INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

An investigation of recurrent neural network architectures using word embeddings for phrase break prediction

Anandaswarup Vadapalli, Suryakanth V Gangashetty

Speech and Vision Lab, IIIT Hyderabad
anandaswarup.vadapalli@research.iiit.ac.in, svg@iiit.ac.in

Abstract

This paper presents our investigations of recurrent neural networks (RNNs) for the phrase break prediction task. With the advent of deep learning, there have been attempts to apply deep neural networks (DNNs) to phrase break prediction. While deep neural networks are able to effectively capture dependencies across features, they lack the ability to capture long-term relations that are spread over time. On the other hand, RNNs are able to capture long-term temporal relations and thus are better suited for tasks where sequences have to be modeled. We model the phrase break prediction task as a sequence labeling task, and show by means of experimental results that RNNs perform better at phrase break prediction as compared to conventional DNN systems.

Index Terms: phrase break prediction, RNN, word embeddings, text to speech synthesis,

1. Introduction

Phrasing is a crucial step in speech synthesis. It breaks long utterances into meaningful units of information and makes the speech more understandable. More importantly, in the context of speech synthesis, phrase breaks are often the first step for other models of prosody, such as intonation prediction and duration modeling. Any errors made in the initial phrasing step are propagated to other downstream prosody models, ultimately resulting in synthetic speech that is unnatural and difficult to understand.

Phrase breaks are manifested in the speech signal in the form of several acoustic cues like pauses as well as relative changes in the intonation and duration of syllables. Acoustic cues such as pre-pausal lengthening of rhyme, speaking rate, breaths, boundary tones and glottalization also play a role in indicating phrase breaks in speech [1–3]. However, representing these non-pause acoustic cues in terms of features is not easy and not well understood [4]. In this paper we restrict ourselves only to pauses in speech, and limit our phrase break models to predicting the locations of pauses while synthesizing speech. This is the approach followed in [5–10].

Traditionally, phrase break prediction has been achieved by using machine learning models like regression trees or HMMs in conjunction with data labeled with linguistic classes (such as part-of-speech (POS) tags, phrase structure etc.) [11–17]. A lot of effort has also been directed towards unsupervised methods of inducing word representations, which can be used as surrogates for POS tags/linguistic classes, in the phrase break prediction task [7, 18, 19].

With the advent of deep learning as well as techniques for deriving/inducing continuous dimensional representations of words, called 'word embeddings', there have been efforts to apply these techniques to phrase break prediction [6, 8–10, 20]. Continuing our work in [9], in this paper we investigate the use of recurrent neural network (RNN) models in conjunction with word embeddings, for phrase break prediction.

The remainder of the paper is organized as follows : Section 2 discusses the motivations for our study. In Section 3 we describe the RNNs used in our work. Section 4 presents our experimental results, with a discussion on the same. Section 5 concludes this work and Section 6 provides possible directions for future work.

2. Motivation for our study

Based on the success of deep learning in speech recognition [21], deep learning techniques have been applied to other areas of spoken language processing, including language modeling, text-to-speech synthesis (TTS), intent determination, sentiment detection / analysis, semantic utterance classification tasks in SLU etc. While deep neural networks are able to effectively capture dependencies across features, they lack the ability to capture long-term relations that are spread over time. On the other hand, RNNs are able to capture long-term temporal relations and thus are better suited for tasks where sequences have to be modeled.

The phrase break prediction task can be described as follows : *"Given an utterance (represented as a sequence of words) to be synthesized by the TTS, each word is annotated with either a break (B) or no-break (NB) tag, indicating whether a pause should be inserted after that word or not"*. This task is similar to several tasks in language processing such as slot filling in spoken language understanding (SLU), part-of-speech tagging etc., and thus is ideal to be modeled as a sequence labeling task.

Following the work of [22,23], where the authors use RNNs to improve results on the slot filling task of spoken language understanding over traditional approaches, we propose to use RNNs for phrase break prediction.

This work has been primarily motivated by the following questions:

1. Does capturing long-term temporal dependencies using a recurrent architecture improve the performance of phrase break prediction, as compared to short-term dependencies captured using a DNN architecture?

2. What is the best recurrent architecture (in terms of both performance as well as computational complexity) for this task?

We experiment our approach on audiobook data and present results which attempt to answer the questions raised above.

http://dx.doi.org/10.21437/Interspeech.2016-885

3. Using RNNs for phrase break prediction

3.1. RNN inputs

Similar to our earlier work in [9] we use word embeddings which are randomly initialized and jointly trained with the network on the task at hand, as inputs to the RNN.

3.2. RNN architectures

In this work we use two different RNN architectures : (1) Elman RNN [24] and (2) Long short term memory (LSTM) [25]. For the sake of completeness we describe both architectures below.

3.2.1. Elman RNN

An Elman RNN is a simple RNN with hidden layer recurrent connections. The Elman RNN architecture is illustrated in Figure 1, where it is unrolled in time across three consecutive word inputs. This architecture consists of an input layer, a hidden layer with recurrent connections (shown as dashed lines in the figure) and an output layer. Each layer represents a set of neurons, and the layers are connected with weights denoted by the matrices U, W and V.

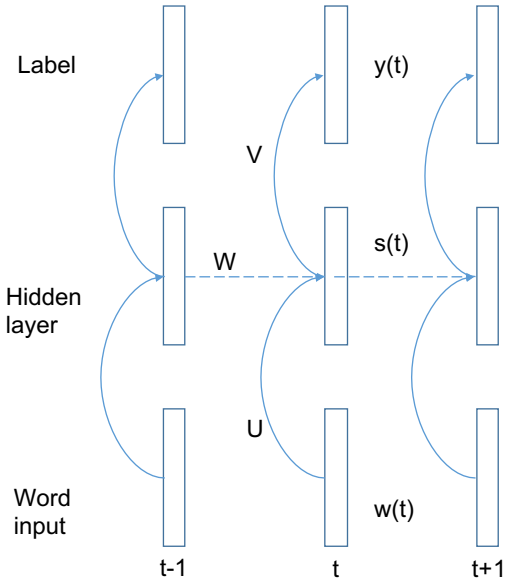

Figure 1: Elman RNN architecture for phrase break prediction

The dynamics of an Elman RNN can be described by the following equations:

$$s(t) = f(Uw(t) + Ws(t-1)) \quad (1)$$
$$y(t) = g(Vs(t)) \quad (2)$$

where, U represents the input to hidden layer weights, W represents the recurrent weights in hidden layer and V represents the hidden to output layer weights. $w(t)$, $s(t)$ and $y(t)$ are the input, hidden layer state and output respectively at time t, while $s(t-1)$ is the hidden layer state at time $t-1$. $f(.)$ and $g(.)$ are the activations of the hidden layer and output layer respectively. In this work, we set $f(.)$ to be the $tanh$ nonlinearity and $g(.)$ to be $softmax$ applied at each time-step. As mentioned in 3.1 the input $w(t)$ to the RNN at time t, is the word embedding corresponding to the word at time t.

The RNN is trained using standard backpropagation through time to maximize the data conditional likelihood :

$$\prod_t P(y(t)|w(1)\cdots w(t)) \quad (3)$$

The probability distribution is strictly a function of the hidden layer activations, which in turn depend only on the inputs (and their own past values). Thus, the most likely sequence of phrase break labels can be computed as :

$$y_t^* = \arg\max P(y(t)|w(1)\cdots w(t)) \quad (4)$$

This has the advantage of being online and very simple; it is not necessary to do a viterbi search over the possible labelings to find the optimum.

3.2.2. LSTM

We use the standard vanilla LSTM implementation including forget gate and peephole connections described in [25]. Figure 2 is reproduced from [25] for the sake of completeness and ease of understanding the equations describing the dynamics of the LSTM.

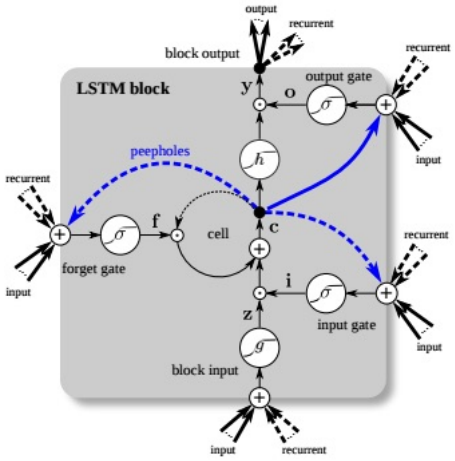

Figure 2: LSTM architecture

The forward pass equations of the LSTM are (from [25]) :

$$z_t = f(W_z x_t + R_z h_{t-1} + b_z) \quad (5)$$
$$i_t = \sigma(W_i x_t + R_i h_{t-1} + p_i \odot c_{t-1} + b_i) \quad (6)$$
$$f_t = \sigma(W_f x_t + R_f h_{t-1} + p_f \odot c_{t-1} + b_f) \quad (7)$$
$$c_t = i_t \odot z_t + f_t \odot c_{t-1} \quad (8)$$
$$o_t = \sigma(W_o x_t + R_o h_{t-1} + p_o \odot c_t + b_o) \quad (9)$$
$$h_t = o_t \odot g(c_t) \quad (10)$$
$$y_t = g(W_{oh} h_t + b_{oh}) \quad (11)$$

where $f(.)$, $g(.)$ are the $tanh$ nonlinearity, and $g(.)$ is $softmax$ applied at each time-step. W_z, W_i, W_f, W_o and R_z , R_i , R_f, R_o are weights from the input and previous state at unit-input, input, forget, output gates respectively. W_{oh} is the hidden layer (LSTM unit) to output layer weights and b_{oh} is the

output layer bias. p_i, p_f and p_o are the peep-hole connections and $\odot$ represents element-wise multiplication. x_t, h_t and y_t are the input, hidden state and output at time t. As in the case of Elman RNN, the input x_t to the LSTM at time t, is the word embedding corresponding to the word at time t.

Similar to the Elman RNN, the LSTM is trained using standard backpropagation through time to maximize the data conditional likelihood :

$$\prod_t P(y_t|x_1 \cdots x_t) \tag{12}$$

and the most likely sequence of phrase break labels can be computed as :

$$y_t^* = \arg\max P(y_t|x_1 \cdots x_t) \tag{13}$$

3.3. Objective function

In all our architectures, we apply a $softmax$ at each time step in the output layer. The objective function used in this case to train the RNN is categorical crossentropy :

$$L = -\sum_c y_c \cdot \log(s_\theta(x)_c) \tag{14}$$

where, c is the number of classes, y_c is the correct value for class c and $s_\theta(x)_c$ is the score assigned by the model to class c, given the current data point x.

3.4. Learning Methods

Training RNNs with long inputs (as is in the case of utterances longer than 10 words) is tricky if the updates are done after each word in the utterance. This is because the predictions at the current word are made with model parameters that are no longer current, and the sequence of predictions does not correspond to one that could be performed with a fixed parameter set. For example, let us consider an utterance with 20 words. If we want to perform an update at the 13th word of the utterance with an unidirectional RNN model, we would have to recompute all the values from the beginning of the utterance in order to get the prediction consistent with the current model parameters.

In order to solve this problem, we use the same technique as in [23], and perform mini-batch gradient descent, with exactly one utterance per mini-batch. For a given utterance, we perform one pass that computes the mean loss for that utterance and then perform a gradient update for that entire utterance. This also enables us to handle utterances of differing lengths.

4. Experimental results

4.1. Data used

We use speech and audio data taken from the following three audiobooks : Emma by Jane Austen (EM), Mansfield Park by Jane Austen (MP) and Pride and Prejudice by Jane Austen (PAP), in the LibriVox database[1]. Each book was recorded by a volunteer and the style of the corpus is "audio book".

For each audiobook dataset, the speech and the corresponding text were segmented using the INTERSLICE tool [26], and a CLUSTERGEN [27] voice was built within the Festival [28] and Festvox [29] frameworks.

[1]http://www.librivox.org

4.1.1. Annotation of phrase breaks

As we do not have a dataset with hand annotated phrase breaks, we derive the location of phrase breaks from the data. Using the Festival utterance structures generated during the CLUSTERGEN voice building process, we extract the location of the breaks introduced by the speaker. We mark all break relations labeled B and BB as breaks and all other break relations as non-breaks. At the end of the process each word in the text corpus is labeled as a break or non-break depending upon whether or not a break occurs after that word.

4.2. Experiment 1 : DNNs vs RNNs

In this experiment, we compare the performances of DNN, Elman RNN and LSTM models on the phrase break prediction task. We used the Keras[2] deep learning library, running the Theano [30, 31] backend to build our models. For all models the dimension of word embeddings was fixed at 50. All the recurrent models used a single hidden layer of 200 units, while the DNN model used two hidden layers of 200 units each. Table 1 shows the parameter as well as hyperparameter values used in our models. All the code as well as data used in this work are available online[3].

Table 2 shows the results of the DNN, RNN and LSTM systems on the phrase break prediction task. We report our results in terms of the F-Measure [34] which is defined as the harmonic mean of precision and recall.

An examination of Table 2 shows that both the Elman RNN and LSTM models significantly outperform the DNN model, while there is no significant difference in the performance of the Elman RNN as compared to the LSTM model. This experimentally proves that capturing long-term temporal dependencies using a recurrent architecture improves the performance of phrase break prediction, as compared to a DNN architecture, and answers the first question raised in Section 2.

Table 2 also shows that for the phrase break prediction task, there is no significant difference between the performance of the Elman RNN model and the LSTM model. Elman RNN models have fewer parameters as compared to LSTMs and so require fewer computational resources to train and run. As there is no significant difference in the performance of Elman RNNs as compared to LSTMs; training a simple Elman RNN is enough for phrase break prediction and there is no necessity to train computationally expensive LSTM models. This observation answers the second question raised in Section 2.

4.3. Experiment 2 : Effect of varying the size of the RNN hidden layer

In this experiment we study the effect of varying the size of the RNN hidden layer on the performance of the model for phrase break prediction. We use the Elman RNN model from Experiment 1 (Section 4.2) with all parameter and hyperparameter values unchanged, except for the hidden (recurrent) layer size. We vary the hidden layer size and compute the performance, in terms of the F-measure. Table 3 shows the results of this experiment.

An examination of the results in Table 3 shows that there is no significant difference in the performance on phrase break prediction when the Elman RNN hidden layer size is varied. This is a somewhat surprising result, as one would expect at

[2]http://keras.io

[3]https://goo.gl/5ODbeh

Table 1: Parameter and hyperparameter values used in our models

Parameters	DNN	Elman RNN	LSTM
Word embedding dimension	50	50	50
hidden layer size	200 : 200	200	200
activation function	$tanh()$	$tanh()$	$tanh()$
minibatch size	32	1	1
initial learning rate	0.01	0.01	0.01
Nesterov momentum	0.9	0.3	0.3
weight initialization	glorot uniform [32]	glorot uniform [32]	glorot uniform [32]
inner cells initialization	–	orthogonal [33]	orthogonal [33]
forget gate bias	–	–	1

Table 2: Performance (in terms of the F-Measure) of the DNN, RNN and LSTM systems on the phrase break prediction task

	F-Measure		
Audiobook	DNN	RNN	LSTM
EM	85.92	92.35	92.55
MP	85.56	92.03	92.17
PAP	85.98	92.55	92.82

least 1 - 2 % increase in the performance corresponding to the increase in the RNN hidden layer size.

Table 3: Effect of varying the hidden (recurrent) layer size on the performance (in terms of the F-measure) of the Elman RNN model on the phrase break prediction task

	Hidden layer size		
Audiobook	200	500	1000
EM	92.35	92.40	92.68
MP	92.03	92.28	92.43
PAP	92.55	92.22	92.70

4.4. Experiment 3 : Effect of varying the dimension of word embeddings

In this experiment we study the effect of varying the word embedding dimension on the performance of the model for phrase break prediction. We use the Elman RNN model from Experiment 1 (Section 4.2) with all the parameters and hyperparemeters unchanged, except for the word embedding dimension. We vary the word embedding dimension and compute the performance, in terms of the F-measure. Table 4 shows the results of this experiment.

An examination of the results in Table 4 shows that there is no significant difference in the performance on phrase break prediction when the word embedding dimension is varied. As with the result obtained in Experiment 2 (Section 4.3) this is a somewhat surprising and counterintuitive result, as one would expect at least 1 - 2 % increase in the performance corresponding to the increase in the word embedding dimensions.

Table 4: Effect of varying the word embedding dimension on the performance (in terms of the F-measure) of the Elman RNN model on the phrase break prediction task

	Word embedding dimension			
Audiobook	50	100	150	200
EM	92.35	92.62	92.74	92.75
MP	92.03	92.69	92.25	92.70
PAP	92.55	92.77	92.75	92.73

5. Conclusions

In this paper we model phrase break prediction as a sequence labeling task, and show by means of experimental results that recurrent models perform significantly better at phrase break prediction as compared to DNN models. We also show that there is no significant difference between the performance of the Elman RNN model and the LSTM model, and that training a simple Elman RNN is enough and there is no necessity to train computationally expensive LSTM models to perform phrase break prediction. We also perform experiments to study the effects of varying the RNN hidden layer dimension and word embedding dimension on the performance of the model for phrase break prediction. The results from these experiments are somewhat surprising and counterintuitive, and require further analysis and study.

6. Scope for future work

In this paper, while our experimental results have shown that recurrent models perform significantly better in terms of the objective F-Measure, on phrase break prediction as compared to DNNs, these results have raised several questions:

1. What specific additional information is captured by the RNN enabling it to have a significantly better performance (in terms of the F-Measure) as compared to a DNN?
2. How much does the improvement in performance (in terms of the objective F-measure) obtained by using RNNs as compared to DNNs translate into a better perceptual listening experience for the end user of the TTS system?

We wish to explore these questions in future work. In addition to these, as part of our future work, we also wish to further study and analyze the results obtained in Sections 4.3 & 4.4 (Experiments 2 & 3).

7. References

[1] C. Wightman, S. Shattuck-Hufnagel, M. Ostendorf, and P. J. Price, "Segmental durations in the vicinity of prosodic phrase boundaries," *Journal Acoustical Society of America*, vol. 91, no. 3, pp. 1707–1717, 1992.

[2] L. Redi and S. Shattuck-Hufnagel, "Variation in realization of glottalization in normal speakers," *Journal of Phonetics*, vol. 29, pp. 407–429, 2001.

[3] H. Kim, T. Yoon, J. Cole, and M. Hasegawa-Johnson, "Acoustic differentiation of L- and L-L% in switchboard and radio news speech," in *Proceedings of Speech Prosody*, Dresden, 2006.

[4] K. Prahallad, E. V. Raghavendra, and A. W. Black, "Learning speaker-specific phrase breaks for text-to-speech systems," in *Proceedings of 7th ISCA Speech Synthesis Workshop (SSW7)*, Kyoto, Japan, 2010, pp. 148–153.

[5] A. Parlikar and A. W. Black, "Minimum error rate training for phrasing in speech synthesis," in *Procedings of 8th ISCA Speech Synthesis Workshop (SSW8)*, Barcelona, Spain, September 2013, pp. 13–16.

[6] O. Watts, A. Stan, Y. Mamiya, A. Suni, J. M. Burgos, and J. M. Montero, "The Simple4All entry to the blizzard challenge 2013," in *Proceedings of the 2013 Blizzard Challenge Workshop*, August 2013.

[7] A. Vadapalli, P. Bhaskararao, and K. Prahallad, "Significance of word-terminal syllables for prediction of phrase breaks in Text-to-Speech systems in Indian languages," in *Proceedings of 8th ISCA Speech Synthesis Workshop (SSW8)*, Barcelona, Spain, September 2013, pp. 189–194.

[8] O. Watts, J. Yamagishi, and S. King, "Unsupervised continuous-valued word features for phrase-break prediction without a part-of-speech tagger," in *Proceedings of Interspeech*, Florence, Italy, August 2011, pp. 2157–2160.

[9] A. Vadapalli and K. Prahallad, "Learning continuous-valued word representations for phrase break prediction," in *15th Annual Conference of the International Speech Communication Association (INTERSPEECH 2014)*, Singapore, September 2014, pp. 41–45.

[10] O. Watts, S. Gangireddy, J. Yamagishi, S. King, S. Renals, A. Stan, and M. Giurgiu, "Neural net word representations for phrase-break prediction without a part of speech tagger," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 2599–2603.

[11] A. Parlikar and A. W. Black, "A grammar based approach to style specific phrase prediction," in *Proceedings of Interspeech*, Florence, Italy, August 2011, pp. 2149–2152.

[12] M. Wang and J. Hirschberg, "Automatic classification of intonational phrase boundaries," *Computer Speech and Language*, vol. 6, pp. 175–196, 1992.

[13] E. Navas, I. Hernez, and I. Sainz, "Evaluation of automatic break insertion for an agglutinative and inflected language," *Speech Communication*, vol. 50, no. 11-12, pp. 888–899, 2008.

[14] P. Taylor and A. W. Black, "Assigning phrase breaks from part-of-speech sequences," *Computer Speech and Language*, vol. 12, pp. 99–117, 1998.

[15] H. Schmid and M. Atterer, "New statistical methods for phrase break prediction," in *Proceedings of 20th International conference on Computational Linguistics,COLING '04*, Geneva, Switzerland, 2004.

[16] A. Bonafonte and P. Agüero, "Phrase break prediction using a finite state transducer," in *Proceedings of 11th International Workshop on Advances in Speech Technology*, 2004.

[17] B. Busser, W. Daelemans, and A. van den Bosch, "Predicting phrase breaks with memory-based learning," in *Proceedings of 4th ISCA Speech Synthesis Workshop*, 2001.

[18] A. Parlikar and A. W. Black, "Data-driven phrasing for speech synthesis in low-resource langauges," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan, March 2012.

[19] N. S. Krishna and H. A. Murthy, "A new prosodic phrasing model for Indian language Telugu," in *INTERSPEECH-2004-ICSLP*, vol. 1, Oct 6-11 2004, pp. 793–796.

[20] O. Watts, "Unsupervised learning for text-to-speech synthesis," Ph.D. dissertation, University of Edinburgh, 2012.

[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[22] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE Transactions Audio, Speech and Language Processing*, vol. 23, no. 3, pp. 530–539, Mar. 2015. [Online]. Available: http://dx.doi.org/10.1109/TASLP.2014.2383614

[23] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *14th Annual Conference of the International Speech Communication Association (INTERSPEECH 2013), Lyon, France, August 25-29, 2013*, 2013, pp. 3771–3775.

[24] J. L. Elman, "Finding structure in time," *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.

[25] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *CoRR*, vol. abs/1503.04069, 2015. [Online]. Available: http://arxiv.org/abs/1503.04069

[26] K. Prahallad, "Automatic building of synthetic voices from audio books," Ph.D. dissertation, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2010.

[27] A. W. Black, "CLUSTERGEN: A statistical parametric synthesizer using trajectory modeling," in *Proceedings of Interspeech*, Pittsburgh, USA, 2006.

[28] A. W. Black and P. Taylor, "The festival speech synthesis system," Human communication research center, University of Edinburgh, Tech. Rep., January 1997, available Online: http://www.cstr.ed.ac.uk/projects/festival.

[29] A. W. Black and K. Lenzo, "Building voices in the festival speech synthesis system," 2002, available Online: http://festvox.org/bsv.

[30] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[31] 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 *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010.

[32] Y. Bengio and X. Glorot, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of AISTATS 2010*, vol. 9, May 2010, pp. 249–256.

[33] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *CoRR*, vol. abs/1312.6120, 2013. [Online]. Available: http://arxiv.org/abs/1312.6120

[34] C. J. van Rijsbergen, *Information Retrieval*. Butterworth, 1979.