LINEARLY AUGMENTED DEEP NEURAL NETWORK

*Pegah Ghahremani**

Johns Hopkins University, Baltimore
pghahre1@jhu.edu

Jasha Droppo, Michael L. Seltzer

Microsoft Research, Redmond
{jdroppo,mseltzer}@microsoft.com

ABSTRACT

Deep neural networks (DNN) are a powerful tool for many large vocabulary continuous speech recognition (LVCSR) tasks. Training a very deep network is a challenging problem and pre-training techniques are needed in order to achieve the best results. In this paper, we propose a new type of network architecture, Linear Augmented Deep Neural Network (LA-DNN). This type of network augments each non-linear layer with a linear connection from layer input to layer output. The resulting LA-DNN model eliminates the need for pre-training, addresses the gradient vanishing problem for deep networks, has higher capacity in modeling linear transformations, trains significantly faster than normal DNN, and produces better acoustic models. The proposed model has been evaluated on TIMIT phoneme recognition and AMI speech recognition tasks. Experimental results show that the LA-DNN models can have 70% fewer parameters than a DNN, while still improving accuracy. On the TIMIT phoneme recognition task, the smaller LA-DNN model improves TIMIT phone accuracy by 2% absolute, and AMI word accuracy by 1.7% absolute.

Index Terms— Linear Augmented Network, deep network, pre-training, acoustic modeling

1. INTRODUCTION

Modern speech recognition systems employ deep neural network acoustic models using millions of parameters [1]. Deeper networks can represent certain function classes better than shallower ones and the use of deep networks can offer both computational and statistical efficiency for complex tasks [2]. Deep structure promotes re-use of features and learns a hierarchy of features, which leads to more abstract features at higher layers. More abstract concepts are invariant to most local changes of the input.

But, training a deep network with a large number of layers is a challenging problem. The gradient vanishing problem is the main issue, where the gradient vanishes exponentially with the number of layers it passes through. Unsupervised pre-training is often used to help training deep networks with improved parameter initialization [3]. Another useful approach is to change the loss function by introducing a companion objective to the individual hidden layers, in addition to overall objective at the output layer [4]. Exploiting certain activation functions [5] may help to train deeper networks. Maxout networks can work much better with increased depth that other activation functions, however [6] reported that training on CIFAR-10 with plain SGD was only possible for maxout networks with a depth up to 5 layers. Training of deeper networks was only possible with two-stage training and addition of soft-targets produced from a pre-trained teacher network.

Additionally, training and evaluating the DNN comes with a great computational cost. It has been shown that many parameters in DNN are very small and barely affect output values of each layer [7]. Re-structuring the DNN model based on sparseness of the original model using singular value decomposition (SVD) is used to decrease model size and computation during decoding [8]. However, this method does not reduce model size and computation time during training.

In this paper, we propose a novel type of network architecture with a specific SVD layer structure. This structure enables us to model the network with a substantial parameter count reduction. This architecture can model a very deep network without any pre-training. We show a theoretical connection between new model, pre-training, and how it handles the issue of gradient vanishing problem.

This paper is organized as follows. Section 2 describes our proposed Linear Augmented DNN model. Section 4 describes the training method used for training LADNN model, and Sections 5 presents the results on TIMIT [9] and AMI-HMI tasks, and Section 6 presents the conclusions.

2. LINEAR AUGMENTED DNN MODEL

The conventional DNN model is a composition of several layers, where each layer consists of an affine transformation followed by a nonlinear function. The output unit $\mathbf{y}$ in each layer can be described using the input vector $\mathbf{x}$, weight matrix $\mathbf{W}$, bias vector $\mathbf{b}$, and nonlinear function σ as

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1}$$

The SVD-DNN architecture employs SVD based layers, where the weight matrix in each layer is decomposed into two lower rank matrices with smaller dimensions. In the SVD-DNN, each nonlinearity is preceded by an affine transformation $\mathbf{U}\mathbf{x} + \mathbf{b}$, and followed by a linear transformation $\mathbf{V}$.

$$\mathbf{y} = \mathbf{V}\sigma(\mathbf{U}\mathbf{x} + \mathbf{b}) \tag{2}$$

If we consider this transformation to be the effective layer type for the SVD-DNN, it is clear that it uses a nonlinear mapping to transform from a continuous vector $\mathbf{x}$ to a continuous vector $\mathbf{y}$. If the optimal transformation for each layer contains a linear component, it would be difficult for this SVD-DNN layer type to learn. Nonlinear functions such as the sigmoid would need to allocate many parameters to deconstruct the vector space into overlapping regions, and then rebuild it again.

The proposed LA-DNN architecture is a straightforward modification of Equation 2. It connects the input $\mathbf{x}$ to the output $\mathbf{y}$ using a linear bypass connection. The linear bypass $\mathbf{T}$ can be modeled as

*The work of Pegah Ghahremani was performed during an internship at Microsoft Research.

the identity, diagonal, or a full transformation.

$$\mathbf{y} = \mathbf{V}\sigma(\mathbf{U}\mathbf{x} + \mathbf{b}) + \mathbf{T}\mathbf{x} \quad (3)$$

Figure 2 shows graphical representation of single layer in the LA-DNN model. We interpret the weight matrix $\mathbf{T}$ as weight factors for input dimensions, which determine the portion of linear and nonlinear terms for each output dimension. Using diagonal transformation provides a balance between flexibility and parameter counts in model and it is the best configuration in our experiments. Figure 3 represents the mean and variance of diagonal elements of for each layer in a fully trained LA-DNN model with diagonal bypass connections. The average value of diagonal terms decrease for layers closer to output layer, which shows these layers rely on the nonlinear components and model more complex functions.

The LA-DNN model can be interpreted as a combination of different layer-based classifiers, where the output layer as a final hypothesis is a weighted combination of these classifiers. Figure 1 shows LADNN structure as a composition of different layers. The output of network $\mathbf{y}$ can be written as

$$\mathbf{y} = \mathbf{C_0}\mathbf{x_1} + \mathbf{C_1}\mathbf{f_1}(\mathbf{x_1}) + ... + \mathbf{C_{n-1}}\mathbf{f_{n-1}}(\mathbf{x_{n-1}}) + \mathbf{f_n}(\mathbf{x_n}) \quad (4)$$

$$\mathbf{C_i} = \prod_{\mathbf{j=i+1}}^{\mathbf{n}} \mathbf{T_j} \quad (5)$$

where f_i is the function correspond to layer i and $\mathbf{x_i}$ is the input features for this layer.

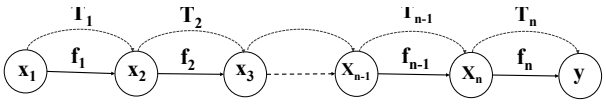

Fig. 1. Linearly Augmented DNN structure as a composition of different layers.

As discussed, the main problem in training very deep neural networks is the gradient vanishing problem [10]. For a very deep conventional DNN, the back-propagated error tends to vanish exponentially as it passes through more and more layers. We can mathematically show that LA-DNN model handles gradient vanishing problem. If the objective function $\mathcal{L}$ is a function of y, then gradient w.r.t parameter w_{ij}^k, which is w_{ij} in layer k is computed as

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^k} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{w_{ij}^k}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \sum_{l=k}^{n} C_I \frac{\partial \mathbf{f_l}(\mathbf{x_l})}{\partial \mathbf{w_{ij}^k}} + f_l \frac{\partial \mathbf{C_l}}{\partial \mathbf{w_{ij}^k}} \quad (6)$$

where

$$\frac{\partial \mathbf{f_l}(\mathbf{x_l})}{\partial \mathbf{w_{ij}^k}} = \frac{\partial \mathbf{f_l}(\mathbf{x_l})}{\partial \mathbf{x_l}} \prod_{m=k}^{l} \frac{\partial \mathbf{x_m}}{\partial \mathbf{x_{m-1}}} \quad (7)$$

So the gradient in LA-DNN model is computed as

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^k} = C_k \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \frac{\partial f(\mathbf{x_k})}{\partial \mathbf{w_{ij}^k}} + ... \quad (8)$$

Assuming the bypass connection as identity or diagonal matrix, C_k is the diagonal matrix with diagonal term in range c_{ii}^k is 0.8^{n-k} to 1.0. So the gradient $\frac{\partial \mathcal{L}}{\partial w_{ij}^k}$ has an extra term which is not exponentially decreasing with layer depth k, where in conventional DNN, the gradient is decaying exponentially as a function of layer depth.

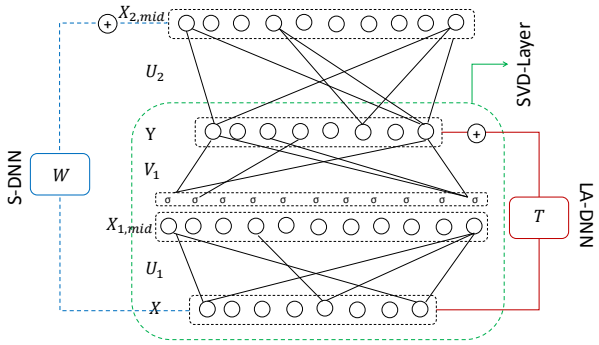

Fig. 2. Linearly Augmented DNN(LA-DNN) vs. Spliced DNN (S-DNN) structure.

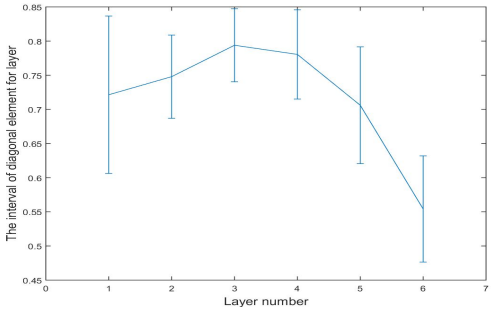

Fig. 3. Mean and Variance of diagonal weight for linear connection for different layers in final model.

3. SPLICED DNN MODEL

Figure 2 represents the structure of Spliced network, which is an extended form of LA-DNN model. In the LA-DNN model, $X_{2,mid}$ is modeled as

$$\mathbf{x_{2,mid}} = \mathbf{U_2}(\mathbf{V_1}\sigma(\mathbf{U_1}\mathbf{x} + \mathbf{b}) + \mathbf{T}\mathbf{x}) \quad (9)$$

We use tied transformation U_2 to model middle layer $\mathbf{x}_{2,mid}$ as a summation of linear part $\mathbf{U_1}\mathbf{T}\mathbf{x}$ and nonlinear part $\mathbf{U_2}(\mathbf{V_1}\sigma(\mathbf{U_1}\mathbf{x}+\mathbf{b}))$. Spliced DNN(S-DNN) uses untied transformation to combine the linear and nonlinear part and model $\mathbf{x_{2,mid}}$ as

$$\mathbf{x_{2,mid}} = \mathbf{U_2}(\mathbf{V_1}\sigma(\mathbf{U_1}\mathbf{x} + \mathbf{b})) + \mathbf{W}\mathbf{x} \quad (10)$$

where $\mathbf{W}$ is a full matrix. This type of network is called spliced DNN, since it splices the input of each layer $\mathbf{x}$ to the input of the next layer $f(\mathbf{x})$, and the output $\mathbf{y}$ is modeled as

$$\mathbf{y} = [\mathbf{U_2}\ \ \mathbf{W}] \begin{bmatrix} f(\mathbf{x}) \\ \mathbf{x} \end{bmatrix} \quad (11)$$

This Spliced DNN has different structure compared to Deep stacking networks [11]. In Deep stacking network, the input of layer l is the outputs of all previous layers stacked together, which increase input dimension, specially in very deep network such as network with 100 layers. The interpretation for expanding the space of output using input space is that some information can be lost from input space in each layer and by splicing the input. In the S-DNN, this information is directly transferred to the output space using linear transformation. The main difference between the LA-DNN and S-DNN is about their

training speed, where the LA-DNN model trains faster than S-DNN. In S-DNN model, the error back propagated using a full transformation matrix during error back-propagation, which slows down the training speed in this model compared to LA-DNN model. Also it uses full transformation as bypass connection in each layer, which increases the number of parameters in this model.

4. LA-DNN TRAINING

This section explains the key features used in training LA-DNN model. The new network architecture has higher learning capacity and it requires fewer parameters compared to the conventional DNN model. The training and validation cross-entropy for the LA-DNN model is much better than conventional DNN model after just few training iterations. Since the LA-DNN is much closer to the basin of attraction of local optimum than the DNN after few iterations, its initial learning rate should be smaller than baseline DNN to stabilize training.

The LA-DNN converges much faster than the baseline DNN, which can lead to over-training. We use number of methods to keep the training stable and avoid over-training in the LA-DNN. Firstly, we used the learning rate auto adjustment technique called "adjust after epoch" in the CNTK tool. It automatically reduces the learning rate by a constant factor, if after an epoch, the cross entropy on the validation set has degraded. Secondly, we enforce small changes in the parameters by gradient clipping, which controls gradient explosion, and employ L_2 regularization.

5. EXPERIMENTS

Experiments were done using CNTK toolkit[12] on TIMIT and AMI to evaluate the performance of proposed model. The input features are normalized by subtracting the mean and dividing by the standard deviation on each dimension to whiten the input features in all experiments. The model parameters are randomly initialized, and the simple SGD optimization method is used in all experiments. We used the Argon speech recognition decoder to decode our experiments, which is a dynamic WFST based decoder.

5.1. Experiment on TIMIT

We first evaluated the proposed model on TIMIT database [9]. The 25ms window size with frame shift 10ms used to extract MFCC features in our experiments. For the TIMIT configuration, the input layer contains 1080 neurons, which is $+/-7$ frames of 24 dimensional MFCCs with their first and second temporal derivatives spliced together. We used SGD as optimization method and we used learning rate auto-adjustment method. The initial learning rate per mini-batch for baseline and LA-DNN model are 1 and 0.4 respectively. Table 1 shows number of layers and layer dimensions in different experiments. As discussed, the proposed model trains faster than the baseline model and we should use smaller initial learning rate to stabilize the training. The best baseline DNN system used contains 2 hidden layers, each with 2048 units. The DNN parameters in both models are initialized randomly and cross entropy criterion used for training.

We did experiments using Sigmoid and Rectified linear unit [13] nonlinearities. As can be seen, the new model got 1.5% absolute WER improvement using networks with 70% fewer parameters. To show the power of LA-DNN model in modeling very deep structure using random initialization, we run a series of experiments with different number of layers. We keep the number of parameters constant in these experiments to investigate the effect of network depth and not involve the effect of adding more parameters. As can be seen, we get more PER improvement by training deeper network, which shows that we can have a more abstract and better generalized model by training deeper networks.

5.2. Experiment on AMI

We used AMI corpus [14] for our experiments, which is around 100 hours of meetings recorded in specifically equipped instrumented meeting rooms at three sites in Europe. We did our experiments on the IHM set.

Our DNN configuration contains 15 frames (7 frames on each side of current frame) of 24 dimensional MFCC feature with its velocity and acceleration. The output layer contain 5000 senones and Table 2 shows the number of hidden layers and their size in different experiments. The DNN is randomly initialized and no pre-training or speaker adaptation methods used in these experiments. The baseline DNN WER results with no pre-training has same WER results reported by [15], where they used stacked restricted Boltzman machines(RBMs) in a greedy layer-wise fashion [16] for parameter initialization. The same 50000 word AMI pronunciation dictionary as [17] used in these experiments. The Language model has 1.6 million trigram and 1.5 million bi-grams.

Table 2 compares different LA-DNNs and conventional DNN using different nonlinearities and same number of layers. As can be seen, the best baseline DNN system using Sigmoid nonlinearity has 37.6 million parameters and it gets 31.67 WER. The best baseline DNN system using ReLU nonlinearity gives 31.54% WER. The main problem with Rectified linear units and baseline model is about model depth, where the units in the first layers are mostly inactive or saturated and the parameters are not trained in the beginning layers. The LA-DNN architecture handles this issue using bypass connection and the best LA-DNN model uses Rectified linear units as nonlinearity. The best LA-DNN system has 10.5 million parameters with Rectified Linear units nonlinearity. This model achieves a word error rate of 30.68%, which is 0.86% absolute better than the best baseline DNN.

As can be seen, the S-DNN gives good improvement compared to the baseline DNN. It is less sensitive to over-training compared to LA-DNN model and it is required to use l2-regularization technique during training. On the other hand, S-DNN uses full transformation for transferring the information to the next layer, which results in slower convergence and larger model size compared to the LA-DNN. As can be seen, the best S-DNN has 13.1 million parameters and this model achieves word error rate of 30.44%, which is 1.1% absolute better than the best baseline DNN.

To verify the advantage of the proposed model in building very deep networks, we built different LA-DNN models with approximately the same number of parameters and different numbers of layers. Table 3 presents the experimental results, where it is clear that LA-DNN model overcomes the pre-training problem. The deeper, smaller networks have the best performance. The model using 48 layers got 29.9% WER, which is about 1.7% better than our best DNN model. We tried to train the baseline DNN with larger number of layers, but it degrades the results and we can not train very deep baseline network such as 12 layers starting from random initialization and without pre-training. We also tried to increase number of parameters for deepest network with 48 layers and as can be seen it give 0.28 WER improvement relative to a smaller network with same depth.

	Network Size			Training		Validation		
Model	Num.of.H Layers	Layer.Size	Params	CE	% Frame Err	CE	% Frame Err	% PER
DNN+Sig	2	2048X2048	**10.9M**	0.66	21.39	1.23	37.67	**23.63**
LA-DNN+Sig	4	1024X512	5.9M	0.61	20.5	1.18	35.83	22.39
LA-DNN+Sig	6	1024X512	8.0M	0.60	20.3	1.18	35.7	22.28
LA-DNN+Sig	8	1024X256	5.6M	0.58	19.7	1.19	35.7	22.08
LA-DNN+Sig	12	512X512	7.2M	0.65	22.08	1.16	35.74	22.08
LA-DNN+ReLU	3	1024X256	2.9M	0.61	20.7	1.20	35.77	22.39
LA-DNN+ReLU	6	1024X256	4.5M	0.54	18.6	1.22	35.5	22.08
LA-DNN+ReLU	12	512X256	3.8M	0.55	19.2	1.21	35.5	21.8
LA-DNN+ReLU	24	256X256	3.5M	0.55	19.31	1.21	35.3	22.06
LA-DNN+ReLU	48	256X128	**3.4M**	0.56	19.5	1.21	35.4	**21.7**

Table 1. Results on TIMIT using different models.

	Network Size			Training		Validation		
Model	Num.of.H Layers	Layer.Size	Params	CE	Frame Err	CE	Frame Err	WER
DNN+Sigmoid	6	2048X2048	**37.62M**	1.46	37.83	2.11	49.3	**31.67**
DNN+Sigmoid	6	1024X1024	12.52M	1.59	40.75	2.13	50.0	32.43
DNN+ReLU	6	1024X1024	12.52M	1.45	37.8	1.98	47.2	31.54
LA-DNN+Sigmoid	6	2048X512	18.4M	1.35	35.3			31.88
LA-DNN ReLU	6	1024X512	**10.5M**	1.34	35.77	2.02	47.3	**30.68**
Spliced-DNN+Sigmoid	4	1024X512	10M	1.53	39.2	2.08	49.13	31.86
Spliced-DNN+ReLU	6	1024X512	**13.1M**	1.42	37.0	1.95	46.8	**30.44**

Table 2. Results on AMI using different models.

	Network Size			Training		Validation		
Model	Num.of.H Layers	Layer.Size	Params	CE	% Frame Err	CE	% Frame Err	% WER
DNN+ReLU	6	1024X1024	12.52M	1.45	37.8	1.98	47.2	**31.54**
LA-DNN+ReLU	3	2048X512	12.1M	1.34	35.57	2.03	47.8	31.5
LA-DNN+ReLU	6	1024X512	10.5M	1.34	35.76	2.00	47.3	30.68
LA-DNN+ReLU	12	1024X256	8.9M	1.319	35.15	2.01	47.2	30.41
LA-DNN+ReLU	12	512X512	9.6M	1.34	35.77	1.98	46.9	30.22
LA-DNN+ReLU	24	512X256	8.2M	1.34	35.7	2.00	47.01	30.18
LA-DNN+ReLU	48	256X256	**7.9M**	1.35	35.9	1.97	47.01	**29.98**
LA-DNN+ReLU	48	512X256	**14.4M**	1.25	33.9	2.00	46.7	**29.7**

Table 3. Results on AMI using LA-DNN model with different number of layers.

6. CONCLUSION

In this work, we presented some promising results on using new proposed LA-DNN model for speech and phoneme recognition tasks. We show that we can get better results using a smaller network that converges faster than the baseline DNN model. More interestingly, we show that we can train very deep networks without pre-training, which is not possible using the regular DNN architecture. This capability in modeling deeper network with smaller number of parameters gives us the possibility to investigate the effect of depth and to construct a more abstract and generalized model.

7. REFERENCES

[1] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "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] Yoshua Bengio, Aaron Courville, and Pierre Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al., "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, pp. 153, 2007.

[4] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu, "Deeply-supervised nets," *arXiv preprint arXiv:1409.5185*, 2014.

[5] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[6] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[7] Dong Yu, Frank Seide, Gang Li, and Li Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4409–4412.

[8] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition.," in *INTERSPEECH*, 2013, pp. 2365–2369.

[9] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon Technical Report N*, vol. 93, pp. 27403, 1993.

[10] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[11] Li Deng, Xiaodong He, and Jianfeng Gao, "Deep stacking networks for information retrieval," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3153–3157.

[12] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, et al., "An introduction to computational networks and the computational network toolkit," Tech. Rep., Tech. Rep. MSR, Microsoft Research, 2014, http://codebox/cntk, 2014.

[13] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[14] Jonathan G Fiscus, Jerome Ajot, and John S Garofolo, "The rich transcription 2007 meeting recognition evaluation," in *Multimodal Technologies for Perception of Humans*, pp. 373–389. Springer, 2008.

[15] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals, "Hybrid acoustic models for distant and multichannel large vocabulary speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 285–290.

[16] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[17] Thomas Hain, Luká Burget, John Dines, Philip N Garner, František Grézl, Asmaa El Hannani, Marijn Huijbregts, Martin Karafiat, Mike Lincoln, and Vincent Wan, "Transcribing meetings with the amida systems," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 2, pp. 486–498, 2012.