EFFICIENT NON-LINEAR FEATURE ADAPTATION USING MAXOUT NETWORKS

Steven J. Rennie, Xiaodong Cui, and Vaibhava Goel

IBM Thomas J. Watson Research Center
NY, USA

{sjrennie, cuix, vgoel}@us.ibm.com

ABSTRACT

In this paper we present a simple and effective method for doing non-linear feature adaptation using Maxout networks. The technique overcomes the need to sample the partition function during training, and overcomes the need to compute the Jacobian term and its gradient for each training case. Results on the Switchboard 1 task demonstrate that the approach can improve a state-of-the-art hybrid ASR system that utilizes i-vectors.

Index Terms— Deep Neural Networks, Maxout networks, i-vectors, Jacobian, Speaker adaptation.

1. INTRODUCTION

The question of how to effectively adapt pattern recognition systems based on feed-forward deep neural networks (DNNs) when additional random variables, related to the task, are known, has received much attention recently [1–6]. In practice, simply adapting the parameters of a globally trained DNN to produce a conditional one is generally not effective, unless the adaptation is highly constrained, since such adaptation is essentially limited to reinforcing DNN's own predictions.

One approach that has proven to be very effective for both speaker recognition and speaker-adaptive ASR has been to augment the input data by concatenating a low-dimensional feature embedding derived from label-specific data [1, 7] . I-vectors, for example, utilize a simple generative model of the data to extract such features, by performing factor analysis on the means of each speaker w.r.t. one or more GMM-based universal background models (UBMs) [1], and remain a state-of-the-art technique for speaker adaptation today.

Adaptation based on generative models has the advantage that the model must adapt to explain the observed data rather than the DNN's predictions. In, addition, ASR system adaptation based on generative models are less sensitive to decoding errors, since the decoded words generally have high acoustic confusability with what was actually said.

Maximum likelihood linear regression (MLLR) methods [8, 9] are historically among the most successful techniques for adaptation in traditional ASR systems based on GMMs, and have been used in conjunction with regression trees over the set of Gaussians in the acoustic model to implement conditionally linear feature transformations. However, such transformations cannot be trivially converted into a feature transformation for use by a DNN (and as we will show, are easily outperformed by DNN-based feature transformations). Constrained MLLR (CMLLR), on the other hand, utilizes a single linear transformation and so can be implemented as a data transformation, and remains an important component of state-of-the-art DNN-based ASR systems.

A natural approach to generalizing CMLLR is to utilize a DNN to estimate a non-linear feature transformation [5]. However, this is not straightforward because the Jacobian term implied by introducing such non-linearities must be taken into account when the data transformation is being learned. In [5] the general problem is considered, and maximum likelihood non-linear transformations (MLNTs) are learned incrementally in network "blocks" by using importance sampling. In [5], the authors showed that the approach significantly outperforms CMLLR when utilized by a discriminatively trained, GMM-based ASR system. This general framework, however, is computationally intensive, because sampling (100 samples per Gaussian in [5]) must be utilized to estimate the partition function for each new training case.

In this paper, we propose a simple and effective approach to non-linear feature adaptation using Maxout networks, which we call feature Maxout networks (FMNs). This approach utilizes a cascade of network layers with the Maxout non-linearity, when each hidden layer is restricted to have d hidden units, where d is the input feature dimension. Since the network is conditionally linear for any given input, the Jacobian term and its gradient can be computed in closed form, and so sampling is trivially avoided. This configuration, furthermore, has the advantage that it is trivial to initialize from a linear model, and so the determinant and gradient of the effective Jacobian for each layer and training case can be approximated by that of the initial linear seed transformation, with negligible loss in performance—this eliminates the need to compute a unique Jacobian term for each new training sample.

Results on the the English Transtac task reported in [5] show that these networks perform on-par with the more general approach presented in [5], at a fraction of the computational cost. Results on the Switchboard 1 LVCSR task indicate that the transformations can even improve the WER performance of state-of-the-art hybrid systems that utilize i-vectors.

2. BACKGROUND: MAXIMUM LIKELIHOOD NON-LINEAR TRANSFORMATION (MLNT)

In [5], the authors introduce DNNs for ML feature transformation using GMM-HMMs. For a given feature transformation, $y = f(x)$, they consider learning f under the (conditional) model, $p(x) = \frac{\exp\{-E(f(x))\}}{Z}$, where $Z \equiv \int_x \exp\{-E(f(x))\}dx$ is the partition function. The gradient of (conditional) log-likelihood of the model L w.r.t. the DNN weights W of a given layer is:

$$\frac{\partial L}{\partial W} = \frac{\partial E}{\partial W} + \int_x p(x)\frac{\partial E}{\partial W}dx, \tag{1}$$

where $\frac{\partial E}{\partial W} = \frac{\partial f(x)}{\partial W}\Sigma^{-1}[f(x) - \mu]$ for Gaussian p(). Because $f(x)$ is non-linear, the integral in (1) (and Z) cannot be evaluated in closed

form even for Gaussian $p()$, and so the authors utilize importance sampling to estimate these quantities. Transformations are trained incrementally in blocks, using the following procedure:

1. Estimate a CMLLR transform for the current output of the network, $y_{n-1} = x_n$.
2. Initialize the next network block, $y_n = f(x_n)$, to the CMLLR transform (autoencoder, MSE criterion).
3. For each training case, Gaussian:
   (a) generate (100) samples in x_n using the CMLLR model as the proposal distribution, $q(x_n)$.
   (b) Re-weight samples by $\frac{p(x_n)}{q(x_n)}$ and approximate the gradient of the weights (i.e. the integral in 1) of block n.
   (c) Re-use these samples/weights to update the gradient of the weights of lower blocks $\forall i < n$ (a further approx.).
4. update the weights of the block using (S)GD.
5. repeat for each block added.

This approach significantly outperforms CMLLR when utilized by a discriminatively trained, GMM-based ASR system, but is clearly computationally intensive, because sampling (100 samples per Gaussian in [5]) must be utilized to estimate the gradient for each training case. For further details consult [5].

3. FEATURE MAXOUT NETWORKS (FMNS)

A limitation of the MLNT approach [5] is that for each Gaussian, the (gradient of) the partition function Z must be estimated for each new transformation at (training)/test time. An alternative approach, which leads to non-linear transformations that can be directly applied at decoding time like CMLLR, is to instead enforce differential conservation of probability [1]. The pdf of a feature space $x \in \mathbb{R}^n$ can be defined w.r.t. the pdf of a transformed feature space, $y = f(x)$, $y \in \mathbb{R}^n$, by:

$$p(x) = \left|\frac{\partial y}{\partial x}\right| p(y), \tag{2}$$

by conservation of probability, for an invertible transformation $f()$. For an invertible linear transformation, $y = Ax$, this reduces to:

$$p(x) = |A|\, p(y). \tag{3}$$

Similarly, it is immediately clear that for any mapping that is *conditionally* linear given x:

$$p(x) = |A_x|\, p(y), \tag{4}$$

where $A_x \equiv A(x)$. Therefore for conditionally linear transformations, the Jacobian term ($\log |A_x|$), and the gradient of the Jacobian term (A_x^{-T}) can readily be computed. Note however, that this implies that A_x must be invertible everywhere, and so DNNs based on rectified linear (ReLU) units are not, in general, suitable for the task. Maxout networks [11] generalize rectified linear ($max[0, a]$) units, utilizing non-linearities of the form:

$$s_j = \max_{i \in C(j)} a_i, \tag{5}$$

where the a_i are defined by inner products with the outputs of the layer below:

$$a_i = \sum_k w_{ik} x_k + b_i. \tag{6}$$

[1] Note that for linear models, these two approaches are equivalent.

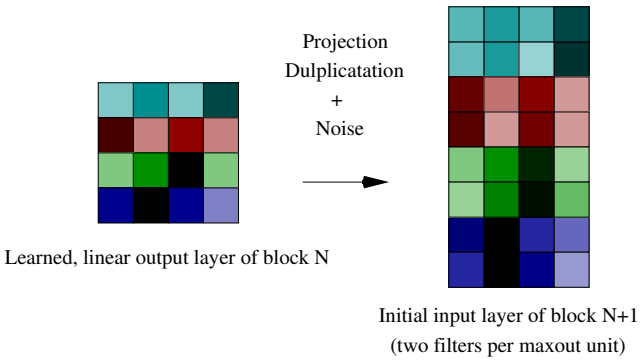

Fig. 1. As in [5] for MLNTs, FMNs are learned block by block. For FMNs, a new block is defined by evolving the linear output layer of the previous block into a maxout input layer for the current block, by duplicating projections and adding noise, as depicted. A new linear final layer is then initialized to the identity plus noise.

Maxout networks have been investigated by several ASR researchers and found to be very effective acoustic models [12, 13], particularly when trained using annealed dropout [14–16]. Here we utilize Maxout networks for maximum likelihood based feature adaptation.

Maxout units naturally have multiple detection "modes", and so are ideally suited for generalizing CMLLR. Specifically, if we restrict our hidden layers to have d hidden units and f filters per unit, we can initialize a Maxout hidden layer from a linear one by duplicating each row f times, and then adding noise:

$$a_i = a_j^{init} + \epsilon_i, \tag{7}$$

$\forall i \in C(j)$, $\forall j$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$. This process is depicted and described further in Figure 1. Note that this restricts only the number of hidden units that can be utilized—as many linear projections as desired can be taken by suitably defining f. This alleviates the need to pre-train the network to produce a desired initial transformation (e.g. identity or CMLLR), which is usually necessary to control overfitting, and is typically achieved by training the network to produce the corresponding initial target features.

An additional advantage of this approach is that when a Maxout network for feature transformation is restricted in this way, the determinant of the Jacobian can be approximated by the linear transform that it is initialized with. We have verified empirically that this results in negligible performance loss. This eliminates the need to compute the inverse of the Jacobian for every training case ($O(d^3)$ for feature dimension d), which means that we can optimize the transformation in an efficient manner using SGD even when d is large.

In summary, the process of building and FMN proceeds as follows:

1. Convert the final layer from linear to maxout as described above (CMLLR transform for first block).
2. Add a new linear layer above, this, together with the new maxout layer, defines a new network "block".
3. Train the entire network. $\frac{\partial L}{\partial A_x} = T_1 + T_2 = \frac{\partial \log p(y)}{\partial A_x} + A_x^{-T}$ for all layers & training cases, compute T_1 using backprop.
4. Repeat for the next block, until overfit is observed or a prespecified maximum number of blocks reached.

Note that, in contrast with MLNTs, which have and unconstrained hidden layer size, no pre-training of the new block is required.

4. EXPERIMENTS

Experiments were conducted on a Transtac English task, and the Switchboard 1 LVCSR tasks, which are described in the sections that follow, along with the models that were used for adaptation and decoding for these tasks. All FMNs were trained using stochastic gradient decent (SGD) with an initial learning rate of $1e^{-5}$, which was cut by a factor of a half whenever the (maximum likelihood) objective increased over an epoch of the test data for a given speaker. To be consistent with the usual process for training CMLLR and ensure stable adaptation, learning was done by computing GMM state posteriors using the initial model (CMLLR for the first block, the learned n-block model when learning block n+1), and then completing 1 or 2 complete (SGD-based) M-steps to update the parameters of the FMN (1 if not specified). For all experiments, the acoustic weight used by the baseline model when adapted with CMLLR was held fixed for all models, and the number of epochs used to adapt each FMN block was held fixed at 19, unless otherwise explicitly noted. This in some cases slightly reduced the gains afforded by FMNs, but makes the comparison of results more meaningful.

4.0.1. Transtac

We compare the test-time only adaptation performance of FMNs on the 11 speaker Transtac English task that MLNT was tested on in [5], which contains about 3-4 minutes of 16 KHz data recorded in a quiet environment, for each speaker. The baseline acoustic model utilized was trained under the boosted MMI (BMMI) criterion on feature MMI (FMMI) features [] using 60 hours of clean speech. The model has 3K quinphone states and 50K Gaussians.

4.0.2. Switchboard 1

The Switchboard 1 training data consists of 262 hours of segmented audio from English telephone conversations between two strangers on a pre-assigned topic. The test set is the Switchboard portion of the Hub5 2000 evaluation set which contains 2.1 hours (21.4K words, 40 speakers) of Switchboard data. The decoding vocabulary has 30.5K words and 32.9K pronunciations and all decodings were performed with a 4M 4-gram language model. The DNN acoustic models utilized for all experiments were annealed dropout trained Maxout networks with 4 hidden layers of 1414 units and a final hidden layer of 512 units, which is used to predict 9K context-dependent states. Dropout was applied to all hidden layers of the network, and annealed to zero over the first 10 of 25 epochs of training. These models were learned on ± 5 spliced, (CMLLR or FMN) adapted LDA features, with 100-dimensional i-vectors appended, as described in [1]. Feature transformations for Switchboard 1 were learned based on a 300K Gaussian FSA model over 40 dim. LDA features. Discriminative training of the hybrid DNNs was done using Hessian-free sequence (HF) training under the minimum Bayes risk (MBR) criterion [17], where noted.

5. RESULTS

5.1. Transtac

Table 1 compares the performance of FMNs with that of the general maximum likelihood non-linear transform (MLNT) approach reported in [5]. On this task FMNs perform on-par with the general MLNT approach, while alleviating the computationally expensive procedure of sampling the partition function for each training case and each Gaussian (100 samples per Gaussian were used in [5]). The use FMNs also alleviates the need to pre-train each new block to predict the features produced by FMLLR on the previous block, as described previously. We also investigated the performance FMNs as a function of the number of filters per hidden maxout unit on this task, and found that using more than 2 filters per unit did not lead to better adaptation performance.

Blocks	MLNT [5] (sampling)	FMNs f=2	FMNs f=4
1	24.7	24.5	24.6
2	24.2	24.3	24.3
3	24.0	24.0	24.2
4	23.7	23.9	24.0
5	**23.6**	**23.6**	23.7

Table 1. Word error rate (WER) as a function of number of blocks and filters per maxout unit (f) on a Transtac English task [5]. The baseline FMMI+BMMI GMM system WER is 25% after CMLLR (and 27.3% before).

5.2. Test-time only adaptation on Switchboard 1

Table 2 compares the performance FMNs to FMLLR and MLLR as implemented in the Attila Toolkit [18], when ML-trained FSA models are used to decode Hub5'00. The main purpose of this initial experiment was to see how FMN transformations compare with model space MLLR. Looking at the results, we can see that FMNs can match and then outperform FMMLR+MLLR with just one and two non-linear transformation blocks, respectively. On Hub5'00 this MLLR recipe estimates 5-7 transforms per speaker. One and two block FMNs with 2 filters per maxout unit conversely utilize the equivalent of 3 and 5 transformations per speaker in terms of total number of parameters, respectively.

Model	WER (%)
FMLLR (2 M-steps)	18.7
FMLLR+MLLR	18.2
FMN, 1 block (1 M-step)	18.4
FMN, 1 block, (2 M-steps)	18.1
FMN, 2 blocks (1 M-steps)	18.1
FMN, 2 blocks (2 M-steps)	17.8

Table 2. Test-time adaptation of an FSA model ML-trained on the Hub5'00 SWB task.

Model	WER (%)	Errors
Baseline	11.0	2352
Baseline FMLLR on HF txt	11.0	2365
FMN, 1 block (1 M-step)	10.9	2336
FMN, 1 block (1 M-step + 1 epoch)	10.9	2330
FMN, 1 block (1 M-step + 2 epochs)	10.9	2328
FMN, 1 block (1 M-step + 3 epochs)	**10.8**	**2323**
FMN, 1 block (1 M-step + 4 epochs)	10.8	2330

Table 3. Test-time adaptation of a DNN trained on FSA+ 100-dim i-vector features on the Switchboard 1 (300 hrs). The number of epochs of the second M-step that was executed are as indicated.

Model	WER (Errors)	
	Exact JT	Approx. JT
Baseline DNN	16.1% (3446)	-
FMN DNN, 1 block	15.9% (3400)	15.9% (3403)
FMN DNN, 2 blocks	15.7% (3374)	16.0% (3419)
FMN DNN, 3 blocks	15.7% (3359)	15.9% (3403)
FMN DNN, 4 blocks	15.6% (3340)	15.6% (3345)
FMN DNN, 5 blocks	15.8% (3389)	15.8% (3389)

Table 4. Multi-pass decoding using DNNs XE-trained on 50 hours of SWB1 data (transformed LDA features+100-dim i-vectors). Result using FMNs trained using the exact and linear approximation to the Jacobian term (JT) of each layer are depicted.

Table 3 investigates using FMNs for test-time only adaptation in conjunction with a state-of-the-art, HF-trained Maxout network that utilizes annealed dropout during training [14–16], and appends 100-dimensional i-vector features to (transformed) LDA features. The performance of this highly optimized network improves with test-time only FMN feature adaptation, which is encouraging. However, we observed that test-time only adaptation for DNNs using FMNs must be very carefully applied, since the features fed into the DNN are not exactly matched to those presented during training (these results are probably overtuned). Additional FMN blocks did not result in further performance improvements.

5.3. Train & test-time adaptation on Switchboard 1 50 hr subset

Table 4 compares the performance of FDNN features using annealed dropout, cross-entropy (XE) trained Maxout models trained on a 50 hour subset of Switchboard 1. Here FDNN-based speaker adaptation is applied at both training and test time, and at test time, FMN block n, is trained using the text hyp. generated by the DNN that utilizes FMN block $n-1$ to adapt its features. By using 4 FMN blocks, the WER of the baseline system on Hub4'00 is reduced by 0.5% absolute. Note that FMNs trained using a linear approximation to the Jacobian (greedily, by fixing lower layers) achieve very similar performance. Note also that for the FMNs trained in this paper that $d = 40$ and so the exact Jacobian term gradient can be computed efficiently relative to the cost of computing the DNN acoustic model.

Model	WER (Errors)	
	XE	HF
Baseline DNN	16.1% (3446)	14.4% (3084)
FMN DNN, 1 super-block	15.8% (3393)	14.2% (3046)

Table 5. Two-step decoding using DNNs (XE,ST)-trained on 50 hours of SWB1 data (transformed LDA features+100-dim I-vectors).

Table 5 compares the performance of FDNN features using annealed dropout, and cross-entropy (XE) and Hessian-free sequence (HF) trained Maxout models trained on a 50 hour subset of Switchboard 1. Here an FMN with two Maxout layers per block (rather than the usual 1) is utilized, which we call a superblock. Here we can see that 2-step decoding results in gains for both the XE and HF trained models.

Model	WER (Errors)	
	XE	HF
Baseline DNN	12.5% (2672)	11.0% (2359)
FMN DNN (4 blks.)	12.3% (2640)	10.9% (2334)
Baseline CNN	-	11.2% (2398)
Baseline DNN + CNN	-	10.0% (2153)
FMN DNN + CNN	-	9.9% (2126)

Table 6. Results on the full Switchboard 1 task. Here "+" denotes posterior averaging to derive likelihoods for the hybrid ASR system.

6. TRAIN & TEST-TIME ADAPTATION ON SWITCHBOARD 1

Table 6 reports results on on the full Switchboard 1 task. Again, small but consistent gains are realized using a four block FMN. Note that re-estimating the CMLLR transform used by the baseline DNN using the HF hyps. (here and throughout the paper) did not improve performance on Switchboard 1. The FMN-adapted DNN trained on FSA+i-vector features, when combined with an AD-trained Maxout CNN trained on log-Mel features (for details consult [14]), achieves a WER of 9.9%, which is a new record for the task.

7. DISCUSSION

In this paper we have presented a simple and efficient approach to doing non-linear feature adaptation using Maxout networks, which avoids sampling the partition function, and can avoid computing the Jacobian term for each new training case. We have shown that FMNs perform on-par with more general MLNTs, and shown that such adaptation can consistently (albeit slightly) improve the performance of a highly optimized hybrid DNN system without tuning the acoustic weight or number of iterations of adaptation applied per block. From a practical vantage point, all of our experimentation so far indicates that the technique is simple enough, fast enough, and robust enough to be a reliable component of commercial ASR systems in data-plenty scenarios.

Note however, that we have not characterized nor experimented with the performance of FMNs as a function of the amount of available adaptation data, as both the Switchboard 1 and Transtac datasets provide several minutes of data per train/test speaker. The investigation of early stopping approaches (e.g. number of blocks to learn for a given amount of data) and forms of maximum-a posteriori (MAP) adaptation [10] for FMNs in data-limited scenarios are interesting future research directions. Other possible future work includes utilizing FMNS in conjunction with more powerful generative models such as RBMs to adapt higher dimensional features, investigating the use of such transforms for other adaptation applications such as noise robustness, and exploring the possibility of deriving i-vector-like features from such transformations.

Two limitations of the FMNs presented in this work are that the network topology is highly constrained, and the adaptation parameters are not shared. Currently we are investigating the use of FMNs to generalize speaker-codes [7] to unsupervised (no reference text) adaptation scenarios.

8. ACKNOWLEDGEMENTS

The authors would like to thank George Saon, Samuel Thomas, and the anonymous reviews for helpful feedback.

9. REFERENCES

[1] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 55–59.

[2] Andrew Senior and Ignacio Lopez-Moreno, "Improving dnn speaker independence with i-vector inputs," in *Proc. ICASSP*, 2014.

[3] Yajie Miao, Hao Zhang, and Florian Metze, "Towards speaker adaptive training of deep neural network acoustic models," 2014.

[4] Pawel Swietojanski and Steve 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.

[5] Xiaodong Cui and Vaibhava Goel, "Maximum likelihood nonlinear transformations based on deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4320–4324.

[6] Roger Hsiao, Tim Ng, Stavros Tsakalidis, Long Nguyen, and Richard Schwartz, "Unsupervised adaptation for deep neural network using linear least square method," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[7] Shaofei Xue, Ossama Abdel-Hamid, Hui Jiang, Lirong Dai, and Qingfeng Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 1713–1725, 2014.

[8] Christopher J Leggetter and Philip C Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995.

[9] Mark JF Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[10] Olivier Siohan, Tor André Myrvoll, and Chin-Hui Lee, "Structural maximum a posteriori linear regression for fast hmm adaptation," *Computer Speech & Language*, vol. 16, no. 1, pp. 5–24, 2002.

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

[12] Yajie Miao, Florian Metze, and Seema Rawat, "Deep maxout networks for low-resource speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 398–403.

[13] Pawel Swietojanski, Jinyu Li, and Jui-Ting Huang, "Investigation of maxout networks for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7649–7653.

[14] George Saon, Hong-Kwang J Kuo, Steven Rennie, and Michael Picheny, "The ibm 2015 english conversational telephone speech recognition system," *arXiv preprint arXiv:1505.05899*, 2015.

[15] Steven J Rennie, Vaibhava Goel, and Samuel Thomas, "Annealed dropout training of deep networks," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 159–164.

[16] Steven J Rennie, Pierre L Dognin, Xiaodong Cui, and Vaibhava 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.

[17] Brian Kingsbury, Tara N Sainath, and Hagen Soltau, "Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[18] Hagen Soltau, George Saon, and Brian Kingsbury, "The ibm attila speech recognition toolkit," in *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, 2010, pp. 97–102.