AFRNN: Stable RNN with Top Down Feedback and Antisymmetry

Tim Schwabe TIM.SCHWABE@RUB.DE
Ruhr University Bochum,
Faculty of Computer Science

Tobias Glasmachers TOBIAS.GLASMACHERS@INI.RUB.DE
Ruhr University Bochum,
Faculty of Computer Science

Maribel Acosta MARIBEL.ACOSTA@RUB.DE
Ruhr University Bochum,
Faculty of Computer Science

Editors: Emtiyaz Khan and Mehmet Gönen

Abstract

Recurrent Neural Networks are an integral part of modern machine learning. They are good at performing tasks on sequential data. However, long sequences are still a challenge for those models due to the well-known exploding/vanishing gradient problem. In this work, we build on recent approaches to interpreting the gradient problem as instability of the underlying dynamical system. We extend previous approaches to systems with top-down feedback, which is abundant in biological neural networks. We prove that the resulting system is stable for arbitrary depth and width and confirm this empirically. We further show that its performance is on par with long short-term memory (LSTM) models and related approaches on standard benchmarks.

Keywords: Recurrent Neural Networks, Gradient Stability, Dynamical Systems

1. Introduction

Recurrent Neural Networks (RNNs) are networks containing at least one circular path of connections, in the simplest case realized as a self-connection. Being stateful models, they are usually applied for processing sequential data. Instead of receiving input once, as feedforward networks do, they are fed with a sequence of inputs over multiple time steps. The vanilla version of a single-layer recurrent neural network that we assume is given by

$$x_t = \sigma(W_{\text{rec}} x_{t-1} + W_{\text{in}} u_t + b) \tag{1}$$

This equation maps an external input u_t at time t and the output of the network from the previous time x_{t-1} through weight matrices W_{rec} and W_{in} together with a bias b to the new output of the system, using a nonlinearity σ. We consider networks consisting of layers of neurons, hence x_t and u_t are vectors (or higher-dimensional tensors).

RNNs find a widespread application and are used for example in machine translation (Wu et al. (2016) and in autonomic driving (Grigorescu et al. (2019)). Due to their

expressive power and their ability to capture dynamic patterns, they can be used to model complex systems from physics and engineering.

A longstanding problem of RNNs is the exploding/vanishing gradient problem (Pascanu et al., 2012). It refers to the fact that for long time horizons, the norm of the gradient either shrinks to very small values or grows to (numerically unstable) large values. The behavior is governed by the eigenspectrum of the weight matrices, with eigenvalues smaller than one corresponding to shrinkage, while eigenvalues larger than one result in an explosion. Thus, finding solutions that mitigate this problem is of crucial importance for the practical use of RNNs. We will discuss several approaches in the next section.

An essential line of thought is the interpretation of an RNN as a time-discretized differential equation. Differential equations describe dynamical systems ranging from physics to biology and finance. The most simple form of a first-order ordinary differential equation (ODE) is given by $\frac{dx}{dt} = f(x,t)$ (Strogatz, 2018). The corresponding forward Euler discretization is (Shampine and Thompson, 2007)

$$x_{n+1} = x_n + h \cdot f(x_n, t_n). \tag{2}$$

By setting $f(x_n, t_n) = \sigma(W_{rec}x_{t-1} + W_{in}u_t + b)$, we can identify this equation as an RNN (with an additional residual connection x_n). A similar formulation can also be found for the vanilla RNN. By analyzing the underlying ODE, we can make statements about the stability of the network and possible modes of behavior. A further option is to use other integration schemes like Runge-Kutta to solve the RNN trajectory through time. A similar approach has been developed by Chen et al. (2018).

Another exciting addition to recurrent neural networks is top-down feedback. This means that recurrent connections do not only appear within each layer but also between layers. More concretely, higher layers project their states back to lower layers through weight matrices. Top-down connections are abundant in the human brain (Sikkens et al., 2019) and have been shown to play an important role in the visual pathways of mice and monkeys (Kar et al., 2019; Pak et al., 2020). In the context of artificial neural networks, top-down feedback has been shown to improve image classification (Zamir et al., 2017) and also improve long short-term memory (LSTM) models on several different benchmarks (Chung et al., 2015). We will come back to top-down feedback in Section 3 when we incorporate it into our model.

The central contribution of this work is the proposal of a novel recurrent network architecture. It combines the following properties:

- provably stable behavior,
- gradients neither vanish nor explode,
- incorporation of top-down feedback,
- flexible connectivity between layers, with a sparse default.

The remainder of this paper is structured as follows. Section 2 discusses related work. Our approach AFRNN is described in Section 3. Then, Section 4 presents an analysis of the dynamics of the system using our proposed approach in comparison to the state of the art.

Then, in Section 5 we show results of training the AFRNN on the pixel-by-pixel MNIST and pixel-by-pixel CIFAR-10 benchmarks as well as on the task of predicting double pendulum trajectories.[1] Lastly, Section 6 presents our conclusions and an outlook to future work.

2. Related Work

Several approaches were proposed for mitigating the exploding/vanishing gradient problem. The most prominent one is the LSTM architecture proposed by Hochreiter and Schmidhuber (1997). The core idea here is to have a memory cell that has a linear self-connection through time with a constant weight of one. This prevents the gradient from changing over time. A simple approach to prevent exploding gradients is to use gradient clipping, which rescales the gradient when it becomes too large (Pascanu et al., 2012).

Another line of thought is regularizing the recurrent weight matrices in specific ways. For example, Arjovksy et al. parameterize the recurrent matrices so that they only represent unitary matrices (Arjovsky et al., 2016). These approaches aim to generate unitary Jacobians of the system so that the gradients do not grow large or shrink significantly over time.

This paper builds on the work by Chang et al. (2019), which suggests that the exploding/vanishing gradient problem can be attributed to the instability of the ODE underlying the RNN model. Informally, a stable ODE has the property that small perturbations of the initial state stay small over time. On the other hand, the gradient of the network with respect to its parameters quantifies the sensitivity of the output to changes in the parameters. Thus, a system with a stable ODE does not yield exploding gradients. However, if the system is asymptotically stable, it may yield vanishing gradients. Therefore, the goal is to create a system on the verge of instability. The following theorem gives the formal condition of stability:

Theorem 1 (Condition for Stability; Chang et al. 2019) *The solution of an ODE is stable if* $\max\left\{\Re\left(\lambda_i\left(J(t)\right)\right)\right\} \leq 0, \forall t \geq 0$, *were* $\Re$ *denotes the real part of a complex number and* λ_i *is the* i*-th eigenvalue of the Jacobian.*

The previous theorem states that the system is stable whenever the real part of the largest eigenvalue of the Jacobian J of the system is smaller than or equal to zero. To mitigate the vanishing/exploding gradient problem, it is thus sensible to design a system where the equality to zero holds approximately. Chang et al. (2019) therefore propose the recurrent weight matrices to be antisymmetric, i.e., to fulfill $\mathbf{M}^T = -\mathbf{M}$. An important fact about antisymmetric matrices is that they only do have imaginary eigenvalues, and hence $\Re(\lambda_i(\mathrm{M})) = 0, \forall\, \mathbf{M} \in \mathbb{R}^{nxn} s.t.\ \mathbf{M}^T = -\mathbf{M}$. Then, Chang et al. (2019) propose the following ODE:

$$\frac{d\mathbf{h}(t)}{dt} = \tanh((\mathbf{W}_h - \mathbf{W}_h^T)\mathbf{h}(t) + \mathbf{V}_h\mathbf{x}(t) + \mathbf{b}_h). \tag{3}$$

Using explicit Euler discretization, the derived *antisymmetric RNN* (ARNN) is given as

1. The code to reproduce the visualizations and experiments of our work can be found in the GitHub repository at https://github.com/TimEricSchwabe/afrnn_code.

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \epsilon \cdot \tanh((\mathbf{W}_h - \mathbf{W}_h^T)\mathbf{h}_{t-1} + \mathbf{V}_h\mathbf{x}(t) + \mathbf{b}_h). \tag{4}$$

This RNN is very similar to the vanilla RNN, except that it incorporates a "skip connection" $\mathbf{h}_{t-1}$, a parameter ϵ that denotes the step size of the explicit Euler step, and lastly and most importantly, it has an antisymmetric recurrent weight matrix $\mathbf{W}_h - \mathbf{W}_h^T$. We refer the reader to Chang et al. (2019) for a proof of the stability of the ODE.

Despite that the ODE of this system is stable, the discretized system might not be. More precisely, a system solved using explicit Euler is stable if $\max|1 + \epsilon\lambda_i(\mathbf{J}_t)| \leq 1$. Here, $\mathbf{J}_t$ is the Jacobian of the system at timestep t. Hence, to make the discretized version stable, Chang et al. (2019) suggest adding "diffusion" to the system; they subtract a small number γ from the recurrent weight matrix and obtain the following system:

$$\mathbf{h}_t = \mathbf{h}_{t-1} + \epsilon \cdot \tanh((\mathbf{W}_h - \mathbf{W}_h^T - \gamma\mathbf{I})\mathbf{h}_{t-1} + \mathbf{V}_h\mathbf{x}(t) + \mathbf{b}_h). \tag{5}$$

The authors show empirically that this system exhibits stable dynamics. They further show that this model exceeds the performance of the LSTM and other models that regularize the recurrent weight matrices on the *pixel-by-pixel MNIST* and *pixel-by-pixel CIFAR-10* benchmark problems.

These are promising results, informed by theoretical considerations. Moreover, they pave a way towards better RNNs. Our goal is now to extend the ARNN to a system with multiple layers that does incorporate top-down feedback between the layers. In the next section, we will describe this system in detail and give stability guarantees.

3. Antisymmetric Feedback RNN

We previously mentioned that top-down feedback is a promising direction for making RNNs more expressive. Our central research question is how to incorporate top-down feedback into an RNN and still guarantee a stable system. To this end, we propose the **Antisymmetric Feedback RNN** (AFRNN). The system for 2 layers is exemplary shown in Figure 1.

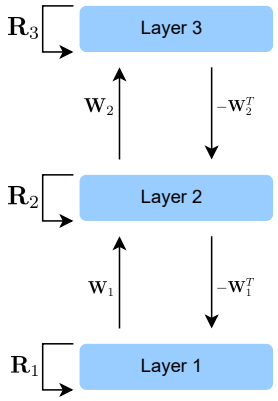

Figure 1: Sketch of an AFRNN with three layers. External input and bias are omitted for readibility.

We will first describe this system with two recurrent layers and afterward describe the general system for an arbitrary number of layers. The following set of equations defines a two-layer AFRNN:

$$\mathbf{g}_{t+1} = \mathbf{g}_t + \epsilon \tanh((\mathbf{R} - \mathbf{R}^T)\mathbf{g}_t + \mathbf{W}\mathbf{h}_t + \mathbf{E}\mathbf{x}_t + \mathbf{b}^{\{g\}}) \tag{6}$$

$$\mathbf{h}_{t+1} = \mathbf{h}_t + \epsilon \tanh(-\mathbf{W}^T\mathbf{g}_t + (\mathbf{S} - \mathbf{S}^T)\mathbf{h}_t + \mathbf{b}^{\{h\}}) \tag{7}$$

Those equations define two connected recurrent layers. A bottom layer $\mathbf{g}$ that receives external input $\mathbf{x}$ and a top layer $\mathbf{h}$ that receives input from the bottom layer through the feedforward connection matrix $\mathbf{W} \in \mathbb{R}^{D(\mathbf{h})\times D(\mathbf{g})}$. Here, let $D(\mathbf{g})$ and $D(\mathbf{h})$ be the dimensionalities of layer $\mathbf{g}$ and layer $\mathbf{h}$. $\mathbf{R} - \mathbf{R}^T \in \mathbb{R}^{D(\mathbf{g})\times D(\mathbf{g})}$ and $\mathbf{S} - \mathbf{S}^T \in \mathbb{R}^{D(\mathbf{h})\times D(\mathbf{h})}$ are the recurrent matrices for layer $\mathbf{g}$ and $\mathbf{h}$ and are both antisymmetric, similar to the ARNN. We add a connection by which the bottom layer receives feedback from the top layer, mediated by the top-down feedback matrix $-\mathbf{W}^T$. This feedback matrix is defined to be the negative transpose of the feedforward matrix $\mathbf{W}$ that projects from $\mathbf{g}$ to $\mathbf{h}$. This symmetry is crucial for making the overall system stable, as we will show later. The matrix $\mathbf{E} \in \mathbb{R}^{D(\mathbf{g})\times D(\mathbf{x})}$ is the matrix mapping external input $\mathbf{x}$ to layer $\mathbf{g}$. Lastly, $\mathbf{b}^{\{g\}}$ and $\mathbf{b}^{\{h\}}$ are bias vectors for the two layers. The underlying ODEs of the two layers are given by

$$\frac{d\mathbf{g}}{dt} = \tanh((\mathbf{R} - \mathbf{R}^T)\mathbf{g} + \mathbf{W}\mathbf{h} + \mathbf{E}\mathbf{x} + \mathbf{b}^{\{g\}}) \tag{8}$$

$$\frac{d\mathbf{h}}{dt} = \tanh(-\mathbf{W}^T\mathbf{g} + (\mathbf{S} - \mathbf{S}^T)\mathbf{h} + \mathbf{b}^{\{h\}}). \tag{9}$$

To prove the stability of this system, we start with the following lemma.

Lemma 2 (Chang et al. 2019) *If $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is an antisymmetric matrix and $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is an invertible diagonal matrix, then the eigenvalues of $\boldsymbol{DW}$ are imaginary.*

$$\Re(\lambda_i(\boldsymbol{DW})) = 0, \quad \forall i = 1, 2, ..., n. \tag{10}$$

Now, the generalization of the AFRNN to N arbitrarily sized layers is given by the following theorem.

Theorem 3 (Stability of the AFRNN) *Consider the system of coupled ODEs*

$$\dot{\mathbf{g}}^{\{n\}} = \frac{d\mathbf{g}^{\{n\}}}{dt} = \tanh\left(\left[\sum_{k=1}^{N} \mathbf{W}^{\{kn\}}\mathbf{g}^{\{k\}}\right] + \mathbf{E}^{\{n\}}\mathbf{x} + \mathbf{b}^{\{n\}}\right), \quad i \in [1, N]. \tag{11}$$

Assume $\mathbf{g}^{\{n\}} \in \mathbb{R}^{D(n)}$, where $D(n) \in \mathbb{N}$ denotes the dimensionality of the n-th layer and $\mathbf{W}^{\{kn\}} \in \mathbb{R}^{D(k)\times D(n)}$ denotes the matrix from layer k to layer n. $\mathbf{E}^{\{n\}} \in \mathbb{R}^{D(n)}$ denotes the matrix mapping external input to layer n. Therefore it holds $\forall n \neq 1, \mathbf{E}^{\{n\}} = \mathbf{0}$. $\mathbf{b}^{\{n\}}$ is the bias for layer n.
This system is stable if all $\mathbf{W}^{\{nn\}}$ are antisymmetric and if $\mathbf{W}^{\{kn\}} = -\mathbf{W}^{\{nk\}^T}$.

Proof The Jacobian matrix $\boldsymbol{J}$ of the overall system is given by a $\sum_{n=1}^{N} D(n) \times \sum_{m=1}^{N} D(m)$ matrix that consists of $N \times N$ block matrices. The block-matrix $\boldsymbol{P}^{nm}$ at position nm, where $n \in 1, ..., N, m \in 1, ..., N$ is given by

$$P_{ij}^{\{nm\}} = \frac{\partial \dot{g}_i^{\{n\}}}{\partial g_j^{\{m\}}} = W_{ij}^{\{mn\}} \tanh' \left(\left[\sum_{k=1}^{N} \boldsymbol{W}_i^{\{kn\}} \mathbf{g}^{\{k\}} \right] + \mathbf{E}_i^{\{n\}} \mathbf{x} + b_i^{\{n\}} \right). \tag{12}$$

Here, $\boldsymbol{W}_i^{\{kn\}}$ denotes the i-th row-vector of the matrix $\boldsymbol{W}^{\{kn\}}$ and $\tanh'$ is the derivative of the tanh function.

We can split $\mathbf{J}$ into the product of a diagonal and a block matrix like

$$\boldsymbol{J} = \boldsymbol{D}\boldsymbol{B}. \tag{13}$$

The elements of the diagonal matrix $\boldsymbol{D}$ are given by

$$D_{ll} = \tanh' \left(\left[\sum_{k=1}^{N} \boldsymbol{W}_{\chi(l)}^{\{kf\}} \mathbf{g}^k \right] + \mathbf{E}_{\chi(l)}^{\{f\}} + b_{\chi(l)}^{\{f\}} \right), \quad \text{so that } f \text{ fulfills} \sum_{i=0}^{f-1} D_i < l \leq \sum_{i=0}^{f} D_i \tag{14}$$

The function $\chi(k)$ calculates the proper index of the current neuron for each block and is given by

$$\chi(l) = \left(|\sum_{i=0}^{f-1} D_i - l| \right). \tag{15}$$

Here we choose the convention $D_0 = 0$. Details about matrix $\mathbf{D}$ are given in Appendix A.

The matrix $\boldsymbol{B}$ consists of $N \times N$ blocks and the block at position nm is simply given by $\boldsymbol{W}^{\{mn\}}$. Because of Lemma 2, we know that if $\boldsymbol{B}$ is antisymmetric, the eigenvalues of $\boldsymbol{J}$ are all imaginary. This is exactly the case when the block-matrix at position nm equals the negative transpose of the block-matrix at position mn. Thus, it must hold that

$$\boldsymbol{W}^{\{ij\}} = -\boldsymbol{W}^{\{ji\}^T}. \tag{16}$$

Note that this leads to all $\boldsymbol{W}^{\{ii\}}$ being antisymmetric. Under this restrictions on $\boldsymbol{B}$, the eigenvalues of $\boldsymbol{J}$ are all imaginary and by Proposition 1 the system is stable. ■

One important thing to note about the theorem is that $\mathbf{W}^{\{ij\}} = -\mathbf{W}^{\{ji\}^T}$ is required. As mentioned above, this means that the feedback matrix between two layers equals the negative transpose of the feedforward matrix between those layers. This condition is especially fulfilled by the zero matrix. This means that we only need feedback projections between consecutive layers. Because the weights are shared between feedforward and feedback matrices, the architecture does not have any additional parameters compared to a vanilla RNN with multiple layers. That is, the number of parameters grows only linear with increasing depth.

4. Analysis of the Dynamics of the System

After presenting the foundation of the AFRNN and showing its stability properties analytically, we will investigate its behavior and performance empirically. In Section 4.1, we analyze the trajectories of the system visually to demonstrate its stable behavior. In Section 4.2, we investigate the Jacobian of the system directly. Similar to Chang et al. (2019), we also introduce diffusion into the model to ensure stability after discretization. The resulting model thus looks like

$$\frac{d\mathbf{g}}{dt} = \sigma((\mathbf{R} - \mathbf{R}^T - \gamma\mathbf{I})\mathbf{g}_t - (\mathbf{W} - \gamma\mathbf{I})^T\mathbf{h}_t + \mathbf{E}\mathbf{x}_t + \mathbf{b}^g) \tag{17}$$

$$\frac{d\mathbf{h}}{dt} = \sigma((\mathbf{W} - \gamma\mathbf{I})\mathbf{g}_t + (\mathbf{S} - \mathbf{S}^T - \gamma\mathbf{I})\mathbf{h}_t + \mathbf{b}^g) \tag{18}$$

4.1. Trajectories of the System

A stable system is expected to exhibit predictable trajectories. We will thus visualize the values of different neurons of the system over time. In Figure 2, we can see the different trajectories of neurons. Figure 2(a) shows neurons of the upper layer of a ARNN with 2 layers (2-ARNN). We can see that the values of the neurons are not bounded but grow unbounded over time, although linearly. Figure 2(b) shows the trajectories of an AFRNN with two layers. As we can see, the trajectories remain bounded and stable over time. This empirically underlies the above stability theorem. Using the Feedback Connection, we can keep the system stable and bounded, preventing gradients from exploding.

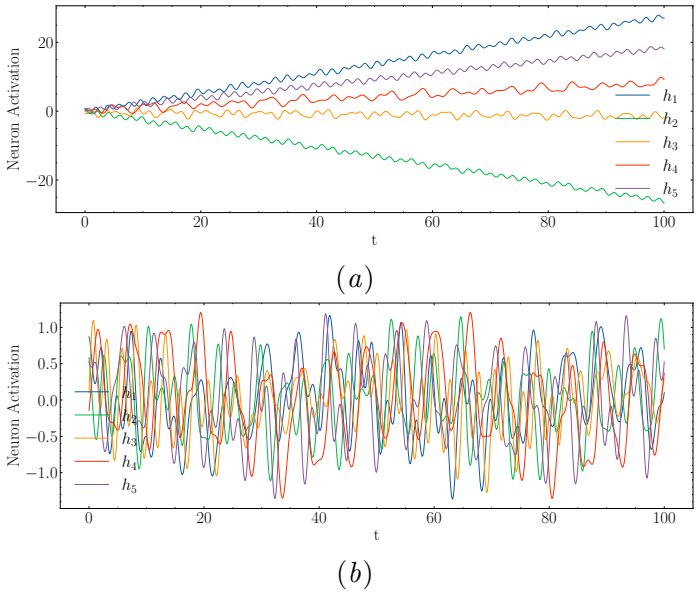

Figure 2: Trajectories of neural activations from the upper layer of (a) 2-ARNN and (b) AFRNN. The dynamics of the AFRNN remain bounded and stable.

Figure 3 visualizes the dynamics of the 2-ARNN and AFRNN once more. Visualized are the state-space trajectories of two neurons of the top layer and bottom layer for two slightly different initial conditions. The trajectories of the 2-ARNN diverge in opposite directions (cf. Figure 3(a)). Although the system is analytically stable, since the proof in Chang et al. (2019) is valid for external input (and the input from the bottom layer represents external input), the periodicity of the bottom layer leads to the divergent behavior. This may still lead to either exploding gradients (as the differences become large) or even to vanishing gradients as the large values may saturate the tanh nonlinearities.

The trajectories for the AFRNN, on the other hand, stay close to each other and remain bounded, as can be seen in Figure 3(b).

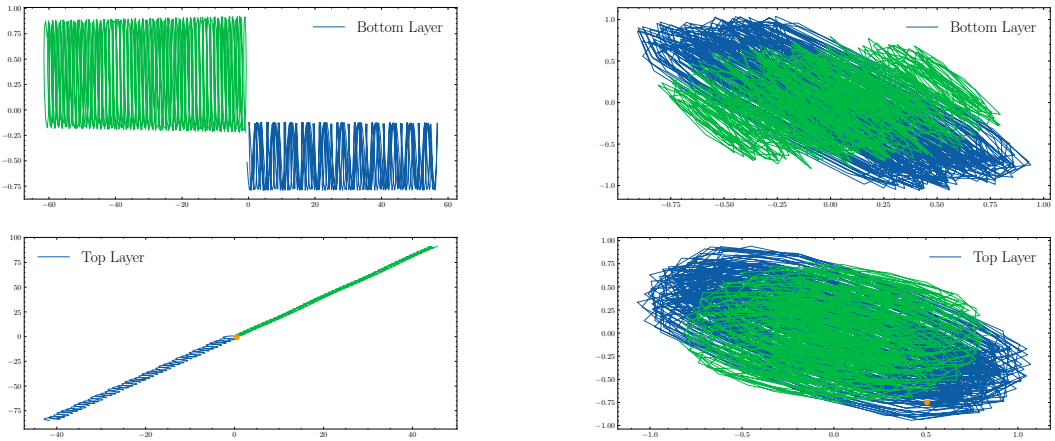

Figure 3: Trajectories for different initial conditions for (a) 2-ARNN and (b) AFRNN. The different trajectories stay close to each other for the AFRNN, and diverge for the 2-ARNN.

4.2. Jacobian of the System

We also investigate the stability of our network by inspecting the gradients directly. Therefore, we calculate the Jacobian matrix $\frac{dg_i^T}{dg_j^t}$ over time, i.e. for every timestep $t \in 0, .., T-1$. g_i^t refers to the state of the i-th neuron in layer g at timestep t. The eigenvalues of this matrix indicate the behavior of the system. If all absolute values of the eigenvalues are close to one, the system does not suffer significantly from the vanishing/exploding gradient problem. Figure 4 shows the average norm of the eigenvalue of the Jacobians of the different networks over time. As we can see, the eigenvalues of the LSTM quickly decays to zero. This is in line with the results of Chang et al. (2019). We also observe that the trajectories of the original ARNN and the ARNN with two layers (2-ARNN) are very similar. This is in line with the theoretical finding that the ARNN is also stable in the presence of external input (as which the input of the first layer can be interpreted). More interestingly, it is

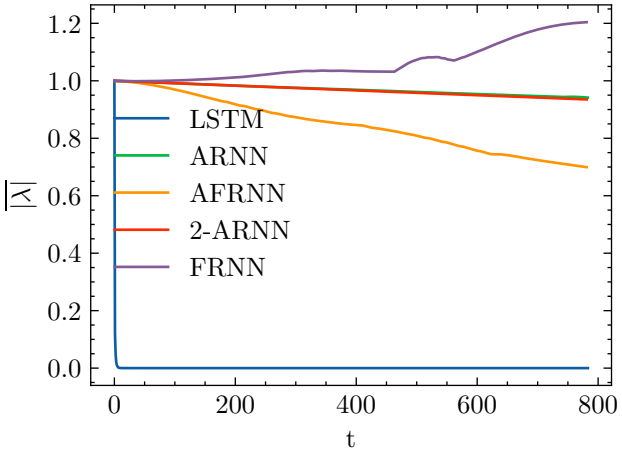

Figure 4: Mean absolute value of the Jacobian of the system over time.

apparent that the AFRNN is empirically stable since the mean eigenvalue stays below 1. The decreasing slope is more significant than for the ARNN but still exhibits linear decay which enables long-term learning. Contrary, the FRNN (AFRNN without the constraint $\boldsymbol{W}^{\{ij\}} \stackrel{!}{=} -\boldsymbol{W}^{\{ji\}^T}$) is not stable, as the values quickly grow larger than 1. These results support the analytical findings of the stability of the AFRNN.

5. Experiments

To evaluate whether the constrained feedback is beneficial in learning long-term dependencies, we compared the AFRNN, ARNN, FRNN and 2-ARNN on two popular benchmarks for RNNs, namely the *pixel-by-pixel MNIST* (Section 5.1) and the *pixel-by-pixel cifar10* (Section 5.2) problems. In both problems, each image of the dataset is flattened into one long vector, which is sequentially fed to the system. In case of *MNIST* the system receives one single pixel per timestep, which results in 784 timesteps in total. The goal of the network is to predict the class of the image in the last time step. To solve this problem with high accuracy, long-term dependencies need to be modeled. In addition, we also compare the architectures on the task of predicting trajectories of a double-pendulum (Section 5.3).[2]

5.1. Pixel-by-Pixel MNIST

The MNIST dataset (Deng (2012)) consists of $60,000$ images for training and $10,000$ images for validation. All images are greyscaled. Each image is represented as a 28×28 matrix, which is flattened before being fed to the model. Each image shows a handwritten digit from 0 to 9. The different tested models consist of the recurrent part (ARNN, 2-ARNN, FRNN, AFRNN), an input-to-hidden projection matrix, and hidden-to-output projection matrix. Finally, a softmax function is applied to the output layer. The result is a 10-dimensional vector, which one-hot encodes the digit shown in the image. For all networks, the hyperparameters are set to $\epsilon = 0.01$ and $\gamma = 0.001$, which were found through manual

2. The link to the source code to reproduce our experiments is provided in Footnote 1 above.

Model	Accuracy	Loss	Hidden Units	Parameters
2-ARNN	91.9%	0.26	128	50826
FRNN	89.1%	0.36	128	67210
AFRNN	**97.2%**	**0.12**	128	50826
ARNN	95.2%	0.16	128	17930

Table 1: Results of the different architectures on the *pixel-by-pixel MNIST* problem. AFRNN achieves the best results.

tuning on a small subset of the data. The sizes of the (hidden) recurrent layers are always set to 128. As a loss function, binary cross-entropy between the output of the softmax and the true one-hot vector for the image is used. As an optimizer, we use adagrad (Duchi et al. (2011)) with a learning rate of 10^{-2}. The batch size is set to 128. Each model is trained for 100 epochs. The results are summarized in Table 1.

Among the four compared models, the AFRNN performs the best. Most importantly, it performs better than the same model without the constraints on the feedback matrices. This shows that the stability of the AFRNN is beneficial for learning long-term dependencies on real data. We also note that the AFRNN performs better than the 2-ARNN. This is presumably since the feedback allows for a more expressive model. The FRNN however does have more parameters since the feedback matrix is not conditioned on the feedforward matrix. Finally, the AFRNN also performs better than the ARNN in our implementation. We however note that the results for the ARNN obtained in Chang et al. (2019) are better than the results reported here. We decided to compare our results, because we did not perform any grid search of hyperparameters. Finally, we tried training an LSTM on the problem. We used the standard implementation from PyTorch (Paszke et al. (2019)). However, the model did not converge on the dataset, so we excluded it from the results. We note that there are good results for training LSTM on this dataset (e.g. Arjovsky et al. (2016)).

5.2. Pixel-by-Pixel CIFAR10

The CIFAR10 dataset (Krizhevsky et al. (2009)) consists of $60,000$ RGB images, each having a size of 32×32 pixels. Similarly to MNIST, we flatten the image before feeding it to the model. However, instead of a single greyscale pixel, we feed three values corresponding to the RGB values of the current pixel into the model at each timestep. The total number of timesteps per image is thus 1024. The images represent one of 10 possible classes, like *airplane, bird* and *cat*. We use the same model architectures as in Section 5.1, except for the different input dimensionality. We used the same loss, optimizer, batch size, number of epochs, and learning rate as for MNIST. The obtained results are given in Table 2.

We observe that the AFRNN performs better than the 2-ARNN, similar to what we observed in the MNIST experiment. We however observe that the FRNN slightly outperforms the AFRNN. There are two possible reasons for this. First, the FRNN has more parameters as the feedback matrix is not conditioned on the feedback matrix. Thus, the model has greater expressivity. Secondly, we assume that the pixel values over time are more randomly

Model	Accuracy	Loss	Hidden Units	Parameters
2-ARNN	48.4%	1.46	128	51082
FRNN	**50.3%**	**1.40**	128	67466
AFRNN	49.3%	1.43	128	51082
ARNN	48.6%	1.45	128	18186
LSTM	28.4%	1.98	128	69386

Table 2: Results of the different architectures on the *pixel-by-pixel cifar10* problem.

distributed in CIFAR10 than in MNIST, as the whole image displays a natural scene. In MNIST, on the other hand, large parts of the image are black, and thus the model receives no input. We hypothesize that the first implicitly regularizes the FRNN network and thus relaxes the need for a stable architecture. We also note that the ARNN performs worse than the AFRNN in our implementation, similarly to the LSTM does (although Chang et al. (2019) reports better results for both models). We also note that better results for this problem are reported in the literature (e.g. Gu et al. (2021)).

5.3. Double Pendulum Trajectories

Lastly, we compare the different models on the task of predicting the trajectories of a physical dynamical system, namely the (planar) double pendulum. A double pendulum consists of two coupled pendulums. Although conceptually simple, it can exhibit rich dynamics, including chaotic behavior. A chaotic system is loosely defined as one for which two infinitesimally close trajectories will diverge exponentially with time. Chaotic system trajectories appear nondeterministic, although the underlying differential equation is deterministic. The state of the system can be fully described by the angles θ_1, θ_2, and angular velocities $\dot{\theta}_1, \dot{\theta}_2$ between each pendulum and the vertical axis. A sketch of a double pendulum is shown in Figure 5.

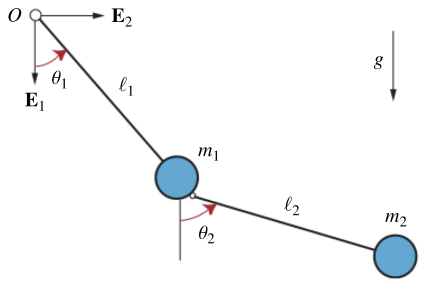

Figure 5: Sketch of a double pendulum (Source: Win, CC BY NC)

The equations describing the evolution of the angles and angle-velocities are given by

$$\theta_1'' = \frac{-g(2m_1+m_2)\sin(\theta_1) - m_2 g \sin(\theta_1-2\theta_2) - 2\sin(\theta_1-\theta_2)m_2(\theta_2'^2 L_2 + \theta_1'^2 L_1 \cos(\theta_1-\theta_2))}{L_1(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))}, \tag{19}$$

$$\theta_2'' = \frac{2\sin(\theta_1-\theta_2)(\theta_1'^2 L_1(m_1+m_2) + g(m_1+m_2)\cos(\theta_1) + \theta_2'^2 L_2 m_2 \cos(\theta_1-\theta_2))}{L_2(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))}. \tag{20}$$

Model	Validation Loss	Hidden Units	Parameters
2-ARNN	33.5	100	31004
FRNN	25.8	100	41004
AFRNN	**19.4**	100	31004
ARNN	19.8	200	44000
LSTM	50.3	100	123604

Table 3: Results of the different architectures on the double pendulum prediction task.

Here, L_1 and L_2 denote the lengths of the two pendulums, while m_1 and m_2 denote the masses.

We now investigate the ability of the different architectures to predict the trajectories of a double pendulum. We believe that the AFRNN is especially suitable for systems with highly nonlinear dynamics. For this, consider the task of predicting the next value of an ODE discretized in time, given an initial value. Further, assume that the initial value is the only input given, i.e., the prediction is not performed in an autoregressive way, where the output representing the next timestep is fed back into the network as input. The only information that is carried from timestep to timestep in an RNN is thus the hidden state. Consider a two-layer RNN. If there is no feedback connection, the lower layers only receive their past state, which is a worse embedding/approximation to the desired output than the upper layer. Suppose a feedback connection from the upper to the lower layer is introduced. In that case a better embedding is provided, and the network can perform a better nonlinear computation in the next timestep as it does not need to refine the embedding with the first layer first. The concrete task for the models is to predict the values of $\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2$ for 30 timesteps, given the values of those variables for the initial timestep only. It is thus different from an autoregressive prediction, where the model receives the previous state in each timestep. Instead, following the first timestep, the model will only receive a 1 as a placeholder for each timestep.

For the data, we fixed $m_1 = m_2 = 1\text{kg}$ and $L_1 = L_2 = 1\text{m}$. Using those parameters, we generate 1000 trajectories with different initial conditions. The initial conditions are the values for both pendulums' angles and the angular velocity. For those four values, random values are generated between -90 and 90. All trajectories represent three seconds of evolution with time steps of 0.1s. We train the different network architectures on 900 of those trajectories, while 100 trajectories are used as a test set.

The results are presented in Table 3. We report the mean squared error over the whole trajectory and all four variables. We observe that the AFRNN outperforms the other architectures once more. We attribute this to the fact that large parts of the input are constant and thus do not regularize the FRNN. Furthermore, we assume our above explanation to hold for this case.

6. Conclusion and Future Work

We have presented a novel neural network architecture, the Antisymmetric Feedback RNN (AFRNN). It is close in spirit to a vanilla RNN in the sense that it works with plain neurons, in contrast to LSTM or GRU cells. It is expressive in the sense that it can contain any

number of layers. It is provably stable as a dynamical system by forcing anti-symmetry properties in its various connection matrices, and hence does not suffer from vanishing or exploding gradients when processing data with a large time horizon. Finally, in its proposed standard form, it is scalable to great depth, with the number of weights growing only linear with depth, which is no different from a standard feedforward network. Finally, the new architecture performs well on standard benchmark problems. We therefore conclude that AFRNNs are a valuable addition to existing RNN architectures.

Further research directions include applying the antisymmetric top-down feedback to popular architectures like LSTM, enforcing antisymmetry in other top-down architectures and scaling the proposed architecture to larger systems for problems like speech recognition. More concretely, we want to evaluate the model on the ETTh1 time series prediction task as well as on the WMT2014 machine translation dataset. We also want to apply the antisymmetric feedback principle to transfromer architectures.

References

The double pendulum. https://rotations.berkeley.edu/the-double-pendulum/. Accessed: 2022-06-20.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR, 2016.

Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International conference on machine learning*, pages 2067–2075. PMLR, 2015.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

Sorin Mihai Grigorescu, Bogdan Trasnea, Tiberiu T. Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *CoRR*, abs/1910.07738, 2019. URL http://arxiv.org/abs/1910.07738.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *CoRR*, abs/2111.00396, 2021. URL https://arxiv.org/abs/2111.00396.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B Issa, and James J DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature neuroscience*, 22(6):974–983, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alexandr Pak, Esther Ryu, Claudia Li, and Alexander A Chubykin. Top-down feedback controls the cortical representation of illusory contours in mouse primary visual cortex. *Journal of Neuroscience*, 40(3):648–660, 2020.

Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012. URL http://arxiv.org/abs/1211.5063.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Lawrence F Shampine and Skip Thompson. Initial value problems. *Scholarpedia*, 2(3):2861, 2007.

Tom Sikkens, Conrado A Bosman, and Umberto Olcese. The role of top-down modulation in shaping sensory processing across brain states: implications for consciousness. *Frontiers in systems neuroscience*, page 31, 2019.

Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1308–1317, 2017.

Appendix A. Example of Diagonal Matrix D of J

In the following, we provide an example how the diagonal matrix $\mathbf{D}$ is constructed. For the sake of space, we focus on the values of $\boldsymbol{W}$ to show how the indexes f and $\chi(l)$ are computed, and omit the rest of the elements from Equation 14.

The example shows $\mathbf{D}$ for a AFRNN with 2 layers, where the first layer is of dimension 2 and the second layer is of dimension 4. As we can see, f stays constant as long as l is within the range of the current layer, and increases by 1 when the submatrix for the next layer is reached. We also see that the values for $\chi(l)$ increment within the section of each layer, starting at 1. As soon as the next layer is reached, it is reset to 1. This illustrates the use of the constraint in Equation 14 as well as the formula for $\chi(l)$ to properly index this matrix.

$$
\mathbf{D} = \begin{array}{c} \\ \text{Layer 1} \\ \text{Layer 2} \end{array}
\begin{array}{c}
\overbrace{\phantom{\boldsymbol{W}_1^{\{k1\}}\boldsymbol{W}_2^{\{k1\}}}}^{\text{Layer 1}}\ \overbrace{\phantom{\boldsymbol{W}_1^{\{k2\}}\boldsymbol{W}_2^{\{k2\}}\boldsymbol{W}_3^{\{k2\}}\boldsymbol{W}_4^{\{k2\}}}}^{\text{Layer 2}} \\
\left[\begin{array}{cccccc}
\boldsymbol{W}_1^{\{k1\}} & & & & & \\
 & \boldsymbol{W}_2^{\{k1\}} & & & & \\
 & & \boldsymbol{W}_1^{\{k2\}} & & & \\
 & & & \boldsymbol{W}_2^{\{k2\}} & & \\
 & & & & \boldsymbol{W}_3^{\{k2\}} & \\
 & & & & & \boldsymbol{W}_4^{\{k2\}}
\end{array}\right]
\end{array}
$$