Lexicon-Free Speech Recognition with Recurrent Neural Networks

Pius Friesch friesch@kth.se
Martin Hwasser hwasser@kth.se
Wojciech Kryściński wkr@kth.se
Amund Vedal amund@kth.se

Abstract

In this work we implement and analyze an end-to-end speech recognition system. The system is entirely based on neural network models, using a character-level language model combined with the RNN-based acoustic model and CTC decoder. Finally, we train and evaluate our model on the LibriSpeech ASR dataset and compare it to a the results of our main reference paper.

1 Introduction

A complete automatic speech recognition (ASR) system traditionally consisted of several components, made to either quantize interesting parts of raw sound input (feature extraction), dividing the features into (sub-)phoneme classes (acoustic model), and decoding the classified sequences using lexical and language models, resulting in hypotheses of which words were spoken in the recorded utterances.

According to [17], such a system could require thousands of lines of code and a thorough understanding of each part to make improvements or conduct research. This made it hard for researchers to build their own versions for experiments. It was also challenging to optimize and specialize (one language at a time), and yielded so-called n-best lists of words rather than single-word hypotheses.

1.1 Related work

Previous approaches to included deep neural networks (DNN) into speech recognition have been made by replacing Gaussian mixture models (GMM) for acoustic modeling [16, 19, 12, 14, 5]. Even though this brought significant improvements compared to the traditional ASR pipeline, they still depend on laboriously engineered systems. Additionally, these systems still need training data which is labeled up to phoneme level. A method to approach this problem has been introduced as Connectionist Temporal Classification (CTC) [8] to label unsegmented speech data without the need of phoneme alignment. In addition, demonstrated by Eyben et al. [6], Bi-directional Long Short-Term Memory (BLSTM) networks are capable of recognizing graphemes without the need for phonemisation dictionaries and language model based techniques. Classic feed-forward neural networks are not suitable for transcribing connected time series such as speech. Recurrent neural networks such as Long Short-Term Memory (LSTM) [6], where one or more of the hidden layers is connected to itself, are able to store information over long periods of time by continuously updating an internal state vector. The network is trained directly on the text transcripts using the CTC loss function. This means that no pronunciation lexicon or phonetic representation is needed. Graves and Jaitly [9] also showed successful results using deep BLSTMs with the CTC loss function mapping audio input features to character sequences. These were re-ranked at the word-level from a HMM-DNN baseline, producing competitive results on the Wall Street Journal corpus [17].

Hannun et al. [13], on the other hand, proposed a method purely based on basic RNNs, which are easier to compute while not providing the same long term memory as LSTMs. Amodei et al. [1] extends on this work using Gated Recurrent Units (GRU) [4], which speeds up calculations at a slight performance cost compared to LSTMs.

Maas et al. [17] iterates on these approaches but at the character-level. As such, the need for a lexicon is eliminated and out-of-vocabulary (OOV) words and fragments can be transcribed. Doing this, the authors are able to accurately and efficiently perform transcription using only the deep bi-directional recurrent neural network (DBRNN) and a character-level language model (CLM). The CTC loss function trained the neural network to do maximum likelihood of letter sequences given acoustic features as input. This work was on par with high performance HMM-GMM systems while having a much less complex pipeline. However, a more complex, attention-based model with significantly stronger results was also presented in [3] in the same year.

2 Method

In this project we attempt replicate the system architecture proposed by Maas et al. [17]. The authors built a complete speech recognition system consisting of three components: a neural network for mapping speech from MFCC components to characters, a character-level language model and a beam search decoding algorithm. A graph of the pipeline is shown in figure 1. Each of these components will be described in more detail in the following subsections.

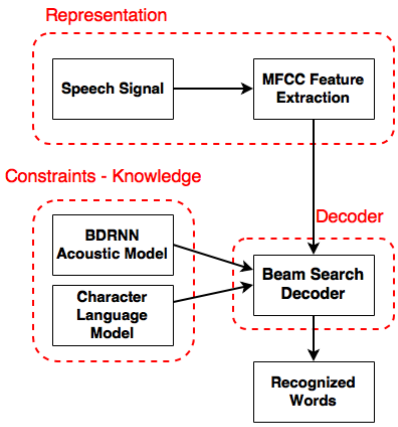


Figure 1: Complete SSR system pipeline proposed by authors. Graph inspired by course lecture slides.

2.1 Mel-Frequency Cepstral Coefficients

The shape of the human vocal tract manifests itself as an envelope of the short time power spectrum through which the sounds we utter are filtered. The task of Mel-Frequency Cepstral Coefficients (MFCCs) is to represent this envelope artificially [24], so that recorded sounds filtered through them can yield a response that can be recognized as phonemes. We use MFCC responses as the input to out acoustic model, as was done in [17].

2.2 Acoustic modeling

Acoustic modeling in Maas et al. [17] was done using a Deep Bidirectional Recurrent Neural Network (DBRNN). The architecture was a mix of feed-forward and recurrent layers.

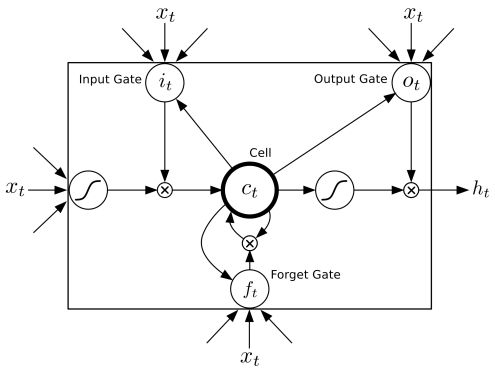
Recurrent neural networks are a class of artificial neural networks that allow recurrent connections in the architecture. This extension enables units to build an internal representation of the recently seen data and combine it with new inputs. This trait becomes particularly important when working with sequential data like speech, since the information at each time step is conditionally dependent on what came before. Unfortunately the RNN unit has two issues that reduce its practical value. It can take into account only previous context and it can only model short-time dependencies [11, 9, 22].

In order to overcome the mentioned problem, bidirectional networks were introduced. Schuster et al. [21] proposed to split the units into two independent parts, one working with previous context and the other with future context. It is important to note that the input to the cell is distributed both to the

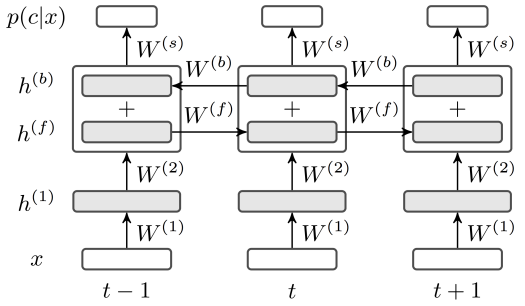
forward and backward states and the output from those states is combined as the output of the cell, but there is no communication between the states within one cell. Using this architecture allows the network to discover more complex dependencies in the data, at the cost of not being able to operate in on-line mode [21], since the full sequence is required.

Similarly to regular neural networks, multiple layers of recurrent units can be stacked on top of each other with the output of one layer being the input of the next. This allows the network to build increasingly complex representations of the input data. It is both possible to build fully recurrent deep architectures or mix feed-forward and recurrent layers in one architecture [10, 17]. An example DBRNN is shown in Figure 2b.

As acoustic modeling is essentially a classification task, and the output layer of the DBRNN is a Softmax function. In this setting, the model outputs the distribution $p(c \mid X)$ over the possible output labels (which is the set of legal characters in a language). Usually the set of allowed characters is small enough that a regular Softmax layer can be used (as opposed to models with large number of outputs where techniques such as Hierarchical Softmax must be used).



(a) Schematics of an LSTM cell. Image from [9]



(b) Schematics of a DBRNN unrolled over 3 time steps. Image from [17]

Figure 2: Components of the acoustic model

CTC objective function

In [17], the acoustic model was trained using the Connectionist Temporal Classification (CTC) objective function. CTC allows RNNs to be trained using datasets with unaligned input-output pairs. This objective function expects a specific form of output from the network in which there is a single output unit for each possible label and one additional unit corresponding to a null emission (*blank*). To allow for learning using unaligned data, all possible alignments leading to the target output must be taken into consideration when computing the loss value. In order to do so, the CTC function defines a "collapsing" function $\kappa(\cdot)$ that removes all blanks and neighboring repeated characters from a given character sequence. Using the inverse of this function all possible alignments can be generated and averaged out to yield the final loss. This is shown in eq. 1 where $C_W = \{C : \kappa(C) = W\}$. It is worth noting that there is an assumption of conditional independence of labels C given the input sequence X. This procedure is efficiently implemented using dynamic-programming. CTC is a differentiable function so gradient-based training, such as backpropagation through time (BPTT) can be used with this objective function [8, 9, 17].

$$\begin{aligned} \mathcal{L}_{CTC}(X, W) &= \sum_{C_W} p(C \mid X) \\ &= \sum_{C_W} \prod_{t=1}^{|X|} p(c_t \mid X) \end{aligned} \tag{1}$$

2.3 Language modeling

In Maas et al. [17] a character language model created using a neural network is proposed.

The objective of language modeling is to learn the joint probability function of sequences of words in a language [2]. This is another case of modeling sequential data. Usually this task is done on the word-level, but this is not a necessity. In case of Character Language Models (CLM) the task is to predict the next character given characters in the previous context. Formally this can be written as $p(x_{t+1} \mid x_{\leq t})$, where x is the text sequence, and $x_{\leq t}$ is the context seen up to time t. Creating models on the character-level instead of word-level has two benefits. It allows to easily work with Out-of-Vocabulary (OOV) sequences, sequence fragments and disfluencies, which is particularly important in the context of speech recognition. It also allows to significantly limit the modeled space, since CLMs only learn dependencies between legal characters of a language which is a set a few orders of magnitude smaller than the word vocabulary of a language [9, 18, 23].

Creating neural language models can be done with different architectures. Maas et al. [17] tests two architectures for this task, a Deep (feed-forward) Neural Network (DNN) and a Recurrent Neural Network.

The major deficiency of DNNs when compared to RNNs is that they are limited to working with fixed-length context spans that must be specified at the creation of the network and cannot be easily modified at later stages. In contrast, RNNs do not limit the size of the context and in theory allow the use of arbitrary-length contexts (but this is highly dependent on the type of RNN unit, as explained in Section 2.2).

2.4 Naive decoding

The simplest way of decoding the acoustic model outputs, chosen as baseline for [17], is to use a greedy approach directly on the DBRNN output. This procedure greedily picks the character with highest probability at each time step. The great benefit of this produce is that it is computationally inexpensive. The process of decoding is shown in eq. 2.

$$\begin{aligned} W^* = \underset{W}{\operatorname{argmax}}\, p(W \mid X) &\approx \kappa(\underset{C}{\operatorname{argmax}}\, p(C \mid X)) \\ &= \kappa(\underset{C}{\operatorname{argmax}} \prod_{t=1}^{|X|} p(c_t \mid X)) \end{aligned} \tag{2}$$

Beam search decoding

A more complex way of decoding the acoustic model is to use outputs from the DBRNN and merge them with the independently built character level language model. These two models can be combined using the beam search algorithm, explained thoroughly in [9, 17]. Beam search is a general search algorithm based on Breadth-First search. At each step the algorithm expands only k most promising branches, where k is called the width of the beam.

In comparison to naive decoding, this technique assigns probabilities to final hypotheses after considering (integrating out) all possible character sequences that are consistent with the hypothesis under the κ collapsing function. As the algorithm progresses through the output sequence, at each step it builds a list of k best hypothesis considering the outputs from the DBRNN and the language model.

3 Experiments

3.1 Dataset

We used the LibriSpeech ASR corpus [20], which was extracted from English audio-books under the LibriVox project. It contains 1000 hours of speech sampled at 16kHz. The speakers are both male and female (equally balanced), and most of them have an US American accent. The utterances are aligned and labeled by sentence or sub-sentence, which means this dataset would not be suitable if phoneme labels were necessary. The corpus was made freely available by the original author. LibriSpeech comes in 100, 360 and 500 hour packages, where we used the smallest one.

3.1.1 Preprocessing

The input speech samples were divided into frames of length *25ms* with *10ms* shift. These frames were further preprocessed by extracting basic MFCC features with the zeroth coefficient. This step yielded a 13 dimensional feature vector for each frame of the utterance. No particular normalization of these features were executed.

Transcripts were transformed into a "one-hot" representation on a character-level, and extended with an additional "blank" symbol required by the CTC objective function. This yielded a 29 dimensional sparse vector for each character in the transcription. As we use CTC, the no word or phoneme alignment is necessary.

The corpus used for language model training made available by the authors in preprocessed form. We transformed the text to upper-case, and removed all special characters. The set of available characters thus consisted of all upper-case letters of the English language, plus the 'space' and 'apostrophe' characters.

3.2 Implementation

We followed the architecture described in [17]. The acoustic model has five hidden layers with the third layer containing a bi-directional, recurrent connection. The model was served data with a context window of $\pm$ 10 frames. Each of the hidden layers had 1824 hidden units. Each layer consisted off basic recurrent units.

The character language model used two layers with recurrent connections and 512 units, inspired by [17]. Here, we attempt a slight variation, using two LSTM layers rather than basic RNNs to see how these architectures compare. This CLM still operates on one input character at a time.

We used TensorFlow and Python to build the networks.

3.3 Evaluation

To gain additional insight into how each component is performing, we evaluate both the system as a whole and each part separately. For the acoustic model, we use the two metrics word error rate (WER) and character error rate (CER), which are both commonly used in the field. For the language model we use the three metrics CER, perplexity and loss (bits-per-character), as suggested in [7]. Then, the output of the whole system is evaluated with WER, making it directly comparable with systems of other papers. Performance tests are conducted on previously unseen test sets.

We decided to use a different dataset than Maas et al. [17] in our experiments, meaning that our results aren't directly comparable with those presented in the original paper. In the mentioned paper, the authors measure the acoustic model by itself before testing the whole system, which we believe is a good way to assess the parts of the system separately. We chose this dataset because it has a reasonable size and quality, is freely available and the proposed model has not been evaluated on it by Maas et al. [17].

4 Results

Acoustic model To train our acoustic model, we used *train-clean* (360h), *dev-clean* (5h) and *test-clean* (5h) subsets of LibriSpeech [20] for training/validation/testing respectively. Our final error rates can be seen in Table 3, along with some examples of model predictions in Table 1.

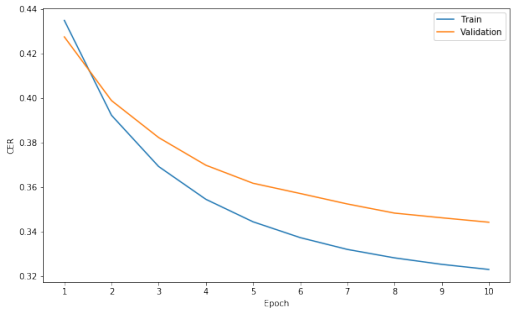


Figure 3: Comparing the character error rate for training and validation sets, acoustic model. Corresponds to "CTC no LM" in table 3

Truth:	the cuisine is of the best and the chefs rank at the top of their art
Pred:	the qwmsnn is of the basta an theshu ringd ot the op o er ur
Truth:	to the left hand i turned with that reliance with which the little child runs to his mother when he has fear or when he is afflicted
Pred:	to the lafh penda tornd with that e lnans wih which the lita te rone ww as mother wend e hus fear we wan hes of catde
Truth:	little pearl who was as greatly pleased with the gleaming armour as she had been with the glittering frontispiece of the house spent some time looking into the polished mirror of the breastplate
Pred:	litle parl a was s gratly plase woit the glaming armer as she had ben wath the glitering fntos pece of the has pand some tine igin to he polise mere of the brest plaht

Table 1: Examples of predictions of our trained acoustic model "CTC no LM"

Language model For the language model, our initial aim was to run tests using 1M lines (20M words) for training similar to the 31M word corpus in [17]. Training with such a massive corpus in combination with using the more complex LSTM-cell rather than basic RNN turned out infeasible, however, due to computational restraints. We instead limited ourselves to two experiments using approximately 50K lines (1M words) and 100K lines (2M words), all from LibriSpeech, to train our language model.

Test results			
Model	CER	Loss	Perplexity
LSTM-2, 50K lines	0.3979	1.284	3.611
LSTM-2, 100K lines	0.3976	1.271	3.564

Table 2: Comparing results from training on 50K vs 100K lines of text. Examples of lines are shown as "True" in Table 3

Model (ours)	CER	LS		
CTC no LM	33.7	84.9		
CTC+LSTM-2 (50k)	(no time)	(no time)		
CTC+LSTM-2 (100k)	(no time)	(no time)		
Model (by Maas et al)	CER	EV	CH	SWBD
CTC no LM	27.7	47.1	56.1	38.0
CTC+RNN	24.9	33.0	41.7	24.2
CTC+RNN-3	24.7	30.8	40.2	21.4

Table 3: Evaluation of DBRNN trained using CTC by decoding with character-level language models. We report WER on the LibriSpeech dataset (LS), loosely compared to word error rate achieved by Maas et al. [17] on the datasets Eval2000 (EV), Switchboard (SWBD) and CallHome (CH) subsets (lower is better). Unfortunately, the computation to obtain results for the whole model.

Beam Search Decoding with Language Model

Truth:	the cuisine is of the best and the chefs rank at the top of their art
CTC no LM:	the cgrsin is of the best and de shes wreing to the popof der art
CTC+LSTM-2 (50k):	the con is of the best and the charing to the top of their art
Truth:	the three friends were astounded
CTC no LM:	the tre frands wer stoundid
CTC+LSTM-2 (50k):	the three friends were standing at
Truth:	of course these are not the entire menus but of all the well prepared dishes these are their best
CTC no LM:	if course o besern at the intireman ersm bt a al the wal pr herditios thes arn their best
CTC+LSTM-2 (50k):	the course of bear at the entire manner but all the properdities these are their best

Table 4: Examples before and after beam search decoding on the CTC probabilities in combination with the language model (beam width 10)

5 Discussion and Conclusions

CER and WER of the acoustic model Comparing to our reference main paper [17], we note how our CER of 33.7 on the acoustic model is slightly higher than their 27.7. What's more alarming is that the WER of our acoustic model is very high at 84.9. We suspect that this may be due to the characteristics of the LibriSpeech dataset, the style of which is very literary and non-colloquial, with many long and unusual words. Moreover, by training using the CTC loss function, the model only considers characters and not words. As such, the model has not learned how to spell correctly, but rather learned how to map acoustic features to characters. As reported in the original paper, the big improvement of combining the acoustic model with the language model can mostly be seen in the WER [17] (see "CTC no LM" in Table 3). The authors reported large reductions in WER only after adding the language model. We believe there's room for improvement of the acoustic model, first and foremost by training on more data, however even on the 360 hour dataset, training one epoch took more than 9 hours on one NVIDIA Tesla K80.

Language model When training language model LSTM, an interesting observation was that the model plateaued relatively early (around 2000 iterations), with very small improvements after. A possible reason for this could be that the learning rate was set too high, or that the model was not complex enough to model all the dependencies in the training dataset. A solution could be using early stopping as regularization technique, as mentioned in [10]. This problem requires a more thorough analysis and could be a topic for future projects.

Beam search decoding To improve the WER of the output of the acoustic model, we implemented the beam search algorithm as proposed in Maas et al. [17]. As seen in Table 4 some misspellings where fixed by the decoding and it has the potential to lower WER significantly. However, these samples took from several minutes to several hours for longer utterances to compute. Even with a lower beam width as reported in Maas et al. [17]. Especially, longer utterances take substantial time to generate probabilities for the next character from the language model. Therefore, it is not feasible to compute the WER on the test set with our implementation. We assume that serious engineering efforts need to be applied to significantly improve computation time, which would be out of the scope of the project. Thus we weren't able to report the WER after decoding of the test set. We take from this that Beam search decoding is a expensive step in a inference pipeline and thus might be a bottleneck for online systems.

Data size vs. language model complexity A challenge we faced when trying to expand on the work from Maas et al. [17] was that of datasize vs model complexity and computational time. We originally chose the to expand the language model of the authors using an LSTM rather than basic RNN to try to improve results, but quickly understood that this simplification probably was intentional from the beginning. LSTM's are painfully slow to train, forcing to (for example) rely on hyperparameter values suggested by other authors and our own intuition rather than doing a thorough search ourselves. We also had to set certain architecture-independent hyperparameters, such as batch size, in a way that would allow us to fully utilize the GPU's computational power. We found that the speed of learning could change up to three-fold (3x) depending on the hyperparameter values. In the case of smaller batches, not the computation but the time to transfer the data between main memory and GPU memory became one bottleneck, while another was the size of the data. In the end, we were forced to lower the data size from the 20M words we started with (close to Maas' 31M) to 2M words (100K lines) for our CLM. In comparing our results to that of the author, and comparing the CER for 50K to 100K lines, we question whether the more complex LSTM should have been sacrificed for larger data in this case.

Implementation problems When using Keras we encountered two major problems that made us fall back to a pure TensorFlow implementation. The first problem was implementing the CTC loss function needed for the acoustic model. This became problematic in Keras as there was no straightforward way of doing this without implementing a dummy loss function and passing the labels as inputs in the network in order to decode the outputs and compute loss in a so-called Lambda layer. Furthermore, the workaround of passing the labels along with the inputs had other implications such as not being able to save/restore the model due to the graph validation enforced by Keras.

The second problem arose when training Language Models. Following the architecture proposed by Maas, our requirement was the the LM would take a single character at a time. Unfortunately, the high level implementation of RNN units in Keras only allowed us to train models with a fixed context size. Multiple workarounds to this problem were tested but none of them were successful.

Our conclusion is, that Keras seems to be a great library, for fast prototyping of new models. However, when functionality is needed that is not provided by Keras, adding it ourselves negates the that ease and can become more difficult than a implementation from the ground up with Tensorflow.

Choice of dataset and usefulness of our system In hindsight, we ponder what impact the choice of dataset has on the ASR system for different tasks. For example, since we trained both acoustic and language models using LibriSpeech, our system is likely to do a good job at transcribing recitations of books similar to the ones present in LibriSpeech dataset, with a rich vocabulary (see Tables 1), low noise and very few mistakes. However, the usefulness of such a system is highly application depended. There may be very few occasions where we wish to transcribe speech with very rich vocabulary, and as such, choosing to train a speech recognizer on readings of audio books, rather than conversational datasets as in [17], may not make sense. Of course, such philosophical lamentations aren't intended to report a bad result, since we built the model for the sake of learning, but rather to reflect on how the usefulness of an ASR system depends on the choice of dataset used for training.

References

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. Deep speech 2: End-to-

end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[3] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell. *CoRR*, abs/1508.01211, 2015.

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

[5] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[6] Florian Eyben, Martin Wöllmer, Björn W. Schuller, and Alex Graves. From speech to letters - using a novel neural network architecture for grapheme based ASR. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2009, Merano/Meran, Italy, December 13-17, 2009*, pages 376–380, 2009.

[7] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[8] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 369–376, 2006.

[9] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1764–1772, 2014.

[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

[11] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[12] Roger Grosse, Rajat Raina, Helen Kwong, and Andrew Y Ng. Shift-invariance sparse coding for audio classification. *arXiv preprint arXiv:1206.5241*, 2012.

[13] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[14] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[15] Sepp Hochreiterand and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[16] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[17] Andrew L. Maas, Ziang Xie, Dan Jurafsky, and Andrew Y. Ng. Lexicon-free conversational speech recognition with neural networks. In *Proceedings the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2015.

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

[19] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[21] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45(11):2673–2681, 1997.

[22] Ilya Sutskever. *Training Recurrent Neural Networks*. PhD thesis, University of Toronto, 2013.

[23] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[24] Vibha Tiwari. Mfcc and its applications in speaker recognition. *International journal on emerging technologies*, 1(1):19–22, 2010.