Multi-task Learning with CTC and Segmental CRF for Speech Recognition

Liang Lu[1], *Lingpeng Kong*[2]

[1]Toyota Technological Institute at Chicago, USA
[2]School of Computer Science, Carnegie Mellon University, USA

llu@ttic.edu, lingpenk@cmu.edu

Abstract

Segmental Conditional Random Field (SCRF) and Connectionist Temporal Classification (CTC) have been well studied for acoustic modeling. The former defines the loss function at the sequence level, while the latter defines the loss at the frame level. When combined with neural networks for feature extraction, both models can be trained end-to-end. In this paper, we study the joint training approach of the two models using an interpolated loss functions for speech recognition, where both SCRF and CTC share the same recurrent neural network (RNN) encoder used for feature extraction. We show that the joint training approach improves the recognition accuracy of both SCRF and CTC models due to the regularization effect, which is similar to the observation of HMMs where sequence-discriminative training using a sequence-level loss can benefit from the cross entropy (CE) based regularization. Moreover, we show that CTC can also be used to pretrain the RNN encoder, which improves the convergence speed of the joint model.

Index Terms: speech recognition, end-to-end training, CTC, segmental RNN

1. Introduction

State-of-the-art speech recognition accuracy has been significantly improved over the past few years since the application of deep neural networks [1, 2]. Recently, it has been shown that with the application of both neural network acoustic model and language model, the automatic speech recognizer has approached the human-level accuracy on switchboard conversational speech recognition benchmark using around 2000 hours of transcribed data [3]. While the progress is mainly driven by well engineered neural network architectures and a large amount of training data, the hidden Markov model (HMM) that has been the backbone for speech recognition for decades is still playing the central role in the deep learning revolution for speech processing. Though tremendously successful for the problem of speech recognition, the HMM-based pipeline factorizes the whole system into several components, and building these components separately may be computationally less efficient when developing a large scale system using thousands to hundred of thousands of data, e.g. [4].

Along with the mainstream study of using the hybrid HMM/NN framework for speech recognition, recently, there has been an increasing interest in end-to-end training approaches for this task. The key idea is to directly map the input acoustic frames to output characters or words without the intermediate alignment to context-dependent phones used by HMMs. In particular, three architectures have been proposed for the goal of end-to-end learning, i.e., connectionist temporal classification (CTC) [5, 6, 7, 8], sequence-to-sequence with attention model [9, 10, 11], and neural network segmental conditional random field (SCRF) [12, 13]. These end-to-end models significantly simplify the pipeline of speech recognition. They do not require intermediate alignment or segmentation as HMMs, instead, the alignment or segmentation is marginalized out during training for CTC and SCRF or inferred by the attention mechanism. In terms of the recognition accuracy, however, the end-to-end models are usually left behind the HMM-based counterpart. Though CTC has been shown to outperform HMM systems[14], the improvement is based on the use of context dependent phone targets and very large amount of training data. Therefore, it has almost the same system complexity as HMM acoustic models in that case. When the training data is less abundant, it has been shown that the accuracy of CTC systems degrade significantly [15].

However, the end-to-end models have the flexibility to be combined together to mitigate the weakness of each other. For instance, Kim et al. [16] proposed a multitask learning approach to train a joint attention and CTC model using a shared encoder. It is shown that the CTC auxiliary task can help the attention model to overcome the misalignment problem in the initial few epochs, and speed up the convergence of the attention model. Another nice property of the multi-task learning approach is that the joint model can still be trained end-to-end. Inspired by this work, we study the end-to-end training of the joint CTC and SCRF model using an interpolated loss function. The key difference of our study from [16] is that the two loss functions of CTC and attention model are locally normalized for each output token and they are both trained using the cross entropy criterion. However, the SCRF loss function is normalized at the sequence-level, which is similar to the sequence discriminative training objective function for HMMs. From this perspective, the interpolation of CTC and SCRF loss functions is analogous to the sequence discriminative training of HMMs with CE regularization to overcome overfitting, where a sequence-level loss is also interpolated with a frame-level loss, eg. [17]. Similar to the observations in [16], we demonstrate that the joint training approach improves the recognition accuracies of both CTC and SCRF acoustic models. Besides, we also show that CTC can be used to pretrain the neural network feature extractor to speed up the convergence of the joint model. Experiments were performed on the TIMIT database.

2. Segmental Conditional Random Fields

SCRF is a variant of the linear-chain CRF model where each output token corresponds to a segment of input tokens instead of a single input instance. In the context of speech recognition, given a sequence of input vectors of T frames $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$ and its corresponding sequence of output labels $\boldsymbol{y} = \{y_1, \cdots, y_J\}$, the zero-order linear-chain CRF defines the sequence-level conditional probability as

$$P(\boldsymbol{y} \mid \boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \prod_{t=1}^{T} \exp f\left(y_t, \boldsymbol{x}_t\right), \tag{1}$$

where $Z(\boldsymbol{X})$ denotes the normalization term, and $T = J$. Extension to higher order models is straightforward, but it is usually computationally much more expensive. The model defined as Eq (1) requires the length of $\boldsymbol{X}$ and $\boldsymbol{y}$ to be equal, which makes it infeasible for speech recognition directly because the lengths of the input and output sequences are not equal. For the case that $T \geq J$ as in speech recognition, SCRF defines the sequence-level conditional probability with the auxiliary segment labels $\boldsymbol{E} = \{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_J\}$ as

$$P(\boldsymbol{y}, \boldsymbol{E} \mid \boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \prod_{j=1}^{J} \exp f\left(y_j, \boldsymbol{e}_j, \bar{\boldsymbol{x}}_j\right), \tag{2}$$

where $\mathbf{e}_j = \langle s_j, n_j \rangle$ is a tuple of the beginning (s_j) and the end (n_j) time tag for the segment of y_j, and $n_j > s_j$ while $n_j, s_j \in [1, T]$; $y_j \in \mathcal{Y}$ and $\mathcal{Y}$ denotes the vocabulary set; $\bar{\boldsymbol{x}}_j$ is the embedding vector of the segment corresponding to the token y_j; In this case, $Z(\boldsymbol{X})$ sums over all the possible $(\boldsymbol{y}, \boldsymbol{E})$ pairs, i.e.,

$$Z(\boldsymbol{X}) = \sum_{\boldsymbol{y}, \boldsymbol{E}} \prod_{j=1}^{J} \exp f\left(y_j, \boldsymbol{e}_j, \bar{\boldsymbol{x}}_j\right). \tag{3}$$

Similar to other CRF-based models, the function $f(\cdot)$ is defined as

$$f\left(y_j, \boldsymbol{e}_j, \bar{\boldsymbol{x}}_t\right) = \mathbf{w}^\top \Phi(y_j, \boldsymbol{e}_j, \bar{\boldsymbol{x}}_j), \tag{4}$$

where $\Phi(\cdot)$ denotes the feature function, and $\mathbf{w}$ is the weight vector. Most of conventional approaches for SCRF-based acoustic models use manually defined feature function $\Phi(\cdot)$, where the features and segment boundary information are provided by an auxiliary system. In [13], we proposed an end-to-end training approach for SCRFs, where $\Phi(\cdot)$ was defined with neural networks, and the segmental level features were leaned by RNNs. The model was referred as segmental RNN (SRNN), and it will used as the implementation of the SCRF acoustic model for multi-task learning in this study.

2.1. Feature Function and Acoustic Embedding

SRNN uses an RNN to learn segmental level acoustic embeddings. Given the input sequence $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$, and we need to compute the embedding vector $\bar{\boldsymbol{x}}_j$ in Eq. (4) corresponding to the segment $\boldsymbol{e}_j = \langle s_j, n_j \rangle$. Since the segment boundaries are known, it is straightforward to employ an RNN to map the segment into a vector as

$$\begin{bmatrix} \boldsymbol{h}_{s_j} \\ \boldsymbol{h}_{s_j+1} \\ \vdots \\ \boldsymbol{h}_{n_j} \end{bmatrix} = \begin{bmatrix} \mathtt{RNN}(\boldsymbol{h}_0, \boldsymbol{x}_{s_j}) \\ \mathtt{RNN}(\boldsymbol{h}_{s_j}, \boldsymbol{x}_{s_j+1}) \\ \vdots \\ \mathtt{RNN}(\boldsymbol{h}_{n_j-1}, \boldsymbol{x}_{n_j}) \end{bmatrix} \tag{5}$$

where $\boldsymbol{h}_0$ denotes the initial hidden state, which is initialized to be zero. $\mathtt{RNN}(\cdot)$ denotes the nonlinear recurrence operation used in an RNN, which takes the previous hidden state and the feature vector at the current timestep as inputs, and produce an updated hidden state vector. Given the recurrent hidden states, the embedding vector can be simply defined as $\bar{\boldsymbol{x}}_j = \boldsymbol{h}_{n_j}$ as in our previous work [13]. However, the drawback of this implementation is the large memory cost, as we need to store the array of hidden states $\{\boldsymbol{h}_{s_j}, \cdots, \boldsymbol{h}_{n_j}\}$ for all the possible segments $\langle s_j, n_j \rangle$. If we denote H as the dimension of an RNN hidden state, the memory cost will be in the order of $O(T^2 H)$, where T is the length of $\boldsymbol{X}$. It is particularly a problem for the joint model as the CTC model requires additional memory space. In this work, we adopted another approach that requires much less memory. In this approach, we use an RNN to read the whole input sequence as

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \begin{bmatrix} \mathtt{RNN}(\boldsymbol{h}_0, \boldsymbol{x}_1) \\ \mathtt{RNN}(\boldsymbol{h}_1, \boldsymbol{x}_2) \\ \vdots \\ \mathtt{RNN}(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T) \end{bmatrix} \tag{6}$$

and we define the embedding vector for segment $\boldsymbol{e} = \langle k, t \rangle$ as

$$\bar{\boldsymbol{x}}_j = \begin{bmatrix} \boldsymbol{h}_{s_j} \\ \boldsymbol{h}_{n_j} \end{bmatrix} \tag{7}$$

In this case, we only provide the context information for the feature function $\Phi(\cdot)$ to extract segmental features. We refer this approach as context aware embedding. Since we only need to read the input sequence once, the memory cost is in the order of $O(TH)$, which is much smaller. The cost, however, is the slightly degradation of the recognition accuracy. This model is illustrated by Figure 1.

The feature function $\Phi(\cdot)$ also requires a vector representation of the label y_j. This embedding vector can be obtained using a linear embedding matrix as the common practice in RNN language models. More specifically, y_j is firstly represented as a one-hot vector $\boldsymbol{v}_j$, and it is then mapped into a continuous space by a linear embedding matrix $\boldsymbol{M}$ as

$$\boldsymbol{u}_j = \boldsymbol{M}\boldsymbol{v}_j \tag{8}$$

Given the acoustic embedding $\bar{\boldsymbol{x}}_j$ and label embedding $\boldsymbol{u}_j$, the feature function $\Phi(\cdot)$ can be represented as

$$\Phi(y_j, \boldsymbol{e}_j, \bar{\boldsymbol{x}}_j) = \sigma(\boldsymbol{W}_1 \boldsymbol{u}_j + \boldsymbol{W}_2 \bar{\boldsymbol{x}}_j + \boldsymbol{b}), \tag{9}$$

where σ denotes a non-linear activation function such as `sigmoid` or `tanh`; $\boldsymbol{W}_1, \boldsymbol{W}_2$ and $\boldsymbol{b}$ are weight matrices and a bias vector. corresponds to one layer or multiple layers of linear or non-linear transformation. In fact, it is straightforward to stack multiple nonlinear layers in the feature function.

2.2. Loss Function

For speech recognition, the segmentation labels $\boldsymbol{E}$ are usually unknown in the training set. In this case, we cannot train the model directly by maximizing the conditional probability as Eq. (2). However, the problem can be addressed by marginalizing out the segmentation variable as

$$\begin{aligned} \mathcal{L}_{scrf} &= -\log P(\boldsymbol{y} \mid \boldsymbol{X}) \\ &= -\log \sum_{\boldsymbol{E}} P(\boldsymbol{y}, \boldsymbol{E} \mid \boldsymbol{X}) \\ &= -\log \underbrace{\sum_{\boldsymbol{E}} \prod_{j} \exp f\left(y_j, \boldsymbol{e}_j, \bar{\boldsymbol{x}}_j\right)}_{\equiv Z(\boldsymbol{X}, \mathbf{y})} + \log Z(\boldsymbol{X}), \end{aligned} \tag{10}$$

where $Z(\boldsymbol{X}, \boldsymbol{y})$ denotes the summation over all the possible segmentations when only $\boldsymbol{y}$ is observed. To simplify notations,

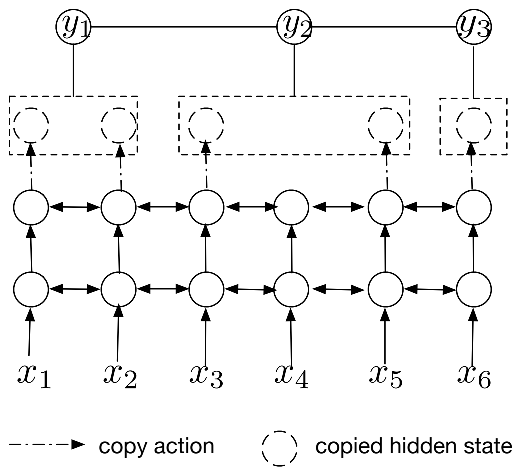

Figure 1: *A Segmental RNN with the context aware embedding. The acoustic segmental embedding vector is composed by the hidden states from the RNN encoder corresponding to the beginning and end time tags.*

the objective function $\mathcal{L}_{scrf}$ is define with only one training utterance.

However, the number of possible segmentations is exponential with the length of $\boldsymbol{X}$, which makes the naive computation of both $Z(\boldsymbol{X}, \boldsymbol{y})$ and $Z(\boldsymbol{X})$ impractical. To address this problem, a dynamic programming algorithm can be applied, which can reduce the computational complexity to $O(T^2 \cdot |\mathcal{Y}|)$ [18]. The computational cost can be further reduced by limiting the maximum length of all the possible segments. The readers are referred to [13] for further details including the decoding algorithm.

3. Connectionist Temporal Classification

CTC also directly computes the conditional probability of $P(\boldsymbol{y} \mid \boldsymbol{X})$, with the key difference from SCRF in that it normalizes the probabilistic distribution at the per-frame level. To address the problem of length mismatch between the input and output sequences, CTC allows repetitions of output labels and introduces a special blank token $(-)$, which represents the probability of not emitting any valid label at a particular time step. The conditional probability is then obtained by summing over all the probabilities of all the paths that corresponding to $\boldsymbol{y}$ after merging the repeated labels and removing the blank tokens, i.e.,

$$P(\boldsymbol{y} \mid \boldsymbol{X}) = \sum_{\boldsymbol{\pi} \in \Phi(\boldsymbol{y})} P(\boldsymbol{\pi} \mid \boldsymbol{X}), \quad (11)$$

where $\Phi(\boldsymbol{y})$ denotes the set of all the possible paths that correspond to $\boldsymbol{y}$ after repetitions of labels and insertions of the blank token. Now the length of $\boldsymbol{\pi}$ is the same as $\boldsymbol{X}$, the probability $P(\boldsymbol{\pi} \mid \boldsymbol{X})$ is then simply approximated by the independence assumption as

$$P(\boldsymbol{\pi} \mid \boldsymbol{X}) \approx \prod_{t=1}^{T} P(\pi_t \mid \boldsymbol{x}_t), \quad (12)$$

where $\pi_t \in \mathcal{Y} \cup \{-\}$, and $P(\pi_t \mid \boldsymbol{x}_t)$ can be computed using the softmax function. The training criterion for CTC is to maximize the conditional probability of the ground truth labels, which is equivalent to minimize the negative log likelihood as

$$\mathcal{L}_{ctc} = -\log P(\boldsymbol{y} \mid \boldsymbol{X}), \quad (13)$$

which can be reformulated as the CE criterion. More details regarding the computation of the loss and the backpropagation algorithm to train CTC models can be found in [19].

Table 1: *Phone error rates of baseline CTC and SRNN models.*

Model	features	#Layer	Dim	dev	eval
SRNN	FBANK	3	128	19.2	20.5
SRNN	fMLLR	3	128	17.6	19.2
SRNN	FBANK	3	250	18.1	20.0
SRNN	fMLLR	3	250	16.6	17.9
CTC	FBANK	3	128	20.0	21.8
CTC	fMLLR	3	128	17.7	18.4
CTC	FBANK	3	250	17.7	19.9
CTC	fMLLR	3	250	16.7	17.8

Table 2: *Results of three types of acoustic features.*

Model	features	Dim	dev	eval
SRNN	FBANK	250	18.1	20.0
+MTL	FBANK	250	17.5	18.7
SRNN	fMLLR	250	16.6	17.9
+MTL	fMLLR	250	15.9	17.5
CTC	FBANK	250	17.7	19.9
+MTL	FBANK	250	17.2	18.9
CTC	fMLLR	250	16.7	17.8
+MTL	fMLLR	250	16.2	17.4

4. Joint Training Loss

Training the two models jointly is trivial. We can simply interpolate the CTC and SCRF loss functions as

$$\mathcal{L} = \lambda \mathcal{L}_{ctc} + (1 - \lambda) \mathcal{L}_{scrf}, \quad (14)$$

where $\lambda \in [0, 1]$ is the interpolation weight. The two models share the same neural network for feature extraction. In this work, we focus on the RNN with long short-term memory (LSTM) [20] units for feature extraction. Other type of neural architecture, e.g., convolutional neural network (CNN) or combinations of CNN and RNN, may be considered for future work.

5. Experiments

Our experiments were performed on the TIMIT database, and both the SRNN and CTC models were implemented using the DyNet toolkit [21]. We followed the standard protocol of the TIMIT dataset, and our experiments were based on the Kaldi recipe [22]. We used the core test set as our evaluation set, which has 192 utterances. Our models were trained with 48 phonemes, and their predictions were converted to 39 phonemes before scoring. The dimension of $\mathbf{u}_j$ was fixed to be 64, and the dimension of $\mathbf{w}$ in Eq. (4) is also 64. We set the initial SGD learning rate to be 0.1, and we exponentially decay the learning rate by 0.75 when the validation error stopped decreasing. We also subsampled the acoustic sequence by a factor of 4 using the hierarchical RNN as in [13]. Our models were trained with dropout regularization [23], using an specific implementation for recurrent networks [24]. The dropout rate was 0.2 unless specified otherwise. Our models were randomly initialized with the same random seed.

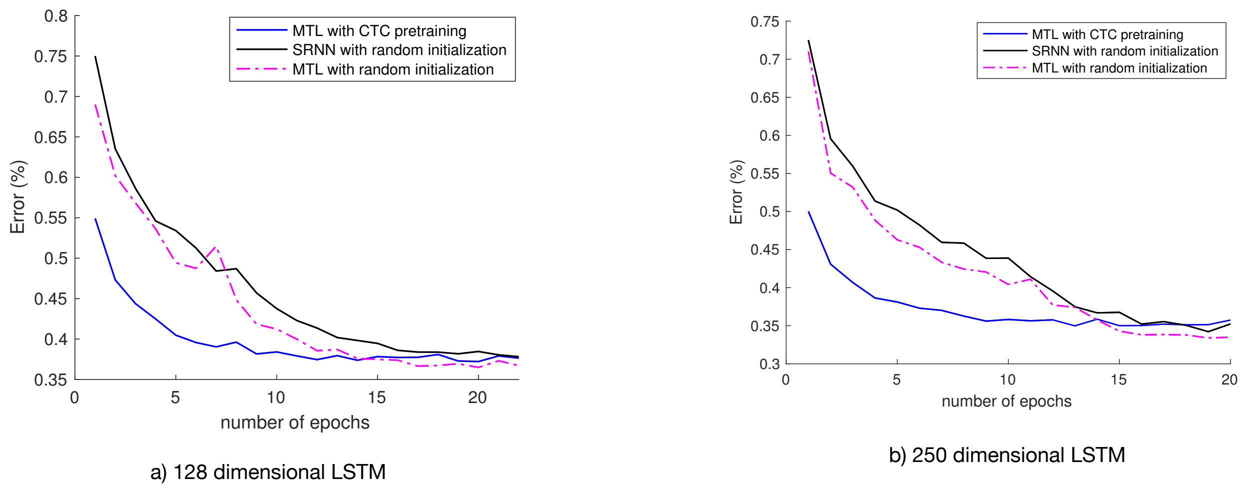

Figure 2: *Convergence curves with and without CTC pretraining in multi-task learning framework.*

5.1. Baseline Results

Table 1 shows the baseline results of SRNN and CTC models using two different kinds of features. The FBANK features are 120-dimensional with delta and delta-delta coefficients, and the fMLLR features are 40-dimensional, which were obtained from a Kaldi baseline system. We used a 3-layer bidirectional LSTMs for feature extraction, and we used the greedy best path decoding algorithm for both models. Our SRNN and CTC achieved comparable phone error rate (PER) for both kinds of features. However, for the CTC system, Graves et al. [25] obtained better result using about the same size of neural network (3 hidden layers with 250 hidden units of bidirectional LSTMs) than ours (18.6% vs.19.9%). Apart from the implementation difference using different code bases, Graves et al. [25] applied the prefix decoding with beam search, which may have lower search error than our best path decoding algorithm.

5.2. Multi-task Learning Results

Table 2 shows results of multi-task learning for CTC and SRNN using the interpolated loss as Eq. (14). We only show results of using LSTMs with 250 dimensional hidden states. The interpolation weight was set to be 0.5. In our experiments, tuning the interpolation weight did not further improve the recognition accuracy. From Table 2, we can see from the table that multi-task learning improve recognition accuracies of both SRNN and CTC acoustic models, which may due to the regularization effect of the joint training loss. The improvement for FBANK feature is relatively much larger than fMLLR features.

One of the major drawbacks of SCRF models is their high computational cost. In our experiments, the CTC model is around 3 - 4 times faster than the SRNN model which uses the same RNN encoder. The joint model by multi-task learning is slightly more expensive than the standalone SRNN model. To cut down the computational cost, we investigated if CTC can be used to pretrain the RNN encoder to speed up the training of the joint model. This is analogous to sequence training of HMM acoustic models, where the network is usually pretrained by the frame-level CE criterion. Figure 2 shows the convergence curves of the joint model with and without CTC pretraining, and we see pretraining indeed improves the convergence speed of the joint model.

6. Conclusion

We investigated multi-task learning with CTC and SCRF for speech recognition in this paper. Using an RNN encoder for feature extraction, both CTC and SCRF can be trained end-to-end, and the two model can be trained together by interpolating the two loss functions. From experiments on the TIMIT dataset, the multi-task learning approach improved the recognition accuracies of both CTC and SCRF acoustic models. We also showed that CTC can be used to pretrain the RNN encoder, and speed up the training of the joint model. In future, we will study the multi-task learning approach for larger scale speech recognition tasks, where the CTC pretraining approach may be more helpful to overcome the problem of high computational cost.

7. Acknowledgements

We thank the NVIDIA Corporation for the donation of a Titan X GPU.

8. References

[1] 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.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks." in *Interspeech*, 2011, pp. 437–440.

[3] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[4] H. Soltau, H. Liao, and H. Sak, "Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition," *arXiv preprint arXiv:1610.09975*, 2016.

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

[6] 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.

[7] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech

recognition with recurrent neural networks," in *Proc. ICASSP*. IEEE, 2015, pp. 4280–4284.

[8] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *Proc. ASRU*. IEEE, 2015, pp. 167–174.

[9] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*, 2015, pp. 577–585.

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

[11] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 4960–4964.

[12] O. Abdel-Hamid, L. Deng, D. Yu, and H. Jiang, "Deep segmental neural networks for speech recognition." in *Proc. INTERSPEECH*, 2013, pp. 1849–1853.

[13] L. Lu, L. Kong, C. Dyer, N. Smith, and S. Renals, "Segmental recurrent neural networks for end-to-end speech recognition," in *Proc. INTERSPEECH*, 2016.

[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] G. Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models," in *Proc. INTERSPEECH*, 2016.

[16] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017.

[17] 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.

[18] S. Sarawagi and W. W. Cohen, "Semi-markov conditional random fields for information extraction." in *Proc. NIPS*, vol. 17, 2004, pp. 1185–1192.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. ACM, 2006, pp. 369–376.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn *et al.*, "DyNet: The Dynamic Neural Network Toolkit," *arXiv preprint arXiv:1701.03980*, 2017.

[22] 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.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[24] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

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