Theoretical and practical survey on RNN gradient issues

Lucas Degeorge
ENSAE Paris
lucas.degeorge@ensae.fr

Charles Heitz
ENSAE Paris
charles.heitz@ensae.fr

Corentin Pla
ENSAE Paris
corentin.pla@ensae.fr

Abstract

Recurrent Neural Networks (RNNs) were specifically crafted for handling sequential data processing, such as audio feeds or time series analysis. Nevertheless, a common challenge encountered when employing RNNs is the occurrence of exploding or vanishing gradients. This article delves into the theoretical underpinnings behind these issues and investigates practical solutions to address them. We explore contemporary architectures like LSTM and GRU, which have emerged as effective remedies for mitigating the problems associated with gradient instability.

1 Introduction

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed for sequential data processing. They are well-suited for tasks like natural language processing, speech recognition, and time-series analysis. They are designed to capture information from previous time steps and save it in their hidden states (a memory). However, the potential of RNNs has been marred by the issues of vanishing and exploding gradients during training.

The vanishing gradient problem arises when the gradients computed during backpropagation become extremely small as they are propagated through time, making it difficult for the network to learn meaningful representations for long sequences. On the other hand, the exploding gradient problem occurs when gradients become exceedingly large, causing instability in the training process. Both issues can lead to ineffective learning, preventing RNNs from capturing information over extended temporal spans.

Vanishing and exploding gradient issues come from the nature of recurrent connections in neural networks. As information traverses through the network over multiple time steps, the repeated application of weight matrices in the backpropagation process can either diminish the gradients to insignificance or inflate them to impractical magnitudes.

Over the years, researchers have proposed innovative solutions to mitigate this two issues. One of the breakthroughs came with the introduction of Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber [3]. LSTM networks have a gating mechanism to selectively update and forget information, thereby alleviating the vanishing gradient problem. Additionally, Gated Recurrent Units (GRUs) [6] emerged as an alternative with a simpler structure, offering comparable performance.

This work aims to provide a theoretical survey of the vanishing and exploding gradient issues in RNNs, and a practical demonstration of the problems. Moreover, it explores alternatives, such as advanced architectures and training strategies, that have been developed to overcome these challenges.

In the subsequent sections, we delve into the theoretical foundations of the vanishing/exploding gradient problem, examine its practical implications on training RNNs, and explore the evolution of alternative architectures and methodologies that have been devised to tackle these challenges.

2 Related work

Vanishing and Exploding Gradient Issues The challenges associated with training RNNs have long been recognized, as highlighted in the foundational work by Bengio et al. (1994) [1]. Pascanu et al. (2013) [5] contribute a comprehensive analysis of the vanishing and exploding gradient problems, utilizing analytical, geometric, and dynamical systems perspectives. Their work aims to improve our understanding of the underlying issues and justifies a simple yet effective solution.

Learning Long-Term Dependencies Bengio et al. (1994) [2] investigate the broader challenge of learning long-term dependencies with gradient descent in recurrent neural networks. They reveal the inherent trade-off between efficient learning by gradient descent and capturing information for extended periods. Hochreiter et al. (2003) [4] focus on the practical difficulty of training RNNs to learn long-term dependencies. They demonstrate that traditional gradient descent techniques face limitations, particularly when robustly latching bits of information with certain attractors. This work emphasizes the need for alternative optimization methods and architectures to overcome the challenges posed by long-term dependencies in RNNs.

Possible solutions and advanced Architectures Pascanu et al. (2013) [5] propose a gradient norm clipping strategy for addressing exploding gradients, and a soft constraint for the vanishing gradients problem. Hochreiter and Schmidhuber [3] introduced the LSTM architecture in 1997 with input, forget, and output gates, allowing the network to selectively retain, discard, and output information. Cho et al. [6] presented the GRU as an alternative architecture to LSTM. The GRU demonstrated comparable performance to LSTMs while being computationally more efficient. Chung et al. (2014) [7] provide an empirical evaluation of recurrent units, specifically focusing on those with gating mechanisms, like LSTM units and GRUs. Their study reveals the superiority of advanced recurrent units over traditional ones, and the effectiveness of gating mechanisms in addressing sequence modeling challenges.

3 Theoretical aspects

3.1 RNN and backpropagation

A generic RNN is defined in the following way. Given an input sequence $\mathbf{x} = (x_1, \dots, x_T) \in \left(\mathbb{R}^d\right)^T$, the hidden states $h_t \in \mathbb{R}^d$ for $t = 0, \dots, T$ are defined by:

$$h_t = \begin{cases} 0 & \text{if } t = 0 \\ g(\mathbf{W}_{rec} h_{t-1} + \mathbf{W}_{in} x_t + b) & \text{if } t > 0 \end{cases} \tag{1}$$

In Equation 1, $\mathbf{W}_{rec}$ and $\mathbf{W}_{in}$, the recurrent and input weight matrices, belong to $\mathcal{M}_d(\mathbb{R})$ and $b \in \mathbb{R}^d$ is the bias. For the general case, we denote θ the parameters (the two matrices and the bias) of the model. d is the dimension of the hidden space. g is an activation function, usually the hyperbolic tangent (denoted $\tanh$) and the ReLU function.

Moreover, we denote $\mathcal{E}$ the error that measures the performances of the network. One can break this error into individual costs for each step:

$$\mathcal{E} = \sum_{t=0}^{T} \mathcal{E}_t$$

Using the chain rule, one can determine the equations giving the gradients:

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad \text{where} \quad \frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{k=1}^{t} \left(\frac{\partial \mathcal{E}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \theta} \right)$$

In the previous equation, the Jacobian $\frac{\partial h_t}{\partial h_k}$ transports the error in time from step t back to step k. It can be rewritten the following way:

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} \mathbf{W}_{rec}^T \operatorname{diag}\left(g'\left(h_{i-1}\right)\right) \tag{2}$$

This product of $t-k$ Jacobian matrices is responsible for the vanishing or exploding gradient issues. Taking the Frobenius norm in the Equation 2, we get:

$$\left\|\frac{\partial h_t}{\partial h_k}\right\| \leq \prod_{i=k+1}^{t} \left\|\frac{\partial h_i}{\partial h_{i-1}}\right\| = \prod_{i=k+1}^{t} \left\|\mathbf{W}_{rec}^T \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\| \tag{3}$$

We denote $\underline{\lambda}$ and $\bar{\lambda}$ the smallest and largest eigenvalues of W_{rec} and γ the upper bound of g' the derivative of activation function g.

Pascanu et al. [5] claim (Equation (6) of their paper) that if $\bar{\lambda}$ of W_{rec} (and hence W_{rec}^T) is smaller than $\frac{1}{\gamma}$,

$$\forall i, \left\|\frac{\partial h_i}{\partial h_{i-1}}\right\| \leq \left\|\mathbf{W}_{rec}^T\right\| \left\|\operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\| < \frac{1}{\gamma}\gamma < 1$$

such that, using 3, $\left\|\frac{\partial h_t}{\partial h_k}\right\|$ tends to zero for large t. However, this proof is fallacious as the authors assumed $\left\|W_{\text{rec}}^T\right\| \leq \bar{\lambda}$ when in fact, $\left\|W_{\text{rec}}^T\right\| \geq \lambda_i$ for any eigenvalue λ_i of $\mathbf{W}_{rec}^T$.

Nevertheless, their conclusion on the link between the eigenvalues of $\mathbf{W}_{rec}$ and the vanishing or exploding gradient issues turns out to be correct, thanks to a similar result we will show.

Theorem. Assuming that g' is bounded by α (lower bound) and γ (upper bound). We have:

$$(\underline{\lambda}\alpha)^{t-k}\sqrt{d} \leq \left\|\frac{\partial h_t}{\partial h_k}\right\| \leq (\bar{\lambda}\gamma)^{t-k}\sqrt{d} \tag{4}$$

Proof. We consider $\mathbf{W}_{rec}^T = U\Sigma V^*$ the SVD decomposition of $\mathbf{W}_{rec}^T$ (U and V are unitary matrices and Σ is a diagonal matrix with diagonal coefficient being either the eigenvalues of $\mathbf{W}_{rec}^T$ or zero.

Taking the Frobenius norm in 2, and as U and V are unitary matrices, we have

$$\begin{aligned}
\left\|\frac{\partial h_t}{\partial h_k}\right\| &= \left\|\prod_{i=k+1}^{t} U\Sigma V^* \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\| \\
&= \left\|U\Sigma V^* \operatorname{diag}\left(g'\left(h_k\right)\right) \prod_{i=k+2}^{t} U\Sigma V^* \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\| \\
&= \left\|\Sigma V^* \operatorname{diag}\left(g'\left(h_k\right)\right) \prod_{i=k+2}^{t} U\Sigma V^* \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\|
\end{aligned}$$

As $\|DA\| \leq \bar{D}\|A\|$ for any diagonal matrix D (whose largest eigenvalue is $\bar{D}$) and any matrix A, we have

$$\begin{aligned}
\left\|\frac{\partial h_t}{\partial h_k}\right\| &\leq \bar{\lambda}\left\|V^* \operatorname{diag}\left(g'\left(h_k\right)\right) \prod_{i=k+2}^{t} U\Sigma V^* \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\| \\
&\leq \bar{\lambda}\left\|\operatorname{diag}\left(g'\left(h_k\right)\right) \prod_{i=k+2}^{t} U\Sigma V^* \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\| \\
&\leq \bar{\lambda}\gamma\left\|\prod_{i=k+2}^{t} U\Sigma V^* \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)\right\|
\end{aligned}$$

By iterating this process,

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| \leq \bar{\lambda}^{t-k} \gamma^{t-k} \left\| I_d \right\| = \bar{\lambda}^{t-k} \gamma^{t-k} \sqrt{d}$$

Following a similar argument and using $\underline{D}\|A\| \leq \|DA\|$, we get the corresponding lower bound:

$$|\underline{\lambda}|^{t-k} \cdot \alpha^{t-k} \sqrt{d} \leq \left\| \frac{\partial h_t}{\partial h_k} \right\|$$

$\square$

This theorem allows us to state the following links between the eigenvalues of $\mathbf{W}_{rec}$ and the vanishing or exploding gradient issues:

- if $\bar{\lambda}\gamma < 1$, $\left\| \frac{\partial h_t}{\partial h_k} \right\|$ tends to 0 as t increases, so the **gradient explodes**
- if $\bar{\lambda}\alpha > 1$, $\left\| \frac{\partial h_t}{\partial h_k} \right\|$ tends to $+\infty$ as t increases, so the **gradient vanishes**

3.2 LSTM cells

LSTM cells are gated cells defined in the following way. Given an input sequence $\mathbf{x} = (x_1, \ldots, x_T) \in \left(\mathbb{R}^d\right)^T$, the following states and gates, belonging to $\mathbb{R}^d$, are defined by:

$$\begin{aligned}
i_t &= \sigma\left(\mathbf{W}_{in,i} x_t + \mathbf{W}_{rec,i} h_{t-1} + b_i\right) \\
f_t &= \sigma\left(\mathbf{W}_{in,f} x_t + \mathbf{W}_{rec,f} h_{t-1} + b_f\right) \\
g_t &= \tanh\left(\mathbf{W}_{in,g} x_t + \mathbf{W}_{rec,g} h_{t-1} + b_g\right) \\
o_t &= \sigma\left(\mathbf{W}_{in,o} x_t + \mathbf{W}_{rec,o} h_{t-1} + b_o\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh\left(c_t\right)
\end{aligned}$$

where, for each $t \in [0, T]$, h_t is the hidden state, c_t is the cell state, x_t is the input. i_t f_t, g_t, g_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function, and $\odot$ is the Hadamard product.h_0 and c_0 are two zero vectors

The different matrices $\mathbf{W}_{rec,\_}$ and $\mathbf{W}_{in,\_}$ play the same role as $\mathbf{W}_{rec}$ and $\mathbf{W}_{in}$ from equation 1. Same for the bias $b\_$.

As for the RNN, one can determine the gradient of an LSTM network.

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{t=1}^{T} \left(\sum_{k=1}^{t} \frac{\partial \mathcal{E}_t}{\partial h_t} \frac{\partial h_t}{\partial c_t} \frac{\partial c_t}{\partial c_k} \frac{\partial c_k}{\partial \theta} \right) \quad \text{where} \quad \frac{\partial c_t}{\partial c_k} = \prod_{j=k+1}^{t} \frac{\partial c_j}{\partial c_{j-1}}$$

In the case of LSTM, this is $\frac{\partial c_t}{\partial c_k}$ that transports the error in time. It is also a product of Jacobian matrices. One can then compute one of these Jacobian matrices $\frac{\partial c_j}{\partial c_{j-1}}$. Detailed computations are given in Appendix A.1). Biases have been omitted.

$$\begin{aligned}
\frac{\partial c_j}{\partial c_{j-1}} = \; & \operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,f} x_j + \mathbf{W}_{rec,f} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,f}^T \cdot \operatorname{diag}\left(o_{j-1} \odot \tanh'\left(c_{j-1}\right)\right) \cdot \operatorname{diag}(c_{j-1}) \\
& + \operatorname{diag}(f_j) \\
& + \operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,i} x_j + \mathbf{W}_{rec,i} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,i}^T \cdot \operatorname{diag}\left(o_{j-1} \odot \tanh'\left(c_{j-1}\right)\right) \cdot \operatorname{diag}(g_j) \\
& + \operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,c} x_j + \mathbf{W}_{rec,c} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,c}^T \cdot \operatorname{diag}\left(o_{j-1} \odot \tanh'\left(c_{j-1}\right)\right) \cdot \operatorname{diag}(i_j)
\end{aligned} \tag{5}$$

The first, third and fourth terms in Equation 5 seem to have the same structure as the term $\mathbf{W}_{rec}^T \operatorname{diag}\left(g'\left(h_{i-1}\right)\right)$ from the RNN Jacobian matrix $\frac{\partial h_i}{\partial h_{i-1}}$. They will therefore undoubtedly suffer from the same vanishing or exploding problem when the eigenvalues of $\mathbf{W}_{rec,f}$, $\mathbf{W}_{rec,i}$ and $\mathbf{W}_{rec,c}$ satisfy some conditions.

However, the presence of the forget gate's activation vector in the gradient (the second term in Equation 5) can prevent the vanishing gradient problem. In cases where the cumulative partial derivatives trend toward zero up to time step $t_a < T$, *ie*: $\sum_{t=1}^{t_a} \frac{\partial E_t}{\partial \theta} \to 0$, the network can find a suitable parameter update of the forget gate at time step $t_a + 1$ such that $\frac{\partial E_{t_a+1}}{\partial \theta} \nrightarrow 0$.

Moreover, the presence of the output gate's activation vector in the first, third and fourth terms of Equation 5 could prevent the exploding gradient issue by reducing at 0 the exploding term along some direction v.

3.3 GRUs

The case of GRUs is quite similar to the LSTM one. The states and gates are however not updated in the same way. Given an input sequence $\mathbf{x} = (x_1, \dots, x_T) \in \left(\mathbb{R}^d\right)^T$, we define [1]:

$$\begin{aligned}
r_t &= \sigma\left(\mathbf{W}_{in,r} x_t + \mathbf{W}_{rec,r} h_{t-1} + b_r\right) \\
z_t &= \sigma\left(\mathbf{W}_{in,z} x_t + \mathbf{W}_{rec,z} h_{t-1} + b_z\right) \\
n_t &= \tanh\left(\mathbf{W}_{in,n} x_t + b_{in,n} + r_t \odot \left(\mathbf{W}_{rec,n} h_{(t-1)} + b_{rec,n}\right)\right) \\
h_t &= (1 - z_t) \odot n_t + z_t \odot h_{(t-1)}
\end{aligned}$$

where h_t is the hidden state, x_t is the input. r_t, z_t and n_t are the reset, update, and new gates, respectively.

Again, one can determine the gradients:

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{t=1}^{T} \left(\sum_{k=1}^{t} \frac{\partial \mathcal{E}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \theta} \right) = \sum_{t=1}^{T} \left(\sum_{k=1}^{t} \frac{\partial \mathcal{E}_t}{\partial h_t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial \theta} \right)$$

where the Jacobian matrix $\frac{\partial h_i}{\partial h_{i-1}}$ is (detailed computation are given in Appendix A.2)

$$\begin{aligned}
\frac{\partial h_i}{\partial h_{i-1}} &= -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,z} x_i + \mathbf{W}_{rec,z} h_{i-1}\right)\right) \cdot \mathbf{W}_{rec,z}^T \cdot \operatorname{diag}(n_i) \\
&\quad + \operatorname{diag}(1 - z_i) \cdot \operatorname{diag}\left(\tanh'\left(\mathbf{W}_{in,n} x_i + \mathbf{W}_{rec,n} h_{i-1}\right)\right) \cdot \mathbf{W}_{rec,n}^T \cdot \operatorname{diag}(r_i) \\
&\quad + \operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,z} x_i + \mathbf{W}_{rec,z} h_{i-1}\right)\right) \cdot \mathbf{W}_{rec,z}^T \cdot \operatorname{diag}(h_{i-1}) \\
&\quad + \operatorname{diag}(z_i)
\end{aligned} \tag{6}$$

As for the LSTM, the fourth term, the update gate's activation vector, allows the network to prevent the vanishing gradient problem. And the presence of n_i, $1 - z_i$ and r_i in the first, second and third term could prevent the exploding gradient issue.

4 Experiments

To test the practical validity of this result, we have implemented two different experiences. The first one is a pathological synthetic problem: the temporal order problem. The second one is a natural problem: music note prediction. The code and all the data used for the two experiments are available here. See the `README.md` file on the repository for detail about the code.

4.1 Temporal order problem

The input is a sequence of length T of distractor symbols $\{c, d, e, f\}$. At two random points in time (in the beginning and middle of the sequence) a symbol within $\{A, B\}$ is inserted. The task is to determine the order of the two symbols (either AA, AB, BA, BB).

[1]The calculation of new gate n_t subtly differs from the original paper. In the original implementation, the Hadamard product between r_t and the previous hidden state h_{t-1} is done before the multiplication with the weight matrix $\mathbf{W}_{rec,n}$ and addition of bias $b_{rec,n}$. This is done as in PyTorch, on purpose for efficiency

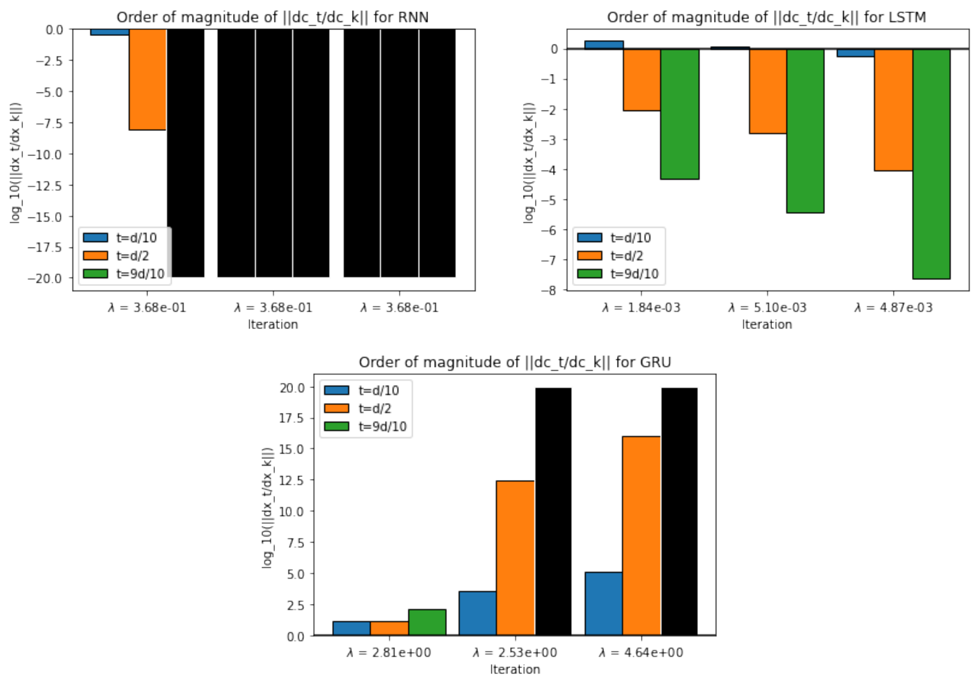

Table 1: $\log_{10}\left(\left\|\frac{\partial h_t}{\partial h_k}\right\|\right)$ for RNN (top left), LSTM (top right) and GRU (bottom) during temporal order problem task. A black bar means a value of $\left\|\frac{\partial c_t}{\partial c_k}\right\|$ equals to zero (negative) or $+\infty$ (positive). k is equal to 1 and t equals $\frac{T}{10}$, $\frac{T}{2}$ and $\frac{9T}{10}$ where T is the length of the input sequence. The values of $\bar{\lambda}$ of reported below the x-axis.

The sequence entries are uniformly sampled from the distractor symbols everywhere except at two random positions. The first position is uniformly sampled from $[\frac{T}{10}, \frac{2T}{10}]$. The second is uniformly sampled from $[\frac{4T}{10}, \frac{5T}{10}]$.

For this task, we used a model with a hidden space of dimension $d = 50$, with a $\tanh$ as an activation function (as Pascanu et al. [5] did). The optimizer was the stochastic gradient descent (SGD) and the learning rate 5.10^{-1}.

Table 1 shows the magnitude (*ie:* $\log_{10}$) of $\left\|\frac{\partial h_t}{\partial h_k}\right\|$ (or $\left\|\frac{\partial c_t}{\partial c_k}\right\|$ for LSTM) during the training of an RNN, a LSTM network and a GRUs network for the . There are three groups of bars per figure, corresponding to three timestep during the training. The first one is at 10% (in term of iterations), the second one at 50% and the third one at 90%. For each time step, each bar corresponds to one value of $t \in [\frac{T}{10}, \frac{T}{2}, \frac{9T}{10}]$.

We, first, observe that, for the RNN and the LSTM network, $\bar{\lambda} < 1$ and the gradient vanishes. For the GRUs network, $\bar{\lambda} > 1$ and the gradient explodes. The result of Section 3 is therefore found experimentally. It appears that the LSTM was efficient at preventing the vanishing gradient issue as $\log_{10}\left(\left\|\frac{\partial c_t}{\partial c_k}\right\|\right)$ does not tend to zero during the training. However, the GRUs network was not able to prevent the exploding gradient issue during the training.

For both three cases, the models were not able to succeed in this task, even for the LSTM who managed to reduce vanishing gradient. The results were very poor.

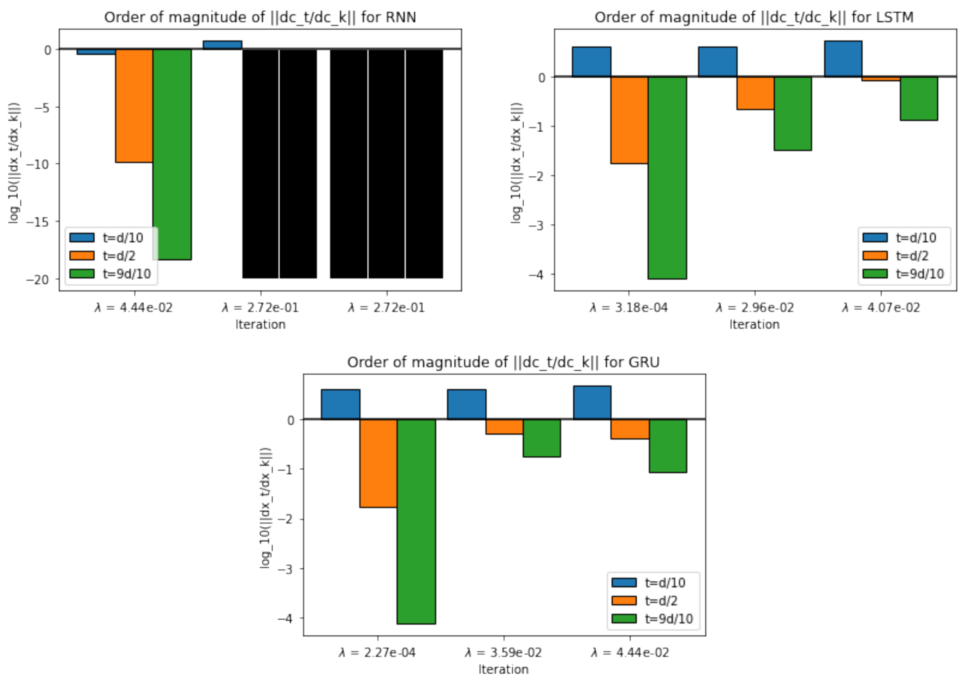

Table 2: $\log_{10}\left(\left\|\frac{\partial h_t}{\partial h_k}\right\|\right)$ for RNN (top left), LSTM (top right) and GRU (bottom) during music note prediction task. See caption of Table 1 for more detail

4.2 Music note prediction

Given a sequence of piano notes, the objective is to predict the future notes. We use the dataset ADL Piano MIDI [8][2]. It is a dataset of 11,086 piano pieces from different genres. For each of the pieces, we only consider the notes (the pitches in the MIDI format) of the first instrument. We also consider that only one note can be played at the same time and that each note lasts the same amount of time.

We used a model with a hidden space of dimension $d = 256$, with a $\tanh$ as an activation function. The optimizer was the stochastic gradient descent (SGD) and the learning rate 10^{-1}.

Table 2 shows the order of magnitude for the music note prediction task. Again, the theoretical result shown in Section 3 can be observed. Contrary to the temporal order problem task, both the LSTM network and the GRUs network were able to prevent the vanishing gradient issue.

Table 3 shows the generations of music notes given a first random note of the RNN, the LSTM network and GRUs network. The generated notes are in red and the probability distribution at each timestep is plotted in shadow of greys. Thanks to the prevention of gradient issues, the LSTM and the GRUs network seems to perform better.

5 Conclusion

This work underscores the importance of the spectral radius of the recurrent weight matrix and of the bounds of the derivative of the activation function for the efficiency of a RNN's training. However, thanks to their gated architecture, LSTM and GRUs network could prevent such issues.

Experimental results indicate that LSTMs effectively prevent vanishing gradient problems, maintaining stable gradient flow during training. GRUs face challenges in preventing exploding gradients on toy data but succeed to prevent vanishing gradient with real data. Despite this, implementing measures to address gradient issues, especially in LSTMs, leads to improved overall network performance.

[2]The dataset is available here

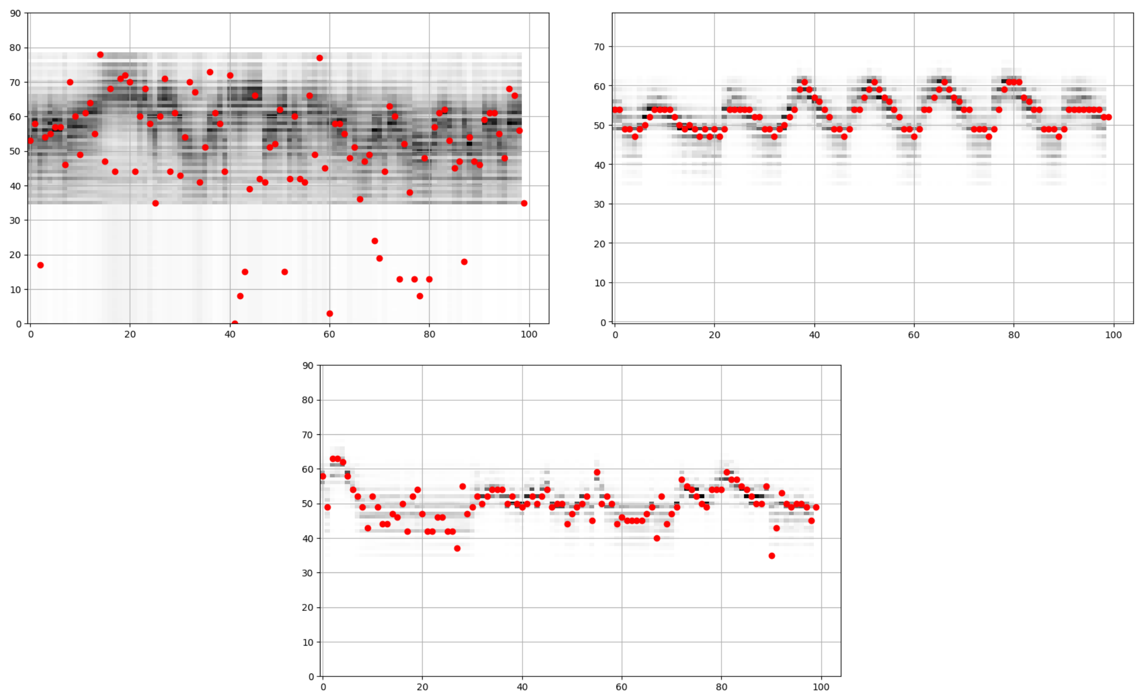

Table 3: Generation from RNN (top left), LSTM (top right) and GRUs (bottom)

References

[1] Y. Bengio, P. Frasconi, and P. Simard, "Problem of learning long-term dependencies in recurrent networks," Feb. 1993, 1183–1188 vol.3. DOI: `10.1109/ICNN.1993.298725`.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 5, pp. 157–66, Feb. 1994. DOI: `10.1109/72.279181`.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: `10.1162/neco.1997.9.8.1735`.

[4] F. Informatik, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," *A Field Guide to Dynamical Recurrent Neural Networks*, Mar. 2003.

[5] R. Pascanu, T. Mikolov, and Y. Bengio, *On the difficulty of training recurrent neural networks*, 2013. arXiv: `1211.5063 [cs.LG]`.

[6] K. Cho, B. van Merrienboer, C. Gulcehre, *et al.*, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014. arXiv: `1406.1078 [cs.CL]`.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, 2014. arXiv: `1412.3555 [cs.NE]`.

[8] L. N. Ferreira, L. H. Lelis, and J. Whitehead, "Computer-generated music for tabletop role-playing games," AIIDE'20, 2020.

A Appendix: Computation of Jacobian matrices

A.1 LSTM

For any j, c_j is defined by

$$\begin{aligned} c_j &= f_j \odot c_{j-1} + i_j \odot g_j \\ &= \sigma\left(\mathbf{W}_{in,f} x_j + \mathbf{W}_{rec,f} h_{j-1} + b_f\right) \odot c_{j-1} \\ &+ \sigma\left(\mathbf{W}_{in,i} x_j + \mathbf{W}_{rec,i} h_{j-1} + b_i\right) \odot \tanh\left(\mathbf{W}_{in,g} x_j + \mathbf{W}_{rec,g} h_{j-1} + b_g\right) \end{aligned}$$

Then,

$$\frac{\partial c_j}{\partial c_{j-1}} = \underbrace{\frac{\partial f_j}{\partial c_{j-1}} \cdot \operatorname{diag}\left(c_{j-1}\right)}_{A_j} + \underbrace{\operatorname{diag}\left(f_j\right) \cdot I_d}_{B_j} + \underbrace{\frac{\partial i_j}{\partial c_{j-1}} \cdot \operatorname{diag}\left(g_j\right)}_{C_j} + \underbrace{\operatorname{diag}\left(i_j\right) \cdot \frac{\partial g_j}{\partial c_{j-1}}}_{D_j}$$

with

$$\begin{aligned} A_j &= -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,f} x_j + \mathbf{W}_{rec,f} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,f}^T \cdot \frac{\partial h_{j-1}}{\partial c_{j-1}} \cdot \operatorname{diag}(c_{j-1}) \\ &= -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,z} x_j + \mathbf{W}_{rec,z} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,z}^T \cdot \operatorname{diag}\left(o_{j-1} \odot \tanh'\left(c_{j-1}\right)\right) \cdot \operatorname{diag}(c_{j-1}) \end{aligned}$$

$$B_j = \operatorname{diag}(f_j)$$

$$\begin{aligned} C_j &= -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,i} x_j + \mathbf{W}_{rec,i} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,i}^T \cdot \frac{\partial h_{j-1}}{\partial c_{j-1}} \cdot \operatorname{diag}(g_j) \\ &= -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,i} x_j + \mathbf{W}_{rec,i} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,i}^T \cdot \operatorname{diag}\left(o_{j-1} \odot \tanh'\left(c_{j-1}\right)\right) \cdot \operatorname{diag}(g_j) \end{aligned}$$

$$\begin{aligned} D_j &= -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,c} x_j + \mathbf{W}_{rec,c} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,c}^T \cdot \frac{\partial h_{j-1}}{\partial c_{j-1}} \cdot \operatorname{diag}(i_j) \\ &= -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,c} x_j + \mathbf{W}_{rec,c} h_{j-1}\right)\right) \cdot \mathbf{W}_{rec,c}^T \cdot \operatorname{diag}\left(o_{j-1} \odot \tanh'\left(c_{j-1}\right)\right) \cdot \operatorname{diag}(i_j) \end{aligned}$$

A.2 GRU

For any i, h_i is defined by $h_i = (1 - z_i) \odot n_i + z_i \odot h_{i-1}$. Then,

$$\frac{\partial h_i}{\partial h_{i-1}} = \underbrace{-\frac{\partial z_i}{\partial h_{i-1}} \cdot \operatorname{diag}\left(n_i\right)}_{A_i} + \underbrace{\operatorname{diag}\left(1 - z_i\right) \cdot \frac{\partial n_i}{\partial h_{i-1}}}_{B_i} + \underbrace{\frac{\partial z_i}{\partial h_{i-1}} \cdot \operatorname{diag}\left(h_{i-1}\right)}_{C_i} + \underbrace{\operatorname{diag}\left(z_i\right) \cdot I_d}_{D_i}$$

with

$$A_i = -\operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,z} x_i + \mathbf{W}_{rec,z} h_{i-1}\right)\right) \cdot \mathbf{W}_{rec,z}^T \cdot \operatorname{diag}(n_i)$$

$$\begin{aligned} B_i &= \operatorname{diag}(1 - z_i) \cdot \operatorname{diag}\left(\tanh'\left(\mathbf{W}_{in,n} x_i + \mathbf{W}_{rec,n} h_{i-1}\right)\right) \cdot \frac{\partial\left(r_i \odot \left(\mathbf{W}_{rec,n} h_{(t-1)} + b_{rec,n}\right)\right)}{\partial h_{i-1}} \\ &= \operatorname{diag}(1 - z_i) \cdot \operatorname{diag}\left(\tanh'\left(\mathbf{W}_{in,n} x_i + \mathbf{W}_{rec,n} h_{i-1}\right)\right) \cdot \mathbf{W}_{rec,n}^T \cdot \operatorname{diag}(r_i) \end{aligned}$$

$$C_i = \operatorname{diag}\left(\sigma'\left(\mathbf{W}_{in,z} x_i + \mathbf{W}_{rec,z} h_{i-1}\right)\right) \cdot \mathbf{W}_{rec,z}^T \cdot \operatorname{diag}(h_{i-1})$$

$$D_i = \operatorname{diag}(z_i)$$