Long Short-Term Memory with Slower Information Decay

Hsiang-Yun Sherry Chien [1 2] **Javier S. Turek** [3] **Nicole M. Beckage** [3] **Vy A. Vo** [3] **Christopher Honey** [1]
Ted L. Willke [3]

Abstract

Learning to process long-range dependencies has been a challenge for recurrent neural networks. Despite improvements achieved by long short-term memory (LSTMs), its gating mechanism results in exponential decay of information, limiting their capacity of capturing long-range dependencies. In this work, we present a *power law forget gate*, which instead has a slower rate of information decay. We propose a power law-based LSTM (pLSTM) based on the LSTM but with a power law forget gate. We test empirically the pLSTM on the copy task, sentiment classification, and sequential MNIST, all with long-range dependency tasks. The pLSTM solves these tasks outperforming an LSTM, specially for long-range dependencies. Further, the pLSTM learns sparser and more robust representations.

1. Introduction

Recurrent neural networks (RNNs) are powerful models for capturing the structure of sequence data; they store information from the past in internal hidden states and have recurrent connections to allow these internal states to be combined with input information. These internal representations become "context" as described in Elman 1990. Long Short-Term Memory networks (LSTMs, Hochreiter & Schmidhuber 1997) have successfully improved RNNs' performance by adding gates to the recurrent unit to control the information flow, thus avoiding vanishing gradients and enabling processing of longer contexts.

Recently, (Mahto et al., 2021) shows that information stored in the memory cell in LSTMs decays exponentially, making it difficult for LSTMs to learn information beyond a few hundreds of timesteps. (Tallec & Ollivier, 2018) proposes the "chrono initialization" to initialize the forget gate bias in an LSTM based on the prior knowledge of maximum inter-dependency length between sequence elements, allowing for retaining information longer in memory. However, obtaining such prior knowledge is non-trivial and dependent on both the data and task at hand (Mahto et al., 2021). Gu et al. (2020) proposed a modified gating mechanism, the refine gate, by implementing an adjustment function to the original forget gate activation. An additional gating parameter is added for the network to learn how to adjust the activation curves according to different tasks. The refined gate also has a uniform initialization modified from Tallec & Ollivier (2018), which allows the network to capture a wide range of temporal dependencies. Their refine gate also decays at an exponential rate decay.

In this work, we propose a novel power law-based forget gate, that improves the ability of the LSTM model to learn long-range dependencies. This gating mechanism is designed to combat the exponential decay of information by enforcing the memory cell state to follow a power law information decay. Then, we enhance the LSTM network with our power law-based forget gate, which is called power law-gated LSTM (pLSTM). We test the pLSTM network empirically, showing its long-range dependency learning capability without prior knowledge, and obtaining more robust representations.

2. Information Decay in LSTM

Gated RNNs introduce a gating mechanism that controls the flow of information and overcomes vanishing gradients during backpropagation. A gate functions as a switch that controls the amount of information that is allowed to pass through it. LSTMs (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014) are examples of recurrent models that exploit gating mechanisms to support the retention of information for longer inputs. In particular, the LSTM defines an input gate i_t, a forget gate

[1]Department of Psychological and Brain Sciences, Johns Hopkins University, Maryland, USA [2]Work done while interning at Intel Labs. [3]Intel Labs, Hillsboro, Oregon, USA. Correspondence to: Hsiang-Yun Sherry Chien <hsiangyun.chien@gmail.com>, Javier S. Turek <javier.turek@intel.com>.

Proceedings of the LatinX in AI (LXAI) Research workshop at ICML 2021.

f_t, a cell state update gate $\tilde{c}_t$, and an output gate o_t:

$$\begin{aligned} f_t &= \sigma\left(U_f x_t + W_f h_{t-1} + b_f\right) \\ i_t &= \sigma\left(U_i x_t + W_i h_{t-1} + b_i\right) \\ o_t &= \sigma\left(U_o x_t + W_o h_{t-1} + b_o\right) \\ \tilde{c}_t &= \tanh\left(U_c x_t + W_c h_{t-1} + b_c\right) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh\left(c_t\right), \end{aligned} \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function, and $\odot$ stands for the Hadamard (element-wise) product. The hidden state h_t tracks the model dynamics and partially controls the activation of the gates, whereas the memory cell state c_t stores information.

Mahto et al. (2021) analyses LSTMs in a "free-input regime" case where there is no external input to the network after timestep t_0, i.e., $x_t = 0$ for $t > t_0$. Information leakage through the hidden state is also ignored, e.g. $W_c = 0$, $W_f = 0$, and $b_c = 0$. The cell state update in Equation 1 becomes

$$c_t = f_t \odot c_{t-1} = c_0 \odot e^{\log(f_0)(t-t_0)}, \tag{2}$$

where $f_0 = \sigma\left(b_f\right)$ and c_0 is the cell state at timestep t_0. Equation 2 shows that the cell state follows an exponential decay with a $\log\left(f_0\right)$ factor under such conditions. Consequently, the forget gate bias term, b_f, controls the rate of decay. A high bias value b_f would yield a slower decay rate. This fact was previously observed by Tallec & Ollivier (2018), who derived the "chrono initialization" method, which enables LSTMs to learn to capture information for longer sequences. A caveat of the chrono initialization method is that the maximum dependency range T_{max} *should be known a priori*.

3. A Power Law Gated LSTM

3.1. Slower Information Decay

The exponential decay due to the forget gate in Equation 2 poses a limitation to capturing information for long-term dependencies. Instead, a power law function of the form $(t - t_0 + 1)^{-p}$ can grant us a slower information decay rate. Therefore, we propose a forget gate based as follows:

$$f_t = \left(\frac{t - t_0 + 1}{t - t_0}\right)^{-p}, \tag{3}$$

where t_0 is a reference time point indicating the start of information decay. Equation 3 is a recursive definition of $(t - t_0 + 1)^{-p}$.

In general, when new information is to be captured by the recurrent network at timestep t, the forget gate f_t is expected to be 0. Namely, we need to control the reference time t_0 for this objective, setting it equal to t. Thus, we define a new variable k_t that can be updated by the network as needed to keep the reference time. We define the complete gating mechanism for the power law-based forget gate by

$$f_t = \left(\frac{t - k_t + 1}{t - k_t}\right)^{-p} \tag{4}$$

$$k_t = r_t \odot t + (1 - r_t) \odot k_{t-1} \tag{5}$$

$$r_t = \sigma\left(U_r x_t + W_r h_{t-1} + b_r\right) \tag{6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{7}$$

where a reset gate r_t controls when the reference time k_t should be updated based on the input and the recurrent information. U_r, W_r are its weight parameters, and b_r its bias parameters. Specifically, when the reset gate is activated $r_t \to 1$, the reference time k_t would be reset to the current time point t; otherwise, k_t is kept as the prior reference time k_{t-1} to allow information to decay. This recurrent definition of the variable k_t adds to the cell memory of an LSTM. Further, we propose to initialize it to $k_0 = 0$ (as $t \geq 1$).

Equation 4 has a recurrent coefficient that is smaller than 1 (due to the negative sign in the power), suggesting that any positive power $p > 0$ is acceptable. However, the power law decay is slower as p approaches 0, suggesting that long-range dependencies would be better captured with small values of p. Unlike its exponential counterpart, the power law recurrence Equation 4 depends both on the timestep t and the reference timestep k_t. We consider the power p as a learnable parameter. This allows each recurrent unit to learn the best rate of decay. Given that the range $(0, 1)$ is the most useful for the power p, we compute p as $p = \sigma(\hat{p})$, and allow $\hat{p} \in \mathbb{R}$ to be any real value. Typically, we initialize $\hat{p}$ such that p is uniformly distributed on the $(0, 1)$ range.

In sum, we define the power law-gated LSTM (pLSTM) cell based on the LSTM definition but with the forget gate f_t derived from Equations 4-7. We consider the input gate i_t to be a separate gate as in the LSTM. However, in practice it is possible to replace the input gate by $i_t = 1 - f_t$, which yields $\sim 3/4$ of total parameters. The argument is that keeping and removing previous information in the cell state c_t should be coupled (Van Der Westhuizen & Lasenby, 2018).

3.2. Backpropagating Through the Power Law Gate

The power law gate follows the same principles as the LSTM gates, thus avoiding vanishing gradients during backpropagation through time. Nevertheless, we derive the gradient of the forget gate with respect to the reference time in Equation 4 to investigate the effect when learning. The term $\partial f_t / \partial k_t$ holds complex functionality of the gate and its derivative is given by

$$\frac{\partial f_t}{\partial k_t} = -p \odot \frac{(t - k_t)^{p-1}}{(t - k_t + 1)^{p+1}}. \tag{8}$$

When the power $p \in (0, 1)$, the partial derivative $\partial f_t / \partial k_t \to \infty$. This numerical instability makes the backpropagation through time unstable. To overcome this, we add an ε term in Equation 4 such that $f_t = \left(\frac{t-k_t+1}{t-k_t+\varepsilon}\right)^{-p}$. ε should be set to a small value; in experiments reported here $\varepsilon = 0.001$.

4. Experimental Results

We tested the new power law forget gate incorporated into an LSTM on several tasks that require retaining information over long timescales. We compared the performance of the power law LSTM (pLSTM) to a vanilla LSTM. All experiments were implemented in Pytorch 1.7, used a learning rate of 0.001 and optimized with Adam ($\beta_1 = 0.9$, and $\beta_2 = 0.999$) unless otherwise specified.

4.1. Copy Task

We examine whether the power law forget gate in the pLSTM results in a slower rate of information decay. We compare the performance of LSTM, LSTM with chrono initialization (Tallec & Ollivier, 2018), and the pLSTM on the copy task (Hochreiter & Schmidhuber, 1997). The copy task was designed to examine the networks' ability to memorize information for an amount of time T. An input sequence consists $n = 10$ "target" elements uniformly drawn from $m = 8$ possible symbols: $a_0, \ldots, a_{m-1}$. Then, a "dummy" symbol a_m is repeated for T timesteps. At position $T+n$, a signal symbol a_{m+1} is presented followed by $n-1$ dummy symbols. The output sequence consists on $n+T$ dummy symbols, followed by the n "target" elements from the input sequence. A model must memorize the n target elements, and, after keeping these in memory for T time steps, output them in order after the signal symbol. In our experiments, we examined the models' ability to retain information over $T = 200$, 500 and 1000 timesteps. The training dataset is 100K sequences, and the validation dataset 10K sequences.

For all the experiments, we set batch size to 128 and hidden unit size to 128, following prior studies (Arjovsky et al., 2016; Tallec & Ollivier, 2018). Optimization was performed using RMSprop (Tieleman & Hinton, 2012) with a moving average parameter of 0.9. We measure accuracy as the proportion of successfully memorized elements out of the 10 target elements. The LSTM-chrono initializes the forget gate bias with $T_{max} = {}^{3T}/_{2}$, following Tallec & Ollivier (2018). Figure 1 shows validation accuracy over training iterations for $T = 1000$. Similar results were obtained for $T = 200$ and 500 (see supp. material). The pLSTM was able to learn the copy task in all conditions. It also converged faster than LSTM-chrono for $T = 200$ and 500. This suggests that the power law gate helps pLSTM to retain information over long-timescales, with no need for prior knowledge of task demands to initialize the network.

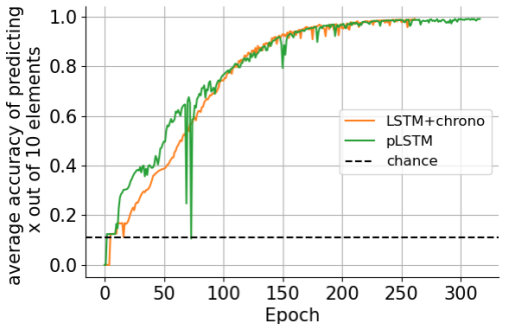

Figure 1. **Accuracy on copy task with** $T = 1000$, the pLSTM and LSTM-chrono both learned the task perfectly, with a similar rate of convergence. LSTM failed to converge (not shown).

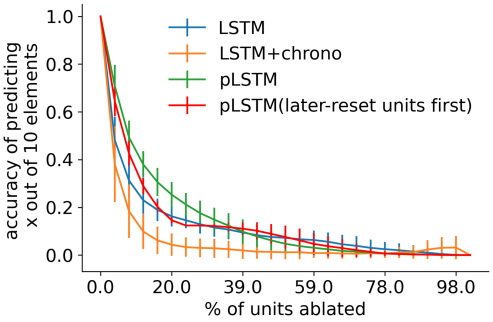

Figure 2. **Unit ablation:** Copy task accuracy ($T = 200$) when 5 units are ablated at a time ($\sim 4\%$ of total), for a total of 100 ablated units. The accuracy dropped more slowly in the pLSTM. This indicates that the pLSTM uses a sparse, local representation over relatively few units to solve the task. Moreover, when ablating the critical later-reset units first, the performance dropped faster.

Further, we tested model robustness by measuring the model performance after random unit ablation. We randomly sample 100 of the 128 units and ablated 5 units at a time. Figure 2 depicts the performance on the ablation study. When randomly ablating units in the network, the pLSTM (green) showed slower decay of task performance compared to LSTM (blue) and LSTM-chrono (orange) in $T = 200$ (See supp material for $T = 500$). The results showed that the pLSTM is more robust to unit ablation. Furthermore, LSTM and LSTM-chrono initialization lead to a distributed representation for solving the task and thus is fragile to ablation. Additionally, we examine whether ablating "later-reset units" first (red; units that reset only after signal symbol is seen) decrease the performance. Results show that these units are particularly important for the pLSTM performance than other units, suggesting that pLSTM learns a sparser and robust representation.

4.2. Sequential MNIST

Next, we examine the performance of the pLSTM on the sequential MNIST classification task and the permuted se-

Table 1. Test accuracy for pixel-by-pixel digit classification. Average results for successfully trained models over 3 random seeds.

METHOD	MNIST	PMNIST
LSTM 256 (OURS)	98.7%	91.3%
LSTM 512 (OURS)	98.6%	91.7%
PLSTM 256	**99.1%**	94.4%
PLSTM 512	**99.1%**	**95.6%**

quential MNIST (pMNIST) (Le et al., 2015). In this task, a digit image is converted to a 784-length sequence of pixels by reading it row after row from top to bottom. In pMNIST, the element order in the sequences is permuted. We partition the MNIST training set into 50K sequences for training and 10K for validation. The test set contains 10K sequences.

We evaluated the LSTM and pLSTM architectures with hidden sizes of 256 and 512 units. As noted by Gu et al. (2020), training LSTMs with more hidden units can be unstable and the model failed to converge with some seeds. We report, on three successful seeds, the average performance of the models in Table 1. The results show that the pLSTM captures long-range information better than the LSTM and other recurrent architectures. Unlike the LSTM, the pLSTM was able to successfully train every run.

4.3. Sentiment Classification

We also looked at the performance of the pLSTM on a sentiment classification task. In the IMDB dataset, each of 50k reviews are classified as either positive or negative (Maas et al., 2011), with 25k reviews in the training set. We pad or truncate the reviews to a fixed length of 400 words, and set aside a random 10% of the training set for validation. We train for 20 epochs and select the epoch with the lowest validation loss as our model. Following previous work, we used a 100d pretrained GLoVe embedding with a dictionary size of 25K (Rusch & Mishra, 2020). We test single layer LSTM, LSTM-chrono, and pLSTM, matching the number of parameters ($\sim$118K parameters; 128 units for the LSTM and LSTM-chrono, 154 units for pLSTM; dropout rate 0.2). The learning rate was optimized separately via grid search (LSTM, LSTM-chrono: 0.001; pLSTM: 0.0005). We report results on the test set averaged across 3 random seeds.

The pLSTM achieved a higher accuracy (.881) over the LSTM (.868) and LSTM-chrono (.870). Further analysis showed that the pLSTM selectively improves performance for longer sequence lengths (Figure 3 in supp. material).

5. Conclusion

In this work, we proposed a novel gating mechanism, the power law forget gate. We implemented it in an LSTM, called pLSTM. The new forget gate effectively results in the decay of cell state information via a power law function. This type of slow decay allows the network to learn long-range information over hundreds of timesteps. We empirically tested the performance of the pLSTM tasks requiring long-range dependencies. We showed that the pLSTM can effectively capture information over a wide range of distances. Further analyses showed that the pLSTM learned to perform the copy task using only a small subset of units, making the model more robust.

References

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *ICML*, 2016.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, 2014.

Elman, J. L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Gu, A., Gulcehre, C., Paine, T., Hoffman, M., and Pascanu, R. Improving the gating mechanism of recurrent neural networks. In *ICML*, 2020.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–80, 1997.

Le, Q. V., Jaitly, N., and Hinton, G. E. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *ACL*, 2011.

Mahto, S., Vo, V. A., Turek, J. S., and Huth, A. G. Multi-timescale representation learning in LSTM language models. In *ICLR*, 2021.

Rusch, T. K. and Mishra, S. Coupled oscillatory recurrent neural network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies. In *ICLR 2021*, 2020.

Tallec, C. and Ollivier, Y. Can recurrent neural networks warp time? In *ICLR*, 2018.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural networks for machine learning*, 2012.

Van Der Westhuizen, J. and Lasenby, J. The unreasonable effectiveness of the forget gate. *arXiv:1804.04849*, 2018.