The RWTH/UPB/FORTH System Combination for the 4th CHiME Challenge Evaluation

Tobias Menne[1], *Jahn Heymann*[2], *Anastasios Alexandridis*[3,4], *Kazuki Irie*[1], *Albert Zeyer*[1], *Markus Kitza*[1], *Pavel Golik*[1], *Ilia Kulikov*[1], *Lukas Drude*[2], *Ralf Schlüter*[1], *Hermann Ney*[1], *Reinhold Haeb-Umbach*[2], *Athanasios Mouchtaris*[3,4]

[1]Human Language Technology and Pattern Recognition,
Computer Science Department, RWTH Aachen University, Aachen, Germany
[2]Paderborn University, Department of Communications Engineering, Paderborn, Germany
[3]FORTH-ICS, Signal Processing Laboratory, Heraklion, Crete, Greece, GR-70013
[4]University of Crete, Department of Computer Science, Heraklion, Crete, Greece, GR-70013

[1]<surname>@cs.rwth-aachen.de, [2]<surname>@nt.uni-paderborn.de,
[3],[4]{analexan,mouchtar}@ics.forth.gr

Abstract

This paper describes automatic speech recognition (ASR) systems developed jointly by RWTH, UPB and FORTH for the 1ch, 2ch and 6ch track of the 4th CHiME Challenge. In the 2ch and 6ch tracks the final system output is obtained by a Confusion Network Combination (CNC) of multiple systems. The Acoustic Model (AM) is a deep neural network based on Bidirectional Long Short-Term Memory (BLSTM) units. The systems differ by front ends and training sets used for the acoustic training. The model for the 1ch track is trained without any preprocessing. For each front end we trained and evaluated individual acoustic models. We compare the ASR performance of different beamforming approaches: a conventional superdirective beamformer [1] and an MVDR beamformer as in [2], where the steering vector is estimated based on [3]. Furthermore we evaluated a BLSTM supported Generalized Eigenvalue beamformer using NN-GEV [4]. The back end is implemented using RWTH's open-source toolkits RASR [5], RETURNN [6] and rwthlm [7]. We rescore lattices with a Long Short-Term Memory (LSTM) based language model. The overall best results are obtained by a system combination that includes the lattices from the system of UPB's submission [8]. Our final submission scored second in each of the three tracks of the 4th CHiME Challenge.

1. Background

This paper describes ASR systems for the 1ch, 2ch and 6ch tracks of the 4th CHiME Challenge. In contrast to the provided baseline system [9] the back end has been replaced completely and is described in Section 2.2. Furthermore we developed additional systems using different front ends. The front ends are described in Section 2.1. All experimental results presented in this work (Sections 3 and 4) are obtained with the official training set following the rules of the CHiME challenge.

2. Contributions

2.1. Front ends

In addition to the baseline (BL) front end we developed three other front ends that utilize different beamformers. The final enhanced signal at the output of each beamformer is given by:

$$\boldsymbol{Z}(k,l) = \boldsymbol{w}(k,l)^H \boldsymbol{X}(k,l) \tag{1}$$

where k, l denote the frequency index and time-frame, respectively, $\boldsymbol{w}(k,l)$ is the $M \times 1$ vector of beamformer filter coefficients for a given front end, $\boldsymbol{X}(k,l)$ is the $M \times 1$ vector of microphone array signals in the Short-time Fourier Transform (STFT) domain, and M denotes the number of microphones.

We also improved the microphone failure detection mechanism of [2], so as to better identify corrupted microphones. The enhanced microphone failure detection was used in the front ends described in Sections 2.1.2 & 2.1.3.

2.1.1. Microphone failure detection

Our microphone failure detection mechanism is based on measuring the consistency of the energies (calculated in each time frame) between the microphone signals. To do that, we construct M time-series $e_m(l), m = 1, \dots, M$, each one containing the energy of the signal for $l = 1, \dots, L$ frames, where L denotes the total number of frames in the utterance. Then, for each microphone m, the average correlation coefficient r_m^{AV} between $e_m(l)$ and $e_n(l)$ for $n \neq m$ is calculated. A microphone is considered to have failed if r_m^{AV} is less than a threshold δ, which was set empirically to 0.8. These microphones, in addition to the microphones which are considered to have failed by the system of [2], are excluded from further processing.

2.1.2. MVDR beamformer with steering vector estimation

This front end (MV) utilizes a minimum variance distortionless response (MVDR) beamformer with diagonal loading, similar to the one in [2]. The filter coefficients are calculated as:

$$\boldsymbol{w}_{\text{MVDR}}(k,l) = \frac{\left[\boldsymbol{R}_{\text{n}}(k) + \epsilon \operatorname{diag}(|\boldsymbol{X}(k,l)|^2)\right]^{-1} \boldsymbol{d}(k)}{\boldsymbol{d}(k)^H \left[\boldsymbol{R}_{\text{n}}(k) + \epsilon \operatorname{diag}(|\boldsymbol{X}(k,l)|^2)\right]^{-1} \boldsymbol{d}(k)} \tag{2}$$

where $\epsilon = 10^{-3}$ is the diagonal loading term, $\boldsymbol{d}(k)$ is the steering vector, $\boldsymbol{R}_{\text{n}}(k)$ is the spatial correlation matrix of noise, and diag($\mathbf{x}$) denotes the conversion of vector $\boldsymbol{x}$ to a diagonal matrix.

For the estimation of the unknown quantities $\boldsymbol{d}(k)$ and $\boldsymbol{R}_{\text{n}}(k)$ we use the method of [3], which does not require knowledge of the array geometry or the speaker location. We assume that each frequency bin contains either speech and noise or is dominated only by noise. This assumptions allows the clustering of the STFT coefficients into two classes: the noisy (i.e., speech + noise) and the noise-only class. The clustering is performed by modeling the STFT coefficients at each frequency with a 2-component complex Gaussian mixture model. To associate each Gaussian component to its correct class, we measure the ratio of the first to second largest eigenvalues of the estimated covariance matrices. The component for which this ratio is the largest is assigned to the noisy class.

The spatial correlation matrices of speech, $\boldsymbol{R}_{\mathrm{s}}(k)$, noise, $\boldsymbol{R}_{\mathrm{n}}(k)$, and noisy signals, $\boldsymbol{R}_{\mathrm{sn}}(k)$, are then estimated based on the posterior probabilities of each bin to belong to the noisy or noise-only class as:

$$\begin{aligned}
\boldsymbol{R}_{\mathrm{sn}}(k) &= \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{X}(k,l)\boldsymbol{X}(k,l)^H \quad (3)\\
\boldsymbol{R}_{\mathrm{n}}(k) &= \frac{1}{\sum_{l=1}^{L}\lambda_n(k,l)}\sum_{l=1}^{L}\lambda_n(k,l)\boldsymbol{X}(k,l)\boldsymbol{X}(k,l)^H \quad (4)\\
\boldsymbol{R}_{\mathrm{s}}(k) &= \boldsymbol{R}_{\mathrm{sn}}(k) - \boldsymbol{R}_{\mathrm{n}}(k) \quad (5)
\end{aligned}$$

where $\lambda_n(k,l)$ denotes the posterior probability that the time-frequency bin (k,l) is dominated by noise.

Finally, the steering vector for each frequency bin k is estimated as the principal component of $\boldsymbol{R}_{\mathrm{s}}(k)$. For the 6ch track the spatial correlation matrix of noise which is used in Eq. (2) is estimated from Eq. (4), while for the 2ch track it is estimated from 400 ms to 800 ms of context immediately before the utterance, as it was shown to produce better recognition performance in the 2ch case. Each utterance was processed using frames of 512 samples with 50% overlap, windowed with sine windows and an FFT size of 512 samples, while channel 2 was excluded from processing.

2.1.3. Superdirective beamformer using time-delays

The superdirective beamformer maximizes the array gain, while maintaining a minimum constraint on the white noise gain [1]. The beamformer filter coefficients are computed as:

$$\boldsymbol{w}_{\mathrm{SD}}(k,l) = \frac{\left[\boldsymbol{\Gamma}(k) + \epsilon\boldsymbol{I}\right]^{-1}\boldsymbol{d}(k,l)}{\boldsymbol{d}(k,l)^H\left[\boldsymbol{\Gamma}(k) + \epsilon\boldsymbol{I}\right]^{-1}\boldsymbol{d}(k,l)} \quad (6)$$

where $\boldsymbol{I}$ is the identity matrix and ϵ is the diagonal loading term which is used to control the white noise gain (WNG) constraint. $\boldsymbol{\Gamma}(k)$ is the noise coherence matrix for frequency bin k (assumed to be spherically isotropic diffuse [10]) whose elements are given by:

$$\Gamma_{ij}(k) = \mathrm{sinc}\left(\frac{2\pi f d_{ij}}{c}\right) \quad (7)$$

where f is the frequency in Hz, $c = 343$ m/s is the speed of sound and d_{ij} denotes the distance between the ith and jth microphone. Finally, the steering vector is represented by:

$$\boldsymbol{d}(k,l) = \left[e^{-j2\pi f\tau_1(l)} \quad \cdots \quad e^{-j2\pi f\tau_M(l)}\right] \quad (8)$$

where $\tau_i(l)$ denotes the time delay to the ith microphone for time-frame l, which was estimated using the nonlinear SRP-PHAT pseudo-spectrum [2]. To determine ϵ, we start from $\epsilon = 0$ and iteratively increase it by 0.05 until the WNG becomes equal or greater than -10 dB.

This front end (SD) is used in the 6ch track, as well as in the 2ch track. For both tracks, we used frames of 1024 samples with 50% overlap, windowed with sine windows and an FFT size of 1024, while channel 2 was excluded from processing.

2.1.4. BLSTM supported GEV

The Generalized Eigenvalue (GEV) front end (GE) maximizes the signal-to-noise ratio after the beamforming operation:

$$\boldsymbol{w}_{\mathrm{GEV}}(k) = \underset{\boldsymbol{d}}{\mathrm{argmax}}\,\frac{\boldsymbol{d}(k)^{\mathrm{H}}\boldsymbol{R}_{\mathrm{s}}(k)\boldsymbol{d}(k)}{\boldsymbol{d}(k)^{\mathrm{H}}\boldsymbol{R}_{\mathrm{n}}(k)\boldsymbol{d}(k)}. \quad (9)$$

Maximizing this equation leads to the generalized eigenvalue problem and its solution to the beamforming vector $\boldsymbol{w}_{\mathrm{GEV}}(k)$ for each frequency. Similar to the MVDR beamformer described above, this beamformer only relies on the signal statistics, i.e. no assumptions on the microphone array configurations are made. In contrast to the MVDR however, the GEV can introduce arbitrary distortions because the magnitude of each beamforming vector can be chosen arbitrarily. We therefore normalize the steering vectors using Blind Analytic Normalization (BAN) [11]. This postfilter normalizes the Acoustic Transfer Function (ATF) from the target source to unit gain for each frequency.

The spatial correlation matrices needed for the beamforming operation are estimated using time-frequency masks from a neural network [12][4]. Here, we calculate two masks, one for the target and one for the distortion. These masks do not necessarily sum to one. We only want to take those time-frequency bins into consideration where the respective source is surely predominant. To calculate the masks we treat each microphone separately and then use median pooling to condense the masks into one for each source. This strategy makes the mask estimation immune to corrupted channels. It also allows us to use the same front end for the 6 channel, as well as for the 2 channel track without making any changes to the network. The network is the same as described in [12] and is trained using binary masks as targets.

2.2. Back end

2.2.1. Data sets

The participants of the CHiME 4 Challenge were given a training corpus that was derived from the WSJ0 SI-84 data set (approx. 18 hours) recorded with a close talk microphone (channel 0) and 6 distant microphones (channels 1-6). First off we trained a fairly standard GMM/HMM acoustic model on the quasi-clean data (channel 0 of the real training data as well as the booth training data and the original WSJ corpus) in order to use its alignments on all other channels without having to re-align the data for every subsequent experiment. We further created a flattened training set (referred to as a set of front facing microphones FC) by simply concatenating the channels $\{1,3,4,5,6\}$ of both real and simulated data into a 90 hours corpus. We mostly discard the second channel since the corresponding microphone points away from the speaker, resulting in a slightly worse quality. In order to investigate the effect of beamforming on the overall ASR performance, we further define an extension of the flattened set FC by adding the beamformed signal to the concatenation. The resulting 108 hours corpus is referred to as set FC+B.

For the processing of test data we followed the rules of the challenge. In the 1ch track, no beamforming is required. In the 2ch and 6ch tracks, we first beamform all available channels into a single signal before decoding. The recognition was done using the standard 5k lexicon and baseline 5-gram count LM, followed by lattice rescoring with a neural network language model.

2.2.2. *RWTH's BLSTM acoustic model*

The first back end was implemented using RWTH's open-source toolkits RASR [5] and RETURNN [6]. We will refer to this back end as *"R"* and the Kaldi baseline back end as *"K"*. The architecture and training algorithms for the speaker independent and speaker adapted AM are identical. The AM is a Deep Neural Network (DNN) with five BLSTM layers of size 600. The mini-batch training is carried out using stochastic gradient descent with Nadam [13] and the learning rate reduction is controlled by Newbob [14]. The initial learning rate is set to 10^{-3} and the gradient is distorted by Gaussian noise [15] with an initial variance of 0.3. The cross-entropy training is regularized by a dropout rate of 10% and L_2 norm of the weights with a factor of 0.01. The decoding pipeline is shown in Figure 1. It differs from a standard two-pass decoding strategy by an additional LM rescoring with a neural network LM after both passes.

2.2.3. *UPB's wide residual BLSTM acoustic model*

The lower part of the second back end follows a slightly modified design of a Wide Residual Network (WRN) [16] with $d = 22$ and $k = 5$, where d describes the depth of the network (i.e. number of layers) while k is a multiplicative factor for the number of channels (i.e. the width of the network). This number increases with the depth as follows: $16 \rightarrow k{\cdot}16 \rightarrow k{\cdot}32 \rightarrow k{\cdot}64$. We halve the frequency resolution by using a stride of 2 each time we increase the number of channels except for the first time. On top of these layers are two BLSTM layers with 512 units for each direction and a final fully connected layer. We call this configuration Wide Residual BLSTM Network (WRBN) (or "W" for short in Section 3).

For training, we first extract the alignments with the baseline back-end and the GEV front end using all six channels. We then train this network with a cross-entropy criterion and Adam [17]. To prevent overfitting we use dropout on the input of each layer. Additionally we use it on the hidden-hidden transitions of the BLSTM. Here, we sample the mask once per sequence [18]. We use 80 dimensional mean-normalized log-Mel filter bank features as input. Their delta and delta-delta features act as extra channels. The network is trained on the unprocessed training data from all six channels. Instead of training on a mini-batch of a few frames, we train it on a whole utterance with full backpropagation through time. This allows the WRN and BLSTM to exploit the full temporal context. Also, it enables us to use Batch-Normalization (BN) [19] in an effective way. Here, we do not rely on statistics estimated on the training or development data. We can normalize the networks activations using the utterance statistics during test time. This is not possible when working with frames because their high correlation due to their overlap prohibits a good estimation of the statistics. For (speaker) adaptation, we train an additional layer consisting of a 80×80 weight matrix for each speaker and each track. That layer with tied weights is applied to all three feature channels equally.

2.2.4. *RNN language model lattice rescoring*

For the RASR back end (R) we carried out lattice rescoring [20] with an LSTM-RNN language model [21, 22] as follows. For each of the two rescoring steps (speaker independent and adapted) shown in Figure 1, a specific model was used. The first pass lattices were always rescored with a small LSTM model we refer to as L1. For the rescoring of the second pass lattices we compared our own LSTM model L2 with the baseline RNN model LB. The model L1 is based on a one-layer standard LSTM while L2 is based on a 3-layer LSTM with highway connections. The size of all hidden layers is set to 500 for both models. For the training of model L2 we applied a dropout rate of 20% on the non-recurrent connections. The output layer is factorized with word classes trained using the exchange algorithm [23]. We used 100 classes for L1 and 70 for L2. For the training of the model L2, the sentences with high OOV rates were removed from the training data, exactly as described in [24]. The model L1 was trained without this pre-processing. The interpolation weights between the baseline 5-gram count model and the LSTM model were optimized w.r.t. the perplexity on the development data [25]. We used RWTH's open-source toolkit rwthlm [7] for both training and rescoring.

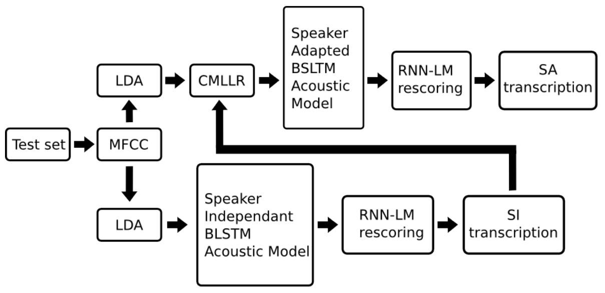

Figure 1: Decoding pipeline

For the UPB back end (W) we employ a two layer LSTM language model with 650 hidden units each – similar to the example provided by [26]. Instead of training on an endless word stream (initial state of next batch is end state of current batch), we found that training on full sentences from the provided language model training data in a random mini-batch improved cross validation scores slightly. Again we use Adam [17] for optimization for 39 epochs and apply a dropout rate of 50% (this time in the vertical connections only). Global gradient clipping with a maximum value of 5 is used. All weight matrices and bias vectors, including the embedding matrix, are initialized with random weights sampled from a uniform distribution in $[-0.1, 0.1]$. We experiment with restricted training sets limiting the maximum number of unknown symbols during training. This yielded reduced cross validation perplexities. Nevertheless, we finally selected a model trained on unrestricted training data, since this lead to the lowest WER on the dev set.

2.3. System combination

For every track we obtain the final recognition result by performing confusion network combination (CNC) of multiple systems. The lattices are first converted to individual confusion networks [27, 28] and the combination is performed by aligning the confusion networks in the order of increasing word error rate. We optimize the system weights w.r.t. the WER on the real dev set using the downhill simplex algorithm. The frame-wise CN construction algorithm is described in greater detail in Section 4.4.4 of [29].

3. Experimental evaluation

The following section gives an overview over the effects of the different components of the final system in the 6 channel track. The following notation is used. The columns FE and BE describe the front end and back end used in the decoding step, respectively. The front end is either the baseline model (BL), the superdirective beamformer (SD), the minimum variance distortionless response beamformer (MV) or the GEV beamformer (GE). The RWTH back end (R) is described in Sections 2.2.2 and the UPB back end (W) is summarized in Section 2.2.3. See [9] for the description of the baseline Kaldi back end (K). All optimization have been done exclusively on the real dev set. The results on the simulated data are only provided for reference. All tables show absolute word error rates (WER) in percent.

3.1. MVDR configurations

The MVDR beamformer uses a slightly different configuration to estimate the spatial correlation matrix of noise used in Eq. (2) for the 2ch and 6ch track. As described in Section 2.1.2, for the 6ch track the spatial correlation matrix of noise is estimated using the time-frequency masks derived from the complex Gaussian Mixture Model (MV-NoiseMasks), while for the 2ch track the matrix is estimated from 400 ms to 800 ms of context immediately before the utterance (MV-NoiseContext).

The respective configurations have shown to yield better results on the real dev set, which has been used exclusively for selection and optimization. Table 1 shows the recognition performance of the MVDR beamformer with the two configurations. In the following, MV will denote the MVDR front end with the best performing configuration for each track, i.e., MV-NoiseMasks for the 6ch track and MV-NoiseContext for the 2ch track. These systems were trained on the FC+B data set.

Table 1: Configuration comparison for MVDR beamformer

Track	MVDR configuration	Dev		Test	
		real	simu	real	simu
6ch	MV-NoiseContext	3.69	4.44	5.56	6.26
	MV-NoiseMasks	3.57	4.73	5.58	6.28
2ch	MV-NoiseContext	4.94	7.09	8.77	10.17
	MV-NoiseMasks	5.09	7.61	10.53	12.01

3.2. Speaker adaptation and lattice rescoring

Table 2 shows the effect of speaker adaptation (SA) using Constrained Maximum Likelihood Linear Regression (CMLLR) and lattice rescoring using an RNN-LM. It can be seen that both components have a significant influence on the performance. A relative improvement of 40% can be reached by using the RWTH back end presented in Section 2.2.2 with speaker adaptation and lattice rescoring compared to the baseline back end (compare first and last row).

Table 2: Effect of speaker adaptation (SA) and lattice rescoring on the 6ch track evaluated on system trained on the FC training set.

System				Dev		Test	
FE	BE	SA	RNN-LM	real	simu	real	simu
BL	K	+	LB	5.75	6.76	11.49	10.89
	R	-	-	7.80	8.71	11.81	13.89
		-	L1	5.87	6.43	9.35	10.55
		+	-	6.22	8.10	9.69	11.70
		+	LB	5.76	7.37	8.92	10.67
		+	L2	4.34	5.65	6.83	8.16

3.3. Front end performance

In order to compare the front ends, we evaluate speaker adapted systems trained on the FC training set and apply LM rescoring with RNNs. Table 3 shows the performance of different beamformers on the 6ch track test data using both Kaldi and RWTH back ends. The results indicate that all front ends presented here have a positive effect on the ASR performance on the real data. The best front end (GE) leads to a further relative improvement of up to 41% over the baseline front end (BL).

Table 3: Comparison of front ends on a speaker adapted model with lattice rescoring for the 6ch track.

System		Dev		Test	
FE	BE	real	simu	real	simu
BL	K	5.75	6.76	11.49	10.89
SD		5.47	6.34	11.47	10.42
MV		4.63	5.44	8.73	8.62
GE		3.70	3.72	5.76	4.24
BL	R	4.34	5.65	6.83	8.16
SD		3.89	5.14	6.59	7.99
MV		3.90	5.23	5.65	8.36
GE		3.27	3.41	4.02	3.93

3.4. Including beamformed signal in the training

Table 4 shows the effect of extending the training set of the speaker adapted model by the pre-processed training data. The recognition is performed with the RWTH back end and includes lattice rescoring with the model L2. It can be seen that only minor improvements on the real data can be achieved. In the case of the baseline front end (BL) the performance on the simulated data even degrades. Nevertheless we decided to use the extended training set (FC+B) for further experiments.

Table 4: Effect of enhancing the training set FC consisting of channels 1,3-6 by the data obtained by pre-processing the training set with the matching front end on the 6ch track (FC+B).

System		Dev		Test	
FE	Training set	real	simu	real	simu
BL	FC	4.34	5.65	6.83	8.16
	FC+B	4.11	5.77	6.82	8.53
SD	FC	3.89	5.14	6.59	7.99
	FC+B	3.74	5.03	6.52	7.84
MV	FC	3.90	5.23	5.65	8.36
	FC+B	3.57	4.73	5.58	6.28
GE	FC	3.27	3.41	4.02	3.93
	FC+B*	3.05	2.79	3.77	2.67

* This system has not been available at time of evaluation and is only included here for completeness

3.5. System combination

Table 5 shows the single systems used for system combination and Table 6 shows the result of combining multiple systems. It can be seen that in case of using only the RWTH back end (R) each additional front end has a positive effect on the performance on the real data. However, the optimization algorithm reduces the weight of the system using the baseline front end (BL) to zero when we include UPB's back end W (last row). The best result obtained for the real evaluation data on the 6 channel track is 2.91%, which is a relative improvement of almost 75% over the baseline system.

Table 5: Single systems used for system combination in the 6 channel track

System				Dev		Test	
ID	FE	BE	Training set	real	simu	real	simu
1	BL	R	CH1,3-6+BL	4.11	5.77	6.82	8.53
2	SD		CH1,3-6+SD	3.74	5.03	6.52	7.84
3	MV		CH1,3-6+MV	3.57	4.73	5.58	6.28
4	GE		CH1,3-6	3.27	3.41	4.02	3.93
5	GE	W	CH1-6	2.73	2.34	3.48	2.76

Table 6: Results of CNC system combination of different systems for the 6 channel track

System weights					Dev		Test	
5	4	3	2	1	real	simu	real	simu
	0.50	0.50			2.75	3.13	3.57	3.56
	0.33	0.33	0.33		2.70	3.18	3.55	3.77
	0.40	0.25	0.25	0.10	2.61	3.07	3.40	3.46
0.45	0.55				2.48	2.47	3.12	2.90
0.43	0.33	0.34			2.25	2.30	2.98	2.61
0.35	0.20	0.20	0.25		2.19	2.34	2.91	2.68
0.35	**0.20**	**0.20**	**0.25**	**0.00**	**2.19**	**2.34**	**2.91**	**2.68**

3.6. Systems for 1 and 2 channel tracks

Table 7 shows the results of the single systems and the final system combination in the 1ch and 2ch track of the challenge. All shown systems have been included in the system combination. In the 1ch track no pre-processing has been used. Table 8 shows the breakdown of the results by environment for the best systems in each track.

Table 7: Single system, system combination results for the 1ch and 2ch track and best system combination result for the 6ch track.

Tr.	System			Dev		Test	
	FE	BE	Training set	real	simu	real	simu
1ch	-	K	CH5	11.58	12.99	23.77	20.82
	-	R	CH1,3-6	7.42	9.86	12.02	15.22
	-	W	CH1-6	5.19	6.69	9.34	11.11
	COM_1ch			**5.14**	**7.40**	**9.29**	**12.36**
2ch	BL	K	CH5	8.25	9.51	16.63	15.33
	MV			8.11	9.12	16.41	14.03
	SD			8.03	9.15	15.97	15.15
	GE			6.93	8.03	13.76	9.90
	BL	R	CH1,3-6+BL	5.43	7.35	9.12	11.74
	MV		CH1,3-6+MV	4.94	7.09	8.77	10.17
	SD		CH1,3-6+SD	5.53	7.54	9.60	12.33
	GE		CH1,3-6	4.90	6.48	7.69	7.49
	GE	W	CH1-6	3.54	4.05	5.96	5.16
	COM_2ch			**3.02**	**4.04**	**5.32**	**5.27**
6ch	COM_6ch			**2.19**	**2.34**	**2.91**	**2.68**

4. Post evaluation results

The following results were not part of the submitted system for the 4th CHiME challenge. Table 9 shows the performance gain obtained by the sequence-discriminative training of the BLSTM acoustic model w.r.t. the sMBR criterion [30] in the 6ch track. The cross-entropy (CE) model is trained on the FC data set using the RWTH back end and the GE front end. This model is used to initialize the sMBR training. The lattices for the sMBR training were generated with a 3-gram language model. During the training we use state priors which were calculated from the output layer of the CE model as was proposed in [31]. In order to prevent overfitting we used CE-smoothing [32] with a factor of 0.1. Table 10 shows that replacing the CE model (4) by the sMBR model (4+) in the combination reduces the WER from 2.9 to 2.7% on the real test set.

Table 8: Breakdown of the best results by environment.

Tr.	Environment	Dev		Test	
		real	simu	real	simu
1ch	BUS	6.59	5.88	13.12	8.91
	CAF	5.90	9.88	9.84	15.11
	PED	3.45	6.14	7.57	11.95
	STR	4.60	7.71	6.63	13.45
2ch	BUS	3.91	3.44	7.52	3.68
	CAF	3.07	5.43	5.04	6.33
	PED	2.36	3.38	4.39	5.49
	STR	2.73	3.91	4.33	5.57
6ch	BUS	2.61	2.06	3.16	2.19
	CAF	2.01	2.92	2.65	2.95
	PED	2.05	2.12	2.93	2.99
	STR	2.11	2.26	2.91	2.60

Table 9: Effect of sequence training of the BLSTM acoustic model in the RWTH back end. Results with the GE front end on the 6ch track.

System		Dev		Test	
ID	Criterion	real	simu	real	simu
4	CE	3.27	3.41	4.02	3.93
4+	sMBR	2.77	3.11	3.43	3.30

Table 10: Effect of replacing the CE system (4) by the sMBR trained system (4+) in the system combination. Results on the 6ch track.

System weights						Dev		Test	
5	4	4+	3	2	1	real	simu	real	simu
0.35	0.20		0.20	0.25	0.00	2.19	2.34	2.91	2.68
0.30		0.25	0.20	0.10	0.15	2.09	2.32	2.71	2.47

5. Conclusion

In this paper we presented a detailed analysis of the acoustic models developed by RWTH, UPB and FORTH for the 4th CHiME challenge. Our joint submission based on confusion network combination of multiple systems scored second in each of the three tracks of the challenge. More specifically, we compared four different front-ends and found that the BLSTM supported GEV-beamformer consistently leads to the best ASR results in 2ch and 6ch tracks. Further we found that extending the training data set by the beamformed data only works well on real test data.

We plan to further investigate the sequence training of BLSTM back ends, since the post evaluation results have shown clearly, that further performance gain can be achieved and transferred to the final system combination.

6. Acknowledgments

This research was partially supported by Ford Motor Company and the Deutsche Forschungsgemeinschaft (DFG) under Contract No. Schl2043/11-1 and No. Ha3455/11-1. The research has also received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 644283, project "LISTEN".

7. References

[1] H. Cox, R. Zeskind, and M. Owen, "Robust adaptive beamforming," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 35, no. 10, pp. 1365–1376, Oct. 1987.

[2] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Scottsdale, AZ, USA, Dec. 2015, pp. 504–511.

[3] T. Higuchi, N. Ito, T. Yoshioka, and T. Nakatani, "Robust MVDR beamforming using time-frequency masks for online/offline ASR in noise," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Shanghai, China, Mar. 2016, pp. 5210–5214.

[4] J. Heymann, L. Drude, and R. Haeb-Umbach, "Neural network based spectral mask estimation for acoustic beamforming," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Shanghai, China, Mar. 2016, pp. 196–200.

[5] S. Wiesler, A. Richard, P. Golik, R. Schlüter, and H. Ney, "RASR/NN: The RWTH neural network toolkit for speech recognition," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Florence, Italy, May 2014, pp. 3313–3317.

[6] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, and H. Ney, "RETURNN: The RWTH extensible training framework for universal recurrent neural networks," *arXiv preprint arXiv:1608.00895, submitted for publication at ICASSP 2017*, 2016.

[7] M. Sundermeyer, R. Schlüter, and H. Ney, "rwthlm - the RWTH Aachen University neural network language modeling toolkit," in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 2093–2097.

[8] J. Heymann, L. Drude, and R. Haeb-Umbach, "Wide residual BLSTM network with discriminative speaker adaptation for robust speech recognition," *submitted to the CHiME 4 workshop*, 2016.

[9] E. Vincent, S. Watanabe, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *submitted to Computer Speech and Language*, 2016.

[10] B. F. Cron and C. H. Sherman, "Spatial-correlation functions for various noise models," *Journal of the Acoustical Society of America*, vol. 34, no. 11, pp. 1732–1736, Nov. 1962.

[11] E. Warsitz and R. Haeb-Umbach, "Blind acoustic beamforming based on generalized eigenvalue decomposition," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 15, no. 5, pp. 1529–1539, Jul. 2007.

[12] J. Heymann, L. Drude, A. Chinaev, and R. Haeb-Umbach, "BLSTM supported GEV beamformer front-end for the 3rd CHiME challenge," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Scottsdale, AZ, USA, Dec. 2015, pp. 444–451.

[13] T. Dozat, "Incorporating Nesterov momentum into Adam," Stanford University, CS 229 Machine Learning, Tech. Rep., 2015. [Online]. Available: http://cs229.stanford.edu/proj2015/054_report.pdf

[14] D. Johnson. (2004) QuickNet, Speech group at ICSI, Berkeley. [Online]. Available: http://www1.icsi.berkeley.edu/Speech/faq/nn-train.html

[15] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," in *Proc. Int. Conf. on Learning Representations (ICLR) Workhop Track*, San Juan, Puerto Rico, May 2016.

[16] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference*, York, UK, Sep. 2016.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learning Representations*, San Diego, CA, USA, May 2015.

[18] S. Semeniuta, A. Severyn, and E. Barth, "Recurrent dropout without memory loss," *CoRR*, vol. abs/1603.05118, 2016. [Online]. Available: http://arxiv.org/abs/1603.05118

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. on Machine Learning*, Lille, France, Jul. 2015, pp. 448–456.

[20] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 661–665.

[21] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling." in *Proc. Interspeech*, Portland, OR, USA, Sep. 2012, pp. 194–197.

[22] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[23] S. Martin, J. Liermann, and H. Ney, "Algorithms for bigram and trigram word clustering," *Speech communication*, vol. 24, no. 1, pp. 19–37, 1998.

[24] T. Yoshioka, N. Ito, M. Delcroix, A. Ogawa, K. Kinoshita, M. Fujimoto, C. Yu, W. J. Fabian, M. Espi, T. Higuchi *et al.*, "The NTT CHiME-3 system: Advances in speech enhancement and recognition for mobile multi-microphone devices," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Scottsdale, AZ, USA, Dec. 2015, pp. 436–443.

[25] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *Proc. Interspeech*, Denver, CO, USA, Sep. 2002, pp. 901–904.

[26] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[27] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech & Language*, vol. 14, no. 4, pp. 373–400, 2000.

[28] G. Evermann and P. Woodland, "Posterior probability decoding, confidence estimation and system combination," in *NIST Speech Transcription Workshop*, vol. 27, 2000, p. 78.

[29] B. Hoffmeister, "Bayes risk decoding and its application to system combination," Ph.D. dissertation, RWTH Aachen University, Computer Science Department, Aachen, Germany, Jul. 2011.

[30] D. Povey and B. Kingsbury, "Evaluation of proposed modifications to MPE for large scale discriminative training," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Honolulu, HI, USA, Apr. 2007, pp. 321–324.

[31] V. Manohar, D. Povey, and S. Khudanpur, "Semi-supervised maximum mutual information training of deep neural network acoustic models," in *Proc. Interspeech*, Dresden, Germany, Sep. 2015, pp. 2630–2634.

[32] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 2013, pp. 6664–6668.