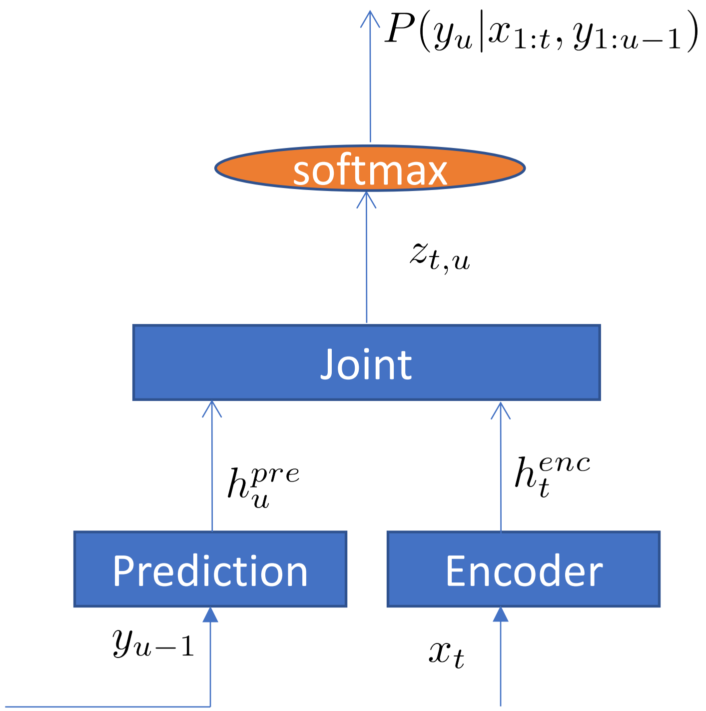
Developing streaming end-to-end models for automatic speech recognition in industry

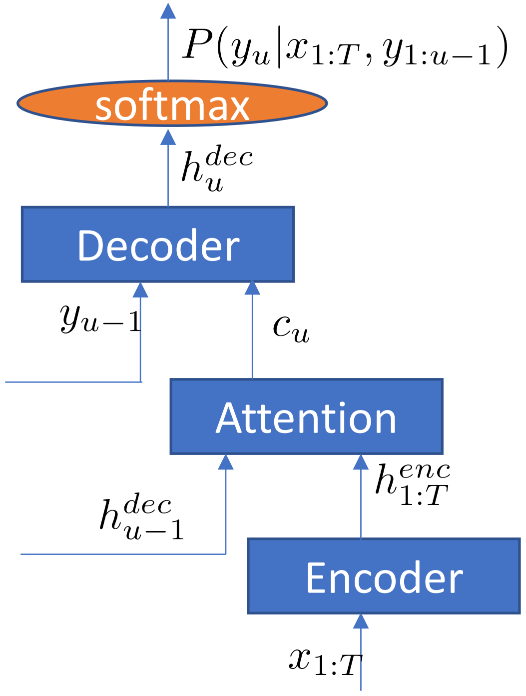
Jinyu Li

Microsoft

E2E Models



Transducer

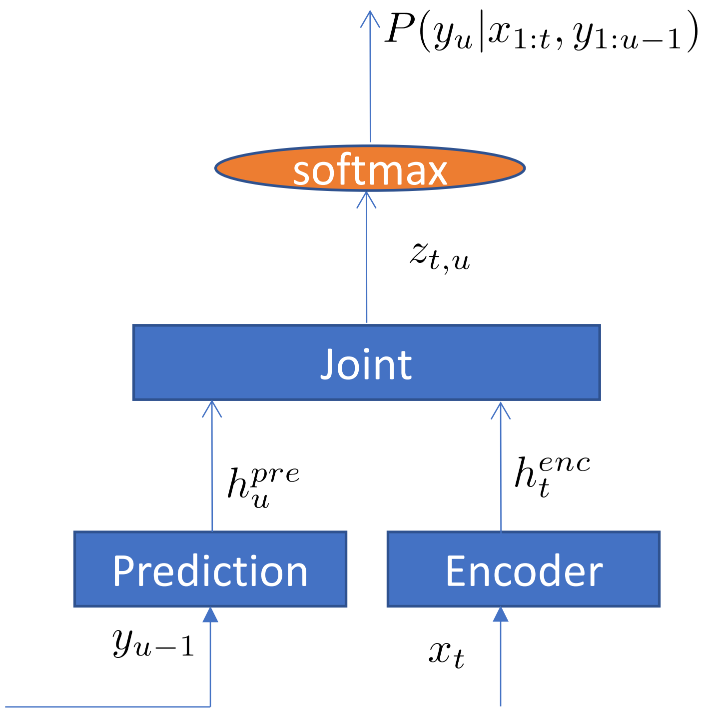


Sequence to sequence (S2S)

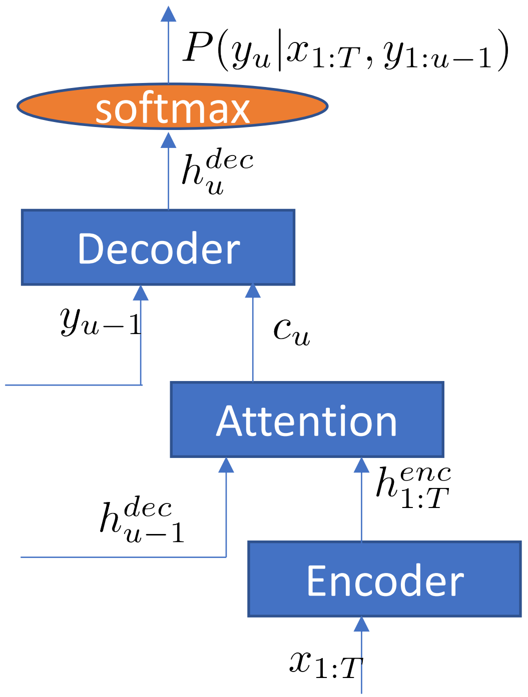
E2E Models

	Transducer	S2S
Attention mechanism	No	Yes
Building block	RNN or Transformer	RNN or Transformer
Streaming	Natural	Need to covert full attention to partial attention
Ideal operation scenario	streaming	offline

E2E Models



RNN-Transducer (RNN-T)



Sequence to sequence (S2S)

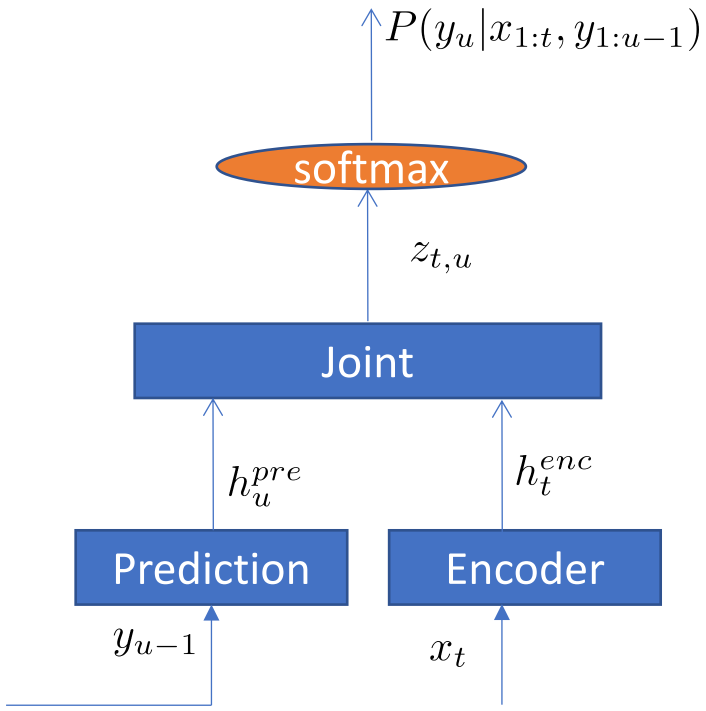
High Performance RNN-T Model

Jinyu Li et al., "Developing RNN-T Models Surpassing High-Performance Hybrid Models with Customization Capability," in Proc. Interspeech, 2020.

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

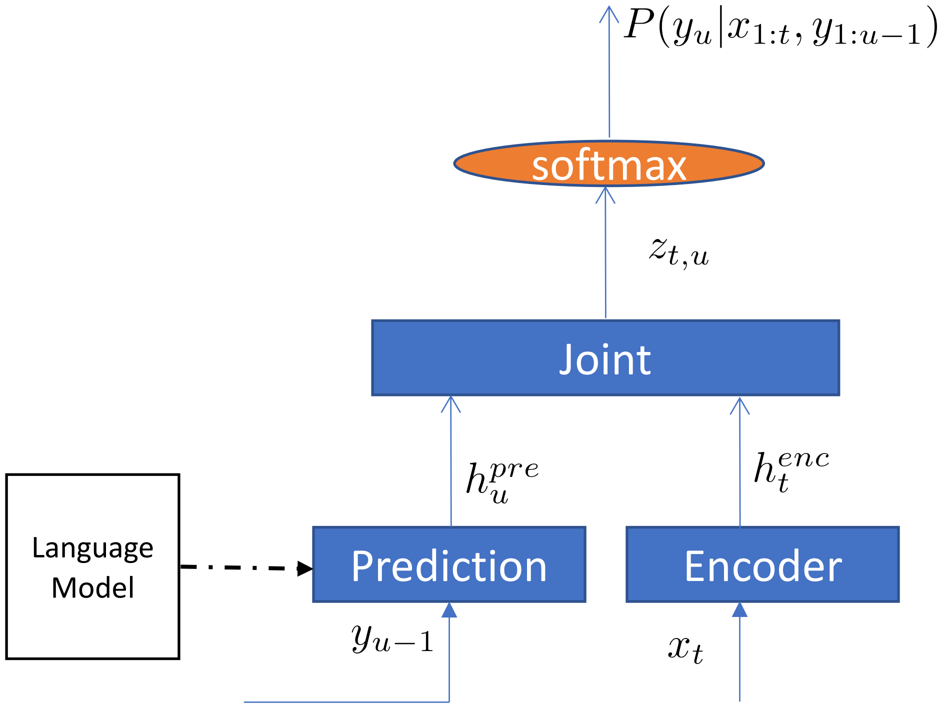
After the joint network, there are 3 functions to get loss:

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

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

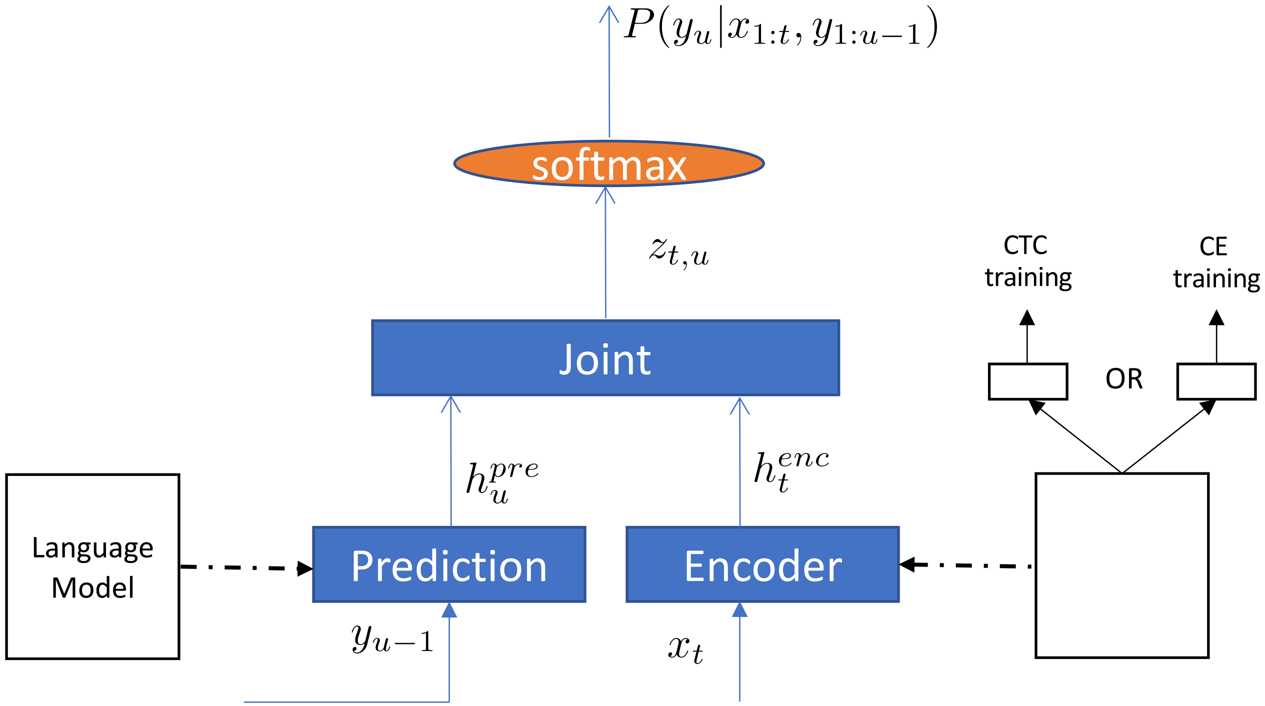
Jinyu Li, Rui Zhao, Hu Hu, Yifan Gong, "Improving RNN Transducer Modeling for End-to-End Speech Recognition," in Proc. ASRU, 2019.

Initialization



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

Initialization

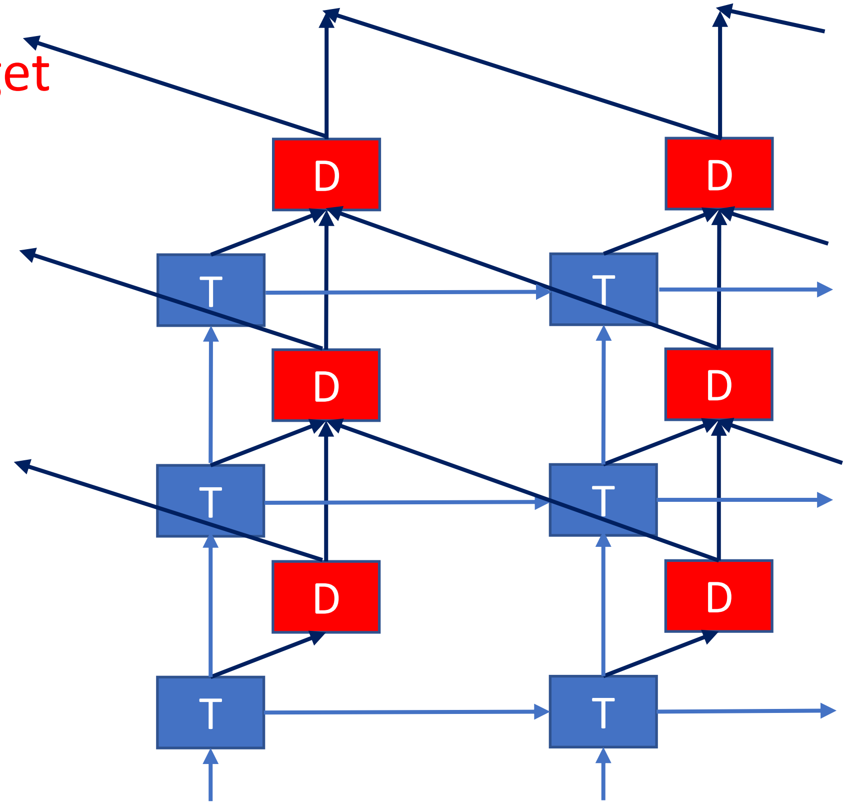


- Initializing the prediction network with a pre-trained LM is not effective.

- 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



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

$$h_t^l = LSTM(h_{t-1}^l, h_t^{l-1})$$

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

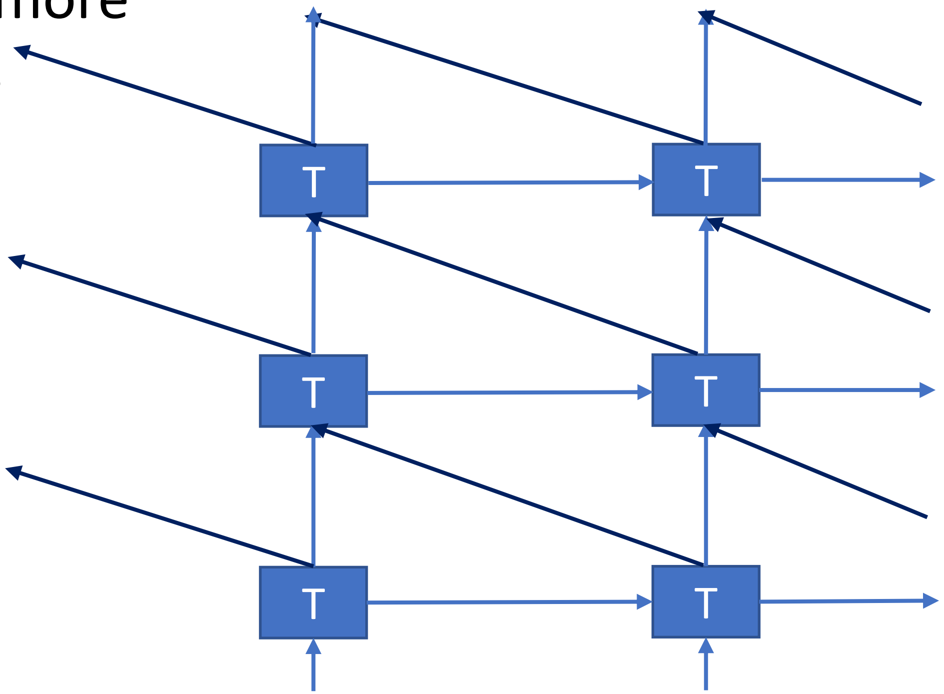
J. Li, et al., "Improving Layer Trajectory LSTM With Future Context Frames", in Proc. ICASSP 2019.

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)

Microsoft

High-Performance Hybrid Models

Hybrid	CE WER%	MMI WER%	T/S WER%	Parameter number	Encoder lookahead
LSTM	14.75	13.01	11.49	30 M	0
cltLSTM	11.15	10.36	9.34	63 M	480 ms

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

J. Li, et al., "High-Accuracy and Low-Latency Speech Recognition with Two-Head Contextual Layer Trajectory LSTM Model," in Proc. ICASSP, 2020.

RNN-T Models

We use MpN_FxL to denote the encoder structure and use 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

Models	Random	CTC	CE
1600p800_4x6	10.55	10.40	9.33

- 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

Encoder network	Layers	Lookahead Frames /layer	Cell size	Projection size	WER
1280p640_x6	6	0	1280	640	11.25

Comparison of RNN-T Models

Encoder network	**Layers**	**Lookahead Frames /layer**	**Cell size**	**Projection size**	WER
1280p640_x6	6	0	1280	640	11.25
1280p640_4x6	6	4	1280	640	9.81

Comparison of RNN-T Models

Encoder network	Layers	Lookahead Frames /layer	Cell size	Projection size	WER
1280p640_x6	6	0	1280	640	11.25
1280p640_4x6	6	4	1280	640	9.81
1600p800_4x6	6	4	1600	800	9.33
2048p640_4x6	6	4	2048	640	9.27
2048p640_4x8	8	4	2048	640	9.28
2560p800_4x6	6	4	2560	800	8.88
2560p800_2x6	6	2	2560	800	9.05

Comparison of RNN-T Models

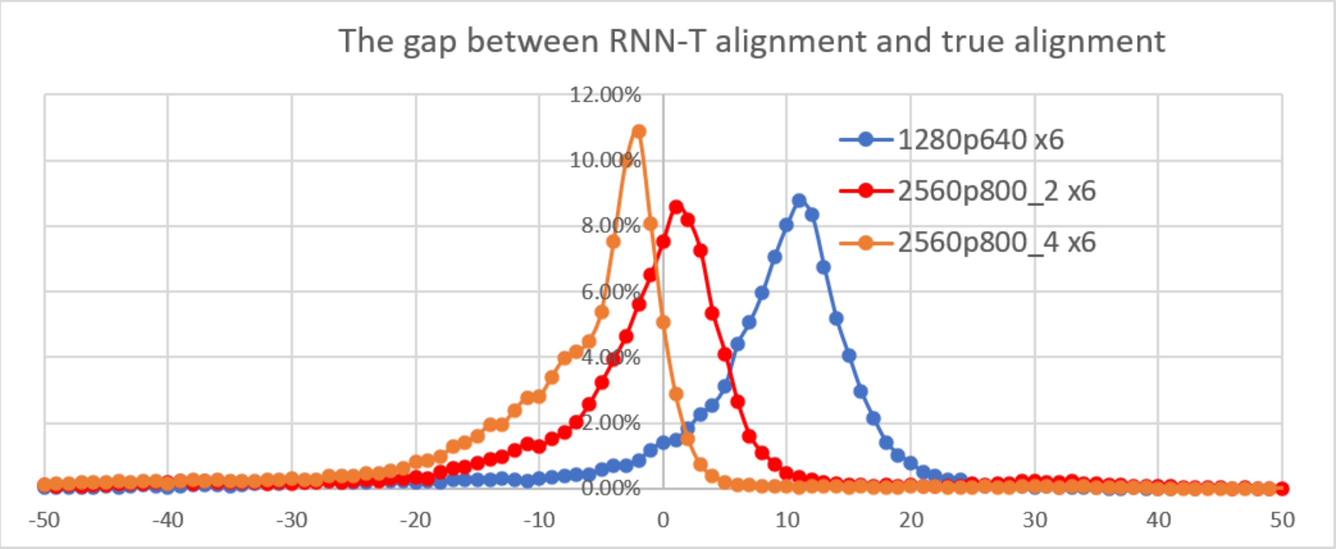
Encoder network	Layers	Lookahead Frames /layer	Parameter number	Encoder lookahead	WER
1280p640_x6	6	0	62 M	0	11.25
1280p640_4x6	6	4	62 M	720 ms	9.81
1600p800_4x6	6	4	94 M	720 ms	9.33
2048p640_4x6	6	4	87 M	720 ms	9.27
2048p640_4x8	8	4	119 M	960 ms	9.28
2560p800_4x6	6	4	147 M	720 ms	8.88
2560p800_2x6	6	2	147 M	360 ms	9.05

RNN-T vs. Hybrid

	WER	Encoder lookahead	Size
LSTM (Hybrid)	11.49	0	124 Mb AM + 5 Gb decoding graph
1280p640_x6	11.25	0	248 Mb
cltLSTM (Hybrid)	9.34	480	272 Mb + 5 Gb decoding graph
2560p800_2x6	9.05	360	588 Mb

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

Y. Huang, et al., “Rapid RNN-T Adaptation Using Personalized Speech Synthesis and Neural Language Generator," in Proc. Interspeech 2020.

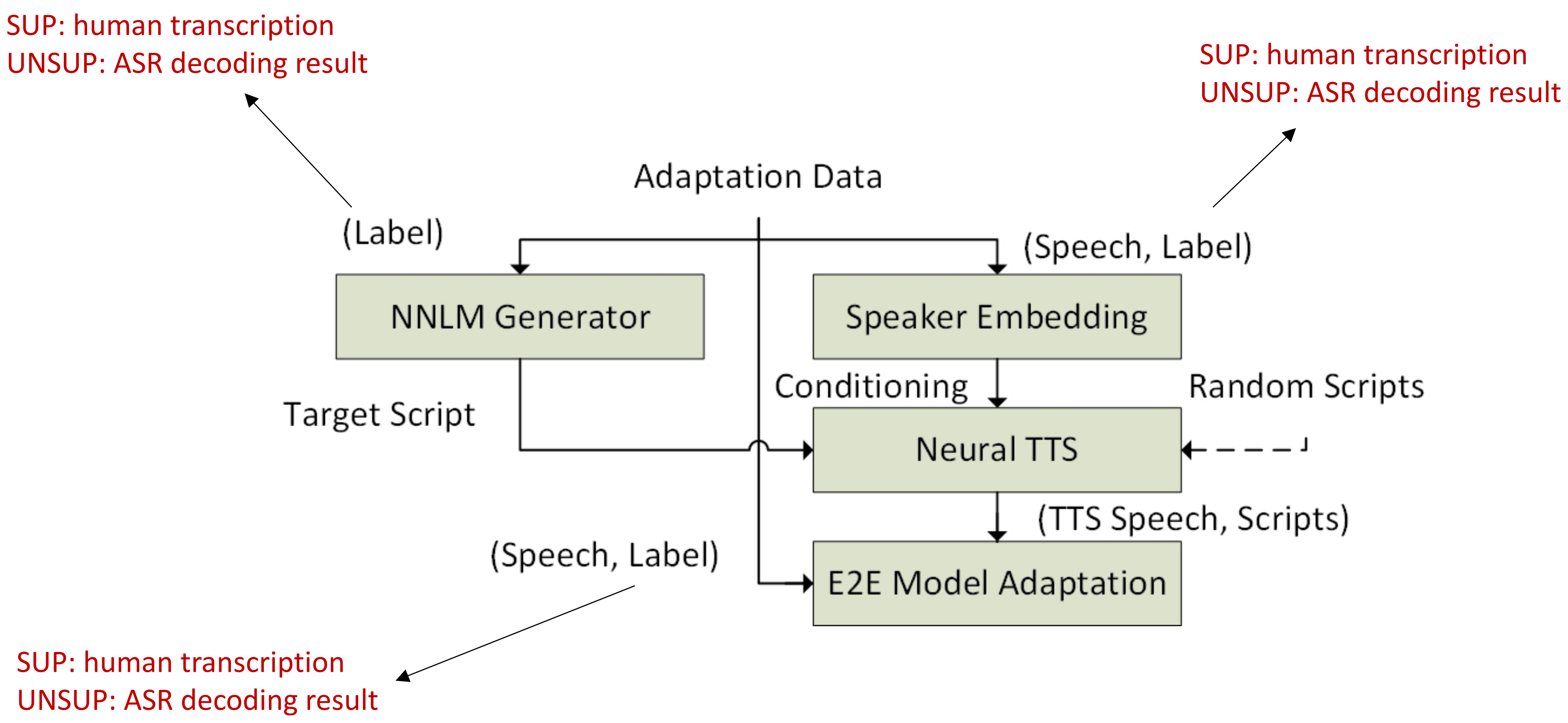
Rapid Speaker Adaptation - Challenges

- 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



Y. Huang, et al., "Using personalized speech synthesis and neural language generator for rapid speaker adaptation," in Proc. ICASSP 2020.

Microsoft

Adaptation Results

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

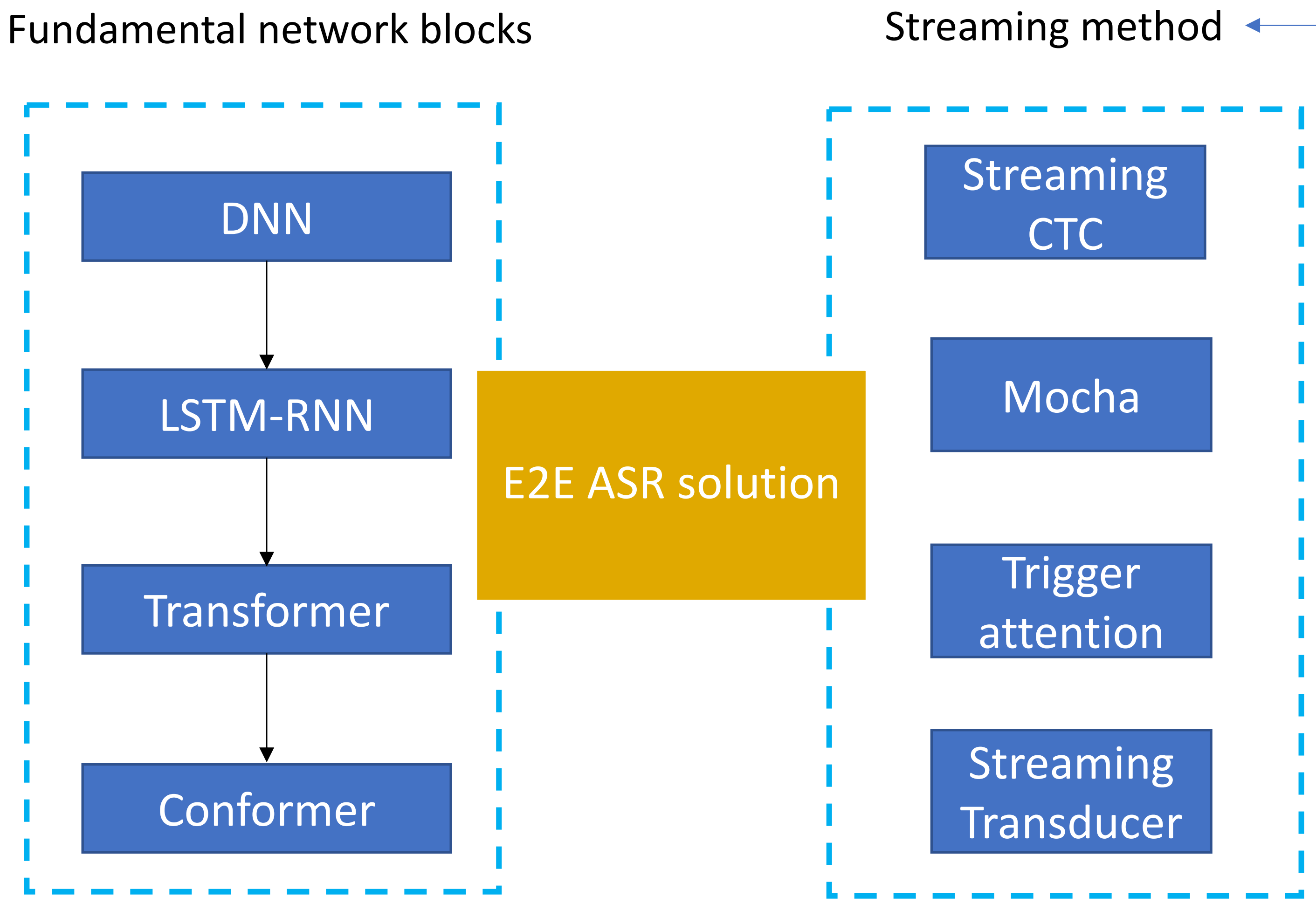
Model	1min	**WER.R**	10min	**WER.R**
Baseline (RNNT)	14.15	**NA**	14.15	**NA**
SUP	14.31	-1.14	13.34	5.71
$SUP_{+tar(200)}(w)$	12.51	**11.58**	11.93	**15.71**
UNSUP	14.05	0.72	13.51	4.55
$UNSUP_{+tar(200)}(w)$	13.03	**7.95**	13.02	**8.00**
$UNSUP_{+tar(200)}(w)(f,f,f)$	12.78	**9.68**	12.59	**11.02**

Streaming Transformer Transducer for speech recognition on large-scale dataset

X. Chen, et al., "Developing Real-time Streaming Transformer Transducer for Speech Recognition on Large-scale Dataset." *arXiv preprint arXiv:2010.11395* (2020).

Background

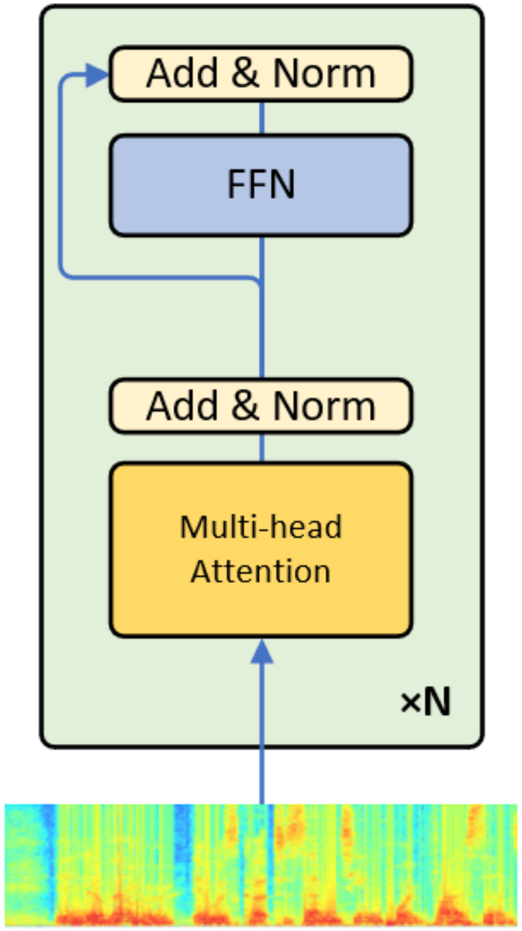
For **offline SR**, **Transformer** model shows better accuracy than LSTM, and **Conformer** further improves its results.



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

J. Li et al., "On the Comparison of Popular End-to-End Models for Large Scale Speech Recognition," in Proc. Interspeech, 2020.

Transformer Recap



Input

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:

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

Step 1.3. Linear combination values:

$$Hidden = \alpha V_i$$

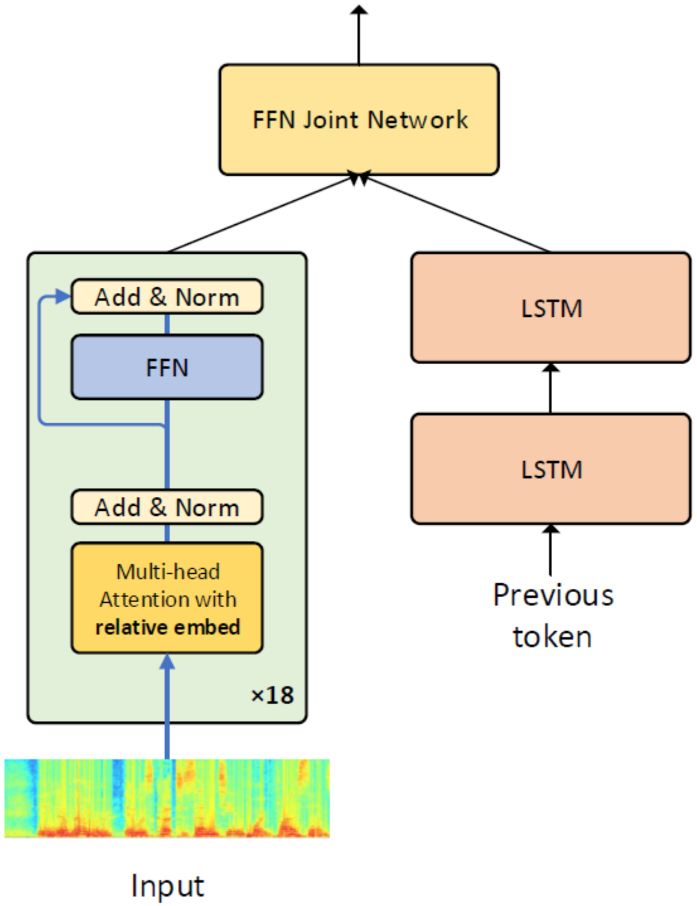
Step 2. Residual Connection and layer normalization.

Step 3. Feed-forward network

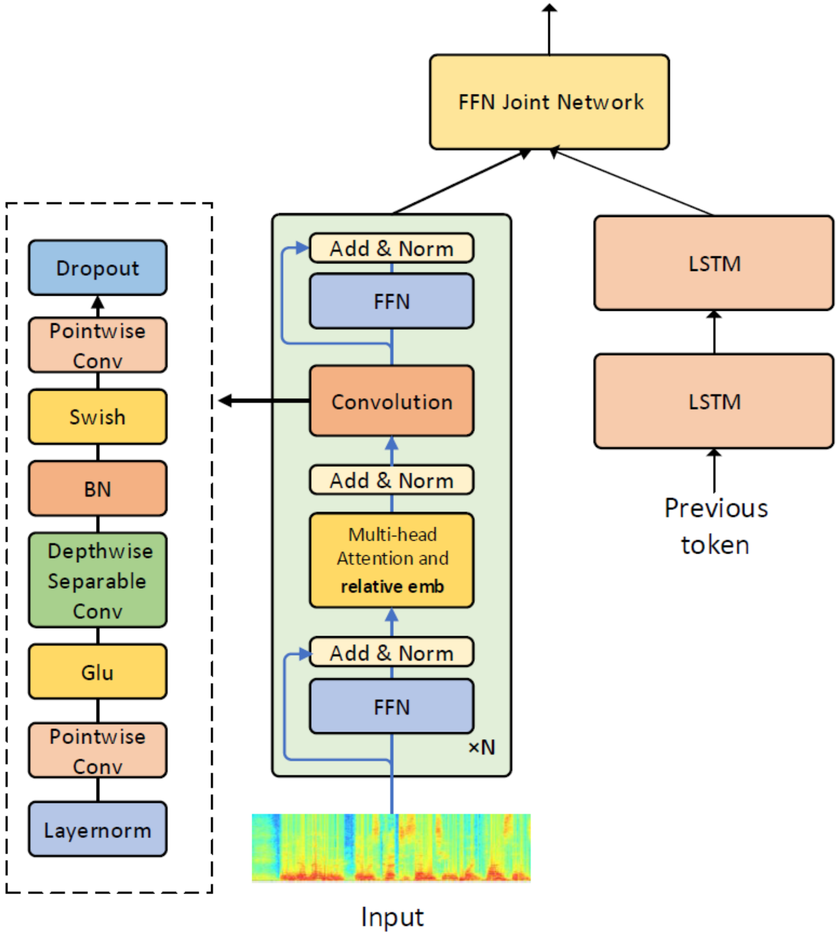
Step 4. Residual Connection and layer normalization.

Vaswani et al. "Attention is all you need" NIPS 2017

Architecture



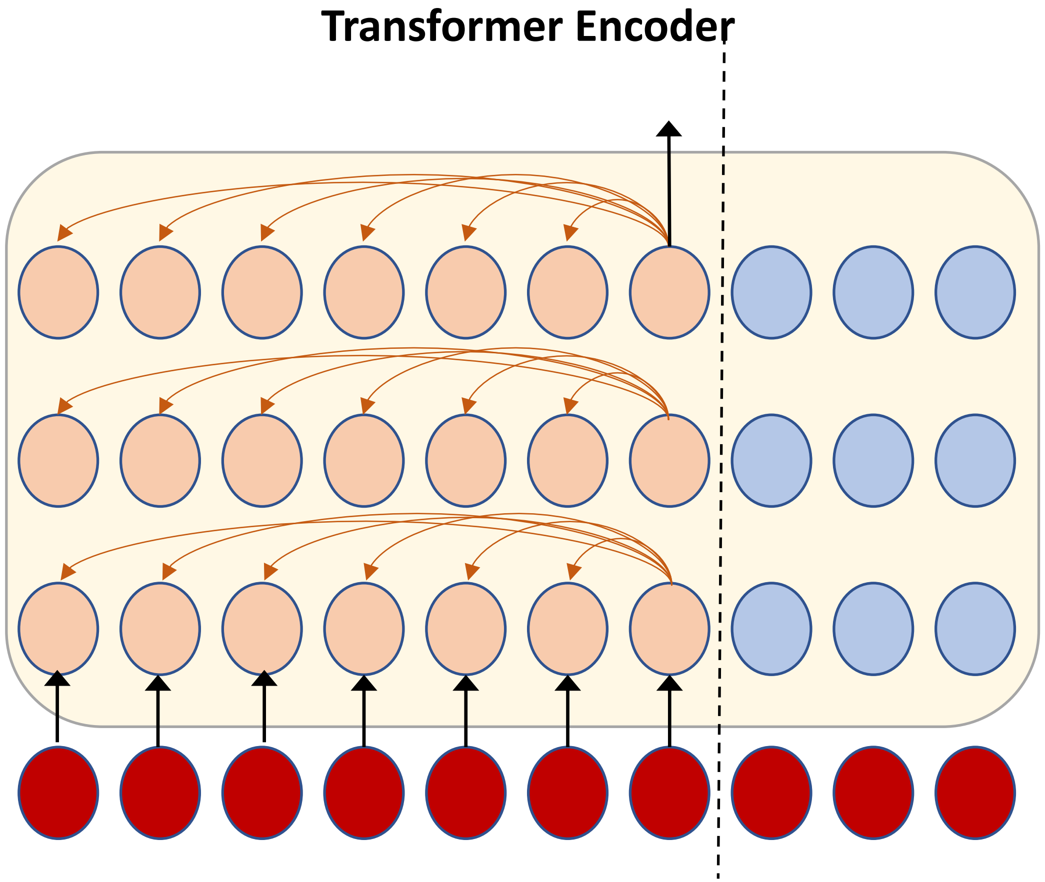
Transformer Transducer



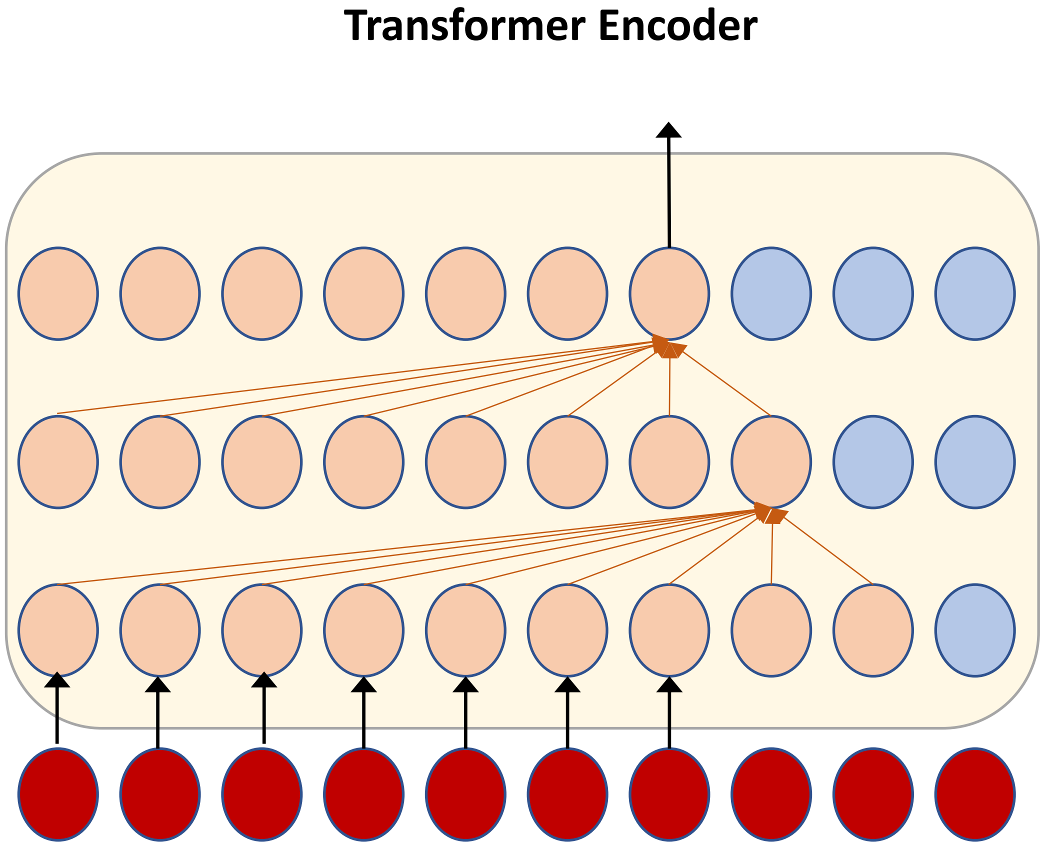
Conformer Transducer

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

Challenges



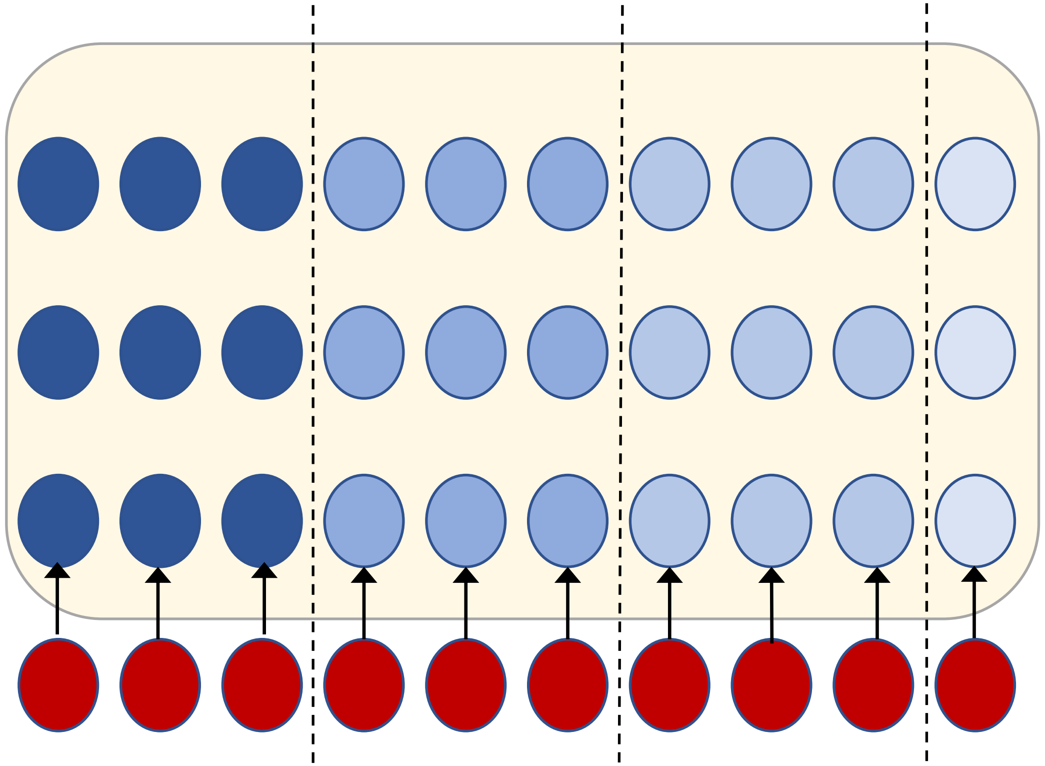
1) **Memory and runtime cost** increase linearly with respect to the history length.



2) **Look-ahead window** grows linearly with number of layers for small lookahead scenario.

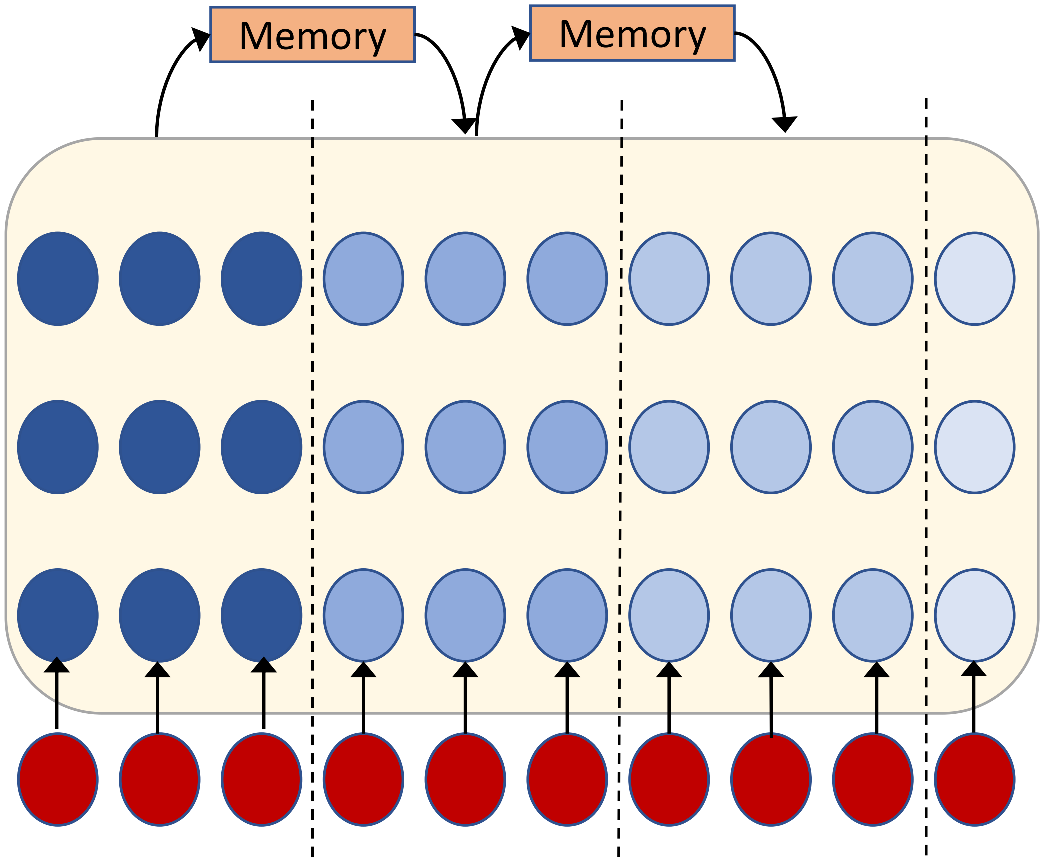
Zhang et al., "Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss," in Proc. ICASSP 2020

Existing Solution



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 $\{\boldsymbol{x}_\tau\}$, binary attention mask $\{m_{t,\tau}\}$ to control range of input $\{\mathbf{x}_\tau\}$ to use

$$\alpha_{t,\tau} = \frac{\mathrm{m}_{t,\tau}\exp(\beta\left(W_q\boldsymbol{x}_t\right)^T(W_k\boldsymbol{x}_\tau))}{\sum_{\tau'}\mathrm{m}_{t,\tau'}\exp(\beta\left(W_q\boldsymbol{x}_t\right)^T(W_k x_{\tau'}))} = softmax(\beta\boldsymbol{q}_t^T\boldsymbol{k}_\tau, m_{t,\tau})$$

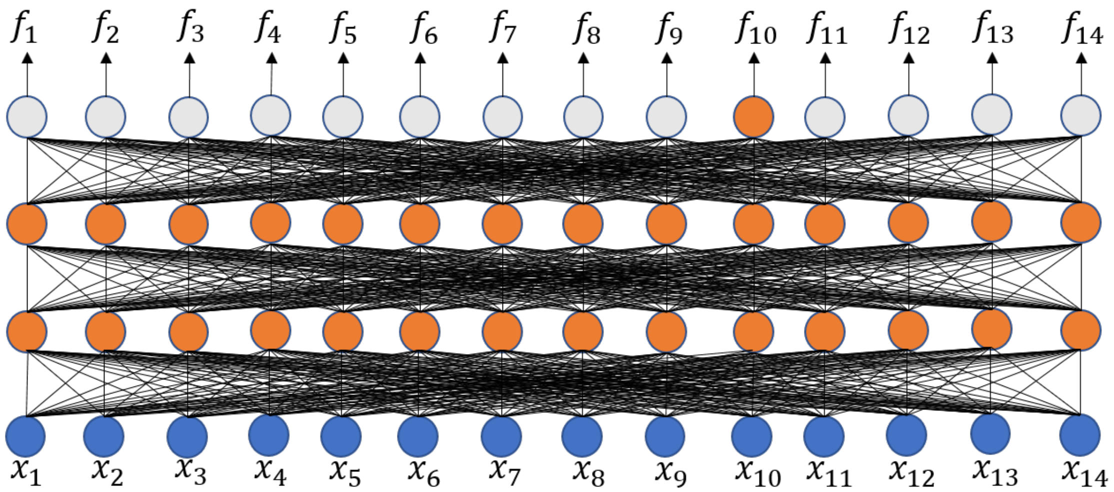
- Apply attention weight over value vector $\{\boldsymbol{v}_\tau\}$

$$z_t = \sum_\tau \alpha_{t,\tau} W_v \boldsymbol{x}_\tau = \sum_\tau \alpha_{t,\tau}\boldsymbol{v}_\tau$$

- 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

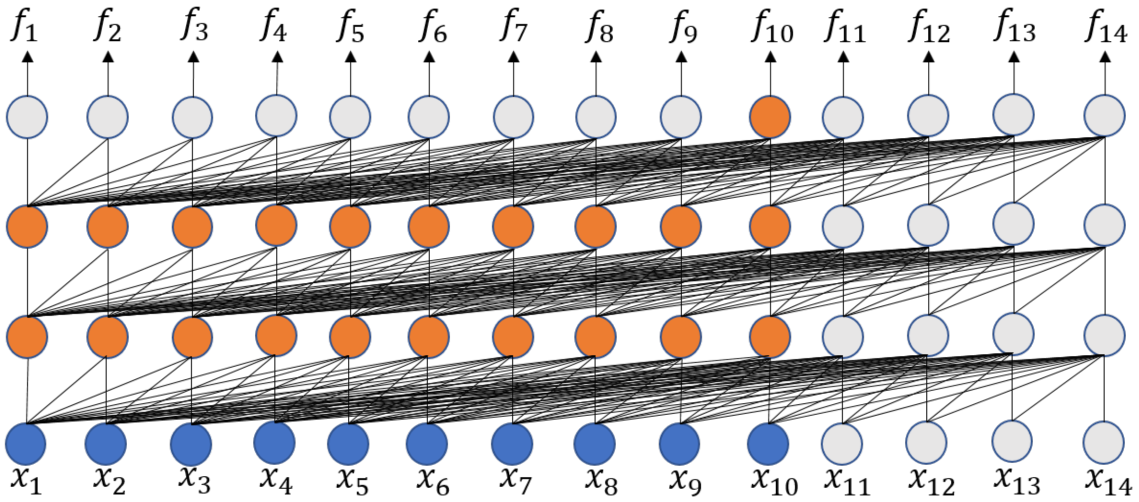
Frame Index

$$
\begin{array}{c}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14
\end{array}
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}.
$$

Attention Mask

Attention Mask is All You Need

- 0 lookahead, full history



Predicting output for x_{10}

Frame Index

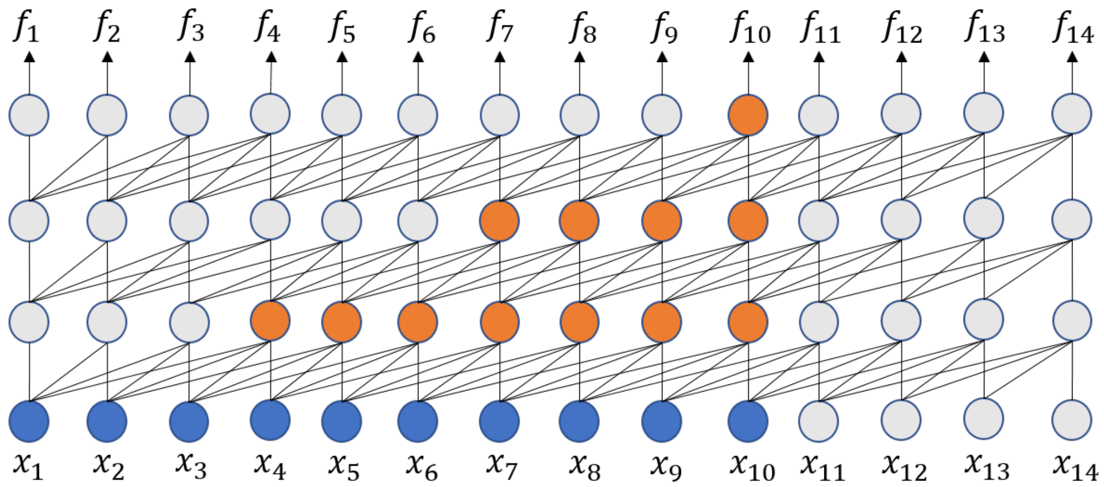
$$
\begin{array}{c}
1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14
\end{array}
\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}.
$$

Attention Mask

Memory and runtime cost increase linearly

Attention Mask is All You Need

- 0 lookahead, limited history (3 frames)



Predicting output for x_{10}

Frame Index

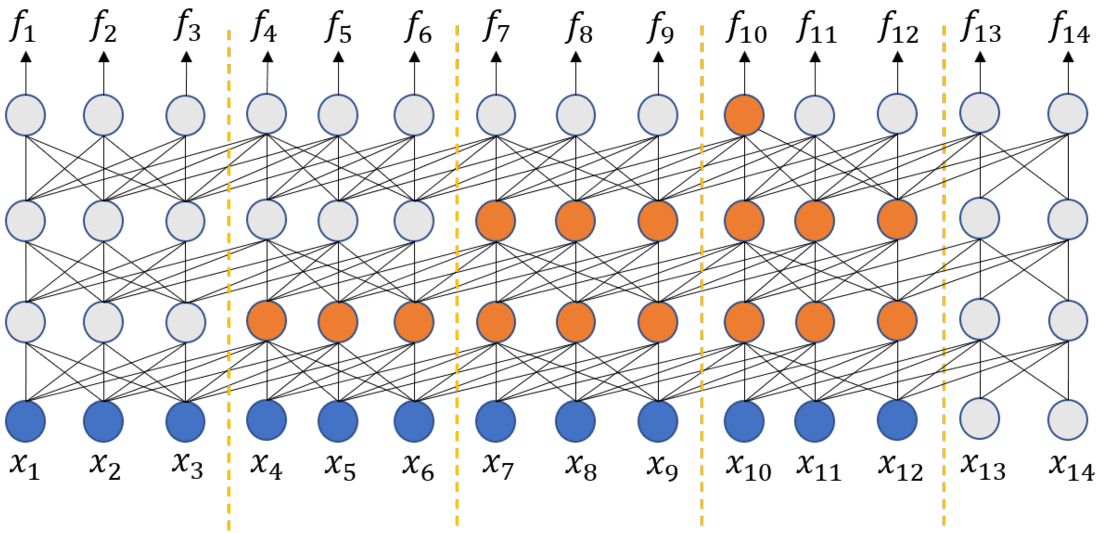
$$\begin{array}{r} 1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7\\ 8\\ 9\\ 10\\ 11\\ 12\\ 13\\ 14 \end{array}
\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0\\
0&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0\\
0&0&0&0&0&1&1&1&1&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&1
\end{bmatrix}.$$

Attention Mask

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)



Predicting output for x_{10}

Look-ahead window [0, 2]

Frame Index

$$
\begin{array}{c}
1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14
\end{array}
\begin{bmatrix}
1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&1&1&1&0&0\\
0&0&0&0&0&0&1&1&1&1&1&1&0&0\\
0&0&0&0&0&0&1&1&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&1&1&1&1&1
\end{bmatrix}.
$$

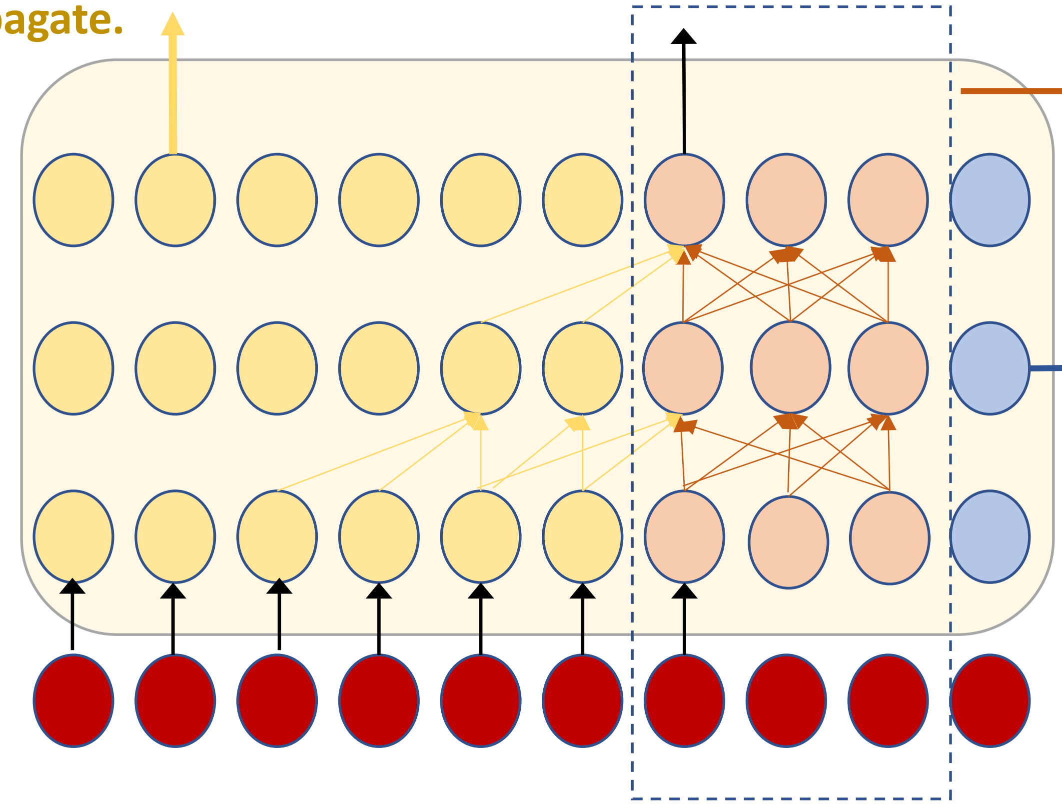
Attention Mask

Microsoft

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.**

$chunk_i$



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

Microsoft

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

	#hist #frames	WER (%)	RTF (#thread) 1	4
RNN-T	$+\infty$	9.86	1.56	0.46
T-T	$+\infty$	8.79	3.44	2.57
T-T	60	8.88	2.38	1.75
C-T	$+\infty$	8.78	4.02	2.56
C-T	60	8.80	2.41	1.83

- 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

	#hist len	WER (%)	RTF (#batch size)				
			1	2	5	10	15
RNN-T	$+\infty$	9.86	0.46	0.31	0.26	0.21	0.20
T-T	60	8.88	1.75	0.69	0.38	0.26	0.19
C-T	60	8.80	1.83	0.95	0.48	0.36	0.25

- 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)

WER and RTF results with lookahead

	#hist frame	#lookahead (ms)	WER (%)	RTF (#thread)	
				1	4
Hybrid	$+\infty$	480	9.34	-	-
RNN S2S	$+\infty$	720	9.61	-	-
Trans. S2S	$+\infty$	[480, 960]	9.16	-	-
Trans. S2S	$+\infty$	$+\infty$	7.82	-	-
RNN-T	$+\infty$	360	9.11	1.52	0.43
T-T	60	[0,720]	8.28	0.40	0.16
C-T	60	[0,720]	8.19	0.45	0.22
T-T	$+\infty$	$+\infty$	7.78	0.39	0.15
C-T	$+\infty$	$+\infty$	7.69	0.36	0.15

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

8-bit quantization

	Precision	WER (%)	RTF
RNN-T	float32	9.11	1.56
	int8	9.13	0.43
T-T	float32	8.28	0.40
	int8	8.50	0.22
C-T	float32	8.19	0.45
	int8	8.40	0.26

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

Streaming End-to-End Multi-talker Speech Recognition

L. Lu, et al., "Streaming end-to-end multi-talker speech recognition," arXiv preprint 2020.

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**

  Z. Chen, et al., “Continuous Speech Separation: Dataset and Analysis”, ICASSP 2020

- **Hybrid joint training approach**

  D. Yu, et al., “Recognizing multi-talker speech with permutation invariant training”, Interspeech, 2017

  Y. Qian, et al., “Single-channel multi-talker speech recognition with permutation invariant training”, Speech Communication, 2018

- **(Offline) End-to-end approach**

  S. Settle, et al., “End-to-end multi-speaker speech recognition”, ICASSP 2018

  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

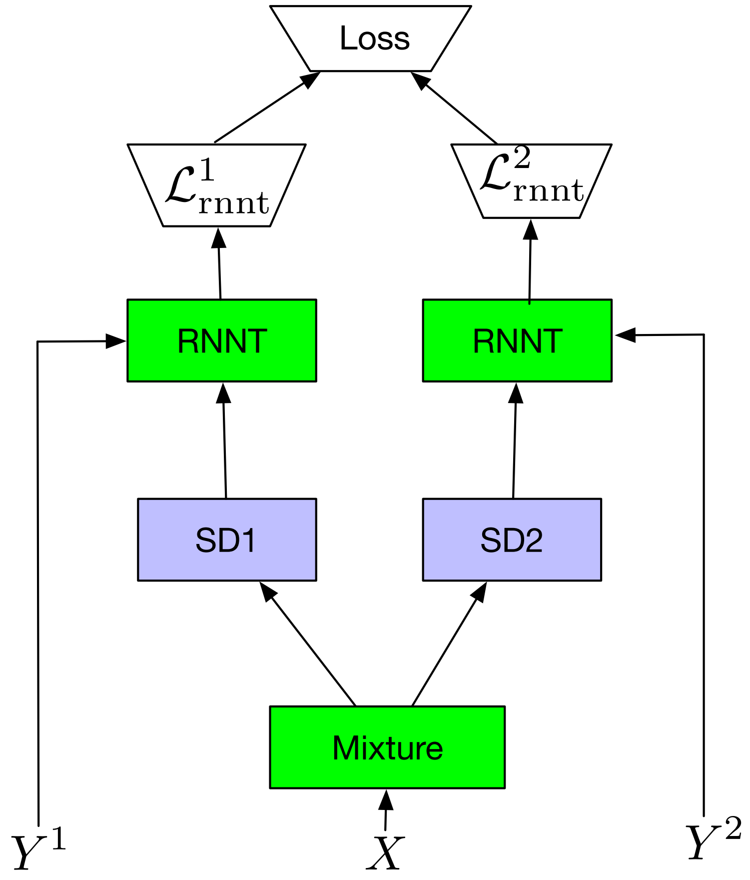
  A. Tripathi, et al. “End-to-end multi-talker overlapping speech recognition”, ICASSP 2020

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

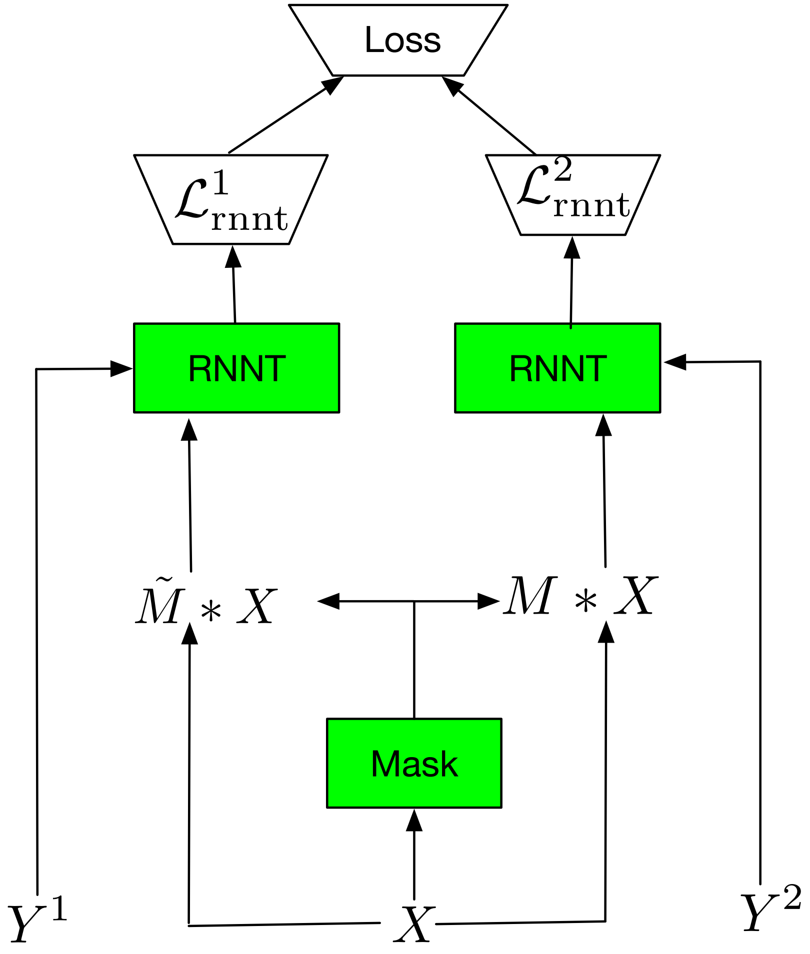
Model Structure

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

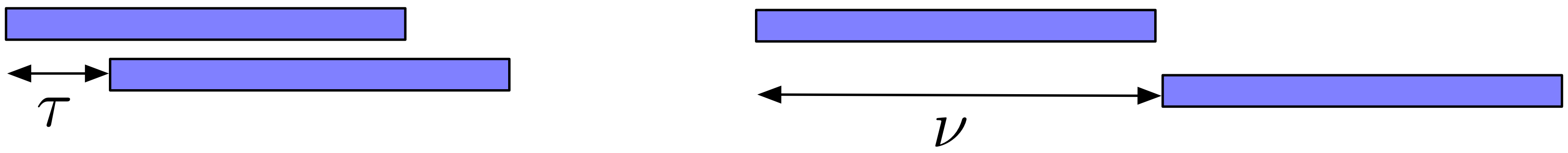
$$\mathcal{L}_{\mathrm{heat}}(X, Y^1, Y^2) = \mathcal{L}_{\mathrm{rnnt}}(Y^1, H_1) + \mathcal{L}_{\mathrm{rnnt}}(Y^2, H_2)$$

  - Based on the timing information to fix the error assignment

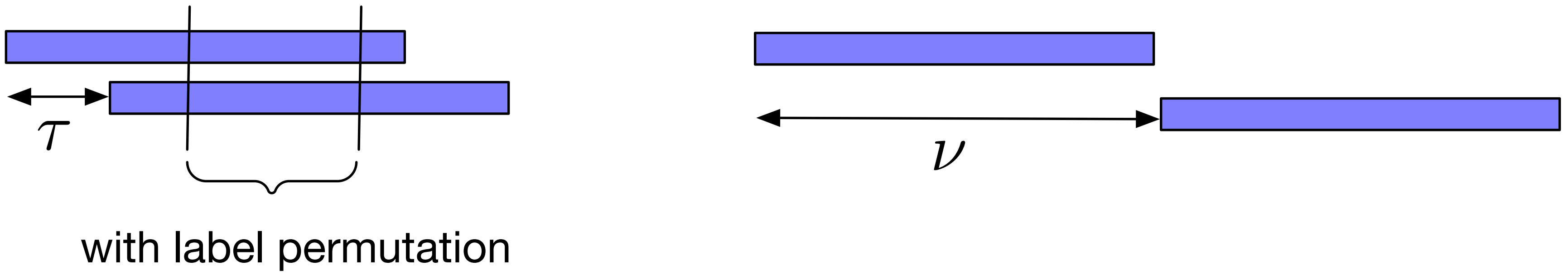
  - Computationally more scalable
  - Similar approach has been studied in:

    A. Tripathi, et al. “End-to-end multi-talker overlapping speech recognition”, in Proc. ICASSP 2020

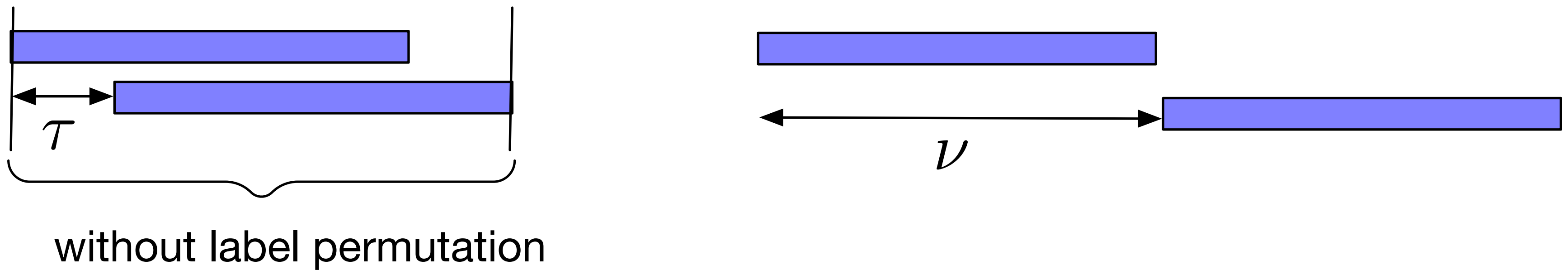
Why it works?



Why it works?

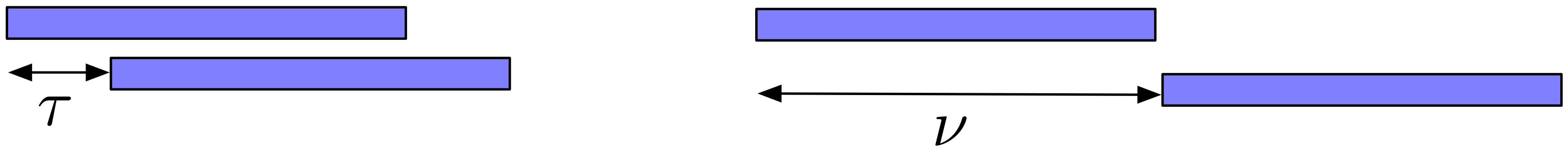


Why it works?



Experiments

- LibrispeechMix: simulated overlapped speech dataset derived from Librispeech
- In our experiments, we only consider 2-speaker case
- Investigating two conditions τ= 0 and τ= 0.5
- Overlapped data sampled form [τ , ν]



SD-based model

Module	Type	Depth	Shape
Mixure	Conv2D	4	$\begin{bmatrix} \texttt{conv2d}(3,64,3,3) \\ \texttt{conv2d}(64,64,3,3) \\ \texttt{Maxpool}(3,1) \\ \texttt{conv2d}(64,128,3,3) \\ \texttt{Maxpool}(3,1) \\ \texttt{conv2d}(128,128,3,3) \\ \texttt{Maxpool}(3,1) \\ \texttt{Linear} \end{bmatrix}$
SD1	LSTM	2	(1024, 1024)
SD1	LSTM	2	(1024, 1024)
RNNT-A	LSTM	2	(1024, 1024)
RNNT-L	LSTM	2	(1024, 1024)

Mask-based model

Module	Type	Depth	Shape
Mask	Conv2D	4	$\begin{bmatrix} \texttt{conv2d}(3,64,3,3) \\ \texttt{conv2d}(64,64,3,3) \\ \texttt{Maxpool}(3,1) \\ \texttt{conv2d}(64,128,3,3) \\ \texttt{Maxpool}(3,1) \\ \texttt{conv2d}(128,128,3,3) \\ \texttt{Maxpool}(3,1) \\ \texttt{Linear} \\ \texttt{Sigmoid} \end{bmatrix}$
RNNT-A	LSTM	6	(771, 1024)
RNNT-L	LSTM	2	(1024, 1024)

Results

Train	Model	Loss	$\tau = 0$		$\tau = 0.5$	
			dev	test	dev	test
$\tau = 0.5$	SD	PIT	12.0	12.1	11.3	11.4
		HEAT	11.8	11.7	10.9	10.9
	Mask	PIT	14.1	14.1	13.8	13.1
		HEAT	13.4	13.1	12.3	12.2
$\tau = 0$	SD	PIT	13.1	13.2	11.8	11.9
		HEAT	12.5	12.5	11.2	11.3

Results

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

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

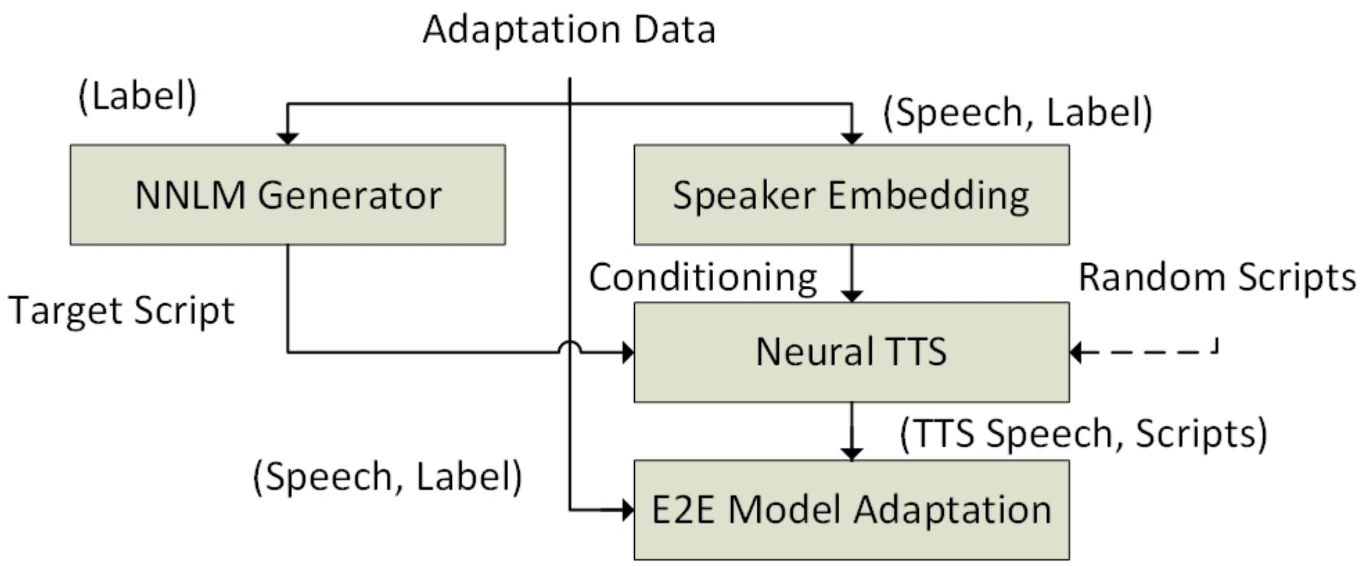
Train	Model	Size	Latency	$\tau = 0$ dev	$\tau = 0$ test
$\tau = 0.5$	SURT	80M	120 ms	11.8	11.7
	PIT-S2S [18]	160.7M	∞	–	11.1

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

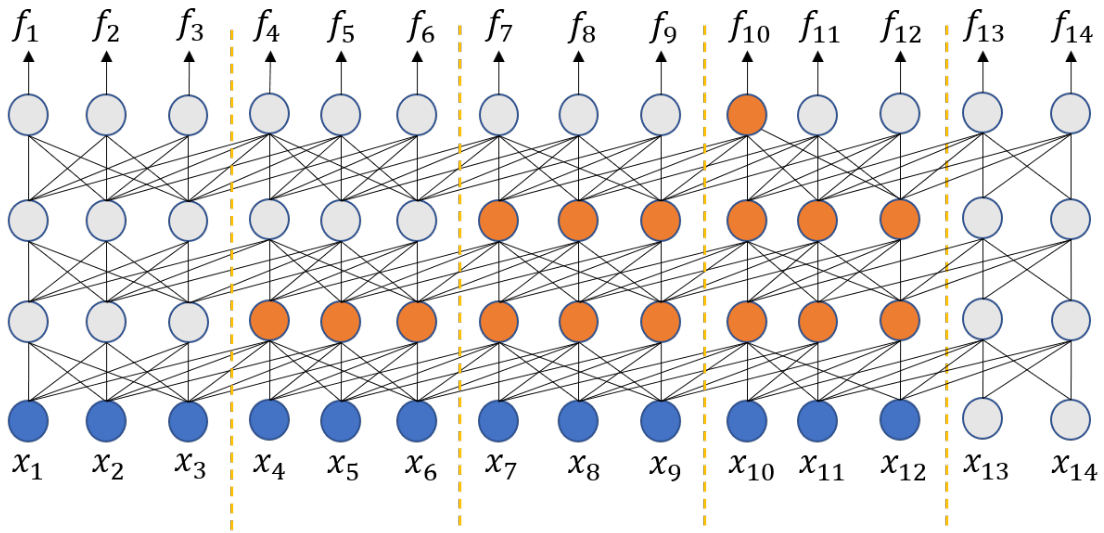
Conclusions

- 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.



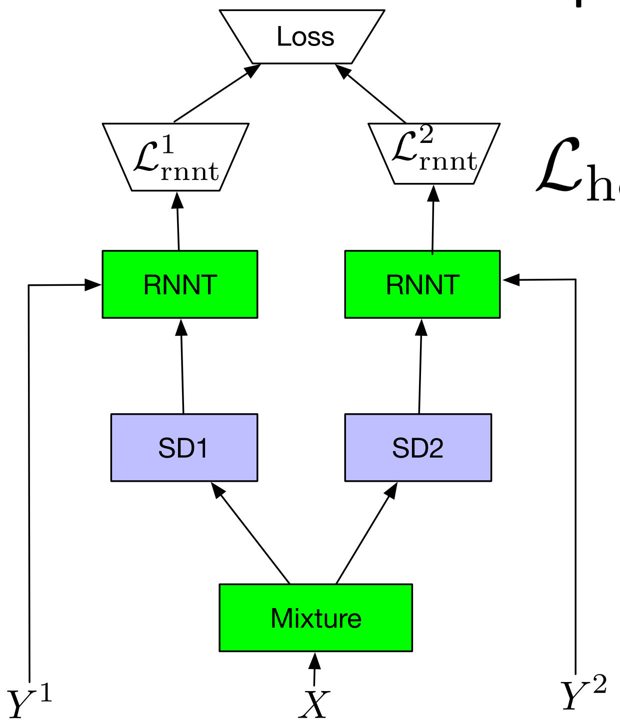
Conclusions

- 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.



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

Thank You!