INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Dynamic Layer Normalization for Adaptive Neural Acoustic Modeling in Speech Recognition

Taesup Kim[1], *Inchul Song*[2], *Yoshua Bengio*[1]

[1]MILA, Université de Montréal, Canada
[2]Samsung Advanced Institute of Technology, Republic of Korea
taesup.kim@umontreal.ca, inchul2.song@samsung.com, yoshua.bengio@umontreal.ca

Abstract

Layer normalization is a recently introduced technique for normalizing the activities of neurons in deep neural networks to improve the training speed and stability. In this paper, we introduce a new layer normalization technique called *Dynamic Layer Normalization* (DLN) for adaptive neural acoustic modeling in speech recognition. By dynamically generating the scaling and shifting parameters in layer normalization, DLN adapts neural acoustic models to the acoustic variability arising from various factors such as speakers, channel noises, and environments. Unlike other adaptive acoustic models, our proposed approach does not require additional adaptation data or speaker information such as i-vectors. Moreover, the model size is fixed as it dynamically generates adaptation parameters. We apply our proposed DLN to deep bidirectional LSTM acoustic models and evaluate them on two benchmark datasets for large vocabulary ASR experiments: WSJ and TED-LIUM release 2. The experimental results show that our DLN improves neural acoustic models in terms of transcription accuracy by dynamically adapting to various speakers and environments.

Index Terms: speech recognition, adaptive acoustic model, dynamic layer normalization

1. Introduction

Neural acoustic models have improved the transcription accuracy in speech recognition significantly over the past several years [1]. Recurrent neural networks, which have cyclic connections to hold long-term temporal contextual information, are a powerful tool for modeling sequence data such as speech. In particular, the Long Short-Term Memory (LSTM) architecture [2], which overcomes some modeling weaknesses of RNNs, has been shown to outperform DNNs and conventional RNNs for large vocabulary speech recognition [3, 4]. Despite this, neural acoustic models still suffer from the mismatch between training and testing environments. When a trained model is tested against unseen speakers or environments, its recognition accuracy can degrade substantially.

Adaptive acoustic modeling aims to adapt acoustic models to the acoustic variability across different speakers or environments. Approaches to the adaptation of neural acoustic models fall into two groups. In auxiliary feature-based adaptation [5, 6, 7], acoustic feature vectors are augmented by speaker-specific features such as i-vectors [8] computed for each speaker. On the other hand, in model-based adaptation [9, 10, 11], the model parameters are directly updated based on adaptation data. As shown in [12], model-based adaptation typically brings more improvement than auxiliary feature-based adaptation. However, model-based adaptation has some drawbacks that limit its applicability in practice. For example, adaptation data needs to be gathered for each new speaker and the model size grows as the number of speakers or environments increases.

Layer normalization [13] is a recently introduced normalization method to improve the training speed and stability for various neural network models. It fixes the mean and variance of the summed inputs within each layer and a pair of trainable scaling and shifting parameters are used to adjust the normalized values. In neural style transfer [14, 15, 16], a style transfer neural network is used to transfer an input image in the style of another one. Recently, it has been observed that, instead of training separate style transfer networks for each style being modeled, it is sufficient to specialize only the scaling and shifting parameters in instance normalization, which is similar to layer normalization, for each specific style [17]. Motivated by this work, we investigate the use of layer normalization as a way to adapt neural acoustic models to different acoustic styles arising from different speakers and environments.

In this paper, we introduce a new layer normalization technique called *Dynamic Layer Normalization* (DLN) for adaptive neural acoustic modeling in speech recognition. By dynamically generating the scaling and shifting parameters in layer normalization based on the input sequence, DLN adapts acoustic models to different speakers and environments. A feed forward neural network is introduced to extract from each input sequence an utterance summarization feature vector that is used to generate parameters in DLN. The whole network is jointly trained with gradient descent. Unlike other approaches in adaptive acoustic modeling, our proposed method does not require additional adaptation data or speaker information such as i-vectors. Moreover, the model does not need to be updated for each new speaker or environment as it dynamically generates adaptation parameters. We evaluate our proposed DLN applied on training deep bidirectional LSTM acoustic models on two benchmark datasets for large vocabulary ASR experiments: the Wall Street Journal [18] and TED-LIUM release 2 [19].

The rest of this paper is organized as follows. In Section 2, we discuss past research related to this work. In Sections 3 and 4, we describe our proposed method and experimental results, respectively. Finally conclusions follow in Section 5.

2. Related Work

Adaptive Acoustic Modeling Adaptive acoustic modeling can be broadly categorized into two groups: 1) auxiliary feature-based and 2) model-based adaptation. Most of auxiliary feature-based adaptation methods use i-vectors [8] as auxiliary features in addition to input acoustic features. I-vectors can be considered as basis vectors spanning a subspace of speaker variability. In [5, 6], i-vectors were used to augment the input acoustic features in DNN-based acoustic models and it was shown that appending i-vectors for each speaker resulted in im-

http://dx.doi.org/10.21437/Interspeech.2017-556

provements in the transcription accuracy. Tan et al [7] studied the speaker-aware training of LSTM acoustic models based on i-vectors.

On the other hand, model-based adaptation directly updates neural acoustic model parameters based on adaptation data. Liao [9] investigated speaker adaptation of DNN-based acoustic models using adaptation data through supervised and unsupervised adaptation and showed how L2 regularization on the speaker independent model improved generalization. In [10, 11], a speaker independent model was adapted to a specific speaker with speaker dependent parameters at each hidden layer. These parameters were estimated with adaptation data for each speaker and used to scale the hidden activations in the speaker independent model. Model-based adaptation typically brings more improvement than auxiliary feature-based adaptation as shown in [12]. However, adaptation data needs to be collected for each new speaker and speaker-specific parameters must be maintained and estimated for each speaker, which results in an increased model size.

Layer Normalization Layer normalization [13] was proposed to normalize the activities of neurons $x \in \mathbb{R}^N$ to reduce the covariate shift problem by fixing the mean and variance of x within each layer in deep neural networks. It can be defined as a linear mapping function LN with two sets of trainable parameters, scaling α and shifting β:

$$LN\left(x;\alpha,\beta\right) = \alpha \odot \left(\frac{x-\mu}{\sigma}\right) + \beta,$$
$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i-\mu_i\right)^2}$$

where x_i is the i^{th} element of x, μ and σ are the mean and standard deviation taken across the elements of x, respectively. x is first normalized with μ and σ and then scaled and shifted by $\alpha \in \mathbb{R}^N$ and $\beta \in \mathbb{R}^N$. The scaling and shifting parameters are learned along with the original model parameters to restore the representation power of the network. For example, by setting $\alpha = \sigma$ and $\beta = \mu$, the original activations can be recovered. Contrary to other normalization techniques such as batch normalization [20], it can be easily applied to recurrent neural networks since it performs exactly the same computation at training and test times. It has been shown that layer normalization is very effective at stabilizing the hidden state dynamics in recurrent neural networks.

Hypernetworks Hypernetworks [21] were proposed to dynamically generate the weights of neural networks through weight-generating sub-networks. The whole network is trained jointly with gradient descent. When applied to recurrent neural networks, the network weights can vary across different time steps. Hypernetworks are closely related to our work in that that they also generate some of model parameters. However, the goal of hypernetworks is to relax the weight-sharing property of recurrent neural networks to control the trade off between the number of model parameters and model expressiveness.

3. Proposed Model

3.1. Baseline Architecture

In this paper, we propose a new layer normalization technique called Dynamic Layer Normalization (DLN) and apply it to neural acoustic models based on Long Short-Term Memory (LSTM) [2]. LSTM has been shown to handle complex temporal dynamics in acoustic signals well. Among a number of variants of LSTM, we use the one proposed in [4] called LSTM with Recurrent Project Layer (LSTMP) that has an additional recurrent projection layer to reduce the model size by mapping the hidden state into a lower-dimensional space. To stabilize the hidden state dynamics and encourage faster convergence during training, layer normalization is applied. Thus, our baseline acoustic model is defined as a composite function as follows:

$$\begin{aligned}
i_t &= \sigma\big(LN\left(W_i x_t;\alpha_i,\beta_i\right) + LN\left(U_i h_{t-1};\alpha'_i,\beta'_i\right)\big)\\
f_t &= \sigma\big(LN\left(W_f x_t;\alpha_f,\beta_f\right) + LN\left(U_f h_{t-1};\alpha'_f,\beta'_f\right)\big)\\
o_t &= \sigma\big(LN\left(W_o x_t;\alpha_o,\beta_o\right) + LN\left(U_o h_{t-1};\alpha'_o,\beta'_o\right)\big)\\
c'_t &= \tanh\big(LN\left(W_{c'} x_t;\alpha_{c'},\beta_{c'}\right) + LN\left(U_{c'} h_{t-1};\alpha'_{c'},\beta'_{c'}\right)\big)\\
c_t &= f_t \odot c_{t-1} + i_t \odot c'_t\\
h_t &= W_p\left(o_t \odot \tanh\left(LN\left(c_t;\alpha_c,\beta_c\right)\right)\right)
\end{aligned} \tag{1}$$

where i_t, f_t, o_t, and c_t are input gate, forget gate, output gate, and cell state, respectively. Layer normalization LN is applied separately on input-to-hidden and hidden-hidden connections as proposed in [13][1]. $W_p \in \mathbb{R}^{d' \times d}$ is a linear projection that maps the hidden state into a lower d'-dimensional space, where d is the size of the cell state c_t. In this work, we do not use peephole connections.

In speech recognition, an input sequence $x = (x_1, x_2, ..., x_T)$ of length T, where x_t represents a frame-level acoustic feature vector, is given at once to the system. It is therefore beneficial not only to use previous context but also future context with bidirectional recurrent neural networks (BRNN) [3, 22]. Combining BRNN and LSTMP in a deep architecture, the l^{th} hidden layer is defined by the forward and backward LSTMPs whose outputs are concatenated and fed into the following layer:

$$h_t^l = \begin{bmatrix} \overrightarrow{h}_t^l \\ \overleftarrow{h}_t^l \end{bmatrix} \quad \begin{aligned} \overrightarrow{h}_t^l &= \text{LSTMP}_{\overrightarrow{\theta}^l}\left(h_t^{l-1}, \overrightarrow{h}_{t-1}^l\right) \\ \overleftarrow{h}_t^l &= \text{LSTMP}_{\overleftarrow{\theta}^l}\left(h_t^{l-1}, \overleftarrow{h}_{t+1}^l\right) \end{aligned}$$

where h_t^{l-1} is the output of the previous hidden layer $l-1$, $\overrightarrow{h}_t^l$ ($\overleftarrow{h}_t^l$) is the hidden state in the forward (backward) LSTMP with

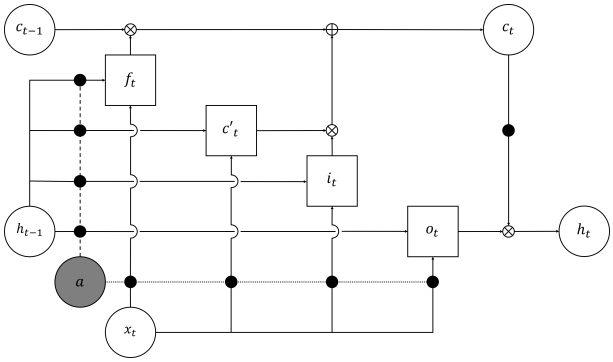


Figure 1: *Adaptation parameter generation process in dynamic layer normalization. The shaded circle a represents the utterance summarization feature vector. As indicated by dashed lines and small black circles, the vector a is used to generate the scaling and shifting parameters in layer normalization for the gates and cell state to dynamically adapt the model.*

[1] In our implementation, we follow the approach used in [13]. (`https://github.com/ryankiros/layer-norm`)

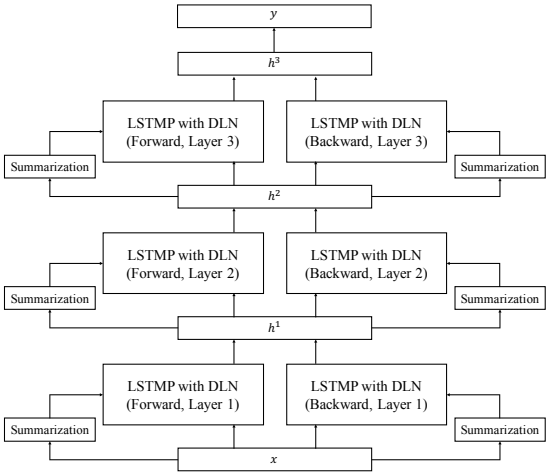


Figure 2: *The architecture of our proposed DLN applied to deep bidirectional LSTMP. Utterance summarization feature vectors are extracted in each layer and direction and used to dynamically generate the parameters of layer normalization for that layer.*

parameter set $\overrightarrow{\theta}$ ($\overleftarrow{\theta}$), and h_t^l is the output of the hidden layer l, at time step t.

The output layer is defined by an affine transformation followed by a softmax function:

$$y_t = \text{softmax}\Big(W_y h_t^L + b_y\Big)$$

where L is the total number of hidden layers. The output vector y_t represents the probability distribution over all possible labels. In this paper, we follow the the standard approach used in hybrid systems [23]. Frame-level state targets are provided on the training set by a forced alignment given by a GMM-HMM system. The softmax output layer has as many units as the total number of possible HMM states. The network is then trained by minimizing the negative log-likelihood of the frame-level target labels. As in [23], the posterior probabilities returned by the network are not divided by state priors during decoding.

3.2. Dynamic Layer Normalization

Based on the deep bidirectional LSTMP with layer normalization, we propose adaptive neural acoustic models that can adapt the model based on the input sequence to handle the acoustic variability in acoustic signals due to different speakers, channels and environments. Model adaptation is done by dynamically generating the scaling and shifting parameters in layer normalization based on the input sequence rather than learning them as other parameters in neural networks. For each input sequence, different layer normalization parameters are generated, which results in effectively adapting neural acoustic models to different input sequences.

To capture the acoustic variability in different layers and directions, each forward and backward LSTMP layer has a separate utterance-level feature extractor network, which is trained jointly with the main acoustic model. The utterance summarization feature vector a^l at layer l is extracted as follows:

$$a^l = \frac{1}{T}\sum_{t=1}^{T} \tanh\Big(W_a^l h_t^{l-1} + b_a^l\Big) \quad \text{where} \quad W_a^l \in \mathbb{R}^{p' \times d}$$

where a nonlinear transformation is applied to the output h_t^{l-1} of the previous hidden layer at each time step t, which is followed by average pooling to obtain a fixed-length vector. We set p' to be less than d to reduce the computational cost of parameter generation later. The utterance summarization feature vector a^l is then used to generate the scaling and shifting parameters, α_g^l and β_g^l, at layer l as follows:

$$\alpha_g^l = W_{\alpha g}^l a^l + b_{\alpha g}^l \qquad \beta_g^l = W_{\beta g}^l a^l + b_{\beta g}^l$$

where g is one of $\{i, f, o, c'\}$ in Equation 1. The process of dynamically generating adaptation parameters based on utterance summarization feature vectors is depicted in Figure 1.

Figure 2 shows the architecture of a deep bidirectional LSTMP with DLN. Note that DLN does not need any additional adaptation data or speaker information such as i-vectors. Moreover, the model size does not change because adaptation parameters are dynamically generated.

In order to extract more discriminative utterance summarization features that represent various factors in acoustic signals, we add a penalty term, L_{var}, to the loss to encourage each feature a_i^l in the utterance summarization feature vector a^l to be highly varied across the utterances within each mini-batch during training:

$$\text{L}_{\text{var}} = -\lambda \frac{1}{L}\sum_{l=1}^{L}\frac{1}{p'}\sum_{i=1}^{p'} \text{var}(a_i^l) \tag{2}$$

where the variance $\text{var}(\cdot)$ is computed over the minibatch, L is the total number of hidden layers, and λ is a hyperparameter that weights the contribution of the penalty relative to the loss.

Table 1: *Corpus Statistics*

Corpus		Train	Dev	Test
WSJ	# Utterances	37416 (81h)	503	333
	# Speakers	283	10	8
TED-LIUM Release 2	# Utterances	92973 (212h)	507	1155
	# Speakers	5076 (1495)	38 (8)	59 (11)

4. Experiments

4.1. Datasets

We evaluate our proposed methods on two benchmark datasets for large vocabulary automatic speech recognition experiments: the Wall Street Journal (WSJ) corpus [18] and TED-LIUM corpus release 2 [19]. The WSJ corpus primarily consists of read speech with texts drawn from a machine-readable corpus of Wall Street Journal news text. The TED-LIUM corpus release 2 is composed with segments of public talks extracted from the TED website. The collective summary of statistics for each corpora is given in Table 1.

For both datasets, each frame in the acoustic signal is represented by 40 log Mel-filterbank outputs (plus energy), together with their first and second derivatives. Each utterance is then represented as a sequence of frames where the size of each frame is 123.

4.2. Network Architecture and Training

All neural acoustic models in the experiments have three bidirectional LSTMP hidden layers, with 512 LSTM cells and 256 recurrent projection units in each of the forward and backward directions. Layer normalization is applied to all layers as in Equation 1, and only one bias term β (shifting parameter) for each $g \in \{i_t, f_t, o_t, c_t'\}$ is used in our implementation to avoid

Table 2: *Experimental results*

(a) *Wall Street Journal*

Model	Size	Dev FER Dev WER	Test FER Test WER
LSTMP w/ LN	10.44M	22.68% 7.26%	23.71% 4.50%
LSTMP w/ DLN	12.94M	21.81% 7.09%	23.35% 4.63%

(b) *TED-LIUM Release 2*

Model	Size	Dev FER Dev WER	Test FER Test WER
LSTMP w/ LN	10.81M	24.05% 14.18%	24.68% 13.50%
LSTMP w/ DLN	13.32M	23.27% 13.62%	23.82% 12.82%

unnecessary redundancy between β_i and β'_i. The Adam optimizer [24] is used for training models with the initial learning rate set to 0.001. The mini-batch size is set to 16. All weights are initialized by orthogonal initialization [25] and biases are set to zero. To reduce the computational cost of generating parameters in DLN, the size of utterance summarization feature vector, p', is set to 64. All models are implemented in Theano [26] using the Lasagne neural network library [27].

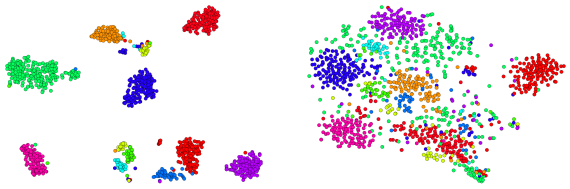
4.3. Results and Discussion

Wall Street Journal Experiments We follow the standard Kaldi recipe s5 [28] for preparing speech data. A baseline GMM-HMM system is trained on the 81 hours training set (train-si284) by Kaldi recipe tri4b, which consists of LDA preprocessing of data, with MLLT and SAT for adaptation. We then generate a forced alignment to obtain frame-level targets. There are 3436 triphone states in total. We use the dataset test-dev93 as the development set and test-eval92 as the test set.

Table 2 (a) shows the results of the experiments reported in terms of Frame Error Rates (FER) and Word Error Rates (WER). As shown in the table, the model trained with DLN outperforms the baseline model for both of the dev and test sets in terms of FER, but the improvement was not similarly shown on the WER. This is suspected that the WSJ corpus has a small number of speakers and was recorded under clean conditions that other environmental factors wouldn't effect the acoustic variability. Moreover, the proposed regularizer for DLN does not help much, so λ is set to 0.

TED-LIUM Experiments Speech data is prepared by following the standard Kaldi recipe s5_r2. The speakers are split up into 3-minute chunks for better generalization and fast per-utterance decoding. Table 1 shows both the increased number of speakers and the original number of speakers in parentheses. We first train a baseline GMM-HMM system on the 212 hours training set by Kaldi recipe tri3 and generate a forced alignment with 4174 triphone states in total. For DLN, we set λ to 10, which gave the best result on the dev set.

The experimental results are shown in Table 2 (b). The larger TED-LIUM corpus contains far more utterances and speakers than the WSJ corpus and was recorded from various environments. As shown in the table, the model trained with DLN is able to adapt to the high variability in the corpus and outperforms the baseline model on both of the dev and test sets.



(a) *First layer* (b) *Third layer*

Figure 3: *The utterance summarization feature vectors extracted from (a) the first layer and (b) the third layer in our proposed model are plotted using t-SNE in 2D-space. Each color represents a distinct speaker identity. The test set of the TED-LIUM corpus contains 11 different speakers.*

Discussion We visualize the utterance summarization feature vectors by using *t-SNE* [29], which is a technique for visualizing high dimensional data into 2D space. We use the test set of the TED-LIUM corpus and plot the feature vectors from the first and third layers. Figure 3 shows how the utterance summarization feature vectors are clustered and correlated with the speaker identity. The feature vectors from the first layer are clustered and highly correlated to speaker identities such that the number of clusters are similar to the original number of speakers, which is 11, even though no speaker information is used. On the other hand, the feature vectors from the third layer are more scattered. This can be interpreted that the feature vectors from the lower layers, which are closer to the input acoustic signals, represent speaker-related features and those from the higher layers are related to other factors.

For both of the datasets, we trained larger baseline models by adding one more layer to make their sizes similar to those trained with DLN. However, their performances degraded due to overfitting. On the other hand, DLN utilizes an increased model capacity effectively for generating adaptation parameters that led to performance improvements and empirically showed faster convergence as well during training.

5. Conclusions

In this paper, we have proposed a new layer normalization technique called DLN for adaptive neural acoustic model training in speech recognition. By dynamically generating scaling and shifting parameters for layer normalization based on the input sequence, DLN adapts neural acoustic models to various speakers and environments. Unlike other adaptive acoustic models, DLN does not require additional adaptation data or contextual information such as speaker identity. In addition, the model size does not increase as it dynamically generates adaptation parameters. We have shown through experimental evaluation that DLN improves neural acoustic models in terms of transcription accuracy.

As future work, we plan to investigate other ways to extract more useful summarization features from the input sequence to help generate adaptation parameters.

6. Acknowledgements

The authors would like to thank Aaron Courville for valuable comments and also acknowledge the support of the following agencies for research funding and computing support: NSERC, Samsung, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

7. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[3] A. Graves, S. Fernández, and J. Schmidhuber, *Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition*, 2005, pp. 799–804.

[4] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." in *Interspeech*, 2014, pp. 338–342.

[5] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *ASRU*, 2013, pp. 55–59.

[6] V. Gupta, P. Kenny, P. Ouellet, and T. Stafylakis, "I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6334–6338.

[7] T. Tan, Y. Qian, D. Yu, S. Kundu, L. Lu, K. C. Sim, X. Xiao, and Y. Zhang, "Speaker-aware training of lstm-rnns for acoustic modelling," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5280–5284.

[8] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech & Language Processing*, vol. 19, no. 4, pp. 788–798, 2011. [Online]. Available: http://dx.doi.org/10.1109/TASL.2010.2064307

[9] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7947–7951.

[10] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 171–176.

[11] P. Swietojanski, J. Li, and S. Renals, "Learning hidden unit contributions for unsupervised acoustic model adaptation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 8, pp. 1450–1463, 2016.

[12] Y. Miao, H. Zhang, and F. Metze, "Speaker adaptive training of deep neural network acoustic models using i-vectors," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 11, pp. 1938–1949, 2015.

[13] L. J. Ba, R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016. [Online]. Available: http://arxiv.org/abs/1607.06450

[14] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," *CoRR*, vol. abs/1603.03417, 2016. [Online]. Available: http://arxiv.org/abs/1603.03417

[15] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016. [Online]. Available: http://arxiv.org/abs/1603.08155

[16] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016. [Online]. Available: http://arxiv.org/abs/1607.08022

[17] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," *CoRR*, vol. abs/1610.07629, 2016. [Online]. Available: http://arxiv.org/abs/1610.07629

[18] 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.

[19] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the ted-lium corpus with selected data for language modeling and more ted talks." in *LREC*, 2014, pp. 3935–3939.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift." in *ICML*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456.

[21] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov 1997.

[23] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[25] M. Henaff, A. Szlam, and Y. LeCun, "Orthogonal rnns and long-memory tasks," *arXiv preprint arXiv:1602.06662*, 2016.

[26] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[27] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, D. M. de Almeida, B. McFee, H. Weideman, G. Takács, P. de Rivaz, J. Crall, G. Sanders, K. Rasul, C. Liu, G. French, and J. Degrave, "Lasagne: First release." Aug. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.27878

[28] 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. IEEE Signal Processing Society, 2011.

[29] L. van der Maaten and G. E. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.