CTC-synchronous Training for Monotonic Attention Model

Hirofumi Inaguma[1] Masato Mimura[1] Tatsuya Kawahara[1]
[1]Graduate School of Informatics, Kyoto University, Japan

Background: End-to-end ASR

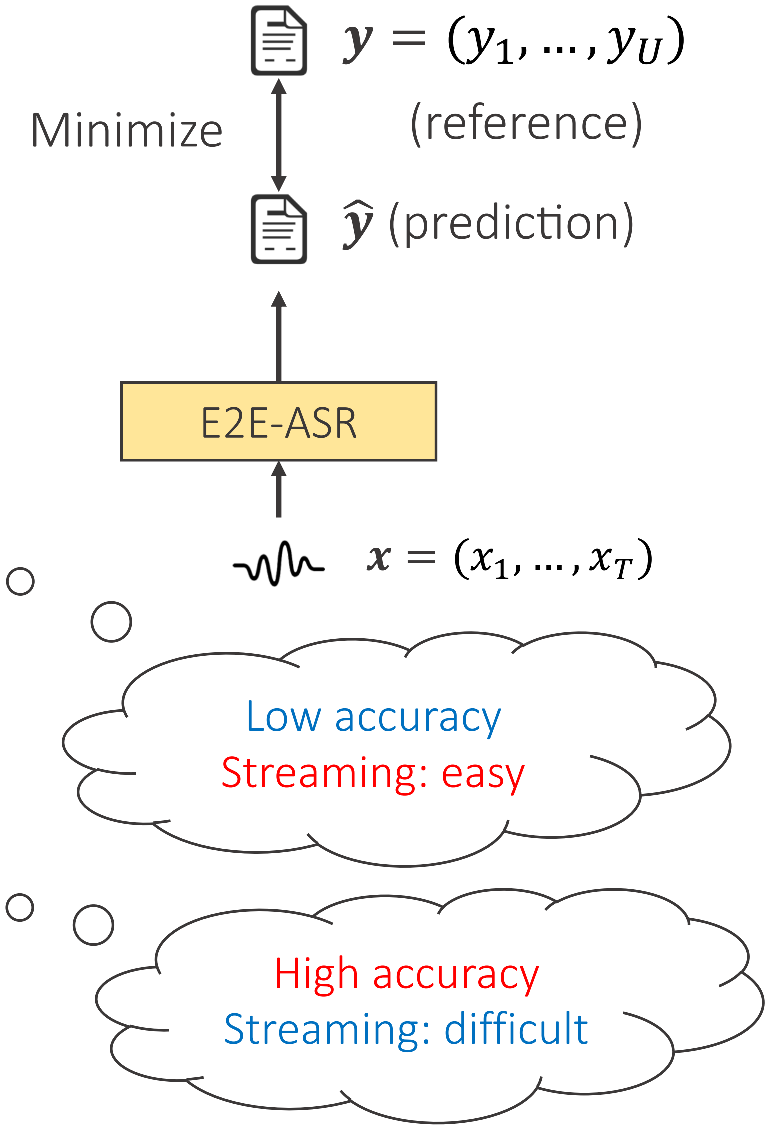
Time-synchronous model ($|\boldsymbol{x}| = |\hat{\boldsymbol{y}}|$)

- Connectionist temporal classification (CTC) [Graves et al., 2006]
- RNN-Transducer (RNN-T) [Graves et al., 2013]
- Recurrent neural aligner (RNA) [Sak et al., 2017]

Label-synchronous model ($|\boldsymbol{x}| \neq |\hat{\boldsymbol{y}}|$)

- Attention-based RNN encoder-decoder [Bahdanau et al., 2016]
- Transformer [Vaswani et al., 2017]

The entire encoder outputs are required to generate the initial token



- RNN-T is dominant for streaming E2E-ASR in the industry
  - ➢ Memory-consuming, thus requires distributed training and small vocabulary etc.
  - ➢ Large search space because of frame-wise predictions

Streaming attention-based models

- Simple
- Good results
- Efficient training
- Linear time decoding

Neural Transducer [Jailty et al.,2015]

- Perform attention mechanism for a fixed size of block

Hard monotonic attention [Raffel et al., 2017]

- Learn to detect token boundaries via stochastic binary decision
- Extension: Monotonic chunkwise attention (MoChA) [Chiu et al., 2018]

Triggered attention [Moritz et al., 2018]

- Perform global attention over encoder memories truncated by CTC spikes

Adaptive computation steps (ACS) [Li et al., 2018]

- Learn how many tokens to generate with encoder outputs

Continuous Integrate-and-Fire (CIF) [Dong et al., 2019]

- Fine-grained version of ACS

And more…

- Windowing approaches
- Incremental decoding
- Reinforcement learning

Hard monotonic attention (HMA) [Raffel+ 2017]

Test time

h_j: encoder state
s_i: decoder state

$$e_{i,j} = \mathrm{MonotonicEnergy}(h_j, s_i)$$

Not differentiable

$p_{i,j} = \sigma(e_{i,j})$ (selection probability)

$z_{i,j} \sim \mathrm{Bernoulli}(p_{i,j})$ (If $z_{i,j} = 1$, $c_i = h_j$)

Points
- Linear-time decoding $\mathrm{O}(T)$ during inference
- HMA has options to
  - (1) stop at the current frame j
  - (2) move forward to the next frame $j+1$
- Introduce a binary decision process $z_{i,j}$ to decide whether to attend to h_j or not

Training time

$$\alpha_{i,j} = p_{i,j} \sum_{k-1}^{j} \left(\alpha_{i-1,k} \prod_{l=k}^{j-1} (1 - p_{i,l}) \right)$$

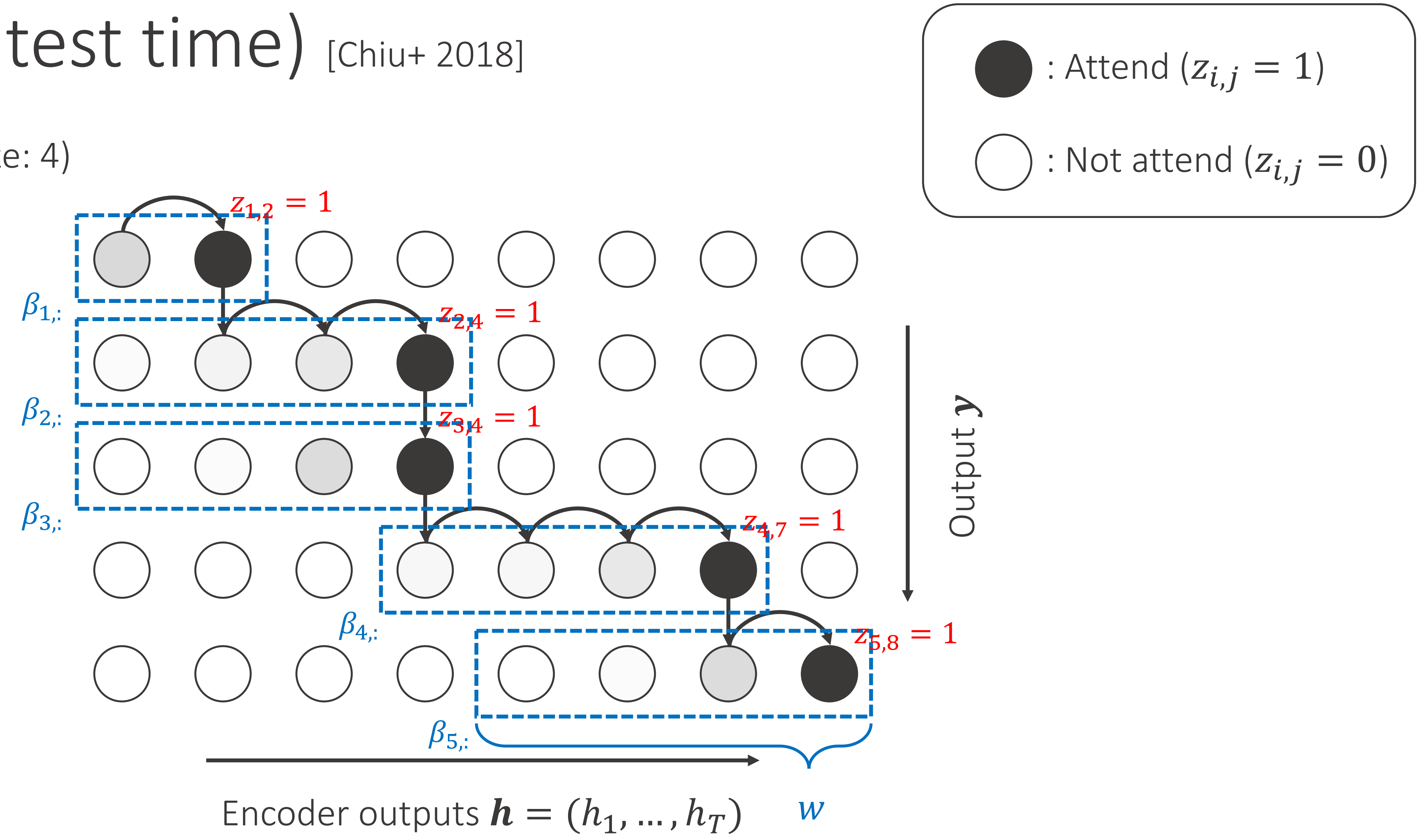
$$= (1 - p_{i,j-1}) \frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}$$

$p_{i,j} = \sigma(e_{i,j} + \varepsilon)$, $\varepsilon \sim \mathcal{N}(0,1)$

Calculate expected alignments $\alpha_{i,j}$

MoChA (test time) [Chiu+ 2018]

e.g., $w = 4$ (chunk size: 4)



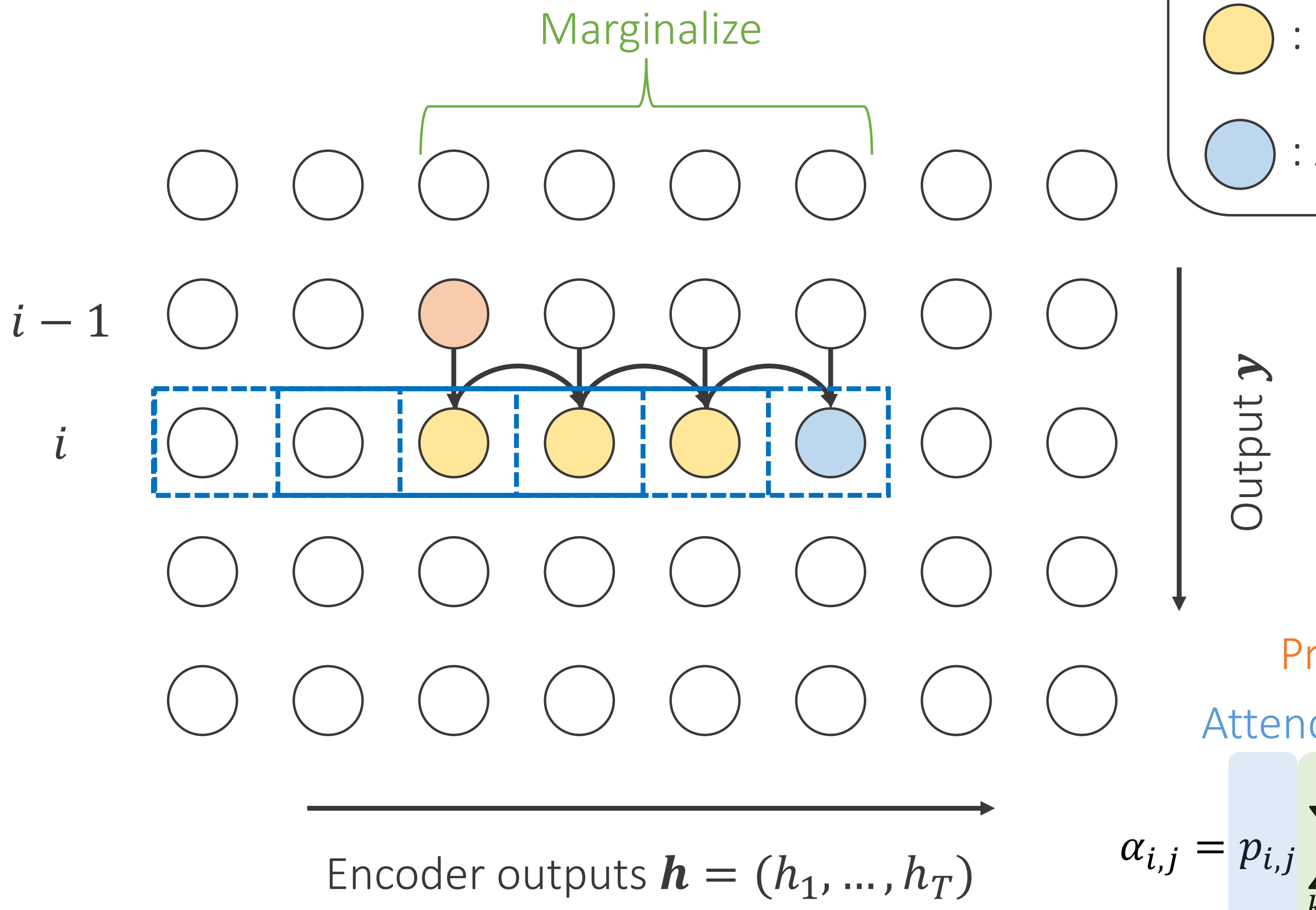
1. *Monotonic attention: whether to attend or not*
2. *Chunkwise attention: soft attention over a small window*

MoChA (training time) [Chiu+ 2018]

: Attend at $(i-1)$-th step

: Not attend

: Attend at $\boldsymbol{i}$-th step



Previous attention

Attend

Not attend

$$\alpha_{i,j} = p_{i,j} \sum_{k-1}^{j} \left(\alpha_{i-1,k} \prod_{l=k}^{j-1} (1 - p_{i,l}) \right)$$

$$= (1 - p_{i,j-1}) \frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}$$

Can be implemented efficiently in parallel with j

Optimization problem

Recap

$$\alpha_{i,j} = (1 - p_{i,j-1})\frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}$$

$$p_{i,j} = \sigma(e_{i,j})$$

1. **$\sum_j \boldsymbol{\alpha_{i,j}} = \mathbf{1}$ is not satisfied during training**
   - $\alpha_{i,j}$ is <u>NOT globally normalized</u> over the whole encoder outputs $\{h_j\}_{j=1,\ldots T}$
     - ➢ $\alpha_{i,j}$ is not a valid probability distribution
     - ➢ $\alpha_{i,j}$ attenuates quickly during marginalization
     - ➢ Selection probability $p_{i,j}$ is not learnt well
   - Enlarge the mismatch between training and test time

2. **Alignment errors are propagated to later token generation**
   - $\alpha_{i,j}$ depends on past alignments
   - <u>Backward algorithm cannot be used</u> for $\alpha_{i,j}$
     - ➢ $\alpha_{i,j}$ is not a valid probability distribution
     - ➢ Autoregressive decoder

Problematic for long and noisy speech utterances

- Model needs to learn (1) **a proper scale of $\boldsymbol{\alpha_{i,j}}$** and (2) **accurate decision boundaries** (j s.t. $\alpha_{i,j} = 1$) at the same time

Quantity regularization

- Add a regularization term to encourage $\sum_j \alpha_{i,j} = 1$

$$\mathcal{L}_{\text{qua}} = |U - \sum_{i=1}^{U} \sum_{j=1}^{T} \alpha_{i,j}|$$

$$\mathcal{L}_{\text{total}} = (1 - \lambda_{\text{ct}c})\mathcal{L}_{\text{s}2s} + \lambda_{\text{ctc}}\mathcal{L}_{\text{ctc}} + \lambda_{\text{qua}}\mathcal{L}_{\text{qua}} \quad (\lambda_{\text{qua}} \geq 0)$$

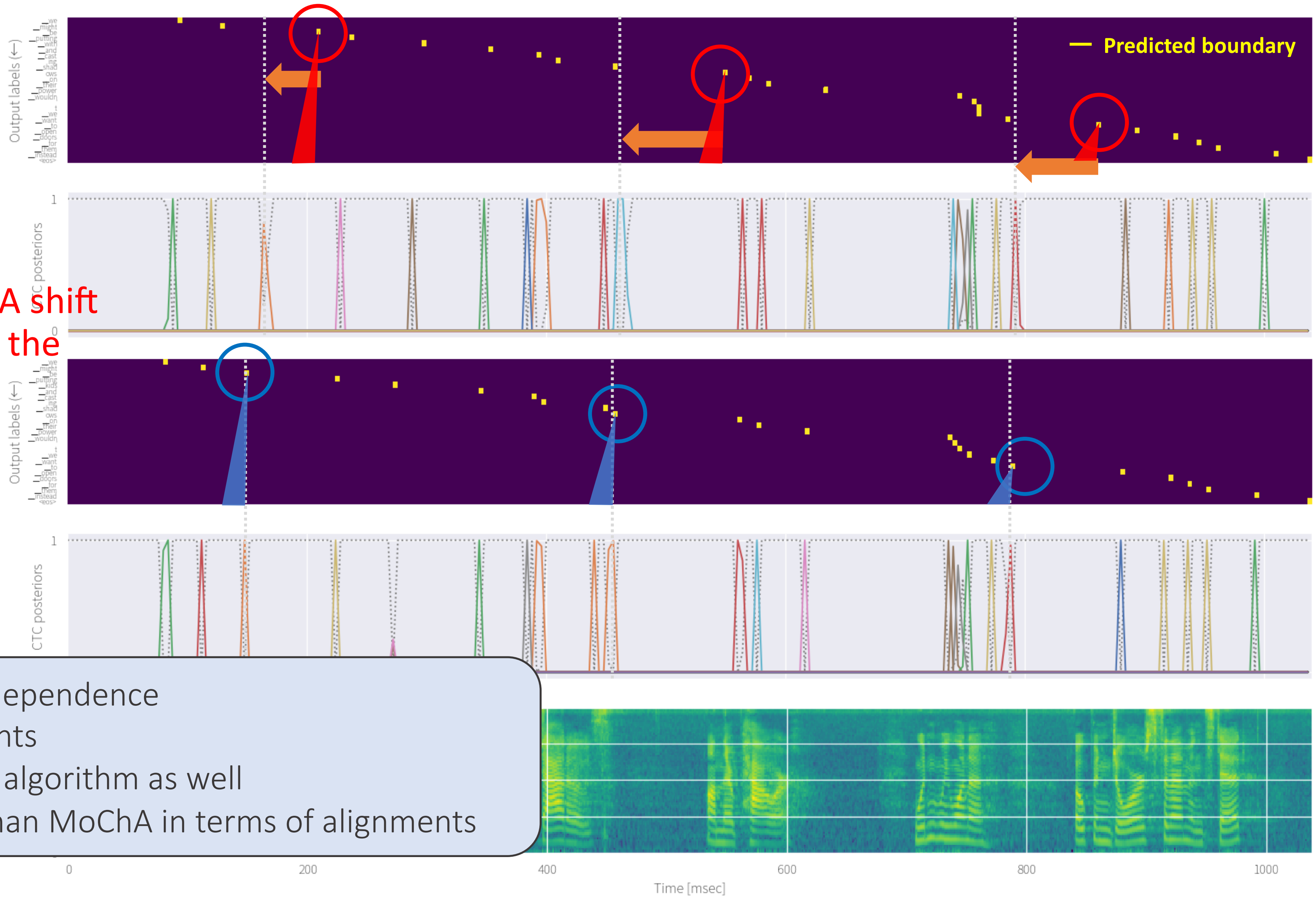
- Quantity loss is not effective on large-scale data (3.4k hours) [Inaguma+ 2020], but helpful for small and medium size data (<1k hours)

Preliminary: Comparison of boundary positions (CTC vs.MoChA)

Baseline w/ quantity regularization

Decision boundaries of MoChA shift to the right side (future) from the corresponding CTC spikes

Proposed



- CTC assumes conditional independence
  - ➢ Robust to past alignments
- CTC leverages the backward algorithm as well
  - ➢ CTC is more accurate than MoChA in terms of alignments

Proposed method: CTC-synchronous training (CTC-ST)

- Leverage CTC's posterior spikes as reference boundaries for MoChA
- MoChA is trained to mimic the CTC model to generate the similar decision boundaries
- External alignments from hybrid ASR are not required [Inaguma+ 2020]

Objective function

CTC boundary Expected MoChA boundary

$$\mathcal{L}_{\mathrm{sync}} = \frac{1}{U}\sum_{i=1}^{U} |\mathrm{b}_i^{\mathrm{ctc}} - \sum_{j=1}^{T} j\alpha_{i,j}|$$

$$\mathcal{L}_{\mathrm{total}} = (1-\lambda_{\mathrm{ctc}})\mathcal{L}_{\mathrm{mocha}} + \lambda_{\mathrm{ctc}}\mathcal{L}_{\mathrm{ctc}} + \lambda_{\mathrm{qua}}\mathcal{L}_{\mathrm{qua}} + \lambda_{\mathrm{sync}}\mathcal{L}_{\mathrm{sync}} \quad (\lambda_{\mathrm{sync}} \geq 0)$$

- Unless otherwise noted, λ_{qua} is set to 0 when using CTC-ST

Extraction of CTC alignments

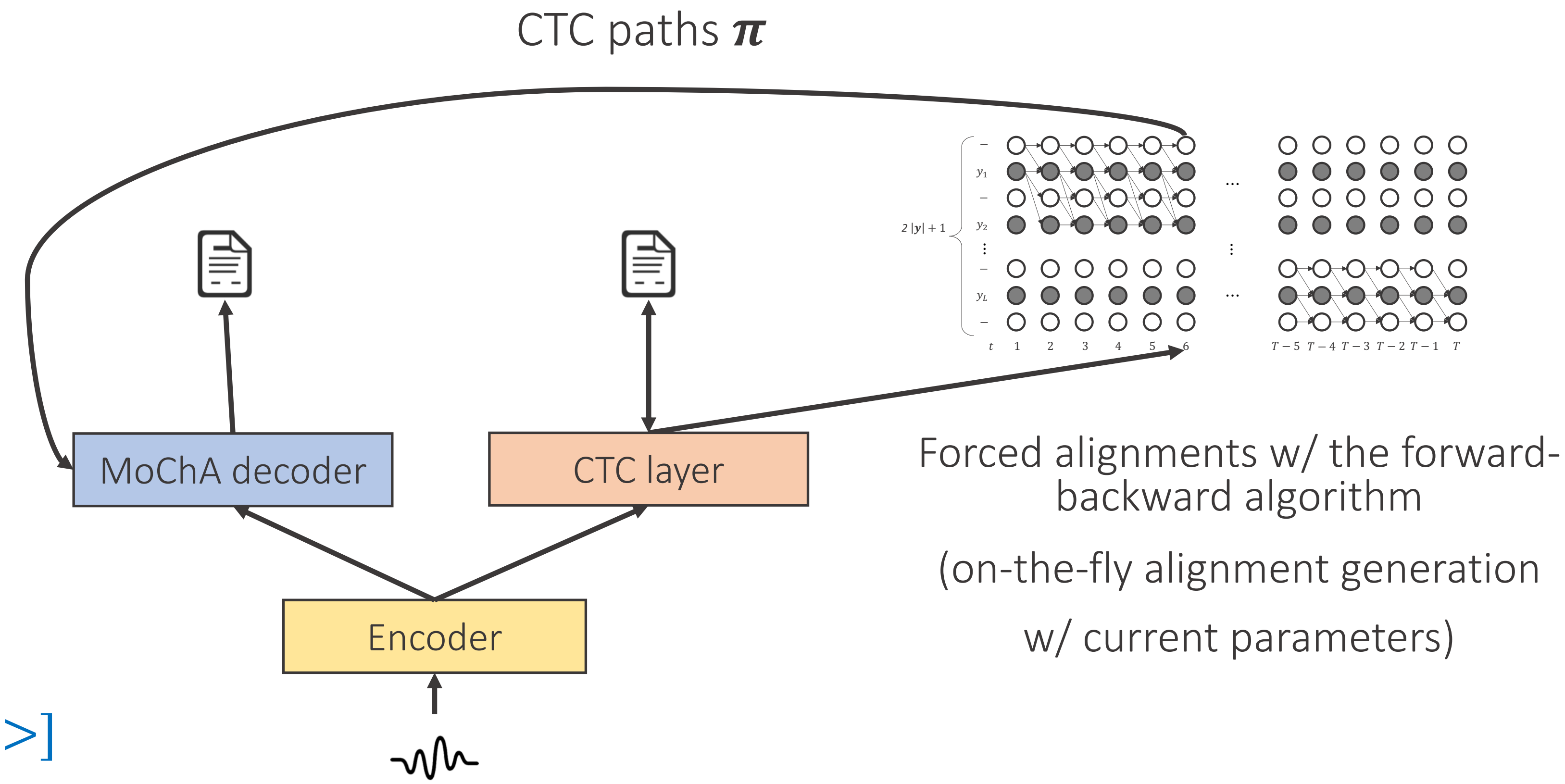
- Encoder network is shared between both branches
- Both branches are jointly optimized
- CTC alignments are extracted via forced alignment over the transcription

Most probable CTC path $\bar{\pi}$

$$\bar{\pi} = [-CC-AAA-TT-]$$

Use the leftmost index

$$\bar{\pi} = [-C--A---T-<eos>]$$



Curriculum learning strategy

- Applying CTC-ST from scratch is inefficient because $\sum_{j=1}^{T} \alpha_{ij} \ll 1$ in the early training stage
  - ➢ Difficult to estimate the expected boundary positions $\sum_{j=1}^{T} j\alpha_{i,j}$ accurately
  - ➢ Propose curriculum learning strategy composed of two stages

Stage-1 (expected to learn a proper scale of α_{ij})

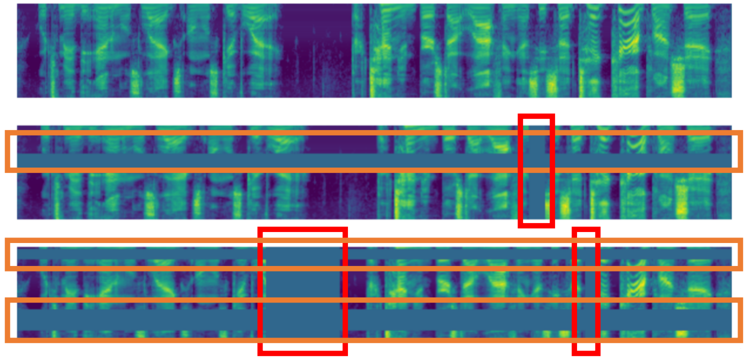
- Train BLSTM encoder + MoChA with quantity regularization until convergence

Stage-2 (expected to learn boundary location)

- Initialize with model parameters in stage-1
- Train latency-controlled BLSTM (LC-BLSTM) encoder + MoChA with CTC-ST

NOTE: When using the unidirectional LSTM encoder, the same encoder is used in both stages

Combination with SpecAugment



SpecAugment [Park et al., 2019]

- On-the-fly data augmentation method over input log-mel filterbank features
- Zero out successive frames in time and frequency bins

Recap

$$\alpha_{i,j} = \left(1 - p_{i,j-1}\right)\frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}$$

Problem of SpecAugment for MoChA

- Recurrency of $\alpha_{i,j}$ can be easily collapsed after the masked region
- The naïve MoChA did not obtain any gains with SpecAugment
- CTC can estimate boundaries accurately even right after the masked region thanks to the conditional independence assumption per frame
- CTC-ST is expected to improve the effectiveness of SpecAugment for MoChA

Experimental condition

Corpus	TEDLUM2 (210h, lecture), Librispeech (960h, read)
Feature	80-dim log-mel fbank
Output unit	BPE 10k units
Architecture	Offline: 4-layer CNN -> 512-dim (per direction) 5-layer BLSTM encoder Streaming: 4-layer CNN -> 512-dim 5-layer LC-BLSTM encoder or 4-layer CNN -> 1024-dim 5-layer unidirectional LSTM encoder
	Decoder: 1024-dim 1-layer LSTM w: 4 (window size for chunkwise attention in MoChA)
Optimization	Adam
Loss weight	$\lambda_{\mathrm{ctc}} = 0.3$, $\lambda_{\mathrm{qua}} = 1.0$, $\lambda_{\mathrm{sync}} = 1.0$
Decoding	Beam width: 10, shallow fusion with external 4-layers of LSTM-LM

Main results: TEDLIUM2 (210h)

Latency-controlled BLSTM

LC-BLSTM-N_l+ N_r

hop size (ms) lookahead frame (ms)

	Model	%WER
Offline	LSTM - standard attention	11.9
	BLSTM - standard attention (T1)	9.5
	BLSTM - MoChA	12.6
	+ Quantity regularization (T2)	**9.8**
	+ CTC-ST	10.2
Streaming	LSTM - MoChA	15.0
	+ CTC-ST	**13.2**
	LC-BLSTM-40+20 - MoChA	12.2
	+ CTCT-ST	**10.5**
	LC-BLSTM-40+40 - MoChA (T5)	11.3
	+ CTC-ST (T6)	**9.9**
	+ Quantity regularization	10.1

init.

init.

22.2% (⬆)

12.0% (⬆)

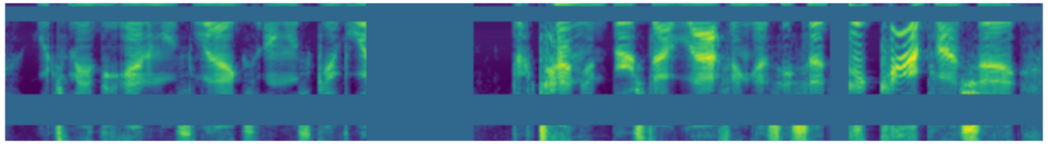
13.9% (⬆)

12.3% (⬆)

- Combination of CTC-ST and quantity regularization was not effective
  - ➢ CTC-ST has a similar effect to improve the scale of α_{ij}
- Curriculum learning was effective

Results with SpecAugment

F {



Frequency mask size

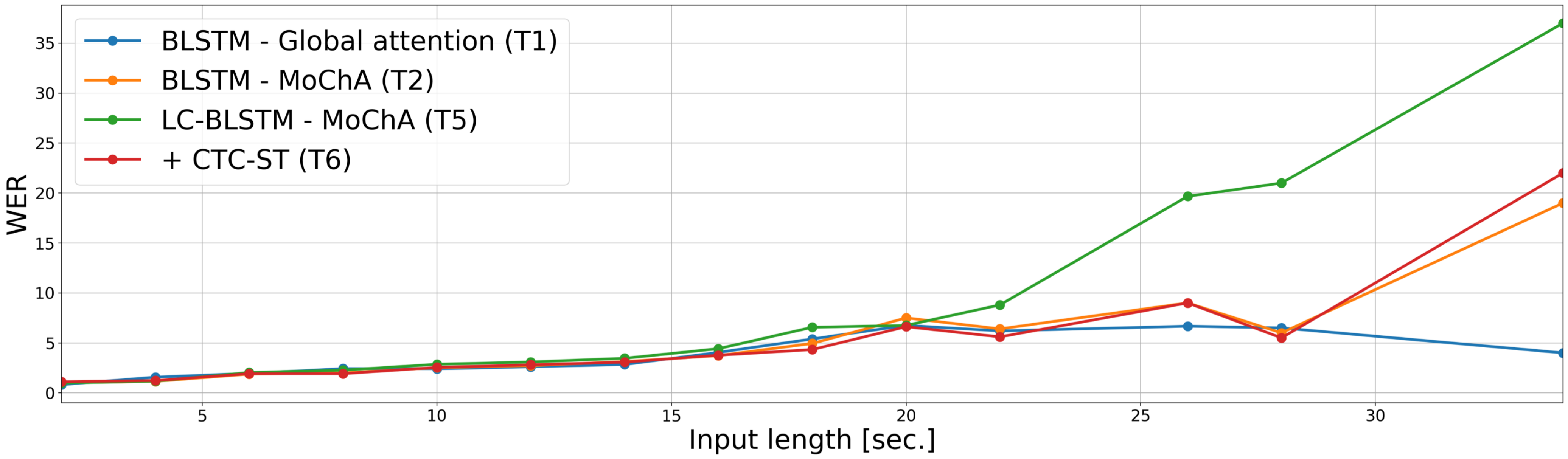
T Time mask size

Model		*F*	*T*	%WER
Offline	Transformer [Karita et al., 2019]	30	40	8.1
	BLSTM - standard attention [Zeyer et al., 2019]	N/A	N/A	8.8
	BLSTM - standard attention	-	-	9.5
		27	100	8.1
Streaming	LC-BLSTM-40-+40 - MoChA (seed: BLSTM - MoChA)	-	-	11.3
		27	100	12.8
		27	50	11.0
		13	50	11.2
	+ CTC-ST	-	-	9.9
		27	100	9.0
		27	50	**8.6**
		13	50	9.0

13.1% (⬆)

- MoChA did not benefit from SpecAugment w/o CTC-ST
- CTC-ST was robust to the input mask size
- Achieved the comparable performance to the offline model (8.1 vs. 8.6)

WER distributions as a function of sequence length



- CTC-ST improved WER for long utterances

Results on Librispeech (960h)

	Model	%WER Test-clean	%WER Test-other
Offline	BLSTM - standard attention	3.1	9.5
	+ SpecAugment ($F=27, T=100$)	2.8	7.6
	BLSTM - MoChA	3.6	10.5
	+ Quantity regularization (T2)	**3.3**	**10.0**
Streaming	LSTM - MoChA	5.3	14.5
	+ CTC-ST	4.7	13.6
	+ SpecAugment ($F=13, T=50$)	**4.2**	**11.2**
	LC-BLSTM-40+40 - MoChA	4.1	11.2
	+ SpecAugment ($F=27, T=100$)	5.0	9.7
	+ SpecAugment ($F=13, T=50$)	4.0	9.5
	+ CTC-ST	3.9	11.2
	+ SpecAugment ($F=27, T=100$)	3.6	9.2
	+ SpecAugment ($F=27, T=50$)	**3.5**	**9.1**
	+ SpecAugment ($F=13, T=50$)	3.6	9.4

init.

init.

8.3/4.7% (⬆)

11.3/6.2% (⬆)

10.2/18.7% (⬆)

Comparison with previous works

Model	%WER	
	Test-clean	Test-other
LSTM - MoChA + MWER [Kim et al,. 2019]	5.6	15.6
LSTM - MoChA + {BPE, char}-CTC + SpecAugment [Garg et al., 2019]	4.4	15.2
LSTM - MoChA + CTC-ST (ours)	**4.2**	**11.2**
LC-BLSTM - sMoChA [Miao et al, 2019]	6.0	16.7
LC-BLSTM - MTA [Miao et al., 2020]	4.2	12.3
LC-BLSTM - MoChA + CTC-ST (ours)	3.9	11.2
+ SpecAugment	**3.5**	**9.1**

Conclusion

- Improving optimization of MoChA with CTC-synchronous training
- Leveraged CTC alignments as an effective guide for MoChA to correct error propagation from past decision boundaries
- CTC-ST significantly improved recognition performances especially for long utterances
- CTC-ST can bring out the full potential of SpecAugment for MoChA
- Explicit interaction between CTC and MoChA <u>on the decoder side</u>
    - ➢Joint CTC/Attention is performed on the encoder side