Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations

David Krueger[1], Tegan Maharaj[2], János Kramár[2,⋆], Mohammad Pezeshki[1,⋆]
Nicolas Ballas[1], Nan Rosemary Ke[1], Anirudh Goyal[1]
Yoshua Bengio[1†], Hugo Larochelle[3], Aaron Courville[1‡], Chris Pal[2]
[1] MILA, Université de Montréal, firstname.lastname@umontreal.ca.
[2] École Polytechnique de Montréal, firstname.lastname@polymtl.ca.
[3] Twitter, hlarochelle@twitter.com.
⋆ Equal contributions. †CIFAR Senior Fellow. ‡CIFAR Fellow.

June 14, 2021

Abstract

We propose zoneout, a novel method for regularizing RNNs. At each timestep, zoneout stochastically forces some hidden units to maintain their previous values. Like dropout, zoneout uses random noise to train a pseudo-ensemble, improving generalization. But by preserving instead of dropping hidden units, gradient information and state information are more readily propagated through time, as in feedforward stochastic depth networks. We perform an empirical investigation of various RNN regularizers, and find encouraging results: zoneout gives significant performance improvements across tasks, yielding state-of-the-art results in character-level language modeling on the Penn Treebank dataset and competitive results on word-level Penn Treebank and permuted sequential MNIST classification tasks.

1 Introduction

We address the issue of regularization in recurrent neural networks (RNNs). Recurrent nets have recently been used to achieve state-of-the-art results in many machine learning domains including speech recognition [Miao et al., 2015], computer vision [Yao et al., 2015], language modeling [Kim et al., 2015] and machine translation [Bahdanau et al., 2014]. Solving machine learning tasks requires high capacity models, such as RNNs, that are flexible enough to capture complex relationships in the data. Regularizing neural networks often yields large performance gains when data is a limiting factor, as indicated by the frequent use of early stopping.

The use of RNNs is also motivated by their ability to sequentially construct fixed-length representations of arbitrary-length sequences by folding new observations into their hidden state using an input-dependent transition operator. The repeated application of the same transition operator, however, can make the dynamics of an RNN sensitive to minor perturbations in the hidden state.

Motivated by the desire to improve robustness in the hidden dynamics of RNNs, and inspired by dropout [Hinton et al., 2012, Srivastava et al., 2014], we propose a novel regularizer, called **zoneout**, which stochastically modifies the transition operator during training. Instead of setting some units' activations to 0, as in dropout, zoneout replaces some units' activations with their activations from the previous time-step. As in dropout, we use the expectation of the random noise at test time.

Compared with dropout, zoneout is appealing because it preserves information flow forwards and backwards through the network. Moreover, RNNs are frequently trained with output parameters which are shared across time-steps[1], and this parameter sharing encourages recurrent units' activations to be semantically consistent

[1]This is the case both when training with targets at every time-step, e.g. as in language modelling, or with a single target at the end of a variable-length sequence, e.g. as in sentiment analysis.

across time. Thus zoning out a unit in an RNN preserves not only its activation, but its meaning as well.

We empirically evaluate zoneout on classification using the permuted sequential MNIST dataset, and on language modeling using the Penn Treebank dataset, where we demonstrate competitive performance. In particular, we show the benefit of our approach compared with the most successful application of dropout on recurrent connections [Semeniuta et al., 2016] and other proposed regularizers for RNNs.

2 Related work

Concurrent with our work, Singh et al. [2016] propose zoneout for feedforward nets, calling the technique **SkipForward**. In their experiments on feedforward nets, zoneout is outperformed by stochastic depth, dropout, and their proposed **swapout** technique, which randomly drops either or both of the identity or residual connections, and gives the best performance on image recognition tasks. Contrarily, we find that zoneout outperforms stochastic depth and recurrent dropout in our experiments with RNNs.

2.1 Relationship to dropout

Zoneout is named after the popular dropout [Srivastava et al., 2014] regularization technique. In zoneout, instead of dropping out (being set to 0), units *zone out* and are set to their previous value ($h_t = h_{t-1}$). Zoneout, like dropout, trains a pseudo-ensemble [Bachman et al., 2014], but uses a stochastic "identity-mask" rather than a zero-mask. We conjecture that identity-masking is more appropriate for RNNs, since it makes it easier for the network to preserve information from previous time-steps going forwards, and it facilitates, rather than hinders, the flow of gradient information going backwards. Zoneout can also be seen as a selective application of dropout to only some of the nodes in a modified computational graph, as shown in Figure 1. This justifies our use of the expected identity-mask at test time.

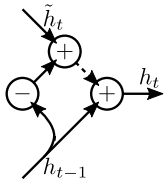


Figure 1: Zoneout as a special case of dropout. $\tilde{h}_t$ is the hidden activation without zoneout; the hidden state h_t has zoneout applied stochastically as represented by the dashed line. This can be seen as dropout on the corresponding input node, which represents the difference $\tilde{h}_t - h_{t-1}$.

2.2 Dropout in RNNs

Initially successful applications of dropout in RNNs [Pham et al., 2013, Zaremba et al., 2014] only apply dropout to feed-forward connections ("up the stack"), and not recurrent connections ("forward through time"). Bayer et al. [2013] successfully apply fast dropout [Wang and Manning, 2013], a deterministic approximation of dropout, to RNNs. We now describe several recent works [Semeniuta et al., 2016, Moon et al., 2015, Gal, 2015] that propose variants of dropout for use in RNNs.

Semeniuta et al. [2016] apply **recurrent dropout** to the *updates* to LSTM memory cells (or GRU states), i.e. they drop out the input/update gate in LSTM/GRU. Their motivation is to prevent the loss of long-term memories stored in the memory cells of an LSTM (or the states of a GRU). Their proposed method is similar to zoneout applied to these models. Zoneout differs, however, in that zoned-out units' states are preserved *exactly*. This difference is most salient when zoning out the hidden states (not the memory cells) of an LSTM, for which there is no analogue in recurrent dropout. Whereas saturated output gates or output nonlinearities would cause recurrent dropout to suffer from the vanishing gradient problem [Bengio et al., 1994], the identity-masked units in zoneout still propagate gradients effectively in this situation. Furthermore,

while the recurrent dropout method is specific to LSTMs and GRUs, zoneout generalizes their intuition to any model that sequentially builds distributed representations of its input.

Also motivated by preventing memory loss, Moon et al. [2015] propose **rnnDrop**. This technique amounts to using the same dropout mask at every time-step, which the authors show results in improved performance on speech recognition in their experiments. Gal [2015] proposed the same technique and evaluated it on natural language processing tasks. His motivation, however, came from a variational inference perspective, i.e. he views dropout masks as applied to parameters rather than hidden units.

2.3 Relationship to Stochastic Depth

Zoneout can also be viewed as a per-unit version of **stochastic depth** [Huang et al., 2016]. Recently introduced as a method to efficiently train very deep feed-forward neural networks, stochastic depth randomly drops entire layers. To our knowledge, this method has not been extended to a recurrent setting. This is equivalent to zoning out all of the units of a layer at the same a time in a feedforward network. In a typical RNN, there is a new input at each time-step, causing issues for a naive implementation of stochastic depth. Firstly, Huang et al. [2016] train deep residual networks (ResNets [He et al., 2015]) and preserve the identity connections when dropping a layer, so that the inputs from the previous layer are fed into the next layer in the stack. Including such residual connections in recurrent nets led to instability in our experiments, presumably due to the parameter sharing in RNNs. Zoning out an entire layer in an RNN also means the input at the corresponding time-step is completely ignored, whereas zoning out individual units allows the RNN to process each element of its input sequence.

3 Zoneout and preliminaries

We now explain zoneout in full detail, and compare with other forms of dropout in RNNs. We start by reviewing recurrent neural networks (RNNs).

3.1 Recurrent Neural Networks

Recurrent neural networks process data $x_1, x_2, \ldots, x_T$ sequentially, constructing a corresponding sequence of representations, $h_1, h_2, \ldots, h_T$. The transition operator, $\mathcal{T}$, of a recurrent neural network converts the present hidden state and input into a new hidden state: $h_t = \mathcal{T}_t(h_{t-1}, x_t)$. Thus each hidden state is trained (implicitly) to remember and emphasize all task-relevant aspects of the preceding inputs, and to incorporate new inputs into its representation of state.

Another way to see zoneout is as a modification of the transition operator, similar to dropout. For both methods, the masking coefficient, d_t, is a Bernoulli random vector sampled independently at each time-step during training, and is replaced by its expectation at test time. Then, zoneout mixes the original transition operator $\tilde{\mathcal{T}}_t$ with the identity operator, as opposed to the null operator used in dropout:

$$\text{Zoneout:} \qquad \mathcal{T}_t = d_t \odot \tilde{\mathcal{T}}_t + (1 - d_t) \odot 1 \qquad\qquad \text{Dropout:} \qquad \mathcal{T}_t = d_t \odot \tilde{\mathcal{T}}_t + (1 - d_t) \odot 0$$

In simple RNNs, the hidden state is computed by a standard "linear layer", i.e., an affine transform followed by a nonlinearity ϕ:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b),$$

A naive application of dropout here is to simply apply a stochastic zero-mask to the updated h_t; zoneout is similarly straightforward.

3.2 Long short-term memory

In long short-term memory RNNs (LSTMs), the hidden state is divided into memory cells c_t which are intended for internal long-term storage, and hidden state, h_t which is used as a transient representation of state at time-step t. In an LSTM, c_t and h_t are computed via a set of four "gates", including the forget gate, f_t, which directly connects c_t to the memories of the previous time-step c_{t-1}, via an element-wise multiplication. Large values of the (ironically named) forget gate cause the cell to remember most (but not all) of its previous value. The other gates control the flow of information in (i_t, g_t) and out (o_t) of the cell. Each gate has its own weight matrices and bias vector; for example the forget gate has W_{xf}, W_{hf}, and b_f. For brevity, we will write these as W_x, W_h, b.

An LSTM is defined as follows:

$$\begin{aligned} i_t, f_t, o_t &= \sigma(W_x x_t + W_h h_{t-1} + b) \\ g_t &= \tanh(W_{xg} x_t + W_{hg} h_{t-1} + b_g) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

A naive application of dropout in LSTMs zero-masks either or both of the memory cells and hidden states. For example, here is an LSTM with dropout applied to the memory cells:

$$\begin{aligned} i_t, f_t, o_t &= \sigma(W_x x_t + W_h h_{t-1} + b) \\ g_t &= \tanh(W_{xg} x_t + W_{hg} h_{t-1} + b_g) \\ c_t &= d_t \odot (f_t \odot c_{t-1} + i_t \odot g_t) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

Alternatives abound, however; masks can be applied to any subset of the gates, cells, and states. Recently, Semeniuta et al. [2016] proposed zero-masking the input gate (which they call *recurrent dropout*):

$$\begin{aligned} i_t, f_t, o_t &= \sigma(W_x x_t + W_h h_{t-1} + b) \\ g_t &= \tanh(W_{xg} x_t + W_{hg} h_{t-1} + b_g) \\ c_t &= (f_t \odot c_{t-1} + d_t \odot i_t \odot g_t) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

When the input gate is masked like this, there is no additive contribution from the input or hidden state, and the value of the memory cell simply decays according to the forget gate.

3.3 Zoneout in LSTMs

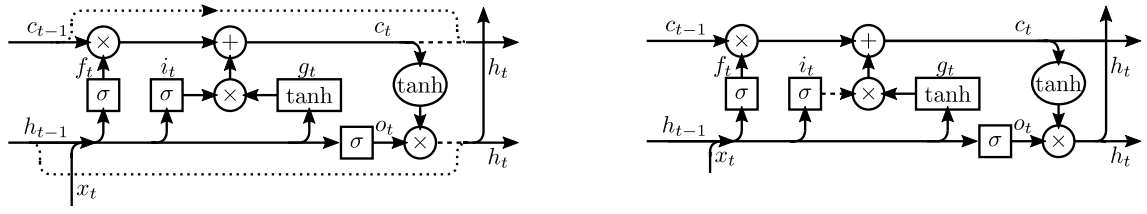


Figure 2: (a) Zoneout, vs (b) the recurrent dropout strategy of [Semeniuta et al., 2016] in an LSTM. Dashed lines are zero-masked, and, in zoneout, the corresponding dotted lines are masked with the corresponding opposite zero-mask. Rectangular nodes represent so-called "linear" layers.

In *zoneout*, the values of the hidden state and memory cell may stick to their previous value (stochastically), or be updated as usual. This introduces stochastic identity connections between subsequent time steps:

$$c_t = d_t \odot c_{t-1} + (1 - d_t) \odot \big(f_t \odot c_{t-1} + i_t \odot g_t\big)$$
$$h_t = d_t \odot h_{t-1} + (1 - d_t) \odot \big(o_t \odot \tanh\big(f_t \odot c_{t-1} + i_t \odot g_t\big)\big)$$

We also experiment with a variant of recurrent dropout that reuses the input dropout mask to zoneout the corresponding output gates:

$$\begin{aligned} i_t, f_t, o_t &= \sigma(W_x x_t + W_h h_{t-1} + b) \\ g_t &= \tanh(W_{xg} x_t + W_{hg} h_{t-1} + b_g) \\ c_t &= (f_t \odot c_{t-1} + d_t \odot i_t \odot g_t) \\ h_t &= ((1 - d_t) \odot o_t + d_t \odot o_{t-1}) \odot \tanh(c_t) \end{aligned}$$

The motivation for this variant is to prevent the network from being forced (by the output gate) to expose a memory cell which has not been updated, and hence may contain misleading information.

4 Experiments and Discussion

We evaluate zoneout's performance on the following tasks:

- Classification of hand-written digits on permuted sequential MNIST [Le et al., 2015].
- Word-level language modeling on the Penn Treebank corpus [Marcus et al., 1993].
- Character-level language modeling on the Penn Treebank corpus

We first investigate zoneout with a shared zoneout mask on cells and hiddens on the above tasks and compare its performance with other regularizers. In order to better study the dynamics of memory in LSTM and zoneout's effects on those dynamics, we also explore models with different zoneout probabilities on cells and hidden states.

4.1 Sequential Permuted MNIST

In sequential MNIST, pixels of an image representing a number [0-9] are presented to a recurrent neural network one at a time, in lexographic order (left to right, top to bottom). The task is to classify the number shown in the image. In permuted sequential MNIST (pMNIST), the pixels are presented in a (fixed) random order.

We compare recurrent dropout and zoneout to a vanilla LSTM baseline on this task. All models are trained for 150 epochs using RMSProp [Tieleman and Hinton, 2012] with a decay rate of 0.5 for the moving average of gradient norms. The learning rate is set to 0.001 and the gradients are clipped to a maximum norm of 1.0 [Pascanu et al., 2013]. The recurrent dropout probability and zoneout probabilites are both set to 0.15. As shown in Table 1, zoneout gives a significant performance boost compared to the LSTM baseline and outperforms recurrent dropout, although recurrent batch normalization outperforms all three. Figure 3 shows the training and validation curves for this task.

4.2 Penn Treebank Language Modeling Dataset

The Penn Treebank language model corpus contains 1 million words with predefined training, validation and test splits. The model is trained to predict the next word or character, and is evaluated based on the perplexity for word-level, or bits per character (BPC) for character-level, as is typical for these tasks[2].

[2] These metrics are deterministic functions of the negative log-likelihood (NLL). Specifically, perplexity is the exponentiated NLL, and BPC is NLL divided by the natural logarithm of 2.

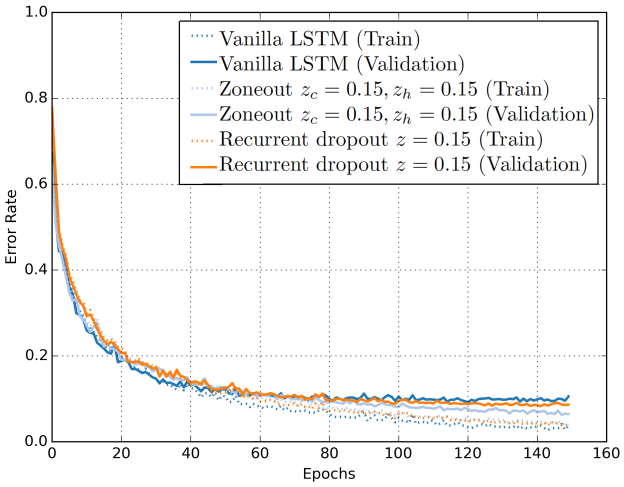


Figure 3: Training and validation error rates for vanilla LSTM, recurrent dropout, and zoneout on the task of permuted sequential MNIST digit classification.

Table 1: Error rates of different models on the permuted sequential MNIST digit classification task. Zoneout outperforms recurrent dropout.

*p*MNIST		
Model	**Valid**	**Test**
LSTM	0.092	0.102
Recurrent dropout	0.083	0.075
Zoneout	0.063	0.069
BatchNorm	-	0.046

4.2.1 Character-level

For the character-level task, we use an LSTM with one hidden layer with 1000 units. We train on non-overlapping sequences of 100 in batches of 32. We optimize using Adam with a learning rate of 0.002 and gradient clipping with threshold 1.

Figure 4a shows our exploration of different zoneout models, with different probabilities of zoneout for cells and hiddens. Results are reported in 2; we find that zoneout on cells with probability 0.5 and zoneout on states with probability 0.05 both outperform recurrent dropout, and combining these leads to our best-performing model ($z_c = 0.5, z_h = 0.05$).

We compare zoneout to recurrent dropout (for $p \in \{.05, .2, .5, .7\}$), weight noise ($\sigma = 0.075$), and norm stabilizer ($\beta = 50$) [Krueger and Memisevic, 2015]. Figure 4b shows the best performance achieved with each of these regularizers, as well as an unregularized LSTM baseline.

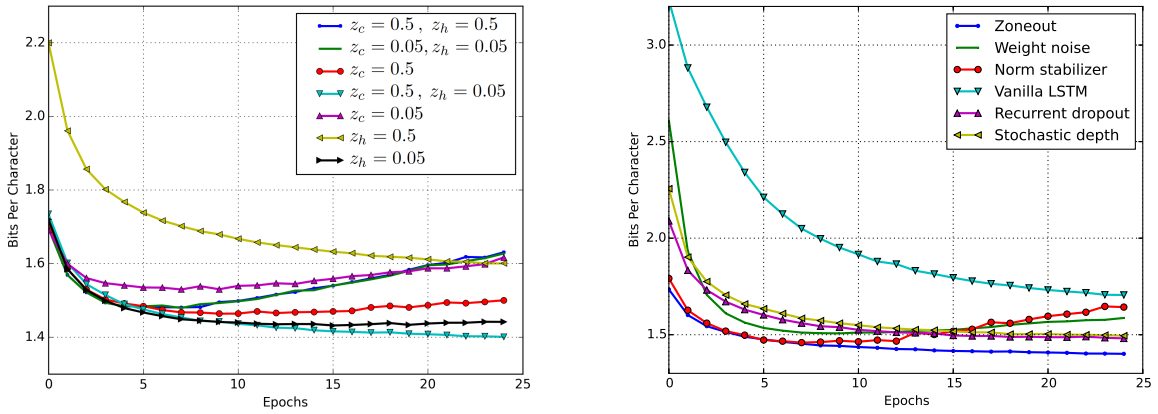
Zoneout with probability $z_c = 0.5$ on cells and $z_h = 0.05$ on hiddens achieves state-of-the-art BPC of 1.29. These results are shown in Table 2. By training on overlapping sequences, we are able to further improve this result to 1.27 BPC.

4.2.2 Word-level

For the word-level task, we use similar settings to Semeniuta et al. [2016] and Mikolov et al. [2014]. Unlike these previous works, however, we slice the data into non-overlapping sequences, and we do not use the last hidden states from one batch to initialize the next batch's hidden states. This means we effectively train on

Table 2: Bits per character (BPC) of different models on the character-level Penn Treebank Corpus, with data taken in non-overlapping slices

character-level Penn Treebank		
Model	**Valid**	**Test**
$z_c = 0.5, \ z_h = 0.5$	1.66	
$z_c = 0.05, z_h = 0.05$	1.417	
$z_c = 0.5$	1.464	
$z_c = 0.5, \ z_h = 0.05$	1.362	
$z_c = 0.05$	1.53	
$z_h = 0.5$	1.582	
$z_h = 0.05$	1.431	
LSTM	1.664	1.398
Recurrent dropout	1.481	1.382
BatchNorm	–	1.32
Zoneout	1.362	1.297



(a) (b)

Figure 4: Validation BPCs (bits per character) on Character-level Penn Treebank, for (a) different probabilities of zoneout on cells (z_c) and hidden states (z_h), and (b) a comparison with other regularizers

less data, and so our results are not directly comparable to those reported in Semeniuta et al. [2016].

The model is an LSTM with a single hidden layer of 256 units and weights initialized uniformly between -0.05 and +0.05. We train on non-overlapping sequences of length 35, in batches of 32, and optimize using SGD with momentum of 0.99 and gradient clipping threshold of 10. We start with learning rate of 1, reducing it by a factor of 1.5 after every epoch without an improvement in validation performance after Semeniuta et al. [2016].

We perform a grid search over zoneout/dropout probabilities $\{0.5, 0.25, 0.05\}$ for the following methods: zoneout on cells, zoneout on hiddens, zoneout on hiddens and cells (same p, different masks), recurrent dropout, recurrent dropout with output zoneout, and stochastic depth. We also compare with norm-stabilizer ($\beta = 50$), weight noise ($\sigma = 0.075$), and an an unregularized vanilla LSTM. We report the best performance achieved with a given technique in Table 3. The top three performing models all used some form of zoneout.

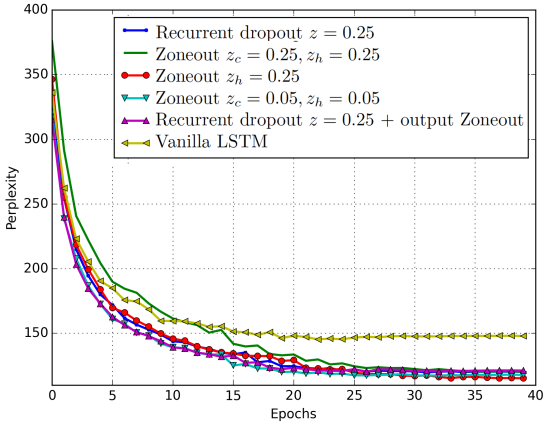


Figure 5: Training and validation perplexity for vanilla LSTM, recurrent dropout, zoneout and weight noise on the task of word-level Penn Treebank Corpus.

Table 3: Perplexity for word-level language modelling task on Penn Treebank Corpus. Zoneout and output zoneout outperform competitors.

word-level Penn Treebank	
Model	**Valid**
Zoneout ($p_{\text{states}} = 0.25$)	115.2
Recurrent dropout with output zoneout ($p = 0.25$)	115.7
Zoneout ($p_{\text{cells}} = p_{\text{states}} = 0.05$)	117.3
Recurrent dropout ($p = 0.5$)	119.9
Stochastic depth ($p = 0.25$)	129.9
Norm-stabilizer ($\beta = 50$)	141.8
Zoneout ($p_{\text{cells}} = 0.5$)	143.6
Baseline (no regularization)	145.4
Weight noise ($\sigma = 0.075$)	172.0

5 Conclusion

We have introduced zoneout, a novel way to regularize RNNs. Instead of *dropping out* hidden units, we *zoneout* hidden units, stochastically replacing their activations with the activations from one timestep earlier. We find zoneout brings significant performance improvements on the permuted sequential MNIST and Penn Treebank datasets, outperforming alternative regularizers, and achieving state of the art performance on character-level language modelling.

We conjecture that the improved generalization of zoneout arises from two main factors:

- The stochasticity introduced by zoneout makes the network more robust to changes in the hidden state.
- The identity connections of zoneout improve the flow of information through the network.

Future work could allow recurrent networks to adaptively set the probabilities of updating various units based on the input sequence. This could be used to learn a dynamic version of the hierarchical RNN proposed by Hihi and Bengio [1996], where the network would learn when to update different dimensions of its hidden state, which could be used to store information about different time-scales.

Acknowledgments

We thank the developers of Theano [Theano Development Team, 2016], as well as Fuel and Blocks [van Merriënboer et al., 2015]. We also acknowledge the computing resources provided by ComputeCanada and CalculQuebec. We are grateful to students at MILA, especially Çağlar Gülçehre, Marcin Moczulski, Chiheb Trabelsi, and Christopher Beckham, for feedback and helpful discussions. We also thank IBM and Samsung for their support.

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government[3].

References

Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pages 3365–3373, 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

J. Bayer, C. Osendorfer, D. Korhammer, N. Chen, S. Urban, and P. van der Smagt. On Fast Dropout and its Applicability to Recurrent Networks. *ArXiv e-prints*, November 2013.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

Yarvin Gal. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *ArXiv e-prints*, December 2015.

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

Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In D. S. Touretzky and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 493–499. MIT Press, 1996. URL `http://papers.nips.cc/paper/1102-hierarchical-recurrent-neural-networks-for-long-term-dependencies.pdf`.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

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

David Krueger and Roland Memisevic. Regularizing rnns by stabilizing activations. *arXiv preprint arXiv:1511.08400*, 2015.

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

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.

[3] David Krueger is funded by DARPA/AFRL.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. *arXiv preprint arXiv:1507.08240*, 2015.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

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

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

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves Recurrent Neural Networks for Handwriting Recognition. *ArXiv e-prints*, November 2013.

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

S. Singh, D. Hoiem, and D. Forsyth. Swapout: Learning an ensemble of deep architectures. *ArXiv e-prints*, May 2016.

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.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:2, 2012.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619, 2015. URL `http://arxiv.org/abs/1506.00619`.

Sida Wang and Christopher Manning. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning*, pages 118–126, 2013.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4507–4515, 2015.

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