INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Incorporating a Generative Front-end Layer to Deep Neural Network for Noise Robust Automatic Speech Recognition

Souvik Kundu[1], *Khe Chai Sim*[1], *Mark Gales*[2]

[1]School of Computing, National University of Singapore, Republic of Singapore
[2]Engineering Department, Cambridge University, Cambridge, CB2 1PZ U.K.
[1]{souvik,simkc}@comp.nus.edu.sg, [2]mjfg@eng.cam.ac.uk

Abstract

It is difficult to apply well-formulated model-based noise adaptation approaches to Deep Neural Network (DNN) due to the lack of interpretability of the model parameters. In this paper, we propose incorporating a generative front-end layer (GFL), which is parameterised by Gaussian Mixture Model (GMM), into the DNN. A GFL can be easily adapted to different noise conditions by applying the model-based Vector Taylor Series (VTS) to the underlying GMM. We show that incorporating a GFL to DNN yields 12.1% relative improvement over a baseline multi-condition DNN. We also show that the proposed system performs significantly better than the noise aware training method, where the per-utterance estimated noise parameters are appended to the acoustic features.

Index Terms: automatic speech recognition, noise robustness, deep neural network, vector Taylor series.

1. Introduction

The performance of both Deep Neural Network (DNN) and Gaussian Mixture Model (GMM) based acoustic models degrades significantly in the presence of background noise or channel mismatch conditions. This essentially necessitates noise adaptation. In traditional GMM-HMM systems, plenty of research has been done for noise robustness over the past two decades. Methods that have been proposed to address the noise robustness in GMM-HMM system can be categorized into feature enhancement and model-based adaptation [1] approaches. Typical feature enhancement techniques include denoising the speech features so that they match with the acoustic model trained on clean data, e.g. [2, 3]. On the other hand, model-based adaptation techniques distorts the clean model probability distribution based on the noisy test input feature, e.g. [4, 5]. However, better recognition accuracy is achieved when the relation between clean and noisy speech is exploited such as vector Taylor series (VTS)-based methods [6–10]. Traditional VTS approaches require a model trained on entirely clean data. To make use of the multi-condition data a noise adaptive training (NAT) approach is proposed in [11].

Recently, context-dependent DNN-HMM (CD-DNN-HMM) [12] has replaced the GMM-based systems due to the DNN's capability of modeling arbitrary non-linear distribution which in turn results better recognition accuracy. However, there is not much work done for noise robustness in DNN-based systems. Usually, feature enhancement techniques that were proposed for GMM-HMM system, can be applied to DNN-HMM system as well [13]. Another direction for noise robustness in DNN-based systems is network adaptation. One simple yet powerful approach is training the DNN with multi-style data [14]. In noise aware training (NaT) [14], noise parameters are estimated by averaging head and tail frames of each utterance and augmented with the regular features for DNN training. In NaT, authors claimed that the DNN can itself learn the complex relation between clean and corrupted speech due to the feature augmentation. However, our experimental results on Aurora-4 [15] do not validate the claim. Instead, we found that the system performs better if we augment the features with VTS estimated noise parameters. To fully benefit from both the powerful modeling capability of DNN and effective noise compensation of VTS, an adaptive training algorithm is proposed in [16]. Factorial Hidden Restricted Boltzman Machine (FHRBM) [17] was proposed to explicitly model the noise distribution and how the noise affects speech. Ideal Hidden-Activation Mask (IHM) [18] is inspired from the existing spectral masking techniques. Instead of masking the noise in spectral domain, IHM discards the DNNs inconsistent hidden activation units.

There is an increasing interest on combining the GMM and DNN based models for feature processing as well as adaptation. For instance, in [19], the temporally varying weight regression (TVWR) framework is proposed for unsupervised speaker adaptation. In [20], posterior values from both the feature space adapted DNN and GMM are combined at the state level during decoding. GMM-derived features have also been used in [21] and [22] for speaker adaptation in DNN-based system. So far, all the approaches proposed on combining GMM and DNN for acoustic modeling are primarily for speaker adaptation. In this paper, we propose a novel adaptation approach for noise robustness by incorporating a VTS adapted generative GMM-based front-end layer to DNN. We use log-likelihood scores of the GMMs as features for training the DNN. To make use of the multi-condition data, we use Noise Adaptive Training (NAT) [11] for adapting the front-end layer. We also experimentally justify the benefit of having an adaptable generative front-end layer by comparing with DNN-based noise aware training (NaT). From now on, we will refer our proposed approach as GFL-DNN as it is the combination of noise-adapted GMM-based Front-end Layer and DNN.

The rest of the paper is organized as follows: In Section 2, we introduce the concept of GFL-DNN. In Section 3, adaptation of the GFL is detailed. Experimental results are given in Section 4 and the paper is concluded in Section 5.

[1]This research was supported by the Google Faculty Research Award and the Singapore Ministry of Education Academic Research Fund Tier 2 (Official Project No: MOE2014-T2-1-068).

http://dx.doi.org/10.21437/Interspeech.2016-760

2. Generative Front-end Layer DNN

Our proposed GFL-DNN differs from the traditional CD-DNN-HMM system by having an adaptable generative GMM-based front-end layer.

2.1. Incorporating a generative front-end layer

One of the challenges with adapting DNN is due to the lack of meaningful interpretation of its model parameters, rendering it difficult to apply well-formulated algorithms such as the model-based noise compensation techniques. Applying Gaussian Mixture Models as a feature extractor provides an adaptable generative front-end layer (GFL) for the DNN. To derive the input features for the feed forward DNN, we rely on log-likelihoods of the GMMs. A typical GMM-HMM system consists of some HMM states where the observation probability of each HMM state is modeled by a GMM. Let us consider the pth state observation probability of an HMM is modeled with a GMM consisting of M_p Gaussians. Then the joint log-likelihood of the state p for a frame t of utterance u, represented as $\boldsymbol{y}_{u,t}$, can be given as:

$$l_{u,t}^{(p)} = \log\Bigg[\sum_{q=1}^{M_p} \frac{w_{p,q}}{(2\pi)^{\frac{D}{2}}|\boldsymbol{\Sigma}_{p,q}|^{\frac{1}{2}}} \times \exp\left(-\frac{1}{2}(\boldsymbol{y}_{u,t}-\boldsymbol{\mu}_{p,q})^T\boldsymbol{\Sigma}_{p,q}^{-1}(\boldsymbol{y}_{u,t}-\boldsymbol{\mu}_{p,q})\right)\Bigg], \quad (1)$$

where $l_{u,t}^{(p)}$ and D are the log-likelihood and feature dimension respectively. In this way, for each vector $\boldsymbol{y}_{u,t}$, we will have a P dimensional log-likelihood vector where P is the total number of GMMs. If the log-likelihood vector corresponding to $\boldsymbol{y}_{u,t}$ is given as $\boldsymbol{l}_{u,t}$, then $\boldsymbol{l}_{u,t} = [l_{u,t}^{(1)} \quad l_{u,t}^{(2)} \quad \dots \quad l_{u,t}^{(P)}]^T$. We also perform a speaker mean normalization on log-likelihood features before feeding to the DNN. Note that, co-variance of the Gaussian is always considered diagonal.

2.2. GFL as a polynomial kernel

The log-likelihood vector extracted from the GFL can be treated as a transformation on a polynomial kernel. If every HMM state observation probability is modeled with only one Gaussian ($M_p = 1$), the log-likelihood feature vectors can be obtained by applying a linear transform on the squared expansion of the regular features and augmented with an unit bias. If P be the total number of states, log-likelihood corresponding to state p, i.e. p^{th} Gaussian, can be given as:

$$l_{u,t}^{(p)} = r_p + \boldsymbol{s}_p^T\,\boldsymbol{y}_{u,t} + \boldsymbol{v}_p^T\,\boldsymbol{y^2}_{u,t}\ , \quad (2)$$

where

$$r_p = \log\left(\frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_p|^{\frac{1}{2}}}\right) - \frac{1}{2}\boldsymbol{\mu}_p^T\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\mu}_p\ , \quad (3)$$

$$\boldsymbol{s}_p = \begin{bmatrix} \frac{\mu_{p_1}}{\sigma_{p_1}} & \frac{\mu_{p_2}}{\sigma_{p_2}} & \dots & \frac{\mu_{p_D}}{\sigma_{p_D}} \end{bmatrix}^T\ , \quad (4)$$

$$\boldsymbol{v}_p = \begin{bmatrix} -\frac{1}{2\sigma_{p_1}} & -\frac{1}{2\sigma_{p_2}} & \dots & -\frac{1}{2\sigma_{p_D}} \end{bmatrix}^T\ , \quad (5)$$

and $\boldsymbol{y^2}_{u,t}$ is the square of input feature $\boldsymbol{y}_{u,t}$. Exploiting Eq. (1), the log-likelihood vector can be represented as:

$$\boldsymbol{l}_{u,t} = \boldsymbol{W}\,[1 \quad \boldsymbol{y}_{u,t} \quad \boldsymbol{y^2}_{u,t}]^T\ , \quad (6)$$

where $\boldsymbol{W}$ is a matrix constructed by Gaussian parameters and can be given as:

$$\boldsymbol{W} = \begin{bmatrix} r_1 & \boldsymbol{s}_1^T & \boldsymbol{v}_1^T \\ r_2 & \boldsymbol{s}_2^T & \boldsymbol{v}_2^T \\ \vdots & \vdots & \vdots \\ r_P & \boldsymbol{s}_P^T & \boldsymbol{v}_P^T \end{bmatrix} \quad (7)$$

In GFL-DNN, we use $(2D+1)$ HMM states to construct the front-end layer to preserve the full degree of freedom even if each of the HMM state observation probability is modeled with only one Gaussian where D is the feature dimension. As the MFCC-Δ-Δ features are 39 dimensional, we used 79 HMM states for adapting the front-end. If each HMM state observation probability is modeled with multiple Gaussians, the log-likelihood vector can be represented as a complex non-linear transform on the polynomial expansion of the feature vector and can be represented as:

$$\boldsymbol{l}_{u,t} = \Phi([1 \quad \boldsymbol{y}_{u,t} \quad \boldsymbol{y^2}_{u,t}]^T)\ , \quad (8)$$

where $\Phi(.)$ is a non-linear transform determined by the Gaussian parameters.

3. Adaptation of the GFL

3.1. Noise Adaptive Training with VTS compensation

In this section, we will discuss the steps of Noise Adaptive Training (NAT) [11] as it is an essential component in the proposed approach. Let us assume that a clean speech vector $\boldsymbol{x}$ is corrupted by environment distortions, resulting vector $\boldsymbol{y}$, can be given as:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{h} + g(\boldsymbol{n} - \boldsymbol{x} - \boldsymbol{h})\ , \quad (9)$$

where

$$g(\boldsymbol{n} - \boldsymbol{x} - \boldsymbol{h}) = \boldsymbol{C}\ \log(1 + \exp(\boldsymbol{C}^\dagger(\boldsymbol{n} - \boldsymbol{x} - \boldsymbol{h})))\ , \quad (10)$$

$\boldsymbol{h}$ and $\boldsymbol{n}$ are the channel distortion vector and additive noise vector respectively. $\boldsymbol{C}$ and $\boldsymbol{C}^\dagger$ are discrete cosine transform (DCT) and inverse discrete cosine transform (IDCT) matrices respectively. Note that, Eq. (9) is valid for MFCC features. After applying VTS, as given in [23], the mean update equations of the qth Gaussian belongs to the pth state of an HMM can be given as:

$$\boldsymbol{\mu}_{\boldsymbol{y}}^{(pq)} \approx \boldsymbol{\mu}_{\boldsymbol{x}}^{(pq)} + \boldsymbol{\mu}_{\boldsymbol{h}} + g(\boldsymbol{\mu}_{\boldsymbol{n}} - \boldsymbol{\mu}_{\boldsymbol{x}}^{(pq)} - \boldsymbol{\mu}_{\boldsymbol{h}}) \quad (11)$$

$$\boldsymbol{\mu}_{\boldsymbol{\Delta y}}^{(pq)} \approx \boldsymbol{J}_{(pq)}\boldsymbol{\mu}_{\boldsymbol{\Delta x}}^{(pq)} \quad (12)$$

$$\boldsymbol{\mu}_{\boldsymbol{\Delta\Delta y}}^{(pq)} \approx \boldsymbol{J}_{(pq)}\boldsymbol{\mu}_{\boldsymbol{\Delta\Delta x}}^{(pq)} \quad (13)$$

In Eqs. (11)-(13), $\boldsymbol{J}_{(pq)}$ is the Jacobian for the qth Gaussian of the pth state w.r.t. $\boldsymbol{x}$ or $\boldsymbol{h}$. The variance update equation can be given as:

$$\boldsymbol{\Sigma}_{\boldsymbol{y}}^{(pq)} \approx f(\boldsymbol{J}_{(pq)}, \boldsymbol{K}_{(pq)}, \boldsymbol{\Sigma}_{\boldsymbol{x}}^{(pq)}, \boldsymbol{\Sigma}_{\boldsymbol{n}})\ , \quad (14)$$

where

$$f(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}, \boldsymbol{d}) = \text{diag}(\boldsymbol{A}\boldsymbol{c}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{d}\boldsymbol{B}^T) \quad (15)$$

and $\boldsymbol{K}_{(pq)}$ is the Jacobian w.r.t. $\boldsymbol{n}$ and is equal to $(\boldsymbol{I} - \boldsymbol{J}_{(pq)})$. $\boldsymbol{\Sigma}_{\boldsymbol{\Delta y}}^{(pq)}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Delta\Delta y}}^{(pq)}$ can be updated similarly. Note that, additive noise mean is considered static and channel distortion is considered invariant.

In standard VTS, a clean canonical model is required for compensation. To make use of the multi-condition data, NAT [11] is proposed where a pseudo clean model is derived iteratively from the multi-condition model. Key steps of NAT are illustrated in Fig. 1. In NAT, a GMM-HMM system is trained from the multi-condition data which is treated as an initialized model. Then the noise parameters are initialized for every utterance - channel mean is zero and mean and variance of the additive noise is the mean and variance of the head and tail frames

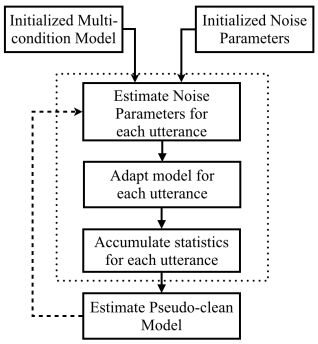

Figure 1: *Schematic diagram of Noise Adaptive Training.*

respectively. After the initialization, additive and channel noise parameters are re-estimated. Based on the re-estimated noise parameters, the model is compensated according to Eqs. (11)-(14) for every utterance. A pseudo clean model is derived based on the statistics accumulated from all the utterances in the training set. As given in Fig. 1, noise parameter re-estimation and pseudo clean model derivation is done in an iterative fashion until likelihood converges. The derived pseudo clean model at the end can be treated as a canonical model in test time during standard VTS compensation. The detailed derivation steps can be found in [11, 23].

3.2. Noise adaptation of the GFL

In the proposed GFL-DNN system, we use NAT [11] for adapting the GMM-based generative front end layer. Based on the derived pseudo clean model after few iterations of NAT, noise parameters are estimated by a two-pass decoding for the multi-condition test data. Before extracting the log-likelihood features, we compensate the pseudo clean model based on the estimated noise parameters. As we estimate the noise parameters for each utterance and compensate the pseudo clean model, transformations in Eq. (6)-(8) will be noise dependent and utterance specific. Similar to Eq. (6)-(8), log-likelihood vectors can be represented as:

$$\boldsymbol{l}_{u,t} = \boldsymbol{W}^{(n)} [1 \quad \boldsymbol{y}_{u,t} \quad \boldsymbol{y^2}_{u,t}]^T \tag{16}$$

$$\boldsymbol{l}_{u,t} = \Phi^{(n)}([1 \quad \boldsymbol{y}_{u,t} \quad \boldsymbol{y^2}_{u,t}]^T) \ , \tag{17}$$

where $\boldsymbol{W}^{(n)}$ is noise adapted linear transform and $\Phi^{(n)}(.)$ is noise adapted non-linear transform. Note that, both $\boldsymbol{W}^{(n)}$ and $\Phi^{(n)}(.)$ are utterance specific.

4. Experiments

4.1. Experimental Setup

We used Aurora-4 corpus [15] to justify the effectiveness of the proposed GFL-DNN system. In Aurora-4 corpus, two channel recordings were made at 16 kHz. Channel 1 is same microphone for all speakers. Channel 2 is chosen by sampling from a set of 18 different microphones. There are 7 background noise conditions for each channel recording type which are matched for the train and test data. The training set is multi-conditional. For experimentation, the test set is divided into four subsets: A (Clean Speech and Channel 1), B (Noisy speech and Channel 1), C (Clean speech and Channel 2) and D (Noisy speech and Channel 2). We report word error rate (WER) for each of the set as well as for the entire set. In this paper, we used Kaldi [24] and CNTK [25] for training and evaluating the models.

We use MFCC-Δ-Δ feature to construct the baseline. The features are 39 dimensional and consist of 13 static, 13 delta and 13 acceleration coefficients. We perform an 11 frame context

Table 1: *Description of two different noise estimators.*

Estimator	Description
N1	We estimate the additive and channel noise parameters by applying VTS on top of a GMM-HMM system consists of around 2500 HMM states and 15K Gaussians in total. We compensate the GFL based on this.
N2	Noise estimator and GFL are same. We iteratively perform NAT to estimate the noise parameters and the pseudo clean model. The derived pseudo clean model is then treated as GFL to extract log-likelihood features.

Table 2: *WER obtained using MFCC-Δ-Δ feature (Baseline), polynomial expansion of the same and log-likelihood features extracted from the GFL (GFL-DNN) when it can be expressed as a linearly transformed kernel.*

System	Adap-tation	WER (%)				
		A	B	C	D	Avg.
Baseline	–	3.3	7.8	7.5	19.3	12.4
Feature Expansion	–	3.3	7.9	7.9	19.6	12.6
GFL-DNN	NO	3.3	8.4	7.5	20.0	12.9
GFL-DNN (N1)	YES	3.2	7.8	7.0	17.3	**11.5**
GFL-DNN (N2)	YES	3.6	8.1	8.2	18.6	12.3

expansion and a per dimension mean variance normalization before feeding the features to DNN. The baseline multi-condition DNN configuration consists of 429 input units (39×11), 7 hidden layers each consists of 2048 units and an output layer consists of 2031 units. We used ReLU for activation function. Dropout and momentum were also incorporated during the training of DNN. As we can see in Table 2, the average baseline performance we obtained is 12.4% WER. Note that, for rest of all the experiments we do not change the size of context window and DNN configuration except the input layer size. The log-likelihood features are 79 dimensional. For additive noise parameter initialization we use first and last 20 frames for every utterance.

4.2. Proposed GFL-DNN performance

In this section we analyze the performance of GFL-DNN system. Note that, in Table 2 and 3, when we perform noise adaptation, we use two types of noise estimator namely N1 and N2. Description of N1 and N2 is given in Table 1.

Table 2 shows the performance of baseline system, with expanded features and when the GFL can be expressed as a linear transformation on the expanded features as described in Section 2.2. It is shown that comparable performance to the baseline is achieved when expanded features, i.e. $([1 \quad \boldsymbol{y}_{u,t} \quad \boldsymbol{y^2}_{u,t}]^T)$ are used for training the DNN. However, performance degrades when a Gaussian parameter based linear transformation is applied on the expanded features to get the log-likelihood features for training the DNN. Significant performance improvement is obtained when we perform noise adaptation on the GFL, i.e. apply a noise-adapted per-utterance linear transformation on the expanded features. Better performance is obtained in case of N1 compared to N2 as N1 is providing a more robust noise estimation due to having a large number of Gaussians.

Table 3 shows the results when log-likelihood features are extracted from the GFL where each GMM consists of multi-

Table 3: *WER obtained using log-likelihood features extracted from the GFL (GFL-DNN) when it can be expressed as a non-linearly transformed kernel.*

System	Adaptation	WER (%)				
		A	B	C	D	Avg.
Baseline	–	3.3	7.8	7.5	19.3	12.4
GFL-DNN	NO	3.9	8.7	8.8	20.7	13.5
GFL-DNN (N1)	YES	3.3	7.7	7.0	16.0	**10.9**
GFL-DNN (N2)	YES	3.3	7.8	6.5	15.9	**10.9**

Table 4: *Performance comparison between Noise aware Training (NaT) and adapted GFL-DNN system. "NO" in "VTS" field signifies that only additive noise parameters are estimated by averaging the head and tail frames of every utterance.*

System	VTS	WER (%)				
		A	B	C	D	Avg.
Baseline	–	3.3	7.8	7.5	19.3	12.4
NaT	NO	3.3	7.8	7.6	19.1	12.3
	YES	3.4	7.4	7.2	17.9	11.6
Adapted GFL-DNN	NO	3.9	8.5	8.4	19.1	12.7
	YES	3.3	7.7	7.0	16.0	**10.9**

ple Gaussians and can be expressed as given in Eq. (8) and (17). In Table 3, the GFL used for log-likelihood feature extraction consists of 79 GMMs and around 4K Gaussians in total. Similar to linear transformation based GFL, we obtain a performance degradation when only Gaussian parameter based non-linear transformation is applied on the expanded features. Performance improves significantly when noise adaptation is performed on the GFL, i.e., extracting log-likelihood based on Eq. (17). In Table 3, same average error rate is obtained using N1 and N2 noise estimator. Hence it can be concluded that, if we apply NAT on a GFL with larger number of Gaussians, it can provide similar noise estimation to the VTS-based bigger GMM-HMM system used in N1 noise estimator. We obtained similar performance by varying the number of NAT iterations. This might be due to the fact that the variations arise due to different number of noise estimation iterations, can be learned by the DNN.

Table 2 and 3 show that performance degrades if linear or non-linear transformation is applied on the expanded features without applying adaptation. This might be due to the fact that the DNN is not being able to reconstruct the transformation during training. When we are updating the GFL parameters for a particular utterance by maximizing the likelihood, log-likelihood vector corresponding to every frame provides more informative feature representation which becomes helpful for the DNN and in turn resulting better recognition accuracy. If we compare the performance of GFL-DNN of Table 2 and 3 without adaptation, we can see that degradation is higher in case of non-linear transformation as it is harder for the DNN to reconstruct the non-linear transformation during training. However, when noise adaptation is performed, non-linear transformation based log-likelihood features provide significantly better recognition accuracy. This is because of using a larger number of Gaussians facilitate generating more robust per-utterance based transformations.

4.3. Performance Comparison with Noise Aware Training

One simple yet powerful approach for adapting DNN is noise aware training (NaT). The NaT proposed in [14] only considers the mean of additive noise parameter. Per utterance additive noise mean vector is initialized as an average of head and tail frames. The authors claimed that, due to augmentation of the additive noise mean vector with regular features during the DNN training, DNN will itself learn the relation given in Eq. (9). Table 4 shows that better recognition accuracy can be acheived if we do NaT with VTS compensated noise parameters. It is evident that DNN is not able to learn everything that VTS does. When the GFL is compensated with only initialized additive noise mean, GFL-DNN does not perform well. However, GFL-DNN performs significantly better than NaT even VTS adapted NaT when VTS adapted noise parameters are used for compensation. Hence, it can be concluded that using an adaptable generative front-end layer provides better recognition accuracy than aware training where the estimated noise parameters are given directly to DNN, hoping that the DNN will learn the meaningful relationship. Besides, having a generative front-end layer gives more control and interpretation that leads to better adaptability of the entire system.

Table 5: *WER obtained with per-utterance Cepstral Mean Normalization (UttCMN) with MFCC-Δ-Δ feature and log-likelihood features obtained from the GFL for training DNN. When we adapt GFL, N2 noise estimator is used.*

Features	Adaptation	WER(%)				
		A	B	C	D	Avg.
MFCC-Δ-Δ+ UttCMN	–	2.9	7.1	7.5	17.1	11.1
GFL-Loglike+ UttCMN	NO	3.5	7.5	7.5	17.9	11.7
	YES	3.5	7.7	6.5	16.1	**10.9**

4.4. Effect of utterance mean normalization

In the previous experiments, we used per-speaker mean subtraction on all types of features before feeding to the DNN. Table 5 shows the results when per-utterance mean subtraction is applied on the features before feeding to the DNN. Configuration of GFL is same as in Table 3. For MFCC-Δ-Δ and un-adapted GFL log-likelihood features, the average performance improves compared to the results given in Table 3. However, average performance of adapted GFL log-likelihood features remains same. As VTS adaptation is already an utterance based adaptation, utterance based CMN could not improve further. Although the average performance of adapted GFL is similar to MFCC-Δ-Δ, improvement is significant in case of set C and D (Channel mismatch).

5. Conclusions

Having an adaptable generative front-end layer facilitates applying model based adaptation approaches such as VTS and thereby increase the system adaptability. In this paper, we proposed a noise adaptation strategy where a generative GMM-based system is used as a front-end layer to DNN. In Aurora-4 task, we obtained around 12.1% relative improvement compared to the baseline. Performance of the proposed system is also compared with noise aware training (NaT). We show that VTS estimated noise parameters yield better performance for noise aware training compared to simply relying on the average of silence frames for feature augmentation. We obtained around 5.7% relative improvement for VTS based NaT compared to the regular NaT. Our proposed system yields 6% and 11.4% relative improvements compared to the VTS-based NaT and regular NaT respectively. So far we have not managed to get similar performance of the unadapted GFL compared to the baseline. Having such a GFL will improve the performance when adaptation will be applied even in case of utterance based normalization. This will be a part of our future work.

6. References

[1] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach, "An overview of noise-robust automatic speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 22, no. 4, pp. 745–777, 2014.

[2] D. Macho, L. Mauuary, B. No, Y. M. Cheng, D. Ealey, D. Jouvet, H. Kelleher, D. Pearce, and F. Saadoun, "Evaluation of a noise-robust dsr front-end on aurora databases." in *Proc. Interspeech*, 2002.

[3] D. Yu, L. Deng, J. Droppo, J. Wu, Y. Gong, and A. Acero, "A minimum-mean-square-error noise reduction algorithm on mel-frequency cepstra for robust speech recognition," in *Proc. ICASSP*, 2008, pp. 4041–4044.

[4] C. J. Leggetter and P. 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.

[5] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains." *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.

[6] P. J. Moreno, B. Raj, and R. M. Stern, "A vector taylor series approach for environment-independent speech recognition," in *Proc. ICASSP*, 1996, pp. 733–736.

[7] D. Y. Kim, C. K. Un, and N. S. Kim, "Speech recognition in noisy environments using first-order vector taylor series." *Speech Communication*, vol. 24, no. 1, pp. 39–49, 1998.

[8] A. Acero, L. Deng, T. T. Kristjansson, and J. Zhang, "Hmm adaptation using vector taylor series for noisy speech recognition." in *Proc. Interspeech*, 2000, pp. 869–872.

[9] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "High-performance hmm adaptation with joint compensation of additive and convolutive distortions via vector taylor series." in *Proc. ASRU*, 2007, pp. 65–70.

[10] J. Du and Q. Huo, "An improved VTS feature compensation using mixture models of distortion and IVN training for noisy speech recognition," *IEEE/ACM Transactions on Audio, Speech & Language Processing*, vol. 22, no. 11, pp. 1601–1611, 2014.

[11] O. Kalinli, M. L. Seltzer, J. Droppo, and A. Acero, "Noise adaptive training for robust automatic speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 18, no. 8, pp. 1889–1901, 2010.

[12] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14 –22, 2012.

[13] T. Yoshioka and M. J. F. Gales, "Environmentally robust ASR front-end for deep neural network acoustic models," *Computer Speech & Language*, vol. 31, no. 1, pp. 65–86, 2015.

[14] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP*, 2013.

[15] N. Parihar, J. Picone, D. Pearce, and H. Hirsch, "Performance analysis of the aurora large vocabulary baseline system," in *European Signal Processing Conference*, 2004, pp. 553–556.

[16] B. Li and K. C. Sim, "Noise adaptive front-end normalization based on vector taylor series for deep neural networks in robust speech recognition," in *Proc. ICASSP*, 2013, pp. 7408–7412.

[17] S. J. Rennie, P. Fousek, and P. L. Dognin, "Factorial hidden restricted boltzmann machines for noise robust speech recognition." in *Proc. ICASSP*, 2012, pp. 4297–4300.

[18] B. Li and K. C. Sim, "An ideal hidden-activation mask for deep neural networks based noise-robust speech recognition," in *Proc. ICASSP*, 2014, pp. 200–204.

[19] S. Liu and K. C. Sim, "On combining DNN and GMM with unsupervised speaker adaptation for robust automatic speech recognition," in *Proc. ICASSP*, 2014, pp. 195–199.

[20] X. Lei, H. Lin, and G. Heigold, "Deep neural networks with auxiliary gaussian mixture models for real-time speech recognition," in *Proc. ICASSP*, 2013, pp. 7634–7638.

[21] N. A. Tomashenko and Y. Y. Khokhlov, "Speaker adaptation of context dependent deep neural networks based on map-adaptation and gmm-derived feature processing," in *Proc. Interspeech*, 2014, pp. 2997–3001.

[22] ——, "Gmm-derived features for effective unsupervised adaptation of deep neural network acoustic models," in *Proc. Interspeech*, 2015, pp. 2882–2886.

[23] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "A unified framework of HMM adaptation with joint compensation of additive and convolutive distortions," *Computer Speech & Language*, vol. 23, no. 3, pp. 389–405, 2009.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, Dec. 2011.

[25] D. Yu, A. Eversole, M. L. Seltzer, K. Yao, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, G. Chen, H. Wang, J. Droppo, A. Agarwal, C. Basoglu, M. Padmilac, A. Kamenev, V. Ivanov, S. Cypher, H. Parthasarathi, B. Mitra, Z. Huang, G. Zweig, C. Rossbach, J. Currey, J. Gao, A. May, B. Peng, A. Stolcke, M. Slaney, and X. Huang, "An introduction to computational networks and the computational network toolkit," Tech. Rep., August 2014.