Stochastic Dropout: Activation-level Dropout to Learn Better Neural Language Models

Allen Nie
Symbolic Systems Program
Stanford University
`anie@stanford.edu`

Abstract

Recurrent Neural Networks are very powerful computational tools that are capable of learning many tasks across different domains. However, it is prone to overfitting and can be very difficult to regularize. Inspired by Recurrent Dropout [1] and Skip-connections [2], we describe a new and simple regularization scheme: Stochastic Dropout. It resembles the structure of recurrent dropout, but offers skip-connection over the recurrent depth. We reason the theoretical construct of such method and compare its regularization effectiveness with feedforward dropout and recurrent dropout. We demonstrate that Stochastic Dropout not only offers improvement when applied to vanilla RNN models, but also outperforms feedforward Dropout on word-level language modeling. At last, we show that the model can achieve even better result if stochastic dropout and feedforward dropout are combined.

1 Introduction

Overfitting has always been a problem for blackbox learning algorithms such as neural networks. They typically overfit the training data, and fail to generalize . The problem of overfitting is largely driven by the limited amount of training data, resulting in the model trying to fit the training noise perfectly but fail to recognize the real underlying structure, and successfully generalize [3]. Such problem is significant in a wide range of machine learning applications, such as image recognition, scene classification, and especially natural language processing.

Historically, many have worked on effective methods to prevent such problem. People have tried to apply regularization techniques on various parts of a neural network: weight decay on weight parameters (L_2 regularization) [4], dropout on hidden nodes [3], DropConnects on weights [5], data augmentation on input [6], stochastic pooling on the pooling layer for convolutional neural network [7], and Disturb Label on loss layer [8].

Most of the regularization techniques are developed on feedforward networks, but we are seeing the emergence of regularization on Recurrent Neural Networks. It is proven to be difficult to regularize a time-sequence dynamic system, notably due to it computing on two directions. Semeniuta et al [1] described a dropout scheme along the recurrent depth on the input update gate g. Cooijmans et al. [9] applied batch normalization to the gate activations computation as well as hidden state update. Such methods demonstrated effectiveness and importance of regularization in Recurrent Neural Network.

Beyond regularization, recent discoveries have suggested the power of skip-connections on neural networks. He et al. [10] demonstrated the power of simple additive skip-connection to allow deeper and better learning. Srivastava et al. [11] showed how to construct gate-like gradient highway that greatly expands the computing capability of convolutional neural network. In the domain of recurrent neural network, Zhang et al.[2] also described patterns of skip-connection on recurrent neural networks, and argue for the importance of recurrent depth as compared to feedforward depth.

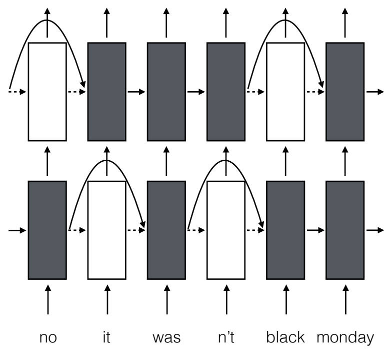


Figure 1: **Stocdrop schemes**: Illustration of proposed stocdrop methods. Every time step has δ probability of being skipped, the c_{t-1} output gets copied directly as c_{c+1}.

We propose a new scheme that not only regularizes recurrent neural network similar to the dropout methods, but effectively builds skip connections as well that allow network to learn better representations. We further demonstrate the effectiveness of this technique on word-level and character-level neural language modeling, and show superior performance compared to previous baselines on Penn Treebank task.

2 Related Work

2.1 Regularization

Dropout probably is the most discussed and explored regularization technique since its conception [3]. The effectiveness of dropout allows the building of large networks, and its ensemble effect is also well-documented. The original dropout is developed on a feedforward architecture. The recurrent depth direction dropout is formally studied by Moon et al. [12], Gal [13] and Semeniuta et al. [1]. Dropout effectively forces all neurons to participate in the final task, and by limiting the number of participating hidden units, it restricts the computational power of each sub-network, preventing the ability to overfit. Dropout is not only used for hidden state units. It has also been applied to input data augmentation, as argued by Wager et al. [14] for creating pseudo information-corrupted dataset.

2.2 Skip-connections

Skip connections in neural network have been explored by many. The earliest work in skip connections are examined in the context of convolutional neural network, which inherits the simple feedforward architecture of fully-connected networks. Highway Networks [11] builds gradient highway that through a transform gate, computed via a mixture of input and hidden states (similar to the gate calculation of a recurrent neural network), the network learns to allow a portion of input data (as well as intermediate layer input) to flow through. Building skip-connections allow bigger and deeper networks to be trainable, and continuing with this projection, ResNet [10] was developed. Instead of using a convolution as transform gate, ResNet did an additive application of the raw previous layer input and the convolved output. It allows faster computation time as well as even deeper depth. The latest discovery in feedforward skip-connection is Stochastic Depth Network [15]. Stochastic Depth Network skips a layer via either a constant probability or a probability with linear decay. It allows even deeper network with faster training time.

Skip-connections are also explored in Recurrent Neural Network. In Zhang et al. [2], the architecture of a recurrent neural network is described as having feedforward depth, recurrent depth, and recurrent skip-coefficients. Zhang explored several neural network with additional additional connections between two stacked hidden layers, but did not skip any computation, as Stochastic Depth Network would.

2.3 Language Modeling

Language modeling is one of the most long-lasting tasks in natural language processing. The traditional methods use n-gram with smoothing algorithms that uses discrete representation of words (normally one-hot encoded vector) as input representation [16]. Bengio et al. [17] first described the possibility of moving towards a more continuous representation of words and how they get combined. Word-level language is first explored in modern recurrent neural network structure by Mikolov [18], as well as character-level [19]. Character-level language modeling has the advantage of a small softmax projection the $|V_c|$ is much smaller than $|V_w|$. Deep recurrent neural network has been used by Graves [20], and convolution neural network has also been explored by Kim [21].

3 Data

TokenType	Train	Val	Test
CHAR	5017k	393k	442k
WORD	930k	74k	82k

Table 1: The amount of data in training, validation, and test set.

Due to various constraints, instead of One Billion Word Benchmark [22] or Text8 [19], we choose to train our language model on the more moderate and manageable task: Penn Treebank task [23]. It is composed of 4.5 Million English words, and is used by numerous models as the benchmark for language modeling tasks. We split the data in on character level as well as word level. We use the same split as in Mikolov et al. [19] for character-level modeling, and we use the same split as Graves [20] for word-level modeling. We tokenized the most frequent 10,000 words ($|V|$), and the outside vocabularies are labeled with `<UNK>` token.

4 Method

4.1 Neural Language Modeling

We define the language modeling task for a sequence of tokens $x = (x_1, x_2, \cdots, x_T)$, we maximize the probability of such sequence $p(x)$:

$$\begin{aligned} p(x) &= \prod_{t=1}^{T} p(x_t|x_{t-1}\cdots x_1) \\ p(x_t|x_{t-1}\cdots x_1) &= \prod_{l=1}^{d} \frac{exp(f(h_t, p_j))}{\sum_{i'} exp(f(h_t, p_{i'}))} \end{aligned} \tag{1}$$

Traditionally in the n-gram model, markov assumption would be used to simplify the above model to $p(x) = \prod_{t=1}^{T} p(x_t|x_{t-1})$, but arguments can be made that the most immediate or most N immediate tokens don't capture the full history. Recurrent Neural Network based model can represent an arbitrary length text capturing full history (in theory), while in practice such long-term dependency capture is quite difficult due to vanishing gradient. We use softmax to calculate a distribution over all vocabularies. Due to the limited size of our $|V|$, we choose not to use sampled softmax [24] or other sampling-based methods. We calculate the sequence cross-entropy loss as $L(x, \theta) = -\sum_t \log p_\theta(x_t|x_{t-1}\cdots x_1)$.

4.2 Dropout

4.2.1 Forward Dropout

We sample forward dropout mask along the feedforward axis from hidden states to hidden states in a stacked recurrent neural network. In a stacked RNN, every hidden state is the input of the higher level layer, thus forward dropout drops the hidden state input in the previous layer. According to the stacked RNN notation, we define it as:

$$\begin{aligned} d_j^{(l)} &\sim Bernoulli(p) \\ h_t^{(l)} &= U^{(l)} h_t^{(l)} + W^{(l)} d(h_t^{t-1}) \end{aligned} \tag{2}$$

We sample masks at each layer of our stacked RNN construction. In our experiment, we do not observe a noticeable difference if we apply forward dropout mask on the input.

4.3 Recurrent Dropout

There has been multiple methods to implement dropout along the recurrent depth. As discussed in Semeniuta et al. [1], in order to avoid test-time scaling problem: $h_t = ((((h_0 + g_0)p + g_1)p + ...)p + g_t)p$, we should avoid directly applying dropout mask on the hidden states. So we apply recurrent dropout as in Equation 4. We follow the practice in [1] and set up a recurrent dropout mask per sequence, and per batch.

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \cdot \begin{pmatrix} W_x^i & W_h^i \\ W_x^f & W_h^f \\ W_x^o & W_h^o \\ W_x^g & W_h^g \end{pmatrix} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \tag{3}$$

$$\begin{aligned} d_j^{(l)} &\sim Bernoulli(p) \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot (d \cdot g_t) \end{aligned} \tag{4}$$

4.4 Stochastic Dropout

We define stochastic dropout on LSTM, though it can be easily extended to GRU. We choose not to directly corrupt the data, even though it could be very effective and model agnostic [14]. We sample a dropout mask $D_{mask} \sim Bernoulli(p)$ where $D_{mask} \in \mathbb{R}^{\mathrm{T}}$. Instead of directly dropping the input or hidden state, we choose to erase the modulated input $i \cdot g$ that is a mixed representation of previous hidden state and input at current step. We formulate the update rule for LSTM in Equation 5.

$$\begin{aligned} c_t &= f_t \cdot c_{t-1} + i_t \cdot g_t \\ c_t &= (\mathbb{1} - D_{mask}) \cdot c_{t-1} + D_{mask}(i_t \cdot g_t) \end{aligned} \tag{5}$$

By using this formula, when the timestep when $D_{mask}^t = 1$, we have $c_t = c_{t-1}$. The previous cell state gets sent into the current cell state. This effectively converts the prediction probability from $p(x|x_{t-1} \cdots x_1) = p(x|h_t, c_t)$ to $p(x|h_t, c_{t-1})$. Cell state c_{t-1} is used at predicting x_o^{t-1}, as well as to predict x_o^t. It forces the cell state prior to the dropped time step to contain information that helps to predict two adjacent output words. This creates a very different effect compared to forward dropout or recurrent dropout that forbids certain hidden state units or gate activation units to be inactive. More analysis are provided in Section 6.

5 Experiment

5.1 Word Level Modeling

We conduct word-level language modeling experiments on Penn Treebank. All trained RNN models have 2 hidden layers and use LSTM units. Weights are initialized uniformly in $[0.1, 0.1]$. We consider dropout rates for all three types of dropout in $\{0.05, 0.1, 0.15, 0.2\}$. We use report results

Scheme	α	δ	Validation	Test
medium models ($\|h\| = 256$)				
no regularization	0.0	0.0	142	92
forward drop	0.05		144	**72**
	0.1		148	71
	0.15		151	93
	0.2		159	92
stochastic drop		0.05	146	**67**
		0.1	150	69
		0.15	148	86
		0.2	151	86
recurrent drop	0.05		142	67
	0.1		142	**66**
	0.15		150	95
	0.2		144	69
mixed scheme	0.05	0.05	135	**62**
	0.1	0.1	129	68
Mikolov (2012)[18]			NA	83.5
Graves (2013) [20]			NA	122
Zaremba et al. (2014) [25]			82.2	78.4
Kim et al. (2015) [21]			NA	78.9

Table 2: Word-level sequence cross-entropy loss on Penn Treebank with different dropout schemes. α is the feedforward or recurrent dropout rate, δ is the stochastic dropout rate. In mixed scheme, we combined stochastic scheme and forward dropout scheme. We report the perplexity calculated based on sequential cross entropy cost.

for hidden sizes $|h|$ of 256. We unroll for 25 steps and use batches sizes of 50. At the end of the experiment, our models are trained on average 15 epochs.

We train using Adam optimization [26] with an initial learning rate of 0.002. When the decrease in mean validation cross entropy is less than 0.01, we divide the learning rate in half. We allow the learning rate to be decreased 8 times before stopping training and pick the best model based on sequence cross-entropy cost on the validation set. We report the finding in Table 2.

Except Kim et al. [21], the rest of models are implemented in recurrent neural network with one layer, compared to our two-layer design. We can easily observe that our model is outperforming previous models by a big margin even with simple forward dropout. In this setting, stochastic dropout outperforms forward dropout as well. In the mixed shceme, we combine stochastic dropout and feedforward dropout, and thus achieving the best perplexity across all our models.

5.2 Character Level Modeling

We also conducted character level language modeling. We use mostly the same parameter as mentioned in character-level modeling, except we only unroll 50 steps as opposed to 50 steps. We calculate the perplexity based on the cross-entropy loss, and our result is compared with other models.

As we can see, stochastic dropout is a very effective regularization tool compared to the unregularized baseline. It performs quite similar to recurrent dropout, but fails to outperform simple forward dropout in the character-level setting. This shouldn't be too surprising because there are less extraneous characters that are unimportant to the word, than extraneous words that are unimportant to a sequence.

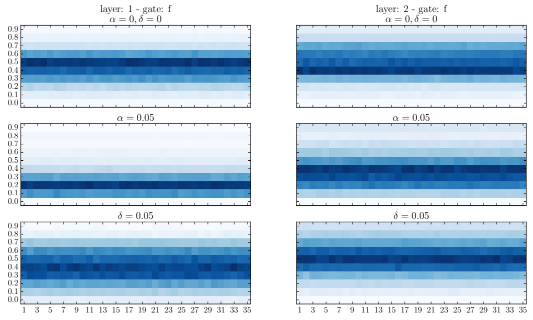
Scheme	δ	Validation	Test
medium models ($\lvert h \rvert = 256$)			
no regularization	0.0	1.012	**1.256**
forward drop	0.05	1.014	1.242
	0.1	1.008	1.254
	0.15	1.007	**1.209**
	0.2	1.007	1.210
stochastic drop	0.05	1.013	1.247
	0.1	1.008	1.241
	0.15	1.008	**1.223**
	0.2	1.007	1.231
recurrent drop	0.05	1.013	1.247
	0.1	1.004	**1.221**
	0.15	1.024	1.225
	0.2	1.022	1.245

Table 3: Character-level sequence cross-entropy loss on Penn Treebank with different dropout schemes. δ represents the shared dropout probability. We report the sequential cross entropy cost.

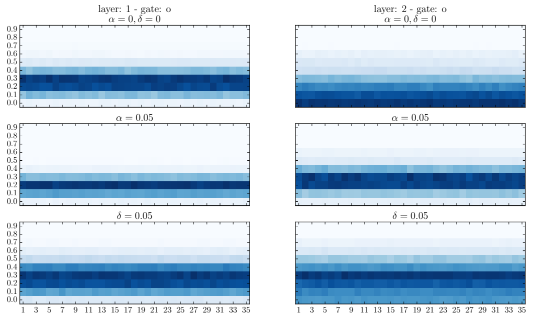
6 Analysis

6.1 Gate Activations

In order to analyze the effect of stochastic dropout, we visualize the gate activations of three models: an un-regularized model, a feedforward dropout regularized model, and a stochastic dropout regularized model. All three models use the most optimal trained parameters from Section 5.1. We only display the gate activations that display the most difference amongst the models, which are the output gates and forget gates.



(a) Activations on f forget gate



(b) Activations on o output gate

Figure 2: **Gate Activations** The comparison of gate activations on two RNN layers among three models: α is the feedforward dropout rate, δ is the stochastic dropout rate. Activations are averaged across all batches.

We can easily observe on Figure 2, the similarity between feedforward dropout-regularized model and stochastic dropout-regularized model, both display similar shift of gate activations compared to the unregularized model. The similarity in the patterns could be an indication that stochastic dropout is regularizing the model akin to other dropout methods, corresponding to our original hypothesis.

We can also observe that stochastic dropout leads to higher means for forget gate in both first and second layer compared to other two models. This can be reasoned as since we force the model to use c_{t-1} to predict x_o^{t-1} and x_o^t, it will learn to hold more information, leading to better result for long-term dependency learning. Also due to the fact that we computed average for gate activations, we can observe that the effect of dropout methods is time-invariant.

Another evidence to notice is that the forget gate and output gate for the un-regularized model on the second layer tend to be more left-saturated - more likely erasing information from the previous time steps. This effect is likely to intensify when the feedforward depth increases. Regularization such as stochastic dropout or feedforward dropout can encourage gate cells to be less saturated.

6.2 Gradient Decay

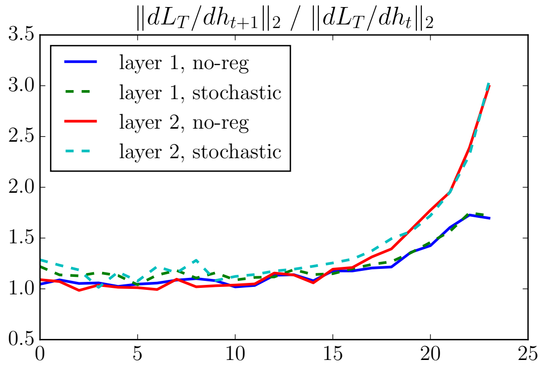


Figure 3: Gradient norm ratios. Computed with respect to cost at final time step.

We set to explore if a model trained with stochastic dropout will have a lower rate of gradient decay through time, since gradient vanishing has been a major problem for recurrent neural network. We calculate the 2-norm of the gradient vector backpropagating from the final time step T to the first timestep. We then calculate the ratio of gradient norm between the previous timestep to current timestep. As we can see in Figure 3, the stochastic dropout-trained model, the gradient tend to show a slower decay compared to a non-regularized model.

7 Future Direction

In this paper, we examined adding skip-connection along the recurrent depth, but we have not examined adding skip-connection along feedforward depth. It has always been particularly difficult to train feedforward deep recurrent neural network, and such difficulty is well explained in Zhang et al. [2] that feedforward depth does not help retaining long-term dependency. However, in the success of deep convolutional neural network [11] [10], one can only wonder what would a feedforward deep recurrent neural network bring if trained successfully.

Obviously this technique can be easily carried over to Gated Recurrent Unit (GRU) [27], but our initial experiments have showed mixed result as to whether this improves performance. Since neural language modeling is a very difficult task, it would be interesting to see this technique being applied to other tasks such as sentiment analysis or machine translation.

8 Conclusion

We have demonstrated with emprical results the effectiveness of dropping modulated input on time steps. It is an effective regularizer, and on word-level experiment, we showed the model outperforming other dropout methods. At last, we show that since recurrent neural network has two depths: recurrent depth and feedforward depth, we can combine the stochastic dropout with feedfoward dropout and achieve even better performance.

Acknowledgements

Special thanks to Ziang Xie, Richard Socher, Daniel Levy, and Stanford Artificial Intelligence Lab for ideas and/or computational support for this research.

References

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

[2] Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Ruslan Salakhutdinov, and Yoshua Bengio. Architectural complexity measures of recurrent neural networks. *arXiv preprint arXiv:1602.08210*, 2016.

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

[4] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[5] 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.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

[8] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. *arXiv preprint arXiv:1605.00055*, 2016.

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

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[11] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[12] Taesup Moon, Heeyoul Choi, Hoshik Lee, and Inchul Song. Rnndrop: A novel dropout for rnns in asr. *Automatic Speech Recognition and Understanding (ASRU)*, 2015.

[13] Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. *arXiv preprint arXiv:1512.05287*, 2015.

[14] Stefan Wager, William Fithian, and Percy Liang. Data augmentation via levy processes. *arXiv preprint arXiv:1603.06340*, 2016.

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. *arXiv preprint arXiv:1603.09382*, 2016.

[16] James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 2000.

[17] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

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

[19] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and J Cernocky. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 2012.

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

[21] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*, 2015.

[22] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[23] 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.

[24] Sébastien Jean Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation.

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

[26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.