Recurrent Neural Network Regularization

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

We present a simple regularization technique for Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units. Dropout, the most successful technique for regularizing neural networks, does not work well with RNNs and LSTMs. In this paper, we show how to correctly apply dropout to LSTMs, and show that it substantially reduces overfitting on a variety of tasks. These tasks include language modeling, speech recognition, image caption generation, and machine translation.

1 Introduction

The Recurrent Neural Network (RNN) is neural sequence model that achieves state of the art performance on important tasks that include language modeling [1], speech recognition [2], and machine translation [3]. It is known that successful applications of neural networks require good regularization. Unfortunately, dropout [4], the most powerful regularization method for feedforward neural networks, does not work well with RNNs. As a result, practical applications of RNNs often use models that are too small because large RNNs tend to overfit. Existing regularization methods give relatively small improvements for RNNs [5]. In this work, we show that dropout, when correctly used, greatly reduces overfitting in LSTMs, and evaluate it on three different problems.

The code for this work will be available by the time of the publication.

2 Related work

Dropout [4] is a recently introduced regularization method that has been very successful with feed-forward neural networks. While much work has extended dropout in various ways [6, 7], there has been relatively little research in applying it to RNNs. The only paper on this topic is by Bayer et al. [8], who focuses on "marginalized dropout" [6], a noiseless deterministic approximation to standard dropout. Bayer et al. [8] claim that conventional dropout does not work well with RNNs because the recurrence amplifies noise, which in turn hurts learning. In this work, we show that this problem can be fixed by applying dropout to a certain subset of the RNNs' connections. As a result, RNNs can now also benefit from dropout.

Independently of our work, Pham et al. [9] developed the very same RNN regularization method and applied it to handwriting recognition. We rediscovered this method and demonstrated strong empirical results over a wide range of problems. Other work that applied dropout to LSTMs is Pachitariu and Sahani [10].

There have been a number of architectural variants of the RNN that perform better on problems with long term dependencies [11, 12, 13, 14, 15, 16]. In this work, we show how to correctly apply dropout to LSTMs, the most commonly-used RNN variant; this way of applying dropout is likely to work well with other RNN architectures as well.

In this paper, we consider the following tasks: language modeling, speech recognition, and machine translation. Language modeling is the first task where RNNs have achieved substantial success [17, 18, 19]. RNNs have also been successfully used for speech recognition [20, 2] and have recently been applied to machine translation, where they are used for language modeling, re-ranking, or phrase modeling [21, 3, 13, 22, 23].

3 Regularizing RNNs with LSTM cells

In this section we describe the deep LSTM (Section 3.1). Next, we show how to regularize them (Section 3.2), and explain why our regularization scheme works.

We let subscripts denote timesteps and superscripts denote layers. All our states are n-dimensional. Let $h_t^l \in \mathbb{R}^n$ be a hidden state in layer l in timestep t. Moreover, let $T_{n,m} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be an affine transform ($Wx + b$ for some W and b). Let $\odot$ be element-wise multiplication and let h_t^0 be an input word vector at timestep k. We use the activations h_t^L to predict y_t, since L is the number of layers in our deep LSTM.

3.1 Long-short term memory units

The RNN dynamics can be described using deterministic transitions from previous to current hidden states. The deterministic state transition is a function

$$\text{RNN} : h_t^{l-1}, h_{t-1}^l \rightarrow h_t^l$$

For classical RNNs, this function is given by

$$h_t^l = f(T_{n,n}h_t^{l-1} + T_{n,n}h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

The LSTM has complicated dynamics that allow it to easily "memorize" information for an extended number of timesteps. The "long term" memory is stored in a vector of *memory cells* $c_t^l \in \mathbb{R}^n$. Although many LSTM architectures that differ in their connectivity structure and activation functions, all LSTM architectures have explicit memory cells for storing information for long periods of time. The LSTM can decide to overwrite the memory cell, retrieve it, or keep it for the next time step. The LSTM architecture used in our experiments is given by the following equations [2]:

$$\begin{aligned}
&\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l \\
&\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \\
&c_t^l = f \odot c_{t-1}^l + i \odot g \\
&h_t^l = o \odot \tanh(c_t^l)
\end{aligned}$$

In these equations, sigm and $\tanh$ are applied element-wise. Figure 1 illustrates the LSTM equations.

3.2 Regularization with Dropout

The main contribution of this paper is a recipe for applying dropout to LSTMs in a way that successfully reduces overfitting. The main idea is to apply the dropout operator only to the non-recurrent connections (Figure 2). The following equation describes it more precisely, where $\mathbf{D}$ is the dropout operator that sets a random subset of its argument to zero:

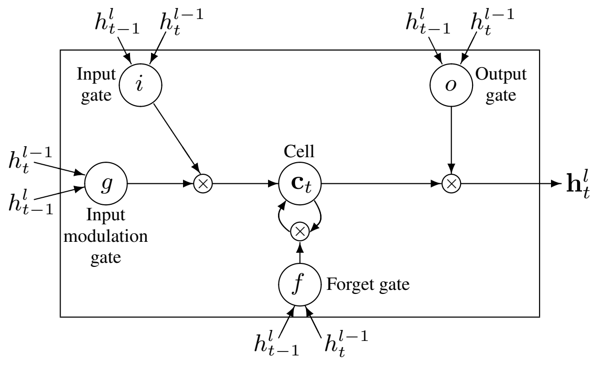

Figure 1: A graphical representation of LSTM memory cells used in this paper (there are minor differences in comparison to Graves [5]).

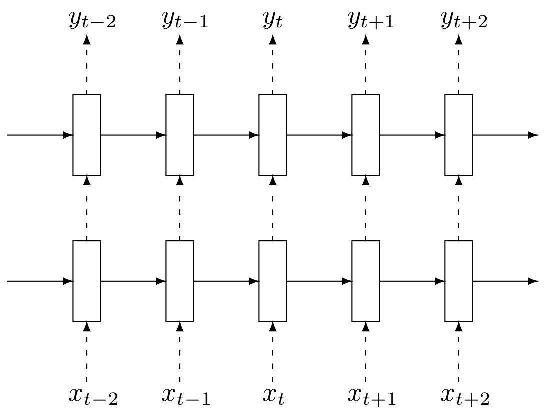

Figure 2: Regularized multilayer RNN. The dashed arrows indicate connections where dropout is applied, and the solid lines indicate connections where dropout is not applied.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} \mathbf{D}(h_t^{l-1}) \\ h_{t-1}^l \end{pmatrix}$$
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$
$$h_t^l = o \odot \tanh(c_t^l)$$

Our method works as follows. The dropout operator corrupts the information carried by the units, forcing them to perform their intermediate computations more robustly. At the same time, we do not want to erase all the information from the units. It is especially important that the units remember events that occurred many timesteps in the past. Figure 3 shows how information could flow from an event that occurred at timestep $t-2$ to the prediction in timestep $t+2$ in our implementation of dropout. We can see that the information is corrupted by the dropout operator exactly $L+1$ times, and this number is independent of the number of timesteps traversed by the information. Standard dropout perturbs the recurrent connections, which makes it difficult for the LSTM to learn to store

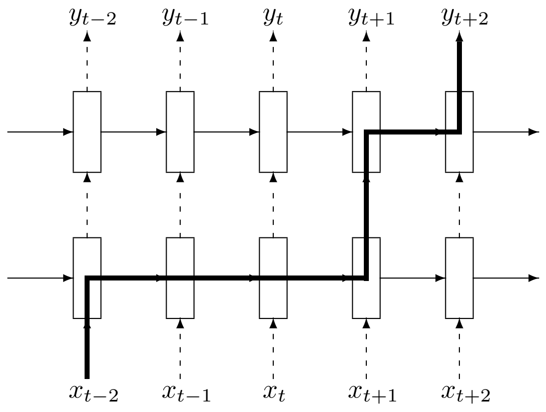

Figure 3: The thick line shows a typical path of information flow in the LSTM. The information is affected by dropout $L + 1$ times, where L is depth of network.

---

the meaning of life is that only if an end would be of the whole supplier. widespread rules are regarded as the companies of refuses to deliver. in balance of the nation 's information and loan growth associated with the carrier thrifts are in the process of slowing the seed and commercial paper.

---

the meaning of life is nearly in the first several months before the government was addressing such a move as president and chief executive of the nation past from a national commitment to curb grounds. meanwhile the government invests overcapacity that criticism and in the outer reversal of small-town america.

Figure 4: Some interesting samples drawn from a large regularized model conditioned on "The meaning of life is". We have removed "unk", "N", "$" from the set of permissible words.

information for long periods of time. By not using dropout on the recurrent connections, the LSTM can benefit from dropout regularization without sacrificing its valuable memorization ability.

4 Experiments

We present results in three domains: language modeling (Section 4.1), speech recognition (Section 4.2), machine translation (Section 4.3), and image caption generation (Section 4.4).

4.1 Language modeling

We conducted word-level prediction experiments on the Penn Tree Bank (PTB) dataset [24], which consists of 929k training words, 73k validation words, and 82k test words. It has 10k words in its vocabulary. We downloaded it from Tomas Mikolov's webpage[1]. We trained regularized LSTMs of two sizes; these are denoted the medium LSTM and large LSTM. Both LSTMs have two layers and are unrolled for 35 steps. We initialize the hidden states to zero. We then use the final hidden states of the current minibatch as the initial hidden state of the subsequent minibatch (successive minibatches sequentially traverse the training set). The size of each minibatch is 20.

The medium LSTM has 650 units per layer and its parameters are initialized uniformly in $[-0.05, 0.05]$. As described earlier, we apply 50% dropout on the non-recurrent connections. We train the LSTM for 39 epochs with a learning rate of 1, and after 6 epochs we decrease it by a factor

[1] `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`

Model	Validation set	Test set
A single model		
Pascanu et al. [19]		107.5
Cheng et al. [25]		100.0
non-regularized LSTM	120.7	114.5
Medium regularized LSTM	86.2	82.7
Large regularized LSTM	82.2	**78.4**
Model averaging		
Mikolov [1]		83.5
Cheng et al. [25]		80.6
2 non-regularized LSTMs	100.4	96.1
5 non-regularized LSTMs	87.9	84.1
10 non-regularized LSTMs	83.5	80.0
2 medium regularized LSTMs	80.6	77.0
5 medium regularized LSTMs	76.7	73.3
10 medium regularized LSTMs	75.2	72.0
2 large regularized LSTMs	76.9	73.6
10 large regularized LSTMs	72.8	69.5
38 large regularized LSTMs	71.9	**68.7**
Model averaging with dynamic RNNs and n-gram models		
Mikolov and Zweig [26]		72.9

Table 1: Word-level perplexity on the Penn Tree Bank dataset.

of 1.2 after each epoch. We clip the norm of the gradients (normalized by minibatch size) at 5. Training this network takes about half a day on an NVIDIA K20 GPU.

The large LSTM has 1500 units per layer and its parameters are initialized uniformly in $[-0.04, 0.04]$. We apply 65% dropout on the non-recurrent connections. We train the model for total 55 epochs. We start with learning rate of 1; after 14 epochs we start to reduce the learning rate by a factor of 1.15 after each epoch. We clip the norm of the gradients (normalized by minibatch size) at 10 [17]. Training this network takes an entire day on an NVIDIA K20 GPU.

For comparison, we trained a non-regularized network. We optimized its parameters to get the best validation performance. The lack of regularization effectively constrains size of the network, forcing us to use small network because larger networks overfit. Our best performing non-regularized LSTM has two hidden layers with 200 units per layer, and its weights are initialized uniformly in $[-0.1, 0.1]$. We train it for 4 epochs with a learning rate of 1 and then we decrease the learning rate by a factor of 2 after each epoch, for a total of 13 training epochs. The size of each minibatch is 20, and we unroll the network for 20 steps. Training this network takes 2-3 hours on an NVIDIA K20 GPU.

Table 1 compares previous results with our LSTMs, and Figure 4 shows samples drawn from a single large regularized LSTM.

4.2 Speech recognition

Deep Neural Networks have been used for acoustic modeling for over half a century (see Bourlard and Morgan [27] for a good review). Acoustic modeling is a key component in mapping acoustic signals to sequences of words, as it models $p(s_t|X)$ where s_t is the phonetic state at time t and X is the acoustic observation. Recent work has shown that LSTMs can achieve excellent performance on acoustic modeling [28], yet relatively small LSTMs (in terms of the number of their parameters) can easily overfit the training set. A useful metric for measuring the performance of acoustic models is frame accuracy, which is measured at each s_t for all timesteps t. Generally, this metric correlates with the actual metric of interest, the Word Error Rate (WER). Since computing the WER involves using a language model and tuning the decoding parameters for every change in the acoustic model, we decided to focus on frame accuracy in these experiments. Table 2 shows that dropout improves the frame accuracy of the LSTM. Not surprisingly, the training frame accuracy drops due to the

Model	Training set	Validation set
Non-regularized LSTM	71.6	68.9
Regularized LSTM	69.4	**70.5**

Table 2: Frame-level accuracy on the Icelandic Speech Dataset. The training set has 93k utterances.

Model	Test perplexity	Test BLEU score
Non-regularized LSTM	5.8	25.9
Regularized LSTM	5.0	29.03
LIUM system		33.30

Table 3: Results on the English to French translation task.

noise added during training, but as is often the case with dropout, this yields models that generalize better to unseen data. Note that the test set is easier than the training set, as its accuracy is higher. We report the performance of an LSTM on an internal Google Icelandic Speech dataset, which is relatively small (93k utterances), so overfitting is a great concern.

4.3 Machine translation

We formulate a machine translation problem as a language modelling task, where an LSTM is trained to assign high probability to a correct translation of a source sentence. Thus, the LSTM is trained on concatenations of source sentences and their translations [29] (see also Cho et al. [13]). We compute a translation by approximating the most probable sequence of words using a simple beam search with a beam of size 12. We ran an LSTM on the WMT'14 English to French dataset, on the "selected" subset from [30] which has 340M French words and 304M English words. Our LSTM has 4 hidden layers, and both its layers and word embeddings have 1000 units. Its English vocabulary has 160,000 words and its French vocabulary has 80,000 words. The optimal dropout probability was 0.2. Table 3 shows the performance of an LSTM trained with and without dropout. While our LSTM does not beat the phrase-based LIUM SMT system [31], our results show that dropout improves the translation performance of the LSTM.

4.4 Image Caption Generation

Model	Test perplexity	Test BLEU score
Non-regularized model	8.47	23.5
Regularized model	7.99	24.3
10 non-regularized models	7.5	24.4

Table 4: Results on the image caption generation task.

We applied the dropout variant to the image caption generation model of Vinyals et al. [32]. The image caption generation is similar to the sequence-to-sequence model of Sutskever et al. [29], but where the input image is mapped onto a vector with a highly-accurate pre-trained convolutional neural network [33], which is converted into a caption with a single-layer LSTM (see Vinyals et al. [32] for the details on the architecture). We test our dropout scheme on LSTM as the convolutional neural network is not trained on the image caption dataset because it is not large (MSCOCO [34]).

Our results are summarized in the following Table 4. In brief, dropout helps relative to not using dropout, but using an ensemble eliminates the gains attained by dropout. Thus, in this setting, the main effect of dropout is to produce a single model that is as good as an ensemble, which is a reasonable improvement given the simplicity of the technique.

5 Conclusion

We presented a simple way of applying dropout to LSTMs that results in large performance increases on several problems in different domains. Our work makes dropout useful for RNNs, and our results suggest that our implementation of dropout could improve performance on a wide variety of applications.

References

[1] Tomáš Mikolov. *Statistical language models based on neural networks*. PhD thesis, Ph. D. thesis, Brno University of Technology, 2012.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[3] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *EMNLP*, 2013.

[4] Nitish Srivastava. *Improving neural networks with dropout*. PhD thesis, University of Toronto, 2013.

[5] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[6] Sida Wang and Christopher Manning. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126, 2013.

[7] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[8] Justin Bayer, Christian Osendorfer, Nutan Chen, Sebastian Urban, and Patrick van der Smagt. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*, 2013.

[9] Vu Pham, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. *arXiv preprint arXiv:1312.4569*, 2013.

[10] Marius Pachitariu and Maneesh Sahani. Regularization and nonlinearities for neural language models: when are they needed? *arXiv preprint arXiv:1301.5650*, 2013.

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

[12] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

[13] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[15] Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A clockwork rnn. *arXiv preprint arXiv:1402.3511*, 2014.

[16] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, 2012.

[17] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[18] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE, 2011.

[19] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[20] Tony Robinson, Mike Hochberg, and Steve Renals. The use of recurrent neural networks in continuous speech recognition. In *Automatic speech and speaker recognition*, pages 233–258. Springer, 1996.

[21] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. Fast and robust neural network joint models for statistical machine translation. In *ACL*, 2014.

[22] Y Chow, M Dunham, O Kimball, M Krasner, G Kubala, J Makhoul, P Price, S Roucos, and R Schwartz. Byblos: The bbn continuous speech recognition system. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, volume 12, pages 89–92. IEEE, 1987.

[23] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[24] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[25] Wei-Chen Cheng, Stanley Kok, Hoai Vu Pham, Hai Leong Chieu, and Kian Ming A Chai. Language modeling with sum-product networks.

[26] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *SLT*, pages 234–239, 2012.

[27] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, 1993.

[28] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao. Sequence discriminative distributed training of long short-term memory recurrent neural networks. In *Interspeech*, 2014.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[30] Holger Schwenk. University le mans, 2014. `http://www-lium.univ-lemans.fr/~schwenk/cslm_joint/paper`.

[31] Holger Schwenk, Patrik Lambert, Loïc Barrault, Christophe Servan, Haithem Afli, Sadaf Abdul-Rauf, and Kashif Shah. Lium's smt machine translation systems for wmt 2011. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 464–469. Association for Computational Linguistics, 2011.

[32] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014.

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. *arXiv preprint arXiv:1405.0312*, 2014.