Factorized Recurrent Neural Architectures for Longer Range Dependence

Francois Belletti **Alex Beutel** **Sagar Jain** **Ed H. Chi**
Google Research; belletti, alexbeutel, sagarj, edchi @google.com

Abstract

The ability to capture Long Range Dependence (LRD) in a stochastic process is of prime importance in the context of predictive models. A sequential model with a longer-term memory is better able contextualize recent observations. In this article, we apply the theory of LRD stochastic processes to modern recurrent architectures, such as LSTMs and GRUs, and prove they do not provide LRD under assumptions sufficient for gradients to vanish. Motivated by an information-theoretic analysis, we provide a modified recurrent neural architecture that mitigates the issue of faulty memory through redundancy while keeping the compute time constant. Experimental results on a synthetic copy task, the Youtube-8m video classification task and a recommender system show that we enable better memorization and longer-term memory.

Introduction

Recurrent Neural Networks (RNNs) [1] were introduced as a means to apply neural modeling to sequences in an attempt to benefit from their expressiveness and their ease of use as black-box models. In applications, particular emphasis has been put on modeling sequences of symbols for Natural Language Processing [2]. To understand languages, linear modeling is challenged by the non-linearity of syntactic dynamics, the need to infer a state in a discrete state space and the appeal of models whose memory can be human-like [3]. Recurrent Neural Networks are parametric non-linear models defined by a recurrent equation

$$[\hat{Y}_{t+1}, M_{t+1}]^T = \Phi_\theta \left(X_t, M_t \right).$$

As we show in the upcoming analysis, the prescription of a bounded state (M) in RNNs to enforce stability is problematic. Squashing non-linear functions destroy information as they attempt to propagate memory through time. Modern popular architectures such as LSTMs [4] or GRUs [5] try to correct the issue by instantiating a separate memory bank whose read/write patterns are functions of the previous memory state and the latest observation. We show that the use of differentiable squashing functions in the form of sigmoid layers as gates, in order to make the read/write behavior of the RNN learnable, leads to leaky gates. In particular, we prove that such gates constantly erase information under reasonable assumptions on the input process.

After having delineated this curse of short memory in LSTMs and GRUs, under common assumptions for RNNs, we devise a computationally efficient strategy to make the memory bank less likely to forget by a simple mechanism: enforcing redundancy. We present the corresponding factorization technique to expand the size of the memory of RNNs. The resulting factorized RNNs feature advantageous properties in terms of time and memory complexity. As we factorize our architecture we provide a larger memory space without the need for more network parameters or computation as is the case with dense unstructured larger RNNs.

Contributions and organization of the paper

1. **Theoretical limitations of classic RNNs:** A novel analysis of LSTMs and GRUs as dynamical systems is presented in Section 1, giving proofs of their inability to provide a LRD memory under assumptions that were proven sufficient in [6] for gradients to vanish.
2. **New modeling insights:** As we seek to render faulty memorization mechanisms less likely to forget, a factorized architecture providing extra memory capacity in a redundant manner is presented in Section 2.
3. **Experimental evidence:** A thorough experimental study shows in Section 3 on different tasks that factorization provide better performance thanks to a better ability to memorize.

Related work

The argument of the present paper relates work concerned with the difficulty of training RNNs to the statistical notions of Long and Short Range Dependence. Most of the

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

work on long memory and RNNs focuses on the difficulties arising during training [7, 8, 4, 6, 9]. To train a recurrent neural network we indeed rely on Back-Propagation-Through-Time [7, 8, 4] which unfolds the recurrent cell along K time-steps where K is for instance the length T of a sequence of interest. This typically creates $O(K)$ non-linearities of the sigmoid or hyperbolic tangent family along the gradient propagation path which are well known to cause vanishing gradient issues [7]. Such squashing operations have been designed to enable Long and Short Term Memory (LSTM) [4] as they maintain the state of the RNN bounded thereby preventing state explosion. Constraining the memory to belong to $[-1, 1]$ guarantees numerical stability but jeopardizes gradient propagation by adding saturating functions. The difficulty of learning LRD in recurrent models has been connected in [6] to some properties of bifurcations in non-linear system theory [10, 11]. The current paper draws connection between vanishing gradients and the absence of Long Range Dependence [12].

1 LONG RANGE DEPENDENCE

The current section delves into the issues created by the presence of bounded non-linearities in RNNs. We first recall the pre-existing analysis of the impact of saturation on gradient propagation prior to stepping away from this approach and use stochastic process theory.

1.1 Beyond the issue of vanishing gradients

In order to guarantee stability of the RNN, the mainstream solution has been to squash the generated candidate so that it remains in a bounded domain. Such a blunt way of guaranteeing stability first led to the design of RNNs described by dynamics [1]

$$[\hat{Y}_{t+1}, M_{t+1}]^T = \tanh(AX_{t+1} + BM_t) \tag{1}$$

where the input $X_{t+1} \in \mathbb{R}^{d_i}$, the predictions $\hat{Y}_t \in \mathbb{R}^{d_o}$, the candidates (state) $M_{t+1} \in \mathbb{R}^{d_h}$ and the learned parameter matrices A and B are respectively in $\mathbb{R}^{d_o+d_h,d_i}$ and $\mathbb{R}^{d_o+d_h,d_h}$.

1.1.1 Tension between memory and stability in Recurrent Neural Networks

In the linear model case, constraints can be set on the family of operators $\{\Phi_\theta | \theta \in \Theta\}$ in order to guarantee that the model will not become unstable and explode as it reads a sequence of inputs $(x_1, \ldots, x_T)$. The strategy applied to the first Recurrent Neural Networks relies on a careful choice of the non-linearities within Φ rather than imposing constraints on the parameter domain Θ. Then hyperbolic tangent function in (1) bounds the state and therefore guarantees stability.

Here we can notice the shift in the way stability is guaranteed from linear models to RNNs. The stability inducing design choice is therefore very blunt and comes with a series of drawbacks that make learning Long Range Dependence challenging and may create issues when trying to make predictions in an unbounded domain such as $\mathbb{R}^d$.

1.1.2 Bounded non-linearities

Squashing states with a function such as $\tanh$ comes at the cost of making gradient back-propagation very difficult over long sequences of inputs.

The vanishing gradient issue in RNNs has been highlighted very early as their main shortcoming as it hinders their ability to learn long term dependencies [3]. If we apply Back-Propagation-Through-Time to the first generation RNN in Eq. (1) we find indeed with an additive loss function

$$\mathcal{L}_\Phi = \frac{1}{N} \nabla_\theta \sum_{i=1}^{N} \sum_{t=t_i^s}^{t_i^e} l(Y_{t+1} - \hat{Y}_{t+1}) \tag{2}$$

that its gradient involves long multiplication chains as:

$$\nabla_\theta \sum_{t=1}^{T} \Big\{ l(\hat{Y}_t, Y_t) \Big\} = \sum_{t=1}^{T} D_1 l(\hat{Y}_t, Y_t) \times \left(\sum_{s=0}^{t-1} \prod_{r=1}^{s} D_2 \Phi_\theta(X_{r+1}, M_r) \nabla_\theta \Phi_\theta(X_{s+1}, M_s) \right). \tag{3}$$

where the many multiplications by $tanh'(AX_{r+1} + B\hat{Y}_r) \times B$ between the output of interest at time t and the source input at time s are likely to vanish in magnitude.

Seeking stability of the RNN as a dynamical system has led to the impossible conciliation of the incentive to bound the state for stability and the need for non-vanishing gradient to learn. One solution in more modern architectures consists of using a separate memory to keep track of the context.

1.1.3 Studying RNNs as dynamical systems

The remainder of the theoretical study proves that the separate memorization space modern RNNs (e.g. GRU and LSTM) allocate to materialize a latent state is not LRD under conditions that guaranteeing gradients are vanishing because of the presence of saturating read/write learnable gates. We give guarantees on RNN as a dynamical system rather than analyze its training procedure in order to delineate clear conclusions on its ability to memorize.

We choose to study whether the family of parametric models we consider can provide LRD to the candidate vector which contextualizes their predictions. The theory we consider is concerned with linear systems [13], non-linear systems [14] and recent developments in LRD [12, 15, 16].

1.2 Dynamical systems and Long Range Dependence

We adopt the definition of a Short Range Dependent (SRD) multivariate process given in [12]. The definition corresponds to the assumptions of Short Range Dependences needed in most of standard proofs of convergence for second order estimation for stationary processes [13].

Definition 1.1 *SRD Multivariate process: A process is SRD if and only if there exists $M \in \mathbb{R}$ such that*

$$\exists t \in \mathbb{N},\ \sum_{h \in \mathbb{N}} \left\| Cor\left(X_t, X_{t+h}\right) \right\|_2 < M.$$

In the present paper "Cor" stands for correlation. In the second order stationary setting the definition trivially reduces to $\sum_{h \in \mathbb{N}} \left\| \mathrm{Cor}\left(X_0, X_h\right) \right\|_2 < +\infty$ as $\mathrm{Cor}\left(X_t, X_{t+h}\right)$ only depends on h. We do not make such assumption in the following and we will indicate under which stationarity conditions the properties we highlight hold.

Lemma 1.1 *Short Range Dependent recursion: Consider a stochastic process $(y_t)_{t \in \mathbb{N}}$ in $\mathbb{R}^d$ with $E(y_t) < +\infty$ such that*

$$y_{t+1} = A_t y_t + \epsilon_t$$

where $\forall t \in \mathbb{N},\ \forall s \geq t, E(y_t \epsilon_s^T) = 0$. If there exists $\alpha > 0$ so that the spectral radii of the operators $\{A_t | t \in \mathbb{N}\}$ are uniformly bounded by $1 - \alpha$ then the process (y) is SRD.

Proof 1.1 *One can write*

$$y_{t+1} = \left(\prod_{s=0}^{t} A_s \right) y_0 + \sum_{s=0}^{t} \left(\prod_{h=s}^{t} A_h \right) \epsilon_s, \text{ then } E\left[y_{t+1} y_0^T \right]$$
$$= \left(\prod_{s=0}^{t} A_s \right) E\left[y_0 y_0^T \right] + \sum_{s=0}^{t} \left(\prod_{h=s}^{t} A_h \right) E\left[\epsilon_s y_0^T \right].$$

Then by using the fact that the Frobenius norm is algebraic with respect to the matrix multiplication operator and the assumption $\forall s \geq t, E(y_t \epsilon_s^T) = 0$,

$$\left\| E\left[y_{t+1} y_0^T \right] \right\|_2 \leq \left(\prod_{s=0}^{t} \left\| A_s \right\|_2 \right) \left\| E\left[y_0 y_0^T \right] \right\|_2.$$

The assumption on the spectral radius of A implies that $\prod_{s=0}^{t} \left\| A_s \right\|_2 \leq (1 - \alpha)^t$ which concludes the proof. ■

1.3 Analysis of LSTM as a dynamical system

Let us remind the reader of the dynamical equations defining the LSTM [4] RNN with input (x_t) and output (h_t):

$$\begin{aligned}
\text{Forget gate:} \quad & f_t = \sigma(W_x^f x_t + W_h^f h_{t-1} + b^f) && (4)\\
\text{Input gate:} \quad & i_t = \sigma(W_x^i x_t + W_h^i h_{t-1} + b^i) && (5)\\
\text{Candidate:} \quad & \tilde{c}_t = \tanh(W_x x_t + W_h h_{t-1} + b) && (6)\\
\text{Context update:} \quad & c_t = f_t \odot \tilde{c}_t + i_t \odot c_{t-1} && (7)\\
\text{Output gate:} \quad & o_t = \sigma\left(W_x^o x_t + W_h^o h_{t-1} + b^o\right) && (8)\\
\text{Output:} \quad & h_t = o_t \odot \tanh(c_t) && (9)
\end{aligned}$$

where $x_t \in \mathbb{R}^d$, $h_t \in \mathbb{R}^{d_h}$, $c_t \in \mathbb{R}^h$, $W_d \in \mathbb{R}^{d,d}$, $W_h \in \mathbb{R}^{d_h,d}$, $b \in \mathbb{R}^h$, $f_t \in \mathbb{R}^{d_h}$, $i_t \in \mathbb{R}^{d_h}$, $o_t \in \mathbb{R}^{d_h}$.

1.4 Analysis of GRU as a dynamical system:

Let us remind the reader of the dynamical equations defining the GRU [5] RNN with input (x_t) and output (h_t):

$$\begin{aligned}
\text{Read gate:} \quad & r_t = \sigma(W_x^r x_t + W_h^r h_{t-1} + b^r) && (10)\\
\text{Update gate:} \quad & u_t = \sigma(W_x^u x_t + W_h^u h_{t-1} + b^u) && (11)\\
\text{Candidate:} \quad & \tilde{h}_t = \tanh(W_x r_t \odot x_t + W_h h_t + b) && (12)\\
\text{Context update:} \quad & h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t && (13)
\end{aligned}$$

where $x_t \in \mathbb{R}^d$, $c_t \in \mathbb{R}^{d_h}$, $W_x \in \mathbb{R}^{d_h,d}$, $W_h \in \mathbb{R}^{d_h,d_h}$ $b \in \mathbb{R}^{d_h}$, $r_t \in \mathbb{R}^{d_h}$, $u_t \in \mathbb{R}^{d_h}$. While the dimension of the input is d, the memory and output have dimension h.

1.4.1 Sigmoid gates and LRD

Lemma 1.2 *Sigmoid gate induces short memory: If the operators $W_x \in \mathbb{R}^{h,d}$, W_h in $\mathbb{R}^{d_h,d_h}$ and the bias vector $b \in \mathbb{R}^{d_h}$ take finite values and the input processes (x_t) and (h_t) are bounded then there exists $\alpha_g > 0$ such that*

$$\left\| \sigma(W_x x_t + W_h h_t + b) \right\|_\infty \leq (1 - \alpha_g).$$

Proof 1.2 *The property is a direct consequence of the definition of the sigmoid function σ and the uniform boundedness of the process $(W_x x_t + W_h h_t + b)$.* ■

The lemma above, combined with lemma 1.1 implies immediately that the presence of gated re-memorization mechanisms in the context update equations of the LSTM network (7) and the GRU network (13) only enables SRD. Therefore we need to rely on the $\tanh$ based candidate generation mechanism in the LSTM (6) and the GRU (12) to enable LRD memory.

1.4.2 Candidate generation in GRU and LSTM and LRD

We use theoretical arguments to prove in the following that under reasonable assumptions LRD cannot occur under the condition presented in [6] as sufficient for gradients to vanish. The proofs only complement pre-existing literature about how basic $\tanh$ based RNNs have difficulties learning LRD patterns.

The following proposition furthers the attempt to relate train time misbehavior of RNNs with the properties of the corresponding dynamical system presented in [6]. The work therein explains how some bifurcations between attraction basins of chaotic attractors [11, 10] of a $\tanh$ RNN that were unraveled in [17] can be related to training time exploding gradients. Such study was

limited to the uni-variate setting. The work in [6] also gives a sufficient conditions at training time on the Jacobian of the RNN to create vanishing gradient issues. Such are the sufficient conditions that were given for multi-variate RNNs:

Proposition 1.1 *Sufficient conditions for gradients to vanish [6]: If the spectral radius of W_h is < 1 then gradient vanish in back-propagation through time.*

We complete the work relating sensitivity to inputs and memory in a more general setting with the following proposition.

Proposition 1.2 *Relating vanishing gradient to SRD: If the spectral radius of W_h is < 1 and the series (h_t) is of constant variance then the RNN is SRD.*

Proof 1.3 *Let us denote F_t the mapping*

$$(h_0, \dots, h_t, x_0, \dots, x_t) \to h_{t+1} = F_t\left(h_0, \dots, h_t, x_0, \dots, x_t\right)$$

is such that the partial Jacobian $D_{h_0} F_t$ has a spectral radius uniformly $< \lambda^t$ where $\lambda < 1$ with the assumption that the spectral radius of W_h is < 1. (a proof can indeed quickly be given by induction as in [6]). Let us then write $E[h_0 h_{t+1}]$ as

$$\int_{h_0 \in B^\infty(1)} h_0 E[F_t\left(h_0, \dots, h_t, x_0, \dots, x_t\right)|h_0]^T d\mu_0$$

where μ_0 is the probability distribution of h_0. We can devise a function $u^*(h_0, \dots, h_t, x_0, \dots, x_t)$ such that $E[h_0 h_{t+1}] = \int_{h_0 \in B^\infty(1)}$

$$h_0 E[h_t|h_0]^T E[D_{h_0} F_t(u^*(h_0, \dots, h_t, x_0, \dots, x_t))|h_0]^T d\mu_0$$

and with the bound on the spectral radius of $D_{h_0} F_t$ we conclude the proof. ■

Theorem 1.1 *Short range dependent context: Under the assumptions of propositions 1.1, 1.2 and 1.3, the context of a GRU or LSTM is SRD if its variance is constant through time under conditions that guarantee vanishing gradients.*

Proof 1.4 *The theorem is a direct consequence of the application of lemma 1.1 after proposition 1.3.* ■

The theorem above show that the dynamics defining LSTM and GRU networks with vanishing gradients only lead to the calculation of a SRD context under the assumption that the operators involved are full rank, that the series of inputs are not entirely deterministic given the filtration associated with the context and the context has constant variance. Any feed-forward post-processing pipeline is likely to be indeed be hampered if the variance structure of (h_t) change through time.

The theorems above establish a link between vanishing gradients and Short Range Dependence. As in [6] we manage to relate sensitivity analysis of RNNs with their memory as dynamical systems. A difference though is that we conduct our analysis entirely in the multi-variate setting.

1.4.3 Information theoretical insights for longer memory

Let us delve into the issues of Short Range Dependence following an information theoretical approach in order to identify how information is memorized in the network. The following proposition shows that the candidate generation we consider can only lead to memorizing inputs in a bounded space perturbed by a non-vanishing amount of noise.

Proposition 1.3 *Decomposition of candidate generation: Assume the input process (x) is bounded and that there exists $\beta > 0$ such that $\forall t \in \mathbb{N}, \|x\|_\infty > \beta$. Assume also that there exists $\gamma > 0$ such that $\forall t \in \mathbb{N}, Var(x_t|h_{s\leq t}) > \gamma$. If the $d \times d$ matrices W_x and W_h are full rank then there exists a function $\phi_t : \mathbb{R}^d \to \mathbb{R}^d$ and a stochastic process (ϵ) with such that $\forall s \geq t, E(h_t \epsilon_s^T) = 0$, $Var[\epsilon_t|h_{s\leq t}] > \nu^2 > 0$ and*

$$\tanh(W_x x_t + W_h h_t + b) = \phi_t(h_t) + \epsilon_t.$$

The assumption we make on the infinite norm of the input process stems from the fact that most successful applications of RNNs in Natural Language Processing [18], automated translation [3, 19] or recommendation rely on an embedding to represent input data. In such a setting, all inputs (x_t) belong to a finite dictionary of learned vectorial representations. Learning such representations with SGD leads to non zero encoding vectorial representations (on all coordinates).

The assumptions we make about the rank of the learned linear operators stem from the practical setting of the use of such models after training by Stochastic Gradient Descent (SGD). In such a setting, the accumulation of noisy gradient updates prevents rank deficiency in W_x and W_h.

Finally, the assumption on the conditional variance of the input process given the state means that we consider a non-degenerate innovation process is entailed in the input. In other words, we consider that new inputs cannot be exactly predicted by a series of previous states. The assumption is natural as if the input could be linearly predicted by the state observing them would not bring novel information in the RNN.

Proof 1.5 *With the assumption on the rank of W_h, there exists $\beta' > 0$ such that $\forall t \in \mathbb{N}$, $\forall i \in 1 \dots d$, $|(W_h h_t)_i| > \beta'$. Using the theorem of intermediary values, one can therefore write*

$$\begin{aligned}
&\tanh(W_x x_t + W_h h_t + b) \\
&= \tanh(W_h h_t + b) + (1 - \tanh^2)(u^*) \odot (W_x x_t) \\
&= \tanh(W_h h_t + b) + g_t(h_t) + \epsilon_t.
\end{aligned}$$

where

$\forall i \in 1 \dots d, u_i^* \in ((W_h h_t + b)_i, (W_h h_t + W_x x_t + b)_i)$,

$g_t(h_t) = E\left[(1 - \tanh^2)(u_t^*) \odot (W_x x_t)|h_{s\leq t}\right]$ *and*

$\epsilon_t = (1 - \tanh^2)(u_t^*) \odot (W_x x_t) - E\left[(1 - \tanh^2)(u_t^*) \odot (W_x x_t)|h_{s\leq t}\right]$.

By definition of ϵ_t, $E\left[\epsilon_t|h_{s\leq t}\right] = 0$ therefore, $\forall s \leq t, E[\epsilon_t h_s] = 0$.

By uniform boundedness of $W_x x_t + b$ and $W_x x_t + W_h h_t + b$, there exists $\gamma' > 0$ such that $\min_{i=1\dots d} |(1 - \tanh^2)(u_t^*)_i| \geq \gamma'$ and as $Var[x_t|h_{s\leq q}] > \gamma$ with the fact that W_x is full rank, we conclude the proof. ■.

Such a proposition shows that as the RNN puts novel information into its memory, a non-vanishing amount of noise impacts all the channels of the memory bank.

Information theory helped [20] characterize the capacity per neural unit of some RNNs. We now employ some information theoretical insights to understand more formally the influence of the number of channels we employ H on the mutual information between h_t and h_{t+1}:

$$I(h_{t+1}, h_t) = H(h_{t+1}) + H(h_t) - H(h_{t+1}, h_t)$$

where H denotes the entropy of random variable [21, 22]. Let us remind the reader of a more general statement made in [23] on the impact of additive Gaussian noise in the multivariate setting.

Theorem 1.2 *Additive Gaussian noise and mutual information [23, 24]: Consider two random variables X and Y in $\mathbb{R}^H$ linearly related by*

$$Y = \eta W X + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \Sigma)$,

$$I(X, Y) = \frac{1}{2} \log \left(I + \eta \Sigma^{-1/2} W^T W \Sigma^{-1/2} \right).$$

The theorem may be immediately used to prove the following:

Proposition 1.4 *Mutual information with Gaussian noise model in RNN: Assuming that $x_t \sim \mathcal{N}(0, diag(\sigma, \ldots, \sigma))$, W_h is full rank with d_h representation dimensions*

$$I(h_t, \tanh(W_x x_t + W_h h_t + b)) = \frac{d_h}{2} \log(1 + \frac{\eta}{\sigma})$$

Proof 1.6 *Let us start by using the fact mutual information is invariant under reparametrization by diffeomorphisms [25]. Therefore, $I(h_t, \tanh(W_x x_t + W_h h_t + b)) = I(W_h h_t, W_x x_t + W_h h_t)$, and a direct application of 1.2 concludes the proof.*

In order to provide a amount of memory in the corresponding noisy encoding setting, we therefore choose to increase the number of memory channels. The following section shows how a redundant recurrent architecture leads to the formation of a higher numbers of encoding channels while keeping the number of parameters of stacked RNNs unchanged. In particular, Theorem 1.1, Proposition 1.3 and 1.4 provide new theoretical insights on LRD in RNNs.

2 FACTORIZED RECURRENT ARCHIECTURES

The previous section highlighted the shortcomings of popular RNNs relying on saturating candidate generation and gating mechanisms. In particular, the memory appears to leak at each time step the RNN is applied. We now develop a strategy to mitigate the issue.

2.1 Modeling challenges

The need for larger memory: One way to render the memory more LRD consists rather trivially in extending it. By allowing for more dimensions to represent a memorized past event, it is more likely that information will be preserved over longer periods of time. In a RNN featuring a candidate generation W_h as in a LSTM or GRU one issue though is the quick rate at which the number of parameters and the computational burden increases with respect to the dimension of the memory. The corresponding rate is indeed quadratic in the size of the memory space. Therefore, we may want to find a strategy to obtain a larger memory without adding too many parameters.

Tight computational constraints: As recurrent models are served in applications involving latency-sensitive predictions such as recommender systems, we are looking for a solution that keeps the compute time of the network we consider mostly unchanged.

2.2 Block diagonal RNNs

Combining the insights above, we understand a larger memory with compartmented gates is suitable to mitigate the issue of non LRD contextualization. Such is the architectural modification we bring to the LSTM and GRU architectures we consider. In the present paper, we refer to the resulting RNNs as "factorized" recurrent neural networks [26]. While our approach is driven by theory we relate it to similar modifications given to LSTMs with large numbers of parameters to speed up their serving in [26]. In the present paper our insight and aim are different. Instead of seeking a strategy to accelerate a network without excessive degradation of its performance we want to enable longer term memory.

We show in the last experimental section that we indeed manage to improve performance, thanks to better memorization abilities, with an equal number of parameters. Furthermore, we give a different interpretation of the factorization procedure as providing another degree of freedom in the specification of the neural architecture that decouples the size of the memory bank from the capacity of the parametrized layers.

2.2.1 Specification of architecture

Let us specify the factorized LSTM and GRU architectures formally. Let g be an even divider of the output dimension h and the input dimension d_h. A "factorized" architecture imposes a g-block-diagonal structure to all the linear operators involved in the RNN. Imposing such structure is equivalent to dividing the input and output spaces into g sub-spaces of equal dimensions and instantiating a RNN on each of the sub-space. Consider inputs $X_t \in \mathbb{R}^{d_h}$, and outputs $\widehat{Y}_{t+1} \in \mathbb{R}^{d_h}$. The inputs are divided into g separate ${}^{d_h}/_{g}$-dimensional vectors $\left\{ X_t^i | i = 1 \ldots g \right\}$ which yields g outputs and states $\left\{ \hat{Y}_{t+1}^i, M_{t+1}^i | i = 1 \ldots g \right\}$ – each of ${}^{d}/_{g}$ dimensions – as one applies g mappings with distinct parameter sets $\left\{ \theta^i | i = 1 \ldots g \right\}$. Prior to concatenating the g outputs of ${}^{d}/_{g}$ dimensions each, one can write factorized recurrent block-diagonal architecture as follows:

$$\forall i \in \{1 \ldots g\}, \; [\hat{Y}_{t+1}^i, M_{t+1}^i]^T = \Phi_{\theta^i} \left(X_t^i, M_t^i \right). \qquad (14)$$

2.2.2 Manipulating memory and learning capacity independently

Grouped LSTMs have been introduced primarily to obtain faster, less memory-demanding recurrent neural networks. One issue with a recurrent neural network with d-dimensional inputs and d_h-dimensional contexts is that the number of parameters grows as $O(((d + d_h) \times d_h)$ while requiring a dense matrix multiply requiring $O(((d + d_h) \times d_h)$ add/multiply operations. A grouped recurrent network is obtained by imposing a block-diagonal structure to the dense linear operators used within the network. If g groups are used, the number of parameters and add/multiply operations is divided by g. That is, with g groups we have $O(g \times ((d + d_h) \times d_h)/g^2)$ compute and memory complexity.

We further improve the performance of the "factorized" architecture's memory in contexts where regime switches are expected.

2.2.3 Block diagonal context switching RNN

As we consider an architecture splitting the input and output spaces and unrolling a RNN on each subspace prior to concatenating the outputs, we can think of each subspace as dedicated to

a particular modality of the input. Such a feature is particularly appealing in the setting of context-switching sequences.

As a user browses the web for instance, they are most likely going through different behavioral models. Professional browsing, leisure related browsing, news related browsing, purchase related browsing can be thought as such different modalities. When trying to predict the next page a user will browse in order to serve an appropriate modification, it may be worthwhile switching context between multiple small models, each of which is a specialist in a particular modality.

One of the advantages of the factorization, which we actually use for our recommendation system numerical experiment, is that enabling such context switching is immediate. We instantiate an auxiliary RNN with g outputs. The outputs are normalized and applied as a mask to the g input subspaces and/or g output subspaces. Such a scheme merges the mixture of expert factorization presented in [27] while contextualizing the switching between experts as the dispatching auxiliary network is itself recurrent. It is noteworthy that such an extension comes at a small computational cost as g is typically small as compared to d and h. We illustrate the architecture in Figure 5.

3 NUMERICAL EXPERIMENTS

The previous section described how out theoretical study informed our design decisions in order to enable longer memory in the RNNs we consider. We now gather empirical evidence showing factorization improves the memory of GRUs and LSTMs.

3.1 Experiments on synthetic data

The first experiment we consider is designed to challenge the ability of RNNs to remember their first input. Inspired by the copy task in [28] we task our RNNs with reading a uniformly distributed random input binary pattern of 6 symbols in $\{0, 1\}^8$ and repeating it. That is, the alphabet we consider has 256 possible letters and a word consists of 6 characters. For 4 repeats, given an input sequence $(\text{word}, 0, 0, 0, 0)$ the RNN has to output $(\text{word}, \text{word}, \text{word}, \text{word}, \text{word})$. All samples generated for a batch are new and therefore there is no train-set/test-set split.

In order to demonstrate a differential in the ability to memorize between our baseline models and their factorized counter-parts we progressively increase the number of times the first word is to be repeated. The model we consider has one hidden layer as we stack two RNNs one on top of another, which is standard in RNNs as they are applied to tasks such as automated translation [29] or speech recognition [30]. We train the networks with RMSProp [31] with minimal hyper-parameter tuning.

The results presented in Figures 1, 2 and 3 demonstrate that the factorized architectures offers better performance in copy task involving long term memory than the baselines we consider. Although the memory space is larger, the compute and memory burdens are identical to the baseline.

3.2 Youtube video classification

Now that we have demonstrated the better performance factorized architectures provide on the LRD synthetic copy-task, let us focus on a real world example LRD very high dimensional sequence classification experiment: the Youtube-8m challenge [32].

The Youtube-8m dataset is a massive dataset of pre-featurized videos frames [32] which leaves researchers with the task of

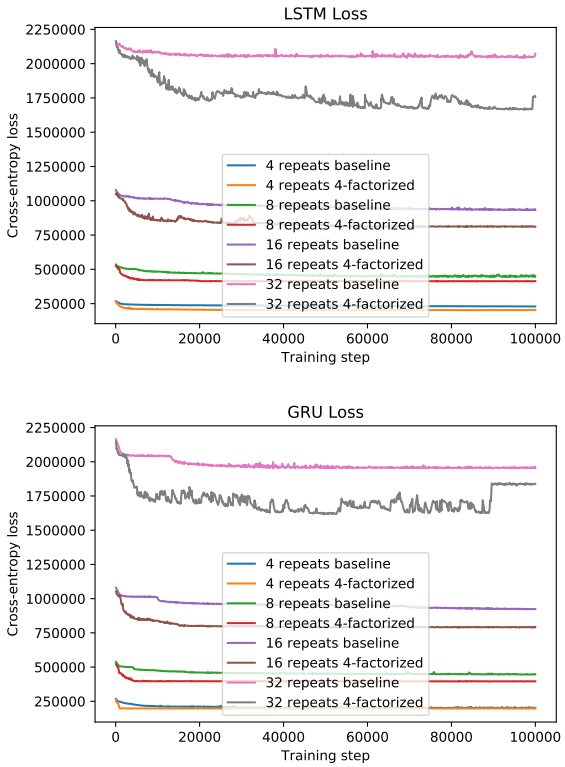

Figure 1: The cross-entropy losses of LSTM (top), GRU (bottom) is smaller when they are factorized to allow for a larger memory space at equal number of parameters. We can appreciate how the gap widens as the number of repeats increases from 4, to 8, 16, and finally 32.

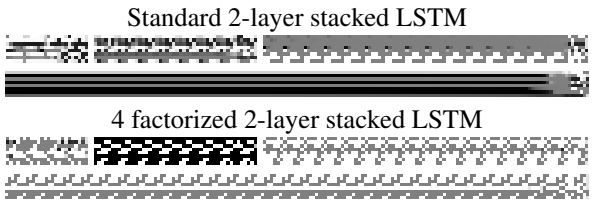

Figure 2: As we ask to copy an initial random symbol 4, 8, 16 and 32 times, our 4 factorized LSTM performs better on the LRD copy task than a its standard counterpart with the sample number of parameters and computations.

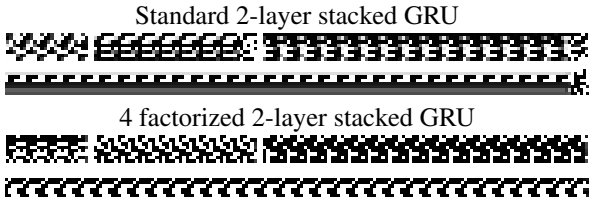

Figure 3: As we ask to copy an initial random symbol 4, 8, 16 and 32 times, our 4 factorized GRU performs better on the GRU copy task than a its standard counterpart with the sample number of parameters and computations.

Model	Hit@1	Perr
Baseline LSTM [32]	0.645	0.573
4-Factorized LSTM small	0.719	0.604
4-Factorized LSTM big	**0.809**	**0.641**

Table 1: Test performance on the Youtube-8m classification experiment. Hit @ 1 rate and Precision at equal recall. The small 4-Factorized as 4 times as few parameters than the baseline and the same amount of memory. The big 4-Factorized has the same number of parameters than the baseline and a larger memory.

building architectures that leverage the temporal structure of the featurized input. This enables the 2-layer LSTM presented in [32] to achieve state of the art results in terms of video classifications. We compare the 2-layer baseline with a factorized architecture where $g = 8$. In this section we try two alternate versions of the factorized architecture: one version uses block diagonal operators of the same size with 8 blocks thereby decreasing the number of parameters and amount of computations needed by a factor of 8, a second version corresponds to the normal factorized architecture we propose where the number of parameters and computations is the same as the baseline but the memory space is 8 times as big. We use ADAM [33] for training with the very same hyper-parameters that were employed to establish the baseline in [32].

As revealed by Figure 4 and Table 2, the factorization we propose offers substantial gains of performance. Classifying a video sequence is indeed a LRD as frames located at the beginning of the video inform the classification decision.

3.3 Latency sensitive recommendations

Neural recommender systems attempt at foreseeing the interest of users under extreme constraints of latency and scale. Recent examples of RNN networks employed to serve recommendations can be found in [34] and in [35] where they provide cutting edge performance. In order to identify the preferences of a given user based on their history, one may need to reach for information located far back into the past to predict the next items for which there should be an impression. Guessing the next item that the user consumes is considered the correct answer. Such a problem setting is indeed common in collaborative filtering [36, 37] recommendations.

3.3.1 Baseline

The baseline model at the core of the work aims at providing recommendations to users accessing a browsing page where impressions are displayed and having previously used the website with the same personal account. The model of interest is a neural sequential predictor mapping a sequence of observations $(t(1), X(1)) \dots (t(T), X(T))$ to a predicted item ξ. Here each observation $X(t)$ is multivariate, it typically entails a consumed item ID $\xi(t)$, a page id $p(t)$ and some metadata such as the timestamp t corresponding to the instant consumption started. The current model relies on a Recurrent Neural Network (RNN) to read through the sequence of past observations and predict the ID of the next item to be consumed by the user. A block diagram illustrates the architecture in Figure 5.

Valuable information for the current prediction is present in old observations thereby requiring our model to be LRD in order to improve the experience of users by providing more accurate recommendations. Such a feature is expected for a problem attempt-

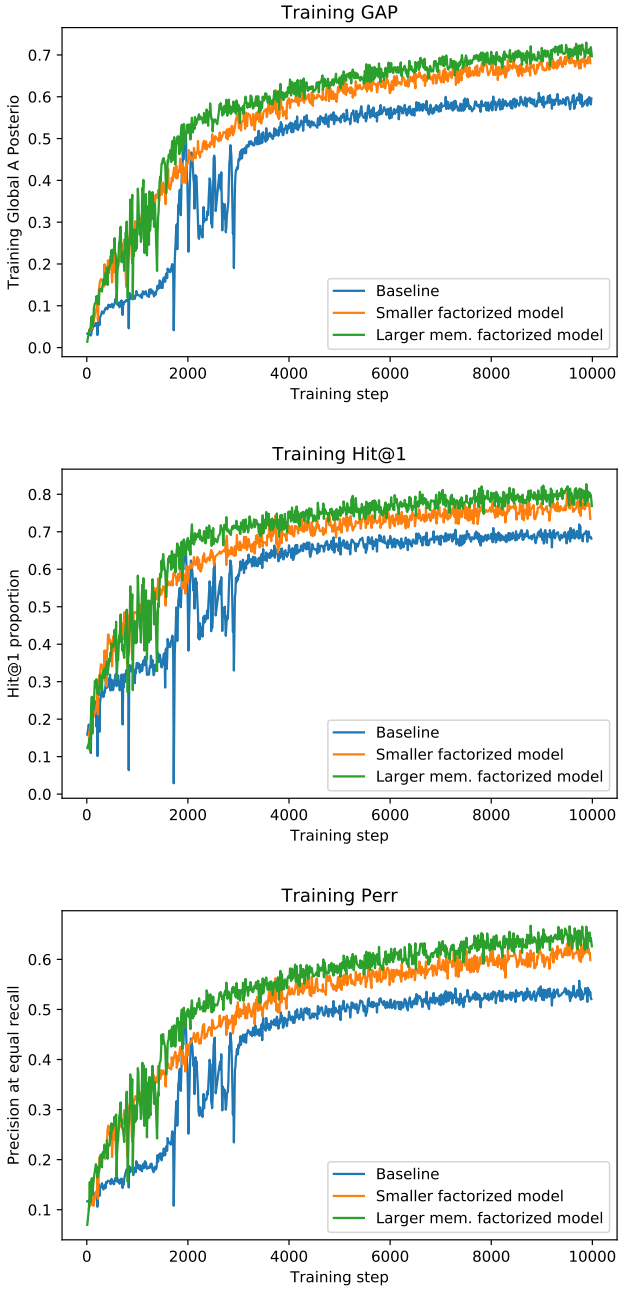

Figure 4: On the youtube 8m classification factorized architectures provide a substantial gain of classification accuracy as compared to their unstructured counterparts. We notice here that even a factorized architecture with fewer parameters than the baseline performs better.

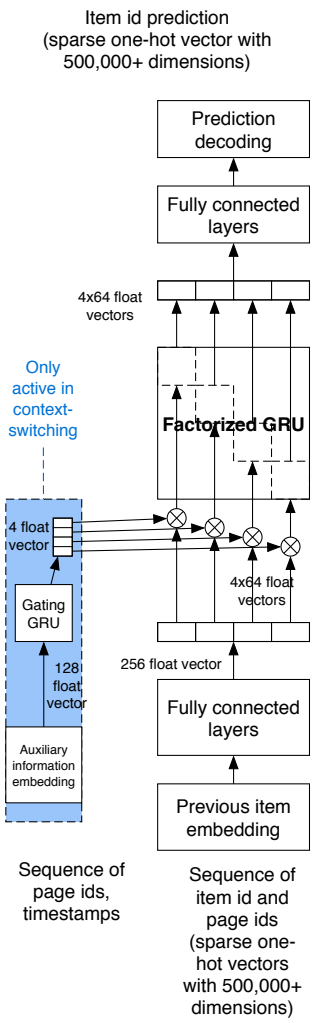

Figure 5: Architecture of the neural recommender we consider. We factorize the GRU at the core of model as in (14).

ing to predict the behavior of a human as one of most salient characteristics of our species is to keep memory of events through months, years and decades.

3.3.2 Experimental setting

The input fed into the recurrent network of interest consists of a timestamped collection (item id, page id, t). The sequences are truncated to 500 items. The item vocabulary comprises of 2M most popular items of the last 48 hours. We present here results obtained on a dataset where only about 700 000 pages are present that correspond to most popular pages. Only data from the last 7 days is used for training and only from the last 2 days for testing. The train/test split is $10/90\%$. The test set does not overlap with the train set and corresponds to the last temporal slice of the dataset. The neural network predicts, for a sample N of negatives, the probability that they are chosen as classically a negative sampling loss is employed in order to leverage observations belonging to a very large vocabulary [38]. Adagrad is used as an optimization procedure as other optimizers have lead to serious divergence issues. The loss being minimized is

$$\sum_{l \in \text{Labels}} w_l \times \text{CrossEntropy}(\text{SampledSoftmax}(\xi(t+1)))$$

Model	MAP@20
Baseline	0.117
Factorized model	0.122
Context switching model	0.126

Table 2: Performance of baseline, factorized and context switching architectures on the recommendation task. It is noteworthy that in this experiment the number of parameters of the fully connected layers located after the GRU at the core of network is higher after factorization.

where the SampledSoftmax [38] uses 20000 randomly sampled negatives and w_l is the weight of each label.

The Mean-average-precision-at-20 (MAP@20) is the main performance metric we monitor. As training time is critical in order to prevent model staleness, we compare the MAP@20 of competing architectures as they reach 5 million steps of optimization.

3.3.3 Context switching versus redundant memory: an ablation study

In the setting of item recommendations we choose to employ the context switching extension introduced in 2.2.3. As the results presented in Table 2 demonstrate factorizing the recurrent model at the core of the recommender is key to improve predictive performance on long sequences of browsing activity. Adding a context-switching mechanism to the factorization in order to account for the fundamental multimodality of browsing further improves predictive performance.

Conclusion

Our theoretical study showed that under reasonable assumptions RNNs such as GRUs and LSTMs with spectral radii leading to vanishing gradients also lead to short range dependency. Motivated by these propositions and theorems, we proposed an advantageous architectural modification to the networks to enable longer memory. Numerical experiments on a diverse set of tasks show that expanding the number of channels of the RNN's memory, while not increasing the memory or computational complexity, provided better results on LRD copy, classification and prediction tasks. Employing temporal dilations as a means to create provably LRD architectures for sequential learning is the topic of our current work.

References

[1] M. I. Jordan, "Serial order: A parallel distributed processing approach," *Advances in psychology*, vol. 121, pp. 471–495, 1997.

[2] D. Jurafsky, *Speech & language processing*. Pearson Education India, 2000.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase

representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[6] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[8] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[9] S. El Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Advances in neural information processing systems*, 1996, pp. 493–499.

[10] J. M. T. Thompson and H. B. Stewart, *Nonlinear dynamics and chaos*. John Wiley & Sons, 2002.

[11] S. H. Strogatz, *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Westview press, 2014.

[12] V. Pipiras and M. S. Taqqu, *Long-range dependence and self-similarity*. Cambridge University Press, 2017, vol. 45.

[13] P. J. Brockwell and R. A. Davis, *Time series: theory and methods*. Springer Science & Business Media, 2013.

[14] G. Adomian, *Nonlinear stochastic systems theory and applications to physics*. Springer Science & Business Media, 1988, vol. 46.

[15] P. Doukhan, G. Oppenheim, and M. Taqqu, *Theory and applications of long-range dependence*. Springer Science & Business Media, 2002.

[16] F. Belletti, E. Sparks, A. Bayen, and J. Gonzalez, "Random projection design for scalable implicit smoothing of randomly observed stochastic processes," in *Artificial Intelligence and Statistics*, 2017, pp. 700–708.

[17] K. Doya, "Bifurcations of recurrent neural networks in gradient descent learning," *IEEE Transactions on neural networks*, vol. 1, pp. 75–80, 1993.

[18] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[20] J. Collins, J. Sohl-Dickstein, and D. Sussillo, "Capacity and trainability in recurrent neural networks," *stat*, vol. 1050, p. 28, 2017.

[21] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[22] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[23] D. Guo, S. Shamai, and S. Verdú, "Additive non-gaussian noise channels: Mutual information and conditional mean estimation," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*. IEEE, 2005, pp. 719–723.

[24] S. Verdu, "Capacity region of gaussian cdma channels: The symbolsynchronous case," in *Proc. 24th Allerton Conf. Communication, Control and Computing*, 1986, pp. 1025–1034.

[25] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[26] O. Kuchaiev and B. Ginsburg, "Factorization tricks for lstm networks," *arXiv preprint arXiv:1703.10722*, 2017.

[27] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[28] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[29] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[30] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[31] Y. Dauphin, H. De Vries, J. Chung, and Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization (2015). arxiv preprint," *arXiv preprint arXiv:1502.04390*.

[32] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[33] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[35] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 495–503.

[36] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[37] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[38] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," *arXiv preprint arXiv:1412.2007*, 2014.