SEMANTIC WORD EMBEDDING NEURAL NETWORK LANGUAGE MODELS FOR AUTOMATIC SPEECH RECOGNITION

Kartik Audhkhasi, Abhinav Sethy, Bhuvana Ramabhadran

IBM Watson
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
Email: {kaudhkha,asethy,bhuvana}@us.ibm.com

ABSTRACT

Semantic word embeddings have become increasingly important in natural language processing tasks over the last few years. This popularity is due to their ability to easily capture rich semantic information through a distributed representation and the availability of fast and scalable algorithms for learning them from large text corpora. State-of-the-art neural network language models (NNLMs) used in automatic speech recognition (ASR) and natural language processing also learn word embeddings optimized to model local N-gram dependencies given training text but are not optimized to capture semantic information. We hypothesize that semantic word embeddings provide diverse information compared to the word embeddings learned by NNLMs. We propose novel feedforward NNLM architectures that incorporate semantic word embeddings. We apply the resulting NNLMs to ASR on broadcast news and show improvements in both perplexity and word error rate.

Index Terms— Word embeddings, neural networks, language modeling, automatic speech recognition, representation learning.

1. INTRODUCTION AND PRIOR WORK

Recent advances in machine learning have led to the emergence of continuous distributed vector representations of words compared to traditional discrete and sparse representations such as one-hot encoding and bag of words. These distributed representations are often derived using algorithms that aim to capture semantic information from text. Two prominent examples include *word2vec* [1] and *Glove* (global vectors for word representation) [2]. The continuous bag of word (CBOW) variant of word2vec learns word representations that best predict the current word in a sentence given the past and future words. This model takes the average of past and future word representations as input to a one hidden layer neural network to predict the current word. Another variant of the word2vec model, Skip-Gram, uses the embedding of the current word to predict the neighboring words.

Glove is similar in spirit to word2vec because it also tries to estimate word embeddings that capture word co-occurrence. However, Glove achieves this through a bilinear approximation of the word co-occurrence matrix. Let $\mathbf{G}$ denote the $V \times D$ matrix containing D-dimensional word embeddings for V words. Let $\mathbf{C}$ be the pairwise word co-occurrence matrix whose $(i,j)^{th}$ entry $\mathbf{C}(i,j)$ is the weighted count of the number of times words i and j occur within a window of length W words. Glove estimates the optimal $\mathbf{G}$ and a V-dimensional bias vector $\mathbf{b}$ using the following weighted least squares optimization problem:

$$\mathbf{G}^*, \mathbf{b}^* = \arg\min_{\mathbf{G},\mathbf{b}} \sum_{i,j=1}^{V} f(\mathbf{C}(i,j)) \Big(\mathbf{G}(i)\mathbf{G}(j)^T + b(i) + b(j) - \log \mathbf{C}(i,j) \Big)^2 , \quad (1)$$

where $\mathbf{G}(i)$ denotes the i^{th} row of $\mathbf{G}$ and f is

$$f(x) = \min \left\{ 1, \left(x/x_{min} \right)^{\alpha} \right\} . \quad (2)$$

Typical values of α and x_{min} are 0.75 and 100 respectively. Semantic word embeddings have found applications in several natural language processing (NLP) tasks including word similarity (analogy) detection [1, 2], named entity recognition [3, 4], machine translation [5–7], text classification [8], and dependency parsing [9].

The recent success of semantic word embeddings in NLP has occurred in parallel with the advent of neural network language models (NNLMs). Models such as feed-forward NNLMs (FNNLMs), recurrent NNLMs (RNNLMs), and uni-/bi-directional long short-term memory (LSTM) RNNLMs are used in almost all state-of-the-art automatic speech recognition (ASR) systems [10–16] in combination with the traditional N-gram LMs that do not learn a continuous distributed word representation.

Our key observation is that the word representations learned by any NNLM are tuned to best predicting the next word in a sentence because NNLM training uses average sentence log-likelihood as the objective function. As an example, Table 1 lists top-5 closest words to five words found using cosine similarity between NNLM embeddings trained on broadcast news and Glove embeddings trained on the Gigaword corpus. We observe that the lists of words closest to a given word are significantly different for NNLM and Glove. We thus hypothesize that semantic word embeddings can offer information diverse to a NNLM. This leads us to the central contribution of this paper - a joint model incorporating semantic words embeddings with a NNLM. The next section describes our proposed model. Section 3.1 presents our Glove training setup. Section 3.2 presents ASR experiments using baseline and semantic word embedding FNNLMs. Section 3.3 presents some initial results by extending the model to RNNLMs and Section 4 concludes the paper.

2. SEMANTIC WORD EMBEDDING NNLM

We first describe the traditional FNNLM from Figure 1 to set the notation. Let $(\mathbf{w}_{i-2}, \mathbf{w}_{i-1}, \mathbf{w}_i)$ be a tuple of three 1-in-V row vectors, where V is the vocabulary size. A FNNLM first converts the

Speech		**Signal**		**Processing**		**Machine**		**Learning**	
NNLM	*Glove*	*NNLM*	*Glove*	*NNLM*	*Glove*	*NNLM*	*Glove*	*NNLM*	*Glove*
Address	Remarks	Message	Signals	Sewer	Processed	Stun	Machines	Learn	Learn
Event	Address	Reply	Signaling	Correctional	Applications	Pellet	Guns	Learned	Teaching
Ceremony	Speeches	Attribute	Indication	Reprocessing	Manufaturing	Celebratory	Gun	Learns	Learned
Statement	Comments	Disservice	Indicating	Placement	Processes	Millimeter	Hand	Complain	Skills
Remarks	Bush	Rebuke	Clear	Telecommunications	Facility	Sharpnel	Automatic	Confirmation	Teach

Table 1. This table shows the top-5 similar words to the words in bold found using cosine similarity on 300-dimensional NNLM and Glove embeddings. We extracted the NNLM embeddings from a FNNLM trained on a 12M broadcast news data set with a heldout set perplexity of 144.9, whereas we trained the Glove embeddings on the Gigaword corpus. The differences in the top-5 lists from the two embeddings highlights their diversity.

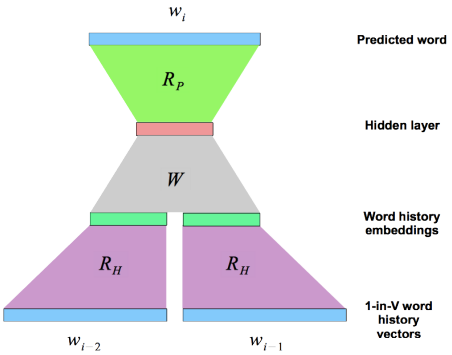

Fig. 1. A simple 3-gram FNNLM takes two 1-in-V word history vectors as input, converts them to their D-dimensional embeddings, passes them through a feed-forward NN with a K-neuron hidden layer, and predicts a V-dimensional PDF for the next word.

word *history* vectors $\mathbf{w}_{i-2}$ and $\mathbf{w}_{i-1}$ to their D-dimensional continuous representation through multiplication with a $V \times D$ embedding matrix $\mathbf{R}_H$. This forms the input

$$\mathbf{x}_i = [\mathbf{w}_{i-2}\mathbf{R}_H \quad \mathbf{w}_{i-1}\mathbf{R}_H] \tag{3}$$

to a neural network with one hidden layer. Let

$$\mathbf{h}_i = \sigma(\mathbf{x}_i\mathbf{W}) \tag{4}$$

be the $1 \times H$ vector at the output of the hidden layer and σ denote the point-wise sigmoid function. The NNLM then multiplies this vector with a $H \times V$ matrix $\mathbf{R}_P$ and computes the conditional probability mass function of the next word,

$$P(\mathbf{w}_i|\mathbf{w}_{i-2}, \mathbf{w}_{i-1}) = s(\mathbf{h}_i\mathbf{R}_P) \tag{5}$$

where s is the V-way soft-max function.

We now describe the semantic word embedding (SWE) FNNLM architecture. Let $\mathbf{G}$ be the $V \times D_G$ semantic word embedding matrix obtained using any algorithm such as word2vec or Glove. The neural network in Figure 2 incorporates semantic embeddings at two stages - as new input features to the FNNLM and as additional weights connecting the hidden to the output layer. The new architecture thus includes the following two modifications:

$$\textit{Feature concatenation:} \quad \mathbf{R}_H^G = [\mathbf{G} \quad \mathbf{R}_H] \quad \text{and} \tag{6}$$

$$\textit{Weight expansion:} \quad \mathbf{R}_P^G = [\mathbf{G} \quad \mathbf{R}_P^T]^T . \tag{7}$$

Concatenating the embedding matrix $\mathbf{G}$ to the word history embedding matrix $\mathbf{R_H}$ is intuitive and is akin to feature stream combination. However, we additionally view the weights of the hidden-to-output layer connections as a word embedding matrix $\mathbf{R}_P$. Hence, we also expand the hidden layer by the semantic embedding size D_G and add new connections between these new hidden neurons and the output layer with weight matrix $\mathbf{G}^T$. Our experiments illustrate that doing both feature concatenation and weight expansion using the semantic embedding matrix $\mathbf{G}$ gives significantly lower perplexity compared with performing only feature concatenation.

In addition to modifying the NNLM architecture, we also force the gradients of the log-likelihood training objective function with respect to the semantic embedding matrix $\mathbf{G}$ to be 0. This means that the pre-trained semantic embeddings are not updated during NNLM training. First, this prevents the risk of these semantic embeddings becoming less diverse with respect to $\mathbf{R}_H$ and $\mathbf{R}_P$ as training proceeds. Second, this also keeps the total number of unknown model parameters only slightly greater than a vanilla NNLM due to a larger input to hidden layer weight matrix $\mathbf{W}$. Our experiments showed that updating the semantic embeddings jointly with the NNLM parameters did not significantly improve held-out perplexity.

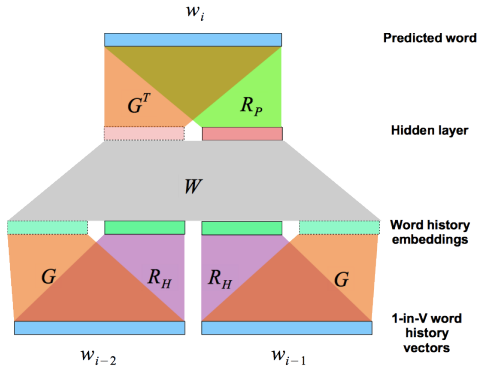

Fig. 2. A feedforward 3-gram SWE-FNNLM concatenates the D_G-dimensional Glove semantic embeddings $\mathbf{G}$ to the history and prediction word embeddings $\mathbf{R_H}$ and $\mathbf{R_P}$.

3. EXPERIMENTS AND RESULTS

We first briefly present the details of our Glove implementation and the data set used for training these semantic embeddings.

3.1. Glove Embeddings

We implemented Glove in Theano [17] to take advantage of the speed-up offered by graphical processing units (GPUs) and used AdaGrad [18] for stochastic gradient-based mini-batch training. We trained Glove on the English Gigaword corpus [19] that consists

of newswire text data collected over several years by the Linguistic Data Consortium at the University of Pennsylvania. The corpus size was approximately 1.9B word tokens after cleaning and we only considered the top 413K words in our model. The Gigaword corpus was a suitable choice because of its focus on news domain data instead of generic data sets such as Wikipedia or Common Crawl.

We used a symmetric window size of 10 words for constructing the word co-occurrence matrix. The resulting matrix was sparse with only 494M non-zero entries out of 170B total entries. Sparsity of the word co-occurrence matrix determines the training speed of Glove because word pairs not co-occurring in the training data receive a weight of 0 in the objective function. We set the embedding dimension to 300 and used the value of the objective function on 1% of the total word co-occurrence counts for stopping the AdaGrad optimization. The model took 2.5 days to train on a single GPU with 10 passes through the training data. The resulting word embeddings gave a 63% classification accuracy on the Google analogy task [1,2]. We next describe our ASR experiments on broadcast news using FNNLMs.

3.2. ASR Experiments on Broadcast News Using FNNLMs

3.2.1. LM Perplexity

We performed experiments on an English broadcast news task and trained all LMs on a 12M word subset of the 350M word training text used in the 2007 IBM GALE speech transcription system [20]. We limited the vocabulary of all LMs to the 20K most frequent words in the corpus. The heldout set consists of reference transcriptions from the *dev04* test sets [20]. We used our IBM NNLM library for training FNNLMs in Theano, and implemented the SWE-FNNLMs in it. We trained all FNNLMs on Nvidia Tesla K40 GPUs using mini-batch based stochastic gradient descent (SGD). Our FNNLM model used a 300-dimensional embedding and 500 hidden neurons with hyperbolic tangent activation function. We picked these topologies based on our prior experience with broadcast news ASR [12]. The SWE-FNNLMs used 300-dimensional semantic embeddings.

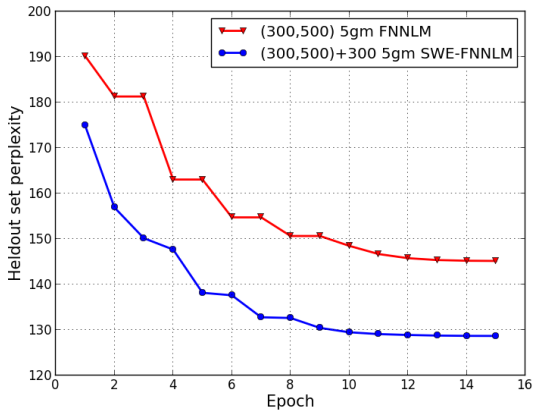

Fig. 3. This figure compares the broadcast news heldout set perplexity using the baseline FNNLM and SWE-FNNLM as training proceeds.

Table 2 lists heldout set perplexities for several NNLMs and a Kneser-Ney (KN) smoothed 6-gm LM trained on the 12M word broadcast news corpus. We note that the SWE-FNNLM offers significant improvement in perplexity over the corresponding baseline FNNLMs. We would like to point out that the SWE-FNNLMs have only slightly greater number of parameters compared to the baseline FNNLMs. The (300,500) 5gm NNLM in Table 2 has roughly 16.6M parameters while (300,500)+300 5gm SWE-NNLM has around 17.9M parameters. This is because we do not update the 300-dimensional semantic embeddings during FNNLM training.

Single LM	**Perplexity**	**% Reduction**
6gm KN	144.5	-
5gm FNNLM (300,500)	144.9	-0.3%
5gm SWE-FNNLM (300,500)+300	*128.5*	*11.1%*
Interpolated LM	**Perplexity**	**% Reduction**
6gm KN + FNNLM	118.3	18.1%
6gm KN + SWE-FNNLM	*111.8*	*22.6%*
6gm KN + FNNLM + SWE-FNNLM	*109.6*	*24.2%*

Table 2. This table presents heldout set perplexities for various LMs on broadcast news. A (D,H) NNLM contains a D-dimensional projection layer and H hidden neurons. The SWE-FNNLM used 300-dimensional semantic word embeddings as denoted by "+300". We observe that the SWE-FNNLM has significantly lower perplexity than the corresponding vanilla FNNLM.

We also observe that SWE-FNNLM model has significantly lower perplexity than the vanilla FNNLM even after interpolation with the 6gm KN LM. Interpolating all three LMs gives minor improvement in perplexity over KN + SWE-FNNLM because the SWE-NNLM already includes a word representation tuned on the LM training data in addition to the semantic embeddings.

As an additional analysis, we compared the per-epoch heldout set log-perplexity for the baseline FNNLM and SWE-FNNLM. Figure 3 shows that SWE-FNNLM maintains a significantly lower heldout perplexity compared with the baseline FNNLM at each epoch. In fact, the SWE-FNNLM perplexity at epoch 5 becomes better than the final converged perplexity of the baseline FNNLM obtained at epoch 15. We attribute this to the pre-trained Glove embeddings in the SWE-FNNLM that already carry significant discriminative information to predict the next word.

The next section presents ASR lattice rescoring experiments on the broadcast news data set using FNNLMs.

3.2.2. ASR Lattice Rescoring

We now discuss our lattice rescoring setup for FNNLMs on broadcast news. Our first ASR system (AM1) used a deep convolutional neural network (CNN)-based hybrid ASR system trained on 400 hours of broadcast news data [21,22]. The decoder vocabulary contained 80K words and the baseline LM was a linear interpolation of 4gm Kneser-Ney (KN)-smoothed LMs trained on different data sources comprising a 350M word corpus. We generated lattices on the *rt04* test set containing 4 hours of speech data with a pruned version of the baseline LM, and then rescored these lattices with the unpruned baseline LM. This resulted in a baseline word error rate (WER) of 11.3%. We then rescored the resulting lattices with

various FNNLMs linearly interpolated with the 4gm KN LM. For comparison, we also present rescoring results with a second acoustic model (AM2) that is a discriminatively trained speaker-adaptive AM trained on 430 hours of broadcast news [14]. The baseline WER of this system is 13.0%.

AM1		
LM	**WER**	**% Reduction**
4gm KN	11.3%	-
4gm KN + FNNLM	11.0%	2.6%
4gm KN + SWE-FNNLM	10.7%	5.3%
AM2		
LM	**WER**	**% Reduction**
4gm KN	13.0%	-
4gm KN + FNNLM	12.7%	2.3%
4gm KN + SWE-FNNLM	12.6%	3.0%

Table 3. This table presents the test set WERs using LMs obtained on the *rt04* broadcast news data set after lattice rescoring using two AMs. Both FNNLMs were trained on a 12M subset of the 350M training set and used a vocabulary short-list consisting of the most frequent 20K words. The baseline 4gm KN LM was trained on the 350M training set with a vocabulary of 80K words.

Table 3 presents the resulting WERs obtained after lattice rescoring. The SWE-FNNLM gives a significant reduction ($p < 0.01$) in WER over the vanilla FNNLM for both AM1 and AM2.

3.3. ASR Experiments on Broadcast News Using RNNLMs

3.3.1. LM Perplexity

We studied the perplexity of RNNLMs on exactly the same setup as for FNNLMs. We used back-propagation through time with mini-batch updates for training the RNNLMs in Theano. Our RNNLMs used a 180-dimensional embedding and 500 hidden neurons with hyperbolic tangent activation function. We fixed the sequence length of the RNNLM to 18 words which is the average length of a sentence in the training data set, and used 8 sequences in each mini-batch. More details of the RNNLM training are provided in [11]. Table 4 shows the heldout set perplexities for various RNNLMs. We observe significant improvements by using SWE-RNNLM.

3.3.2. ASR N-best Rescoring

This section presents ongoing ASR N-best rescoring experiments using SWE-RNNLMs. We would like to point out that these initial results use AM2 from Table 3 and are on the *dev04* set.

Our N-best rescoring setup using RNNLMs is similar to the setup used in [11]. We generated 50-best lists by decoding the development *dev04* set with the baseline acoustic model and 4-gm KN LM trained on 350M words. We generated 50-best sentence-level probabilities from the RNNLM/SWE-RNNLM, and log-linearly interpolated these with the baseline LM score and the acoustic model score. We used the heldout set for tuning the interpolation weights using the simplex algorithm from the SRILM toolkit [23]. Table 5 shows the test set WERs obtained after 50-best rescoring.

We observe that while the RNNLM gives a small improvement over the baseline 4gm KN LM, we do not see a significant additional gain by using a SWE-RNNLM. We hypothesize multiple reasons for this result. First, the embeddings learned by a vanilla RNNLM already incorporate long-range context. This makes them less diverse with respect to the semantic word embeddings estimated using a window of 10 words. Second, a 50-best rescoring framework might not be ideal for seeing the performance benefit of using a SWE-RNNLM. Third, the training of a RNNLM is much more sensitive to hyper-parameters compared to a FNNLM. The training might easily be swayed astray by issues such as dynamic range of the semantic word embeddings. We are currently investigating these issues.

Single LM	**Perplexity**	**% Reduction**
6gm KN	144.5	-
RNNLM (180,500)	145.8	-0.9%
SWE-RNNLM (180,500)+300	*132.1*	*8.6%*
Interpolated LM	**Perplexity**	**% Reduction**
6gm KN + RNNLM	115.7	19.9%
6gm KN + SWE-RNNLM	*111.1*	*23.1%*
6gm KN + RNNLM + SWE-RNNLM	*108.7*	*24.8%*

Table 4. This table presents heldout set perplexities for various LMs on broadcast news. We observe that the SWE-RNNLM has significantly lower perplexity than the corresponding vanilla RNNLM.

LM	**WER**	**% Reduction**
4gm KN	14.2%	-
4gm KN + RNNLM	13.9%	2.1%
4gm KN + SWE-RNNLM	13.9%	2.1%

Table 5. This table gives the test set WERs obtained using RNNLMs on the *dev04* broadcast news data set after 50-best rescoring.

4. CONCLUSION AND FUTURE WORK

This paper presents models for incorporating semantic word embeddings (SWEs) in neural network language models (NNLMs). We hypothesize that the resulting SWE-NNLMs are better than the vanilla NNLMs because they use word embeddings that better capture semantic information than the word embedding learned in a NNLM. Our experiments show significant improvements in both perplexity and word error rate (WER) on a broadcast news ASR system for the case of feed-forward NNLMs (FNNLMs).

Our motivation in extending the use of embeddings to recurrent networks was to explore any additional temporal information that could be captured via the encoded co-occurence information in the embeddings, in addition to the temporal context in the RNNs. Although, the SWE-RNNLM yields significant improvements in perplexity, we did not observe any significant reduction in WER. We plan to explore this further in the future.

5. REFERENCES

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proc. ICLR*,

2013.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," *Proc. EMNLP*, vol. 12, pp. 1532–1543, 2014.

[3] J. Turian, L. Ratinov, Y. Bengio, and D. Roth, "A preliminary evaluation of word representations for named-entity recognition," in *Proc. NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, 2009, pp. 1–8.

[4] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proc. ACL*, 2010, pp. 384–394.

[5] W. Y. Zou, R. Socher, D. M. Cer, and C. D. Manning, "Bilingual word embeddings for phrase-based machine translation.," in *Proc. EMNLP*, 2013, pp. 1393–1398.

[6] I. Sutskever, O. Vinyals, and Q. V. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

[7] S. Gouws, Y. Bengio, and G. Corrado, "Bilbowa: Fast bilingual distributed representations without word alignments," *arXiv preprint arXiv:1410.2455*, 2014.

[8] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proc. ACL*, 2014, vol. 1, pp. 1555–1565.

[9] M. Bansal, K. Gimpel, and K. Livescu, "Tailoring continuous word representations for dependency parsing," in *Proc. ACL*, 2014.

[10] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Cernockỳ, "Empirical evaluation and combination of advanced language modeling techniques.," in *Proc. INTERSPEECH*, 2011, number 1, pp. 605–608.

[11] E. Arisoy, A. Sethy, B. Ramabhadran, and S. Chen, "Bidirectional recurrent neural network language models for automatic speech recognition," in *Proc. ICASSP*, 2015, pp. 5421–5425.

[12] E. Arisoy, S. F. Chen, B. Ramabhadran, and A. Sethy, "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 184–192, 2014.

[13] G. Saon, H. J. Kuo, S. Rennie, and M. Picheny, "The IBM 2015 English Conversational Telephone Speech Recognition System," *arXiv preprint arXiv:1505.05899*, 2015.

[14] X. Chen, X. Liu, M. J. F. Gales, and P. C. Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," *Proc. ICASSP*, 2015.

[15] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Proc. INTERSPEECH*, 2014, pp. 661–665.

[16] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.

[17] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proc. SciPy*, June 2010.

[18] T. Tieleman and G. Hinton, "Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[19] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda, "English Gigaword fifth edition," *Linguistic Data Consortium, LDC2011T07*, 2011.

[20] S. F. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig, "Advances in speech transcription at IBM under the DARPA EARS program," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1596–1608, 2006.

[21] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. ICASSP*, 2013, pp. 8614–8618.

[22] T. N. Sainath, B. Kingsbury, A. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for LVCSR," in *Proc. ASRU*, 2013, pp. 315–320.

[23] A. Stolcke, "SRILM-an extensible language modeling toolkit," in *Proc. INTERSPEECH*, 2002.