The Hitachi/JHU CHiME-5 system: Advances in speech recognition for everyday home environments using multiple microphone arrays

Naoyuki Kanda\textsuperscript{1}, Rintaro Ikeshita\textsuperscript{1}, Shota Horiguchi\textsuperscript{1}, Yusuke Fujita\textsuperscript{1}, Kenji Nagamatsu\textsuperscript{1}, Xiaofei Wang\textsuperscript{2}, Vimal Manohar\textsuperscript{2}, Nelson Enrique Yalta Soplin\textsuperscript{2}, Matthew Maciejewski\textsuperscript{2}, Szu-Jui Chen\textsuperscript{2}, Aswin Shanmugam Subramanian\textsuperscript{2}, Ruizhi Li\textsuperscript{2}, Zhiqi Wang\textsuperscript{2}, Jason Naradowsky\textsuperscript{2}, L. Paola Garcia-Perera\textsuperscript{2}, Gregory Sell\textsuperscript{2}
Step-by-Step Improvements for Dev

Word error rate (%)

- Baseline
- Data Augmentation
- CNN-TDNN-LSTM (1ch)
- CNN-TDNN-RBiLSTM (1ch)
- 4ch Input
- LF-sMBR
- WPE
- CGMM-Mask-MVDR
- Mask-NN-MVDR
- AM combination
- Frontend combination
- Frontend combination
- Hypothesis deduplication
- RNN-LM
- Array combination

Single-array

Multiple-array

AM
Frontend
Decoding
LM
Acoustic Model Training Pipeline

Step 1.
GMM-AM
- 1ch worn L
- 1ch worn R
- 1ch worn L+R

Step 2.
Alignment & Cleanup
- 1ch worn L+R
- GMM
- Alignment & Cleanup
- “Cleaned” 1ch worn L+R & phone-state alignment
- Full set
- Alignment Expansion
- “Cleaned” full set & phone-state alignment

Step 3.
1ch AM
- “Cleaned” full set & phone-state alignment
- iVector Training
- iVector Extraction
- LF-MMI AM Training
- LF-MMI AM (1ch)
- Acoustic Feature Extraction

Step 4.
4ch AM
- LF-MMI AM (1ch)
- Add 4ch input branch
- LF-MMI AM (4ch)
- Weighted iVector Extraction
- LF-MMI AM Training
- LF-MMI AM (4ch)
- LF-sMBR AM Training [1]
- LF-sMBR AM (4ch)

Acoustic Model Training Pipeline

**Step 1. GMM-AM**
1ch worn L
1ch worn R
1ch worn L+R

- GMM (mono) → GMM (tri) → GMM (LDA-MLLT) → GMM (SAT) → GMM

**Data augmentation**
40h -> 4,500h

**Step 2. Alignment**
1ch worn L+R

- Alignment & Cleanup
- “Cleaned” 1ch worn L+R & phone-state alignment

**Step 3. 1ch AM**
“Cleaned” full set & phone-state alignment

- iVector Training → iVector Extraction → LF-MMI AM Training → LF-MMI AM (1ch)
- Acoustic Feature Extraction

**Step 4. 4ch AM**
Array 1(4ch) → Array 6(4ch)

- Add 4ch input branch
- Weighted iVector Extraction → LF-MMI AM Training → LF-MMI AM (4ch)
- Acoustic Feature Extraction

### Effect of data augmentation with baseline AM

<table>
<thead>
<tr>
<th>Data</th>
<th>Data Augmentation</th>
<th>Training Epoch</th>
<th>Worn-Dev</th>
<th>Ref-Array-Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>L, R, L+R</td>
<td>✓</td>
<td>4</td>
<td>44.05</td>
<td>79.65</td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>✓</td>
<td>4</td>
<td>44.49</td>
<td>78.72</td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>✓</td>
<td>4</td>
<td>48.92</td>
<td>78.51</td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>✓</td>
<td>2</td>
<td>45.82</td>
<td>77.26</td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>✓</td>
<td>1</td>
<td>45.37</td>
<td>76.31</td>
</tr>
</tbody>
</table>

(*) Reverb. & noise perturbation was applied only for worn microphone data.

---

<table>
<thead>
<tr>
<th>Worn (Raw, CH1)</th>
<th>Array (BeamFormIt)</th>
<th>Speed &amp; Volume</th>
<th>Reverb. &amp; Noise(*)</th>
<th>Bandpass</th>
</tr>
</thead>
<tbody>
<tr>
<td>L, R, L+R</td>
<td>1</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>1</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>1 ... 6</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>1 ... 6</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>L, R, L+R</td>
<td>1 ... 6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

---

Speed: 0.9, 1.0, 1.1  
Volume: 0.125 – 2.0  
Reverberation: Generate impulse responses of simulated rooms by image method. Follow the settings of {small, medium}-size rooms in [1].  
Noise: Add non-speech region of array data with SNR of {20,15,10,5, 0}  
Bandpass: Randomly-selected frequency band was cut off. (leave at least 1,000 Hz band within the range of less than 4,000 Hz)

Acoustic Model Training Pipeline

**Step 1.**
GMM-AM

1 ch worn L
1 ch worn R
1 ch worn L+R

- GMM (mono) → GMM (tri) → GMM (LDA-MLLT) → GMM (SAT) → GMM

**Step 2.**
Alignment

1 ch worn L+R

- GMM → Alignment & Cleanup → “Cleaned” 1 ch worn L+R & phone-state alignment → Alignment Expansion → “Cleaned” full set & phone-state alignment

**Step 3.**
1 ch AM

- “Cleaned” full set & phone-state alignment → iVector Training → iVector Extraction → LF-MMI AM Training → LF-MMI AM (1 ch)

**Step 4.**
4 ch AM

- LF-MMI AM (1 ch) → Add 4 ch input branch → LF-MMI AM (4 ch)

4 ch AM

- Array 1(4 ch)
  - CH1 → Weighted iVector Extraction → LF-MMI AM Training → LF-MMI AM (4 ch)

- Array 6(4 ch)
  - CH1, 2, 3, 4 → Acoustic Feature Extraction → LF-MMI AM Training


4ch CNN-TDNN-RBiLSTM

\[
\log |x_{i,f,t}| \\
i \in \{1, 2, 3, 4\}
\]

\[
\cos(\angle(x_{i,f,t}) - \angle(x_{1,f,t})) \quad (i = 2, 3, 4),
\sin(\angle(x_{i,f,t}) - \angle(x_{1,f,t})) \quad (i = 2, 3, 4).
\]
4ch CNN-TDNN-RBiLSTM

(1) LF-MMI update

log $|x_{i,f,t}|$

$i = 1, 2, 3, 4$
4ch CNN-TDNN-RBiLSTM

\[
\log |x_{i,f,t}| \\
 i = 1, 2, 3, 4
\]

(2) LF-MMI update

\[
\cos(\angle(x_i,f,t) - \angle(x_{1,f,t})) \quad (i = 2, 3, 4), \\
\sin(\angle(x_i,f,t) - \angle(x_{1,f,t})) \quad (i = 2, 3, 4).
\]
4ch CNN-TDNN-RBiLSTM

4ch-branch:

\[
\log |x_{i,f,t}| \\
\quad i (= 1, 2, 3, 4)
\]

(3) LF-sMBR update

\[
\begin{align*}
\cos(\angle(x_{i,f,t}) - \angle(x_{1,f,t})) & \quad (i = 2, 3, 4), \\
\sin(\angle(x_{i,f,t}) - \angle(x_{1,f,t})) & \quad (i = 2, 3, 4).
\end{align*}
\]
Step-by-Step Improvements for Dev

![Graph showing improvements in word error rate over single-array and multiple-array systems.](Image)

- **AM (Acoustic Model)**
- **Frontend**
- **Decoding**
- **LM (Language Model)**

**Single-array** vs. **Multiple-array** systems:
- **Baseline**
- **Data Augmentation**
- **CNN-TDNN-LSTM (1ch)**
- **CNN-TDNN-RBiLSTM (1ch)**
- **4ch Input**
- **LF-sMBR**
- **WPE**
- **CGMM-Mask-MVDR**
- **Mask-NN-MVDR**
- **AM combination**
- **Frontend combination**
- **Hypothesis deduplication**
- **Array combination**

© Hitachi, Ltd. 2018. All rights reserved.
Complex Gaussian Mixture Model

• 3-class mixture: target, non-target, and noise

\[ y_f(t) \sim \alpha_{tgt} \mathcal{N}_\mathbb{C}(0, v_f^{tgt}(t)R_f^{tgt}) + \alpha_{nontgt} \mathcal{N}_\mathbb{C}(0, v_f^{nontgt}(t)R_f^{nontgt}) + \alpha_{noise} \mathcal{N}_\mathbb{C}(0, v_f^{noise}(t)R_f^{noise}) \]

• Mask estimation using EM Algorithm \(\rightarrow\) MVDR-based Beamformer

1. **Train mask estimation (ME) network [1][2]**
   by using mixture of speech (worn non-speaker-overlapped region) and noise (array non-speech region) in the training set

   ![Diagram of Mask Estimation Neural Network]

   ```python
   def gate(x):
       if input.speaker == target_speaker:
           return x
       else:
           return 0
   ```

   (*) we used only non-overlapped regions for adaptation

2. **Speaker adaptation**

   ![Diagram of Speaker Adaptation]

3. Mask inference
Target speaker’s mask is selected only if target speaker’s output value is higher than all other non-targets values.

Example: P01(target) and P02(non-target)
Step-by-Step Improvements for Dev

Word error rate (%)

- **AM**
  - Frontend
  - Decoding

- **Baseline**
- **Data Augmentation**
- **CNN-TDNN-LSTM (1ch)**
- **CNN-TDNN-RBiLSTM (1ch)**
- **4ch Input**
- **LF-sMBR**
- **WPE**
- **CGMM-Mask-MVDR**
- **Mask-NN-MVDR**
- **AM combination**
- **Array combination**
- **RNN-LM**

- **Single-array**
- **Multiple-array**
Language Modeling

- Recurrent neural network based word-LM
  - 2 layer LSTM with 512 nodes, 50% dropout
  - 512 dim embeddings
  - PyTorch implementation

  \[
  = \begin{array}{c|c|c}
    \text{Single-array} & \text{without RNN-LM} & \text{with RNN-LM} \\
    \hline
    \text{Single-array} & 56.40 & 55.15 \\
    \text{Multiple-array} & 54.00 & 52.38 \\
  \end{array}
  \]
  \[
  (* \text{ Results with model combination and hypothesis deduplication })
  \]
Step-by-Step Improvements for Dev

Word error rate (%)

- AM
- Frontend
- Decoding
- LM

Single-array

Multiple-array

- Baseline + Data Augmentation + CNN-TDNN-LSTM (1ch) + CNN-TDNN-RBiLSTM (1ch) + 4ch Input + LF-sMBR + WiFi + CGMM-Mask-MVDR + Mask-NN-MVDR + AM combination + Frontend combination + Hypothesis deduplication + Array combination

© Hitachi, Ltd. 2018. All rights reserved.
Decoding

- Hypotheses combination by N-best ROVER
  - 6 AMs := CNN-TDNN-\{LSTM, BiLSTM, RBiLSTM\} x {3500, 7000} senones
  - 2 Front-ends := Mask Network, CGMM
  - 6 Arrays

We didn’t select array, instead combined hypotheses from each array.

```
| Array1 | Front-end1 | AM1   | Hypothesis_1,1,1 |
| Array1 | Front-end1 | AM2   | Hypothesis_1,1,2 |
| Array2 | Front-end1 | AM3   |                  |
| Array6 | Front-end2 | AM5   | Hypothesis_6,2,5 |
| Array6 | Front-end2 | AM6   | Hypothesis_6,2,6 |
```

Result
Step-by-Step Improvements for Dev

AM

Frontend

Decoding

LM

Multiple-array

Single-array

© Hitachi, Ltd. 2018. All rights reserved.
Hypothesis Deduplication (HD)

- Same words were sometimes recognized for overlapped utterances

<table>
<thead>
<tr>
<th>P05</th>
<th>um</th>
<th>yeah</th>
<th>0.999999</th>
<th>0.858049</th>
</tr>
</thead>
<tbody>
<tr>
<td>P08</td>
<td>can</td>
<td>i</td>
<td>help</td>
<td>with</td>
</tr>
<tr>
<td></td>
<td>0.846968</td>
<td>0.847141</td>
<td>0.753396</td>
<td>0.637141</td>
</tr>
</tbody>
</table>

- Duplicated words with lower confidence were excluded from the hypothesis.
  - HD was applied after ROVER, so precise time boundary could not be used. Minimum edit distance-based word alignment was used to detect word duplication.

<table>
<thead>
<tr>
<th>WER (%) for Dev set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>without HD</th>
<th>with HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-array</td>
<td>56.44</td>
<td>55.15</td>
</tr>
<tr>
<td>1.3% impr.</td>
<td>1.3% impr.</td>
<td></td>
</tr>
<tr>
<td>Multiple-array</td>
<td>53.69</td>
<td>52.38</td>
</tr>
<tr>
<td>1.3% impr.</td>
<td>1.3% impr.</td>
<td></td>
</tr>
</tbody>
</table>

(*) Results with RNN-LM
Final results & Conclusion
## Final Results

### WER (%) without RNN-LM / with RNN-LM

<table>
<thead>
<tr>
<th>Track</th>
<th>Session</th>
<th>Kitchen</th>
<th>Dining</th>
<th>Living</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-array</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>S02</td>
<td>66.37 / 65.13</td>
<td>56.79 / 55.42</td>
<td>50.89 / 49.54</td>
<td><strong>56.40 / 55.15</strong></td>
</tr>
<tr>
<td></td>
<td>S09</td>
<td>55.89 / 55.24</td>
<td>55.94 / 54.37</td>
<td>51.57 / 50.15</td>
<td></td>
</tr>
<tr>
<td>Eval</td>
<td>S01</td>
<td>59.42 / 57.62</td>
<td>44.18 / 41.81</td>
<td>63.85 / 62.33</td>
<td><strong>50.36 / 48.20</strong></td>
</tr>
<tr>
<td></td>
<td>S21</td>
<td>52.11 / 49.68</td>
<td>42.14 / 39.78</td>
<td>46.71 / 44.59</td>
<td></td>
</tr>
<tr>
<td><strong>Multiple-array</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev</td>
<td>S02</td>
<td>61.05 / 59.31</td>
<td>54.56 / 52.96</td>
<td>50.47 / 48.95</td>
<td><strong>54.00 / 52.38</strong></td>
</tr>
<tr>
<td></td>
<td>S09</td>
<td>51.87 / 50.64</td>
<td>52.46 / 50.69</td>
<td>52.48 / 50.46</td>
<td></td>
</tr>
<tr>
<td>Eval</td>
<td>S01</td>
<td>59.82 / 57.01</td>
<td>43.59 / 41.22</td>
<td>62.28 / 60.67</td>
<td><strong>50.59 / 48.24</strong></td>
</tr>
<tr>
<td></td>
<td>S21</td>
<td>54.70 / 51.59</td>
<td>44.12 / 42.17</td>
<td>45.95 / 43.82</td>
<td></td>
</tr>
</tbody>
</table>
### Final Results

#### WER (%)

<table>
<thead>
<tr>
<th>Track</th>
<th>Session</th>
<th>Kitchen</th>
<th>Dining</th>
<th>Living</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-array</strong></td>
<td>Dev</td>
<td>S02 S09</td>
<td>66.37 / 65.13</td>
<td>56.79 / 55.42</td>
<td>50.89 / 49.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>55.89 / 55.24</td>
<td>55.94 / 54.37</td>
<td>51.57 / 50.15</td>
</tr>
<tr>
<td></td>
<td>Eval</td>
<td>S01 S21</td>
<td>59.42 / 57.62</td>
<td>44.18 / 41.81</td>
<td>63.85 / 62.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>52.11 / 49.68</td>
<td>42.14 / 39.78</td>
<td>46.71 / 44.59</td>
</tr>
<tr>
<td><strong>Multiple-array</strong></td>
<td>Dev</td>
<td>S02 S09</td>
<td>61.05 / 59.31</td>
<td>54.56 / 52.96</td>
<td>50.47 / 48.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>51.87 / 50.64</td>
<td>52.46 / 50.69</td>
<td>52.48 / 50.46</td>
</tr>
<tr>
<td></td>
<td>Eval</td>
<td>S01 S21</td>
<td>59.82 / 57.01</td>
<td>43.59 / 41.22</td>
<td>62.28 / 60.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>54.70 / 51.59</td>
<td>44.12 / 42.17</td>
<td>45.95 / 43.82</td>
</tr>
</tbody>
</table>

- Array combination by ROVER worked well for dev, but not effective for eval set.
  - Why? Different types of rooms? Speaker-array distance?

- Anyway, better array combination methods should be pursued.
Our contributions
- Multiple data augmentation
- 4-ch AM with Residual BiLSTM
- Speaker adaptive mask estimation network / CGMM-based beamformer
- Hypothesis Deduplication
- Array combination by ROVER (found not effective for evaluation set)

Our results
- 48.2% WER for evaluation set
- 2nd ranked, with only 2.1 point difference to the best result

Thank you for your attention!
Appendix
Comparison of AM Architectures

<table>
<thead>
<tr>
<th>Model</th>
<th>Input</th>
<th>Training</th>
<th>Worn-Dev</th>
<th>Ref-Array-Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>1ch</td>
<td>LF-MMI</td>
<td>45.37</td>
<td>76.31</td>
</tr>
<tr>
<td>CNN-TDNN-LSTM</td>
<td>1ch</td>
<td>LF-MMI</td>
<td>39.22</td>
<td>68.87</td>
</tr>
<tr>
<td>CNN-TDNN-BiLSTM</td>
<td>1ch</td>
<td>LF-MMI</td>
<td>40.04</td>
<td>68.42</td>
</tr>
<tr>
<td>CNN-TDNN-RBiLSTM</td>
<td>1ch</td>
<td>LF-MMI</td>
<td>39.21</td>
<td>67.46</td>
</tr>
<tr>
<td>CNN-TDNN-RBiLSTM</td>
<td>4ch</td>
<td>LF-sMBR [1]</td>
<td>n/a</td>
<td>64.54</td>
</tr>
<tr>
<td>CNN-TDNN-RBiLSTM</td>
<td>4ch</td>
<td>LF-sMBR [1]</td>
<td>n/a</td>
<td>64.25</td>
</tr>
</tbody>
</table>

## Comparison of Frontend Processing

<table>
<thead>
<tr>
<th>Front-end for 1ch input</th>
<th>Front-end for 4ch input</th>
<th>Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeamFormIt ( = Baseline)</td>
<td>Raw</td>
<td>64.28</td>
</tr>
<tr>
<td>Raw</td>
<td>Raw</td>
<td>63.79</td>
</tr>
<tr>
<td>WPE</td>
<td>WPE</td>
<td>63.49</td>
</tr>
<tr>
<td>CGMM-MVDR</td>
<td>WPE</td>
<td>62.53</td>
</tr>
<tr>
<td>Speaker adaptive mask NN-MVDR</td>
<td>WPE</td>
<td>62.09</td>
</tr>
</tbody>
</table>
Decoding

Hypotheses combination by N-best ROVER
- 6 AMs := CNN-TDNN-{LSTM, BiLSTM, RBiLSTM} x {3500, 7000} senones
- 2 Front-ends := Mask Network, CGMM
- 6 Arrays We didn’t select array. Instead we combined hypotheses from each array.

WER (%) for Dev set

<table>
<thead>
<tr>
<th>AM</th>
<th>Array</th>
<th>Frontend</th>
<th>Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>MaskNet</td>
<td>62.09</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>MaskNet</td>
<td>58.79</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>MaskNet, CGMM</td>
<td>57.55</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>MaskNet, CGMM</td>
<td>55.08</td>
</tr>
</tbody>
</table>

(*) Results w/o RNN-LM
Thank you