Developing streaming end-to-end models for automatic speech recognition in industry

Jinyu Li
E2E Models

Transducer

Sequence to sequence (S2S)
# E2E Models

<table>
<thead>
<tr>
<th></th>
<th>Transducer</th>
<th>S2S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attention mechanism</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Building block</td>
<td>RNN or Transformer</td>
<td>RNN or Transformer</td>
</tr>
<tr>
<td>Streaming</td>
<td>Natural</td>
<td>Need to covert full attention to partial attention</td>
</tr>
<tr>
<td>Ideal operation scenario</td>
<td>streaming</td>
<td>offline</td>
</tr>
</tbody>
</table>
E2E Models

RNN-Transducer (RNN-T)

Sequence to sequence (S2S)
High Performance RNN-T Model

Improving RNN-T Training/Modeling

• Saving GPU memory

• Improving Initialization

• Improving Encoder
High Memory Cost of RNN-T Training

• RNN-T model training has *high memory cost*

The tensor $z_{t,u}$ after encoder and prediction output combination has 3 dimensions: ($T, U, D$), while other models usually work on 2 dimensions.

• $T$: acoustic feature length
• $U$: token sequence length
• $D$: dimension of hidden output
Function Merging

After the joint network, there are 3 functions to get loss:

- Linear: \( h_{t,u} = W_y z_{t,u} + b_y \)
- Softmax: \( \Pr(k|t,u) = \text{softmax}(h^k_{t,u}) \)
- Loss: \( L = -\ln \Pr(y|x) \)

- With chain rule method, we need space for: \( z_{t,u}, h_{t,u}, \Pr(k|t,u), \frac{\partial L}{\partial \Pr(k|t,u)}, \frac{\partial \Pr(k|t,u)}{\partial h^k_{t,u}}, \frac{\partial h^k_{t,u}}{\partial W_y}, \frac{\partial h^k_{t,u}}{\partial b_y} \) and \( \frac{\partial h^k_{t,u}}{\partial z_{t,u}} \)

- With merging linear, softmax and loss, we only need space for: \( z_{t,u}, h_{t,u}, \frac{\partial L}{\partial W_y}, \frac{\partial L}{\partial b_y} \) and \( \frac{\partial L}{\partial z_{t,u}} \).

• Initializing the prediction network with a pre-trained LM is not effective.
 Initialization

- Initializing the prediction network with a pre-trained LM is not effective.

- Initializing the encoder network with
  - CTC criterion
  - CE criterion – alignment is needed: equally divide the word segment by the number of word piece units in this word.
Improving Encoder – Hybrid Model

• Contextual layer trajectory LSTM [1]
  • Decouple the tasks of temporal modeling and target classification with time-LSTM and depth-LSTM, respectively.
  • Use future context frames to incorporate more information for stronger encoder outputs

\[
\zeta_t^{l-1} = \sum_{\delta=0}^{\tau} V_{\delta}^{l-1} g_{t+\delta}^{l-1}
\]

\[
h_t^l = LSTM(h_{t-1}^l, h_t^{l-1})
\]

\[
g_t^l = LSTM(h_t^l, \zeta_t^{l-1})
\]

Improving Encoder – RNN-T Model

• Contextual LSTM (cLSTM)
  • Use future context frames to incorporate more information for stronger encoder outputs.
  • Element-wise product is used to save the computational cost.

\[
\zeta_t^{l-1} = \sum_{\delta=0}^{\tau} v_{\delta}^{l-1} \odot h_{t+\delta}^{l-1}
\]
\[
h_t^l = LSTM(h_{t-1}^l, \zeta_t^{l-1})
\]
Experiment Setup

• Training data:
  • **65 thousand** hours of transcribed anonymized Microsoft data

• Testing data:
  • **1.8 million** words test set covering 13 application scenarios.

• Hybrid models
  • Language model: 5-gram (5 Gb decoding graph)
  • Acoustic models
    • LSTM
    • contextual layer trajectory LSTM (cltLSTM)
High-Performance Hybrid Models

<table>
<thead>
<tr>
<th>Hybrid</th>
<th>CE WER%</th>
<th>MMI WER%</th>
<th>T/S WER%</th>
<th>Parameter number</th>
<th>Encoder lookahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>14.75</td>
<td>13.01</td>
<td>11.49</td>
<td>30 M</td>
<td>0</td>
</tr>
<tr>
<td>cltLSTM</td>
<td>11.15</td>
<td>10.36</td>
<td>9.34</td>
<td>63 M</td>
<td>480 ms</td>
</tr>
</tbody>
</table>

• Our hybrid model training recipe is highly optimized with 3-stage optimization.

RNN-T Models

We use $\text{MpN}_F\times L$ to denote the encoder structure and use $\text{MpN}_x2$ as the prediction network structure.

- $M$: the number of cells
- $N$: the projection layer size
- $F$: the number of lookahead frames at each layer
- $L$: the number of layers
# Impact of Initialization

<table>
<thead>
<tr>
<th>Models</th>
<th>Random</th>
<th>CTC</th>
<th>CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1600p800_4x6</td>
<td>10.55</td>
<td>10.40</td>
<td>9.33</td>
</tr>
</tbody>
</table>

- Learning alignment information for the encoder may help RNN-T training to focus more on reasonable forward-backward paths instead of all the paths.
- All RNN-T models we trained later use CE initialization.
## Comparison of RNN-T Models

<table>
<thead>
<tr>
<th>Encoder network</th>
<th>Layers</th>
<th>Lookahead Frames /layer</th>
<th>Cell size</th>
<th>Projection size</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1280p640_x6</td>
<td>6</td>
<td>0</td>
<td>1280</td>
<td>640</td>
<td>11.25</td>
</tr>
</tbody>
</table>


Comparison of RNN-T Models

<table>
<thead>
<tr>
<th>Encoder network</th>
<th>Layers</th>
<th>Lookahead Frames /layer</th>
<th>Cell size</th>
<th>Projection size</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1280p640_x6</td>
<td>6</td>
<td>0</td>
<td>1280</td>
<td>640</td>
<td>11.25</td>
</tr>
<tr>
<td>1280p640_4x6</td>
<td>6</td>
<td>4</td>
<td>1280</td>
<td>640</td>
<td>9.81</td>
</tr>
</tbody>
</table>
Comparison of RNN-T Models

<table>
<thead>
<tr>
<th>Encoder network</th>
<th>Layers</th>
<th>Lookahead Frames /layer</th>
<th>Cell size</th>
<th>Projection size</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1280p640_x6</td>
<td>6</td>
<td>0</td>
<td>1280</td>
<td>640</td>
<td>11.25</td>
</tr>
<tr>
<td>1280p640_4x6</td>
<td>6</td>
<td>4</td>
<td>1280</td>
<td>640</td>
<td>9.81</td>
</tr>
<tr>
<td>1600p800_4x6</td>
<td>6</td>
<td>4</td>
<td>1600</td>
<td>800</td>
<td>9.33</td>
</tr>
<tr>
<td>2048p640_4x6</td>
<td>6</td>
<td>4</td>
<td>2048</td>
<td>640</td>
<td>9.27</td>
</tr>
<tr>
<td>2048p640_4x8</td>
<td>8</td>
<td>4</td>
<td>2048</td>
<td>640</td>
<td>9.28</td>
</tr>
<tr>
<td>2560p800_4x6</td>
<td>6</td>
<td>4</td>
<td>2560</td>
<td>800</td>
<td>8.88</td>
</tr>
<tr>
<td>2560p800_2x6</td>
<td>6</td>
<td>2</td>
<td>2560</td>
<td>800</td>
<td>9.05</td>
</tr>
</tbody>
</table>
## Comparison of RNN-T Models

<table>
<thead>
<tr>
<th>Encoder network</th>
<th>Layers</th>
<th>Lookahead Frames /layer</th>
<th>Parameter number</th>
<th>Encoder lookahead</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1280p640_x6</td>
<td>6</td>
<td>0</td>
<td>62 M</td>
<td>0</td>
<td>11.25</td>
</tr>
<tr>
<td>1280p640_4x6</td>
<td>6</td>
<td>4</td>
<td>62 M</td>
<td>720 ms</td>
<td>9.81</td>
</tr>
<tr>
<td>1600p800_4x6</td>
<td>6</td>
<td>4</td>
<td>94 M</td>
<td>720 ms</td>
<td>9.33</td>
</tr>
<tr>
<td>2048p640_4x6</td>
<td>6</td>
<td>4</td>
<td>87 M</td>
<td>720 ms</td>
<td>9.27</td>
</tr>
<tr>
<td>2048p640_4x8</td>
<td>8</td>
<td>4</td>
<td>119 M</td>
<td>960 ms</td>
<td>9.28</td>
</tr>
<tr>
<td>2560p800_4x6</td>
<td>6</td>
<td>4</td>
<td>147 M</td>
<td>720 ms</td>
<td>8.88</td>
</tr>
<tr>
<td>2560p800_2x6</td>
<td>6</td>
<td>2</td>
<td>147 M</td>
<td>360 ms</td>
<td>9.05</td>
</tr>
</tbody>
</table>
# RNN-T vs. Hybrid

<table>
<thead>
<tr>
<th></th>
<th>WER</th>
<th>Encoder lookahead</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM (Hybrid)</td>
<td>11.49</td>
<td>0</td>
<td>124 Mb AM + 5 Gb decoding graph</td>
</tr>
<tr>
<td>1280p640_x6</td>
<td>11.25</td>
<td>0</td>
<td>248 Mb</td>
</tr>
<tr>
<td>cltLSTM (Hybrid)</td>
<td>9.34</td>
<td>480</td>
<td>272 Mb + 5 Gb decoding graph</td>
</tr>
<tr>
<td>2560p800_2x6</td>
<td>9.05</td>
<td>360</td>
<td>588 Mb</td>
</tr>
</tbody>
</table>
Encoder Lookahead Doesn’t Translate to Overall Latency

- 1280p640_x6: 11*30ms = 330ms latency
- 2560p800_2x6: (1+12)*30ms = 390ms latency
- 2560p800_4x6: (-2+24)*30ms = 660ms latency

Frame duration is 30ms in the figure.
Personalization RNN-T

Rapid Speaker Adaptation - Challenges

• Massive number of model parameters

• Limited adaptation data (e.g. <=10 min)

• Imperfect supervision (unsupervised)
Our Proposed Approach

• **Approach**
  - Train speaker embedding with small amount of source speech
  - Use neural language generator to generate content relevant text
  - Synthesize content relevant personalized speech
  - Adapt with source speech and synthesized speech

• **Advantages**
  - Fundamentally alleviates data sparsity
  - Gracefully circumvents the obstacle of explicit labeling error
Framework Review

Adaptation Results

- Nice gain is obtained by leveraging TTS data.
- Almost 10% WERR for unsupervised adaptation with only 1 minute data.

<table>
<thead>
<tr>
<th>Model</th>
<th>1min</th>
<th>WER.R</th>
<th>10min</th>
<th>WER.R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (RNNT)</td>
<td>14.15</td>
<td>NA</td>
<td>14.15</td>
<td>NA</td>
</tr>
<tr>
<td>SUP</td>
<td>14.31</td>
<td>-1.14</td>
<td>13.34</td>
<td>5.71</td>
</tr>
<tr>
<td>SUP_{tar(200)}(w)</td>
<td>12.51</td>
<td>11.58</td>
<td>11.93</td>
<td>15.71</td>
</tr>
<tr>
<td>UNSUP</td>
<td>14.05</td>
<td>0.72</td>
<td>13.51</td>
<td>4.55</td>
</tr>
<tr>
<td>UNSUP_{tar(200)}(w)</td>
<td>13.03</td>
<td>7.95</td>
<td>13.02</td>
<td>8.00</td>
</tr>
<tr>
<td>UNSUP_{tar(200)}(w)(f,f,f)</td>
<td>12.78</td>
<td>9.68</td>
<td>12.59</td>
<td>11.02</td>
</tr>
</tbody>
</table>
Streaming Transformer Transducer for speech recognition on large-scale dataset

For offline SR, Transformer model shows better accuracy than LSTM, and Conformer further improves its results.

For online SR, streaming transducer is more robust and shows better accuracy on large-scale dataset.

Transformer Recap

Step 0. Given input $X$

Following operations are conducted on multi-head in parallel, we take the $i$-th head as an example:

Step 1.1 Linear Transformation:

$$Q_i = W_q X_i, K_i = W_k X_i, V_i = W_v X_i$$

Step 1.2. Compute Attention weight:

$$a = \text{softmax}(\frac{Q_i^T K_i}{d_{model}})$$

Step 1.3. Linear combination values:

$$\text{Hidden} = aV_i$$

Step 2. Residual Connection and layer normalization.

Step 3. Feed-forward network

Step 4. Residual Connection and layer normalization.

Vaswani et al. “Attention is all you need” NIPS 2017
Architecture

Zhang et al., “Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss,” in Proc. ICASSP 2020
Challenges

1) **Memory and runtime cost** increase linearly with respect to the history length.

Zhang et al., “Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss,” in Proc. ICASSP 2020

2) **Look-ahead window** grows linearly with number of layers for small lookahead scenario.
1) Chunk-wise method: Modeling chunks independently.

Pros: Efficient training and inference.
Cons: Performance drop significantly due to loss of cross chunk information

2) Memory-based method: A memory to encode history information recurrently.

Pros: History information is well modeled.
Cons: Recurrent structure decreases training speed
Our Solution

• Compute attention weight \( \{\alpha_{t,\tau}\} \) for time \( t \) over input sequence \( \{x_{\tau}\} \), binary attention mask \( \{m_{t,\tau}\} \) to control range of input \( \{x_{\tau}\} \) to use

\[
\alpha_{t,\tau} = \frac{m_{t,\tau} \exp(\beta (W_q x_t)^T(W_k x_{\tau}'))}{\sum_{\tau'} m_{t,\tau'} \exp(\beta (W_q x_t)^T(W_k x_{\tau'}))} = \text{softmax}(\beta q_t^T k_{\tau}, m_{t,\tau})
\]

• Apply attention weight over value vector \( \{v_{\tau}\} \)

\[
z_t = \sum_{\tau} \alpha_{t,\tau} W_v x_{\tau} = \sum_{\tau} \alpha_{t,\tau} v_{\tau}
\]

• Attention Masking is all you need to design for different scenarios
Attention Mask is All You Need

- Offline (whole utterance)

Predicting output for $x_{10}$

Not streamable

Attention Mask
Attention Mask is All You Need

• 0 lookahead, full history

Memory and runtime cost increase linearly
Attention Mask is All You Need

• 0 lookahead, limited history (3 frames)

In some scenario, small amount of latency is allowed
Attention Mask is All You Need

- Small lookahead (at most 2 frames), limited history (3 frames)

**Look-ahead window [0, 2]**

Predicting output for $x_{10}$
Our Method: Masking is all you need

1) Each frame can see fixed numbers of the left chunk, and the left reception field will propagate.

2) In a chunk, all frames can see each other.

3) Future chunk cannot be seen to ensure parallel training of Transformer

- Left reception field = encoder_layers * left_chunk_can_be_seen * chunk_size
- Right reception field = chunk_size/2
Implementation

• Efficient transducer decoder implementation with C++, on CPU
  • Beam search based on prefix tree expansion
  • Caching Query and Key in previous frames, avoid repeated computation

• Model trained with Pytorch (GPU) and exported with Libtorch (CPU)
  • FP16 is applied to speed up training
  • Relative position embedding for performance improvement
Experiment Setup

Training Data: 65k hours Microsoft Internal dataset

Test Data: Audios cover multiple domains, consisting of 1.8M words

Model Size: ~80M parameters

Training Speed: converge in 2 days with 32 V100 GPU
WER and RTF results for zero lookahead

<table>
<thead>
<tr>
<th></th>
<th>#hist frames</th>
<th>WER (%)</th>
<th>RTF (#thread)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN-T</td>
<td>+∞</td>
<td>9.86</td>
<td>1.56</td>
</tr>
<tr>
<td>T-T</td>
<td>+∞</td>
<td>8.79</td>
<td>3.44</td>
</tr>
<tr>
<td>T-T</td>
<td>60</td>
<td>8.88</td>
<td>2.38</td>
</tr>
<tr>
<td>C-T</td>
<td>+∞</td>
<td>8.78</td>
<td>4.02</td>
</tr>
<tr>
<td>C-T</td>
<td>60</td>
<td>8.80</td>
<td>2.41</td>
</tr>
</tbody>
</table>

- T-T and C-T present consistent WER improvement over RNN-T
- 0.1% WER degradation with 60 hist frames, compared to full history
- RTFs for T-T and C-T is 2-4 times higher than RNN-T
  - slow to compute frame by frame for Transformer
WER and RTF results for batching

• By introducing several frame latency, significant speedup could be achieved by grouping multiple frames as a minibatch for forward

• The speedup from T-T and C-T is higher than LSTM
  • Due to the model differences in LSTM and Transformer

• RTF as low as 0.2 could be achieved with 15 frames latency (i.e. 450ms latency)

<table>
<thead>
<tr>
<th></th>
<th>#hist len</th>
<th>WER (%)</th>
<th>RTF (#batch size)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>RNN-T</td>
<td>+∞</td>
<td>9.86</td>
<td>0.46</td>
</tr>
<tr>
<td>T-T</td>
<td>60</td>
<td>8.88</td>
<td>1.75</td>
</tr>
<tr>
<td>C-T</td>
<td>60</td>
<td>8.80</td>
<td>1.83</td>
</tr>
</tbody>
</table>
WER and RTF results with lookahead

<table>
<thead>
<tr>
<th>Model</th>
<th>#hist frame</th>
<th>#lookahead (ms)</th>
<th>WER (%)</th>
<th>RTF (#thread)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid</td>
<td>$+\infty$</td>
<td>480</td>
<td>9.34</td>
<td>-</td>
</tr>
<tr>
<td>RNN S2S</td>
<td>$+\infty$</td>
<td>720</td>
<td>9.61</td>
<td>-</td>
</tr>
<tr>
<td>Trans. S2S</td>
<td>$+\infty$</td>
<td>[480, 960]</td>
<td>9.16</td>
<td>-</td>
</tr>
<tr>
<td>Trans. S2S</td>
<td>$+\infty$</td>
<td>$+\infty$</td>
<td>7.82</td>
<td>-</td>
</tr>
<tr>
<td>RNN-T</td>
<td>$+\infty$</td>
<td>360</td>
<td>9.11</td>
<td>1.52</td>
</tr>
<tr>
<td>T-T</td>
<td>60</td>
<td>[0, 720]</td>
<td>8.28</td>
<td>0.40</td>
</tr>
<tr>
<td>C-T</td>
<td>60</td>
<td>[0, 720]</td>
<td>8.19</td>
<td>0.45</td>
</tr>
<tr>
<td>T-T</td>
<td>$+\infty$</td>
<td>$+\infty$</td>
<td>7.78</td>
<td>0.39</td>
</tr>
<tr>
<td>C-T</td>
<td>$+\infty$</td>
<td>$+\infty$</td>
<td>7.69</td>
<td>0.36</td>
</tr>
</tbody>
</table>

- T-T and C-T trained with lookahead gives consistent improvement
- Beat other S2S models with similar latency
8-bit quantization

- Significant speedup achieved for RNN-T
- The speedup/performance for T-T and C-T is not ideal

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>WER (%)</th>
<th>RTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN-T</td>
<td>float32</td>
<td>9.11</td>
<td>1.56</td>
</tr>
<tr>
<td></td>
<td>int8</td>
<td>9.13</td>
<td>0.43</td>
</tr>
<tr>
<td>T-T</td>
<td>float32</td>
<td>8.28</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td>int8</td>
<td>8.50</td>
<td>0.22</td>
</tr>
<tr>
<td>C-T</td>
<td>float32</td>
<td>8.19</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>int8</td>
<td>8.40</td>
<td>0.26</td>
</tr>
</tbody>
</table>
Streaming End-to-End Multi-talker Speech Recognition

Far-field conversational speech recognition is becoming more important

- Unsegmented continuous recordings
- Speech with 15~25% speech overlap
- Different recording conditions & setup
Background of Multi-talker Speech Recognition

• **Cascaded approach: Speech Separation + ASR**
  

• **Hybrid joint training approach**
  
  

• **(Offline) End-to-end approach**
  
  
  X. Chang, et al., “End-to-end monaural multi-speaker ASR system without pretraining”, ICASSP 2019
  
  N. Kanda, et al., “Serialized output training for end-to-end overlapped speech recognition”, Interspeech 2020
  
Streaming Unmixing and Recognition Transducer (SURT)

- **Streaming** end-to-end multi-taker ASR
  - Employs RNN-T as the backbone
  - Two different model structures
    - Speaker-differentiator based network
    - Mask-based network

- Two different loss functions
  - Permutation Invariant Training
  - Heuristic Error Assignment Training
Model Structure

1. Speaker-differentiator based network
Model Structure

2. Mask-based network
Model Training

• Loss functions
  • Permutation Invariant Training: consider all the possible permutations:

\[ \mathcal{L}_{\text{pit}}(X, Y^1, Y^2) = \min(\mathcal{L}_{\text{rnnt}}(Y^1, H_1) + \mathcal{L}_{\text{rnnt}}(Y^2, H_2), \]
\[ \mathcal{L}_{\text{rnnt}}(Y^2, H_1) + \mathcal{L}_{\text{rnnt}}(Y^1, H_2)) \]

• Drawbacks: computationally expensive and not scalable
• For S-speaker case, PIT needs to compute the RNN-T loss S! times
Model Training

• Heuristic Error Assignment Training (HEAT)
  • Considers only one possible error assignment

\[ \mathcal{L}_{\text{heat}}(X, Y^1, Y^2) = \mathcal{L}_{\text{rnnt}}(Y^1, H_1) + \mathcal{L}_{\text{rnnt}}(Y^2, H_2) \]

• Based on the timing information to fix the error assignment

• Computationally more scalable
• Similar approach has been studied in:
Why it works?
Why it works?

with label permutation

\[ \tau \]

\[ \nu \]
Why it works?

without label permutation

\[ \tau \]

\[ \nu \]
Experiments

• LibrispeechMix: simulated overlapped speech dataset derived from Librispeech
• In our experiments, we only consider 2-speaker case
• Investigating two conditions $\tau=0$ and $\tau=0.5$
• Overlapped data sampled form $[\tau, \nu]$
SD-based model

<table>
<thead>
<tr>
<th>Module</th>
<th>Type</th>
<th>Depth</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixure</td>
<td>Conv2D</td>
<td>4</td>
<td>[conv2d(3, 64, 3, 3), conv2d(64, 64, 3, 3), Maxpool(3, 1), conv2d(64, 128, 3, 3), Maxpool(3, 1), conv2d(128, 128, 3, 3), Maxpool(3, 1)</td>
</tr>
<tr>
<td>SD1</td>
<td>LSTM</td>
<td>2</td>
<td>(1024, 1024)</td>
</tr>
<tr>
<td>SD1</td>
<td>LSTM</td>
<td>2</td>
<td>(1024, 1024)</td>
</tr>
<tr>
<td>RNNT-A</td>
<td>LSTM</td>
<td>2</td>
<td>(1024, 1024)</td>
</tr>
<tr>
<td>RNNT-L</td>
<td>LSTM</td>
<td>2</td>
<td>(1024, 1024)</td>
</tr>
</tbody>
</table>
# Mask-based model

<table>
<thead>
<tr>
<th>Module</th>
<th>Type</th>
<th>Depth</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask</td>
<td>Conv2D</td>
<td>4</td>
<td>conv2d(3, 64, 3, 3)</td>
</tr>
<tr>
<td>RNNT-A</td>
<td>LSTM</td>
<td>6</td>
<td>(771, 1024)</td>
</tr>
<tr>
<td>RNNT-L</td>
<td>LSTM</td>
<td>2</td>
<td>(1024, 1024)</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Train</th>
<th>Model</th>
<th>Loss</th>
<th>$\tau = 0$</th>
<th>$\tau = 0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>dev</td>
<td>test</td>
</tr>
<tr>
<td>$\tau = 0.5$</td>
<td>SD</td>
<td>PIT</td>
<td>12.0</td>
<td>12.1</td>
</tr>
<tr>
<td></td>
<td>HEAT</td>
<td>11.8</td>
<td>11.7</td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td>PIT</td>
<td>14.1</td>
<td>14.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HEAT</td>
<td>13.4</td>
<td>13.1</td>
<td></td>
</tr>
<tr>
<td>$\tau = 0$</td>
<td>SD</td>
<td>PIT</td>
<td>13.1</td>
<td>13.2</td>
</tr>
<tr>
<td></td>
<td>HEAT</td>
<td>12.5</td>
<td>12.5</td>
<td></td>
</tr>
</tbody>
</table>
Results

1. SURT: SD-based network, trained with HEAT

2. PIT-S2S: LSTM-based S2S model, trained with PIT

<table>
<thead>
<tr>
<th>Train</th>
<th>Model</th>
<th>Size</th>
<th>Latency</th>
<th>$\tau = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dev</td>
</tr>
<tr>
<td>$\tau = 0.5$</td>
<td>SURT</td>
<td>80M</td>
<td>120 ms</td>
<td>11.8</td>
</tr>
<tr>
<td></td>
<td>PIT-S2S [18]</td>
<td>160.7M</td>
<td>$\infty$</td>
<td>–</td>
</tr>
</tbody>
</table>
Conclusions

• We reported our recent development of RNN-T models
  • The CE initialization of RNN-T encoder significantly reduced WER by 11.6% relatively
  • The model with future context improved from the zero-lookahead model by 12.8% relatively
  • Surpasses the best hybrid model by 3.1% relative WER reduction and 120 ms less encoder lookahead latency
Conclusions

• Personalization RNN-T
  • Synthesizing TTS audio on top of scripts generated from the neural language model gracefully circumvents the obstacle of explicit labeling error
  • 10% WERR is obtained with unsupervised adaptation of only 1 minute speech.
Conclusions

• **Masking is all you need** – enables high accuracy (much better than RNN-T), low cost and low latency streaming Transformer Transducer.
Conclusions

• Streaming Unmixing and Recognition Transducer (SURT) provides a streaming solution to multi-talker speech recognition.
  • Obtained strong recognition accuracy with very low latency and a much smaller model compared with an offline PIT-S2S model.

\[ \mathcal{L}_{\text{heat}}(X, Y^1, Y^2) = \mathcal{L}_{\text{rnnt}}(Y^1, H_1) + \mathcal{L}_{\text{rnnt}}(Y^2, H_2) \]
References

Thank You!