_{n-1}^{\ell,i})^2 + (1+\Delta t + 2\alpha\Delta t^2)(\mathbf{z}_{n-1}^{\ell,i})^2 + 2\Delta t + 4\alpha\Delta t^4.$$

Denoting $H_n = \alpha(\mathbf{y}_n^{\ell,i})^2 + (\mathbf{z}_n^{\ell,i})^2$ and

$$G := 1 + \beta\Delta t, \quad \beta = \max\{1+2\alpha, 4\alpha^2\}$$

yields the following inequality,

$$H_n \leq G H_{n-1} + 2\Delta t(1 + 2\alpha\Delta t^3). \tag{10}$$

Iterating the above n-times and using the fact that the initial data is such that $H_0 \equiv 0$ we obtain,

$$
\begin{aligned}
H_n &\leq \left(2\Delta t + 4\alpha\Delta t^4\right)\sum_{k=0}^{n-1}(1+\beta\Delta t)^k \\
&\leq \frac{(1+\beta\Delta t)^n}{\beta\Delta t}\left(2\Delta t + 4\alpha\Delta t^4\right) \\
&\leq \frac{1}{\beta}\left(1+2\beta n\Delta t\right)\left(2+4\alpha\Delta t^3\right) \quad \text{as } \Delta t << 1, \\
&\leq 2(1+2\beta t_n) \text{ (from definition of } \beta).
\end{aligned}
\tag{11}
$$

The definition of H clearly implies the desired bound (7). □

C.2. On the exploding gradient problem for UnICORNN and Proof of proposition 3.1 of the main text.

We train the RNN (6) to minimize the loss function,

$$
\mathcal{E} := \frac{1}{N}\sum_{n=1}^{N}\mathcal{E}_n, \quad \mathcal{E}_n = \frac{1}{2}\|\mathbf{y}_n^L - \bar{\mathbf{y}}_n\|_2^2, \tag{12}
$$

with $\bar{\mathbf{y}}$ being the underlying ground truth (training data). Note that the loss function (12) only involves the output at the last hidden layer (we set the affine output layer to identity for the sake of simplicity). During training, we compute gradients of the loss function (12) with respect to the trainable weights and biases $\mathbf{\Theta} = [\mathbf{w}^\ell, \mathbf{V}^\ell, \mathbf{b}^\ell, \mathbf{c}^\ell]$, for all $1 \leq \ell \leq L$ i.e.

$$
\frac{\partial \mathcal{E}}{\partial \theta} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial \mathcal{E}_n}{\partial \theta}, \quad \forall\, \theta \in \mathbf{\Theta}. \tag{13}
$$

We have the following bound on the gradient (13),

Proposition C.2. *Let the time step $\Delta t << 1$ be sufficiently small in the RNN (6) and let $\mathbf{y}_n^\ell, \mathbf{z}_n^\ell$, for $1 \leq \ell \leq L$, be the hidden states generated by the RNN (6). Then, the gradient of the loss function $\mathcal{E}$ (12) with respect to any parameter $\theta \in \mathbf{\Theta}$ is bounded as,*

$$
\left|\frac{\partial \mathcal{E}}{\partial \theta}\right| \leq \frac{1-(\Delta t)^L}{1-\Delta t} T(1+2\gamma T)\overline{\mathbf{V}}(\overline{Y}+\mathbf{F})\mathbf{\Delta}, \tag{14}
$$

with $\bar{Y} = \max\limits_{1\leq n\leq N}\|\bar{\mathbf{y}}_n\|_\infty$, be a bound on the underlying training data and other quantities in (14) defined as,

$$
\begin{aligned}
\gamma &= \max\left(2, \|\mathbf{w}^L\|_\infty + \alpha\right) + \frac{\left(\max\left(2, \|\mathbf{w}^L\|_\infty + \alpha\right)\right)^2}{2}, \\
\overline{\mathbf{V}} &= \prod_{q=1}^{L}\max\{1, \|\mathbf{V}^q\|_\infty\}, \\
\mathbf{F} &= \sqrt{\frac{2}{\alpha}\left(1+2\beta T\right)}, \\
\mathbf{\Delta} &= \left(2 + \sqrt{(1+2\beta T)} + (2+\alpha)\sqrt{\frac{2}{\alpha}\left(1+2\beta T\right)}\right).
\end{aligned}
\tag{15}
$$

Proof. For any $1 \leq n \leq N$ and $1 \leq \ell \leq L$, let $\mathbf{X}_n^\ell \in \mathbb{R}^{2m}$ be the augmented hidden state vector defined by,

$$
\mathbf{X}_n^\ell = \left[\mathbf{y}_n^{\ell,1}, \mathbf{z}_n^{\ell,1}, \ldots, \mathbf{y}_n^{\ell,i}, \mathbf{z}_n^{\ell,i}, \ldots, \mathbf{y}_n^{\ell,m}, \mathbf{z}_n^{\ell,m}\right]. \tag{16}
$$

For any $\theta \in \mathbf{\Theta}$, we can apply the chain rule repeatedly to obtain the following extension of the formula of (Pascanu et al., 2013) to a deep RNN,

$$
\frac{\partial \mathcal{E}_n}{\partial \theta} = \sum_{\ell=1}^{L}\sum_{k=1}^{n}\underbrace{\frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n^L}\frac{\partial \mathbf{X}_n^L}{\partial \mathbf{X}_k^\ell}\frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta}}_{\frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta}}. \tag{17}
$$

Here, the notation $\frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta}$ refers to taking the partial derivative of $\mathbf{X}_k^\ell$ with respect to the parameter θ, while keeping the other arguments constant.

We remark that the quantity $\frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta}$ denotes the contribution from the k-recurrent step at the l-th hidden layer of the deep RNN (6) to the overall hidden state gradient at the step n.

It is straightforward to calculate that,

$$\frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n^L} = \left[\mathbf{y}_n^{L,1} - \overline{\mathbf{y}}_n^1, 0, \dots, \mathbf{y}_n^{L,i} - \overline{\mathbf{y}}_n^i, 0, \dots, \mathbf{y}_n^{L,m} - \overline{\mathbf{y}}_n^m, 0\right]. \tag{18}$$

Repeated application of the chain and product rules yields,

$$\frac{\partial \mathbf{X}_n^L}{\partial \mathbf{X}_k^\ell} = \prod_{j=k+1}^{n} \frac{\partial \mathbf{X}_j^L}{\partial \mathbf{X}_{j-1}^L} \prod_{q=\ell+1}^{L} \frac{\partial \mathbf{X}_k^q}{\partial \mathbf{X}_k^{q-1}}. \tag{19}$$

For any j, a straightforward calculation using the form of the RNN (6) leads to the following representation formula for the matrix $\frac{\partial \mathbf{X}_j^L}{\partial \mathbf{X}_{j-1}^L} \in \mathbb{R}^{2m} \times \mathbb{R}^{2m}$:

$$\frac{\partial \mathbf{X}_j^L}{\partial \mathbf{X}_{j-1}^L} = \begin{bmatrix} \mathbf{B}_j^{L,1} & 0 & \dots & 0 \\ 0 & \mathbf{B}_j^{L,2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \mathbf{B}_j^{L,m} \end{bmatrix}, \tag{20}$$

with the block matrices $\mathbf{B}_j^{L,i} \in \mathbb{R}^{2\times 2}$ given by,

$$\mathbf{B}_j^{L,i} = \begin{bmatrix} 1 - (\hat{\sigma}(\mathbf{c}^{L,i}))^2 \Delta t^2 \left(\mathbf{w}^{L,i} \sigma'(\mathbf{A}_{j-1}^{L,i}) + \alpha\right) & \hat{\sigma}(\mathbf{c}^{L,i}) \Delta t \\ -\hat{\sigma}(\mathbf{c}^{L,i}) \Delta t \left(\mathbf{w}^{L,i} \sigma'(\mathbf{A}_{j-1}^{L,i}) + \alpha\right) & 1 \end{bmatrix}. \tag{21}$$

Similarly for any q, the matrix $\frac{\partial \mathbf{X}_k^q}{\partial \mathbf{X}_k^{q-1}} \in \mathbb{R}^{2m \times 2m}$ can be readily computed as,

$$\frac{\partial \mathbf{X}_k^q}{\partial \mathbf{X}_k^{q-1}} = \begin{bmatrix} \mathbf{D}_{11}^{q,k} & 0 & \mathbf{D}_{12}^{q,k} & 0 & \dots & \dots & \mathbf{D}_{1m}^{q,k} & 0 \\ \mathbf{E}_{11}^{q,k} & 0 & \mathbf{E}_{12}^{q,k} & 0 & \dots & \dots & \mathbf{E}_{1m}^{q,k} & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \mathbf{D}_{m1}^{q,k} & 0 & \mathbf{D}_{m2}^{q,k} & 0 & \dots & \dots & \mathbf{D}_{mm}^{q,k} & 0 \\ \mathbf{E}_{m1}^{q,k} & 0 & \mathbf{E}_{m2}^{q,k} & 0 & \dots & \dots & \mathbf{E}_{mm}^{q,k} & 0 \end{bmatrix}, \tag{22}$$

with entries given by,

$$\mathbf{D}_{i,\bar{i}}^{q,k} = -\Delta t^2 (\hat{\sigma}(\mathbf{c}^{q,i}))^2 \sigma'\left(\mathbf{A}_{k-1}^{q,i}\right) \mathbf{V}_{i\bar{i}}^q, \quad \mathbf{E}_{i,\bar{i}}^{q,k} = -\Delta t \hat{\sigma}(\mathbf{c}^{q,i}) \sigma'\left(\mathbf{A}_{k-1}^{q,i}\right) \mathbf{V}_{i\bar{i}}^q. \tag{23}$$

A direct calculation with (21) leads to,

$$\begin{aligned} \|\mathbf{B}_j^{L,i}\|_\infty &\le \max\left(1 + \Delta t + (|\mathbf{w}^{L,i}| + \alpha)\Delta t^2, 1 + (|\mathbf{w}^{L,i}| + \alpha)\Delta t\right) \\ &\le 1 + \max\left(2, |\mathbf{w}^{L,i}| + \alpha\right) \Delta t + \left(\max\left(2, |\mathbf{w}^{L,i}| + \alpha\right)\right)^2 \frac{\Delta t^2}{2}. \end{aligned} \tag{24}$$

Using the definition of the L^∞ norm of a matrix, we use (24) to the derive the following bound from (20),

$$\begin{aligned} \left\| \frac{\partial \mathbf{X}_j^L}{\partial \mathbf{X}_{j-1}^L} \right\|_\infty &\le 1 + \max\left(2, \|\mathbf{w}^L\|_\infty + \alpha\right) \Delta t + \left(\max\left(2, \|\mathbf{w}^L\|_\infty + \alpha\right)\right)^2 \frac{\Delta t^2}{2} \\ &\le 1 + \gamma \Delta t, \end{aligned} \tag{25}$$

with γ defined in (15).

As $\Delta t < 1$, it is easy to see that,

$$\left\| \frac{\partial \mathbf{X}_k^q}{\partial \mathbf{X}_k^{q-1}} \right\|_\infty \leq \|\mathbf{V}^q\|_\infty \Delta t. \tag{26}$$

Combining (25) and (26) , we obtain from (19)

$$\begin{aligned}\left\| \frac{\partial \mathbf{X}_n^L}{\partial \mathbf{X}_k^\ell} \right\|_\infty &\leq \prod_{j=k+1}^{n} \left\| \frac{\partial \mathbf{X}_j^L}{\partial \mathbf{X}_{j-1}^L} \right\|_\infty \prod_{q=\ell+1}^{L} \left\| \frac{\partial \mathbf{X}_k^q}{\partial \mathbf{X}_k^{q-1}} \right\|_\infty \\ &\leq \Delta t^{L-\ell} \prod_{q=\ell+1}^{L} \|\mathbf{V}^q\|_\infty (1 + 2\gamma(n-k)\Delta t), \quad (\text{as } \Delta t << 1) \\ &\leq \overline{\mathbf{V}} \Delta t^{L-\ell}(1 + 2\gamma t_n),\end{aligned} \tag{27}$$

where the last inequality follows from the fact that $t_n = n\Delta t \leq T$ and the definition of $\overline{\mathbf{V}}$ in (15).

Next, we observe that for any $\theta \in \mathbf{\Theta}$

$$\frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta} = \left[\frac{\partial^+ \mathbf{y}_k^{\ell,1}}{\partial \theta}, \frac{\partial^+ \mathbf{z}_k^{\ell,1}}{\partial \theta} \ldots, \ldots, \frac{\partial^+ \mathbf{y}_k^{\ell,i}}{\partial \theta}, \frac{\partial^+ \mathbf{z}_k^{\ell,i}}{\partial \theta}, \ldots, \ldots, \frac{\partial^+ \mathbf{y}_k^{\ell,m}}{\partial \theta}, \frac{\partial^+ \mathbf{z}_k^{\ell,m}}{\partial \theta} \right]^\top. \tag{28}$$

For any $1 \leq i \leq m$, a direct calculation with the RNN (6) yields,

$$\begin{aligned}\frac{\partial^+ \mathbf{y}_k^{\ell,i}}{\partial \theta} &= \Delta t \hat{\sigma}'(\mathbf{c}^{\ell,i}) \frac{\partial \mathbf{c}^{\ell,i}}{\partial \theta} \mathbf{z}_k^{\ell,i} + \Delta t \hat{\sigma}(\mathbf{c}^{\ell,i}) \frac{\partial^+ \mathbf{z}_k^{\ell,i}}{\partial \theta}, \\ \frac{\partial^+ \mathbf{z}_k^{\ell,i}}{\partial \theta} &= -\Delta t \hat{\sigma}'(\mathbf{c}^{\ell,i}) \frac{\partial \mathbf{c}^{\ell,i}}{\partial \theta} \sigma(\mathbf{A}_{k-1}^{\ell,i}) - \Delta t \hat{\sigma}(\mathbf{c}^{\ell,i}) \sigma'(\mathbf{A}_{k-1}^{\ell,i}) \frac{\partial \mathbf{A}_{k-1}^{\ell,i}}{\partial \theta} - \alpha \Delta t \hat{\sigma}'(\mathbf{c}^{\ell,i}) \frac{\partial \mathbf{c}^{\ell,i}}{\partial \theta} \mathbf{y}_{k-1}^{\ell,i}.\end{aligned} \tag{29}$$

Next, we have to compute explicitly $\frac{\partial \mathbf{c}^{\ell,i}}{\partial \theta}$ and $\frac{\partial \mathbf{A}_{k-1}^{\ell,i}}{\partial \theta}$ in order to complete the expressions (29). To this end, we need to consider explicit forms of the parameter θ and obtain,

If $\theta = \mathbf{w}^{q,p}$, for some $1 \leq q \leq L$ and $1 \leq p \leq m$, then,

$$\frac{\partial \mathbf{A}_{k-1}^{\ell,i}}{\partial \theta} = \begin{cases} \mathbf{y}_{k-1}^{\ell,i}, & \text{if} \quad q = \ell, p = i, \\ 0, & \text{if} \quad \text{otherwise}. \end{cases} \tag{30}$$

If $\theta = \mathbf{b}^{q,p}$, for some $1 \leq q \leq L$ and $1 \leq p \leq m$, then,

$$\frac{\partial \mathbf{A}_{k-1}^{\ell,i}}{\partial \theta} = \begin{cases} 1, & \text{if} \quad q = \ell, p = i, \\ 0, & \text{if} \quad \text{otherwise}. \end{cases} \tag{31}$$

If $\theta = \mathbf{V}_{p,\bar{p}}^q$, for some $1 \leq q \leq L$ and $1 \leq p, \bar{p} \leq m$, then,

$$\frac{\partial \mathbf{A}_{k-1}^{\ell,i}}{\partial \theta} = \begin{cases} \mathbf{y}_k^{\ell-1,\bar{p}}, & \text{if} \quad q = \ell, p = i, \\ 0, & \text{if} \quad \text{otherwise}. \end{cases} \tag{32}$$

If $\theta = \mathbf{c}^{q,p}$for any $1 \leq q \leq L$ and $1 \leq p \leq m$, then,

$$\frac{\partial \mathbf{A}_{k-1}^{\ell,i}}{\partial \theta} \equiv 0. \tag{33}$$

Similarly, if $\theta = \mathbf{w}^{q,p}$ or $\theta = \mathbf{b}^{q,p}$, for some $1 \leq q \leq L$ and $1 \leq p \leq m$, or If $\theta = \mathbf{V}_{p,\bar{p}}^q$, for some $1 \leq q \leq L$ and $1 \leq p, \bar{p} \leq m$, then

$$\frac{\partial \mathbf{c}^{\ell,i}}{\partial \theta} \equiv 0. \tag{34}$$

On the other hand, if $\theta = \mathbf{c}^{q,p}$for any $1 \leq q \leq L$ and $1 \leq p \leq m$, then

$$\frac{\partial \mathbf{c}^{\ell,i}}{\partial \theta} = \begin{cases} 1, & \text{if} \quad q = \ell, p = i, \\ 0, & \text{if} \quad \text{otherwise}. \end{cases} \tag{35}$$

For any $\theta \in \mathbf{\Theta}$, by substituting (30) to (35) into (29) and doing some simple algebra with norms, leads to the following inequalities,

$$\left| \frac{\partial^+ \mathbf{z}_k^{\ell,i}}{\partial \theta} \right| \leq \Delta t \left(1 + \alpha |\mathbf{y}_{k-1}^{\ell,i}| + \max\left(|\mathbf{y}_{k-1}^{\ell,i}|, |\mathbf{y}_k^{\ell-1,\bar{p}}|, 1 \right) \right), \tag{36}$$

and,

$$\left| \frac{\partial^+ \mathbf{y}_k^{\ell,i}}{\partial \theta} \right| \leq \Delta t |\mathbf{z}_k^{\ell,i}| + \Delta t^2 \left(1 + \alpha |\mathbf{y}_{k-1}^{\ell,i}| + \max\left(|\mathbf{y}_{k-1}^{\ell,i}|, |\mathbf{y}_k^{\ell-1,\bar{p}}|, 1 \right) \right), \tag{37}$$

for any $1 \leq \bar{p} \leq m$.

By the definition of L^∞ norm of a vector and some straightforward calculations with (37) yields,

$$\left\| \frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta} \right\|_\infty \leq \Delta t \left(2 + \|\mathbf{z}_k^\ell\|_\infty + (1+\alpha)\|\mathbf{y}_{k-1}^\ell\|_\infty + \|\mathbf{y}_k^{\ell-1}\|_\infty \right). \tag{38}$$

From the pointwise bounds (7), we can directly bound the above inequality further as,

$$\left\| \frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta} \right\|_\infty \leq \Delta t \left(2 + \sqrt{2\left(1 + 2\beta T\right)} + (2+\alpha)\sqrt{\frac{2}{\alpha}\left(1 + 2\beta T\right)} \right). \tag{39}$$

By (18) and the definition of $\overline{Y}$ as well as the bound (7) on the hidden states, it is straightforward to obtain that,

$$\left\| \frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n^L} \right\|_\infty \leq \overline{Y} + \sqrt{\frac{2}{\alpha}\left(1 + 2\beta T\right)} \tag{40}$$

From the definition in (17), we have

$$\left| \frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta} \right| \leq \left\| \frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n^L} \right\|_\infty \left\| \frac{\partial \mathbf{X}_n^L}{\partial \mathbf{X}_k^\ell} \right\|_\infty \left\| \frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta} \right\|_\infty. \tag{41}$$

Substituting (40), (39) and (25) into (41) yields,

$$\left| \frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta} \right| \leq \Delta t^{L-\ell+1} \left(1 + 2\gamma T\right) \overline{\mathbf{V}}(\overline{Y} + \mathbf{F})\mathbf{\Delta}, \tag{42}$$

with $\mathbf{F}$ and $\mathbf{\Delta}$ defined in (15).

Therefore, from the fact that $\Delta t < 1, t_n = n\Delta t \leq T$ and (17), we obtain

$$\left| \frac{\partial \mathcal{E}_n}{\partial \theta} \right| \leq \frac{1 - (\Delta t)^L}{1 - \Delta t} T(1 + 2\gamma T)\overline{\mathbf{V}}(\overline{Y} + \mathbf{F})\mathbf{\Delta}. \tag{43}$$

By the definition of the loss function (12) and the fact that the right-hand-side of (43) is independent of n leads to the desired bound (14).

□

C.3. On the Vanishing gradient problem for UnICORNN and Proof of Proposition 3.2 of the main text.

By applying the chain rule repeatedly to the each term on the right-hand-side of (13), we obtain

$$\frac{\partial \mathcal{E}_n}{\partial \theta} = \sum_{\ell=1}^{L}\sum_{k=1}^{n} \frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta}, \quad \frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta} := \frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n^L}\frac{\partial \mathbf{X}_n^L}{\partial \mathbf{X}_k^\ell}\frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta}. \tag{44}$$

Here, the notation $\frac{\partial^+ \mathbf{X}_k^\ell}{\partial \theta}$ refers to taking the partial derivative of $\mathbf{X}_k^\ell$ with respect to the parameter θ, while keeping the other arguments constant. The quantity $\frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta}$ denotes the contribution from the k-recurrent step at the l-th hidden layer of the deep RNN (6) to the overall hidden state gradient at the step n. The vanishing gradient problem (Pascanu et al., 2013) arises if $\left|\frac{\partial \mathcal{E}_{k,\ell}^{(n,L)}}{\partial \theta}\right|$, defined in (44), $\to 0$ 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 at the ℓ-th layer to error at time step t_n is infinitesimally small.

As argued in the main text, the vanishing gradient problem for RNNs focuses on the possible smallness of contributions of the gradient over a large number of recurrent steps. As this behavior of the gradient is independent of the number of layers, we start with a result on the vanishing gradient problem for a single hidden layer here. Also, for the sake of definiteness, we set the scalar parameter $\theta = \mathbf{w}^{1,p}$ for some $1 \le p \le m$. Similar results also hold for any other $\theta \in \mathbf{\Theta}$. Moreover, we introduce the following *order*-notation,

$$\begin{aligned} \beta = \mathcal{O}(\gamma), &\text{ for } \gamma, \beta \in \mathbb{R}_+ \quad \text{if there exists constants } \overline{C}, \underline{C} \text{ such that } \underline{C}\gamma \le \beta \le \overline{C}\gamma. \\ \mathbf{M} = \mathcal{O}(\gamma), &\text{ for } \mathbf{M} \in \mathbb{R}^{d_1 \times d_2}, \gamma \in \mathbb{R}_+ \quad \text{if there exists constant } \overline{C} \text{ such that } \|\mathbf{M}\| \le \overline{C}\gamma. \end{aligned} \tag{45}$$

We restate Proposition 3.2 of the main text,

Proposition C.3. *Let $\mathbf{y}_n$ be the hidden states generated by the RNN (6). Then the gradient for long-term dependencies, i.e. $k << n$, satisfies the representation formula,*

$$\frac{\partial \mathcal{E}_{k,1}^{(n,1)}}{\partial \mathbf{w}^{1,p}} = -\Delta t \hat{\sigma}(\mathbf{c}^{1,p})^2 t_n \sigma'(\mathbf{A}_{k-1}^{1,p})\mathbf{y}_{k-1}^{1,p}\left(\mathbf{y}_n^{1,p} - \overline{\mathbf{y}}_n^p\right) + \mathcal{O}(\Delta t^2). \tag{46}$$

Proof. Following the definition (44) and as $L = 1$ and $\theta = \mathbf{w}^{1,p}$, we have,

$$\frac{\partial \mathcal{E}_{k,1}^{(n,1)}}{\partial \mathbf{w}^{1,p}} := \frac{\partial \mathcal{E}_n}{\partial \mathbf{X}_n^1}\frac{\partial \mathbf{X}_n^1}{\partial \mathbf{X}_k^1}\frac{\partial^+ \mathbf{X}_k^1}{\partial \mathbf{w}^{1,p}}. \tag{47}$$

We will explicitly compute all three expressions on the right-hand-side of (47). To start with, using (28), (29) and (30), we obtain,

$$\begin{aligned} \frac{\partial^+ \mathbf{X}_k^1}{\partial \mathbf{w}^{1,p}} &= \left[0, 0, \ldots, \ldots, \frac{\partial^+ \mathbf{y}_k^{1,p}}{\partial \mathbf{w}^{1,p}}, \frac{\partial^+ \mathbf{z}_k^{1,p}}{\partial \mathbf{w}^{1,p}}, \ldots, \ldots, 0, 0\right]^\top, \\ \left(\frac{\partial^+ \mathbf{X}_k^1}{\partial \mathbf{w}^{1,p}}\right)_{2p-1} &= \frac{\partial^+ \mathbf{y}_k^{1,p}}{\partial \mathbf{w}^{1,p}} = -\Delta t^2 (\hat{\sigma}(\mathbf{c}^{1,p}))^2 \sigma'(\mathbf{A}_{k-1}^{1,p})\mathbf{y}_{k-1}^{1,p}, \\ \left(\frac{\partial^+ \mathbf{X}_k^1}{\partial \mathbf{w}^{1,p}}\right)_{2p} &= \frac{\partial^+ \mathbf{z}_k^{1,p}}{\partial \mathbf{w}^{1,p}} = -\Delta t \hat{\sigma}(\mathbf{c}^{1,p}) \sigma'(\mathbf{A}_{k-1}^{1,p})\mathbf{y}_{k-1}^{1,p}. \end{aligned} \tag{48}$$

Using the product rule (19) we have,

$$\frac{\partial \mathbf{X}_n^1}{\partial \mathbf{X}_k^1} = \prod_{j=k+1}^{n} \frac{\partial \mathbf{X}_j^1}{\partial \mathbf{X}_{j-1}^1}. \tag{49}$$

Observing from the expressions (20) and (21) and using the *order*-notation (45), we obtain that,

$$\frac{\partial \mathbf{X}_j^1}{\partial \mathbf{X}_{j-1}^1} = \mathbf{I}_{2m \times 2m} + \Delta t \mathbf{C}_j^1 + \mathcal{O}(\Delta t^2), \tag{50}$$

with $\mathbf{I}_{k\times k}$ is the $k \times k$ Identity matrix and the matrix $\mathbf{C}_j^1$ defined by,

$$\frac{\partial \mathbf{X}_j^L}{\partial \mathbf{X}_{j-1}^L} = \begin{bmatrix} \mathbf{C}_j^{1,1} & 0 & \dots & 0 \\ 0 & \mathbf{C}_j^{1,2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \mathbf{C}_j^{1,m} \end{bmatrix}, \tag{51}$$

with the block matrices $\mathbf{C}_j^{1,i} \in \mathbb{R}^{2\times 2}$ given by,

$$\mathbf{C}_j^{1,i} = \begin{bmatrix} 0 & \hat{\sigma}(\mathbf{c}^{1,i}) \\ -\hat{\sigma}(\mathbf{c}^{1,i})\left(\mathbf{w}^{1,i}\sigma'(\mathbf{A}_{j-1}^{1,i}) + \alpha\right) & 0 \end{bmatrix}. \tag{52}$$

By a straightforward calculation and the use of induction, we claim that

$$\prod_{j=k+1}^{n} \frac{\partial \mathbf{X}_j^1}{\partial \mathbf{X}_{j-1}^1} = \mathbf{I}_{2m\times 2m} + \Delta t \mathbf{C}^1 + \mathcal{O}(\Delta t^2), \tag{53}$$

with

$$\mathbf{C}^1 = \begin{bmatrix} \mathbf{C}^{1,1} & 0 & \dots & 0 \\ 0 & \mathbf{C}^{1,2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \mathbf{C}^{1,m} \end{bmatrix}, \tag{54}$$

with the block matrices $\mathbf{C}^{1,i} \in \mathbb{R}^{2\times 2}$ given by,

$$\mathbf{C}^{1,i} = \begin{bmatrix} 0 & (n-k)\hat{\sigma}(\mathbf{c}^{1,i}) \\ -(n-k)\alpha\hat{\sigma}(\mathbf{c}^{1,i}) - \hat{\sigma}(\mathbf{c}^{1,i})\mathbf{w}^{1,i}\sum\limits_{j=k+1}^{n}\sigma'(\mathbf{A}_{j-1}^{1,i}) & 0 \end{bmatrix}. \tag{55}$$

By the assumption that $k << n$ and using the fact that $t_n = n\Delta t$, we have that,

$$\Delta t \mathbf{C}^{1,i} = \begin{bmatrix} 0 & t_n\hat{\sigma}(\mathbf{c}^{1,i}) \\ -t_n\alpha\hat{\sigma}(\mathbf{c}^{1,i}) - \hat{\sigma}(\mathbf{c}^{1,i})\mathbf{w}^{1,i}\Delta t\sum\limits_{j=k+1}^{n}\sigma'(\mathbf{A}_{j-1}^{1,i}) & 0 \end{bmatrix}. \tag{56}$$

Hence, the non-zero entries in the block matrices can be $\mathcal{O}(1)$. Therefore, the product formula (53) is modified to,

$$\prod_{j=k+1}^{n} \frac{\partial \mathbf{X}_j^1}{\partial \mathbf{X}_{j-1}^1} = \mathbf{C} + \mathcal{O}(\Delta t), \tag{57}$$

with the $2m \times 2m$ matrix $\mathbf{C}$ defined as,

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}^{1} & 0 & \dots & 0 \\ 0 & \mathbf{C}^{2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \mathbf{C}^{m} \end{bmatrix}, \tag{58}$$

and,

$$\mathbf{C}^{i} = \begin{bmatrix} 1 & t_n\hat{\sigma}(\mathbf{c}^{1,i}) \\ -t_n\alpha\hat{\sigma}(\mathbf{c}^{1,i}) - \hat{\sigma}(\mathbf{c}^{1,i})\mathbf{w}^{1,i}\Delta t\sum\limits_{j=k+1}^{n}\sigma'(\mathbf{A}_{j-1}^{1,i}) & 1 \end{bmatrix}. \tag{59}$$

Thus by taking the product of (57) with (48), we obtain that,

$$\prod_{j=k+1}^{n} \frac{\partial \mathbf{X}_j^1}{\partial \mathbf{X}_{j-1}^1}\frac{\partial^+ \mathbf{X}_k^1}{\partial \mathbf{w}^{1,p}} = \left[0, 0, \dots, \dots, \frac{\partial^+ \mathbf{y}_k^{1,p}}{\partial \mathbf{w}^{1,p}} + \mathbf{C}_{12}^p\frac{\partial^+ \mathbf{z}_k^{1,p}}{\partial \mathbf{w}^{1,p}}, \mathbf{C}_{21}^p\frac{\partial^+ \mathbf{y}_k^{1,p}}{\partial \mathbf{w}^{1,p}} + \frac{\partial^+ \mathbf{z}_k^{1,p}}{\partial \mathbf{w}^{1,p}} \dots, \dots, 0, 0\right]^\top + \mathcal{O}(\Delta t^2), \tag{60}$$

with $\mathbf{C}^p_{12}, \mathbf{C}^p_{21}$ are the off-diagonal entries of the corresponding block matrix, defined in (59). Note that the $\mathcal{O}(\Delta t^2)$ remainder term arises from the Δt-dependence in (48).

From (18), we have that,

$$\frac{\partial \mathcal{E}_n}{\partial \mathbf{X}^1_n} = \left[\mathbf{y}^{1,1}_n - \overline{\mathbf{y}}^1_n, 0, \dots, \mathbf{y}^{1,i}_n - \overline{\mathbf{y}}^i_n, 0, \dots, \mathbf{y}^{1,m}_n - \overline{\mathbf{y}}^m_n, 0\right]. \tag{61}$$

Therefore, taking the products of (61) and (60) and substituting the explicit expressions in (48), we obtain the desired identity (46). □

C.4. On the vanishing gradient problem for the multilayer version of UnICORNN.

The explicit representation formula (46) holds for 1 hidden layer in (6). What happens when additional hidden layers are stacked together as in UnICORNN (6)? To answer this question, we consider the concrete case of $L = 3$ layers as this is the largest number of layers that we have used in the context of UnICORNN with fully connected stacked layers. As before, we set the scalar parameter $\theta = \mathbf{w}^{1,p}$ for some $1 \le p \le m$. Similar results also hold for any other $\theta \in \mathbf{\Theta}$. We have the following representation formula for the gradient in this case,

Proposition C.4. *Let $\mathbf{y}_n$ be the hidden states generated by the RNN (6). The gradient for long-term dependencies satisfies the representation formula,*

$$\frac{\partial \mathcal{E}^{(n,3)}_{k,1}}{\partial \mathbf{w}^{1,p}} = \Delta t^4 \hat{\sigma}(\mathbf{c}^{1,p}) t_n \frac{\partial^+ \mathbf{z}^{1,p}_k}{\partial \mathbf{w}^{1,p}} \sum_{i=1}^m \bar{\mathbf{G}}_{2i-1,2p-1}\left(\mathbf{y}^{3,i} - \overline{\mathbf{y}}^i\right) + \mathcal{O}(\Delta t^6), \tag{62}$$

with the coefficients given by,

$$\begin{aligned} &\frac{\partial^+ \mathbf{z}^{1,p}_k}{\partial \mathbf{w}^{1,p}} = -\Delta t \hat{\sigma}(\mathbf{c}^{1,p}) \sigma'(\mathbf{A}^{1,p}_{k-1}) \mathbf{y}^{1,p}_{k-1}, \\ &\bar{\mathbf{G}}_{2i-1,2p-1} = \sum_{j=1}^m \mathbf{G}^3_{ij} \mathbf{G}^2_{jp}, \quad \forall\, 1 \le i \le m, \quad \mathbf{G}^q_{r,s} = -(\hat{\sigma}(\mathbf{c}^{q,r}))^2 \sigma'\left(\mathbf{A}^{q,r}_{n-1}\right) \mathbf{V}^q_{rs}, \quad q = 2,3. \end{aligned} \tag{63}$$

Proof. Following the definition (44) and as $L = 3$ and $\theta = \mathbf{w}^{1,p}$, we have,

$$\frac{\partial \mathcal{E}^{(n,3)}_{k,1}}{\partial \mathbf{w}^{1,p}} := \frac{\partial \mathcal{E}_n}{\partial \mathbf{X}^3_n} \frac{\partial \mathbf{X}^3_n}{\partial \mathbf{X}^1_k} \frac{\partial^+ \mathbf{X}^1_k}{\partial \mathbf{w}^{1,p}}. \tag{64}$$

We will explicitly compute all three expressions on the right-hand-side of (64).

In (48), we have already explicitly computed the right most expression in the RHS of (64). Using the product rule (19) we have,

$$\frac{\partial \mathbf{X}^3_n}{\partial \mathbf{X}^1_k} = \frac{\partial \mathbf{X}^3_n}{\partial \mathbf{X}^2_n} \frac{\partial \mathbf{X}^2_n}{\partial \mathbf{X}^1_n} \prod_{j=k+1}^n \frac{\partial \mathbf{X}^1_j}{\partial \mathbf{X}^1_{j-1}}. \tag{65}$$

Note that we have already obtained an explicit representation formula for $\prod\limits_{j=k+1}^n \frac{\partial \mathbf{X}^1_j}{\partial \mathbf{X}^1_{j-1}}$ in (57).

Next we consider the matrices $\frac{\partial \mathbf{X}^3_n}{\partial \mathbf{X}^2_n}$ and $\frac{\partial \mathbf{X}^2_n}{\partial \mathbf{X}^1_n}$. By the representation formula (22), we have the following decomposition for any $1 \le q \le n$,

$$\frac{\partial \mathbf{X}^q_n}{\partial \mathbf{X}^{q-1}_n} = \Delta t^2 \mathbf{G}^{q,n} + \Delta t H^{q,n}, \tag{66}$$

with,

$$\mathbf{G}^{q,n} = \begin{bmatrix} \mathbf{G}^{q,n}_{11} & 0 & \mathbf{G}^{q,n}_{12} & 0 & \dots & \dots & \mathbf{G}^{q,n}_{1m} & 0 \\ 0 & 0 & 0 & 0 & \dots & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \mathbf{G}^{q,n}_{m1} & 0 & \mathbf{G}^{q,n}_{m2} & 0 & \dots & \dots & \mathbf{G}^{q,n}_{mm} & 0 \\ 0 & 0 & 0 & 0 & \dots & \dots & 0 & 0 \end{bmatrix}, \quad \mathbf{G}^{q,k}_{i,\bar{i}} = -(\hat{\sigma}(\mathbf{c}^{q,i}))^2 \sigma'\left(\mathbf{A}^{q,i}_{n-1}\right) \mathbf{V}^q_{i\bar{i}}, \tag{67}$$

and

$$\mathbf{H}^{q,n} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & \dots & 0 & 0 \\ \mathbf{H}^{q,n}_{11} & 0 & \mathbf{H}^{q,n}_{12} & 0 & \dots & \dots & \mathbf{H}^{q,n}_{1m} & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \dots & 0 & 0 \\ \mathbf{H}^{q,n}_{m1} & 0 & \mathbf{H}^{q,n}_{m2} & 0 & \dots & \dots & \mathbf{H}^{q,n}_{mm} & 0 \end{bmatrix}, \quad \mathbf{H}^{q,k}_{i,\bar{i}} = -\hat{\sigma}(\mathbf{c}^{q,i})\sigma'\left(\mathbf{A}^{q,i}_{n-1}\right)\mathbf{V}^{q}_{i\bar{i}}. \tag{68}$$

It is straightforward to see from (68) and (67) that,

$$\mathbf{H}^{3,n}\mathbf{H}^{2,n} \equiv \mathbf{0}_{2m\times 2m}, \quad \mathbf{G}^{3,n}\mathbf{H}^{2,n} \equiv \mathbf{0}_{2m\times 2m}, \tag{69}$$

and the entries of the $2m \times 2m$ matrix $\bar{\mathbf{G}} = \mathbf{G}^{3,n}\mathbf{G}^{2,n}$ are given by,

$$\bar{\mathbf{G}}_{2r-1,2s-1} = \sum_{j=1}^{m} \mathbf{G}^{3,n}_{r,j}\mathbf{G}^{2,n}_{j,s}, \quad \bar{\mathbf{G}}_{2r-1,2s} = \bar{\mathbf{G}}_{2r,2s-1} = \bar{\mathbf{G}}_{2r,2s} = 0, \quad \forall\, 1 \le r,s \le m, \tag{70}$$

while the entries of the $2m \times 2m$ matrix $\bar{\mathbf{H}} = \mathbf{H}^{3,n}\mathbf{G}^{2,n}$ are given by

$$\bar{\mathbf{H}}_{2r,2s-1} = \sum_{j=1}^{m} \mathbf{H}^{3,n}_{r,j}\mathbf{G}^{2,n}_{j,s}, \quad \bar{\mathbf{H}}_{2r-1,2s-1} = \bar{\mathbf{H}}_{2r-1,2s} = \bar{\mathbf{H}}_{2r,2s} = 0, \quad \forall\, 1 \le r,s \le m. \tag{71}$$

Hence we have,

$$\frac{\partial \mathbf{X}^3_n}{\partial \mathbf{X}^2_n}\frac{\partial \mathbf{X}^2_n}{\partial \mathbf{X}^1_n} = \Delta t^4(\bar{\mathbf{G}} + \Delta t^{-1}\bar{\mathbf{H}}). \tag{72}$$

Taking the matrix-vector product of (72) with (60), we obtain

$$\begin{aligned} \frac{\partial \mathbf{X}^3_n}{\partial \mathbf{X}^1_k}\frac{\partial^+ \mathbf{X}^1_k}{\partial \mathbf{w}^{1,p}} &= \Delta t^4\left(\frac{\partial^+ \mathbf{y}^{1,p}_k}{\partial \mathbf{w}^{1,p}} + \mathbf{C}^p_{12}\frac{\partial^+ \mathbf{z}^{1,p}_k}{\partial \mathbf{w}^{1,p}}\right)\left[\bar{\mathbf{G}}_{1,2p-1}, \Delta t^{-1}\bar{\mathbf{H}}_{2,2p-1}, \dots, \bar{\mathbf{G}}_{2m-1,2p-1}, \Delta t^{-1}\bar{\mathbf{H}}_{2m,2p-1}\right]^\top + \mathcal{O}(\Delta t^6) \\ &= \Delta t^4\mathbf{C}^p_{12}\frac{\partial^+ \mathbf{z}^{1,p}_k}{\partial \mathbf{w}^{1,p}}\left[\bar{\mathbf{G}}_{1,2p-1}, \Delta t^{-1}\bar{\mathbf{H}}_{2,2p-1}, \dots, \bar{\mathbf{G}}_{2m-1,2p-1}, \Delta t^{-1}\bar{\mathbf{H}}_{2m,2p-1}\right]^\top + \mathcal{O}(\Delta t^6), \end{aligned} \tag{73}$$

where the last identify follows from the fact that $\frac{\partial^+ \mathbf{y}^{1,p}_k}{\partial \mathbf{w}^{1,p}} = \mathcal{O}(\Delta t^2)$.

Therefore, taking the products of (61) and (73), we obtain the desired identity (62).

□

An inspection of the representation formula (62) shows that as long as the weights are $\mathcal{O}(1)$ and from the bounds (7), we know that $\mathbf{y} \sim \mathcal{O}(1)$, the gradient

$$\frac{\partial \mathcal{E}^{(n,3)}_{k,1}}{\partial \mathbf{w}^{1,p}} \sim \mathcal{O}(\Delta t^5),$$

where the additional Δt stems from the Δt-term in (48). Thus the gradient does not depend on the recurrent step k. Hence, there is no vanishing gradient problem with respect to the number of recurrent connections, even in the multi-layer case.

However, it is clear from the representation formulas (46) and (62), as well as the proof of proposition C.4 that for L-hidden layers in UnICORNN (6), we have,

$$\frac{\partial \mathcal{E}^{(n,L)}_{k,1}}{\partial \mathbf{w}^{1,p}} \sim \mathcal{O}\left(\Delta t^{2L-1}\right). \tag{74}$$

Thus, the gradient can become very small if too many layers are stacked together. This is not at all surprising as such a behavior occurs even if there are no recurrent connections in UnICORNN (6). In that case, we simply have a fully connected deep neural network and it is well-known that the gradient can vanish as the number of layers increases, making it harder to train deep networks.

C.5. Residual stacking of layers in UnICORNN.

Given the above considerations, it makes imminent sense to modify the fully-connected stacking of layers in UnICORNN (6) if a moderately large number of layers ($L \geq 4$) are used. It is natural to modify the fully-connected stacking with a residual stacking, see (Li et al., 2019). We use the following form of residual stacking,

$$\mathbf{y}_n^\ell = \mathbf{y}_{n-1}^\ell + \Delta t \hat{\sigma}(\mathbf{c}^\ell) \odot \mathbf{z}_n^\ell, \tag{75}$$
$$\mathbf{z}_n^\ell = \mathbf{z}_{n-1}^\ell - \Delta t \hat{\sigma}(\mathbf{c}^\ell) \odot [\sigma\left(\mathbf{w}^\ell \odot \mathbf{y}_{n-1}^\ell + \mathbf{x}_n^\ell + \mathbf{b}^l\right) + \alpha \mathbf{y}_{n-1}^\ell], \tag{76}$$

where the input $\mathbf{x}_n^\ell$ corresponds to a residual connection skipping S layers, i.e.

$$\mathbf{x}_n^\ell = \begin{cases} \mathbf{\Lambda}^\ell \mathbf{y}_n^{\ell-S-1} + \mathbf{V}^\ell \mathbf{y}_n^{\ell-1}, & \text{for } l > S \\ \mathbf{V}^\ell \mathbf{y}_n^{\ell-1}, & \text{for } l \leq S \end{cases}.$$

The number of skipped layers is $2 \leq S$ and $\Lambda^\ell \in \mathbb{R}^{m \times m}$ is a trainable matrix.

The main advantages of using a residual staking such as (75) is to alleviate the vanishing gradient problem that arises from stacking multiple layers together and obtain a better scaling of the gradient than (74). To see this, we can readily follow the proof of proposition C.4, in particular the product,

$$\frac{\partial \mathbf{X}_n^L}{\partial \mathbf{X}_n^1} = \prod_{s=1}^{\nu} \frac{\partial \mathbf{X}_n^{L-(s-1)S}}{\partial \mathbf{X}_k^{L-sS}} \prod_{\ell=2}^{L-\nu S} \frac{\partial \mathbf{X}_n^\ell}{\partial \mathbf{X}_n^{\ell-1}} + \prod_{\ell=1}^{L-1} \frac{\partial \mathbf{X}_n^{\ell+1}}{\partial \mathbf{X}_k^\ell}, \tag{77}$$

with,

$$\nu = \begin{cases} \left[\frac{L}{S}\right], & \text{if} \quad L \bmod S \neq 0, \\ \left[\frac{L}{S}\right] - 1, & \text{if} \quad L \bmod S = 0. \end{cases} \tag{78}$$

Here $[x] \in \mathbb{N}$ is the largest natural number less than or equal to $x \in \mathbb{R}$.

Given the additive structure in the product of gradients and using induction over matrix products as in (69) and (70), we can compute that,

$$\frac{\partial \mathbf{X}_n^L}{\partial \mathbf{X}_n^1} = \mathcal{O}\left(\Delta t^{2(\nu+L-\nu S-1)}\right) + \mathcal{O}\left(\Delta t^{2(L-1)}\right). \tag{79}$$

By choosing S large enough, we clearly obtain that $\nu + L - \nu S - 1 < L - 1$. Hence by repeating the arguments of the proof of proposition C.4, we obtain that to leading order, the gradient of the residual stacked version of UnICORNN scales like,

$$\frac{\partial \mathcal{E}_{k,1}^{(n,L)}}{\partial \mathbf{w}^{1,p}} \sim \mathcal{O}\left(\Delta t^{2\nu+2L-2\nu S-1}\right). \tag{80}$$

Note that (80) is far more favorable scaling for the gradient than the scaling (74) for a fully connected stacking. As a concrete example, let us consider $L = 7$ i.e., a network of 7 stacked layers of UniCORNN. From (74), we see that the gradient scales like $\mathcal{O}(\Delta t^{13})$ in this case. Even for a very moderate values of $\Delta t < 1$, this gradient will be very small and will ensure that the first layer will have very little, if any, influence on the loss function gradients. On the other hand, for the same number of layers $L = 7$, let us consider the residual stacking (75) with $S = 3$ skipped connections. In this case $\nu = 2$ and one directly concludes from (80) that the gradient scales like $\mathcal{O}(\Delta t^5)$, which is significantly larger than the gradient for the fully connected version of UnICORNN. In fact, it is exactly the same as the gradient scaling for fully connected UnICORNN (6) with 3 hidden layers (62). Thus, introducing skipped connections enabled the gradient to behave like a shallower fully-connected network, while possibly showing the expressivity of a deeper network.

D. Chaotic time-series prediction: Lorenz 96 system

It is instructive to explore limitations of the proposed UnICORNN. It is straightforward to prove, along the lines of the proof of proposition C.1, that the UnICORNN architecture does not exhibit chaotic behavior with respect to changes in the input. While this property is highly desirable for many applications where a slight change in the input should not lead to a major (possibly unbounded) change in the output, it impairs the performance on tasks where an actual chaotic system has to be learned.

Following the experiment in (Rusch & Mishra, 2021), we aim to predict future states of a dynamical system, following the Lorenz 96 system (Lorenz, 1996):

$$x_j' = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \tag{81}$$

where $x_j \in \mathbb{R}$ for all $j = 1, \dots, 5$ and F is an external force controlling the level of chaos in the system.

We consider two different choices for the external force, namely $F = 0.9$ and $F = 8$. While the first one produces non-chaotic trajectories, the latter leads to a highly chaotic system. We discretize the system exactly along the lines of (Rusch & Mishra, 2021), resulting in 128 sequences of length 2000 for each the training, validation and testing set. Table 2 shows the normalized root mean square error (NRMSE) for UnICORNN as well as for coRNN and an LSTM, where all models have 9k parameters. We can see that UnICORNN performs comparably to coRNN and LSTM in the chaos-free regime (i.e. $F = 0.9$), while performing poorly compared to an LSTM when the system exhibits chaotic behavior (i.e. $F = 8$). This is not surprising, as LSTMs are shown to be able to exhibit chaotic behavior (Laurent & von Brecht, 2017), while coRNN and UnICORNN are not chaotic by design. This shows also numerically that UnICORNN should not be used for chaotic time-series prediction.

Table 2. Test NRMSE on the Lorenz 96 system (81) for UnICORNN, coRNN and LSTM.

Model	$F = 0.9$	$F = 8$	# units	# params
LSTM (Rusch & Mishra, 2021)	2.0×10^{-2}	6.8×10^{-2}	44	9k
coRNN (Rusch & Mishra, 2021)	2.0×10^{-2}	9.8×10^{-2}	64	9k
UnICORNN (L=2)	2.2×10^{-2}	3.1×10^{-1}	90	9k

E. Further experimental results

As we compare the results of the UnICORNN to the results of other recent RNN architecture, where only the best results of each RNN were published for the psMNIST, noise padded CIFAR-10 and IMDB task, we as well show the best (based on a validation set) obtained results for the UnICORNN in the main paper. However, distributional results, i.e. statistics of several re-trainings of the best performing UnICORNN based on different random initialization of the trainable parameters, provide additional insights into the performance. Table 3 shows the mean and standard deviation of 10 re-trainings of the best performing UnICORNN for the psMNIST, noise padded CIFAR-10 and IMDB task. We can see that in all experiments the standard deviation of the re-trainings are relatively low, which underlines the robustness of our presented results.

Table 3. Distributional information (mean and standard deviation) on the results for the classification experiment presented in the paper, where only the best results is shown, based on 10 re-trainings of the best performing UnICORNN using different random seeds.

Experiment	Mean	Standard deviation
psMNIST (128 units)	97.7%	0.09%
psMNIST (256 units)	98.2%	0.22%
Noise padded CIFAR-10	61.5%	0.52%
IMDB	88.1%	0.19%

As emphasized in the main paper and in the last section, naively stacking of many layers for the UnICORNN might result in a vanishing gradient for the deep multi-layer model, due to the vanishing gradient problem of stacking many (not necessarily recurrent) layers. Following section C.5, one can use skipped residual connections and we see that the estimate on the gradients scale preferably when using residual connections compared to a naively stacking, when using many layers. To test this also numerically, we train a standard UnICORNN (6) as well as a residual UnICORNN (res-UnICORNN) (75), with $S = 2$ skipping layers, on the noise padded CIFAR-10 task. Fig. 1 shows the test accuracy (mean and standard deviation) of the best resulting model for different number of network layers $L = 3, \dots, 6$, for the standard UnICORNN and res-UnICORNN. We can see that while both models seem to perform comparably for using only few layers, i.e. $L = 3, 4$, the res-UnICORNN with $S = 2$ skipping connections outperforms the standard UnICORNN when using more layers, i.e. $L = 5, 6$. In particular, we can see that the standard UnICORNN is not able to significantly improve the test accuracy when using more layers, while the res-UnICORNN seems to obtain higher test accuracies when using more layers.

Moreover, Fig. 1 also shows the test accuracy for a UnICORNN with an untrained time-step vector **c**, resulting in a UnICORNN without the multi-scale property generated by the time-step. We can see that the UnICORNN without the multi-scale feature is inferior in performance, to the standard UnICORNN as well as its residual counterpart.

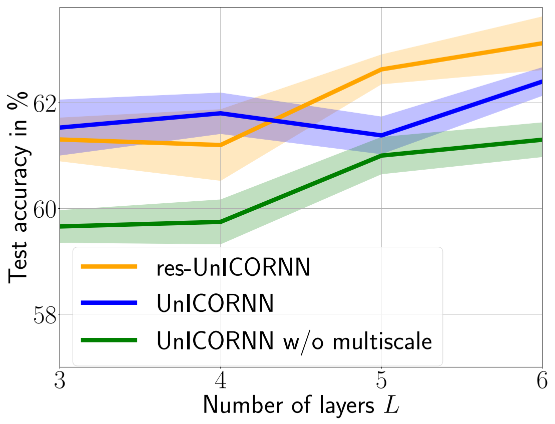

Figure 1. Test accuracies (mean and standard deviation of 10 re-trainings of the best performing model) of the standard UnICORNN, res-UnICORNN and UnICORNN without multi-scale behavior on the noise padded CIFAR-10 experiment for different number of layers L.

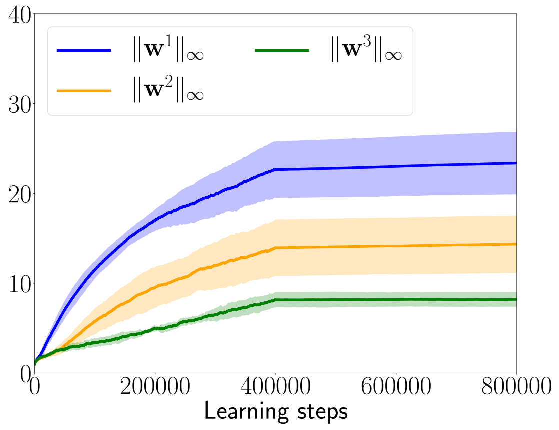

Figure 2. Norms (mean and standard deviation of 10 re-trainings) of the hidden weights $\|\mathbf{w}^l\|_\infty$, for $l = 1, 2, 3$, of the UnICORNN during training.

Finally, we recall that the estimate (14) on the gradients for UnICORNN (6) needs the weights to be bounded, see (15). One always initializes the training with bounded weights. However, it might happen that the weights explode during training. To check this issue, in Fig. 2, we plot the mean and standard deviation of the norms of the hidden weights $\mathbf{w}^l$ for $l = 1, 2, 3$ during training based on 10 re-trainings of the best performing UnICORNN on the noise padded CIFAR-10 experiment. We can see that none of the norms of the weights explode during training. In fact the weight norms seem to saturate, mostly on account of reducing the learning rate after 250 epochs. Thus, the upper bound (14) can be explicitly computed and it is finite, even after training has concluded.

References

Gal, Y. and Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. *Advances in neural information processing systems*, 29:1019–1027, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Laurent, T. and von Brecht, J. A recurrent neural network without chaos. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Li, S., Li, W., Cook, C., Gao, Y., and Zhu, C. Deep independently recurrent neural network (indrnn). *arXiv preprint arXiv:1910.06251*, 2019.

Lorenz, E. N. Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, volume 1, 1996.

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

Rusch, T. K. and Mishra, S. Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*, 2021.

Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.