Toward low-latency and accurate simultaneous interpretations from speech

Hirofumi Inaguma
Ph.D. candidate, Kyoto University, Japan

12/09/2020

Agenda

- ◆ **Streaming end-to-end automatic speech recognition (ASR)**
  - *Monotonic chunkwise attention (MoChA)* [Chiu+ 2018]
  - *How to reduce latency* with alignment information?
  - Where to apply? (encoder/decoder)
    - ➢ *Minimum Latency Training Strategies for Streaming Sequence-to-Sequence ASR [ICASSP 2020]*
  - Leverage CTC alignment (hybrid ASR-free)
    - ➢ *CTC-synchronous Training for Monotonic Attention Model [Interspeech2020]*

- ◆ **Non-autoregressive end-to-end speech translation: A first study**
  - *Conditional masked language model (CMLM)* [Ghazvininejad+ 2019]
  - How to estimate target lengths from speech directly?
    - ➢ Orthros: Non-autoregressive End-to-end Speech Translation with Dual-decoder [under review]

Background: Hybrid ASR system

- Traditional approach (still dominant in production system)

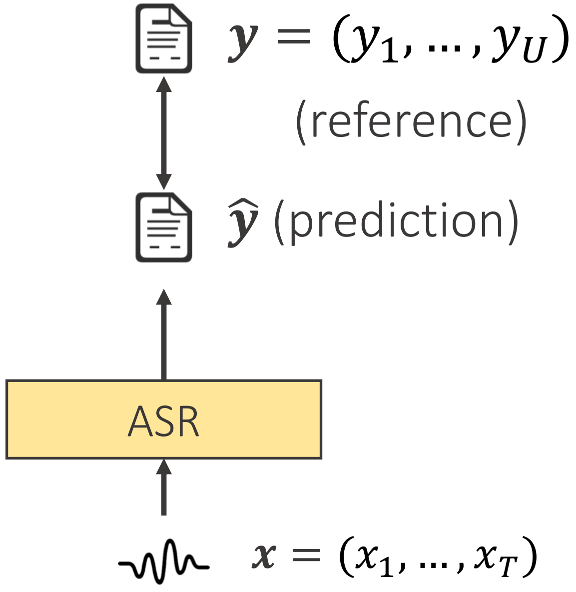
Acoustic model (AM) Language model (LM)

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{\boxed{P(\boldsymbol{x}|\boldsymbol{y})}\,\boxed{P(\boldsymbol{y})}}{P(\boldsymbol{x})}$$

- Inference

$$\begin{aligned}\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{y}} \underline{P(\boldsymbol{x}|\boldsymbol{y})}P(\boldsymbol{y})\end{aligned}$$

$\boldsymbol{y}$ (word)-> $\boldsymbol{p}$ (pronounce)-> $\boldsymbol{s}$ (HMM state)



😊 Rare words, low-resource, module update (customization)

😰 Expertized knowledge

Background: End-to-end ASR system

- Learn a direct mapping function $\varphi(x)$ to maximize $P(\boldsymbol{y}|\boldsymbol{x})$

😊 Quick development, scalability

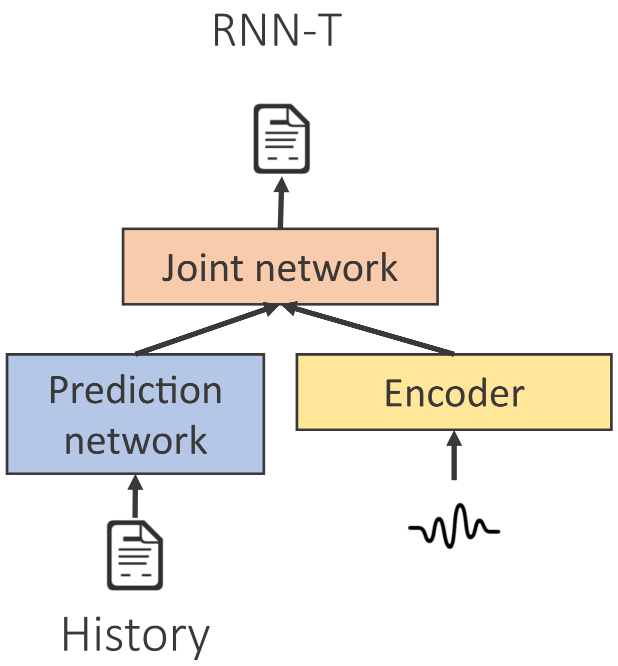
😰 Rare words, low-resource, customization

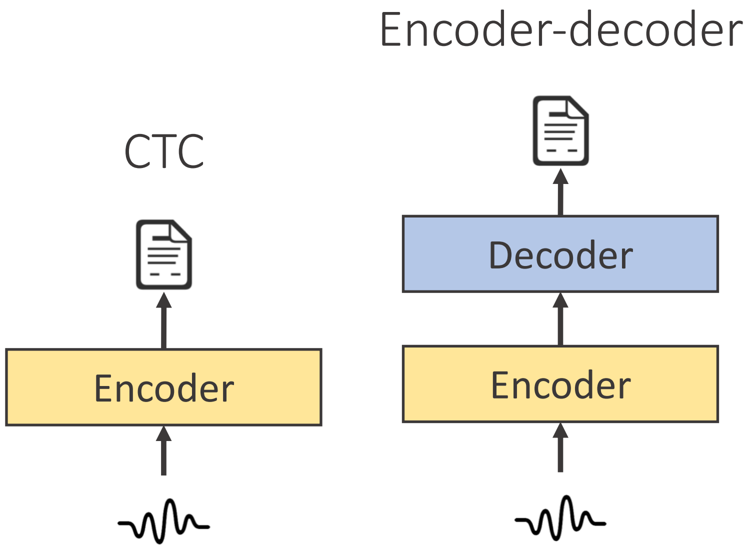
◆**Time-synchronous** model ($|\boldsymbol{x}| = |\hat{\boldsymbol{y}}|$)

- *Connectionist temporal classification (CTC)* [Graves+ 2006]
- *RNN-Transducer (RNN-T)* [Graves+ 2013]
- *Recurrent neural aligner (RNA)* [Sak+ 2017]

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

- *Attention-based RNN encoder-decoder* [Bahdanau+ 2016]
- *Transformer* [Vaswani+ 2017]





Streaming ASR

- Transcribe speech before a speaker finalizes their turn
- Applications
  - ✓Live captioning
  - ✓Dialogue system
  - ✓Simultaneous translation

- RNN-T is dominant in the industry
  - 😊 Stable inference thanks to frame-wise prediction
  - 😰 Memory-consuming training (-> small mini-batch size)
    - ✓Distributed training (a log of GPUs)
    - ✓Efficient implementation (not publicly available in general)
    - ✓Small vocabulary size etc. are required
  - 😰 Large search space due to frame-wise predictions (slow inference)

Challenges in label-synchronous streaming ASR

- Why label-sync. models instead of RNN-T?
  - ➢Small memory consumption
  - ➢Small search space (fast inference)

- Challenges in label-sync. streaming models
  1. Need to modify the decoding scheme
     - ➢The whole encoder outputs are required to generate the first token in general seq2seq models
  2. Poor performance for long-form speech
     - ➢Exposure bias (not occur in frame-synchronous models such as RNN-T)

Streaming attention-based encoder-decoder models

Learn when to generate the next token (segment audio) *on the encoder side*

Flexible audio segmentation policy

◆ **Triggered attention** [Moritz+ 2018]
Global attention over past encoder outputs truncated by CTC spikes

◆ **Adaptive computation steps (ACS)** [Li+ 2018]
Learn how many tokens to generate with encoder outputs (halting mechanism)

◆ **Continuous Integrate-and-Fire (CIF)** [Dong+ 2019]
Fine-grained version of ACS

Fixed audio segmentation policy

◆ **Neural Transducer** [Jailty+ 2015]
Attention mechanism for a fixed size of block

- Simple framework
- Good results
- Efficient training
- Linear time inference

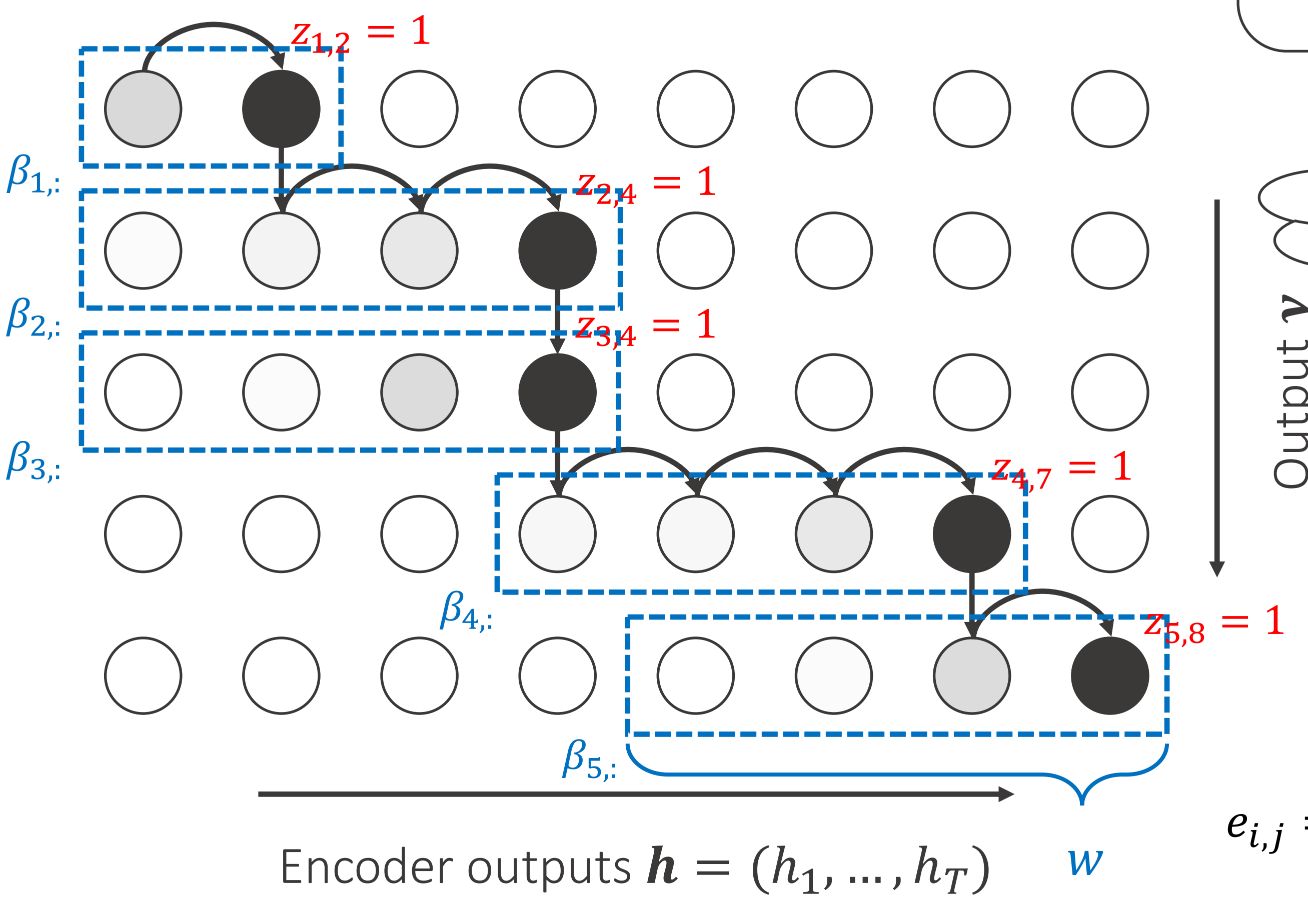
Learn when to generate the next token (segment audio) *on the decoder side*

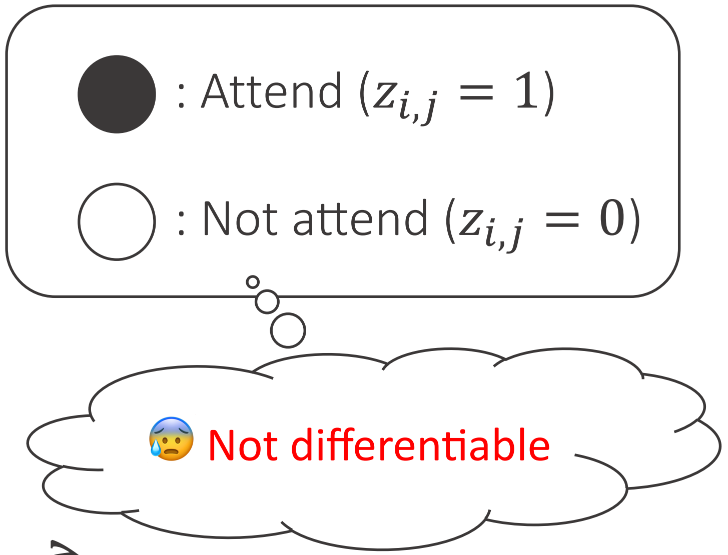
◆ **Monotonic chunkwise attention (MoChA)** [Raffel+ 2017], [Chiu+ 2018]
Learn to detect token boundaries via stochastic binary decision

And more…
- Windowing approaches
- Incremental decoding
- Reinforcement learning

MoChA (test time) [Chiu+ 2018]

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





h_j: encoder state
s_i: decoder state

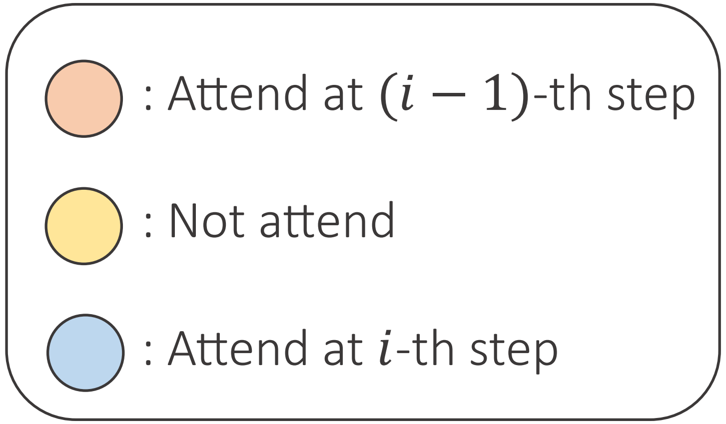
$$e_{i,j} = \mathrm{MonotonicEnergy}(h_j, s_i)$$

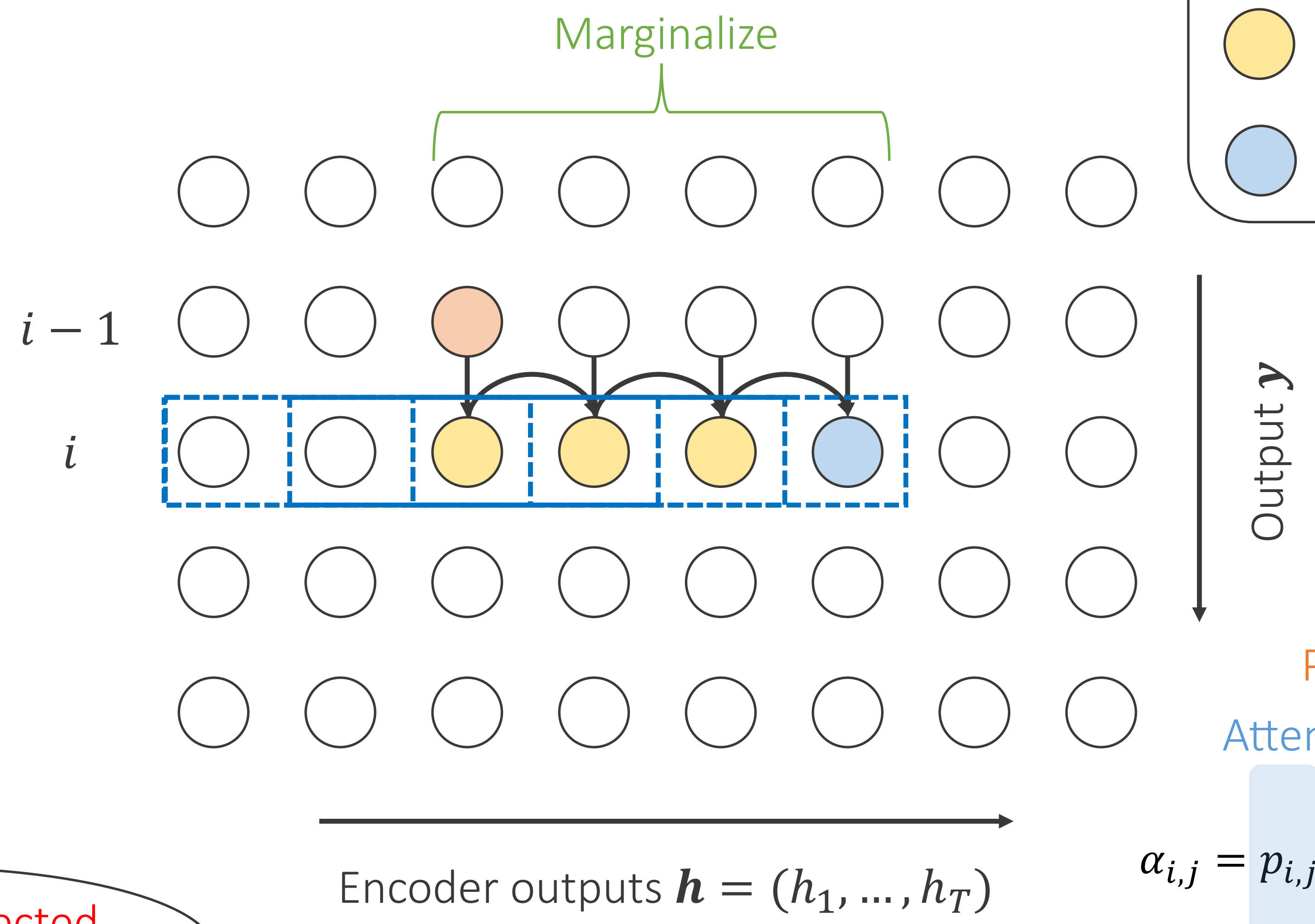
$$p_{i,j} = \sigma(e_{i,j})$$ (selection probability)

$$z_{i,j} \sim \mathrm{Bernoulli}(p_{i,j})$$

1. *Hard monotonic attention* [Raffel+ 2017]: *whether to attend or not*
2. *Chunkwise attention: soft attention over a small window of size* w

MoChA (training time) [Chiu+ 2018]





Calculate expected alignments $\boldsymbol{\alpha}$

Can be implemented efficiently in parallel with j

Previous attention

Attend

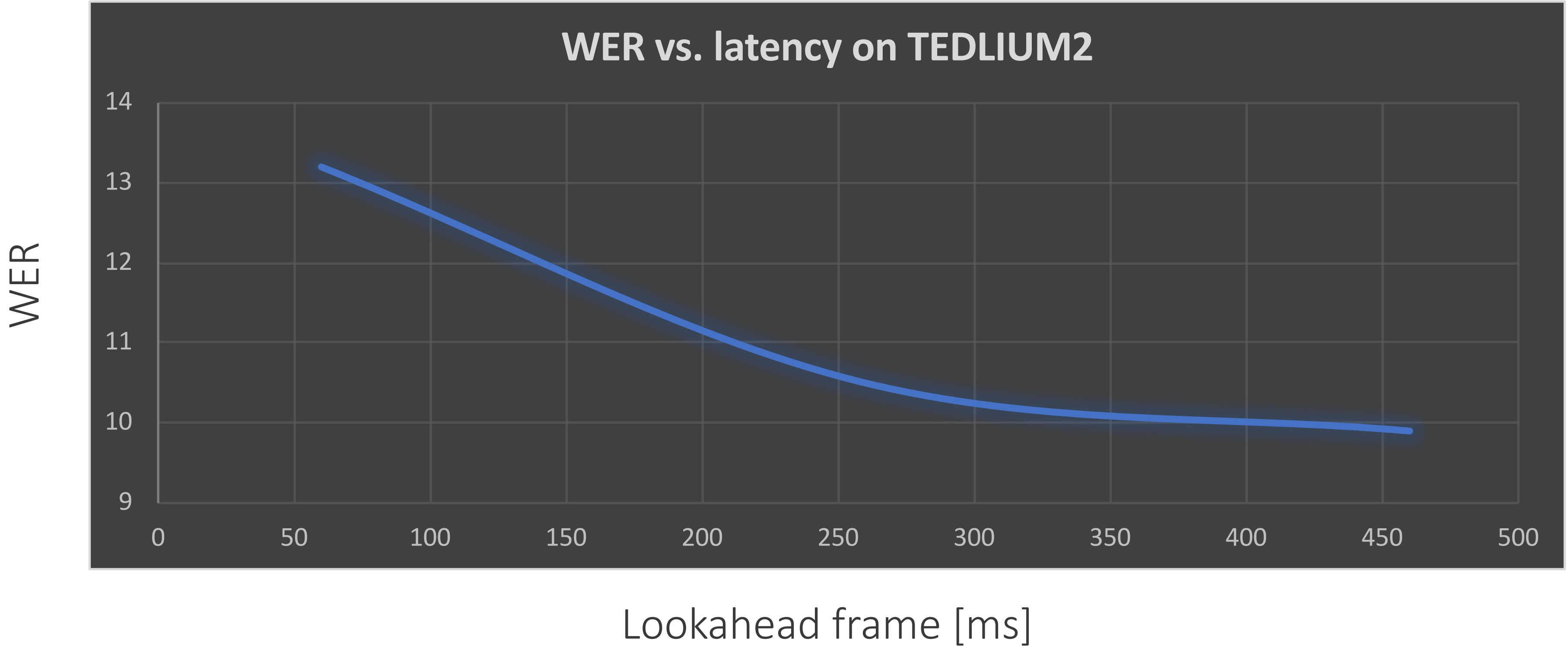
Not attend

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

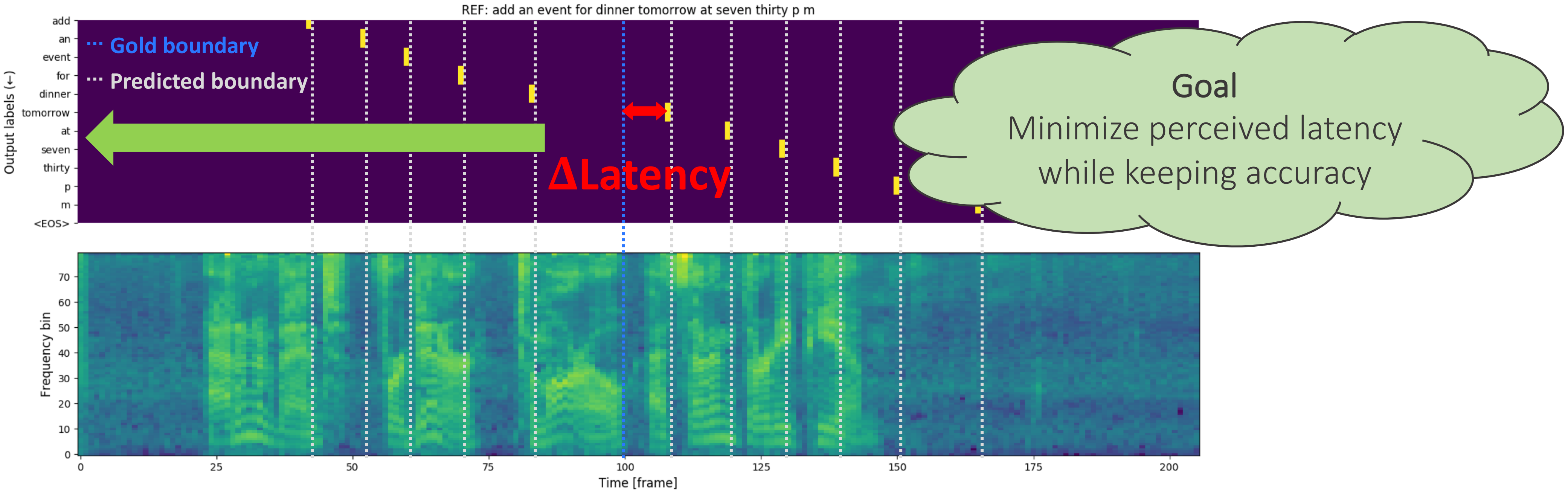
$$= (1 - p_{i,j-1}) \frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}$$

Lookahead latency and accuracy trade-off in streaming ASR

- Future information (lookahead) is very important to improve accuracy
- Large lookahead leads to large **algorithmic** latency
  - ➢Can be controlled on demand



Delayed token generation problem



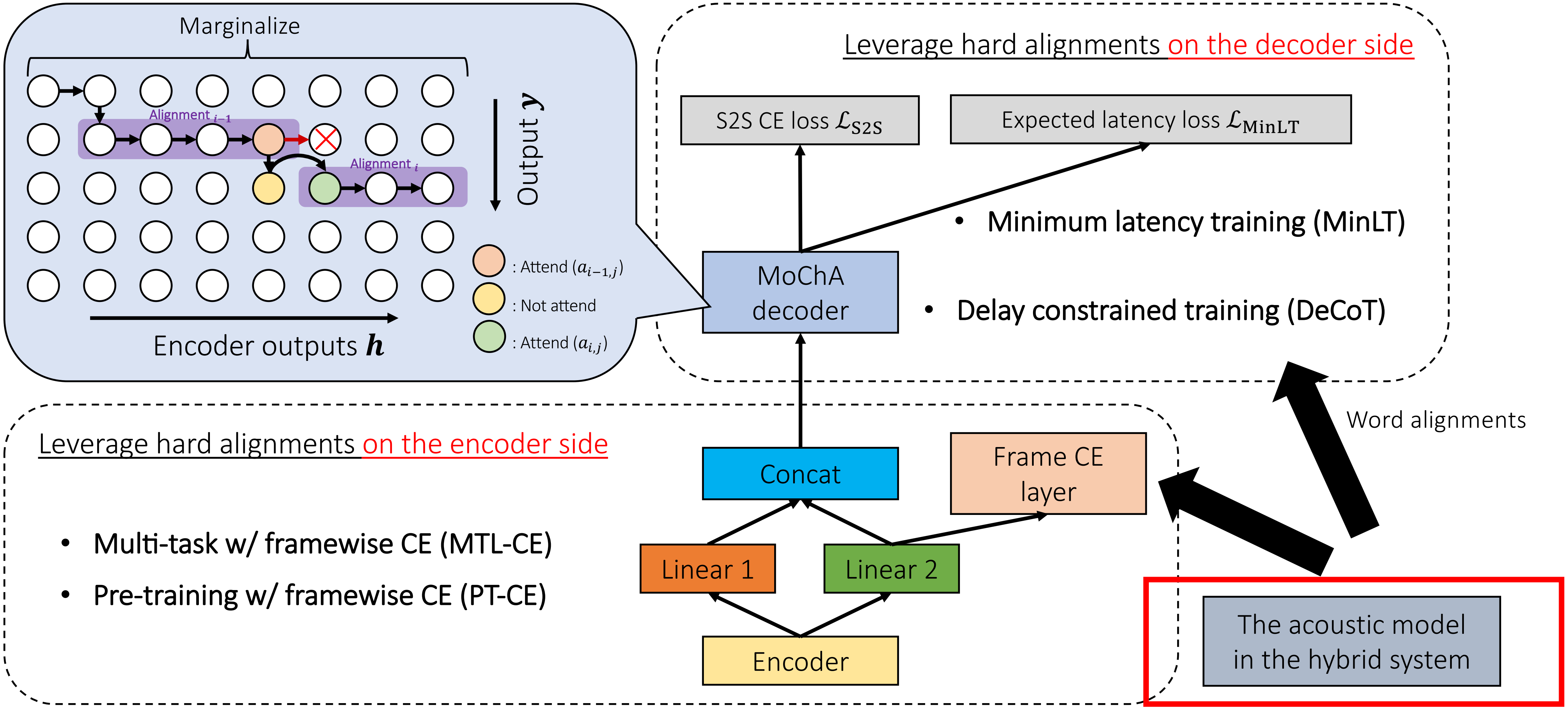
- Decision boundaries (yellow dots) are delayed from the corresponding acoustic boundary
    1. Unidirectional encoder (lacking the future information)
    2. Sequence-level criterion (utilizing as many future frames as possible to maximize the log-likelihood)
- Increase user perceived latency
    - ➢ Similar behaviors have been reported in CTC [sak+ 2015] and RNN-T [Li+ 2019]

Proposed methods

- Leverage external frame-level alignments extracted from the hybrid ASR system

- Investigate where to apply alignment information to streaming encoder-decoder model
  - ➢Encoder side
    1. Multi-task learning with frame-wise CE
    2. Pre-training with frame-wise CE
  - ➢Decoder side
    3. Delay constrained training (DeCoT)
    4. Minimum latency training (MinLT)

Overview



Leveraging word alignments extracted from the hybrid system

1. Multi-task learning w/ framewise CE (MTL-CE)

Train both branches
<u>from scratch</u>

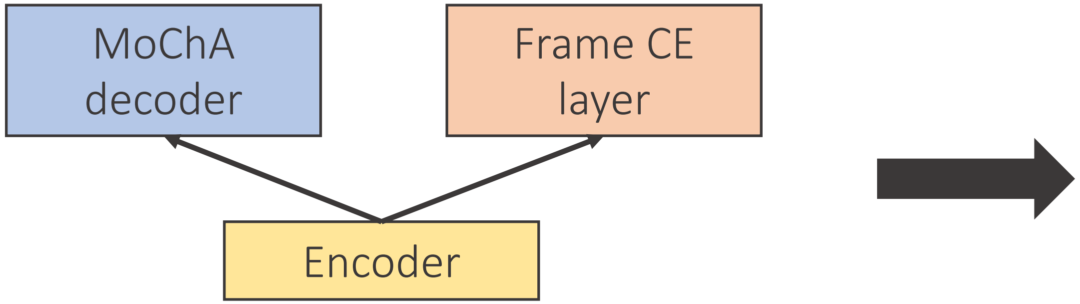
◆Objective function

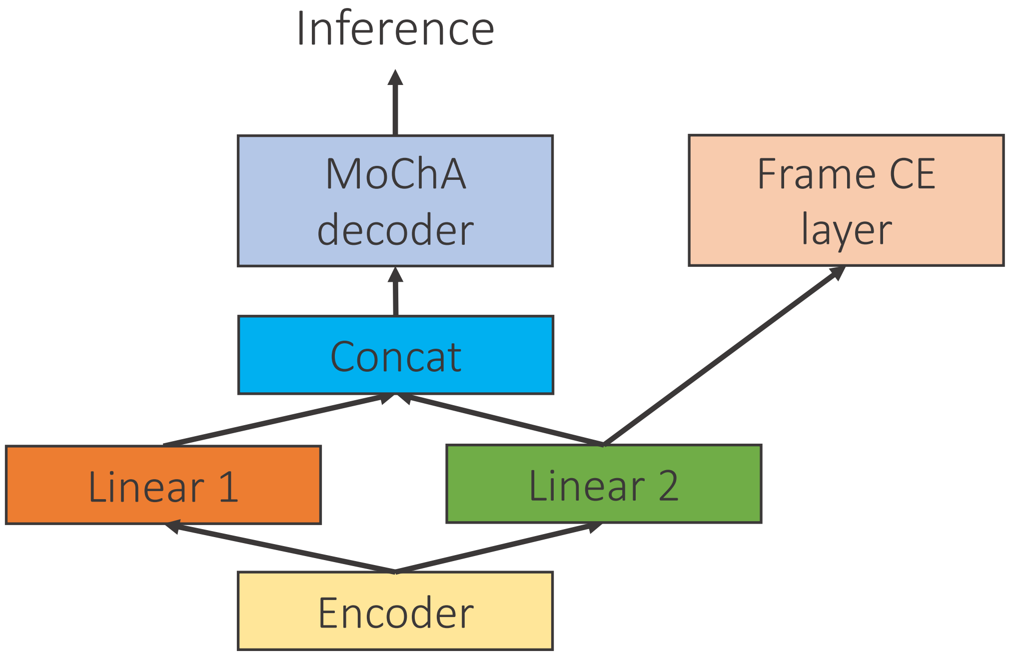
$$\mathcal{L}_{\mathrm{total}} = (1-\lambda_{\mathrm{CE}})\underbrace{\mathcal{L}_{\mathrm{S2S}}(\boldsymbol{y}|\boldsymbol{x})}_{\text{MoChA}} + \lambda_{\mathrm{CE}}\underbrace{\mathcal{L}_{\mathrm{CE}}(\boldsymbol{A}|\boldsymbol{x})}_{\text{Frame CE}} \quad (0 \leq \lambda_{\mathrm{CE}} \leq 1)$$

- Motivation: align encoder outputs to the true acoustic location

◆Insert linear bottleneck layers

- Inspired by the CTC acoustic model [Yu+ 2018]

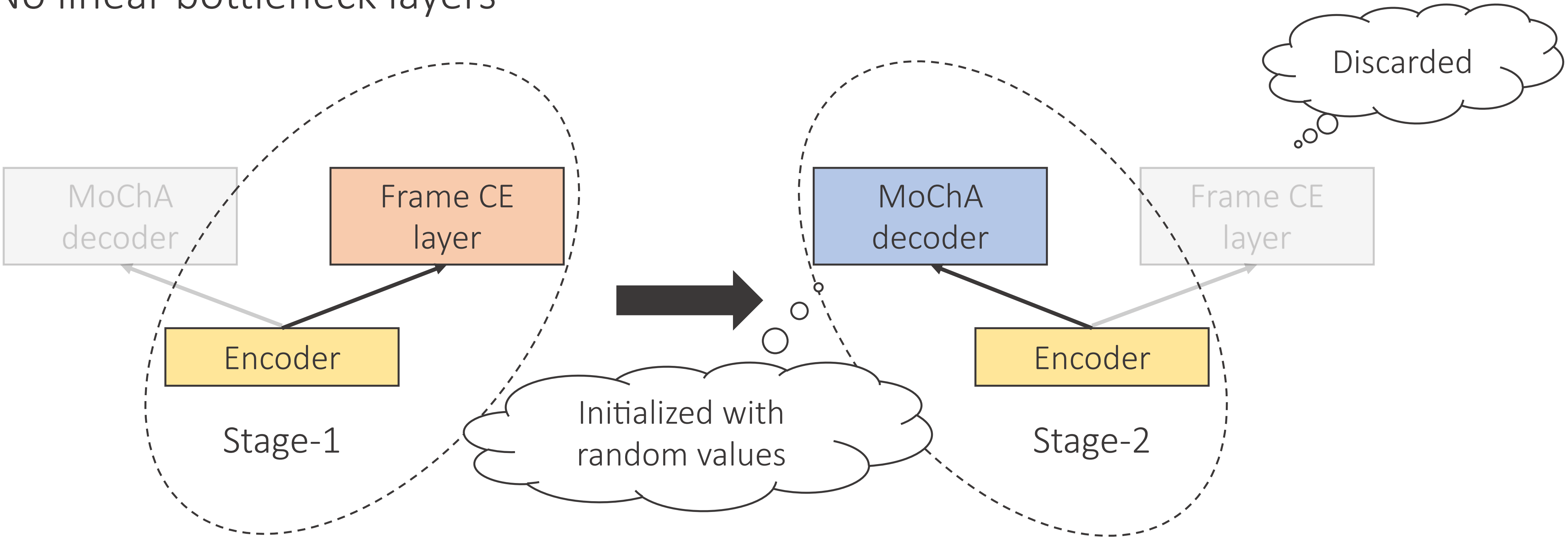




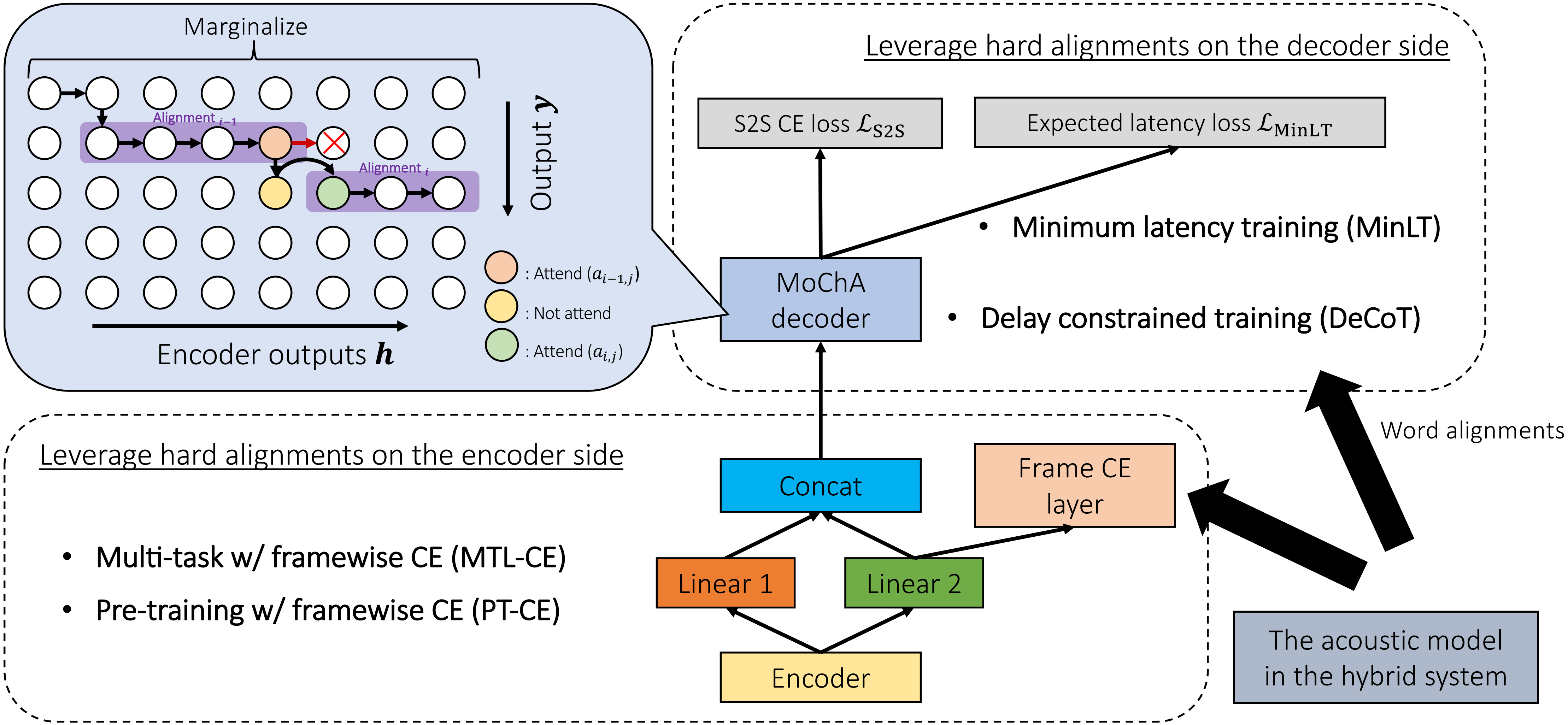
2. Pre-training with framewise CE (PT-CE)

◆ 2-staged training

- Motivation
  - ➢ Start training from well-aligned encoder representations
  - ➢ Do not have to tune the framewise CE weight λ_{CE}
- No linear bottleneck layers

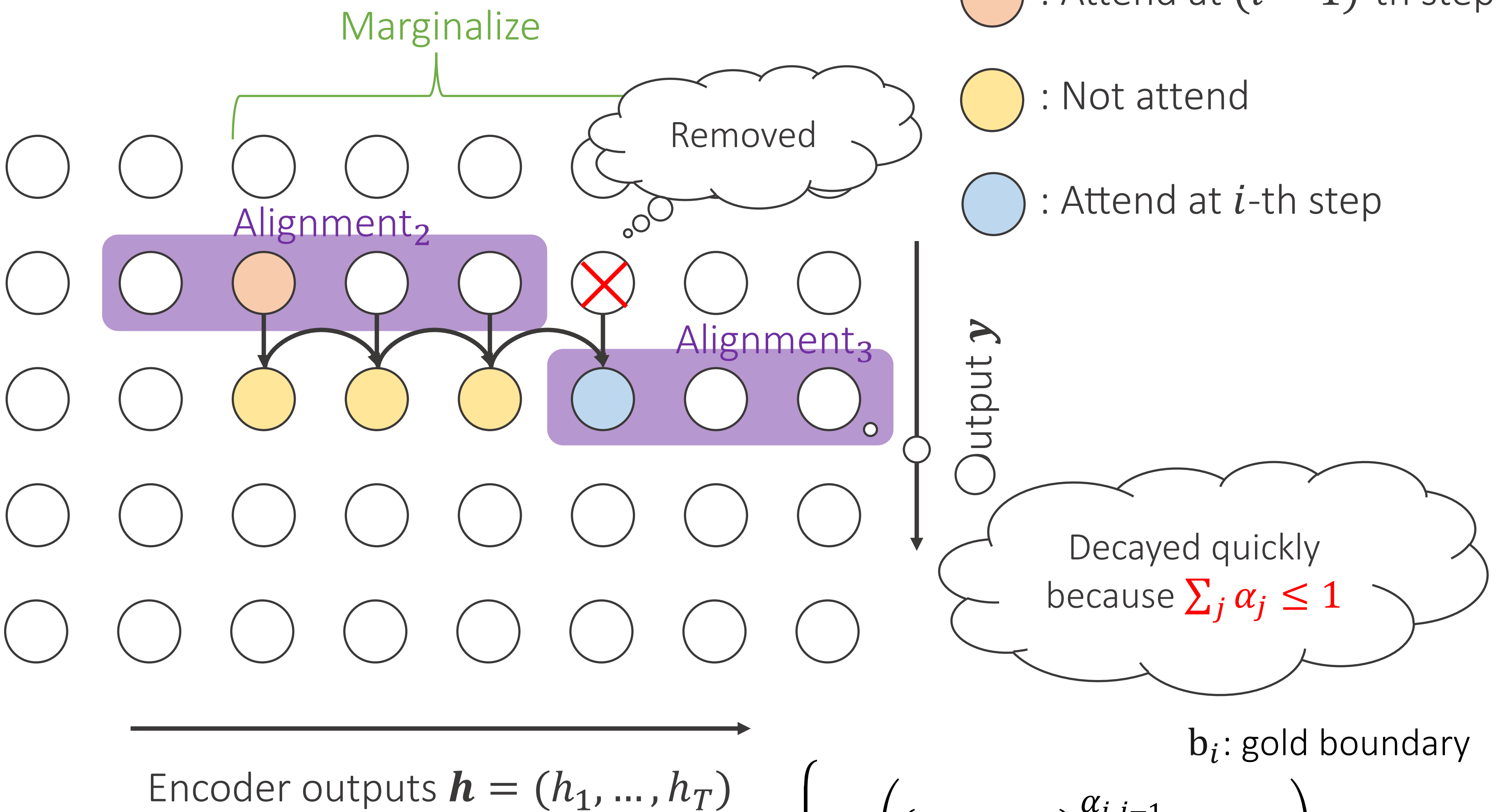


Overview



Marginalize
Alignment $_{i-1}$
Alignment $_i$
Output $\boldsymbol{y}$
: Attend $(a_{i-1,j})$
: Not attend
: Attend $(a_{i,j})$
Encoder outputs $\boldsymbol{h}$
Leverage hard alignments on the decoder side
S2S CE loss $\mathcal{L}_{\mathrm{S2S}}$
Expected latency loss $\mathcal{L}_{\mathrm{MinLT}}$
• Minimum latency training (MinLT)
MoChA decoder
• Delay constrained training (DeCoT)
Word alignments
Leverage hard alignments on the encoder side
Concat
Frame CE layer
• Multi-task w/ framewise CE (MTL-CE)
• Pre-training w/ framewise CE (PT-CE)
Linear 1
Linear 2
Encoder
The acoustic model in the hybrid system

3. Delay constrained training (DeCoT)



b_i: gold boundary

$$\alpha_{i,j} = \begin{cases} p_{i,j}\left(\left(1 - p_{i,j-1}\right)\dfrac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}\right) & (j \leq \mathrm{b}_i + \delta) \\ 0 & (\text{otherwise}) \end{cases}$$

- Remove inappropriate paths whose boundaries surpass the actual acoustic boundary more than a fixed acceptable latency $\boldsymbol{\delta}$ [frames]

3. Delay constrained training (DeCoT)

Quantity regularization

- Add a regularization term to keep $\sum_j \alpha_{i,j} = 1$
- Originally proposed in CIF [Dong+ 2019] with a different motivation

U: the number of tokens in the reference

$$\mathcal{L}_{\mathrm{QUA}} = |U - \sum_{i=1}^{U} \sum_{j=1}^{T} \alpha_{i,j}| \quad \text{(quantity loss)}$$

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{S2S}} + \lambda_{\mathrm{QUA}} \mathcal{L}_{\mathrm{QUA}} \ \ (\lambda_{\mathrm{QUA}} \geq 0)$$

4. Minimum latency training (MinLT)

◆Objective function

- Directly minimize the expected latency $\mathcal{L}_{\mathrm{MinLT}}$

Expected boundary

$$\mathcal{L}_{\mathrm{MinLT}} = \frac{1}{U}\sum_{i=1}^{U} \left| \sum_{j=1}^{T} j\alpha_{i,j} - \mathrm{b}_i \right|$$ (b_i: reference boundary for i-th token)

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{S2S}} + \lambda_{\mathrm{MinLT}}\mathcal{L}_{\mathrm{MinLT}} \quad (\lambda_{\mathrm{MinLT}} \geq 0)$$

- Motivation: reduce latency more flexibly
  - ➢DeCoT assumes <u>the fixed latency for each token</u>

◆Related work

- Latency loss has been investigated in simultaneous NMT [Arivazhagan+ 2019]
- Non-silence frames are <u>not distributed uniformly</u> over the input speech in ASR

Experimental condition

Data	Train: Microsoft Cortana voice assistant (3.4k hours) Validation: Sampled disjoint 4k utterances form the training set Test: 5.6k utterances
Feature	80-dim log-mel fbank (3 frames stacked, 30ms per frame)
Output unit	Mixed units (34k)
Architecture	Offline: 512-dim (per direction) 6-layer BiGRU encoder Streaming: 1024-dim 6-layer GRU encoder Decoder: 512-dim 2-layer GRU
Optimization	Adam
Decoding	Beam width: 8, no LM

- Word-level alignments-> subword-level alignments
  - Divide duration per word by the ratio of the character length of each subword
- Warm start training
  - Start DeCoT and MinLT from the baseline MoChA to stabilize training

Evaluation metric: Token emission latency

- Averaged time difference between <u>a predicted boundary</u> $\widehat{b_i^k}$ and <u>the gold boundary</u> b_i^k

Corpus-level latency (averaged per token)

$$\Delta_{\text{corpus}} = \frac{1}{\sum_{k=1}^{N} |\boldsymbol{y}^k|} \sum_{k=1}^{N} \sum_{i=1}^{|\boldsymbol{y}^k|} (\widehat{b_i^k} - b_i^k)$$

- Report 50-th (*TEL@50*) and 90-th percentile (*TEL@90*)
- Perform teacher-forcing when calculating latency to match the sequence lengths

Results: Alignments on the **encoder** side

Model	WER [%] (↓)	Corpus-level latency [ms] (↓)	
		TEL@50	TEL@90
Baseline MoChA	9.93	300	642
+ MTL-CE ($\lambda_{\mathrm{CE}} = 0.1$)	10.21 5.6%	240 40%	**583**
+ MTL-CE ($\lambda_{\mathrm{CE}} = 0.3$)	10.48	180	591
+ MTL-CE ($\lambda_{\mathrm{CE}} = 0.5$)	11.11	**150**	637
+ PT-CE	12.74	210	687

- MTL-CE reduced latency in proportion to λ_{CE} while degrading WER slightly
- PT-CE also reduced latency but degraded WER too much
- Contrastive results to previous works using CTC + framewise CE objective
  - ➢MoChA is a label-synchronous model
  - ➢Frame-wise CE on the encoder is not compatible with label-wise CE on the decoder

Results: Alignments on the **decoder** side

Model	WER [%] (↓)	Corpus-level latency [ms] (↓)	
		TEL@50	TEL@90
Global attention (offline)	8.44	N/A	N/A
Baseline MoChA	9.93	300	642
+ DeCoT ($\delta = 4$, 120ms)	20.25	30	287
+ DeCoT ($\delta = 8$, 240ms)	14.35	150	210
+ DeCoT ($\delta = 12$, 360ms)	11.40	210	298
+ DeCoT ($\delta = 16$, 480ms)	9.13	240	352
+ DeCoT ($\delta = 24$, 720ms)	**8.87**	270	434
+ DeCoT ($\delta = 32$, 960ms)	9.17	300	497
+ MinLT	9.70	**180**	**319**
+ DeCoT ($\delta = 16$)	12.75	120	239

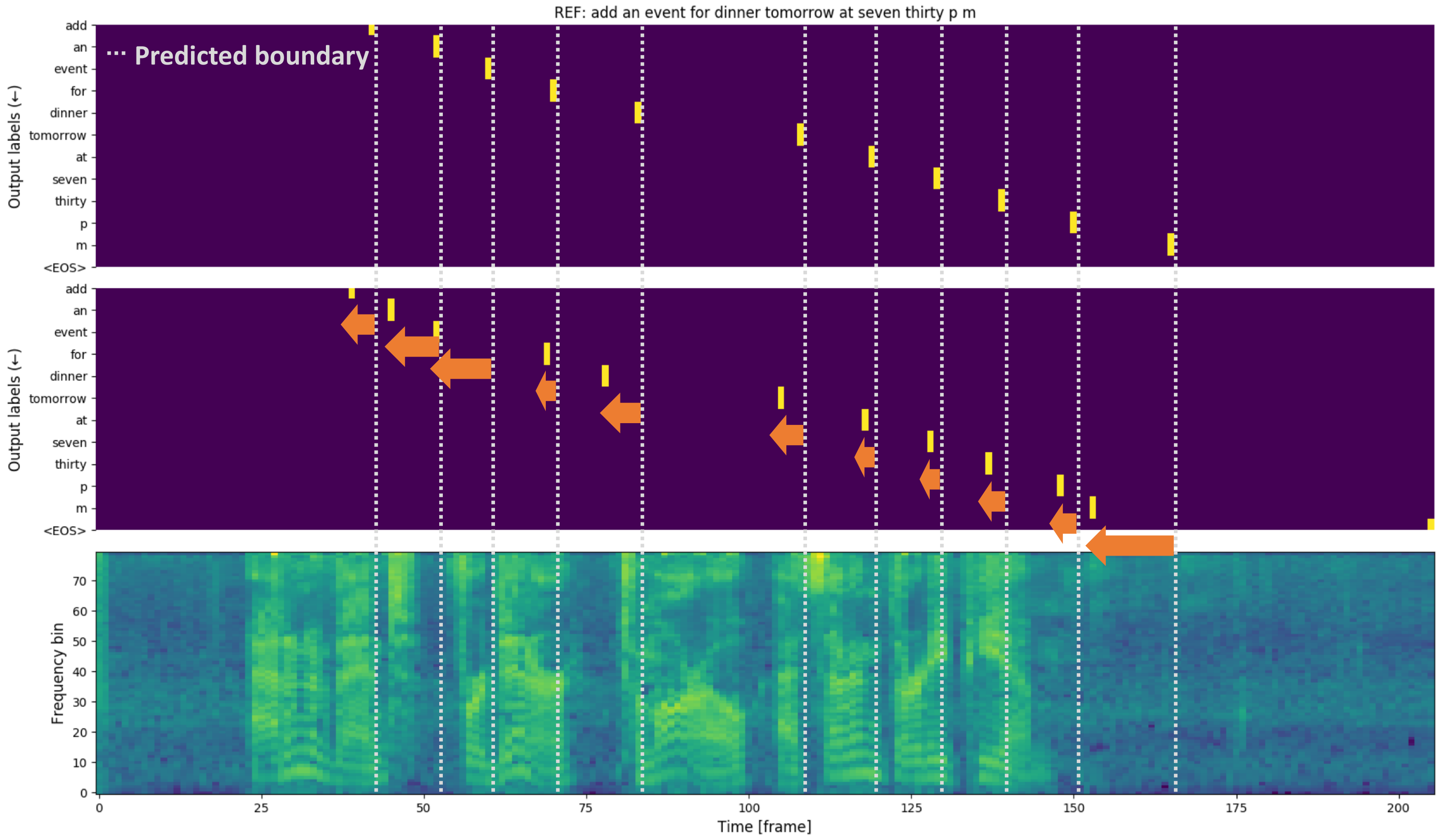
8.0%

40%

62.9%

- DeCoT: large WER reduction and moderate latency reduction (tail part)
- MinLT: small WER reduction and large latency reduction (entire)
- Combination of DeCoT and MinLT reduced latency further, but degraded WER too much

Alignment visualization



Baseline MoChA

DeCoT ($\delta = 16$)

Summary: alignment information from hybrid ASR

- Alignment information is beneficial when applying it **on the decoder side**
  - 😰 This is NOT purely end-to-end

- Can we remove the dependency to hybrid ASR system for alignment extraction?
  - ➢ CTC alignment

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

Related work: Joint CTC-attention [Kim+ 2017]

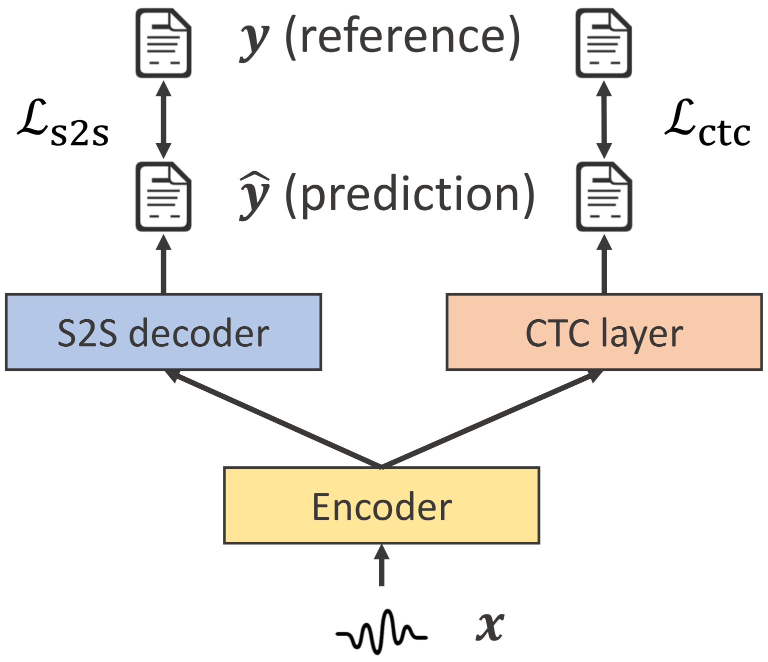
- Auxiliary CTC loss encourages the monotonicity between input and output alignments

Objective function of encoder-decoder model

$$\mathcal{L}_{s2s} = -logP(y|x) = -\sum_{i=1}^{U} logP(y_i|y_{<i}, x)$$

Multitask learning with CTC objective

$$\mathcal{L}_{\mathrm{total}} = (1-\lambda_{\mathrm{ctc}})\mathcal{L}_{\mathrm{s2s}} + \underbrace{\lambda_{\mathrm{ctc}}\mathcal{L}_{\mathrm{ctc}}}_{\text{CTC loss}} \quad (0 \leq \lambda_{\mathrm{ctc}} \leq 1)$$

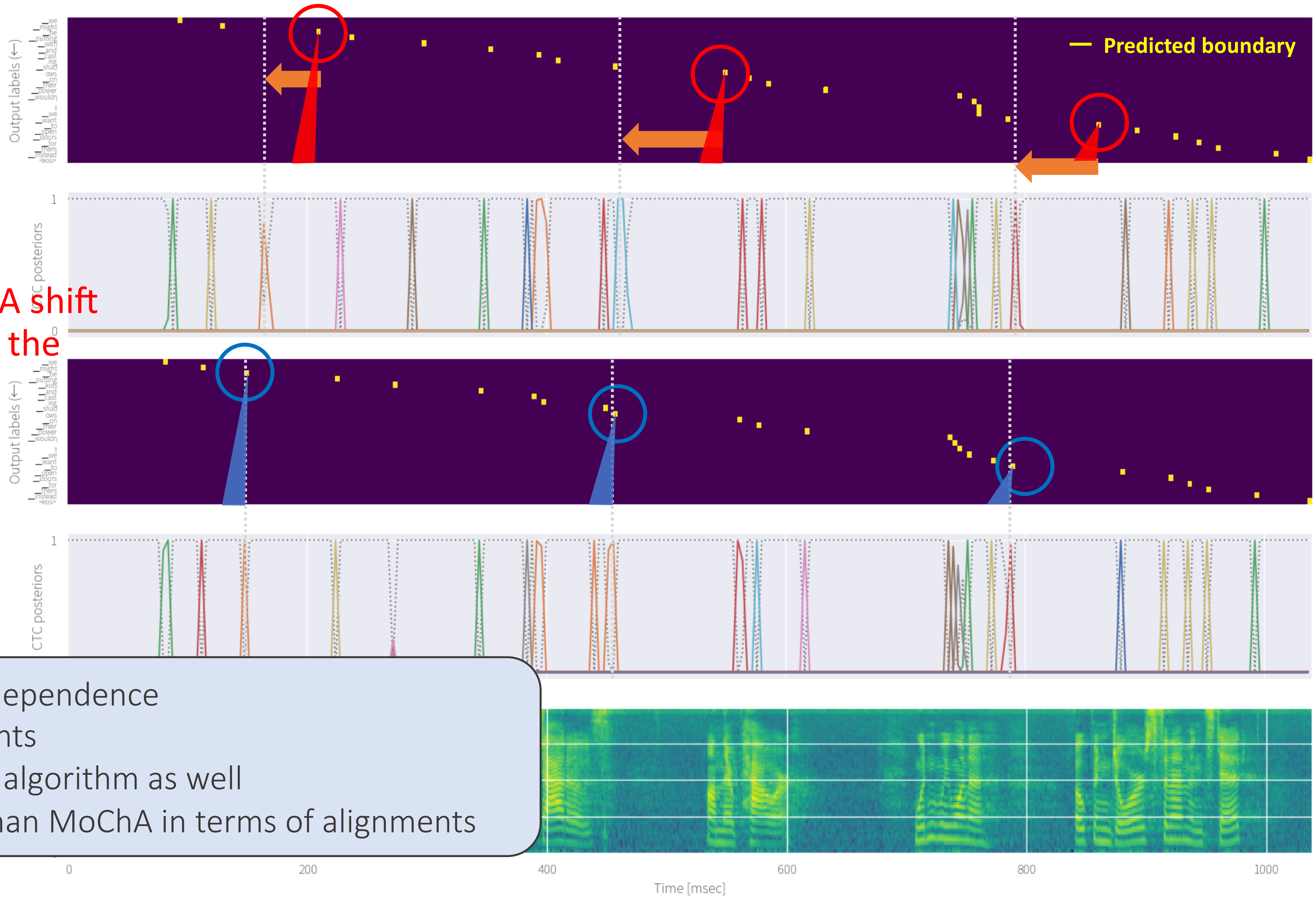


Comparison of boundary positions: CTC vs. MoChA

Baseline

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

Objective function

CTC boundary   Expected MoChA boundary

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

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

Important regularization for baseline model

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

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

Extraction of CTC alignments

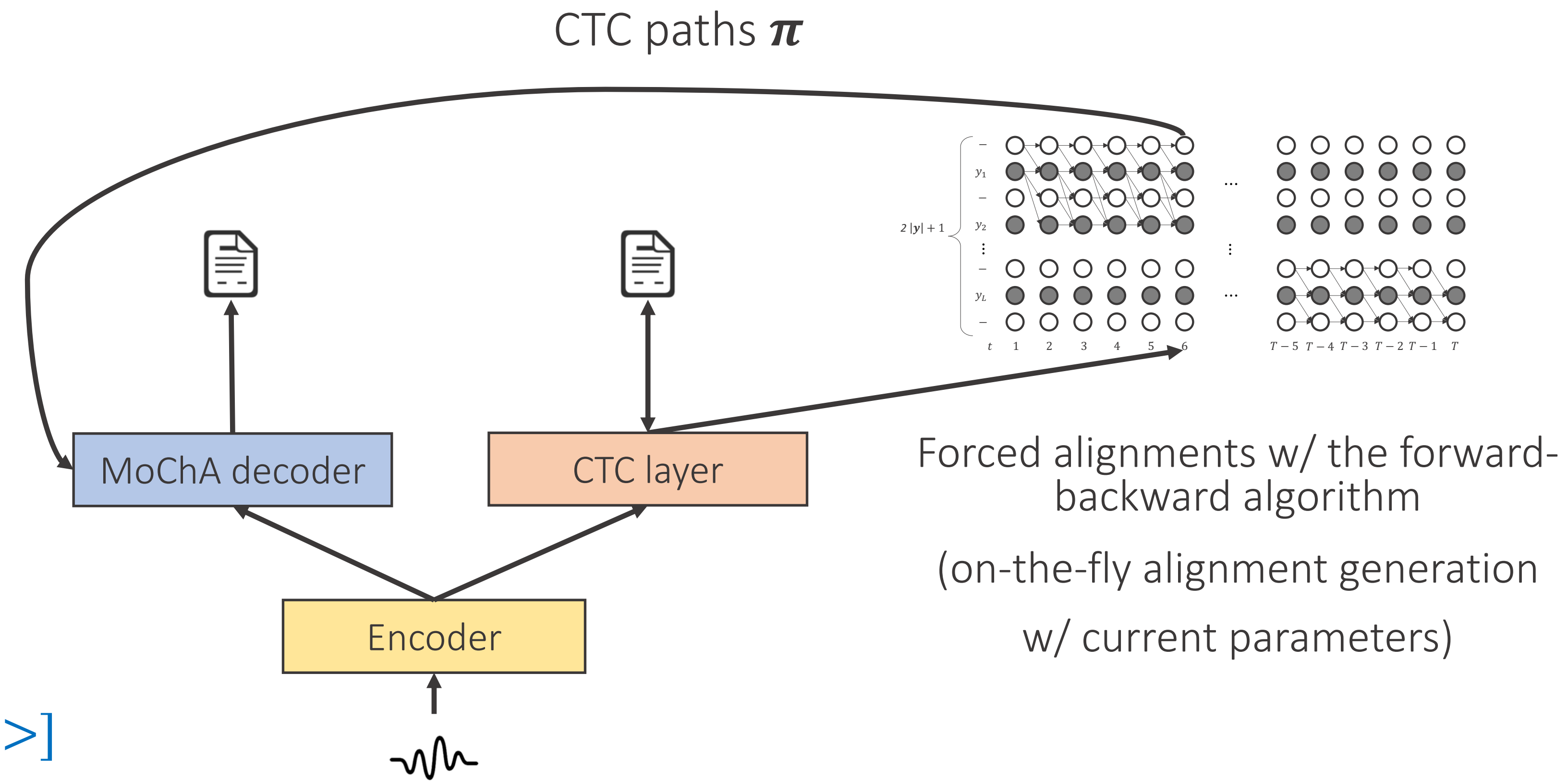
- Encoder network is shared between both branches
- Both branches are jointly optimized
- CTC alignments are extracted via forced alignment over the transcription

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 boundaries $\sum_{j=1}^{T} j\alpha_{i,j}$ accurately
  - ➢ Propose curriculum learning strategy composed of two stages

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

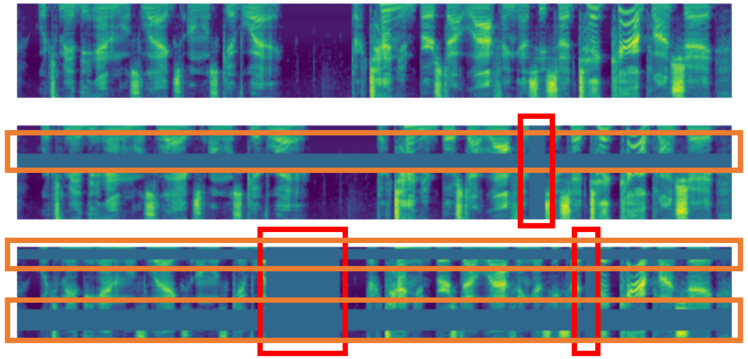
- Train BLSTM encoder + MoChA with <u>quantity regularization</u> until convergence

Stage-2 (expected to learn boundary location)

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

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

Combination with SpecAugment



SpecAugment [Park+ 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	BPE10k 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	$\lambda_{\text{ctc}} = 0.3$, $\lambda_{\text{qua}} = 1.0$, $\lambda_{\text{sync}} = 1.0$
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	BLSTM - Global attention (T1)	9.5
	BLSTM - MoChA	12.6
	+ Quantity regularization (T2)	**9.8**
	+ CTC-ST	10.2
Streaming	UniLSTM - 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

Initialization

Initialization

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 of curriculum learning

Quantity regularization CTC-ST

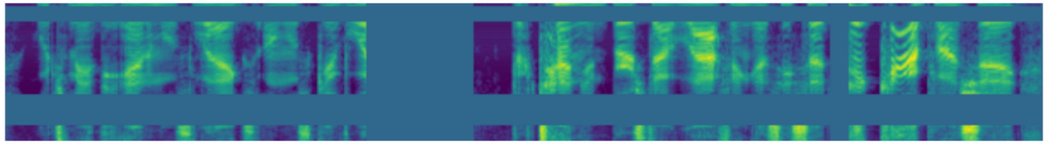
$$\mathcal{L}_{\text{total}} = (1-\lambda_{\text{ctc}})\mathcal{L}_{\text{mocha}} + \lambda_{\text{ctc}}\mathcal{L}_{\text{ctc}} + \lambda_{\text{qua}}\mathcal{L}_{\text{qua}} + \lambda_{\text{sync}}\mathcal{L}_{\text{sync}}$$

Model	Quantity regularization	CTC-ST	WER [%]
LC-BLSTM-40+40 - MoChA (from scratch)	✓	-	12.3
	-	✓	**10.9**
LC-BLSTM-40+40 - MoChA (from BLSTM - MoChA)	-	-	16.9
	✓	-	11.3
	-	✓	9.9
	✓	✓	10.1

- Seeding by BLSTM- MoChA was effective
- 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
  - ➢**Quantity regularization (stage-1)-> CTC-ST (stage-2)**

Results with SpecAugment

F



Maximum frequency mask size

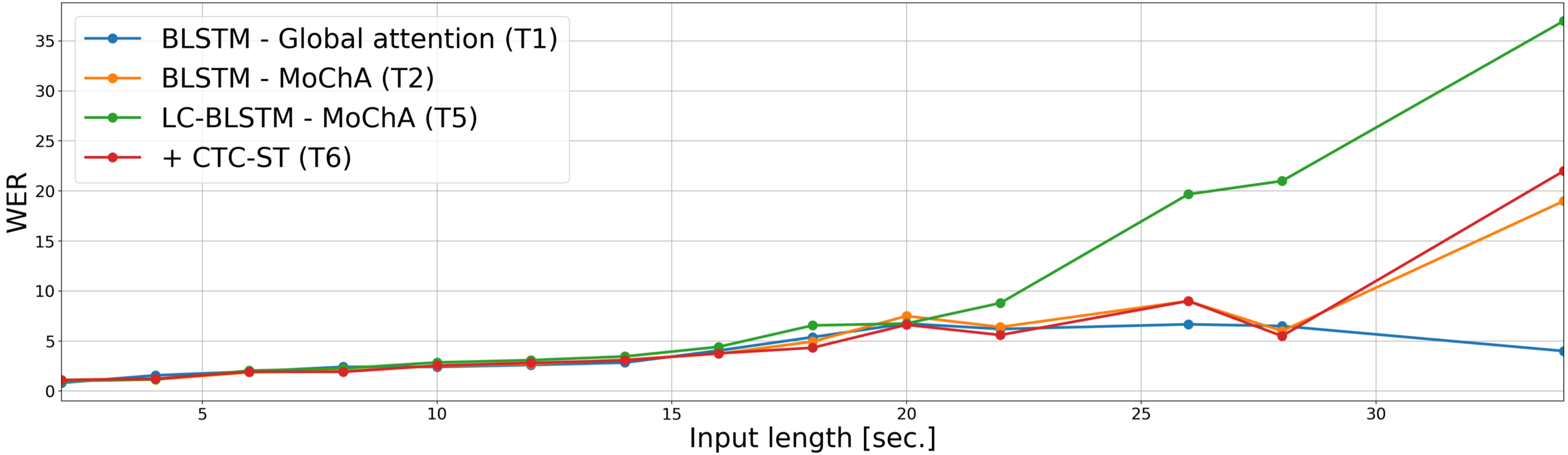
T Maximum time mask size

Model		F	T	WER [%]
Offline	Transformer [Karita+ 2019]	30	40	8.1
	BLSTM - Global attention [Zeyer+ 2019]	N/A	N/A	8.8
	BLSTM - Global 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 vs. input sequence length



- CTC-ST improved WER for long-form utterances

Results on Librispeech (960h)

	Model	WER [%] Test-clean	WER [%] Test-other
Offline	BLSTM - global 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	UniLSTM - MoChA	5.3	14.5
	+ CTC-ST	4.7	13.6
	+ SpecAugment	4.2	11.2
	LC-BLSTM-40+40 - MoChA	4.1	11.2
	+ SpecAugment ($F = 13, T = 50$)	4.0	9.5
	+ SpecAugment ($F = 27, T = 50$)	4.8	9.3
	+ SpecAugment ($F = 27, T = 100$)	5.0	9.7
	+ CTC-ST	3.9	11.2
	+ SpecAugment ($F = 13, T = 50$)	3.6	9.4
	+ SpecAugment ($F = 27, T = 50$)	3.5	9.1
	+ SpecAugment ($F = 27, T = 100$)	3.6	9.2

Initialization

Initialization

8.3/4.7% (↑)

11.3/6.2% (↑)

10.2/18.7% (↑)

Comparison with previous works on Librispeech

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

Hybrid ASR alignment vs. CTC alignment (TEDLIUM2)

† SpecAugment is used

Alignment	Model	WER [%] (↓)	Corpus-level latency [ms] (↓)	
			TEL@50	TEL@90
-	UniLSTM MoChA	15.0	280	680
CTC	+ CTC-ST	13.2	160	360
	+ CTC-ST †	11.6	200	360
Hybrid ASR	+ DeCoT ($\delta = 12$, 480ms) †	11.2	200	320
	+ DeCoT ($\delta = 16$, 640ms) †	**11.0**	**280**	**440**
	+ DeCoT ($\delta = 20$, 800ms) †	11.3	240	400
	+ DeCoT ($\delta = 24$, 960ms) †	11.7	280	480
	+ MinLT †	11.7	240	360

- CTC-ST not only improves WER but also reduces token emission latency
- CTC-ST is as good as DeCoT/MinLT for latency reduction w/o external alignment

Hybrid ASR alignment vs. CTC alignment (Librispeech)

† SpecAugment is used

Alignment	Model	WER [%] (↓)		Corpus-level latency [ms] (↓)	
		test-clean	test-other	TEL@50	TEL@90
-	UniLSTM MoChA	5.3	14.5	360	560
CTC	+ CTC-ST	4.7	13.6	240	400
	+ CTC-ST †	4.2	11.2	280	400
Hybrid ASR	+ DeCoT ($\delta = 16$, 640ms) †	4.3	11.5	320	440
	+ MinLT †	4.7	11.8	320	480

- When training data is large, CTC alignment is very accurate and reliable

Non-autoregressive End-to-end Speech Translation

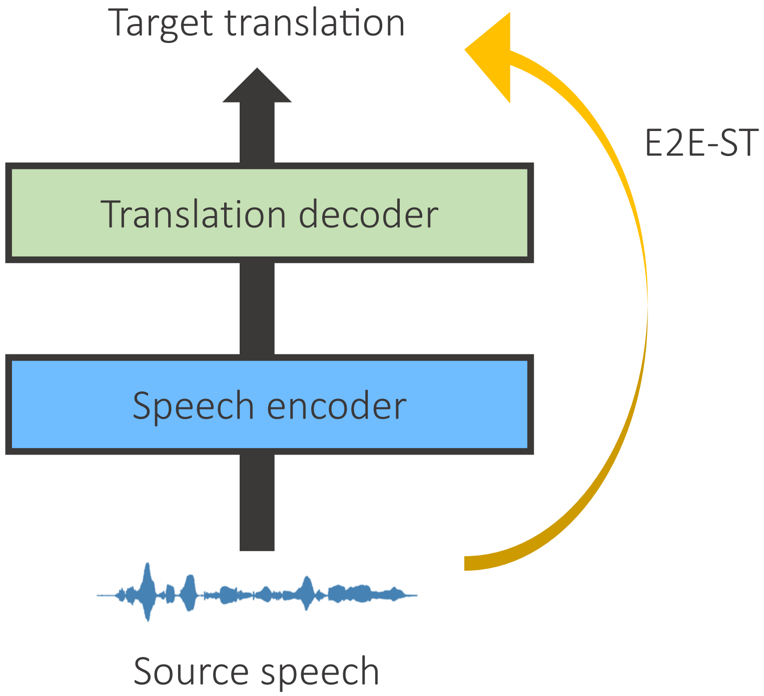
Background: End-to-end speech translation (E2E-ST)

Pros.

- Simplified architecture
- Avoid error propagation from ASR module
- Low-latency inference
- Endangered language documentation

Cons.

- Lack of supervised training data



- Most previous works focused on improving translation quality
- E2E-ST is conceptually suitable for fast decoding than cascaded systems
  - However, such evaluation has not been investigated so far

Low-latency E2E-ST

Low-latency inference

Simultaneous, streaming

Decoding speed-up, offline (this work)

Autoregressive (AR) sequence generation

◆Notation

- $X = (x_1, \dots, x_U)$ (input speech)
- $Y = (y_1, \dots, y_N)$ (target translation)
- $Y^{\mathrm{src}} = (y_1^{\mathrm{src}}, \dots, y_{N_{\mathrm{src}}}^{\mathrm{src}})$ (source transcription)

English speech

Danke (German)

Thank you (English)

◆Autoregressive decoder

➢Decompose a probability distribution of Y given X into a chain of conditional probabilities from left to right

$$P(Y|X) = \prod_{i=1}^{N} P_{\mathrm{ar}}(y_i | y_{<i}, X)$$

➢Optimized with cross-entropy loss $\mathcal{L}_{\mathrm{ar}} = -\log P_{\mathrm{ar}}(Y|X)$

➢Finish decoding after generating <eos>

Non-autoregressive (NAR) sequence generation

◆Motivation

- AR left-to-right decoding still suffers from slow inference
- Incremental decoding does not enjoy the computational power of GPU/TPU
  - ➢Toward parallel sequence generation

◆Non-autoregressive decoder [Gu+ 2018]

➢Assume conditional independence among output tokens

$$P(Y|X) = \prod_{i=1}^{N} P_{\mathrm{nar}}(y_i|X)$$

➢Predict target length in advance

e.g., Fertility model, linear classifier etc.

Modeling choice of NAR decoding

Single forward pass model (faster but less accurate)

Naïve model
- NAT [Gu+ 2018]
- NAT-REG [Wang+ 2019]
- bag-of-ngram loss [Shao+ 2020]

Latent variable model
- FlowSeq [Ma+ 2019]
- Delta posterior [Shu+ 2020]

Alignment model
- CTC [Libovický+ 2018]
- CRF [Sun+ 2019]

Iterative refinement model (more accurate at the cost of speed)

Insertion-based model
- Levenshtein Transformer [Gu+ 2019]
- Insertion-deletion Transformer [Ruis+ 2019]
- KERMIT [Chan+ 2019]
- InDIGO [Gu+ 2019]

Energy-based model
- ENGINE [Tu+ 2020]

Mask-based model
- Conditional masked language model (CMLM) [Ghazvininejad+ 2019]
- Semi-autoregressive training (SMART) [Ghazvininejad+ 2020]
- Aligned XE [Ghazvininejad+ 2020]
- Disentangled Context Transformer [Kasai+ 2020]
- Imputer [Saharia+ 2020]

Modeling choice in E2E-ST

- Single-pass model requires a copy of encoder output to initialize decoder input
  - 😰 Non-silence speech frames are NOT uniformly distributed over input speech
  - 😰 Using intermediate prediction from ASR sub-module (e.g., CTC) contradicts the motivation to alleviate error propagation by E2E modeling

- Iterative refinement model can flexibly trade quality and latency during inference by changing the number of iterations

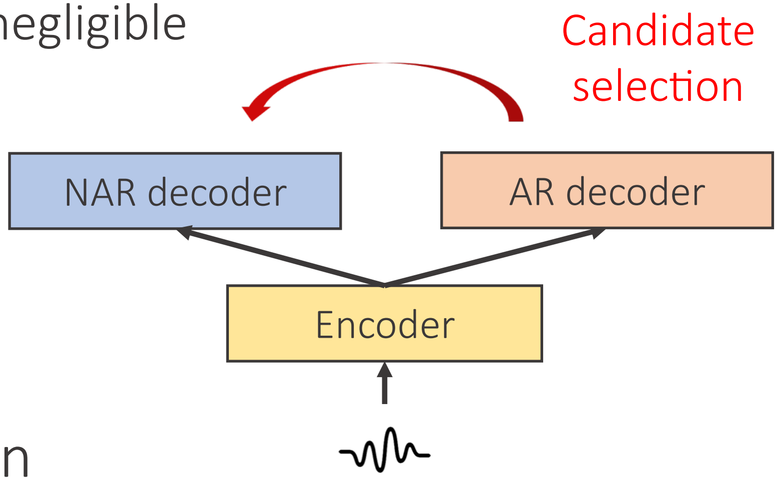
- Want to keep trainability with auxiliary tasks (ASR/MT)
  - ➢ Encoder-decoder architecture

We focus on conditional masked language model (CMLM) [Ghazvininejad+ 2019]

- ✓ Easy implementation
- ✓ Good translation performance

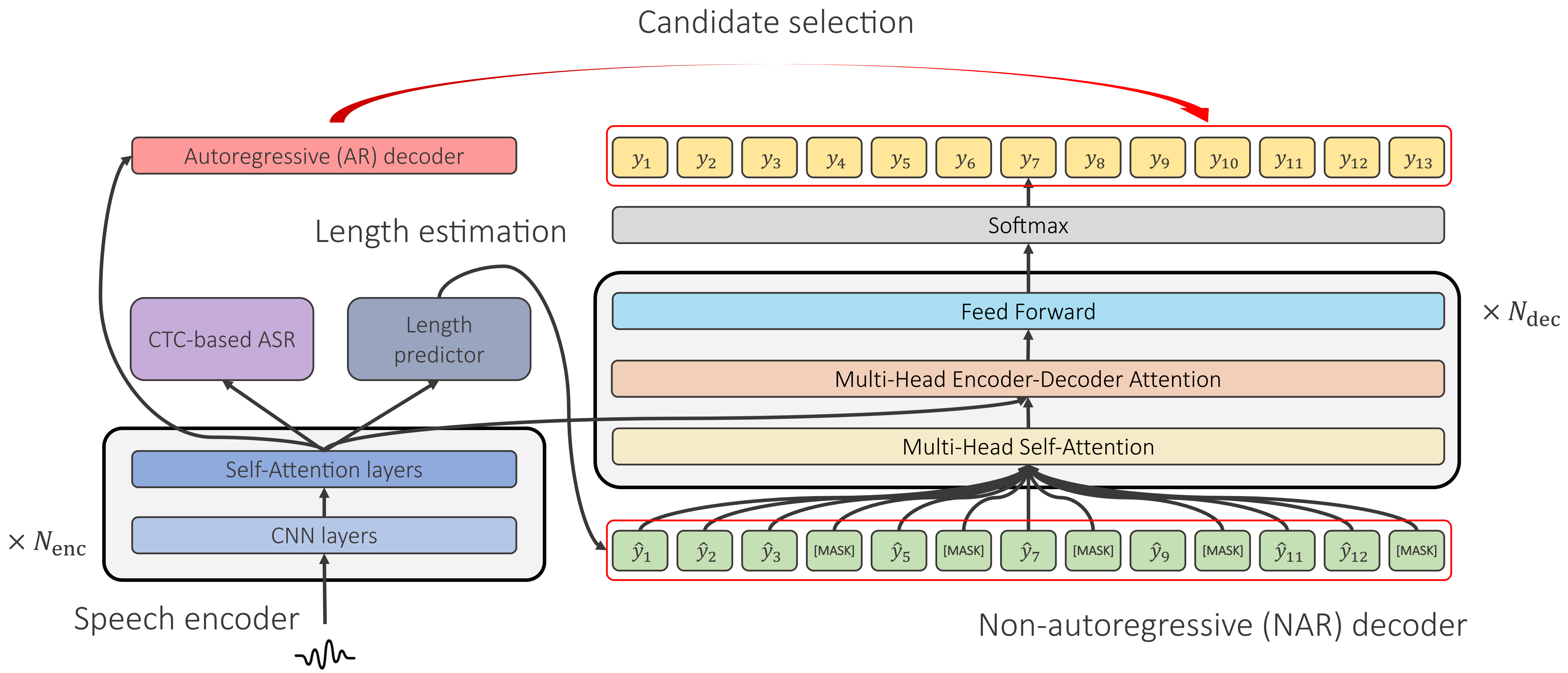
Proposed framework: Orthros

- **Challenge: target length prediction from speech**
  - Flexible sequence length: pause, speaking rate, language etc.
  - $|X| \gg |Y|$ even after downsampling
  - Rescoring multiple candidates from NAR model with separate AR model?
    - ➢ Extra computation for speech encoding by AR model is not negligible



- **Proposed framework: *Orthros***
  - ➢ AR and NAR decoders on the shared speech encoder
  - ➢ Unified architecture, trainable in an end-to-end fashion
  - ➢ Select the most probable candidate from NAR decoder by scores from AR decoder (AR decoder can generate scores in parallel)

System overview: Orthros



CMLM: inference

◆Mask-predict algorithm [Ghazvininejad+ 2019]

- Alternate two operations (*mask*, *predict*) for a constant number of iterations T
- $\hat{Y}^{(t)}_{\mathrm{mask}} \subset Y^{(t-1)}$ (masked tokens at t-th iteration, $1 \leq t \leq T$)
- $\hat{Y}^{(t)}_{\mathrm{obs}} = Y^{(t-1)} \setminus \hat{Y}^{(t)}_{\mathrm{mask}}$ (observed tokens at t-th iteration)
- Initialize $\hat{Y}^{(0)}_{\mathrm{obs}}$ with [MASK]

1. Mask operation

Predicted target length

- Mask out k_t tokens having the lowest confidence scores ($k_t = \left\lfloor \hat{N} \cdot \frac{T-t}{t} \right\rfloor$)

2. Predict operation

- Take the most probable token at every masked position i and update $y_i^{(t)} \leftarrow y_i^{(t-1)}$

$$y_i^{(t)} = \operatorname*{argmax}_{w_i \in V} P_{\mathrm{cmlm}}(w_i | \hat{Y}^{(t)}_{\mathrm{obs}}, X)$$
$$p_i^{(t)} \leftarrow P_{\mathrm{cmlm}}(y_i^{(t)} | \hat{Y}^{(t)}_{\mathrm{obs}}, X)$$

CMLM: inference

◆Target length prediction

- Take top-l sequence lengths from length distribution P_{lp}

◆Length parallel decoding

- Predict multiple l sequences having different lengths in parallel
  - ➢In actual implementation, perform batch-decoding, i.e., input/output tensor size: $[l, \widehat{N}_{\mathrm{max}}]$
- Select the most probable sequence at the last iteration among l candidates

$$score = \frac{1}{\widehat{N}} \sum_{i=1}^{\widehat{N}} \log P_{\mathrm{i,cmlm}}^{(T)}$$

CMLM: training

◆Notation

- $Y_{\mathrm{mask}} \subset Y$ (masked tokens in ground-truth Y)
- $Y_{\mathrm{obs}} \subset Y \setminus Y_{\mathrm{mask}}$ (observed tokens in Y)

◆Training objective

- The number of masked tokens is sampled from uniform distribution $\mathcal{U}(1, N)$

$$\mathcal{L}_{\mathrm{cmlm}} = -\sum_{y \in Y_{mask}} \log P_{\mathrm{cmlm}}(y | Y_{\mathrm{obs}}, X)$$

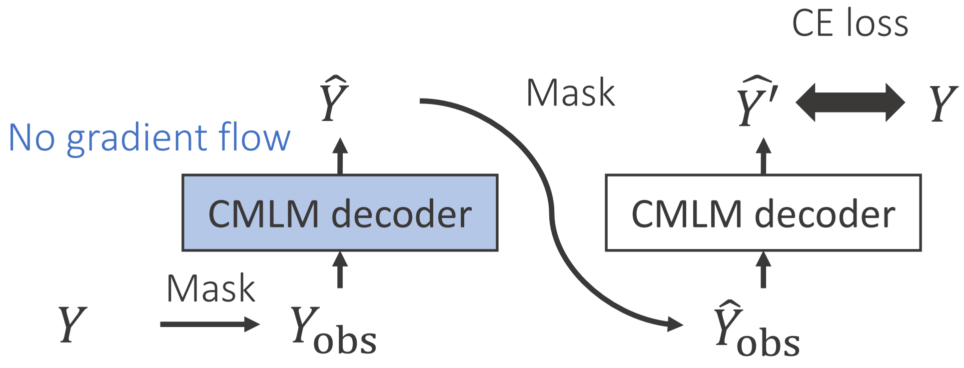
Semi-autoregressive training (SMART) [Ghazvininejad+ 2020]

- Bridge the gap between training and test conditions by feeding output from the model to the CMLM decoder

◆Procedure

1. Obtain prediction at all positions ($\hat{Y}$) from the current model by feeding $Y_{\mathbf{obs}}$
2. Obtain new decoder input $\hat{Y}_{\mathbf{obs}}$ by applying random mask to $\hat{Y}$
3. Train model to predict Y given $\hat{Y}_{\mathbf{obs}}$

Unlike original CMLM, cross-entropy loss is calculated at all position regardless of mask



◆Training objective

$$\mathcal{L}_{\mathbf{cmlm}} = -\sum_{y \in \hat{Y}} \log P_{\mathbf{cmlm}}(y \mid \hat{Y}_{\mathbf{obs}}, X)$$

Orthros: training

◆Training objective

$$\mathcal{L}_{\text{total}} = (1-\lambda_{\text{cmlm}})\underbrace{\mathcal{L}_{\text{cmlm}}(Y|X)}_{\text{NAR decoder}} + \lambda_{\text{ar}}\underbrace{\mathcal{L}_{\text{ar}}(Y|X)}_{\text{AR decoder}}$$

$$+\lambda_{\text{lp}}\underbrace{\mathcal{L}_{\text{lp}}(N|X)}_{\text{Length prediction}} + \lambda_{\text{asr}}\underbrace{\mathcal{L}_{\text{asr}}(Y^{\text{src}}|X)}_{\text{ASR}}$$

- Length prediction: $\mathcal{L}_{\text{lp}}(N|X) = -\log P_{\text{lp}}(N|X)$
- ASR (CTC): $\mathcal{L}_{\text{asr}}(Y^{\text{src}}|X) = -\log P_{\text{ctc}}(Y^{\text{src}}|X)$

Orthros: inference

1. Mask-predict for T iterations

2. Candidate selection with AR decoder
    - After the last iteration, feed outputs from the NAR decoder to the AR decoder in parallel
    - Obtain sequence-level scores from the AR decoder
    - Pick up the most probable candidate among l candidates

$$score = \frac{1}{\widehat{N}} \sum_{i=1}^{\widehat{N}} \log P_{i,\mathrm{ar}}$$

Experimental setting

◆Datasets

- Must-C En-De (229k pairs, 408h), En-Fr (275k pairs, 492h)
- Fisher-CallHome Spanish (Es->En, 138k pairs, 170h)
- Libri-trans (En->Fr, 45k pairs, 100h)

◆Model configuration

- Implemented with ESPnet-ST [Inaguma+ 2020]
- Transformer base/large ($d_{\mathrm{model}} = 256/512, d_{\mathrm{ff}} = 2048, H = 4/8$)
- 2-layers CNN->12-layers encoder, 6-layers decoder
- Sequence-level knowledge distillation (Seq-KD) [Kim+ 2016] from text-based AR MT model
- Vocabulary size
  - ➢ AR: BPE8k (Must-C), 1k (Fisher-CallHome, Libri-trans)
  - ➢ NAR: BPE8k

Evaluation metric

◆ Translation quality

- 4-gram BLEU

◆ Inference speed

- GPU decoding with a NVIDIA TITAN RTX
- Decoding configuration
  - ✓ AR: beam width $b \in \{1,4\}$
  - ✓ NAR: iteration $T \in \{4,10\}$, length beam width $l = 9$
  - ✓ Batch size: 1
- Averaged over 5 runs

Main results: Must-C En-De/En-Fr

Model		En-De BLEU	En-De Latency [ms]	En-De Speedup	En-Fr BLEU
Autoregressive	Transformer (b=1)	21.54	175ms	1.54×	32.26
	Transformer (b=4)	23.12	271ms	1.00×	33.84
	Transformer + Seq-KD (b=1)	23.88	-	-	33.92
	Transformer + Seq-KD (b=4)	24.43	-	-	34.57
Non-autoregressive	CTC (b=1)	19.40	13ms	20.84×	27.38
	Orthros (CMLM, T=4)	18.78	-	-	25.99
	Orthros (CMLM, T=10+AR decoder)	19.62	-	-	27.77
	Orthros (CMLM T=10)	20.89	-	-	28.74
	Orthros (CMLM, T=10+AR decoder)	21.79	-	-	30.31
	Orthros (SMART, T=4)	20.03	46	5.89×	27.22
	Orthros (SMART, T=10+AR decoder)	**21.08**	**61**	**4.44×**	**29.30**
	Orthros (SMART, T=10)	21.25	99	2.73×	29.31
	Orthros (SMART, T=10+AR decoder)	**22.27**	**117**	**2.44×**	**31.07**
	+ BPE8k -> BPE16k	**22.88**	117	2.31×	**32.20**
	+ large (SMART, T=4+AR decoder, l=7)	22.54	59	4.59×	31.24
	+ large (SMART, T=10+AR decoder, l=7)	23.92	113	2.39×	33.05

Semi-autoregressive training (SMART)
- Improved BLEU significantly with no extra latency during inference

n-De/En-Fr

		En-De			En-Fr
		BLEU	Latency [ms]	Speedup	BLEU
Autoregressive	Transformer (b=1)	21.54	175ms	1.54×	32.26
	Transformer (b=4)	23.12	271ms	1.00×	33.84
	Transformer + Seq-KD (b=1)	23.88	-	-	33.92
	Transformer + Seq-KD (b=4)	24.43	-	-	34.57
Non-autoregressive	CTC (b=1)	19.40	13ms	20.84×	27.38
	Orthros (CMLM, T=4)	18.78	-	-	25.99
	Orthros (CMLM, T=10+AR decoder)	19.62	-	-	27.77
	Orthros (CMLM T=10)	20.89	-	-	28.74
	Orthros (CMLM, T=10+AR decoder)	21.79	-	-	30.31
	Orthros (SMART, T=4)	20.03	46	5.89×	27.22
	Orthros (SMART, T=10+AR decoder)	**21.08**	**61**	**4.44×**	**29.30**
	Orthros (SMART, T=10)	21.25	99	2.73×	29.31
	Orthros (SMART, T=10+AR decoder)	**22.27**	**117**	**2.44×**	**31.07**
	+ BPE8k -> BPE16k	**22.88**	117	2.31×	**32.20**
	+ large (SMART, T=4+AR decoder, l=7)	22.54	59	4.59×	31.24
	+ large (SMART, T=10+AR decoder, l=7)	23.92	113	2.39×	33.05

Candidates selection with AR decoder

- Improved BLEU scores is significantly
- This corresponds to performing one more iteration (about +15ms)
- CMLM does not have the ability to generate useful sentence-level scores

n-Fr

		En-De			En-Fr
		EU	Latency [ms]	Speedup	BLEU
		54	175ms	1.54×	32.26
		.12	271ms	1.00×	33.84
Autoregressive	Transformer + Seq-KD (b=1)	23.88	-	-	33.92
	Transformer + Seq-KD (b=4)	24.43	-	-	34.57
Non-autoregressive	CTC (b=1)	19.40	13ms	20.84×	27.38
	Orthros (CMLM, T=4)	18.78	-	-	25.99
	Orthros (CMLM, T=10+AR decoder)	19.62	-	-	27.77
	Orthros (CMLM T=10)	20.89	-	-	28.74
	Orthros (CMLM, T=10+AR decoder)	21.79	-	-	30.31
	Orthros (SMART, T=4)	20.03	46	5.89×	27.22
	Orthros (SMART, T=10+AR decoder)	**21.08**	**61**	**4.44×**	**29.30**
	Orthros (SMART, T=10)	21.25	99	2.73×	29.31
	Orthros (SMART, T=10+AR decoder)	**22.27**	**117**	**2.44×**	**31.07**
	+ BPE8k -> BPE16k	**22.88**	117	2.31×	**32.20**
	+ large (SMART, T=4+AR decoder, l=7)	22.54	59	4.59×	31.24
	+ large (SMART, T=10+AR decoder, l=7)	23.92	113	2.39×	33.05

Vocabulary size
- Large BPE vocabulary improved BLEU scores
- This was not true for AR models (shown in the later slide)

n-De/En-Fr

		En-De			En-Fr
		BLEU	Latency [ms]	Speedup	BLEU
Autoregressive	Transformer (b=1)	21.54	175ms	1.54×	32.26
	Transformer (b=4)	23.12	271ms	1.00×	33.84
	Transformer + Seq-KD (b=1)	23.88	-	-	33.92
	Transformer + Seq-KD (b=4)	24.43	-	-	34.57
Non-autoregressive	CTC (b=1)	19.40	13ms	20.84×	27.38
	Orthros (CMLM, T=4)	18.78	-	-	25.99
	Orthros (CMLM, T=10+AR decoder)	19.62	-	-	27.77
	Orthros (CMLM T=10)	20.89	-	-	28.74
	Orthros (CMLM, T=10+AR decoder)	21.79	-	-	30.31
	Orthros (SMART, T=4)	20.03	46	5.89×	27.22
	Orthros (SMART, T=10+AR decoder)	**21.08**	**61**	**4.44×**	**29.30**
	Orthros (SMART, T=10)	21.25	99	2.73×	29.31
	Orthros (SMART, T=10+AR decoder)	**22.27**	**117**	**2.44×**	**31.07**
	+ BPE8k -> BPE16k	**22.88**	117	2.31×	**32.20**
	+ large (SMART, T=4+AR decoder, l=7)	22.54	59	4.59×	31.24
	+ large (SMART, T=10+AR decoder, l=7)	23.92	113	2.39×	33.05

Large model

- Increasing model capacity was very important for NAR models
- This was not true for AR models (shown in the later slide)

n-De/En-Fr

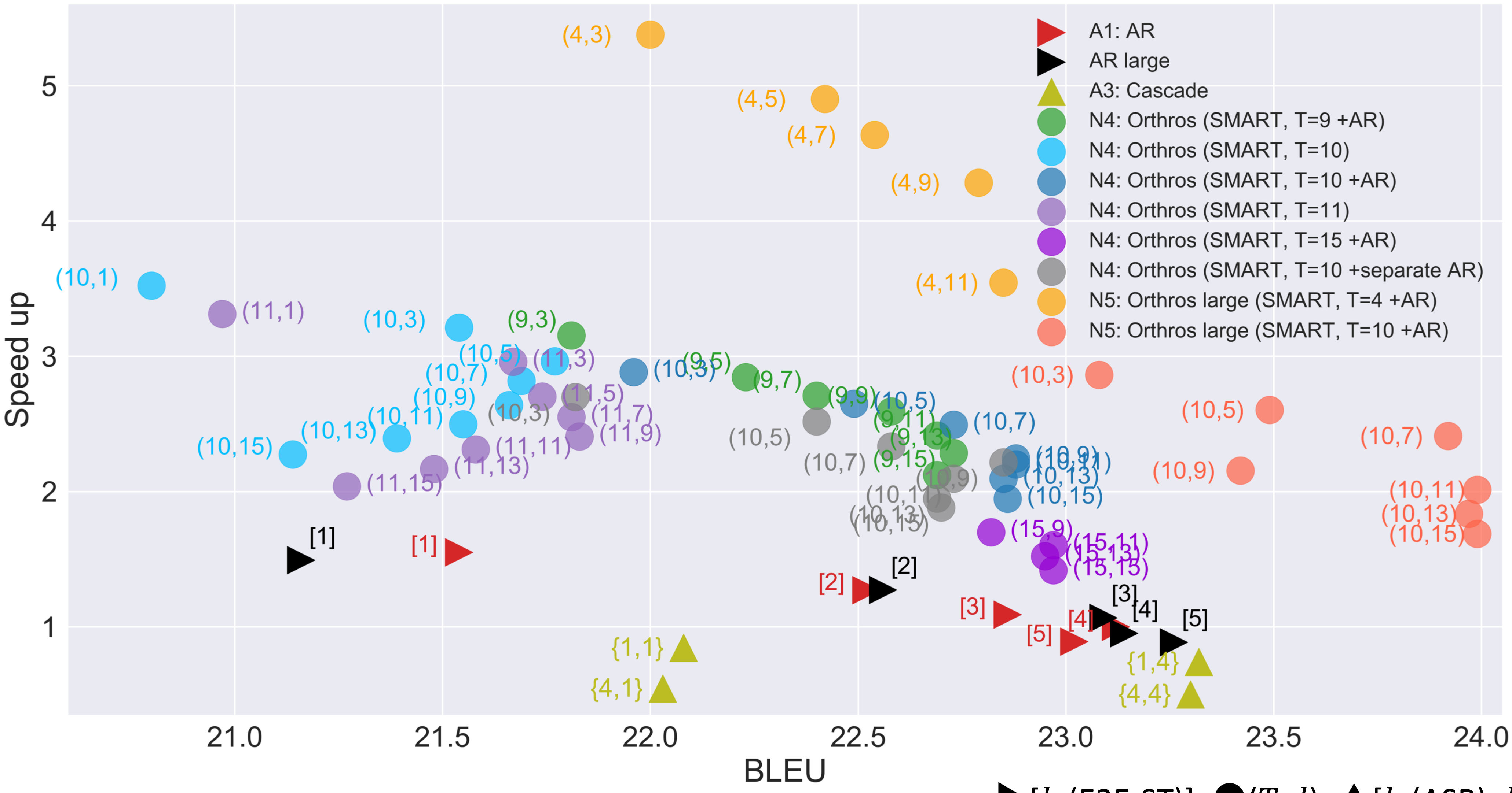
		En-De			En-Fr
		BLEU	Latency [ms]	Speedup	BLEU
		21.54	175ms	1.54×	32.26
Autoregressive		23.12	271ms	1.00×	33.84
	Transformer + Seq-KD (b=1)	23.88	-	-	33.92
	Transformer + Seq-KD (b=4)	24.43	-	-	34.57
Non-autoregressive	CTC (b=1)	19.40	13ms	20.84×	27.38
	Orthros (CMLM, T=4)	18.78	-	-	25.99
	Orthros (CMLM, T=10+AR decoder)	19.62	-	-	27.77
	Orthros (CMLM T=10)	20.89	-	-	28.74
	Orthros (CMLM, T=10+AR decoder)	21.79	-	-	30.31
	Orthros (SMART, T=4)	20.03	46	5.89×	27.22
	Orthros (SMART, T=10+AR decoder)	**21.08**	**61**	**4.44×**	**29.30**
	Orthros (SMART, T=10)	21.25	99	2.73×	29.31
	Orthros (SMART, T=10+AR decoder)	**22.27**	**117**	**2.44×**	**31.07**
	+ BPE8k -> BPE16k	**22.88**	117	2.31×	**32.20**
	+ large (SMART, T=4+AR decoder, l=7)	22.54	59	4.59×	31.24
	+ large (SMART, T=10+AR decoder, l=7)	23.92	113	2.39×	33.05

Main results: Must-C En-De/En-Fr

Model		En-De BLEU	En-De Latency [ms]	En-De Speedup	En-Fr BLEU
Autoregressive	Transformer (b=1)	21.54	175ms	1.54×	32.26
	Transformer (b=4)	23.12	271ms	1.00×	33.84
	Transformer + Seq-KD (b=1)	23.88	-	-	33.92
	Transformer + Seq-KD (b=4)	24.43	-	-	34.57
	CTC (b=1)	19.40	13ms	20.84×	27.38
		18.78	-	-	25.99
		19.62	-	-	27.77
		20.89	-	-	28.74
		21.79	-	-	30.31
		20.03	46	5.89×	27.22
		21.08	**61**	**4.44×**	**29.30**
	Orthros (SMART, T=10)	21.25	99	2.73×	29.31
	Orthros (SMART, T=10+AR decoder)	**22.27**	**117**	**2.44×**	**31.07**
	+ BPE8k -> BPE16k	**22.88**	117	2.31×	**32.20**
	+ large (SMART, T=4+AR decoder, l=7)	22.54	59	4.59×	31.24
	+ large (SMART, T=10+AR decoder, l=7)	23.92	113	2.39×	33.05

NAR vs AR
- Achieved comparable BLEU scores to baseline Transformer
- Seq-KD boosted AR model's performance further
- This differs from MT:
  - MT: large AR teacher-> small AR student
  - E2E-ST: AR MT teacher -> AR E2E-ST student

Quality and latency trade-off on Must-C En-De

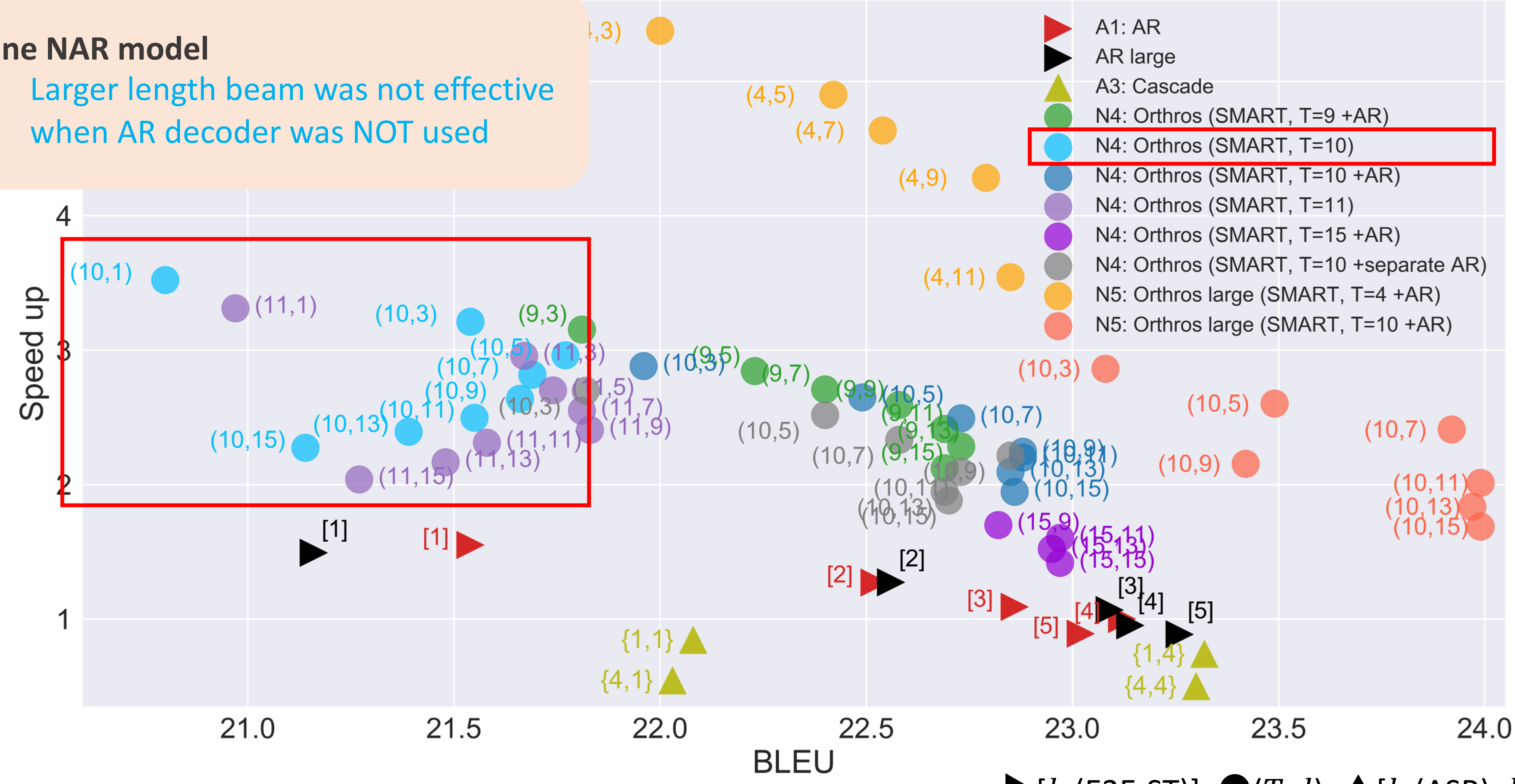


▶[b (E2E-ST)], ●(T, l), ▲[b (ASR), b (MT)]

Quality and latency trade-off on Must-C En-De

Baseline NAR model

- Larger length beam was not effective when AR decoder was NOT used

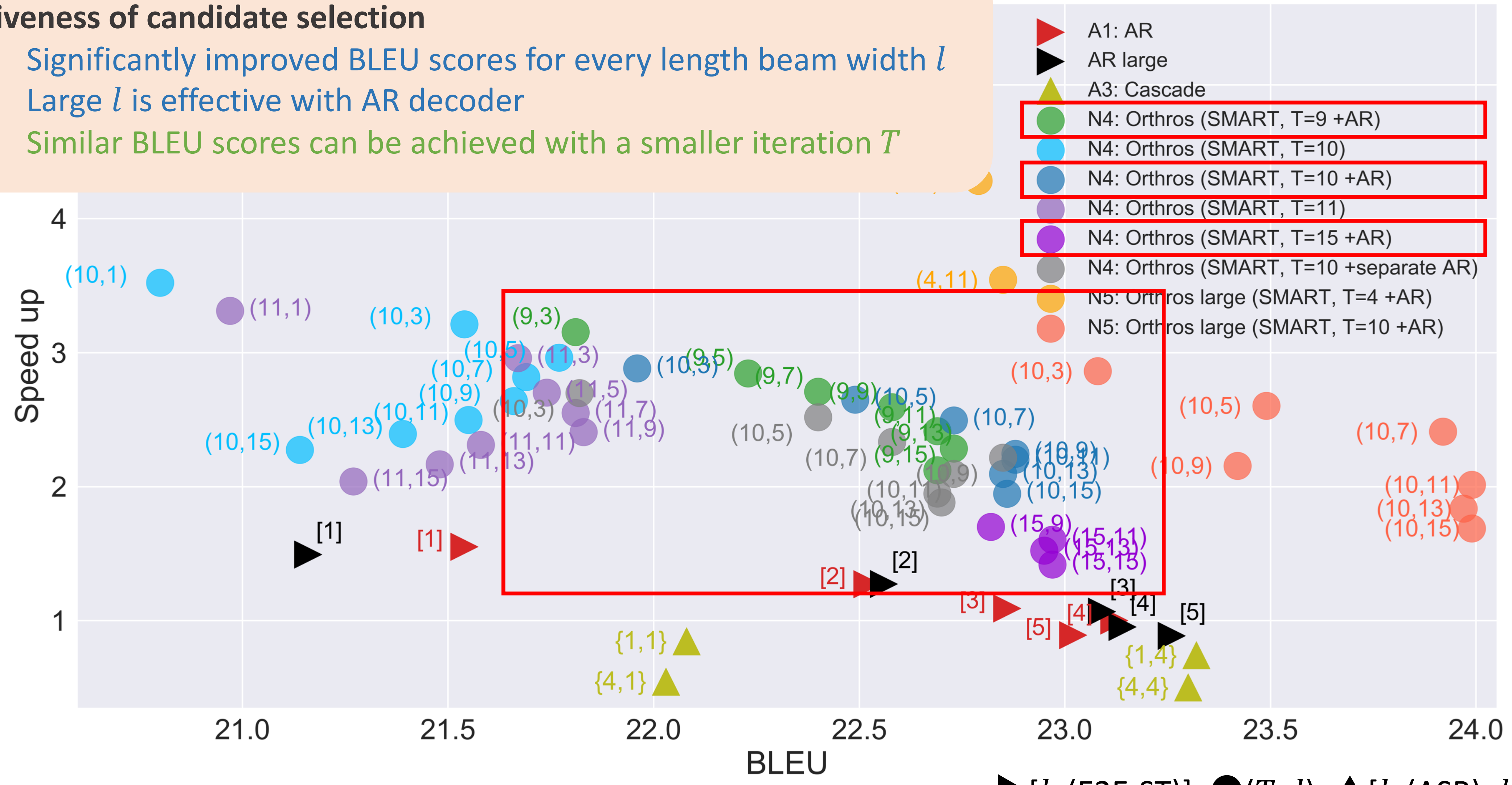


▶[b (E2E-ST)], ●(T, l), ▲[b (ASR), b (MT)]

Quality and latency trade-off on Must-C En-De

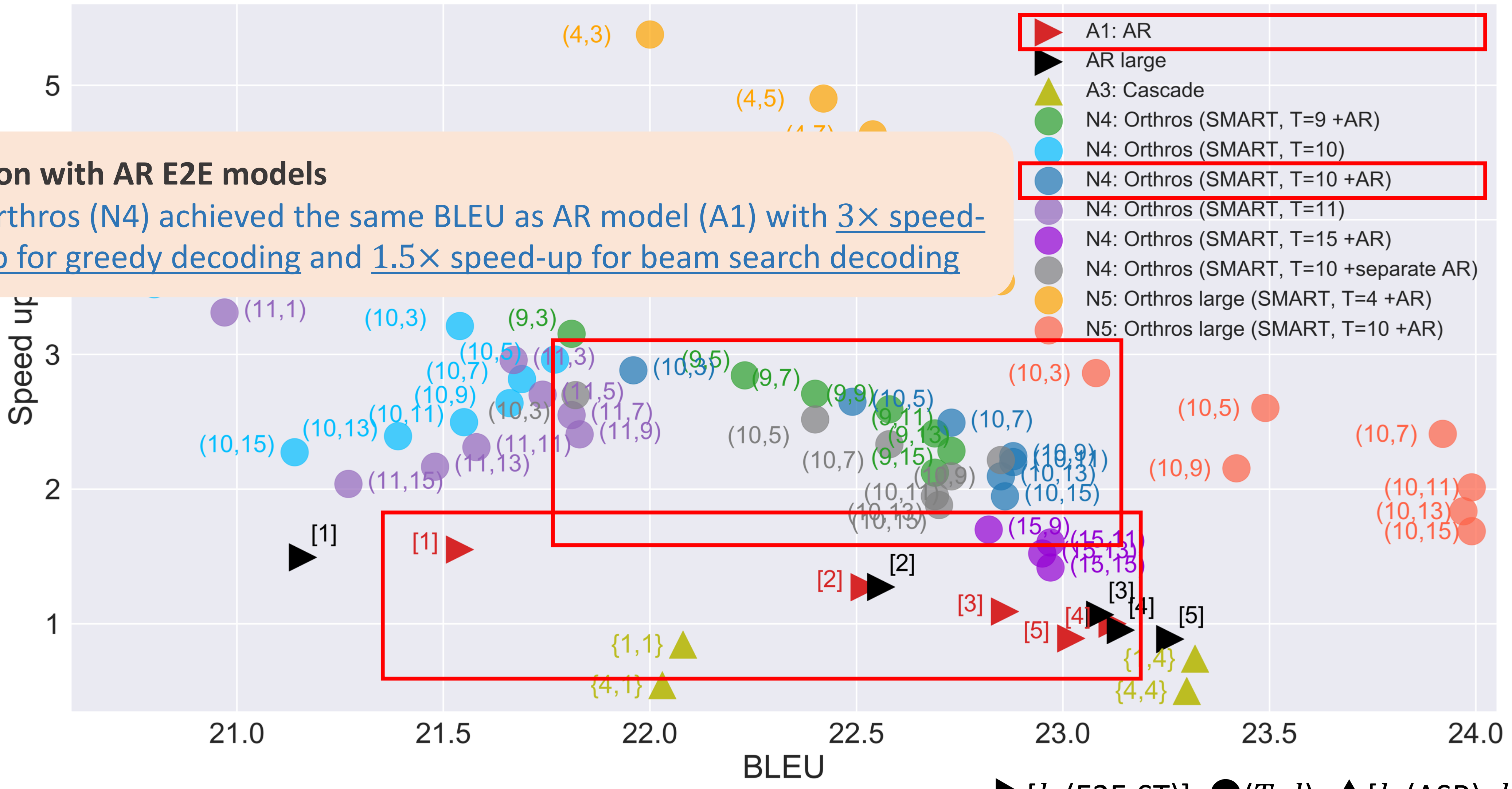
Effectiveness of candidate selection

- Significantly improved BLEU scores for every length beam width l
- Large l is effective with AR decoder
- Similar BLEU scores can be achieved with a smaller iteration T



▶[b (E2E-ST)], ●(T, l), ▲[b (ASR), b (MT)]

Quality and latency trade-off on Must-C En-De

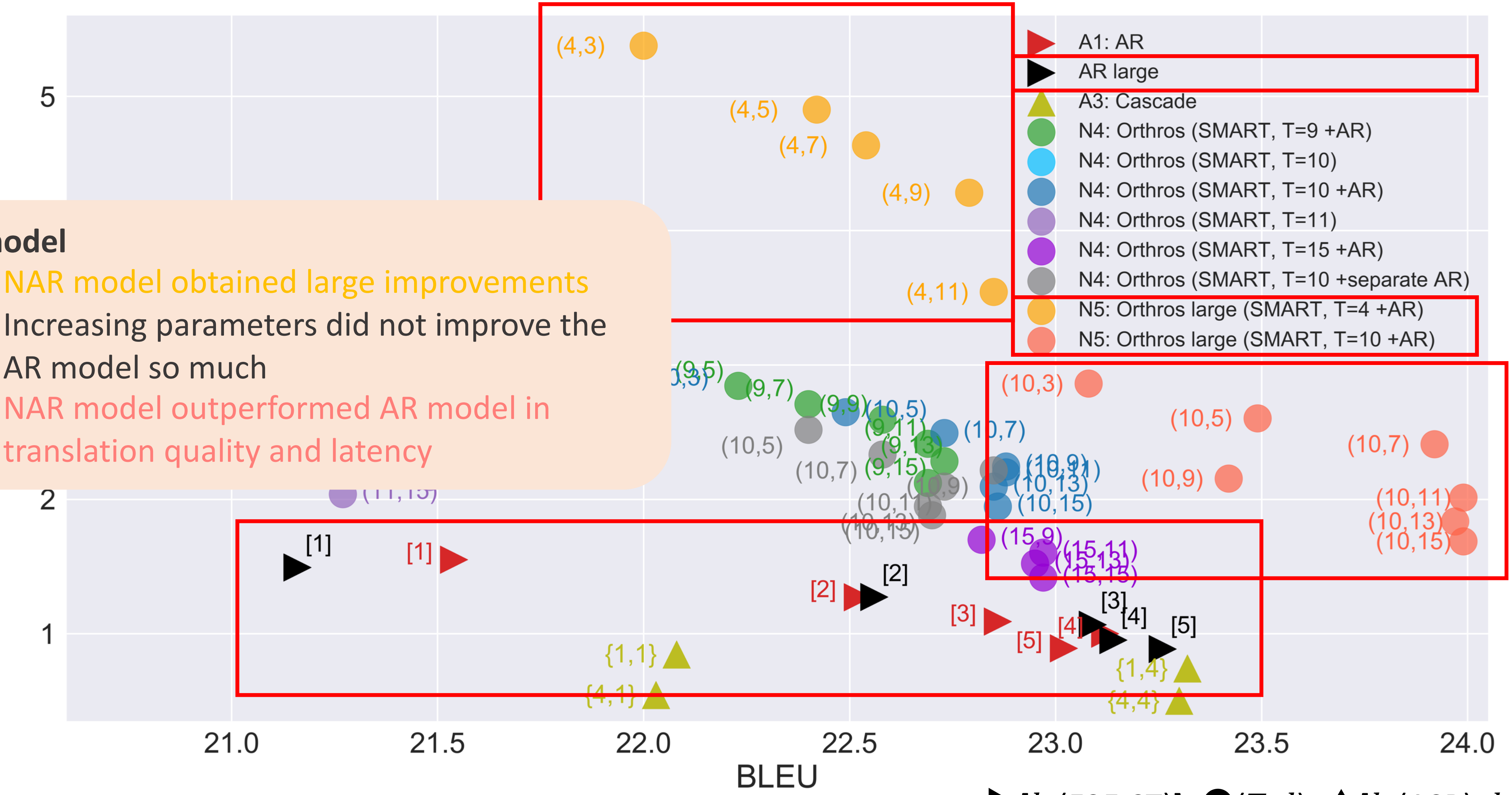


Comparison with AR E2E models

- Orthros (N4) achieved the same BLEU as AR model (A1) with 3× speed-up for greedy decoding and 1.5× speed-up for beam search decoding

▶[b (E2E-ST)], ●(T, l), ▲[b (ASR), b (MT)]

Quality and latency trade-off on Must-C En-De



Large model

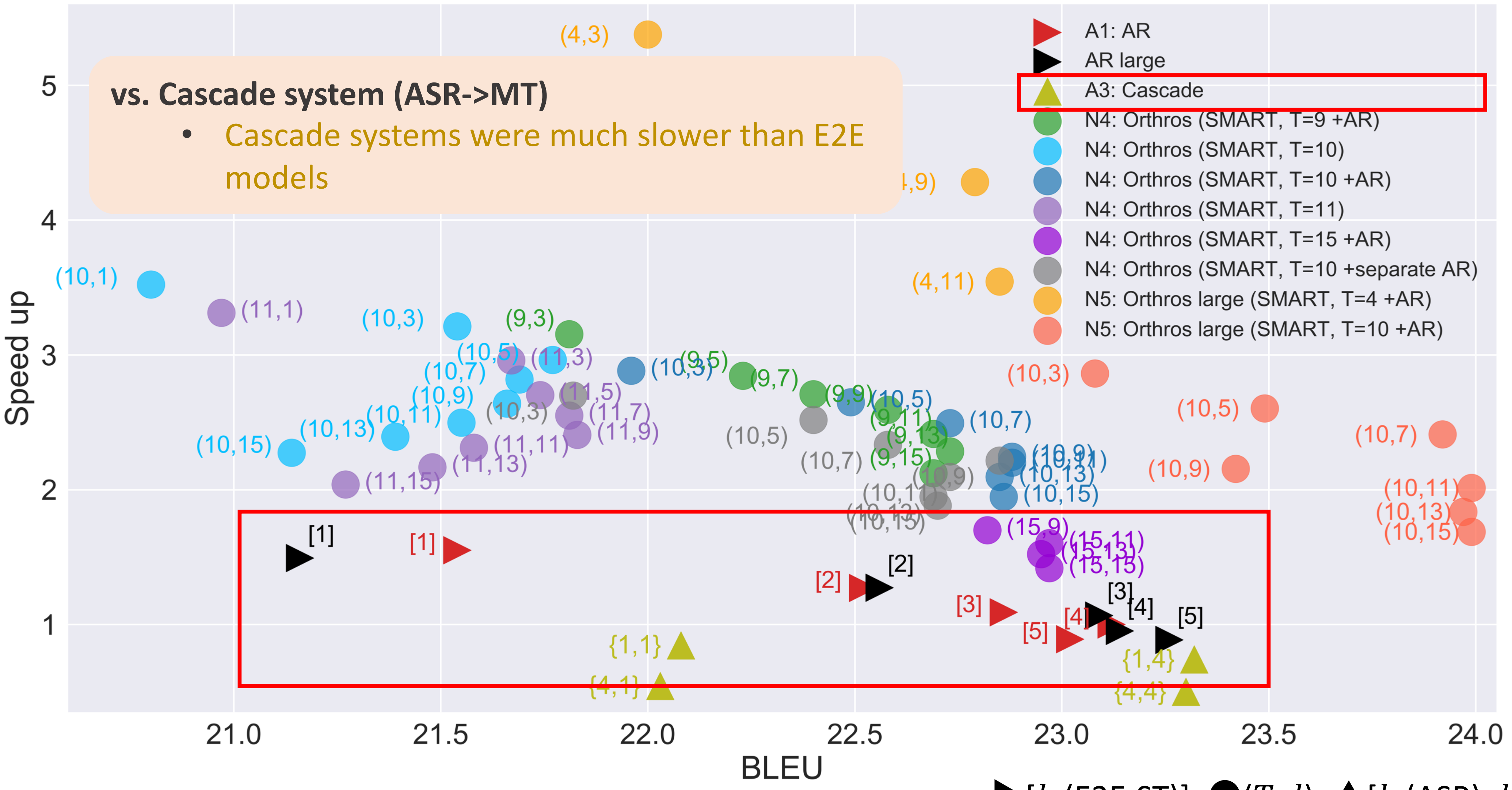
- NAR model obtained large improvements
- Increasing parameters did not improve the AR model so much
- NAR model outperformed AR model in translation quality and latency

▶[b (E2E-ST)], ●(T, l), ▲[b (ASR), b (MT)]

Quality and latency trade-off on Must-C En-De

vs. Cascade system (ASR->MT)

- Cascade systems were much slower than E2E models

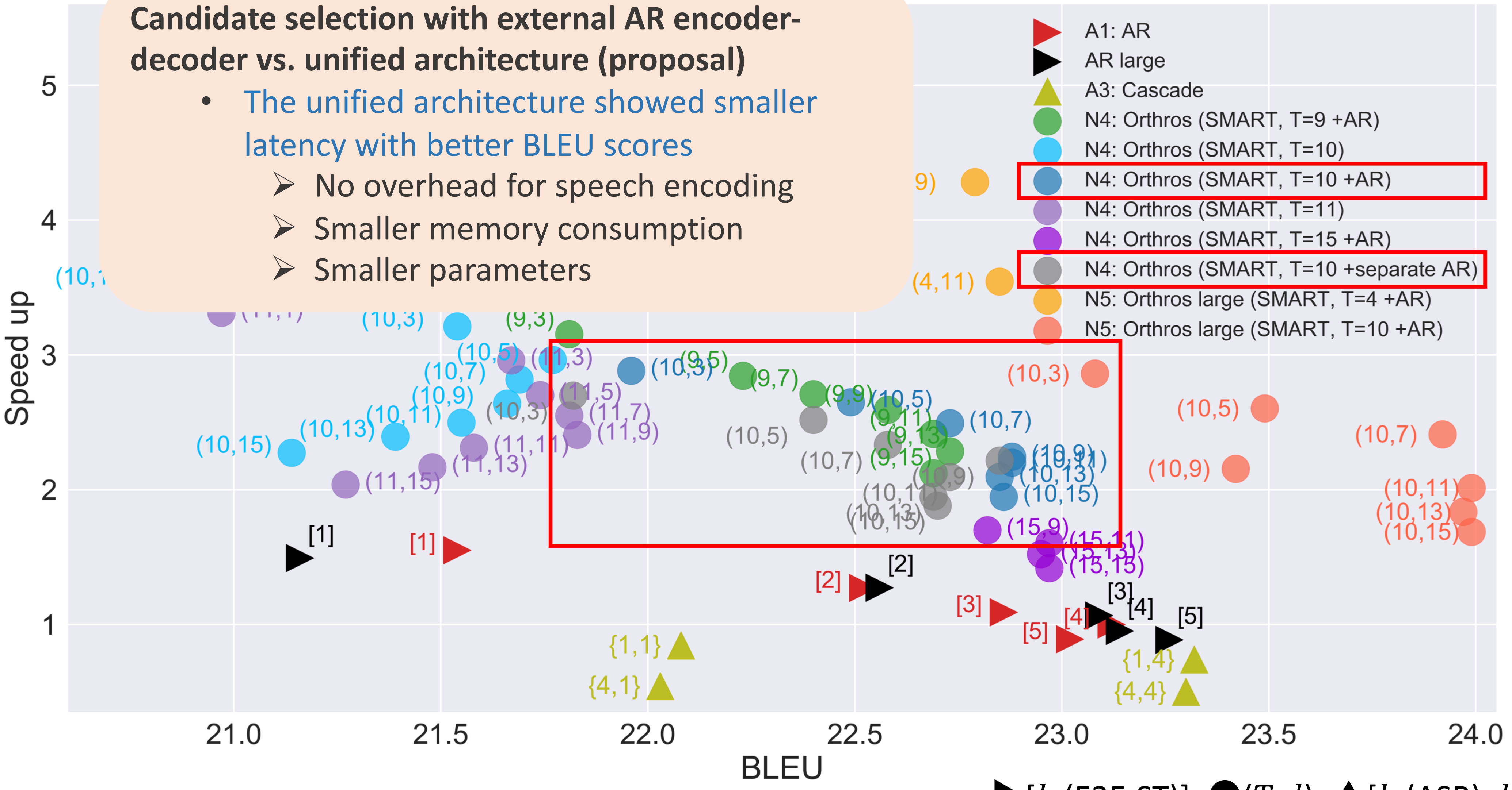


▶[b (E2E-ST)], ●(T, l), ▲[b (ASR), b (MT)]

Quality and latency trade-off on Must-C En-De

Candidate selection with external AR encoder-decoder vs. unified architecture (proposal)

- The unified architecture showed smaller latency with better BLEU scores
  - ➢ No overhead for speech encoding
  - ➢ Smaller memory consumption
  - ➢ Smaller parameters



▶[b (E2E-ST)], ●(T, l), ▲[b (ASR), b (MT)]

Results: Fisher-CallHome Spanish/Libri-trans

Model		BLEU		
		Fisher-CallHome Spanish		
		Fisher-test	CallHome-evltest	Libri-trans
Autoregressive	Transformer (b=1)	48.38	18.07	16.52
	Transformer (b=4)	48.49	18.90	16.84
	Transformer + Seq-KD (b=1)	50.34	19.09	15.91
	Transformer + Seq-KD (b=4)	50.32	19.81	16.44
Non-autoregressive	CTC (b=1)	45.97	15.91	12.10
	Orthros (CMLM, T=4)	46.03	16.71	12.90
	Orthros (CMLM, T=10+AR decoder)	47.80	18.28	13.69
	Orthros (CMLM T=10)	48.56	18.60	14.68
	Orthros (CMLM, T=10+AR decoder)	49.98	19.71	15.43
	Orthros (SMART, T=4)	45.89	17.39	14.17
	Orthros (SMART, T=10+AR decoder)	**48.73**	**19.25**	**14.99**
	Orthros (SMART, T=10)	47.09	18.25	15.11
	Orthros (SMART, T=10+AR decoder)	**50.07**	20.10	**16.08**
	+ BPE8k -> BPE16k	50.18	19.88	16.22

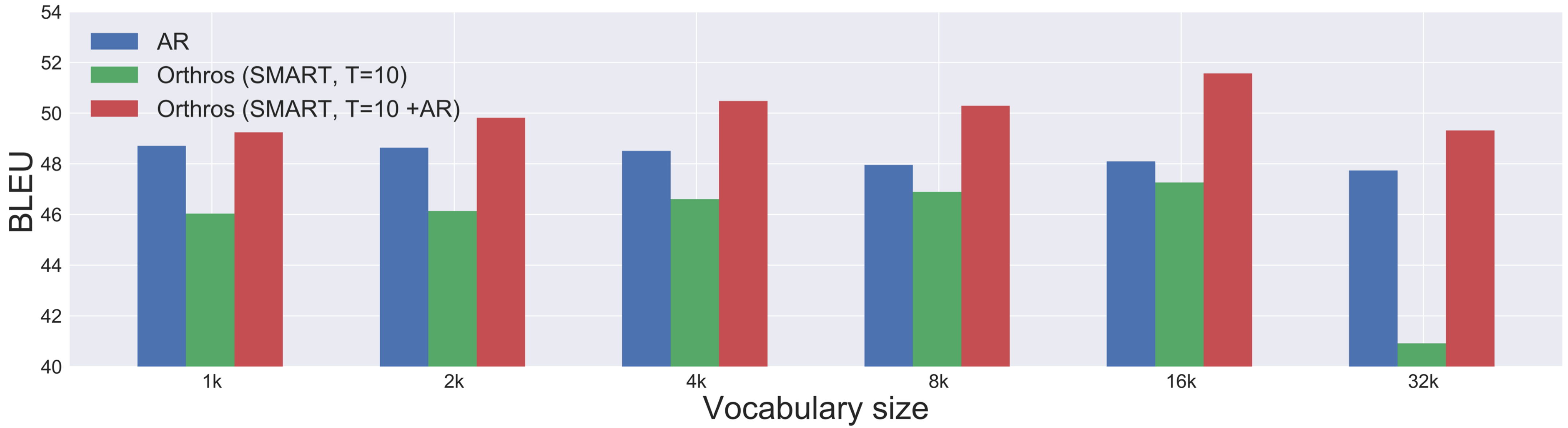
Ablation study on Fisher-CallHome dev set

Model	BLEU			
	$T = 4$		$T = 10$	
	w/o AR	w/ AR	w/o AR	w/ AR
Orthros BPE8k	**45.76**	**49.01**	46.88	**50.28**
- Seq-KD	44.36	47.42	44.25	49.50
- AR decoder	45.53	-	**46.94**	-
+ length prediction w/ CTC	45.41	48.18	46.79	50.05

- Seq-KD was beneficial (multi-modality problem was alleviated)
- Joint training with AR decoder itself had no impact on BLEU scores
- Linear classifier-based length prediction was better than the CTC-based one
  - ➢CTC-based length prediction: $[\hat{N} - |\frac{l}{2}|, \hat{N} + |\frac{l}{2}|]$, where $\hat{N} = \lfloor \alpha \hat{N}_{src} \rfloor$

    (α: hyperparameter, $\hat{N}_{src}$: ASR hypothesis length obtained by CTC greedy decoding)

Effect of vocabulary size on Fisher-CallHome dev set



- AR models have a peak around BPE1k (due to data sparseness, 170h)
- Candidate selection with AR decoder is always effective regardless of BPE size
- Orthros + candidate selection continued to improve until BPE16k
    - ➢Most tokens in vocabulary are "complete" word
    - ➢Complementary effect on the conditional independence assumption

Conclusion and future work

- Perceived latency reduction for streaming encoder-decoder ASR
  - Alignment information is effective on the decoder side
  - CTC alignment is as good as alignment from hybrid ASR system

- Fast non-autoregressive decoding for E2E-ST, *Orthros*
  - ➢ AR decoder + NAR decoder on shared speech encoder
  - ➢ Candidate selection with AR decoder was very effective to estimate target lengths
  - ➢ Reached comparable translation quality to SOTA AR E2E-ST models with more than 2× latency reduction