ON TRAINING THE RECURRENT NEURAL NETWORK ENCODER-DECODER FOR LARGE VOCABULARY END-TO-END SPEECH RECOGNITION

Liang Lu[1], *Xingxing Zhang*[2], *and Steve Renals*[1]

[1]Centre for Speech Technology Research, University of Edinburgh, Edinburgh, UK
[2]Institute for Language, Cognition and Computation, University of Edinburgh, Edinburgh, UK
{liang.lu, x.zhang, s.renals}@ed.ac.uk

ABSTRACT

Recently, there has been an increasing interest in end-to-end speech recognition using neural networks, with no reliance on hidden Markov models (HMMs) for sequence modelling as in the standard hybrid framework. The recurrent neural network (RNN) encoder-decoder is such a model, performing sequence to sequence mapping without any predefined alignment. This model first transforms the input sequence into a fixed length vector representation, from which the decoder recovers the output sequence. In this paper, we extend our previous work on this model for large vocabulary end-to-end speech recognition. We first present a more effective stochastic gradient decent (SGD) learning rate schedule that can significantly improve the recognition accuracy. We then extend the decoder with long memory by introducing another recurrent layer that performs implicit language modelling. Finally, we demonstrate that using multiple recurrent layers in the encoder can reduce the word error rate. Our experiments were carried out on the Switchboard corpus using a training set of around 300 hours of transcribed audio data, and we have achieved significantly higher recognition accuracy, thereby reduced the gap compared to the hybrid baseline.

Index Terms: end-to-end speech recognition, deep neural networks, recurrent neural networks, encoder-decoder.

1. INTRODUCTION

The neural network/hidden Markov model (NN/HMM) hybrid approaches have redefined state-of-the-art speech recognition [1, 2, 3]. In this framework, a neural network is used to estimate the posterior probabilities of HMM states, while the main sequential modelling is carried out by the HMM, incorporating context-dependent phone models, pronunciation models, and language models (LMs). The past few years have seen significant advancements in speech recognition based on this hybrid architecture including using different neural network architectures [4, 5, 6], sequence training [7, 8, 9] and speaker adaptation [10, 11, 12]. However, there has been relatively little focus on the fundamentals of the hybrid architecture. The main advantage of the hybrid approach is that it factorizes the speech recognition problem into several relatively independent subtasks based on a few assumptions and approximations; each module deals with only one of the sub-tasks, thus simplifying the objective. For instance, using neural networks to classify each acoustic frame into one of the HMM states based on the conditional independence assumption is much simpler compared to classifying a set of variable length sequences directly. However, the cost of this divide-and-conquer strategy is that it is difficult to optimise all the modules jointly.

Recently, there has been an increasing interest in end-to-end speech recognition using neural networks without using HMM sequence modelling. One approach is based on the connectionist temporal classifier (CTC) that uses a recurrent neural network (RNN) for feature extraction [13], and competitive results have been achieved on a few tasks [14, 15, 16, 17]. CTC does not rely on a prior alignment between input and output sequences, but integrates over all possible alignments during the model training. The alignment is computed by the forward-backward algorithm as part of the model training. The key difference compared to HMMs is that the output labels can be letters or phonemes instead of the HMM states, and it introduces the `blank` label to discard those frames that are not informative or are noisy when computing the optimal output sequence. However, similar to HMMs, CTC still predicts labels for every frame, and relies on the conditional independence assumption.

Another approach is based on the RNN encoder-decoder which was firstly proposed for machine translation [18, 19], and has been applied to image captioning [20], as well as speech recognition [21, 22, 23, 24]. This model transforms the input sequence of variable length into a fixed dimensional vector representation using the RNN encoder, and the RNN decoder recovers the output sequence from this vector representation. Unlike CTC, this model does not require the alignments between the input and output tokens, and it does not rely on the conditional independence assumption. This model has achieved competitive phoneme recognition accuracy on the TIMIT database [21], and word recognition accuracy on WSJ [23]. Recently, Chan et al [24] obtained good results on the large scale Google Voice Search task. Previously, we investigated this approach for large vocabulary speech recognition on the Switchboard corpus [22], where we focused on architectural and speedup issues for this model. In this paper, we present training strategies that can significantly reduce the word error rate (WER). In particular, we show that improved scheduling of the SGD learning rates can significantly improve the recognition accuracy, and extending the memory of the RNN decoder can further reduce the WER. Finally, using multiple recurrent layers in the encoder can result in a higher recognition accuracy.

2. RNN ENCODER-DECODER WITH ATTENTION

2.1. The model

For sequence to sequence learning, the RNN encoder-decoder directly computes the conditional probability of the output sequence given the input sequence without assuming a fixed alignment. The

This work is funded by the EPSRC Programme Grant EP/I031022/1, Natural Speech Technology (NST). The NST research data collection may be accessed at http://datashare.is.ed.ac.uk/handle/10283/786.

key idea is to introduce the *context vector* obtained from the RNN encoder as a representation of the input sequence, so that the conditional probability can be approximated as

$$P(y_1, \ldots, y_O|\mathbf{x}_1, \ldots, \mathbf{x}_T) \approx \prod_{o=1}^{O} P(y_o|y_1, \ldots, y_{o-1}, \mathbf{c}_o). \quad (1)$$

Note that the context vector $\mathbf{c}_o$ (cf. section 2.2) is updated for each output token y_o. For speech recognition, $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ is usually a sequence of acoustic feature vectors, while $\{y_1, \ldots, y_O\}$ is usually a sequence of class indices corresponding to the output units such as phonemes, letters, or words, etc. In the decoder, the posterior probability of y_o is computed using the *softmax* function after a recurrent hidden layer which takes both the embedding vector of the previous token $\mathbf{y}_{o-1}$ and the current context vector $\mathbf{c}_o$ as inputs, i.e.

$$P(y_o|y_1, \ldots, y_{o-1}, \mathbf{c}_o) = g(\mathbf{s}_o, \mathbf{c}_o) \quad (2)$$

$$\mathbf{s}_o = f(\mathbf{y}_{o-1}, \mathbf{s}_{o-1}, \mathbf{c}_o), \quad (3)$$

where g denotes the *softmax* function, and f refers to the recurrent function; $\mathbf{y}_{o-1}$ is a continuous representation of y_{o-1}, which is obtained from an embedding matrix. The recurrent layer in the decoder performs implicit language modelling, which explains why the encoder-decoder can work reasonably well without any language model. The function of the recurrent hidden state $\mathbf{s}_o$ is to remember the current decoding state, and fuse the information from $\mathbf{y}_{o-1}$ and $\mathbf{c}_o$. As shown in section 2.2, $\mathbf{s}_o$ is also used to compute the attention weights for the context vector. In Eq. (2), it is possible to remove $\mathbf{c}_o$ from the inputs, however, we obtained lower recognition accuracy in our preliminary experiments (results are not given in this paper), indicating that $\mathbf{s}_o$ cannot capture all the information from $\mathbf{c}_o$ by one recurrent hidden layer.

2.2. Attention-based scheme

For the encoder-decoder, it is possible to use a global fixed context vector $\mathbf{c}$ in Eq. (1) as in the machine translation task [25, 18]. However, for long input sequences as in speech recognition, this approach usually does not work, especially when the dimension of $\mathbf{c}$ is relatively small. The more effective approach is to dynamically compute the context vector $\mathbf{c}_o$ given the current decoding state $\mathbf{s}_o$ by the attention-based scheme [19]. More precisely, $\mathbf{c}_o$ is obtained as

$$\mathbf{c}_o = \sum_t \alpha_{ot}\mathbf{h}_t \quad (4)$$

where α_{ot} is the attention weight with the constraint as $\alpha_{ot} \in [0, 1]$ and $\sum_t \alpha_{ot} = 1$. $\mathbf{h}_t$ denote the hidden state of the encoder RNN which transforms the input feature as

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}) \quad (5)$$

In this paper, we always use the bidirectional RNN [26] in the encoder, and we then concatenate the forward and backward hidden state as $\mathbf{h}_t = \left(\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t\right)$. Since the conventional RNN only has limited power to capture the sequential information due to the vanishing gradient problem, in this work, we use the gated recurrent units (GRU) [18] in all the recurrent layers.

In Eq. (4), the weight α_{ot} is computed by a learned alignment model for each $\mathbf{c}_o$, which is implemented as a neural network such that

$$\alpha_{ot} = \frac{\exp(e_{ot})}{\sum_{t'} \exp(e_{ot'})} \quad (6)$$

$$e_{ot} = \mathbf{v}^\top \tanh(\mathbf{W}\mathbf{s}_{o-1} + \mathbf{U}\mathbf{h}_t), \quad (7)$$

where e_{ot} is the relevance score of each hidden representation $\mathbf{h}_t$ with respect to the previous hidden state of RNN decoder $\mathbf{s}_{o-1}$. $\mathbf{W}$ and $\mathbf{U}$ are weight matrices, and $\mathbf{v}$ is a vector so that the output of e_{ot} is a scalar.

Since all the functions used in the encoder-decoder are differentiable, the model can be trained using SGD by maximising the average conditional log-likelihood over the training set as

$$\hat{\mathcal{M}} = \arg\max_{\mathcal{M}} \frac{1}{N} \sum_{n=1}^{N} \log P(y_1^n, \ldots, y_O^n|\mathbf{x}_1^n, \ldots, \mathbf{x}_T^n, \mathcal{M}),$$

where $\mathcal{M}$ denotes the set of model parameters, and N is the number of training utterances. Unlike the hybrid model using the feed-forward neural networks, this model is more complex in using different types of neural components. It leads to the problem that the dynamic range of the gradients for some weights varies significantly, which makes manually tuning the SGD learning rates challenging. Previously, we used the Adadelta algorithm [27] to aumatically tune the learning rate. However, it is still sub-optimal, because when we train the recurrent nets, we clip the gradients as in [28] to avoid the gradient explosion, but this makes the Adadelta algorithm unstable. This issue will be further investigated in section 3.

2.3. Long memory decoder

As discussed before, the hidden state $\mathbf{s}_o$ in Eq. (2) has multiple functions, which may not be well realised by just using one recurrent layer. In this work, we study the approach to improving the capacity of $\mathbf{s}_o$ by feeding in more informative features, which is again learned by a recurrent net. More precisely, we modify the decoder as

$$P(y_o|y_1, \ldots, y_{o-1}, \mathbf{c}_o) = g(\mathbf{s}_o, \mathbf{c}_o) \quad (8)$$

$$\mathbf{s}_o = f(\mathbf{p}_o, \mathbf{s}_{o-1}, \mathbf{c}_o) \quad (9)$$

$$\mathbf{p}_o = f(\mathbf{y}_{o-1}, \mathbf{p}_{o-1}) \quad (10)$$

where we introduce another recurrent layer as in Eq. (10) which only does the implicit language modelling and remembers the decoding history. We then replace $\mathbf{y}_{o-1}$ by the recurrent hidden state $\mathbf{p}_o$ as in Eq. (9) so that hidden state $\mathbf{s}_o$ can receive more information of the decoding history from the input features. This decoder is expected to have longer memory, and may work better without the language model. It is also possible to feed $\mathbf{p}_o$ into the softmax layer as

$$P(y_o|y_1, \ldots, y_{o-1}, \mathbf{c}_o) = g(\mathbf{s}_o, \mathbf{c}_o, \mathbf{p}_o). \quad (11)$$

However, the role of $\mathbf{p}_o$ may be overweighted in this approach, therefor it may not be suitable for the conversational speech recognition task investigated in this paper, where word sequences are less predictable.

2.4. Comparison to CTC

CTC [13] does not directly compute the conditional probability of the output sequence given the input sequence. Instead, it computes the posterior probability of the label l_t for every frame $\mathbf{x}_t$ similar to the hybrid model. In the case of using bi-directional RNN to transform the acoustic feature $\mathbf{x}_t$, this probability is computed using only the softmax function without recurrent layer as

$$P(l_t|\mathbf{x}_t) = g(\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t). \quad (12)$$

Since the classification is performed on the per-frame level, CTC needs to compute the alignment between the acoustic frames and

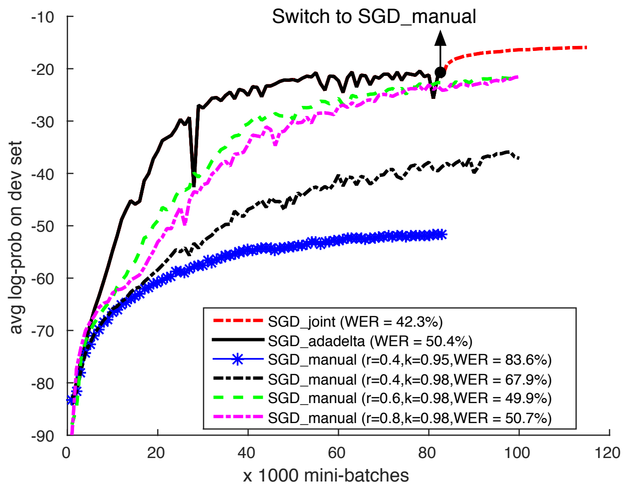

Fig. 1. Comparison of scheduling the SGD learning rate for training the RNN encoder-decoder. The results were obtained by using 24 dimensional FBANK static features. r denotes the initial learning rate and k is the learning rate decay factor. Here, the learning rate was decayed for every 1000 mini-batches by a small factor.

output labels as part of the model training, and as observed in [14], it may be sensitive to the initial alignment. In order to guarantee that the lengths of the input and output sequences are the same, CTC replicates the output labels so that a consecutive frames may correspond to the same label. It then applies a rule to collapse the replicated labels during the decoding, while the RNN decoder does not have this problem. Finally, CTC still requires the independence assumption of the acoustic frames, which is not required in the RNN encoder-decoder approach.

3. RESULTS AND DISCUSSION

3.1. System setup

We report results using the Switchboard corpus of [29] released by LDC with the catalog number as LDC97S62. It has a 300 hour training set, and we show separate results for the Callhome English (CHE) and Switchboard (SWB) evaluation sets. The vocabulary size is around 30,000, and the number of training utterances in Switchboard is 192,701. In this work, we evaluated both the mel-frequency cepstral coefficients (MFCCs) and log-mel filterbanks (FBANKs) as acoustic features, which were obtained using the Kaldi toolkit [30]. In the frond-end, we performed the mean and variance normalisation on the per-speaker basis before we concatenating the features by a context window of ± 5 frames. Following our previous practice [22], we uniformly subsampled the spliced features for each utterance by a ratio of $1/3$, which significantly reduced the training time. It is interesting to see that subsampling was also applied in the CTC-based system which improved the recognition accuracy in [14]. In our experiments, the number of hidden units in the RNN encoder is 1000 unless specified otherwise, and the mini-batch size is 30 utterances.

3.2. SGD learning rate

Manually searching the SGD initial learning rate and the learning rate decay factor (referred to `SGD_manual`) is expensive for training with a large dataset. In addition, the hyper-parameters may depend on the type of features and model configurations. Previously, we applied the Adadelta algorithm [27] to automatically tune the learning rate [22]. However, we found that it did not yield the optimal solution similar to the observation in [21]. As discussed in Section 2.2, the Adadelta algorithm relies on the gradient to adjust the learning rate, however, in oder to avoid the gradient explosion problem, we clip the gradient and that makes the Adadelta algorithm unstable. To address this problem, [21] proposed an approach to fix the gradient before applying Adadelta. In this work, we applied the `SGD_joint` approach, namely, Adadelta followed by `SGD_manual`. More specifically, we first run the Adadelta algorithm until convergence, which usually takes around 10 - 15 epochs for our task. We then switched to `SGD_manual` with small initial learning rate (e.g., 0.01 - 0.02 in this work) for another few epochs to fine tune the model. As shown in Table 1, this approach we achieved significant WER reduction. Note that in these experiments, we used words as the output units in the softmax function in Eq. (2).

Table 1. Comparison of `SGD_adadelta` and `SGD_joint` to schedule the SGD learning rates.

SGD learning rate	Feature	CHE	SWB	Avg
SGD_adadelta [22]	MFCC	59.9	38.8	49.4
SGD_joint	MFCC	55.0	36.2	45.6
SGD_adadelta	FBANK	56.8	34.7	45.8
SGD_joint	FBANK	48.2	26.8	37.6
SGD_joint	FBANK(static)	52.2	31.8	42.1

We also compared two different type of acoustic features, i.e., 39 dimensional MFCCs and 45 dimensional FBANKs both with delta and delta-delta coefficients. Note that in both cases, we spliced the features with the context window of ± 5. Compared to the hybrid systems [31], we obtained a much larger gain by using FBANK features, possibly due to that transforming the features by discrete cosine transform (DCT) makes it more difficult for RNNs to discover the sequential patterns. Since RNN has strong ability in long dependency modelling, it is interesting to know if the dynamic features is still useful in this setup. Contrary to our expectation, we obtained significantly higher WER without the dynamic features in our experiment. In Figure 1 shows convergence of three systems with different SGD algorithms, where we increased the number of filter banks from 15 to 24. Again, `SGD_joint` achieved much better result compared `SGD_adadelta`, and it also converged much faster. However, we did not obtain better results by using larger number of filter banks. In the following experiments, we sticked to the 45 dimensional FBANKs with dynamic coefficients, and the `SGD_joint` optimisation algorithm.

3.3. Results of long memory decoder

We then evaluated the long memory decoder approach discussed in section 2.3. In our experiments, the number of hidden units in recurrent layer Eq. (10) was set to be 300. It is much smaller than the dimension of $\mathbf{c}_o$, which is 2000 with bidirectional RNNs. The intuition is to emphasise the role of the context vector in the decoder as Eq. (9). As shown in Table 2, the long memory decoder described as Eq. (8) - (10) improved the recognition accuracy by more than 1% absolute. However, the decoder defined as Eq. (11) did not work better. We suspect that the decoder may be biased toward the implicit language model. We then rescored the n-best list from the model using a 3-gram language model, which was trained on Switchboard and Fisher transcriptions using the KenLM tookit [32]. However, we only obtained small improvements. The size of the n-best list

Table 2. Results of language model rescoring and using long memory decoder. `LongMem1` is referred to Eq. (8), and `LongMem2` is referred to Eq. (11).

System	Output	CHE	SWB	Avg
EncDec no LM	word	48.2	26.8	37.6
EncDec + 3-gram rescoring	word	47.4	26.2	36.8
EncDec + LongMem1	word	46.5	26.3	36.4
+ 3-gram rescoring	word	**46.0**	**25.8**	**36.0**
EncDec + LongMem2	word	47.1	27.3	37.3
+ 3-gram rescoring	word	46.4	26.5	36.5
EncDec no LM	char	52.7	32.8	42.8
EncDec + 5-gram rescoring	char	51.9	32.6	42.3
EncDec + LongMem1	char	51.6	30.9	41.3
+ 5-gram rescoring	char	50.4	30.5	40.5

was 32, and similar to the observation in [24], increasing the size of the n-best list did not further reduce the WER.

Using word level output units is not optimal. It cannot generalise well to words that are unseen in the training set, and it may not work well for words of low frequency. Furthermore, for very large vocabulary tasks, the softmax layer will be big, which may slow down the model training. Alternative output units are phonemes or characters. In this work, we evaluated the characters as output units, which has the advantages that the pronunciation dictionary is not required, and it is possible to generalise to out-of-vocabulary words. The number of characters in our system is 35 including symbols such as `hyphen`, `slash`, `space`, etc, and tokens corresponding to the noise `[noise]`, `[vocalized-noise]`, `[laughter]`. In our experiments, we observed that the character level encoder-decoder model was computationally more expensive. This is because that the output sequences are much longer, and it requires many more iterations to estimate the attention weights in Eq. (6). Moreover, the character baseline system also performed worse compared to the corresponding word level system as predicting a longer output sequence is more challenging.

3.4. Depth of the encoder

In the previous experiments, we have only used 1 layer of RNN in the encoder after 1 hidden layer of feedforward neural network for feature extraction. In [22], we have shown that using more hidden layers in the feedforward neural network does not reduce the WER significantly. In this work, we investigate if using multiple layers of RNN in the encoder can improve the recognition accuracy. However, this configuration significantly increases the model size, and limited by the size of the GPU memory, we only performed the experiments with character level output units.[1]. The results are given in Table 3, which demonstrate that using multiple RNN layers in the encoder can significantly improve the recognition accuracy. However, the gain is much smaller for the model with 1000 hidden units in the RNN, which may be due to model overfitting. As aforementioned, we used the GRU [18] in all the recurrent layers, which has 2 additional gates compared to the conventional RNN. Adding one more layer of RNN can significantly increase the number of model parameters, especially in the case of using bidirectional RNNs as in this work. After cutting down the number of hidden units to be 500,

[1]Training this model requires large memory since all the hidden states $(\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t)$ for each frame in a minibatch are kept in the memory in order to dynamically compute the context vector $\mathbf{c}_o$.

Table 3. Results of using multiple RNN layers in the encoder.

System	Output	Dim	CHE	SWB	Avg
EncDec – 1 layer	char	1000	52.7	32.8	42.8
EncDec – 2 layer	char	1000	50.3	29.1	39.7
EncDec – 1 layer	char	500	54.1	34.5	44.4
EncDec – 2 layer	char	500	48.4	28.8	38.7
EncDec – 3 layer	char	500	48.2	27.3	37.8

Table 4. Comparison to CTC and DNN-HMM hybrid systems. In [17], the LMs were trained using a corpus of 31 billion words, while in [16, 8], the LMs were trained using the Switchboard and Fisher transcriptions.

System	Output	CHE	SWB	Avg
DNN-HMM sMBR [8]	-	24.1	12.6	18.4
CTC no LM [17]	char	56.1	38.0	47.1
CTC+5-gram	char	47.0	30.8	39.0
CTC+7-gram	char	43.8	27.8	35.9
CTC+NNLM (1 hidden layer)	char	41.1	23.4	32.3
CTC+NNLM (3 hidden layers)	char	39.9	21.8	30.9
CTC+RNNLM (1 hidden layer)	char	41.7	24.2	33.0
CTC+RNNLM (3 hidden layers)	char	40.2	21.4	30.8
Deep Speech [16]	char	31.8	20.0	25.9
EncDec no LM	word	46.5	26.3	36.4
EncDec no LM	char	48.2	27.3	37.8

we achieved significantly lower WER with multiple RNNs in the encoder. In the future, we shall evaluate the long memory decoder in this setup.

3.5. Comparison to CTC

In Table 4, we compare our results to previously published results using CTC on the same dataset [16, 17]. Note that in the CTC systems [16, 17], strong LMs were applied during decoding. According to [17], the LM can significantly improve the recognition accuracy of CTC systems. From Table 4, our encoder-decoder systems achieved much higher recognition accuracy compared to CTC in the case of no LM setting. However, we only obtained marginal improvement by LM rescoring, and therefore, our best system is still far left behind the CTC counterpart. In the future, we shall incorporate the LM directly into the decoder. We also refer to the publicly reported hybrid baseline in [8]. We see that there is still a big gap between the end-to-end and hybrid systems on this dataset. However, we also notice the recent results in [14], where CTC outperformed the hybrid baseline on the Google voice search task.

4. CONCLUSIONS

In this paper, we present the improvements obtained for the RNN encoder-decoder based end-to-end speech recognition on the large vocabulary task. We show a simple yet efficient and effective approach to schedule the SGD learning rates which achieves large gain in our experiments. In principle, the encoder-decoder approach does not need to rely on a language model given enough training data, and we proposed an approach to extend the decoder with long memory to enhance its power for implicit language modelling. Finally, using multiple recurrent layers in the encoder can significantly reduce the WER. In the future, we shall investigate using multiple recurrent layers in the decoder as well as incorporating a language model into the decoder.

5. REFERENCES

[1] H. A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*. Springer, 1994, vol. 247.

[2] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 161–174, 1994.

[3] 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," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[4] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*. IEEE, 2013, pp. 6645–6649.

[5] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. INTERSPEECH*, 2014.

[6] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 10, pp. 1533–1545, 2014.

[7] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization." in *INTERSPEECH*, 2012.

[8] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. INTERSPEECH*, 2013.

[9] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *Proc. ICASSP*. IEEE, 2013, pp. 6664–6668.

[10] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid nn/hmm model for speech recognition based on discriminative learning of speaker code," in *Proc. ICASSP*. IEEE, 2013, pp. 7942–7946.

[11] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. ASRU*. IEEE, 2013, pp. 55–59.

[12] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc. SLT*. IEEE, 2014, pp. 171–176.

[13] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.

[14] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Proc. INTERSPEECH*, 2015.

[15] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," *arXiv preprint arXiv:1507.08240*, 2015.

[16] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger *et al.*, "Deep Speech: Scaling up end-to-end speech recognition," in *arXiv preprint arXiv:1412.5567*, 2014.

[17] A. L. Maas, Z. Xie, D. Jurafsky, and A. Y. Ng, "Lexicon-free conversational speech recognition with neural networks," in *Proc. NAACL*, 2015.

[18] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *Pro. EMNLP*, 2014.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2015.

[20] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. ICML*, 2015.

[21] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results," *arXiv preprint arXiv:1412.1602*, 2014.

[22] L. Lu, X. Zhang, K. Cho, and S. Renals, "A study of the recurrent nerual network encoder-decoder for large vocabulary speech recognition," in *Proc. INTERSPEECH*, 2015.

[23] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," *arXiv preprint arXiv:1508.04395*, 2015.

[24] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[26] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[27] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[28] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *Proc. ICASSP*. IEEE, 2013, pp. 8624–8628.

[29] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. ICASSP*. IEEE, 1992, pp. 517–520.

[30] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlıcek, Y. Qian, P. Schwarz, J. Silovský, G. Semmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[31] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams *et al.*, "Recent advances in deep learning for speech research at microsoft," in *Proc. ICASSP*. IEEE, 2013, pp. 8604–8608.

[32] K. Heafield, "KenLM: Faster and smaller language model queries," in *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 2011, pp. 187–197.