An Intrinsic Difference Between Vanilla RNNs and GRU Models

Tristan Stérin
Computer Science Department
École Normale Supérieure de Lyon
Email: tristan.sterin@ens-lyon.fr

Nicolas Farrugia
Electronics Department
IMT Atlantique
Email: nicolas.farrugia@imt-atlantique.fr

Vincent Gripon
Electronics Department
IMT Atlantique
Email: vincent.gripon@imt-atlantique.fr

***Abstract*—In order to perform well in practice, Recurrent Neural Networks (RNN) require computationally heavy architectures, such as Gated Recurrent Unit (GRU) or Long Short Term Memory (LSTM). Indeed, the original Vanilla model fails to encapsulate middle and long term sequential dependencies. The aim of this paper is to show that gradient training issues, which have motivated the introduction of LSTM and GRU models, are not sufficient to explain the failure of the simplest RNN. Using the example of Reber's grammar, we propose an experimental measure of both Vanilla and GRU models, which suggest an intrinsic difference in their dynamics. A better mathematical understanding of this difference could lead to more efficient models without compromising performance.**

***Index Terms*—*Recurrent Neural Networks*; *Gradient Backpropagation*; *Grammatical Inference*; *Dynamical Systems*.**

I. Introduction

Recurrent Neural Networks (RNN) [1] are a class of artificial neural networks that feature connections between hidden layers that are propagated through time in order to learn sequences. In recent years, such networks, and especially variants such as Gated Recurrent Units (GRU) [2] or Long Short-Term Memory (LSTM) networks [3] have demonstrated remarkable performance in a wide range of applications involving sequences, such as language modeling [4], speech recognition [5], image captioning [6], and automatic translation [7]. Such networks often include a large number of layers (i.e., deep neural networks), each containing many neurons, resulting in a large set of parameters to be learnt. The main reason explaining this challenge are vanishing or exploding gradients while training parameters [8], [9], such problems being likely to emerge when learning any large set of parameters.

The conceptual leap from Vanilla RNN architectures to LSTM or GRU was initially justified by the inability of Vanilla RNN to learn long term sequential dependencies [3]. Theoretically, it is not clear whether this limitation of Vanilla RNN is indeed due to training issues, or to an intrinsic limitation in their architecture. In this paper, we propose to revisit this question by confronting GRU and Vanilla RNN by suggesting the existence of intrinsic theoretical differences between two models. We use the Embedded Reber Grammar and introduce an experimental measure based on dynamical systems theory, in order to highlight a limitation of Vanilla RNN architectures without encountering vanishing or exploding gradient issues.

The remainder of the paper is organized as follows. In Section II, we present the two RNN models that we evaluate, Vanilla RNN and GRU. Section III presents the Embedded Reber Grammar. In Section IV, we perform a set of experiments using both models, and highlight limitations of the Vanilla RNN while no issues with gradients can be demonstrated. in Section V, we introduce a discrepancy measure and show how it can be used to point out the limitations of the Vanilla RNN model. Finally, we give conclusions and perspectives in Section VI.

II. Recurrent Neural Network Models

In this section, we introduce the two models of recurrent neural networks we discuss in this paper, namely Vanilla RNNs and GRU.

A. Vanilla RNNs

Vanilla is the first model of recurrent artificial neural networks that was introduced [1]. Relying on very simple dynamics, this neural network is described with the following set of equations, indexed by time-step t:

$$\boldsymbol{h}_t = \sigma(U_h \boldsymbol{x}_t + W_h \boldsymbol{h}_{t-1}) \quad (1)$$

$$\boldsymbol{y}_t = O\boldsymbol{h}_t \quad (2)$$

where:

- $\boldsymbol{x}_t \in \mathbb{R}^n$ is the input of the RNN,
- $\boldsymbol{h}_t \in \mathbb{R}^k$ is called hidden state of the RNN, and acts as a memory of the current state the dynamics is into. When starting a sequence, it is set to the all zero vector ($\boldsymbol{h}_{-1} = 0$).
- $\boldsymbol{y}_t \in \mathbb{R}^p$ is the output of the RNN,
- The logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is applied component-wise,
- U_h, W_h, O are the network's parameters.

Figure 1 depicts a *Vanilla* RNN with four input neurons, a hidden layer with five neurons and one output neuron. The

 ISBN: 978-1-61208-531-9

output of such a neural network depends on both the input $\boldsymbol{x}_t$ and the hidden state $\boldsymbol{h}_{t-1}$ that stores information about the past values observed in the sequence.

B. GRU

GRU neural networks are a LSTM variant based on the following equations:

$$\begin{aligned}\boldsymbol{z} &= \sigma(U^z \boldsymbol{x}_t + W^z \boldsymbol{h}_{t-1}) \\ \boldsymbol{r} &= \sigma(U^r \boldsymbol{x}_t + W^r \boldsymbol{h}_{t-1}) \\ \boldsymbol{h}' &= tanh(U^{h'} \boldsymbol{x}_t + W^{h'}(\boldsymbol{h}_{t-1} \circ \boldsymbol{r}) \\ \boldsymbol{h}_t &= (1 - \boldsymbol{z}) \circ \boldsymbol{h}' + \boldsymbol{z} \circ \boldsymbol{h}_{t-1} \\ \boldsymbol{y}_t &= O\boldsymbol{h}_t\end{aligned}$$

where:

- $\boldsymbol{x}_t \in \mathbb{R}^n$ is the input vector,
- $\boldsymbol{h}_t \in \mathbb{R}^k$ is the hidden state vector, $\boldsymbol{h}_{-1} = 0$,
- $\boldsymbol{y}_t \in \mathbb{R}^p$ is the output vector,
- $\boldsymbol{r} \in \mathbb{R}^k$ is the "reset" vector, which is multiplied component-wise with the hidden state vector $\boldsymbol{h}_t$ when updating it, hence its name,
- $\boldsymbol{z} \in \mathbb{R}^k$ is a combination vector, which acts as a barycenter vector that combines the previous hidden state and currently estimated one to produce the next one,
- The logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is applied component-wise,
- $U^{z,r,h'}, W^{z,r,h'}, O$ are the network's parameters.

Both models are known to be Turing complete [10] and could theoretically achieve the same tasks. In practice, it is readily seen that it is much harder to encapsulate long term dependencies with Vanilla RNNs than with its LSTM-based counterpart.

III. Performance of Models on Regular and Embedded Reber's Grammar

Here, we introduce the grammatical inference problem associated with Reber's grammar as in [3] (Figure 2). The corresponding regular expression is $BPT^*V(PSE|VE) + BTS^*X(SE|XT^*V(PSE|VE))$.

We examine the ability of neural network models to infer this automaton from a set of examples in the grammar. Each letter is encoded using a one-hot encoding scheme, thus the input dimension n equals the number of different letters ($n = 7$). A word of length m is thus encoded as a sequence of vectors $\boldsymbol{x}_0 \dots \boldsymbol{x}_{m-1}$. A valid word is a word for which the automaton ends in the rightmost state. For instance, $BTSSXXTTTVPSE$ is valid and $BTSSE$ is not. The output space is also of size 7 and gives a –non normalised– probability distribution on the following character given the past input sequence. The RNN output –after softmax– can be interpreted as follows:

$$Pr(\boldsymbol{x}_{t+1}|\boldsymbol{x}_0 \dots \boldsymbol{x}_t)$$

We first train a Vanilla RNN model on this task using a dataset comprising 250 Reber strings. Code can be found at https://github.com/brain-bzh/IARIA17_RNNs/blob/master/ReberGrammars/.

The results are depicted in Figure 3 under the form of heatmaps. Interestingly, this model was able to provide good predictions after training, as we retrieve the edges of the corresponding states in the automaton of Figure 2.

Figure 4 shows the heatmaps of the hidden vectors $\boldsymbol{h}_t$. Here, a Vanilla RNN successfully infers some of the automaton's states, (e.g., $q2$ and $q5$). Hidden states are very similar even when sequences used to reach them are different.

Hence, a Vanilla RNN model can be trained to infer Reber's grammar. This result can be explained by the fact that it is sufficient to recall the two previous letters to infer which state the automaton is in.

We now consider the example of the Embedded Reber's grammar. An automaton corresponding to this grammar is depicted in Figure 5. In short, the Embedded Reber's grammar consists in two copies of Reber's grammar, except that all strings with a T (resp. P) in second position must have a T (resp. P) at the end, thus exhibiting a long term sequential dependency. We train a Vanilla RNN ($k = 18$) and a GRU model ($k = 10$) with about the same number of parameters ($\simeq 590$), using the same dataset comprising 2000 example sequences.

Figure 6 (resp. Figure 7) depicts the output given by a Vanilla RNN (resp. a GRU model) on the prediction task. This figures emphasizes the fact that Vanilla RNNs are unable to capture long term dependencies, whereas GRU successfully do so.

The confusion phenomenon observed in Figure 6 is already true for the hidden layer, as depicted in Figure 8. In the next sections, we investigate hypothesis regarding the origin of such differences of performance between the two models.

IV. Gradient's Coefficients Distribution

A commonly used argument to justify the performance difference between Vanilla RNNs and GRU models, as presented in [8], is related to the gradient instability when learning parameters of Vanilla RNNs, leading to gradient's coefficients that either explode or vanish. We tested this hypothesis by studying the statistical distributions of gradients when training both neural network models. Figure 9 represents the obtained distribution of absolute value of gradient's coefficients for both the training of Vanilla RNNs and the GRU model, over all the examples in the training set (embedded Reber's grammar). Both distributions are similar and discredit the hypothesis that instable gradient's coefficients are responsible for the unability of Vanilla RNNs to correctly infer the embedded Reber's grammar.

 ISBN: 978-1-61208-531-9

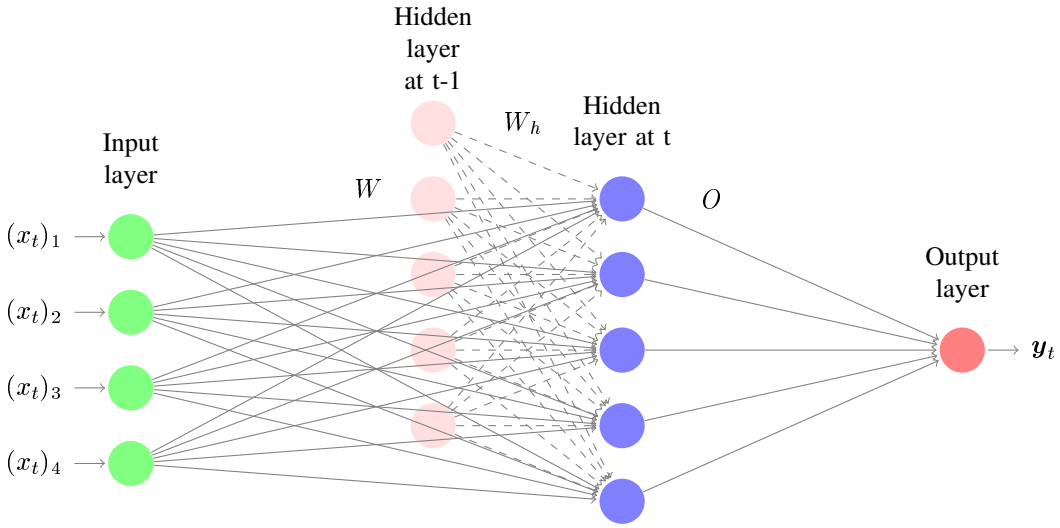

Fig. 1. Vanilla RNN with $n = 4$, $k = 5$ and $p = 1$.

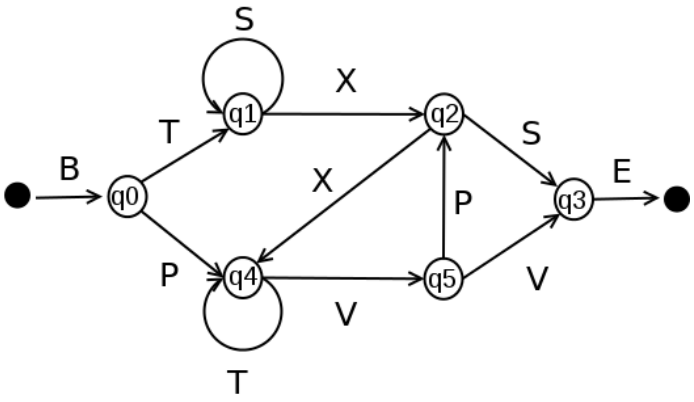

Fig. 2. Reber's grammar automaton.

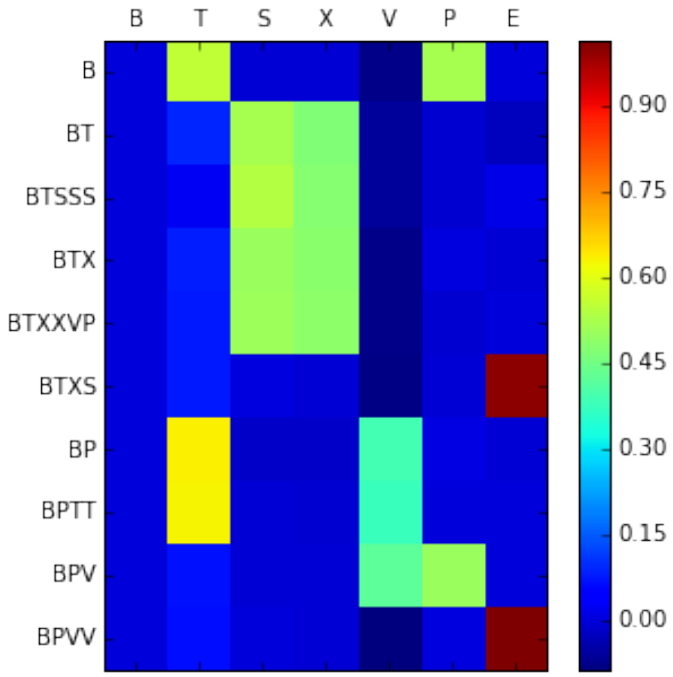

Fig. 3. Heatmaps of the output layer for different sequences on a 5-neurons Vanilla RNN model trained on Reber's gramm.

V. The (l,k)-Discrepancy

We now introduce a formal way to measure the discrepancy capability of a recurrent neural network. More precisely, we introduce a positive quantity that illustrates the ability a model has to propagate long term dependencies.

Definition V.1 ((l,k)-discrepancy)**.** Let us consider:

- Two integers $l, k \in \mathbb{N}$,
- A binary alphabet ($n = 2$), symbolically represented by $\mathbf{0}$ and $\mathbf{1}$,
- A RNN – either Vanilla RNN or GRU model – comprising k neurons,
- The words $u = \mathbf{0}\overbrace{\mathbf{0}\dots\mathbf{0}}^{l-1}$ and $v = \mathbf{1}\overbrace{\mathbf{0}\dots\mathbf{0}}^{l-1}$ differing only by their first bit,
- $\boldsymbol{h}_u$ and $\boldsymbol{h}_v$ the states where the model's hidden state lands after reading u and v.

Then the **(l,k)-discrepancy** $D(l, k)$ of the model is computed with the following process:

- Compute a numerous time (2000 in the following) $||\boldsymbol{h}_u - \boldsymbol{h}_v||_2$ for different random affectations of the model's parameters (sampled over $\mathcal{N}(0, 1)$ in the paper).
- Average these norms, it is $D(l, k)$.

The quantity $D(l, k)$ is integrally computed without training. It summarizes the capacity of the k-neurons network to distinguish between sequences of length l perfectly identical but their first bit.

Figure 10 depicts the evolution of $D(l, k)$ for various values of l and k. Code can be found at https://github.com/brain-

 ISBN: 978-1-61208-531-9

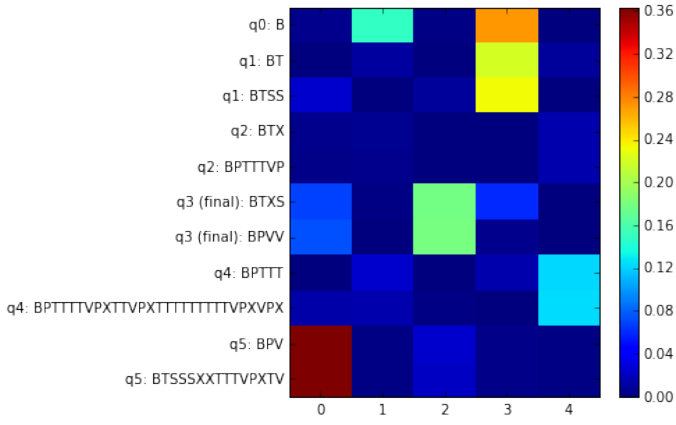

Fig. 4. Heatmaps of the hidden layer for a 5-neurons Vanilla on different observed sequences of a Reber's grammar. Corresponding automaton's states are also listed.

bzh/IARIA17_RNNs/tree/master/lk-discrepancy/.

It shows that as soon as l becomes large, the discrepancy ability of Vanilla RNNs becomes very close to 0, meaning that the two binary sequences are indistinguishable.

In comparison, Figure 11 depicts the discrepancy for the GRU model for various values of l and k. As we can see here, the discrepancy does not goes to 0 for large values of l, suggesting the ability of the dynamics of GRU to maintain long term information in the hidden state of the network.

These profiles provide an intuitive explanation of the results for the Reber's grammar: the Vanilla could succeed in the first problem because of the really short term memory required and definitely failed in the embedded case because of out-of-range dependencies.

VI. Conclusions and Future Research

Through Reber's grammar example we saw that Vanilla and GRU differenciate themselves more than just through learning and gradients arguments. This observation motivates future mathematical research to formally understand their intrinsic difference in term of dynamical systems. A good starting point could reside in the Echo States Network (c.f. [11]) theory and the mathematical tools it develops. Understanding this difference could lead to simpler RNN models than GRU/LSTM, less computationally heavy, and with better long term abilities.

Acknowledgements

This work was funded in part by the CominLabs project "Neural Communication".

References

[1] Pineda and F. J., "Generalization of back-propagation to recurrent neural networks", *Physical review letters*, vol.59, pp.19, 1987.

[2] Cho K., Merriënboer B., Bahdanau D. and Bengio Y., "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches", arXiv, 2014.

[3] Hochreiter S. and Schmidhuber J., "Long short-term memory", *Neural Computation*, vol.9, pp.1735-1780, 1997.

[4] Mikolov T., Karafiát, M., Burget, L., Cernocký, J. and Khudanpur S., "Recurrent neural network based language model", *Interspeech*, vol.3, pp.2, 2010.

[5] Graves A., Mohamed, A. and Hinton G., "Speech recognition with deep recurrent neural networks", *2013 IEEE international conference on acoustics, speech and signal processing.* , IEEE, 2013.

[6] Karpathy, A., and Fei-Fei, L., "Deep visual-semantic alignments for generating image descriptions", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2015.

[7] Sutskever, I., Oriol V. and Quoc V. Le., "Sequence to sequence learning with neural networks", *Advances in neural information processing systems*, 2014.

[8] Bengio Y., Sinard P. and Frasconi P., "Learning Long-Term Dependencies with Gradient Descent is Difficult", *IEEE transactions on neural networks*, vol.5, pp.2, 1994.

[9] Pascanu, R., Tomas M. and Bengio, Y., "On the difficulty of training recurrent neural networks", *International Conference on Machine Learning*, vol.3, pp.28, 2013.

[10] T. Siegelmann H. and D. Sontag E., "On the Computational Power of Neural Nets", *Journal of computer and system sciences*, vol.50, pp.132-150, 1995.

[11] Jaeger H., "The "echo state" approach to analysing ang training recurrent neural networks", TechReport, 2001.

 ISBN: 978-1-61208-531-9

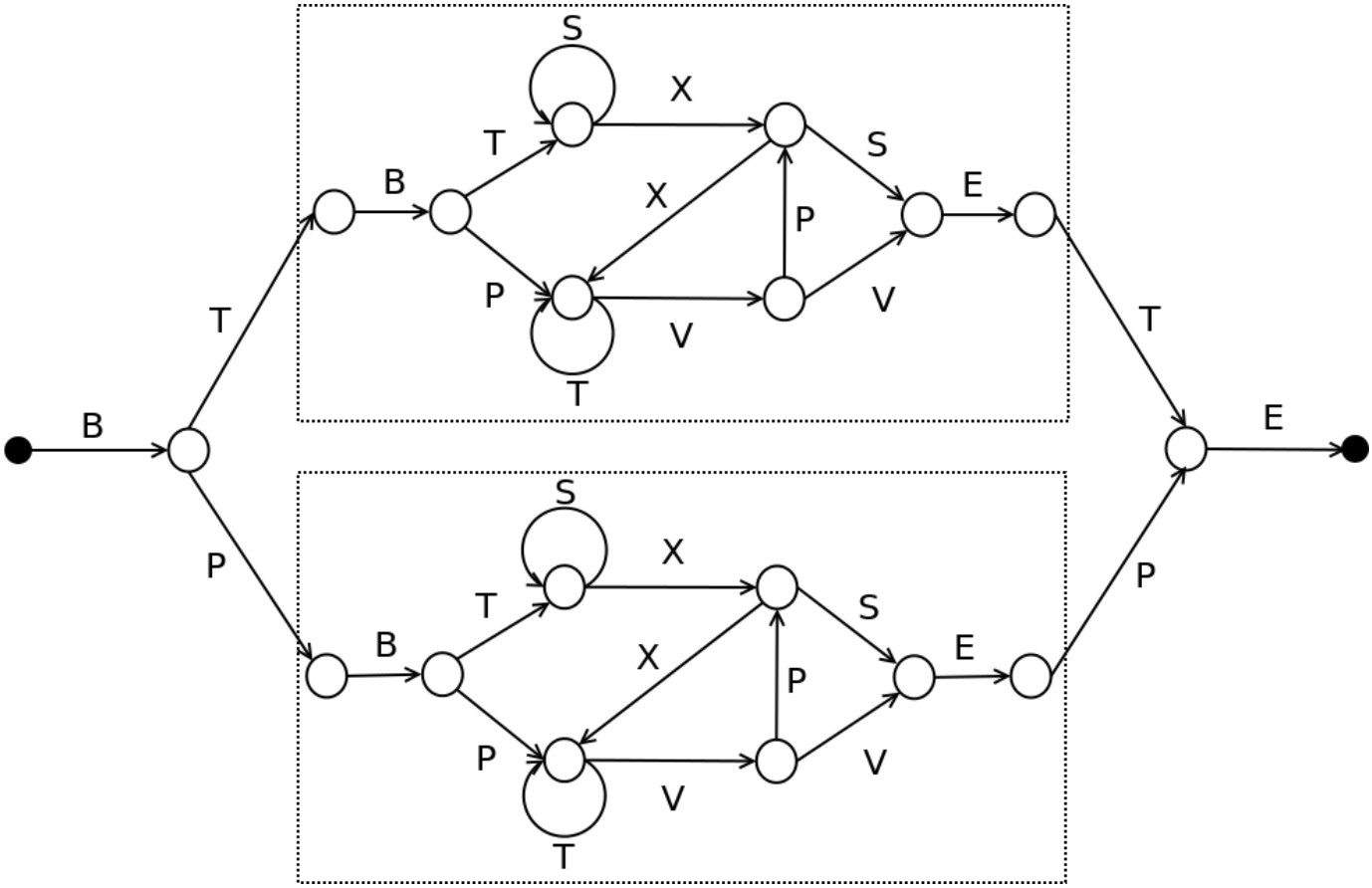

Fig. 5. A depiction of the automaton corresponding to the Embedded Reber's grammar.

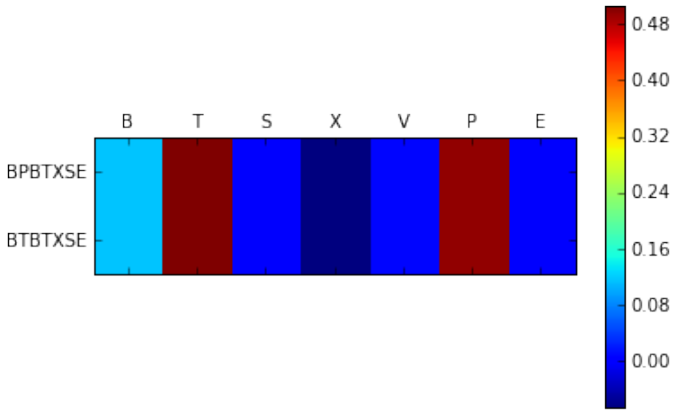

Fig. 6. Vanilla RNN prediction on several sequences of the Embedded Reber's grammar.

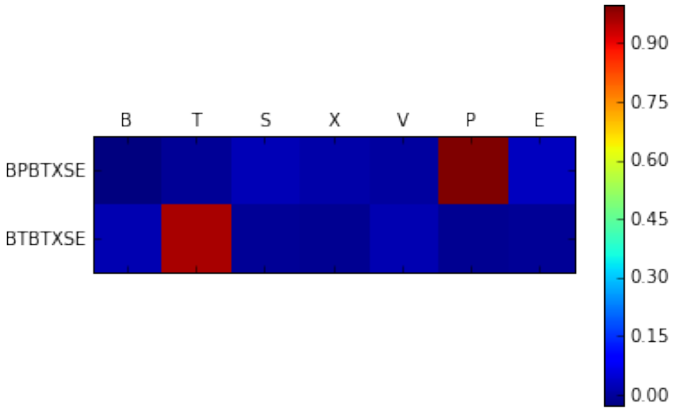

Fig. 7. GRU prediction on several sequences of the Embedded Reber's grammar. The expected letters are correct and reflect the fact the embedded Reber's grammar has been correctly inferred.

 ISBN: 978-1-61208-531-9

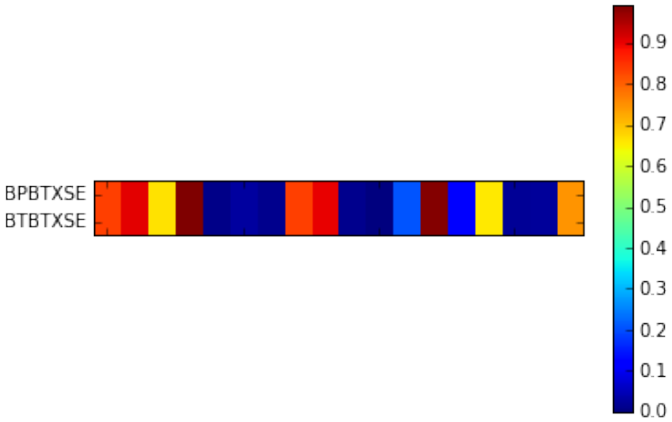

Fig. 8. Hidden layer of a 18-neurons Vanilla RNN after training on the embedded Reber's grammar.

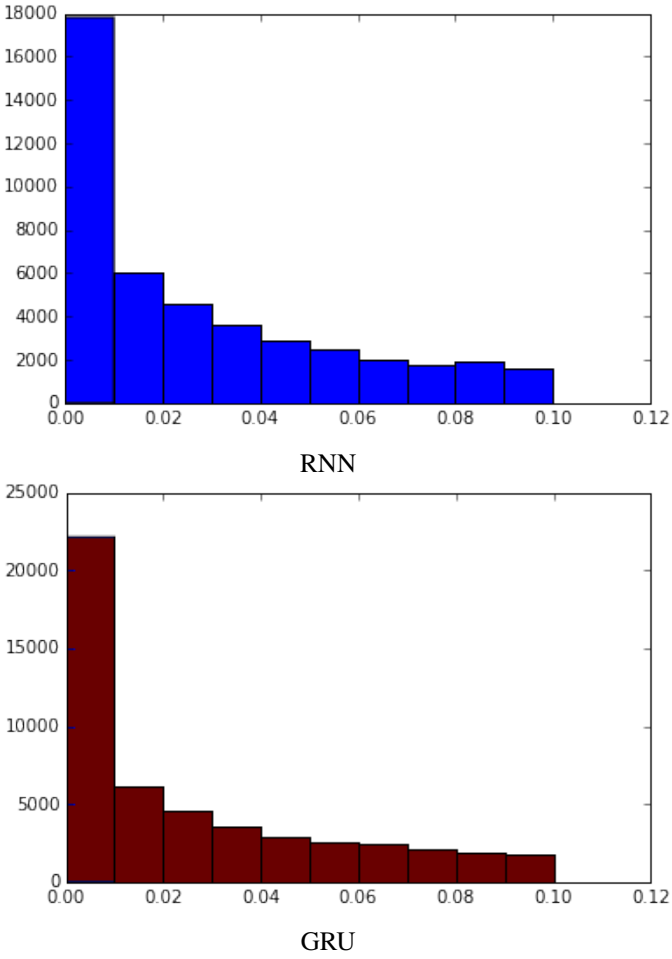

Fig. 9. Gradient's coefficients for both Vanilla RNNs and GRU models when learning the embedded Reber's grammar from 2000 examples.

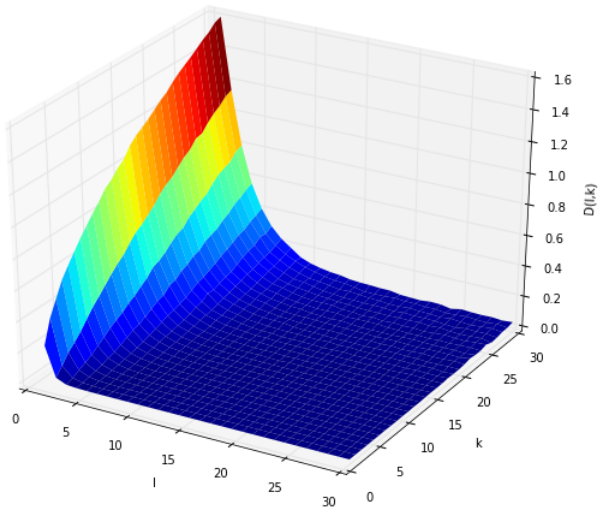

Fig. 10. Evolution of $D(l,k)$ for a Vanilla RNN.

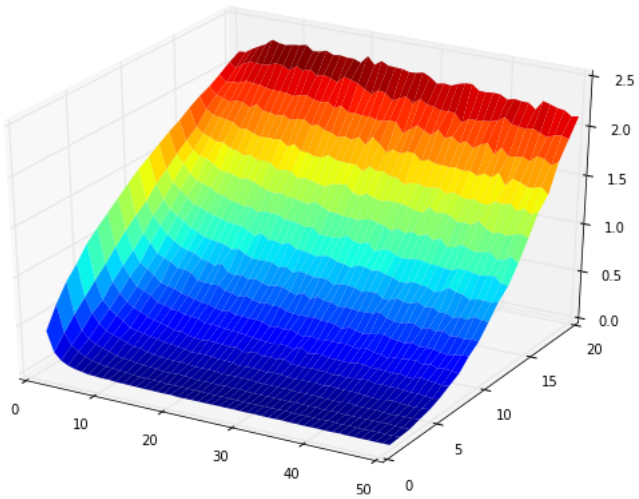

Fig. 11. Evolution of $D(l,k)$ for the GRU model.