A time delay neural network architecture for efficient modeling of long temporal contexts

Vijayaditya Peddinti[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing &
[2]Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD 21218, USA
vijay.p,khudanpur@jhu.edu, dpovey@gmail.com

Abstract

Recurrent neural network architectures have been shown to efficiently model long term temporal dependencies between acoustic events. However the training time of recurrent networks is higher than feedforward networks due to the sequential nature of the learning algorithm. In this paper we propose a time delay neural network architecture which models long term temporal dependencies with training times comparable to standard feed-forward DNNs. The network uses sub-sampling to reduce computation during training. On the Switchboard task we show a relative improvement of 6% over the baseline DNN model. We present results on several LVCSR tasks with training data ranging from 3 to 1800 hours to show the effectiveness of the TDNN architecture in learning wider temporal dependencies in both small and large data scenarios.

Index Terms: time delay neural networks, acoustic modeling, recurrent neural networks

1. Introduction

Modeling the temporal dynamics in speech, to capture the long term dependencies between acoustic events, requires an acoustic model which can effectively deal with long temporal contexts. Two types of approaches to exploit long term temporal contexts are using feature representations, which are designed to present this information to the model in a suitable form, or using acoustic models, which can learn the long term dependencies based on short-term feature representations.

In this paper we adopt the model based approach. Recurrent neural networks (RNNs) which use a dynamically changing contextual window over all of the sequence history rather than a fixed context window have been shown to achieve state-of-art performance on LVCSR tasks [1]. However due to recurrent connections in the network, parallelization during training cannot be exploited to the same extent as in feed-forward neural networks.

Another neural network architecture which has been shown to be effective in modeling long range temporal dependencies is the time delay neural network (TDNN) proposed in [2]. This architecture uses a modular and incremental design to create larger networks from sub-components [3]. Despite being a feed-forward architecture, computing the hidden activations at all time steps is computationally expensive. We propose a sub-sampling technique where hidden activations at only few time steps are computed at each level. Through a proper selection of time steps, at which activations are computed, computation can be reduced, while ensuring that information from all time steps in the input context is processed by the network.

Neural network architectures have been shown to benefit from speaker adaptation. However, speaker adaptation techniques like fMLLR [4] require two passes of decoding. The 2-pass decoding strategy is difficult to use in online speech recognition applications. iVectors which capture both speaker and environment specific information have been shown to be useful for instantaneous and discriminative adaptation of the neural network [5, 6]. In this paper we use iVector based neural network adaptation.

The paper is organized as follows. Section 2 mentions relevant work, Section 3 describes the neural network architecture and training recipe in greater detail. Section 4 describes the experimental setup. Section 5 presents and analyzes the results primarily on the Switchboard task ([7]). It also presents results on other LVCSR tasks which have 3-1800 hours of training data. Section 6 presents the conclusions and the future work.

2. Relevant work

Feature representations such as TRAPs [8], wavelet based multi-scale spectro-temporal representations [9], deep scattering spectra [10] and other modulation feature representations [11] have been proposed to represent long term spectro-temporal dynamics of the signal. These features can be used with standard feed-forward DNNs to model relationships in wide temporal contexts.

On the other hand, recurrent neural network (RNN) architectures such as long term short term memory networks [12, 1] and feed-forward neural network architectures, such as time delay neural networks (TDNNs) [2], have been shown to effectively learn the temporal dynamics of the signal even from short term feature representations.

Due to dependencies between the time-frames being processed in an RNN, parallelization cannot easily be exploited to the same extent as in feedforward networks. Batching of sequences and distributed optimization can be used to parallelize the RNN training process. However the training times are still not competitive with those of feed-forward neural networks, especially when using GPUs. Saon et al. [13] have shown that through unfolding of the recurrent network during training, matrix-matrix operations can be exploited to speed-up training. However the unfolded RNN still has dependencies across hidden-representations computed for various time-steps. Hence an unfolded network of **T** time steps requires sequential computation of **T** hidden representations.

In this paper we use the feed-forward TDNN architecture,

This work was partially supported by NSF Grant No IIA 0530118.

which does not impose any relationship between the length of input-context (i.e., unfolding width used during training, in case of RNNs) and number sequential steps during training. The TDNN is used for modelling long term temporal dependencies from short-term speech features i.e., MFCCs.

3. Neural network architecture

When processing a wider temporal context, in a standard DNN, the initial layer learns an affine transform for the entire temporal context. However in a TDNN architecture the initial transforms are learnt on narrow contexts and the deeper layers process the hidden activations from a wider temporal context. Hence the higher layers have the ability to learn wider temporal relationships. Each layer in a TDNN operates at a different temporal resolution, which increases as we go to higher layers of the network.

The transforms in the TDNN architecture are tied across time steps and for this reason they are seen as a precursor to the convolutional neural networks. During back-propagation, due to tying, the lower layers of the network are updated by a gradient accumulated over all the time steps of the input temporal context. Thus the lower layers of the network are forced to learn translation invariant feature transforms [2].

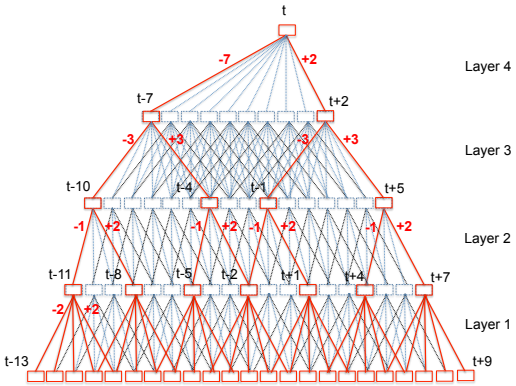

Figure 1: Computation in TDNN with sub-sampling (red) and without sub-sampling (blue+red)

The hyper-parameters which define the TDNN network are the input contexts of each layer required to compute an output activation, at one time step. A sample TDNN network is shown in Figure 1. The figure shows the time steps at which activations are computed, at each layer, and dependencies between activations across layers. It can be seen that the dependencies across layers are localized in time. Layerwise context specification, corresponding to this TDNN, is shown in column 2 of Table 1.

Table 1: Context specification of TDNN in Figure 1

Layer	Input context	Input context with sub-sampling
1	$[-2, +2]$	$[-2, 2]$
2	$[-1, 2]$	$\{-1, 2\}$
3	$[-3, 3]$	$\{-3, 3\}$
4	$[-7, 2]$	$\{-7, 2\}$
5	$\{0\}$	$\{0\}$

3.1. Sub-sampling

In a typical TDNN, hidden activations are computed at all time steps. However there are large overlaps between input contexts of activations computed at neighboring time steps. Under the assumption that neighboring activations are correlated, they can be sub-sampled.

Our approach is, rather than splicing together contiguous temporal windows of frames at each layer, to allow gaps between the frames. In fact, in the hidden layers of the network, we generally splice no more than two frames. For instance, the notation $\{-7, 2\}$ means we splice together the input at the current frame minus 7 and the current frame plus 2. Figure 1 shows this pictorially.

Empirically we found that what seems to work best is to splice together increasingly wide context as we go to higher layers of the network. The configuration in Figure 1, which is fairly typical, splices together frames $t-2$ through $t+2$ at the input layer (which we could write as context $\{-2, -1, 0, 1, 2\}$ or more compactly as $[-2, 2]$); and then at three hidden layers we splice frames at offsets $\{-1, 2\}$, $\{-3, 3\}$ and $\{-7, 2\}$. Table 1 tabulates these contexts (on the right), and compares with a hypothetical setup without sub-sampling. The fact that the differences between the offsets at the hidden layers were chosen to all be multiples of 3 is not a coincidence. We designed it this way in order to ensure that for each output frame, we need to evaluate a small number of hidden layer activations. The frames in red in Figure 1 are the ones we need to evaluate.

With the current sub-sampling scheme the overall necessary computation is reduced during the forward pass and backpropagation, due to selective computation of time steps. The training time of TDNN in Figure 1, without sub-sampling, is $\sim$10x compared to that of DNN with same number of layers. With proposed sub-sampling it is $\sim$2x the training time of DNN. Thus the sub-sampling process speeds up the TDNN training by $\sim$5x. Another advantage of using sub-sampling is the reduction in the model size. Splicing contiguous frames at hidden layers would require us to either have a very large number of parameters, or reduce the hidden-layer size significantly.

Sub-sampling at the middle of the network was also used in stacked bottle-neck networks [14]. In this architecture bottle-neck features were spliced across non-contiguous time steps and used as an input to a second neural network. However the bottle-neck network was not trained jointly with the final neural network.

We use asymmetric input contexts, with more context to the left, as this reduces the latency of the neural network in online decoding, and also because this seems to be more optimal from a WER perspective. Asymmetric context windows of up to 16 frames in past and 9 frames in the future were explored in this paper. It was observed that further extension of context on either side was detrimental to word recognition accuracies, though the frame recognition accuracies improved (this phenomenon is widely known).

A major difference in the current architecture compared to [2] is the use of the p-norm nonlinearity [15], which is a dimension reducing non-linearity. p-norm units with group size of 10 and $p = 2$ were used across all neural networks in our experiments, based on the observations made in [15] [1].

[1] More recent experiments in our TDNN framework show that that ReLU nonlinearity may actually perform better in this context than p-norm, but the full details were not ready in time for this paper and these results are not presented here.

3.2. Input Features

Mel-frequency cepstral coefficients (MFCCs) [16], without cepstral truncation, were used as input to the neural network i.e., 40 MFCCs were computed at each time step. Readers are recommended to see [17] for a more detailed discussion on input data representation used here and its comparison with log Mel features.

On each frame we append a 100-dimensional iVector [18] to the 40-dimensional MFCC input. The MFCC input is not subject to cepstral mean normalization; the intention is to allow the iVector to supply the information about any mean offset of the speaker's data, so the network itself can do any feature normalization that is needed. In order for the mean-offset information to be encoded in the iVector, we estimate the iVector on top of features that have not been mean-normalized. However, the Gaussian posteriors used for the iVector estimation are based on features that have been mean normalized using a sliding window of 6 seconds.

In order to ensure sufficient variety of the iVectors in the training data, rather than estimating a separate iVector per speaker we estimate them in an online fashion, where we only use frames prior to the current frame, including previous utterances of the same speaker. We re-set the utterance history every two utterances, so that the iVectors still have some training-data variety even when there are only a few speakers.

3.3. DNN training

The training recipe detailed in [15], with greedy layer-wise supervised training, preconditioned stochastic gradient descent (SGD) updates, exponential learning rate schedule and *mixing-up*, was used. Parallel training of the DNNs using up to 18 GPUs was done, using the model averaging technique in [17].

3.3.1. Sequence training

Sequence training was done on the DNN, based on a state-level variant of the Minimum Phone Error (MPE) criterion, called sMBR [19] . The training recipe mostly follows [20], although it has been modified for the parallel-training method. Training is run in parallel using 4 GPUs, while periodically averaging the parameters, just as in the cross-entropy training phase.

In the sMBR objective function insertion errors are not penalized, which could lead to larger number of insertions when decoding with sMBR trained acoustic models. Correcting this asymmetry in the sMBR objective function, by penalizing insertions, was shown to improve performance of sMBR models by 10% WER, relative, in a far field recognition task [21]. This modified objective function was used in this paper.

To compute the context-dependent state pseudo-likelihoods from the posteriors estimated by the neural network, the posteriors are divided by a prior. We found that the method of using the mean posterior (computed over a subset of the training data) as the prior [22] gave an improved performance when decoding with sMBR tuned models, so we used this method.

4. Experimental Setup

In Tables 2 and 3, we report results on 300 hour Switchboard conversational telephone speech task. The GMM-HMM system, used to generate the alignments for neural network training, and the language model are similar to those described in [20]. Results comparing DNN and TDNN architectures are presented in Table 2.

4.1. Data augmentation and enhanced lexicon

In this paper we are interested in a single-pass decoding setup which is suitable for the online speech recognition scenario. Thus the use of speaker adaptive feature transform techniques like fMLLR during test conditions is not viable. The use of data augmentation techniques to learn a network that is stable to different perturbations of the data is desirable in this scenario. In [23], speed perturbation of the training data was done to emulate vocal tract length perturbations and speaking rate perturbation. This was shown to provide 4.3% relative improvement across several LVCSR tasks. This data augmentation technique was adopted here. Three copies of the training data corresponding to speed perturbations of 0.9, 1.0 and 1.1 were created.

The iVector based TDNN system relies on the neural network to learn the necessary normalization, based on mean shifts captured in the iVector. However in well curated audio databases there is low variance in audio volume, leading to low variance in iVector *w.r.t.* mean shifts. Performing volume perturbation of the training data, where each recording in the training data was scaled with a random variable drawn from a uniform distribution over $[\frac{1}{8}, 2]$, emulates mean shifts in the MFCC domain. It was observed that volume perturbation of the training data resulted in 1.5% relative improvement in WER across test sets, compared with only speed perturbation.

In [24], the authors show that use of word pronunciation probabilities, to distinguish multiple word pronunciations, is beneficial. Further modeling the probability of silence before and after each pronunciation explicitly was shown to provide greater benefits. The lexicon updated with word pronunciation probabilities and word position dependent silence probabilities was used for decoding the test utterances.

We present results on Switchboard subset as well as the complete Hub5 '00 evaluation set. Only the results in the SWB column should be compared with the Hub5 '00 results presented in [26], [13] and [25].

5. Results

Table 2 compares different TDNNs and the baseline DNN. Each neural network has 4 hidden layers with *p*-norm input dimension of 3000 and group size of 10.

From Table 2, comparing DNN-A and TDNN-A, it can be seen that even with standard temporal contexts TDNNs perform better than DNNs. The number of parameters in the DNN-A system were increased to match the TDNN-A system, by increasing the number of hidden units. The results corresponding to this system are presented in the row titled DNN-A_2. It can be seen that despite matching the number of parameters the TDNN system performs better than the DNN system, for the same temporal context. A comparison of DNN-A, DNN-B and TDNN-D shows that DNNs are not as effective as TDNNs in processing wider temporal contexts. Comparing TDNNs A through E, $[-13, 9]$ was found to be the optimal temporal context.

A smaller TDNN with layer-wise contexts of TDNN-D was built, as it would be suitable for online speech recognizers. The size was reduced by decreasing *p*-norm input dimension from 3000 to 2750. This system has 7.7 million parameters This model was trained on augmented data, it was discriminatively trained and decoded using enhanced lexicon. It was able to achieve a word error rate of 11.0%, which is better than the result reported for unfolded recurrent networks in [13]. Table 3 shows the contributions of each technique.

Table 2: Performance comparison of DNN and TDNN with various temporal contexts

Model	Network Context	Layerwise Context					WER	
		1	2	3	4	5	Total	SWB
DNN-A	$[-7,7]$	$[-7,7]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.1	15.5
DNN-A_2	$[-7,7]$	$[-7,7]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	**21.6**	**15.1**
DNN-B	$[-13,9]$	$[-13,9]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.3	15.7
DNN-C	$[-16,9]$	$[-16,9]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.3	15.7
TDNN-A	$[-7,7]$	$[-2,2]$	$\{-2,2\}$	$\{-3,4\}$	$\{0\}$	$\{0\}$	21.2	14.6
TDNN-B	$[-9,7]$	$[-2,2]$	$\{-2,2\}$	$\{-5,3\}$	$\{0\}$	$\{0\}$	21.2	14.5
TDNN-C	$[-11,7]$	$[-2,2]$	$\{-1,1\}$	$\{-2,2\}$	$\{-6,2\}$	$\{0\}$	20.9	14.2
TDNN-D	$[-13,9]$	$[-2,2]$	$\{-1,2\}$	$\{-3,4\}$	$\{-7,2\}$	$\{0\}$	**20.8**	**14.0**
TDNN-E	$[-16,9]$	$[-2,2]$	$\{-2,2\}$	$\{-5,3\}$	$\{-7,2\}$	$\{0\}$	20.9	14.2

Table 3: Results on SWBD LVCSR task with data augmentation and enhanced lexicon

Acoustic Model + Language Model	WER	
	Total	SWB
TDNN - D + pp	21.9	14.8
TDNN - D + pp + fg	20.4	13.6
TDNN - D + pp + fg + sp + vp	19.2	12.9
TDNN - D + pp + fg + sp + vp + silp	19.0	12.7
TDNN - D + pp + fg + sp + vp + sequence training	17.6	11.4
TDNN - D + pp + fg + sp + vp + sequence training + pa	17.1	11
unfolded RNN + fMLLR features + iVectors [13]	-	12.7
unfolded RNN + fMLLR features + iVectors + sequence training [13]	-	11.3
CNN/DNN joint training + fMLLR features + iVectors [25]	-	12.1
CNN/DNN joint training + fMLLR features + iVectors + sequence training[25]	-	**10.4**

pp : pronunciation probabilities
fg : 4-gram LM rescoring
silp : word position dependent silence probabilities
sp : speed perturbation
vp : volume perturbation
pa : prior adjustment

5.1. Performance of TDNNs on various LVCSR tasks

Table 4: Baseline vs TDNN on various LVCSR tasks with different amount of training data

Database	Size	WER		Rel. Change
		DNN	TDNN	
Res. Management	3h hrs	2.27	2.30	-1.3
Wall Street Journal	80 hrs	6.57	6.22	5.3
Tedlium	118 hrs	19.3	17.9	7.2
Switchboard	300 hrs	15.5	14.0	9.6
Librispeech	960 hrs	5.19	4.83	6.9
Fisher English	1800 hrs	22.24	21.03	5.4

Experiments were done using Kaldi speech recognition toolkit [27] on Resource Management [28], Wall Street Journal [29], Tedlium [30], Switchboard [7], Librispeech [31] and the english portion of Fisher corpora [32]. The amount of training data available for acoustic modeling varies from 3-1800 hours across the setups mentioned. The recipes for these experiments are available in the Kaldi code repository [27] [2].

An average relative improvement 5.52% was observed over the baseline DNN architecture through the use of TDNN architecture to process wider contexts. It is to be noted that the number of parameters in the system are not matched between DNN and TDNN architectures. However the individual systems were tuned for best performance, given the architecture.

In the Resource Management medium-vocabulary task, we did not see gains from TDNNs. This could be due to the slight increase in parameters in the TDNN architecture when processing larger input contexts.

6. Conclusion

The effectiveness of TDNNs in processing wider context inputs was shown in small and large data scenarios. An input temporal context of $[t-13, t+9]$ was found to be optimal. Further using efficient selection of sub-sampling indices speed-ups were be obtained during training. An average relative improvement of 6% was reported across 6 different LVCSR tasks, compared with our previous DNN configuration. Our results are also 2.6% relative better than a previously reported result from the literature using an unfolded RNN architecture operating on speaker adapted features [13]. Our future work involves switching from the p-norm nonlinearity to ReLU, which according to some preliminary experiments seems to work better in the TDNN framework.

7. Acknowledgements

The authors would like to thank Pegah Ghahrmani for discussing results on comparison of p-norm and ReLU layers, in the context of TDNNs.

[2] e.g. `https://svn.code.sf.net/p/kaldi/code/trunk/egs/swbd/s5c/local/online/run_nnet2_ms.sh` in revision 5125

8. References

[1] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," Feb. 2014. [Online]. Available: http://arxiv.org/abs/1402.1128

[2] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, Mar. 1989.

[3] A. Waibel, "Modular construction of time-delay neural networks for speech recognition," *Neural computation*, vol. 1, no. 1, pp. 39–46, 1989.

[4] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted mmi for model and feature-space discriminative training," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2008, pp. 4057–4060.

[5] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q.-F. Liu, "Fast Adaptation of Deep Neural Network based on Discriminant Codes for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. PP, no. 99, pp. 1–1, 2014.

[6] M. Karafiat, L. Burget, P. Matejka, O. Glembek, and J. Cernocky, "iVector-based discriminative adaptation for automatic speech recognition," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, Dec. 2011, pp. 152–157.

[7] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: telephone speech corpus for research and development," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Mar 1992, pp. 517–520 vol.1.

[8] H. Hermansky and S. Sharma, "Temporal patterns (traps) in asr of noisy speech," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Mar 1999, pp. 289–292 vol.1.

[9] N. Mesgarani, S. Shamma, and M. Slaney, "Speech discrimination based on multiscale spectro-temporal modulations," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, May 2004, pp. I–601–4 vol.1.

[10] J. Andén and S. Mallat, "Deep scattering spectrum," *Signal Processing, IEEE Transactions on*, vol. 62, no. 16, pp. 4114–4128, Aug 2014.

[11] S. Thomas, S. Ganapathy, and H. Hermansky, "Phoneme recognition using spectral envelope and modulation frequency features," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2009, pp. 4453–4456.

[12] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2013, pp. 6645–6649.

[13] G. Saon, H. Soltau, A. Emami, and M. Picheny, "Unfolded recurrent neural networks for speech recognition," in *Proceedings of INTERSPEECH*, 2014.

[14] F. Grezl, M. Karafiat, and K. Vesely, "Adaptation of multilingual stacked bottle-neck neural network structure for new language," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 7654–7658.

[15] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2014, pp. 215–219.

[16] S. B. Davis and P. Mermelstein, "Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[17] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," *CoRR*, vol. abs/1410.7455, 2014. [Online]. Available: http://arxiv.org/abs/1410.7455

[18] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788 –798, May 2011.

[19] M. Gibson, "Minimum bayes risk acoustic model estimation and adaptation," Ph.D. dissertation, University of Sheffield, 2008.

[20] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks." in *Proceedings of INTERSPEECH*, 2013, pp. 2345–2349.

[21] V. Peddinti, G. Chen, D. Povey, and S. Khudanpur, "An i-vector based time delay neural network architecture for far field recognition," in *Proceedings of INTERSPEECH*, 2015. [Online]. Available: http://www.danielpovey.com/files/2015_interspeech_aspire.pdf

[22] V. Manohar, D. Povey, and S. Khudanpur, "Semi-supervised maximum mutual information training of deep neural network acoustic models," in *Proceedings of INTERSPEECH*, 2015. [Online]. Available: http://www.danielpovey.com/files/2015_interspeech_entropy.pdf

[23] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of INTERSPEECH*, 2015. [Online]. Available: http://www.danielpovey.com/files/2015_interspeech_augmentation.pdf

[24] G. Chen, H. Xu, M. Wu, D. Povey, and S. Khudanpur, "Pronunciation and silence probability modeling for ASR," in *Proceedings of INTERSPEECH*, 2015. [Online]. Available: http://www.danielpovey.com/files/2015_interspeech_silprob.pdf

[25] H. Soltau, G. Saon, and T. Sainath, "Joint training of convolutional and non-convolutional neural networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 5572–5576.

[26] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 24–29.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011.

[28] P. Price, W. Fisher, J. Bernstein, and D. Pallett, "The darpa 1000-word resource management database for continuous speech recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr 1988, pp. 651–654 vol.1.

[29] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[30] A. Rousseau, P. Deléglise, and Y. Estève, "Ted-lium: an automatic speech recognition dedicated corpus." in *LREC*, 2012, pp. 125–129.

[31] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.

[32] C. Cieri, D. Miller, and K. Walker, "The fisher corpus: a resource for the next generations of speech-to-text." in *LREC*, vol. 4, 2004, pp. 69–71.