# NOISIN: UNBIASED REGULARIZATION FOR RECURRENT NEURAL NETWORKS

**Adji B. Dieng***
Columbia University

**Rajesh Ranganath**
New York University

**Jaan Altosaar**
Princeton University

**David M. Blei**
Columbia University

## ABSTRACT

Recurrent neural networks (RNNS) are powerful models of sequential data. They have been successfully used in domains such as text and speech. However, RNNS are susceptible to overfitting; regularization is important. In this paper we develop NOISIN, a new method for regularizing RNNS. NOISIN injects random noise into the hidden states of the RNN and then maximizes the corresponding marginal likelihood of the data. We show how NOISIN applies to any RNN and we study many different types of noise. NOISIN is unbiased—it preserves the underlying RNN on average. On language modeling benchmarks, NOISIN improves over dropout by as much as $12.2\%$ on the Penn Treebank and $9.4\%$ on the Wikitext-2 dataset.

## 1 OVERVIEW

Recurrent neural networks (RNNS) are powerful models of sequential data (Robinson & Fallside, 1987; Werbos, 1988; Williams, 1989; Elman, 1990; Pearlmutter, 1995). RNNS have achieved state-of-the-art results on many tasks, including language modeling (Mikolov & Zweig, 2012; Yang et al., 2017), text generation (Graves, 2013), image generation (Gregor et al., 2015), speech recognition (Graves et al., 2013; Chiu et al., 2017), and machine translation (Sutskever et al., 2014; Wu et al., 2016).

However RNNS are very flexible and they overfit; regularization is crucial. Many techniques have been developed to address overfitting in RNNS. Some are based on normalization (Ioffe & Szegedy, 2015; Ba et al., 2016; Cooijmans et al., 2016) and others—including what we study in this paper—involve auxiliary noise variables. The most successful noise-based regularizer for neural networks is dropout (Srivastava et al., 2014; Wager et al., 2013; Noh et al., 2017; Zaremba et al., 2014; Gal & Ghahramani, 2016). Still other noise-based regularization prunes the network by dropping updates to the hidden units of the RNN (Krueger et al., 2016; Semeniuta et al., 2016). More recently Merity et al. (2017) extended these techniques.

Involving noise variables in RNNS has been used in contexts other than regularization. For example Jim et al. (1996) analyze the impact of noise on convergence and long-term dependencies and Bayer & Osendorfer (2014); Chung et al. (2015); Fraccaro et al. (2016); Goyal et al. (2017) use auxiliary latent variables to capture the high variability of complex sequential data such as music and audio.

In this paper, we develop NOISIN, an effective new way to regularize an RNN. The idea is to inject random noise into its transition function and then to fit its parameters to maximize the corresponding marginal likelihood of the observations. We can easily apply NOISIN to any flavor of RNN and we can use many types of noise.

NOISIN regularizes the RNN by smoothing its loss, averaging over local neighborhoods of the transition function. Further, NOISIN requires that the noise-injected transition function be *unbiased*. This means that, on average, it preserves the transition function of the original RNN.

*correspondence to Adji B. Dieng: abd2141@columbia.edu

We examine NOISIN built from the LSTM and the LSTM with dropout, which we call the dropout-LSTM, and we explore unbiased noise from several types of distributions. We study performance with two large-scale language modeling tasks. NOISIN improves over the LSTM by as much as $37.3\%$ on the Penn Treebank and $39.0\%$ on the Wikitext-2 dataset; it improves over dropout-LSTM by as much as $12.2\%$ on the Penn Treebank and $9.4\%$ on Wikitext-2.

## 2 NOISIN

NOISIN is built from noise-injected RNNs. We define noise-injected RNNs as any RNN following the generative process

$$\epsilon_{1:T} \sim \varphi(\cdot; \mu, \gamma)\ ;\ \mathbf{z}_t = g_W(\mathbf{x}_{t-1}, \mathbf{z}_{t-1}, \epsilon_t) \text{ and } p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}) = p(\mathbf{x}_t \mid \mathbf{z}_t), \tag{1}$$

where the likelihood $p(\mathbf{x}_t \mid \mathbf{z}_t)$ is an exponential family,

$$p(\mathbf{x}_t|\mathbf{z}_t) = \nu(\mathbf{x}_t) \exp\left((V^\top \mathbf{z}_t)^\top \mathbf{x}_t - A(V^\top \mathbf{z}_t)\right), \tag{2}$$

The noise variables $\epsilon_{1:T}$ are drawn from a distribution $\varphi(\cdot; \mu, \gamma)$ with mean $\mu$ and scale $\gamma$. For example, $\varphi(\cdot; \mu, \gamma)$ can be a zero-mean Gaussian with variance $\gamma^2$. We will study many types of noise distributions.

The noisy hidden state $\mathbf{z}_t$ is a parametric function $g_W$ of the previous observation $\mathbf{x}_{t-1}$, the previous noisy hidden state $\mathbf{z}_{t-1}$, and the noise $\epsilon_t$. Therefore conditional on the noise $\epsilon_{1:T}$, the transition function $g_W$ defines a recurrence relation on $\mathbf{z}_{1:T}$. The function $g_W$ determines the noise-injected RNN. In this paper, we propose functions $g_W$ that meet the criterion described below.

**Unbiased noise injection.** Injecting noise at each time step limits the amount of information carried by hidden states. In limiting their capacity, noise injection is regularization.

Let $\mathbf{z}_t(\epsilon_{1:t})$ denote the unrolled recurrence at time $t$; it is a random variable via the noise $\epsilon_{1:t}$. Under unbiasedness, the transition function $g_W$ must satisfy the relationship

$$\mathbb{E}_{p(\mathbf{z}_t(\epsilon_{1:t}) \mid \mathbf{z}_{t-1})}\left[\mathbf{z}_t(\epsilon_{1:t})\right] = f_W(\mathbf{x}_{t-1}, \mathbf{z}_{t-1}) \tag{3}$$

where $f_W$ is the transition function of the underlying RNN.

What unbiasedness means is that the noise should be injected in such a way that driving the noise to zero leads to the original RNN. Two possible choices for $g_W$ that meet this condition when $\epsilon_t$ has zero mean are the following

$$g_W(\mathbf{x}_{t-1}, \mathbf{z}_{t-1}, \epsilon_t) = f_W(\mathbf{x}_{t-1}, \mathbf{z}_{t-1}) + \epsilon_t \tag{4}$$

$$g_W(\mathbf{x}_{t-1}, \mathbf{z}_{t-1}, \epsilon_t) = f_W(\mathbf{x}_{t-1}, \mathbf{z}_{t-1}) \odot (1 + \epsilon_t). \tag{5}$$

These choices of $g_W$ correspond to additive noise and multiplicative noise respectively. Note $f_W$ can be any RNN including the RNN with dropout or the stochastic RNNs (Bayer & Osendorfer, 2014; Chung et al., 2015; Fraccaro et al., 2016; Goyal et al., 2017). For example to implement unbiased noise injection with multiplicative noise for the LSTM the only change from the original LSTM is to replace its hidden state with

$$\mathbf{z}_t = o_t \odot \tanh(\mathbf{c}_t) \odot (1 + \boldsymbol{\epsilon}_t).$$

where $o_t$ and $\mathbf{c}_t$ are the output gate and the cell state of the Long-Short Term Memory (LSTM) respectively. Such noise-injected hidden states can be stacked to build a multi-layered noise-injected LSTM that meet the unbiasedness condition.

**Objective.** NOISIN minimizes the expected negative log-likelihood under the injected noise,

$$\mathcal{L} = E_{p(\boldsymbol{\epsilon}_{1:T})}\left[\log p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}(\boldsymbol{\epsilon}_{1:T}))\right] = \sum_{t=1}^{T} E_{p(\epsilon_{1:t})}\Big[\log p(\mathbf{x}_t|\mathbf{z}_t(\epsilon_{1:t}))\Big] \tag{6}$$

Notice this objective is a Jensen bound on the marginal log-likelihood of the data,

$$\mathcal{L} \leq \log E_{p(\epsilon_{1:T})}\left[p(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}(\boldsymbol{\epsilon}_{1:T}))\right] = \log p(\mathbf{x}_{1:T}).$$

The expectations in the objective of Equation 6 are intractable due to the nonlinearities in the model and the form of the noise distribution. We approximate the objective using Monte Carlo;

$$\widehat{\mathcal{L}} = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T} \left[ \log p(\mathbf{x}_t | \mathbf{z}_t(\epsilon_{1:t}^{(k)})) \right].$$

When using one sample ($K = 1$), the training procedure is just as easy as for the underlying RNN. The loss becomes

$$\widetilde{\mathcal{L}} = -\sum_{t=1}^{T} \left[ \log \nu(\mathbf{x}_t) + (V^\top \mathbf{z}_t)^\top \mathbf{x}_t - A(V^\top \mathbf{z}_t) \right]. \tag{7}$$

(This expression uses the exponential family likelihood.) Algorithm 1 summarizes the procedure for multiplicative noise. The only change from traditional RNN training is when updating the hidden state in lines 4 and 5.

**Controling the noise level.** NOISIN is amenable to any RNN and any noise distribution. As with all regularization techniques, NOISIN comes with a free parameter that determines the amount of regularization: the spread $\gamma$ of the noise. Certain noise distributions have bounded variance; for example the Bernoulli and the Beta distributions. This limits the amount of regularization one can afford. To circumvent this, we rescale the noise to have unbounded variance. It is the scaled noise that is used in NOISIN.

**Connection to ensemble methods.** NOISIN can be interpreted as an ensemble method. The objective in Equation 6 corresponds to averaging the predictions of infinitely many RNNs at each time step in the sequence. This is known as an ensemble method and has a regularization effect (Poggio et al., 2002). However ensemble methods are costly as they require training all the sub-models in the ensemble. With NOISIN, at each time step in the sequence, one of the infinitely many RNNs is trained and because of parameter sharing, the RNN being trained at the next time step will use better settings of the weights. This makes training the whole model efficient.

**Connection to empirical Bayes.** Consider a noise-injected RNN. Its joint distribution is

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t | \mathbf{z}_t; V) p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}_{t-1}; W)$$

Here $p(\mathbf{x}_t|\mathbf{z}_t; V)$ denotes the likelihood and $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t-1}; W)$ is the prior over the noisy hidden states; it is parameterized by the weights $W$. From the perspective of Bayesian inference this is an unknown prior. When we optimize the objective in Equation 6, we are learning the weights $W$. This is equivalent to learning the prior over the noisy hidden states and is known as empirical Bayes (Robbins, 1964). It consists in getting point estimates of prior parameters in a hierarchical model and using those point estimates to define the prior.

## 3 EMPIRICAL STUDY

We presented NOISIN, a method that relies on unbiased noise injection to regularize any RNN. NOISIN is simple and can be integrated with any existing RNN model. In this section, we focus on applying NOISIN to the LSTM and the dropout-LSTM. We use language modeling as a testbed. Regularization is crucial in language modeling because the input and prediction matrices—which are typically high-dimensional—scale linearly with the size of the vocabulary.

We used NOISIN under two noise regimes: additive noise and multiplicative noise. We found that additive noise uniformly performs worse than multiplicative noise for the LSTM. We therefore report results only on multiplicative noise.

We used NOISIN with several noise distributions: Gaussian, Logistic, Laplace, Gamma, Bernoulli, Gumbel, Beta, and $\chi$-Square. We found that overall the only property that matters with these distributions is the variance. The variance determines the amount of regularization for NOISIN.

We also found that these distributions, when used with NOISIN on the LSTM perform better than the dropout LSTM on the Penn Treebank. Another interesting finding is that NOISIN when applied to the dropout-LSTM performs better than the original dropout-LSTM.

**Table 1:** NOISIN improves the performance of the LSTM and the dropout-LSTM by as much as $12\%$ on the Penn Treebank dataset. This table shows word-level perplexity scores on the medium and large settings for both the validation (or dev) and the test set.

| | Medium | | | Large | | | | Medium | | | Large | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $\gamma$ | Dev | Test | $\gamma$ | Dev | Test | Method | $\gamma$ | Dev | Test | $\gamma$ | Dev | Test |
| None | $--$ | 115 | 109 | $--$ | 123 | 123 | Dropout (D) | $--$ | 80.2 | 77.0 | $--$ | 78.6 | 75.3 |
| Gaussian | 1.10 | 76.2 | 71.8 | 1.37 | 73.2 | 69.1 | D + Gaussian | 0.53 | 73.4 | 70.4 | 0.92 | **70.0** | **66.1** |
| Logistic | 1.06 | 76.4 | 72.3 | 1.39 | 73.6 | 69.3 | D + Logistic | 0.53 | 73.0 | 69.9 | 0.84 | 69.8 | 66.4 |
| Laplace | 1.06 | 76.6 | 72.4 | 1.39 | 73.7 | 69.4 | D + Laplace | 0.53 | 73.1 | 70.0 | 0.92 | 69.9 | 66.6 |
| Gamma | 1.06 | 78.2 | 74.5 | 1.39 | 73.6 | 69.5 | D + Gamma | 0.38 | 73.5 | 70.3 | 0.92 | 71.1 | 68.2 |
| Bernoulli | 0.41 | **75.7** | **71.4** | 0.33 | **72.8** | **68.3** | D + Bernoulli | 0.80 | 73.3 | 70.1 | 0.50 | **70.0** | **66.1** |
| Gumbel | 1.06 | 76.2 | 72.7 | 1.39 | 73.5 | 69.5 | D + Gumbel | 0.46 | 74.5 | 71.2 | 0.92 | 70.2 | 67.1 |
| Beta | 1.07 | 76.0 | 71.4 | 1.50 | 74.4 | 70.2 | D + Beta | 0.20 | **73.0** | **69.2** | 0.70 | 70.0 | 66.2 |
| Chi | 1.50 | 84.5 | 80.7 | 1.20 | 79.2 | 75.5 | D + Chi | 0.29 | 76.1 | 72.8 | 0.82 | 73.0 | 70.0 |

**Table 2:** NOISIN improves the performance of the LSTM and the dropout-LSTM by as much as $9\%$ on the Wikitext-2 dataset. This table shows word-level perplexity scores on the medium and large settings for both the validation (or dev) and the test set. D is short for dropout. $D +$ Distribution refers to NOISIN applied to the dropout-LSTM with the specified distribution.

| | Medium | | | Large | | | | Medium | | | Large | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | $\gamma$ | Dev | Test | $\gamma$ | Dev | Test | Method | $\gamma$ | Dev | Test | $\gamma$ | Dev | Test |
| None | $--$ | 141 | 136 | $--$ | 176 | 140 | Dropout (D) | $--$ | 88.7 | 84.8 | $--$ | 95.0 | 91.0 |
| Gaussian | 1.00 | 92.7 | 87.8 | 1.37 | 87.7 | 83.4 | D + Gaussian | 0.50 | 86.3 | 82.3 | 0.69 | 81.4 | 77.7 |
| Logistic | 1.00 | 93.2 | 88.4 | 1.28 | 88.1 | 83.5 | D + Logistic | 0.40 | 86.4 | 82.5 | 0.77 | 81.6 | 78.1 |
| Laplace | 1.00 | 95.3 | 89.8 | 1.28 | 88.0 | 83.4 | D + Laplace | 0.40 | **85.6** | **82.1** | 0.61 | 83.2 | 79.1 |
| Gamma | 0.72 | 97.6 | 92.9 | 1.39 | 89.2 | 84.5 | D + Gamma | 0.30 | 86.5 | 82.4 | 0.61 | 85.5 | 81.3 |
| Bernoulli | 0.54 | 91.2 | 86.6 | 0.41 | 86.9 | 83.0 | D + Bernoulli | 0.50 | 100.6 | 94.4 | 0.64 | **80.8** | **76.8** |
| Gumbel | 1.00 | 95.4 | 90.9 | 1.28 | 88.7 | 84.0 | D + Gumbel | 0.30 | 86.4 | 82.4 | 0.53 | 83.7 | 80.1 |
| Beta | 0.80 | **91.1** | **87.2** | 1.50 | **86.9** | **82.9** | D + Beta | 0.10 | 86.2 | 82.3 | 0.60 | 81.5 | 77.9 |
| Chi | 0.20 | 111 | 105 | 1.50 | 99.0 | 92.9 | D + Chi | 0.20 | 92.0 | 87.4 | 0.29 | 87.1 | 82.8 |

**Experimental settings.** To assess the capabilities of NOISIN as a regularizer on its own, we used the basic settings for RNN training (Zaremba et al., 2014). We did not use weight decay or pointers (Merity et al., 2016). We considered two settings in our experiments: a medium-sized network and a large network. The medium-sized network has 2 layers with 650 hidden units each. This results in a model complexity of 13 million parameters. The large network has 2 layers with 1500 hidden units each. This leads to a model complexity of 51 million parameters. For the dropout-LSTM, the values used for dropout on the input, recurrent, and output layers were $0.5, 0.4, 0.5$ respectively. The models were implemented in PyTorch. The source code is available upon request.

**Results.** The results on the Penn Treebank are illustrated in Table 1. The best results for the non-regularized LSTM correspond to a small network. This is because larger networks overfit and require regularization. In general NOISIN improves any given RNN including dropout-LSTM. For example NOISIN with multiplicative Bernoulli noise performs better than dropout RNN for both medium and large settings. NOISIN improves the performance of the dropout-LSTM by as much as $12.2\%$ on this dataset. We observe the same trend on the Wikitext-2 dataset as for the Penn Treebank dataset: NOISIN improves the underlying LSTM and dropout-LSTM. For the dropout-LSTM, it improves its generalization capabilities by as much as $9\%$ on this dataset. (See Table 2.)

## 4 DISCUSSION

We proposed NOISIN, a simple method for regularizing RNNs. NOISIN injects noise into the hidden states such that the underlying RNN is preserved. NOISIN maximizes a lower bound on the log marginal likelihood of the data—the expected log-likelihood under the injected noise. On a language modeling benchmark NOISIN improves the generalization capabilities of both the LSTM and the dropout-LSTM.

## References


Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

Lawrence D Brown. Fundamentals of statistical exponential families with applications in statistical decision theory. *Lecture Notes-monograph series*, 9:i–279, 1986.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Katya Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. *arXiv preprint arXiv:1712.01769*, 2017.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pp. 2980–2988, 2015.

Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

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

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, pp. 2199–2207, 2016.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1019–1027, 2016.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, pp. 6716–6726, 2017.

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

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649. IEEE, 2013.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Kam-Chuen Jim, C Lee Giles, and Bill G Horne. An analysis of noise in recurrent neural networks: convergence and generalization. *IEEE Transactions on Neural Networks*, 7(6):1424–1438, 1996.

David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, Aaron Courville, et al. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

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.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. *SLT*, 12:234–239, 2012.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 3, 2010.

Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. In *Advances in Neural Information Processing Systems*, pp. 5113–5122, 2017.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning*, 28:1310–1318, 2013.

Barak A Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, 1995.

Tomaso Poggio, Ryan Rifkin, Sayan Mukherjee, and Alex Rakhlin. Bagging regularizes. Technical report, Massachusetts Institute of Technology, 2002.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Herbert Robbins. The empirical bayes approach to statistical decision problems. *The Annals of Mathematical Statistics*, 35(1):1–20, 1964.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering, 1987.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*, 2016.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 351–359, 2013.

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)*, pp. 1058–1066, 2013.

Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

Ronald J Williams. Complexity of exact gradient computation algorithms for recurrent neural networks. Technical report, Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science, 1989.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: a high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.