Transferring Knowledge from a RNN to a DNN

William Chan[*1], *Nan Rosemary Ke*[*1], *Ian Lane*[1,2]

Carnegie Mellon University
[1]Electrical and Computer Engineering, [2] Language Technologies Institute
williamchan@cmu.edu, rosemary.ke@sv.cmu.edu, lane@cmu.edu

Abstract

Deep Neural Network (DNN) acoustic models have yielded many state-of-the-art results in Automatic Speech Recognition (ASR) tasks. More recently, Recurrent Neural Network (RNN) models have been shown to outperform DNNs counterparts. However, state-of-the-art DNN and RNN models tend to be impractical to deploy on embedded systems with limited computational capacity. Traditionally, the approach for embedded platforms is to either train a small DNN directly, or to train a small DNN that learns the output distribution of a large DNN. In this paper, we utilize a state-of-the-art RNN to transfer knowledge to small DNN. We use the RNN model to generate soft alignments and minimize the Kullback-Leibler divergence against the small DNN. The small DNN trained on the soft RNN alignments achieved a 3.9 WER on the Wall Street Journal (WSJ) eval92 task compared to a baseline 4.6 WER or more than 13% relative improvement.

Index Terms: Deep Neural Networks, Recurrent Neural Networks, Automatic Speech Recognition, Model Compression, Embedded Platforms

1. Introduction

Deep Neural Networks (DNNs) combined with Hidden Markov Models (HMMs) have been shown to perform well across many Automatic Speech Recognition (ASR) tasks [1, 2, 3]. DNNs accept an acoustic context (e.g., a window of fMLLR features) as inputs and models the posterior distribution of the acoustic model. The "deep" in DNN is critical, state-of-the-art DNN models often contain multiple layers of non-linearities, giving it powerful modelling capabilities [4, 5].

Recently, Recurrent Neural Networks (RNNs) have demonstrated even more potential over its DNN counterparts [6, 7, 8]. RNN models are neural network models that contain recurrent connections or cycles in the connectivity graph. RNN models when unrolled, can actually be seen as a very special case of DNN. The recurrent nature of the RNN allows us to model temporal dependencies, which is often the case in speech sequences. In particular, the recurrent structure of the model allows us to store temporal information (e.g., the cell state in LSTM [9]) within the model. In [10], RNNs were shown to outperform DNNs in large commercial ASR systems. And in [8], RNNs have been shown to provide better performance over DNNs in robust ASR.

Currently, there has been much industry interest in ASR for embedded platforms, for example, mobile phones, tablets and smart watches. However, these platforms tend to have limited computational capacity (e.g., no/limited GPU and/or low performance CPU), limited power availability (e.g., small batteries) and latency requirements (e.g., asking a GPS system for driving directions should be responsive). Unfortunately, many state-of-the-art DNN and RNN models are simply too expensive or impractical to run on embedded platforms. Traditionally, the approach is simply to use a small DNN, reducing the number of layers and the number of neurons per layer; however, such approaches often suffer from Word Error Rate (WER) performance degradations [11]. In our paper, we seek to improve the WER of small models which can be applied to embedded platforms.

DNNs and RNNs are typically trained from forced alignments generated from a GMM-HMM system. We refer to this as a hard alignment, the posterior distribution is concentrated on a single acoustic state for each acoustic context. There has been evidence that these GMM alignment labels are not the optimal training labels as seen in [12, 13]. The GMM alignments make various assumptions of the data, such as independence of acoustic frames given states [12]. In this paper, we show soft distribution labels generated from an expert is potentially more informative over the GMM hard alignments leading to WER improvements. The effects of the poor GMM alignment quality may be hidden away in large deep networks, which have sufficient model capacity. However, in narrow shallow networks, training with the same GMM alignments often hurts our ASR performance [11].

One approach is to change the training criteria, rather than trying to match our DNN to the GMM alignments, we can instead try and match our DNN to the distribution of an expert model (e.g., a big DNN). In [14], a small DNN was trained to match the output distribution of a large DNN. The training data labels are generated by passing labelled and unlabelled data through the large DNN, and training the small DNN to match the output distribution. The results were promising, [14] achieved a 1.33% WER reduction over their baseline systems.

Another approach is to train an model to match the softmax logits or internal representation of an expert model [15, 16]. In [15], an ensemble of experts were trained and used to teach a (potentially smaller) DNN. Their motivation was inference (e.g., computational cost grows linearly to the number of ensemble models), however the principle of model compression applies [17]. [15] also generalized the framework, and showed that we can train the models to match the logits of the softmax, rather than directly modelling the distributions which could yield more knowledge transfer.

In this paper, we want to maximize small DNN model performance targeted at embedded platforms. We transfer knowledge from a RNN expert to a small DNN. We first build a large RNN acoustic model, and we then let the small DNN model learn the distribution or soft alignment from the large

[*]Equal contribution

RNN model. We show our technique will yield improvements in WER compared to the baseline models trained on the hard GMM alignments.

The paper is structured as follows. Section 2, begins with an introduction of a state-of-the-art RNN acoustic model. In Section 3, we describe the methodology used to transfer knowledge from a large RNN model to a small DNN model. Section 4 is gives experiments, results and analysis. And we finish in Section 5 with our conclusion and future work discussions.

2. Deep Recurrent Neural Networks

There exist many implementations of RNNs [18], and LSTM is a particular implementation of RNN that is easy to train and does not suffer from the vanishing or exploding gradient problem in Backpropagation Through Time (BPTT) [19]. We follow [20, 21] in our LSTM implementation:

$$i_t = \phi(W_{xi}x_t + W_{hi}h_{t-1}) \quad (1)$$
$$f_t = \phi(W_{xf}x_t + W_{hf}h_{t-1}) \quad (2)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1}) \quad (3)$$
$$o_t = \phi(W_{xo}x_t + W_{ho}h_{t-1}) \quad (4)$$
$$h_t = o_t \odot \tanh(c_t) \quad (5)$$

This particular LSTM implementation omits the the bias and peephole connections. We also apply a cell clipping of 3 to ease the optimization to avoid exploding gradients. LSTMs can also be extended to be a Bidirectional LSTM (BLSTM), to capture temporal dependencies in both set of directions [7].

RNNs (and LSTMs) can be also be extended into deep RNN architectures [22]. There has been evidence that the deep RNN models can perform better than the shallow RNN models [7, 22, 21]. The additional layers of nonlinearities can give the network additional model capacity similar to the multiple layers of nonlinearities in a DNN.

We follow [21], in building our deep RNN; to be exact, the particular RNN model is actually termed a TC-DNN-BLSTM-DNN model. The architecture begins with a Time Convolution (TC) over the input features (e.g., fMLLR) [23]. This is followed by a DNN signal processor which can project the features into a higher dimensional space. The projected features are then consumed by a BLSTM, modelling the acoustic context sequence. Finally a DNN with a softmax layer is used to model the posterior distribution. [21]'s model gave more than 8% relative improvement over previous state-of-the-art DNNs in the Wall Street Journal (WSJ) eval92 task. In this paper, we use the TC-DNN-BLSTM-DNN model as our deep RNN to generate the training alignments from which the small DNN will learn from.

3. Methodology

Our goal is to transfer knowledge from the RNN expert to a small DNN. We follow an approach similar to [14]. We transfer knowledge by training the DNN to match the RNN's output distribution. Note that we train on the soft distribution of the RNN (e.g., top k states) rather than just the top-1 state (e.g., realigning the model with the RNN). In this paper we will show the distribution generated by the RNN is more informative over the GMM alignments. We will also show the soft distribution of the RNN is more informative over taking just the top-1 state generated by the RNN.

3.1. KL Divergence

We can match the output distribution of our DNN to our RNN by minimizing the Kullback-Leibler (KL) divergence between the two distributions. Namely, given the RNN posterior distribution P and the DNN posterior distribution Q, we want to minimize the KL divergence $D_{KL}(P||Q)$:

$$D_{KL}(P(s|x)||Q(s|x)) = \sum_i P(s_i|x) \ln \frac{P(s_i|x)}{Q(s_i|x)} \quad (6)$$
$$= H(P,Q) - H(P) \quad (7)$$

where $s_i \in s$ are the acoustic states, $H(P,Q) = \sum_i -P(s_i|x) \ln Q(s_i|x)$ is the cross entropy term and $H(P) = \sum_i P(s_i|x) \ln P(s_i|x)$ is the entropy term. We can safely ignore the $H(P)$ entropy term since its gradient is zero with respect to the small DNN parameters. Thus, minimizing the KL divergence is equivalent to minimizing the Cross Entropy Error (CSE) between the two distributions:

$$H(P,Q) = \sum_i -P(s_i|x) \ln Q(s_i|x) \quad (8)$$

which we can easily differentiate and compute the pre-softmax activation a (e.g., the softmax logits) derivative:

$$\frac{\partial J}{\partial a_i} = Q(s_i|x) - P(s_i|x) \quad (9)$$

3.2. Alignments

In most ASR scenarios, DNNs and RNNs are typically trained with forced alignments generated from GMM-HMM models to model the posterior distribution. We refer this alignment as a hard GMM alignment because the probability is concentrated on only a single state. Furthermore, the alignment labels generated from GMM-HMM model are not always the optimal for training DNNs [12]. The GMM-HMM makes various assumptions that may not be true (e.g., independence of frames). One possible solution is to use labels or alignments from another expert model, for example in [15] an ensemble of experts was used to teach one model. In this paper, we generate labels from an expert RNN which provide better training targets compared to the GMM alignments.

One possibility is to generate hard alignments from a RNN expert. This is done by first training the RNN with hard alignments from the GMM-HMM model. After the DNN is trained, we then realign the data by taking hard alignments (e.g., top-1 probability state) from the trained RNN. The alignment is hard as it takes only the most probable phoneme state for each acoustic context, and the probability is concentrated on a single phoneme state.

On the other hand, we could utilize the full distribution or soft alignment associated with each acoustic frame. More precisely, for each acoustic context, we take the full distribution of the phonetic states and their probabilities. However, this suffers from several problems. First, during training, we need to either run the RNN in parallel or pre-cache the distribution on disk. Running the RNN in parallel is an expensive operation and undesirable. The alternative is caching the distribution on disk, which would require significant amounts of storage (e.g., we typically have several thousand acoustic states). For example, in WSJ, it would take approximately 1 TiB to store the full distribution of the si284 dataset if we assume using 32-bit floats. We also run into bandwidth issues when loading the training samples from the disk cache. Finally, the entire distribution

Figure 1: We use the hard GMM alignments to first train a RNN, after which we use the soft alignments from the RNN to train our small DNN.

may not be useful, as there will be many states with near zero values; intuition suggests we can just discard those states (e.g., lossy compression).

Our solution sits inbetween the two extremes of taking only the top-1 state or taking the full distribution. We find that the posterior distributions are typically concentrated on only a few states. Therefore, we can make use of almost the full distribution by storing only a small portion of the states probability distribution. We take the states that contains the top 98% of the probability distribution. Note, this is different than taking the top-k states, we take at least n states where we can capture at least 98% of the distribution, and n will vary per frame (on average $n = 6$). We then re-normalize the probability per frame to ensure the distribution sums up to 1. This lossy compression method loses up to 2% of the original probability mass.

4. Experiments and Results

We experiment with the WSJ dataset; we use si284 with approximately 81 hours of speech as the training set, dev93 as our development set and eval92 as our test set. We observe the WER of our development set after every epoch, we stop training once the development set no longer improves. We report the converged dev93 and the corresponding eval92 WERs. We use the same fMLLR features generated from the Kaldi s5 recipe [24], and our decoding setup is exactly the same as the s5 recipe (e.g., big dictionary and trigram pruned language model). We use the tri4b GMM alignments as our hard forced alignment training targets, and there are a total of 3431 acoustic states. The GMM tri4b baseline achieved a dev and test WER of 9.4 and 5.4 respectively.

4.1. Optimization

In our DNN and RNN optimization procedure, we initialized our networks randomly (e.g., no pretraining) and we used Stochastic Gradient Descent (SGD) with a minibatch size of 128. We apply no gradient clipping or gradient projection in our LSTM. We experimented with constant learning rates of [0.1, 0.01, 0.001] and geometric decayed learning rates with initial values of [0.1, 0.01] with a decay factor of 0.5. We report the best WERs out of these learning rate hyperparameter optimizations.

4.2. Big DNN and RNN

We first built several baseline (big) DNN and RNN systems. These are the large networks and not suitable for deployment on mobile platforms. We followed the Kaldi s5 recipe and built a 7 layer DNN and 2048 neurons per hidden layer with DBN pretraining and achieves a eval92 WER of 3.8 [24]. We also followed [21] and built a 5 layer ReLU DNN with 2048 neurons per hidden layer and achieves a eval92 WER of 3.8. Our RNN model follows [21], consists of 2048 neurons per layer for the DNN layers, and 256 bidirectional cells for the BLSTM. The RNN model achieves a eval92 WER of 3.5, significantly better than both big DNN models. Each network has a softmax output of 3431 states matching the GMM model. Table 1 summarizes the results for our baseline big DNN and big RNN experiments.

Table 1: *Wall Street Journal WERs for big DNN and RNN models.*

Model	dev93 WER	eval92 WER
GMM Kaldi	9.4	5.4
DNN Kaldi s5	6.7	3.8
DNN ReLU	6.8	3.8
RNN [21]	6.6	3.5

4.3. Small DNN

We want to build a small DNN that is easily computable by an embedded device. We decided on a 3 layer network (2 hidden layers), wherein each hidden layer has 512 ReLU neurons and a final softmax of 3431 acoustic states matching the GMM. Matrix-Matrix Multiplication (MMM) is an $O(n^2m)$ operation where n is the number of neurons and m is the input size, the effect is approximately a 10 times reduction in number of computations for the hidden layers (when comparing the 4 hidden layers of 2048 neurons vs. a 2 hidden layers of 512 neurons). This will allow us to perform fast interference on embedded platforms with limited CPU/GPU capacity.

We first trained a small ReLU DNN using the hard GMM alignments. We achieved a 4.5 WER compared to 3.8 WER of the big ReLU DNN model on the eval92 task. The dev93 WER is 8.0 for small model vs 6.8 for the large model; the big gap in dev93 WER suggests the big DNN model is able to optimize substantially better. The large DNN model has significantly more model capacity, and thus yielding its better results over the small DNN.

Next, we experimented with the hard RNN alignment. We take the top-1 state of the RNN model and train our DNN towards this alignment. We did not see any improvement, while the dev93 WER improves from 8.0 to 7.8, the eval92 WER degrades from the 4.5 to 4.6. This suggests, the RNN hard alignments are worse labels than the original GMM alignments. The information provided by the RNN when looking at only the top state is no more informative over the GMM hard alignments. One hypothesis is our DNN model overfits towards the RNN hard alignments, since the dev93 WER was able to improve, while the model is unable to generalize the performance to the eval92 test set.

We now experiment with the RNN soft alignment, wherein we can add the soft distribution characteristics of the RNN to the small DNN. We take the top 98% percentile of probabilities of from the RNN distribution and renormalize them (e.g., ensure the distribution sums up to 1). We minimize the KL divergence between the RNN soft alignments and the small DNN.

Table 2: *Small DNN WERs for Wall Street Journal based on different training alignments.*

Alignment	dev93 WER	eval92 WER
Hard GMM	8.0	4.5
Hard RNN	7.8	4.6
Soft RNN	7.4	3.9
Soft DNN	7.4	4.3

We see a significant improvement in WER. We achieve a dev93 WER of 7.4 and eval92 3.9. In the eval92 scenario, our WER improves by over 13% relative compared to the baseline GMM hard alignment. We were almost able to match the WER of the big DNN of 3.8 (off by 3.6% relative), despite the big DNN have many more layers and neurons. The RNN soft alignment adds considerable information to the training labels over the GMM hard alignments or the RNN hard alignments.

We also experimented training on the big DNN soft alignments. The big DNN model is the DNN ReLU model mentioned in table 1, wherein it achieved a eval92 WER of 3.8. Once again, we generate the soft alignments and train our small DNN to minimize the KL divergence. We achieved a dev93 WER of 7.4 and eval92 WER of 4.3. There are several things to note, first, we once again improve over the GMM baseline by 5.9% relative. Next, the dev93 WER is very close to the RNN soft alignment (less than 1% relative), however, the gap widens when we look at the eval92 WER (more than 8% relative). This suggests the model overfits more under the big DNN soft alignments, and the RNN soft alignments provide more generalization. The quality of the RNN soft alignments are much better than big DNN soft alignments. Table 2 summarizes the WERs for the small DNN model using different training alignments.

4.4. Cross Entropy Error

We compute the CSE of our various models against the GMM alignment for the dev93 dataset. We measure the CSE against dev93 since that is our stopping criteria and that is the optimization loss. The CSE will give us a better indication of the optimization procedure, and how our models are overfitting. Table 3 summarizes our CSE measurements. There are several observations, first the big RNN is able to achieve a lower CSE compared to the big DNN. The RNN model is able to optimize better than the DNN as seen with the better WERs the RNN model provides. This is as expected since the big RNN model achieves the best WER.

The next observation is that the small DNNs trained off the soft alignment from the large DNN or RNN achieved a lower CSE and compared to the small DNN trained on the GMM hard alignment. This suggests the soft alignment labels are indeed better training labels in optimizing the model. The extra information contained in the soft alignment helps us optimize better towards our dev93 dataset.

The small DNN trained on the soft RNN alignments and soft DNN alignments give interesting results. These models achieved a lower CSE compared to the large RNN and large DNN models trained on the GMM alignments. However, the WERs are worse than the large RNN and large DNN models. This suggests the small model trained on the soft distribution is overfitting, it is unclear if the overfitting occurs because the smaller model can not generalize as well as the large model, or if the overfitting occurs because of the quality of the soft alignment labels.

Table 3: *Cross Entropy Error (CSE) on WSJ dev93 over our various models.*

Alignment	Model	CSE
GMM	Big RNN	1.27620
GMM	Big DNN	1.28431
Hard RNN	Small DNN	1.52135
Soft RNN	Small DNN	1.24617
Soft DNN	Small DNN	1.24723

5. Conclusion and Discussions

The motivation and application of our work is to extend ASR onto embedded platforms, where there is limited computational capacity. In this paper we have introduced a method to transfer knowledge from a RNN to a small DNN. We minimize the KL divergence between the two distributions to match the DNN's output to the RNN's output. We improve the WER from 4.54 trained on GMM forced alignments to 3.93 on the soft alignments generated by the RNN. Our method has resulted in more than 13% relative improvement in WER with no additional inference cost.

One question we did not answer in this paper is whether the small DNN's model capacity or the RNN's soft alignment is the bottleneck of further WER performance. We did not measure the effect of the small DNN's model capacity on the WER, would we get similar WERs if we increased or decreased the small DNN's size? If the bottleneck is in the quality of the soft alignments, then in princple we could reduce the small DNN's size further without impacting WER (much), however, if model capacity is the issue, then we should not use smaller networks.

On a similar question, we did not investigate the impact of the top probability selection in the RNN alignment. We threshold the top 98% of the probabilities out of convenience, however, how would selecting more or less probabilities affect the quality of the alignments. In the extreme case, wherein we only selected the top-1 probability, we found the model to perform much worse compared to the 98% soft alignments, and even worse than the GMM alignments, this evidence definitely shows the importance of the information contained in the soft alignment.

We could also extend our work similar to [14] and utilize vast amounts of unlabelled data to improve our small DNN. In [14], they applied unlabelled data to their large DNN expert to generate vast quantities of soft alignment labels for the small DNN to learn from. In principle, one could extend this to an infinite amount of training data with synthetic data generation, which has been shown to improve ASR performance [25].

Finally, we did not experiment with sequence training [26], sequence training has almost always shown to help [27], it would be interesting to see the effects of sequence training on these small models, and whether we can further improve the ASR performance.

6. Acknowledgements

We thank Won Kyum Lee for helpful discussions and proofreading this paper.

7. References

[1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 30–42, January 2012.

[2] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," May 2013.

[3] H. Soltau, G. Saon, and T. Sainath, "Joint Training of Convoutional and Non-Convoutional Neural Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014.

[4] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On Rectified Linear Units for Speech Processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[5] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving Deep Neural Networks for LVCSR Using Rectified Linear Units and Dropout," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[6] A. Graves, A. rahman Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[7] A. Graves, N. Jaitly, and A. rahman Mohamed, "Hybrid Speech Recognition with Bidirectional LSTM," in *Automatic Speech Recognition and Understanding Workshop*, 2013.

[8] C. Weng, D. Yu, S. Watanabe, and F. Jung, "Recurrent Deep Neural Networks for Robust Speech Recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014.

[9] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[10] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *INTERSPEECH*, 2014.

[11] X. Lei, A. Senior, A. Gruenstein, and J. Sorensen, "Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices," in *INTERSPEECH*, 2013.

[12] N. Jaitly, V. Vanhoucke, and G. Hinton, "Autoregressive product of multi-frame predictions can improve the accuracy of hybrid models," in *INTERSPEECH*, 2014.

[13] A. Senior, G. Heigold, M. Bacchiani, and H. Liao, "GMM-Free DNN Training," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014.

[14] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning Small-Size DNN with Output-Distribution-Based Criteria," in *INTERSPEECH*, 2014.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," in *Neural Information Processing Systems: Workshop Deep Learning and Representation Learning Workshop*, 2014.

[16] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for Thin Deep Nets," in *International Conference on Learning Representations*, 2015.

[17] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model Compression," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," in *Neural Information Processing Systems: Workshop Deep Learning and Representation Learning Workshop*, 2014.

[19] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies," 2011.

[20] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in *arXiv:1411.4555*, 2014.

[21] W. Chan and I. Lane, "Deep Recurrent Neural Networks for Acoustic Modelling," in *INTERSPEECH (submitted)*, 2015.

[22] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to Construct Deep Recurrent Neural Networks," in *International Conference on Learning Representations*, 2014.

[23] W. Chan and I. Lane, "Deep Convolutional Neural Networks for Acoustic Modeling in Low Resource Languages," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannenmann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," in *Automatic Speech Recognition and Understanding Workshop*, 2011.

[25] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng, "Deep Speech: Scaling up end-to-end speech recognition," in *arXiv:1412.5567*, 2014.

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

[27] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence Discriminative Distributed Training of Long Short-Term Memory Recurrent Neural Networks," in *INTERSPEECH*, 2014.