An exploration of dropout with LSTMs

Gaofeng Cheng[1,3], *Vijayaditya Peddinti*[4,5], *Daniel Povey* [4,5], *Vimal Manohar* [4,5], *Sanjeev Khudanpur*[4,5], *Yonghong Yan*[1,2,3]

[1]Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics
[2]Xinjiang Laboratory of Minority Speech and Language Information Processing, Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences, China
[3]University of Chinese Academy of Sciences
[4]Center of Language and Speech Processing
[5]Human Language Technology Center Of Excellence, Johns Hopkins University, Baltimore, MD, USA

{gfcheng.cn,dpovey}@gmail.com,{vijay.p,vmanoha1,khudanpur}@jhu.edu,yanyonghong@hccl.ioa.ac.cn

Abstract

Long Short-Term Memory networks (LSTMs) are a component of many state-of-the-art DNN-based speech recognition systems. Dropout is a popular method to improve generalization in DNN training. In this paper we describe extensive experiments in which we investigated the best way to combine dropout with LSTMs– specifically, projected LSTMs (LSTMP). We investigated various locations in the LSTM to place the dropout (and various combinations of locations), and a variety of dropout schedules. Our optimized recipe gives consistent improvements in WER across a range of datasets, including Switchboard, TED-LIUM and AMI.

Index Terms: speech recognition, LSTM, DNN, dropout, lattice-free MMI

1. Introduction

The Long Short-Term Memory (LSTM) network [1, 2] is used in many state-of-the-art ASR systems [3], often in the popular 'projected' variant [4].

Dropout [5] is a mechanism to improve generalization of neural nets. It consists of multiplying neural net activations by random zero-one masks during training (random masks are not used in test time, but are approximated by a fixed scale). The dropout probability p determines what proportion of the mask values are one; the original paper suggested that $p = 0.5$ works well for a range of tasks. However, the experience of the speech community is that dropout implemented in this way does usually work for speech recognition tasks, and at the very least requires careful selection of the dropout probability and the use of a time-varying schedule (as is typical for negative results, it is hard to supply a citation for this).

This paper describes the outcome of extensive experiments in which we investigated the best way to use dropout with LSTMs. The issues we investigated are:

- Per-element (conventional) versus per-frame dropout
- Dropout-probability schedules
- Various locations for dropout (e.g. different LSTM gates or combinations thereof).

This work is partially supported by the National Natural Science Foundation of China (Nos.11590770-4, U1536117), the National Key Research and Development Plan (Nos.2016YFB0801203, 2016YFB0801200), the Key Science and Technology Project of the Xinjiang Uygur Autonomous Region (No.2016A03007-1), NSF Grant No CRI-1513128 and IARPA Contract No 2012-12050800010.

The experiments reported in this paper were performed with systems based on lattice-free MMI (LF-MMI) [6]. These systems use a frame shift of 30ms, as opposed to the 10ms of typical systems. We have no reason to believe that there would be any special interaction between dropout and the use of LF-MMI; we focus on the LF-MMI models because they are several times faster to train and give better performance than standard cross-entropy (CE)-trained systems.

In Section 2 we describe prior work. In Section 3 we discuss various details of the proposed method. Experimental setup is described in Section 4 and the results are presented in Section 5. Finally we present our conclusions in Section 6.

2. Prior work

2.1. Prior work on dropout for ASR

A few publications have used dropout for large-scale Automatic Speech Recognition (ASR) tasks. In [7], dropout is applied to a network based on Maxout nonlinearities. The dropout probability started at 0.5 and linearly decreased to 0.0 after 8 epochs, after which no dropout was used. In [8], dropout was applied to LSTMs at the point where the input comes from the previous layer (this is equivalent to our "Location 2" below). In that case, no dropout schedules were used. Improvements in frame accuracy were shown on a Google-internal dataset of Icelandic speech, but no WER results.

2.2. Projected LSTMs

Projected LSTMs (LSTMPs) [4] are an important component of our baseline system, and to provide context for our explanation of dropout we will repeat the equations for them; here x_t is the input to the layer and y_t is the corresponding output.

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{fr}r_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx}x_t + W_{cr}r_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}c_t + b_o) \quad (4)$$

$$m_t = o_t \odot \tanh(c_t) \quad (5)$$

$$p_t = W_{pm}m_t \quad (6)$$

$$r_t = W_{rm}m_t \quad (7)$$

$$y_t = (p_t, r_t) \quad (8)$$

where $\odot$ stands for element-wise multiplication; σ (the sigmoid function) and $\tanh$ are also applied element-wise. In all experiments reported here, p_t and r_t are one quarter of the cell dimension: for example, the cell dimension might be 1024, so p_t

and r_t would have dimension 256 and the output y_t would have dimension 512.

3. Methods proposed

In this section we introduce some terminology to describe the various forms of dropout that we experimented with.

3.1. Per-frame versus per-element dropout

One of the modifications to dropout that we tried is *per-frame* dropout. Unlike per-element dropout in which each element of the dropout mask is chosen independently, in per-frame dropout the entire vector is set to either zero or one. However, this is done separately for each individual part of the network, e.g. if dropout is used for multiple layers or gates then each would have an independently chosen dropout mask.

3.2. Dropout schedules

We experimented with various dropout schedules, and settled on schedules in which the dropout probability p is zero for a time at the start of training and at the end, and rises to a peak somewhere closer to the start of training. Bear in mind that in the Kaldi nnet3-based recipes we use here, the number of epochs is determined in advance (early stopping is not used). Therefore we are able to specify dropout schedules that are expressed relative to the total number of epochs we will use. We express the dropout schedule as a piecewise linear function on the interval $[0, 1]$, where $f(0)$ gives the dropout proportion at the start of training and $f(1)$ gives the dropout proportion after seeing all the data. As an example of our notation, the schedule '0@0,0.2@0.4,0@1' means a function $f(x)$ that is linearly interpolated between the points $f(0) = 0$, $f(0.4) = 0.2$, and $f(1) = 0$. We write this as '$0, 0.2@0.4, 1$' as a shorthand, because the first point is always for $x = 0$ and the last is always for $x = 1$.

After experimenting with various schedules, we generally recommend schedules of the form '0,0@0.2,p@0.5,0', where p is a constant (e.g. 0.3, 0.5 or 0.7) that is different in different models. Because these schedules end at zero, there is no need to do the normal trick of rescaling the parameters before using the models for inference. The optimal p for BLSTMPs is 0.15∼0.10, we use 0.10 here.

3.3. LSTMP with dropout

Some of the locations in the LSTMP laye where we tried putting dropout are as follows (see Figure 1 for illustration):

- Location 1 (dropout before the projection): replace equation (5) with:
  $m_t = (o_t \odot \tanh(c_t)) \odot m_{drop}^{(m_t)}$
- Location 2 (dropout on the output of the layer): replace equation (8) with:
  $y_t = (p_t, r_t) \odot m_{drop}^{(y_t)}$
- Location 3 (dropout on each half of the projection output): replace equation (6), (7) with:
  $p_t = (W_{pm} m_t) \odot m_{drop}^{(p_t)}$
  $r_t = (W_{rm} m_t) \odot m_{drop}^{(r_t)}$
- Location 4 (dropout on i, f and o): replace equation (1), (2), (4) with:
  $i_t = \sigma(W_{ix} x_t + W_{ir} r_{t-1} + W_{ic} c_{t-1} + b_i) \odot m_{drop}^{(i_t)}$

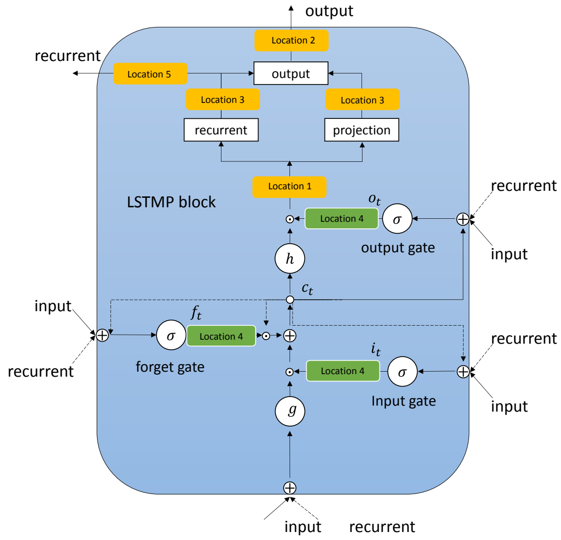

Figure 1: *Dropout location of LSTMP based RNNs architecture with peepholes[2] and LSTM projection[4]. Single memory block is shown for clarity and simplicity.*

$$f_t = \sigma(W_{fx} x_t + W_{fr} r_{t-1} + W_{fc} c_{t-1} + b_f) \odot m_{drop}^{(f_t)}$$
$$o_t = \sigma(W_{ox} x_t + W_{or} r_{t-1} + W_{oc} c_t + b_o) \odot m_{drop}^{(o_t)}$$

- Location 5 (dropout in the recurrence only): replace equation (7) with:
  $r_t = (W_{rm} m_t) \odot m_{drop}^{(r_t)}$

As alternatives to Location 4, we also tried dropout on all possible subsets of the i, f, o gates (experiments not shown), and found that dropping out all 3 was the best. We didn't try dropping out the output of either of the $\tanh$ gates, because it would have the same effect as dropping out the sigmoid gate that each one is multiplied by. In the rest of the paper, when we talk about per-frame dropout, we refer to the per-frame dropout location 4.

4. Experimental Setup

In this paper, HMM-DNN hybrid neural network acoustic models are used. The nnet3 toolkit in Kaldi speech recognition toolkit [9] is used to perform neural network training. The distributed neural network training algorithm is decsribed in [10]. LF-MMI [6] is the training criterion.

40-dimensional Mel-frequency cepstral coefficients (MFCCs) without cepstral truncation are used as the input into the neural network [10][11]. These 40-dimensional features are spliced across $\pm n$ (n may be 1 or 2) frames of context, then appended with a 100-dimensional i-vectors [12] to perform instantaneous adaptation of the neural network [13]. These i-vectors contain information about the mean offset of the speaker's data, so cepstral normalization is not necessary. In all cases we use the speed-perturbation method described in [14] to augment the data 3-fold. Our AMI setup is similar to that described in [15]; one notable feature is that alignments obtained using the individual headset microphone (IHM) data are used to train the single distant microphone (SDM) system.

Table 1: *Model configuration for TDNN-LSTMPs in SWBD, Tedlium and AMI*

	Switchboard/Tedlium		AMI	
Layer	Context	Layer-type	Context	Layer-type
1	[-2,-1,0,1,2]	TDNN	[-1,0,1]	TDNN
2	[-1,0,1]	TDNN	[-1,0,1]	TDNN
3	[-1,0,1]	TDNN	[-1,0,1]	TDNN
4	[0]	LSTMP	[0]	LSTMP
5	[-3,0,3]	TDNN	[-3,0,3]	TDNN
6	[-3,0,3]	TDNN	[-3,0,3]	TDNN
7	[0]	LSTMP	[-3,0,3]	TDNN
8	[-3,0,3]	TDNN	[0]	LSTMP
9	[-3,0,3]	TDNN	[-3,0,3]	TDNN
10	[0]	LSTMP	[-3,0,3]	TDNN
11	-	-	[-3,0,3]	TDNN
12	-	-	[0]	LSTMP

Table 2: *Dropout results on AMI IHM, dropout schedule is '0,0@0.20,p@0.5,0@0.75,0'*

Model	Dropout prob. p	Dropout location	per-frame dropout	WER (%) *dev*	WER (%) *eval*
Baseline				21.0	21.2
BLSTMP	0.3	location1	No	21.1	20.7
			Yes	20.8	20.8
		location2	No	21.1	21.3
			Yes	20.7	20.8
		location3	No	21.2	20.9
			Yes	20.6	20.7
		location4	No	20.8	20.8
			Yes	**20.6**	**20.2**
		location5	No	21.0	21.1
			Yes	20.9	20.8
	0.7	location1	No	21.0	20.8
			Yes	20.5	20.4
		location2	No	20.8	20.7
			Yes	20.6	20.3
		location3	No	20.8	20.9
			Yes	20.8	20.8
		location4	No	21.1	21.1
			Yes	21.2	20.8
		location5	No	21.2	21.0
			Yes	21.0	20.7

4.1. TDNN LSTMPs

A common trend in ASR modeling is to combine different types of layers [16] [17]. We have recently found that a combination of TDNN and LSTM layers can outperform BLSTMs. See Figure 2 for the illustration of a TDNN-LSTMP configuration. Adding per-frame dropout to the LSTMP units in TDNN-LSTMPs reduces the WER significantly. (Fast LSTMPs are efficient implementation of the LSTMPs in Kaldi, in which the 5 nonlinearities and 3 diagonal projections are segregated into a single operation).

4.2. Neural network configuration

We report results on two different model structures: BLSTMPs and TDNN-LSTMPs. Regarding the projection in the LSTM layers, the dimensions of p_t and the recurrence r_t are always one quarter the cell dimension, so for instance if the cell-dim is 1024, the recurrence would be of dimension 256 and the output y_t would be of dimension 512.

Our BLSTMP configuration on Switchboard consists of 3 BLSTMP layers, each with 1024 cells per direction. Our AMI BLSTMPs have 3 layers with 512 cells per direction (this

Table 3: *Dropout results on AMI SDM, dropout schedule is '0,0@0.20,p@0.5,0@0.75,0'*

Model	Dropout prob. p	Dropout location	per-frame dropout	WER (%) *dev*	WER (%) *eval*
Baseline				39.8	42.9
BLSTMP	0.3	location1	No	39.4	42.6
			Yes	38.4	42.0
		location2	No	39.2	42.3
			Yes	38.6	42.1
		location3	No	38.9	42.5
			Yes	38.6	42.2
		location4	No	38.7	42.2
			Yes	**38.2**	**41.5**
		location5	No	39.5	42.6
			Yes	39.0	42.2
	0.7	location1	No	38.9	42.3
			Yes	38.5	41.8
		location2	No	38.6	41.9
			Yes	38.3	41.6
		location3	No	38.9	42.6
			Yes	38.5	42.0
		location4	No	39.7	43.3
			Yes	39.2	42.5
		location5	No	38.9	42.4
			Yes	38.5	42.1

is probably not optimal in terms of WER; the configuration was originally chosen for speed). In all cases our TDNN-LSTMPs have TDNN layers with an output dimension of 1024 and LSTM layers with cell dimensions of 1024. The model details for our TDNN-(fast-)LSTMP systems of number of layers, time-delays and splicing are shown in Table 1.

5. Results

For reasons of space we only show a subset of our experiments, which were very extensive.

Tables 2 and 3 explore the various dropout locations, dropout probabilities, and per-frame versus per-element dropout masks, for BLSTMP models on AMI IHM and SDM respectively. Generally the trend is that per-frame dropout is better than per-element, and the best results are with location 4 and a relatively small maximum dropout probability, of 0.3.

Table 4 focuses only on per-frame dropout and location 4 (this time using both our old LSTM implementation and our new, 'fast' LSTM implementation, which only supports location 4 dropout). The experiments are again on AMI SDM and IHM. The important things to notice are that dropout is always helpful; and that, with dropout, the optimum WER seems to be reached at a later epoch (i.e. it keeps improving for longer). We use a smaller p value on the BLSTMP system, 0.1 vs. 0.3 for the TDNN-LSTMP models, as other experiments (not shown) indicated that this was better. The relative WER reduction from dropout is $\sim$3.0% on SDM and $\sim$4.5% on IHM on average.

Table 5 shows the same experiment on Switchboard and TED-LIUM, with the same conclusions. The relative WER reduction is $\sim 4.4\%$ on Hub5'00 and $\sim 5.1\%$ on the Train-Dev on average (Train-dev is an approximately 3 hour subset of the training data that we hold out). On TED-LIUM, the relative improvement is 7.2% on average with the model trained with by-frame dropout. Any differences between regular and "fast" LSTM are quite minor and likely due to statistical noise and some very minor differences in how the diagonal matrices of the LSTM are trained. The general conclusion that we draw is that our recipe works consistently using both LSTM implemen-

Table 4: *Per-frame dropout on AMI LVCSR, dropout schedule is '0,0@0.20,p@0.5,0'*

Model	Fast LSTMP	Dropout Location	Dropout prob. p	Epoch	SDM		IHM	
					dev	*eval*	*dev*	*eval*
BLSTMP	No	None	None	6	39.8	42.9	21.0	21.2
BLSTMP	No	Location4	0.1	6	**38.3**	**41.6**	**20.6**	**20.4**
TDNN-LSTMP	No	None	None	4	37.4	40.8	20.6	20.4
TDNN-LSTMP	No	Location4	0.3	4	**36.3**	**39.7**	**19.9**	**19.4**
TDNN-LSTMP	No	None	None	5	37.6	40.7	20.8	20.7
TDNN-LSTMP	No	Location4	0.3	5	**35.6**	**39.6**	**19.6**	**19.4**
TDNN-LSTMP	Yes	None	None	4	37.2	40.4	20.6	20.4
TDNN-LSTMP	Yes	Location4	0.3	4	**36.2**	**39.8**	**20.0**	**19.5**
TDNN-LSTMP	Yes	None	None	5	37.4	40.4	21.1	20.8
TDNN-LSTMP	Yes	Location4	0.3	5	**36.1**	**39.6**	**19.8**	**19.6**

Table 5: *Per-frame dropout on SWBD and TED-LIUM, dropout schedule is '0,0@0.20,p@0.5,0'*

Model	Fast LSTMP	Dropout Location	Dropout prob. p	Epoch	SWBD		TED-LIUM	
					Hub5'00	*Train Dev*	*Dev*	*Test*
BLSTMP	No	None	None	4	14.2	12.32		
BLSTMP	No	Location4	0.1	4	**13.6**	**12.13**		
BLSTMP	No	None	None	5	14.2	12.73		
BLSTMP	No	Location4	0.1	5	**13.8**	**11.84**		
TDNN-LSTMP	No	None	None	4	14.0	11.96	8.2	8.3
TDNN-LSTMP	No	Location4	0.3	4	**13.5**	**11.54**	**8.1**	**8.1**
TDNN-LSTMP	No	None	None	5	14.2	12.13	8.5	8.7
TDNN-LSTMP	No	Location4	0.3	5	**13.2**	**11.42**	**7.8**	**8.0**
TDNN-LSTMP	Yes	None	None	4	14.2	12.36	8.7	8.6
TDNN-LSTMP	Yes	Location4	0.3	4	**13.7**	**11.65**	**8.2**	**7.9**
TDNN-LSTMP	Yes	None	None	5	14.5	12.55	8.8	9.2
TDNN-LSTMP	Yes	Location4	0.3	5	**13.8**	**11.65**	**7.9**	**8.0**

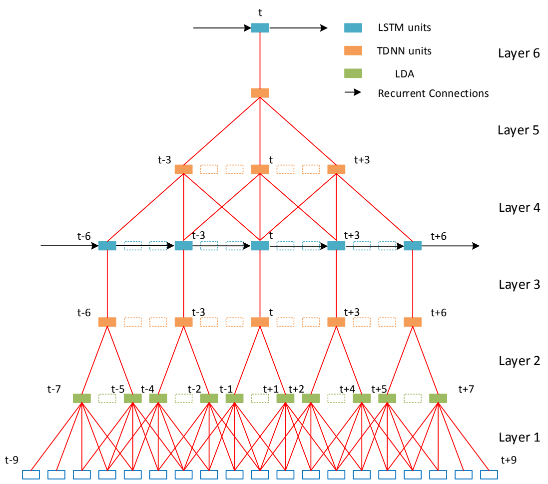

Figure 2: *Diagram of our TDNN-LSTMP configuration. The input context is* {*-2,-1,0,1,2*}*, and the splicing indexes for layer2, layer4 and layer5 are* {*-1,1*}*,* {*-3,0,3*}*,* {*-3,0,3*}*; LSTM time delay is 3.*

tations and a range of datasets. Of course we cannot guarantee that this would work in a setup that was very different from ours. For instance, our LSTMs generally operate with a recurrence that spans 3 time steps instead of 1.

We have not shown our experiments where we tuned the dropout schedule. We did find that it was important to leave the dropout proportion at zero for a certain amount of time at the beginning of training.

6. Conclusions

In this paper, we proposed per-frame dropout and investigated the best way to combine dropout with LSTMs. Our findings are:

- Per-frame, rather than per-element, dropout seemed to work best for us.
- We got the best results from applying dropout to the LSTM's o, f and i gates, with separate (per-frame) dropout masks for each.
- A dropout schedule that starts and ends at zero, and has a maximum value of 0.3 or less, worked well for us.
- For TDNN-LSTM combinations, we favor dropout only for LSTM layers (and not for TDNN layers).

With these configurations we observed relative WER reductions of between 3% and 7% on a range of LVCSR datasets. Our conclusions are a little different from [8], which recommends dropout only for the forward (non-recurrent) connections, like our Location 2; but we should point out that [8] did not report any experimental comparisons with alternative ways of doing dropout.

7. References

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

[2] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, pp. 115–143, 2002.

[3] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[4] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, pp. 157–166, 2014. [Online]. Available: http://arxiv.org/abs/1402.1128

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

[6] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," *Interspeech*, 2016.

[7] S. J. Rennie, P. L. Dognin, X. Cui, and V. Goel, "Annealed dropout trained maxout networks for improved lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5181–5185.

[8] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[9] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584, 2011.

[10] 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

[11] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proceedings of INTERSPEECH*, 2015, pp. 2440–2444.

[12] N. Dehak, P. J. 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.

[13] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[14] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *INTERSPEECH*, 2015.

[15] V. Peddinti, V. Manohar, Y. Wang, D. Povey, and S. Khudanpur, "Far-field asr without parallel data," in *Proceedings of Interspeech*, 2016.

[16] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4580–4584.

[17] L. Deng and J. Platt, "Ensemble deep learning for speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/ensemble-deep-learning-for-speech-recognition/