BLSTM SUPPORTED GEV BEAMFORMER FRONT-END FOR THE 3RD CHIME CHALLENGE

Jahn Heymann, Lukas Drude*, Aleksej Chinaev, Reinhold Haeb-Umbach*

University of Paderborn
Department of Communications Engineering
Warburger Str. 100, Paderborn, Germany

ABSTRACT

We present a new beamformer front-end for Automatic Speech Recognition and apply it to the 3rd-CHiME Speech Separation and Recognition Challenge. Without any further modification of the back-end, we achieve a $53\,\%$ relative reduction of the word error rate over the best baseline enhancement system for the relevant test data set. Our approach leverages the power of a bi-directional Long Short-Term Memory network to robustly estimate soft masks for a subsequent beamforming step. The utilized Generalized Eigenvalue beamforming operation with an optional Blind Analytic Normalization does not rely on a Direction-of-Arrival estimate and can cope with multi-path sound propagation, while at the same time only introducing very limited speech distortions. Our quite simple setup exploits the possibilities provided by simulated training data while still being able to generalize well to the fairly different real data. Finally, combining our front-end with data augmentation and another language model nearly yields a $64\,\%$ reduction of the word error rate on the *real* data test set.

Index Terms— Robust Speech Recognition, Beamforming, Feature Enhancement, Neural Networks

1. INTRODUCTION

Automatic speech recognition has undergone major changes in recent years. With the rise of mobile devices with speech enabled personal assistants, application focus shifted from controlled environments to much noisier environments, which are prevalent if Automatic Speech Recognition (ASR) is used on mobile devices. This transition was supported and made possible by technical advances leading to better accuracy in those noisy environments. Besides sophisticated adaptation techniques, a large portion of the gain can be attributed to the replacement of Gaussian Mixture Model (GMM) acoustic models by Deep Neural Networks (DNNs) [1] [2]. While improving robustness a lot, DNNs are still sensitive to a mismatch between training and test condition [3]. This calls for either more relevant training data, or methods to leverage the power of DNNs even when trained on a relatively small data set, e.g., by employing additional front-end processing steps.

The 3rd CHiME Challenge [4] provides a common task for evaluating and comparing different approaches to noise-robust ASR. It is designed to be close to a real-word application: the task is to recognize speech recorded by six microphones placed around the frame of a tablet computer. The tablet is held by a speaker in a natural position in one of four noisy environments. The available amount of training data can be considered small as of nowadays standards.

In our contribution to the CHiME challenge we focus on front-end processing techniques. To be specific, we are going to take advantage of the multiple recording channels to obtain a beamformed signal of improved quality to be presented to the back-end ASR engine. While the back-end usually employs phase-insensitive features, an acoustic beamformer explicitly exploits the phase of the input signals to form a beam of increased sensitivity towards the location of the speaker. It has thus been shown to improve ASR performance even in the presence of a strong back-end [5]. Although early work exists where the beamforming operation is learnt by the network itself [6], this requires a huge amount of data and the results are still slightly worse.

While the baseline system provided by the CHiME organizers employed a Minimum Variance Distortionless Response (MVDR) beamformer, whose beamforming coefficients are computed from a Direction-of-Arrival (DoA) estimate obtained using SRP-PHAT for speaker tracking [4], our multi-channel processing is quite different. We propose to maximize the signal-to-noise ratio (SNR) of the beamformer output in each frequency bin separately, leading to the Generalized Eigenvalue (GEV) beamformer [7, 8]. This approach has the advantage, that no explicit estimation of the DoA is necessary. The beamformer coefficients are determined by solving a generalized eigenvalue problem instead, employing the Cross-Power Spectral Density (PSD) matrices of speech and noise. We show that these matrices can be obtained with the help of a neural network calculating a soft-mask for the speech and the noise. This allows us to approximate the PSD

*Both authors contributed equally.

matrices even in conditions of very low SNR like the ones encountered in the CHiME challenge.

Another advantage of the proposed beamformer is that the resulting beamformer coefficients are able to model an arbitrary acoustic transfer function, and are not restricted to modeling a pure delay, as is the case of the DoA-controlled MVDR beamformer, making our approach more suitable for reverberant environments.

A potential disadvantage is the speech distortion introduced by the GEV beamformer. These distortions, however, are curbed by a post-filter, which employs a normalizaion towards a distortionless response [7].

The paper is organized as follows. We begin with a short description of the CHiME database. Afterwards, the details of our system are described. We proceed by presenting the results and discussing them. Finally, we draw conclusion.

2. DATABASE

Here we only give a quick overview of the database, for details please refer to [4]. Most importantly, the database is split in two kinds of data, *simu* and *real*. As their names suggest, the *simu* data are simulated using speech signals recorded in a quiet environment and noise signals extracted from noisy signals recorded in four different environments of the *real* data. The *real* data on the other hand are recorded in four real-world noisy environments: in a bus (BUS), a cafe (CAF), on the street (STR) and in a pedestrian area (PED). The noise and the *real* data were recorded using a tablet computer equipped with six omnidirectional microphones. All provided recordings are sampled at $16\,\mathrm{kHz}$.

Both *simu* and *real* data are divided in three data sets: training, development and evaluation. The *simu* data of the training set contain in total 7138 utterances, approximately evenly distributed throughout the environments. The *real* data of the training set consist of 1600 utterances. The development and evaluation test data contain 1640 and 1320 utterances, respectively, for each kind of data.

All prompts are taken from the 5k WSJ0-Corpus [9].

3. SYSTEM OVERVIEW

Figure 1 gives an overview of our speech enhancement front-end. It consists of six bi-directional Long Short-Term Memory (BLSTM) supported neural networks with shared weights, one for each microphone channel. For each channel, the neural network estimates two masks: the first indicates which time-frequency-bins are presumably dominated by speech, while the second one indicates which are dominated by noise. The masks are then condensed to a single speech and a single noise mask using a median filter. They are used to estimate the PSD matrices of speech and noise, from which the beamformer coefficients are obtained by a generalized eigenvalue decomposition. After normalization, the beamforming operation is carried out on the multi-channel input to provide the single-channel enhanced speech signal to be forwarded to the ASR back-end.

Table 1: Network configuration for mask estimation

	Units	Type	Non-Linearity	p_{dropout}
L1	256	BLSTM	Tanh	0.5
L2	513	FF	ReLU	0.5
L3	513	FF	ReLU	0.5
L4	1026	FF	Sigmoid	0.0

3.1. Neural mask estimator

The neural network estimating the masks for the GEV beamformer consists of four layers. Table 1 gives an overview of their configuration.

3.1.1. Network configuration

For the first layer, 256 BLSTM [10, 11] units are used for each direction over the signal. They enable the network to capture necessary context information. In our implementation we omit the peepholes and use an additional weight matrix to join the forward ($\overrightarrow{\mathbf{h}}_t$) and the backward ($\overleftarrow{\mathbf{h}}_t$) direction to obtain the final layer output l_t:

$$\mathbf{i}(t) = \sigma\left(\mathbf{W_{iy}}\mathbf{y}(t) + \mathbf{W_{ih}}\mathbf{h}(t-1) + \mathbf{b_i}\right), \quad (1)$$

$$\mathbf{f}(t) = \sigma\left(\mathbf{W_{fy}}\mathbf{y}(t) + \mathbf{W_{fh}}\mathbf{h}(t-1) + \mathbf{b_f}\right), \quad (2)$$

$$\mathbf{o}(t) = \sigma\left(\mathbf{W_{oy}}\mathbf{y}(t) + \mathbf{W_{oh}}\mathbf{h}(t-1) + \mathbf{b_o}\right), \quad (3)$$

$$\mathbf{c}(t) = \mathbf{f}(t) \odot \mathbf{c}(t-1) + \mathbf{i}(t) \odot \tanh\left(\mathbf{W_{cy}}\mathbf{y}_t + \mathbf{W_{ch}}\mathbf{h}(t-1) + \mathbf{b_c}\right), \quad (4)$$

$$\mathbf{h}(t) = \mathbf{o}(t) \odot \tanh\left(\mathbf{c}(t)\right), \quad (5)$$

$$\mathbf{l}(t) = \mathbf{W}_{\overrightarrow{\mathbf{h}}}\overrightarrow{\mathbf{h}}(t) + \mathbf{W}_{\overleftarrow{\mathbf{h}}}\overleftarrow{\mathbf{h}}(t). \quad (6)$$

Here, $\odot$ denotes element-wise multiplication of vectors.

The input ($\mathbf{y}_t$) for the layer is a single frame of the magnitude spectra of one channel. For the STFT we use a frame size of 1024, a frame shift of 256 and a FFT size of 1024.

The next two layers are feed-forward (FF) layers with a Rectified Linear Unit (ReLU) activation function. We clip the activation at a value of 20. Both layers have 513 units.

The last layer has 1026 units and is split in two parts: The first 513 units estimate the speech mask $\mathrm{IBM}_{\mathbf{X}}$, while the last 513 units estimate the noise mask $\mathrm{IBM}_{\mathbf{N}}$. We do not force the values of the estimated masks to be one or zero. Rather, we restrict them to be in the range between one and zero using a Sigmoid non-linearity for the activation function of both estimates. We also do not enforce non-overlapping masks or masks which sum to one for each time-frequency-bin.

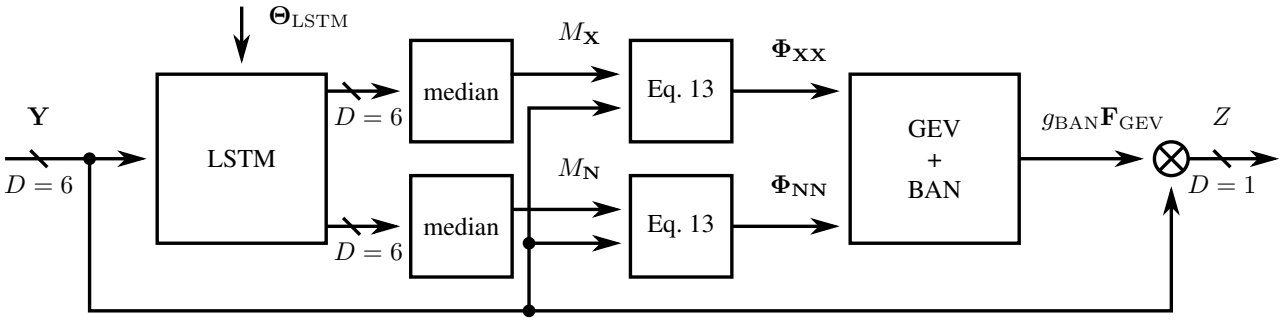

Fig. 1: Enhancement system based on an LSTM net for mask estimation and GEV beamformer for target signal estimation.

3.1.2. Weight initialization & optimization

For the BLSTM layer, they are drawn from a uniform distribution ranging from -0.02 to 0.02, while the ReLU layers are initialized using a normal distribution with $\mu = 0$ and $\sqrt{\sigma} = 0.01$. For the last layer we use the initialization as proposed in [12]. The biases are all intialized with zeros.

We employ RMSProp [13] for training. A fixed learning-rate of 0.1, a momentum of 0.9 and full backpropagation through time [14] is used. If the norm of a gradient is greater than one, we divide the gradient by the norm to scale it [15].

To achieve a better generalization, we use dropout for the input-hidden connection of the BLSTM units [16] and for the input of the two ReLU FF layers [17]. The dropout rate is fixed at $p_{\text{dropout}} = 0.5$ for every layer during the whole training. Additionally we use the development data for cross-validation, stopping the training when the loss does not decrease anymore after 5 epochs of patience.

3.1.3. Normalization

Instead of normalizing the input, we apply the recently proposed batch normalization [18] for each layer. Here, the activation $\mathbf{a} = \mathbf{W}\mathbf{u} + \mathbf{b}$ is replaced with a normalized activation $\text{BN}(\mathbf{a})$, where

$$\text{BN}(\mathbf{a}) = \gamma\hat{\mathbf{u}} + \beta \tag{7}$$

and

$$\hat{\mathbf{u}}(k) = \frac{\mathbf{W}_k\mathbf{u}(k) - \text{E}[\mathbf{W}_k\mathbf{u}(k)]}{\sqrt{\text{Var}[\mathbf{W}_k\mathbf{u}(k)]}}. \tag{8}$$

The normalization is carried out separately for each dimension k.

While the method was originally proposed for feed-forward networks, we also apply it to the BLSTM layer. Here, we normalize the input activation ($\mathbf{W}_{\cdot y}\mathbf{y}_t$) but not the recurrent activation ($\mathbf{W}_{\cdot h}\mathbf{h}_{t-1}$). With the network structure described as above, this is almost equal to normalizing the input on a per-utterance basis for each dimension. The difference is, that the input can still be scaled with the trainable parameters γ and β. Preliminary testing showed faster convergence in some cases.
Also, in contrast to the proposed method [18], we do not use the population estimates for the mean and variance at decoding time. This is done in the original paper to ensure that the output depends on the input in a deterministic way. Changing the mini-batch size or using differently composed mini-batches would lead to different outputs. In our case however, one mini-batch (as seen by the layer inputs) is equal to the features of one utterance. Hence, we can use the same transformation that we used during training and still get a deterministic output.

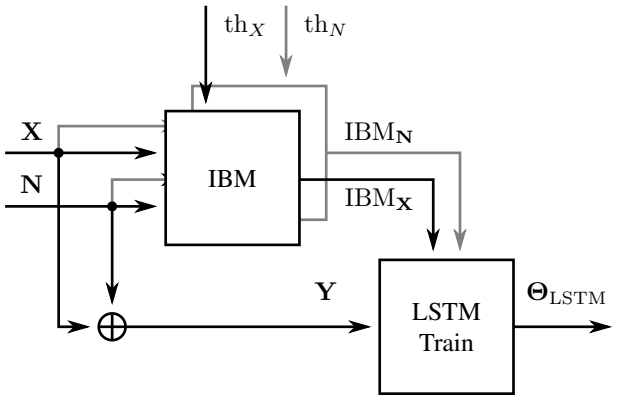

Fig. 2: Training input of the BLSTM mask estimator.

3.1.4. Ideal binary masks as targets

The training setup of the neural network for mask estimation is displayed in Figure 2.

The ideal binary mask for noise ($\mathrm{IBM}_{\mathbf{N}}$) is defined by

$$\mathrm{IBM}_{\mathbf{N}}(t,f) = \begin{cases} 1, & \frac{||\mathbf{X}||}{||\mathbf{N}||} < 10^{\mathrm{th}_{\mathbf{N}}(f)}, \\ 0, & \text{else}, \end{cases} \tag{9}$$

where $||\cdot||$ is the Euclidean norm.

Correspondingly, the ideal binary mask for the target signal ($\mathrm{IBM}_{\mathbf{X}}$) is defined by

$$\mathrm{IBM}_{\mathbf{X}}(t,f) = \begin{cases} 1, & \frac{||\mathbf{X}||}{||\mathbf{N}||} > 10^{\mathrm{th}_{\mathbf{X}}(f)}, \\ 0, & \text{else}. \end{cases} \tag{10}$$

The two thresholds $\mathrm{th}_{\mathbf{X}}$ and $\mathrm{th}_{\mathbf{N}}$ are not identical. They are chosen such that a decision in favor of speech or noise is only taken if the instantaneous SNR is high or low enough, respectively, to ensure a low false acceptance rate. This will result in more reliable cross-power spectral density matrix estimates at the expense of discarding some time-frequency-bins which are categorize to be neither speech nor noise.

Note that the ideal binary masks can only be calculated for the simulated training and development data.

3.1.5. Loss function

The network is trained on all utterances and all channels using the binary cross-entropy cost. With $\mathrm{IBM}_{\nu}(t,f), \nu \in \{\mathbf{X}, \mathbf{N}\}$, being the target mask value for the f-th frequency bin at the t-th frame, $M_{\nu}(t,f)$ the networks estimation for that value, F the total number of frequency bins and T the total number of frames, the loss is given by

$$L = -\frac{1}{F}\frac{1}{2T} \sum_{\nu \in (\mathbf{N},\mathbf{X})} \sum_{t=1}^{T} \sum_{f=1}^{F} \mathrm{IBM}_{\nu}(t,f) \log_2 M_{\nu}(t,f) \tag{11}$$

$$+ (1 - \mathrm{IBM}_{\nu}(t,f)) \log_2 (1 - M_{\nu}(t,f))$$

To accelerate the training procedure, we process each channel in parallel, exploiting the fact that they have the same amount of time steps.

3.1.6. Decoding at test time

At test time, we estimate a masks for noise and speech for each channel. Afterwards, we take the element-wise median of the channels. This increases the robustness of the method in case of a channel failure. If not more than two channels are corrupted, the estimate for the masks will not be affected.

3.2. GEV beamformer with BAN

As stated in the introduction, we propose to maximize the SNR of the beamformer output in each frequency bin separately leading to the GEV beamformer with coefficients [7]:

$$\mathbf{F}_{\mathrm{GEV}}(f) = \arg\max_{\mathbf{F}} \frac{\mathbf{F}^{\mathrm{H}} \mathbf{\Phi}_{\mathbf{XX}}(f) \mathbf{F}}{\mathbf{F}^{\mathrm{H}} \mathbf{\Phi}_{\mathbf{NN}}(f) \mathbf{F}}. \tag{12}$$

Please note that this does not require any assumptions regarding the nature of the acoustic transfer function from the speech source to the sensors or regarding the spatial correlation of the noise [7]. The only assumption which needs to be made is that the target signal is prevalent in the target PSD matrix $\mathbf{\Phi}_{\mathbf{XX}}$ whereas noise is prevalent in the noise PSD matrix $\mathbf{\Phi}_{\mathbf{NN}}$.

Therefore, the non-overlapping masks $M_{\mathbf{X}}$ and $M_{\mathbf{N}}$ are used to calculate the weighted sum of products matrices:

$$\mathbf{\Phi}_{\nu\nu}(f) = \sum_{t=1}^{T} M_{\nu}(t,f) \mathbf{Y}(t,f) \mathbf{Y}(t,f)^{\mathrm{H}}. \tag{13}$$

We decided to estimate a time-invariant beamformer, i.e., the beamforming coefficients do not change in the course of an utterance. This is in contrast to the beamformer of the baseline system, which employs time-variant coefficients in order to track a moving speaker.

The cost function in Eq. (12) is known as the Rayleigh coefficient. The optimization problem leads to the well known solution

$$\mathbf{F}_{\mathrm{GEV}}(f) = \mathcal{P}\left\{\mathbf{\Phi}_{\mathbf{NN}}^{-1} \mathbf{\Phi}_{\mathbf{XX}}\right\} \tag{14}$$

(if the inverse of the noise PSD matrix exists), where $\mathcal{P}\{\cdot\}$ yields the principal component.

Unlike the MVDR beamformer, the GEV beamformer can introduce arbitrary speech distortions. These, however, can be reduced using a single channel post-filter [7]:

$$g_{\mathrm{BAN}}(f) = \frac{\sqrt{\mathbf{F}_{\mathrm{GEV}}^{\mathrm{H}} \mathbf{\Phi}_{\mathbf{NN}}(f) \mathbf{\Phi}_{\mathbf{NN}}(f) \mathbf{F}_{\mathrm{GEV}} / D}}{\mathbf{F}_{\mathrm{GEV}}^{\mathrm{H}} \mathbf{\Phi}_{\mathbf{NN}}(f) \mathbf{F}_{\mathrm{GEV}}}. \tag{15}$$

The filter performs a Blind Analytic Normalization (BAN) to obtain a distortionless response in the direction of the speaker. If speech distortions were removed perfectly, one would eventually arrive at the MVDR beamformer [19, 20].

We now obtain an estimate for the source signal as:

$$Z(t,f) = g_{\mathrm{BAN}}(f) \mathbf{F}_{\mathrm{GEV}}^{\mathrm{H}}(f) \mathbf{Y}(t,f). \tag{16}$$

The estimate $Z(t,f)$ is then provided to the baseline backend for training and recognition.

3.3. Exploiting context

The CHiME Challenge allows to use up to 5 s of preceding context [4], although no guarantee is given that there is only background noise within these 5 s. There might as well be an overlap with another utterance or with the same utterance where the speaker made a mistake. Our setup, however, makes it easy to exploit the context for the case of *real* data. We just prepend the context and use the masking in the same way as we would without context. The additional frames help estimating the PSD matrices of the speech and noise especially if the original utterance is short. Note that this only

works for the *real* data case. For the simulated data, possible speech artifacts within the context are from a different speaker due to the way the data are generated.

We do not explicitly report results without using the context, but preliminary experiment showed that it improves the word error rate (WER) by 1-3 percent points.

3.4. Data augmentation

Neural networks are known to perform better the more data is used for training (e.g. [21]). We try to exploit this fact by augmenting the available simulated data. Namely, we change the SNR by multiplying the noise images with $10^{G/10}$, where G is uniformly sampled in $[-8\,\mathrm{dB}, -1\,\mathrm{dB}]$.

The augmented data are used to train the back-end acoustic model, as well as the neural network estimating the masks. Overall, we generate two new utterances for every utterance of the simulated training data. We also tried to increase this number to ten but results got worse, presumingly due to overfitting to the specific utterances.

3.5. Back-end

For all experiments, the original baseline back-end is used. It features two different acoustic models. One is based on a GMM and the other on a DNN. For GMM training, several techniques are used to improve robustness and recognition accuracy. The final model uses 40 dimensional LDA compressed feature vectors, and is speaker adaptively trained using feature space maximum likelihood linear regression (fMLLR). The DNN has 7 layers with 2048 Sigmoid units, is pre-trained using a restricted Boltzman machine and fine-tuned with cross-entropy cost. Further improvements are achieved by a sequence discriminative training using the state-level Minimum Bayes Risk (sMBR) criterion. For a detailed description refer to [4].

Note that we always perform a matched training for each variation, i.e. the models are trained from scratch using the training data decoded with the front-end variation in question.

3.6. Language model

The CHiME Challenge includes a pre-trained tri-gram language model (LM) which is used in the baseline system. In addition to this, there is also training data available to train a custom language model[1]. We use this training data and the KenLM [22] tool to train a modified Kneser-Ney [23] tri-gram language model. We apply pruning to remove singletons of order three and use the default settings otherwise.

Please note that we use the trained language model only during decoding and not for the sequence training.

[1]Note that we only use the official data from the directory CHiME3/data/WSJ0/wsj0/doc/lng_modl/lm_train/np_data

Table 3: Detailed WERs for the best system

	Development		Evaluation	
	simu	*real*	*simu*	*real*
BUS	5.74	9.44	7.55	17.54
CAF	7.63	6.67	9.58	10.48
PED	5.66	5.78	9.38	11.04
STR	6.45	7.74	9.47	9.99

4. RESULTS

Table 2 shows the results for our proposed system for different acoustic models and with different front-end configurations. We also list the results of the baseline enhancement system for an easier comparison[2]. The results reveal that our proposed system outperforms the baseline by a large margin for the *real* data, despite sharing the same back-end. For the *simu* data we achieve comparable or slightly better results, depending on the configuration. This indicates that our system is able to generalizes better across the different types of data and can leverage the simulated data to learn about the *real* data. Indeed we also trained a (GMM) model[3] without using any *real* data. The influence on the WER was rather small. It only showed a relative drop of $3\,\%$ for the *real* development data and of $4\,\%$ for the *real* evaluation data.

Data augmentation helps to improve the performance. This is especially true for a DNN acoustic model. Adding the augmented training data results in a $6\,\%$ ($11\,\%$) relative improvement for the *real* development (evaluation) data and a $16\,\%$ ($10\,\%$) relative improvement with sequence training. For the GMM on the other hand, improvements are negligible with $0.2\,\%$ ($3\,\%$). This is within our expectation that a DNN can make better use of additional and more diverse data.

The post-filter on the other hand improves results primarily for the GMM acoustic model, leading to a WER almost similar to the the ones from a DNN. For the latter one, the post-filter has a mixed impact. While it is able to improve the results for the development data, the performance for the evaluation data becomes slightly deteriorated. Especially for the simulated we see an increase of the WER by at least $10\,\%$. However, for the more important *real* data, the impact is not so significant. For a sequentially trained model, the WER increases by $3\,\%$, while it decreases by $3\,\%$ if no sequence training is applied. We suspect that the reason for this result is that the post-filter removes some variability from the data which the DNN could exploit for better classification.

Finally, using the trained language model brings another

[2]When available, we use the results reported in [4]

[3]The model corresponds to the third GMM model in Table 2 but is not listed there in favour of a cleaner table

Table 2: Overview of the WERs for different system combination

Backend	Augmentation	BAN	sMBR	KenLM	Development		Evaluation	
					simu	*real*	*simu*	*real*
GMM	*Baseline*		✗	✗	9.79	20.55	10.59	37.36
DNN	*Baseline*		✗	✗	9.27	20.14	12.75	40.17
DNN	*Baseline*		✓	✗	8.17	17.72	11.19	33.76
GMM	✗	✗	✗	✗	10.20	10.42	11.16	16.47
GMM	✓	✗	✗	✗	9.61	10.40	10.53	15.92
GMM	✓	✓	✗	✗	8.88	9.92	9.75	14.65
GMM	✓	✓	✗	✓	8.51	9.56	9.25	13.77
DNN	✗	✗	✗	✗	8.89	9.91	11.26	16.73
DNN	✓	✗	✗	✗	7.82	9.30	10.38	14.88
DNN	✓	✓	✗	✗	7.77	8.84	11.43	14.34
DNN	✓	✓	✗	✓	7.44	8.49	10.59	13.27
DNN	✗	✗	✓	✗	8.06	10.42	10.05	15.75
DNN	✓	✗	✓	✗	7.11	8.70	9.56	14.15
DNN	✓	✓	✓	✗	6.98	8.11	10.72	14.55
DNN	✓	✓	✓	✓	**6.37**	**7.41**	**8.99**	**12.26**

significant gain. We see the biggest impact on the sequentially trained model with a relative improvement of over $8\,\%$ for the development data and over $15\,\%$ for the evaluation data, leading to a final WER of $12.26\,\%$ for the *real* data. Compared with the best baseline enhancement result as reported in [4], we achieve a relative improvement of nearly $64\,\%$.

Table 3 gives a more detailed view on how the WER for the best model is composed. One can immediately see that the big difference between the WER for the *real* evaluation scenario and the other scenarios is caused by a big difference for the bus environment. Listening to some of those utterances, the reason for this is probably a very low SNR combined with some channel failures. Since the bus engine – which is mostly the prevalent noise source – is a rather diffuse noise source, the beamformer is unable to suppress it properly, leaving the back-end to deal with a very noisy utterance. Otherwise, the gap between *real* and *simu* data is present but relatively small.

5. CONCLUSION

In this work we present a new approach to beamforming for ASR. Instead of using the widely employed MVDR beamformer, we opt for a variant which does not rely on a Direction-of-Arrival estimate. The beamformer coefficients are derived from an eigenvalue decomposition of the speech and noise PSD matrices instead. These matrices can be estimated leveraging the power of a discriminatively trained recurrent neural network to obtain the time-frequency-bins where the target source is dominant. This also allows us to exploit context in a natural way. Our system works on a per-utterance basis, requires only one decoding pass and does not use costly ensembles. Nevertheless, we are able to achieve a WER reduction of $53\,\%$ compared to the baseline enhancement system. Further improvements, namely using data augmentation, a post-filter and another language model trained on official data, yield a reduction of nearly $64\,\%$.

6. ACKNOWLEDGEMENTS

This work was in part supported by the Deutsche Forschungsgemeinschaft (DFG) under contract no. Ha3455/11-1. We would like to thank the developers of Theano [24] [25] (used for our front-end system) and Kaldi [26] (back-end) for the tools making this work possible.

7. REFERENCES

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

[2] Dong Yu, Frank Seide, and Gang Li, "Conversational speech transcription using context-dependent deep neural networks.," in *ICML*, 2012.

[3] Jahn Heymann, Reinhold Haeb-Umbach, Pavel Golik, and Ralf Schlüter, "Unsupervised adaptation of a denoising autoencoder by bayesian feature enhancement for reverberant asr under mismatch conditions," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Brisbane, Australia, Apr. 2015, pp. 5053–5057.

[4] Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe, "The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines," in *IEEE 2015 Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2015.

[5] Marc Delcroix, Takuya Yoshioka, Atsunori Ogawa, Yotaro Kubo, Masakiyo Fujimoto, and Nobutaka Ito, "Linear prediction-based dereverberation with advanced speech enhancement and recognition technologies for the reverb challenge," in *REVERB Challenge Workshop*, Florence, Italy, May 2014.

[6] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson, "Speech acoustic modeling from raw multichannel waveforms," .

[7] Ernst Warsitz and Reinhold Haeb-Umbach, "Blind acoustic beamforming based on generalized eigenvalue decomposition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 5, pp. 1529–1539, 2007.

[8] Alexander Krueger, Ernst Warsitz, and Reinhold Haeb-Umbach, "Speech enhancement with a gsc-like structure employing eigenvector-based transfer function ratios estimation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 206–219, jan. 2011.

[9] John Garofalo et al., "CSR-I (WSJ0) complete," *Linguistic Data Consortium, Philadelphia*, 2007.

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

[11] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[12] 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.

[13] Tijmen Tielman and Geoffrey Hinton, "Lecture 6.5 - RMSProp," *COURSERA: Neural Networks for Machine Learning*, 2012.

[14] Paul J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[15] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "Understanding the exploding gradient problem," *CoRR*, vol. abs/1211.5063, 2012.

[16] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[18] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[19] Barry D. Van Veen and Kevin M. Buckley, "Beamforming techniques for spatial filtering," *Digital Signal Processing Handbook*, pp. 61–1, 1997.

[20] Uwe K. Simmer, Joerg Bitzer, and Claude Marro, "Post-filtering techniques," in *Microphone Arrays*, pp. 39–60. Springer, 2001.

[21] Navdeep Jaitly and Geoffrey Hinton, "Vocal tract length pertrubation (VTLP) improves speech recognition," in *International Conference on Machine Learning (ICML) Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[22] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn, "Scalable modified Kneser-Ney language model estimation," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August 2013, pp. 690–696.

[23] Reinhard Kneser and Herman Ney, "Improved backing-off for m-gram language modeling," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, May 1995, vol. 1, pp. 181–184 vol.1.

[24] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[25] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010, Oral Presentation.

[26] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlıcek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," .