Enhancing Monotonic Multihead Attention for Streaming ASR

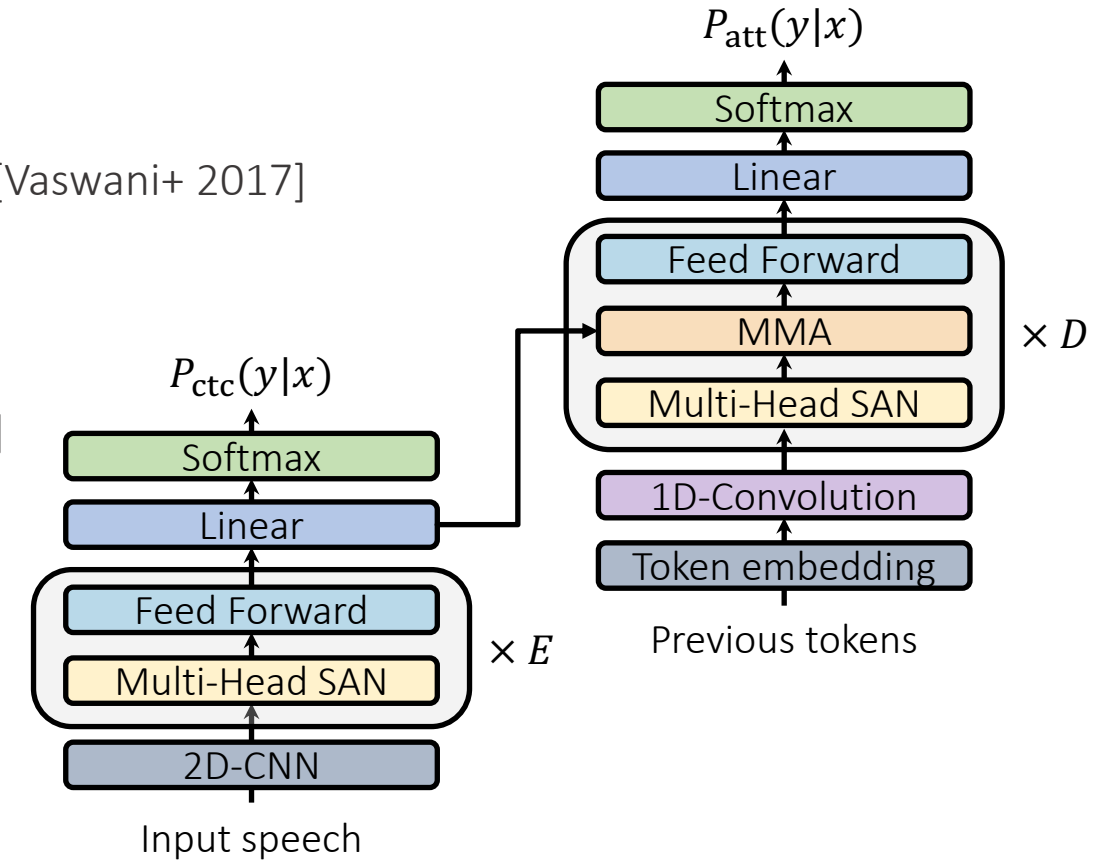
Hirofumi Inaguma[1] Masato Mimura[1] Tatsuya Kawahara[1]
[1]Graduate School of Informatics, Kyoto University, Japan

Transformer ASR

- Effcient long-span sequence modeling [Vaswani+ 2017]
  - ➢Self-attention
  - ➢Multi-head attention
  - ➢Outperform RNN counterparts [Karita+ 2019]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$$

$$O = [head_1; \ldots; head_H]W_o$$



Joint training with CTC loss [Karita 2019+]

$$\mathcal{L}_{\text{total}} = (1 - \lambda_{\text{ctc}})\mathcal{L}_{\text{att}}(y|x) + \lambda_{\text{ctc}}\mathcal{L}_{\text{ctc}} \quad (0 \leq \lambda_{\text{ctc}} \leq 1)$$

Streaming Transformer ASR

Frame-synchronous decoding

- *Connectionist temporal classification (CTC)* [Grave+ 2006, Salazar+ 2019]
- *RNN-Transducer/Transformer-Transducer* [Grave+ 2013, Yeh+ 2019, Zhang+ 2020]
  - ➢ Successful in industry
  - ➢ Large search space because of frame-wise predictions

Label synchronous decoding

- *Triggered attention* [Moritz+ 2020]
- *Continuous Integrate-and-fire (CIF)* [Dong+ 2020]

} Single input-output alignment (segment audio on the encoder side)

- *Monotonic chunkwise attention (MoChA)* [Chiu+ 2018]
  - ➢ Extended to ***monotonic multihead attention (MMA)*** in simultaneous MT [Ma+ 2020] and streaming ASR [Miao+ 2020, Tsunoo+ 2020]
  - ➢ Multiple input-output alignments based linguistic contexts captured in the decoder

Monotonic multihead attention (MMA) [Ma+ 2020]

Extend RNN to Transformer

- Replace every head in cross-attention with a monotonic attention (MA) head

Relationship with previous studies based on MMA

- Miao *et al.* and Tsunoo *et al.* also MoChA components in Transformer ASR
- However, they rely on the whole past encoder frames as context
  - ➢Not appropriate for linear-time decoding with HMA
  - ➢We stick to restricted input context

Background

1. *Hard monotonic attention (HMA)* [Raffel+ 2017]
2. *Monotonic chunkwise attention (MoChA)* [Chiu+ 2018]
3. *Monotonic multihead attention (MMA)* [Ma+ 2020]

Hard monotonic attention (HMA) [Raffel+ 2017]

Test time

h_j: encoder state
s_i: decoder state

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

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

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

Not differentiable

Points

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

Training time

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

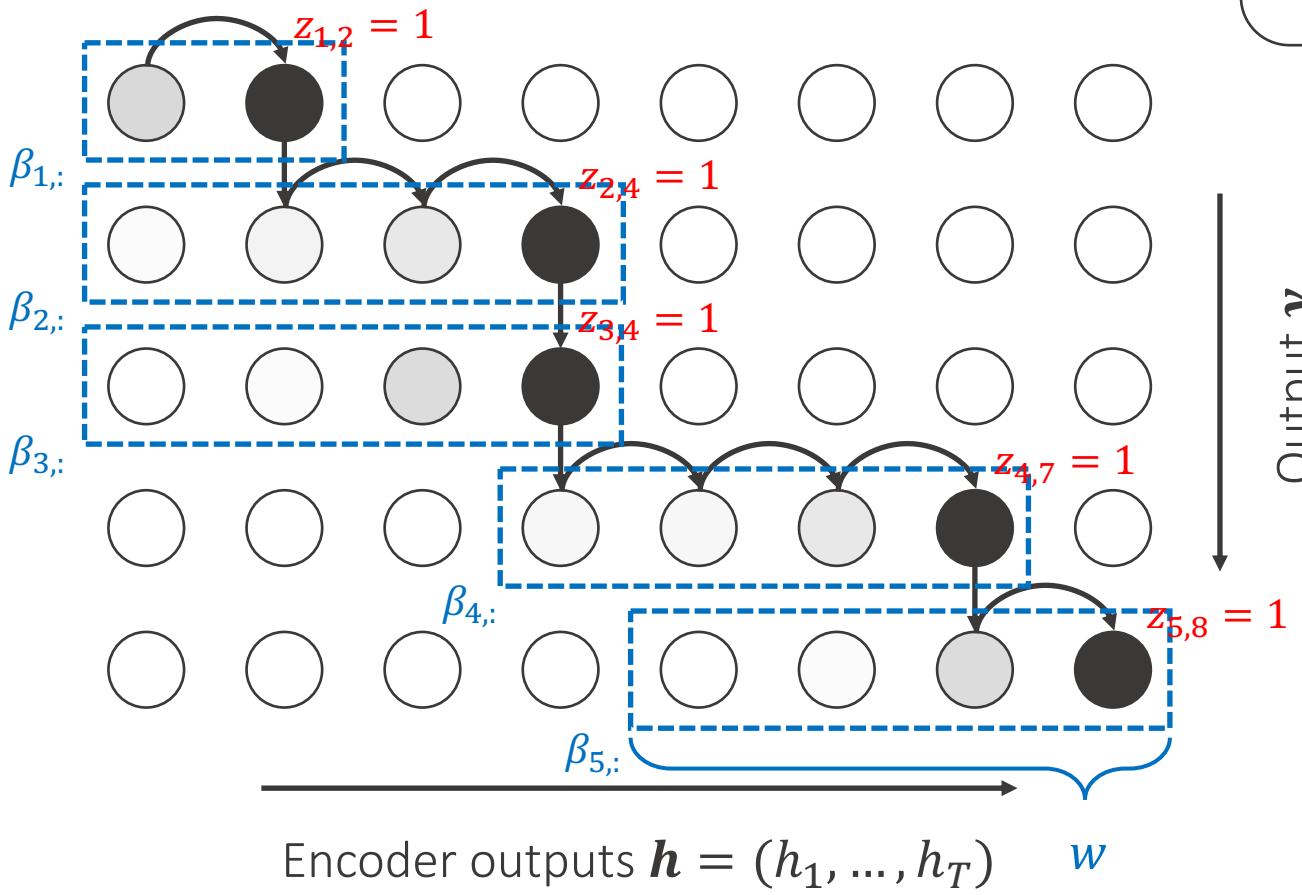
$$= (1 - p_{i,j-1}) \frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}$$

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

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

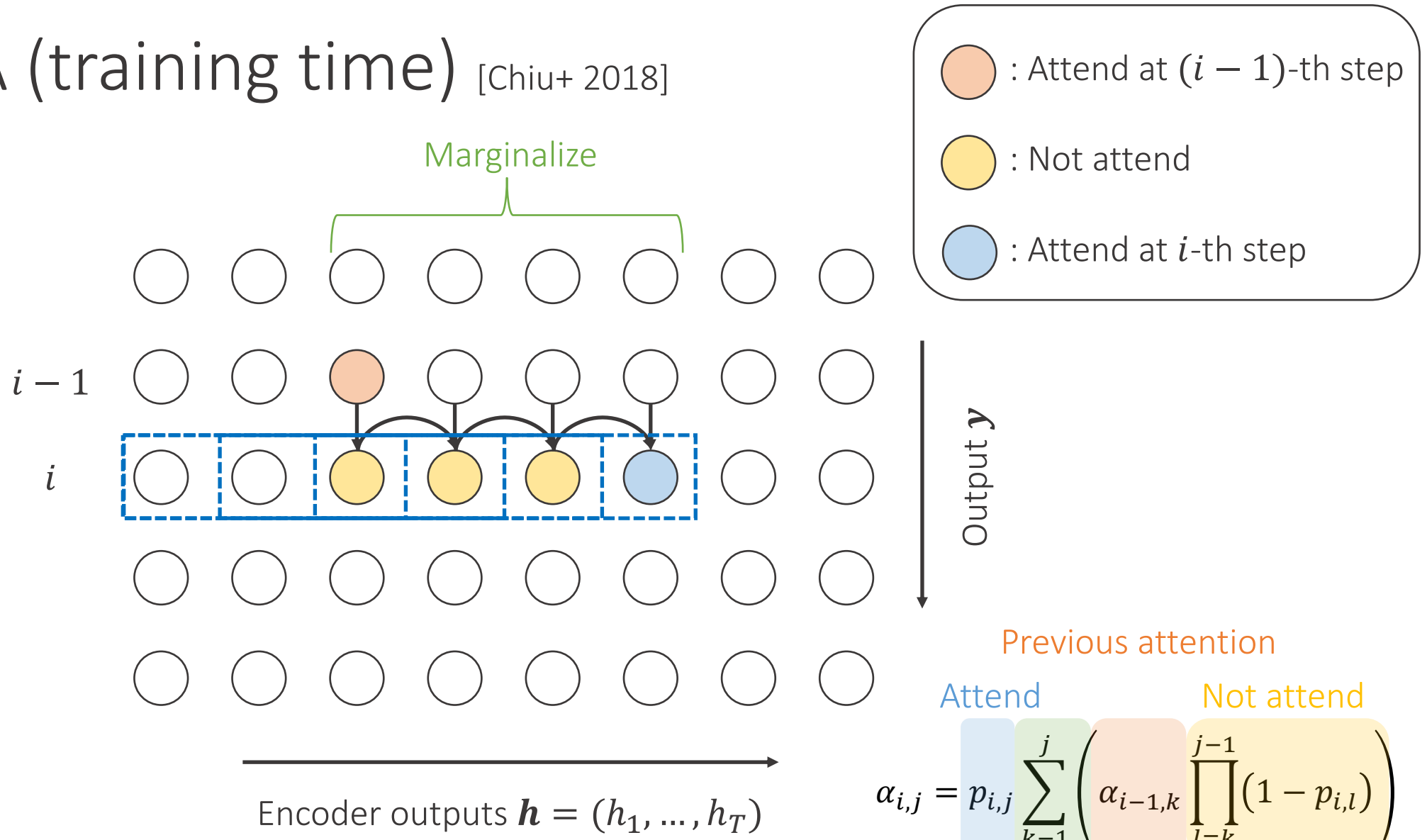
MoChA (test time) [Chiu+ 2018]

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]



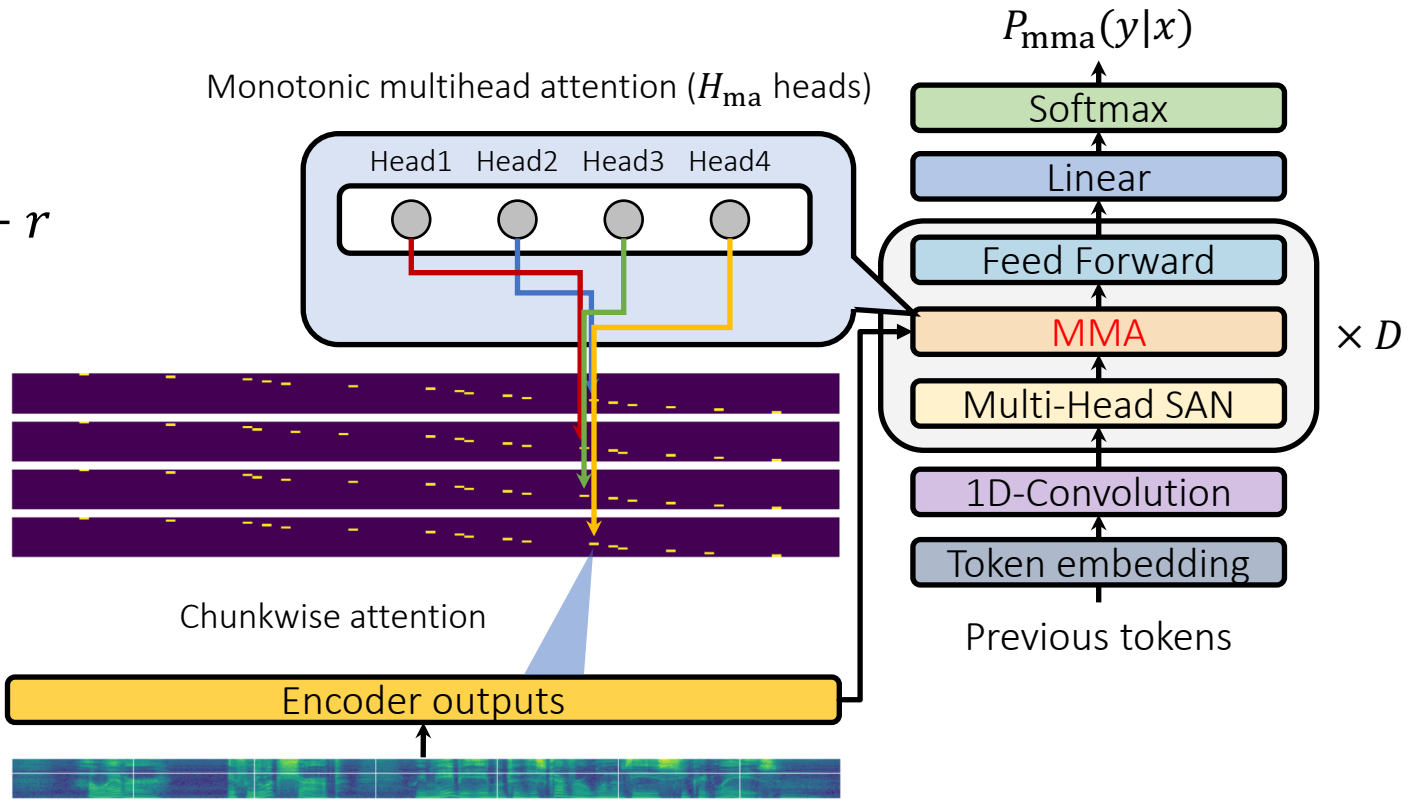
Monotonic multihead attention (MMA) [Ma+ 2020]

- Each MA head can scan encoder frames with different pace
- $z_{i,j}$ is independent each other in the same layer

$$\mathrm{MonotonicEnergy}(s_i, h_j) = \frac{W_s s_i (W_h h_j)^{\mathrm{T}}}{\sqrt{d_k}} + r$$

h_j: encoder state

s_i: decoder state at the l-th layer



Problem specification

Agreement in boundary detection

- The next token cannot be generated until all MA heads detect token boundaries
- If some heads do not learn proper monotonic alignments, they continue to scan memories until the last encoder frame
- Accordingly, the next token generation is delayed
  - ➢ Not suitable for streaming scenario

Goal

- Train MMA so that every MA head can detect boundaries around the corresponding acoustic boundaries

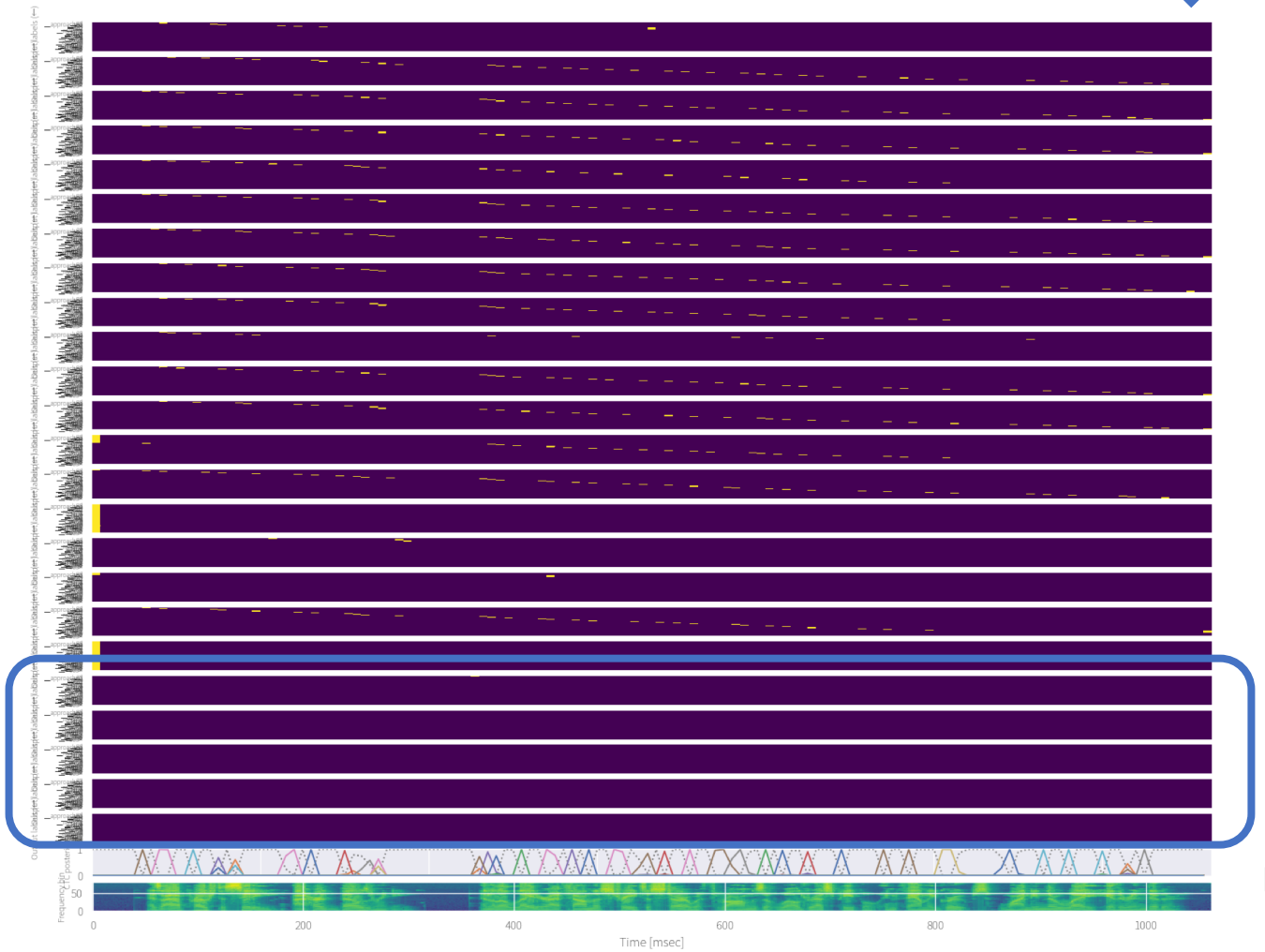
Alignment: Baseline MMA

First token cannot be generated until the bottom decoder layers detect the boundaries

Monotonic alignments are learnt in dominant heads, especially in upper layers

Upper layer

Head4@layer6
Head3@layer6
:



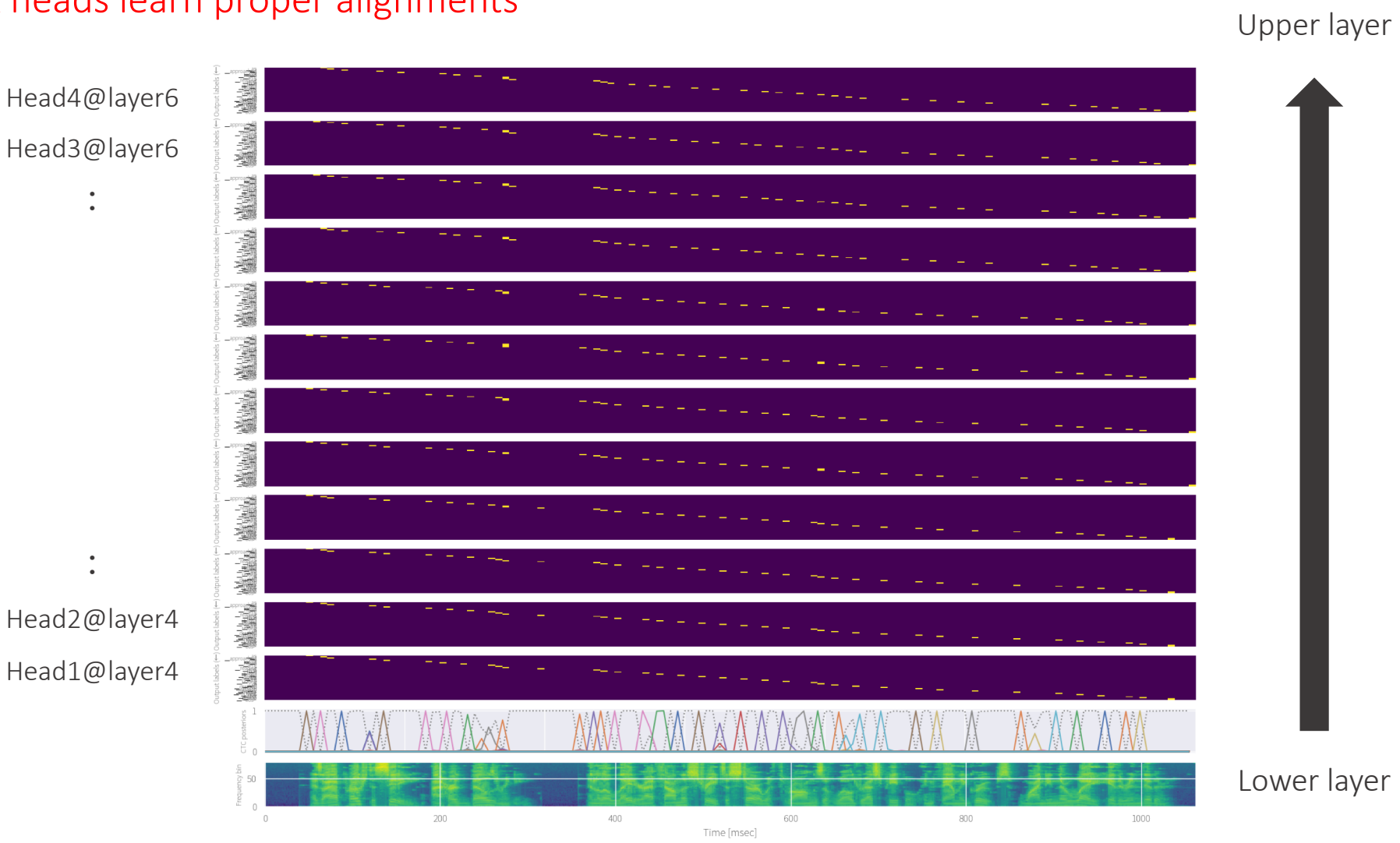
Alignments are not learnt

:
Head2@layer1
Head1@layer1

Lower layer

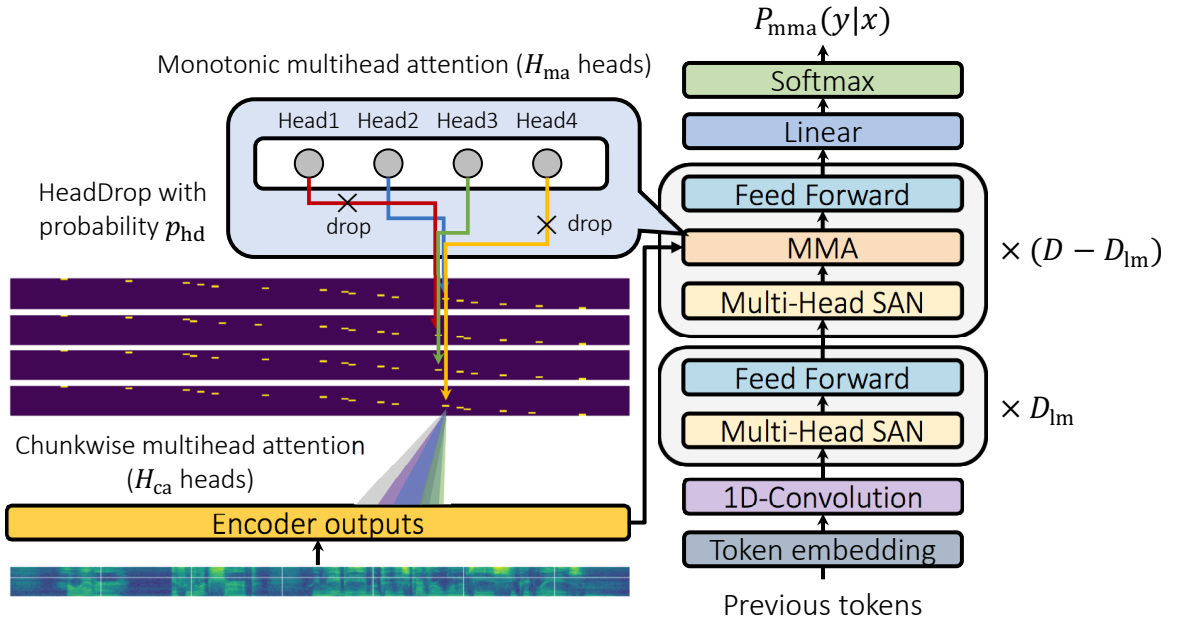
Alignment: MMA w/ proposed enhancement

All MA heads learn proper alignments



Proposed methods: Overview

1. **HeadDrop regularization**
   - Stochastically mask out some MA heads during training to encourage the rest MA heads to learn alignments
2. **Head pruning in lower decoder layers**
   - MA heads in lower decoder layers do not learn clear alignments
   - Remove HMA functions in the lower layers
3. **Chunkwise multihead attention**
   - Perform multiple chunkwise attention (CA) on top of every single MA head
4. **Head-synchronous beam search decoding**
   - Force non-activated MA heads to activate to improve consensus among MA heads



1. HeadDrop regularization

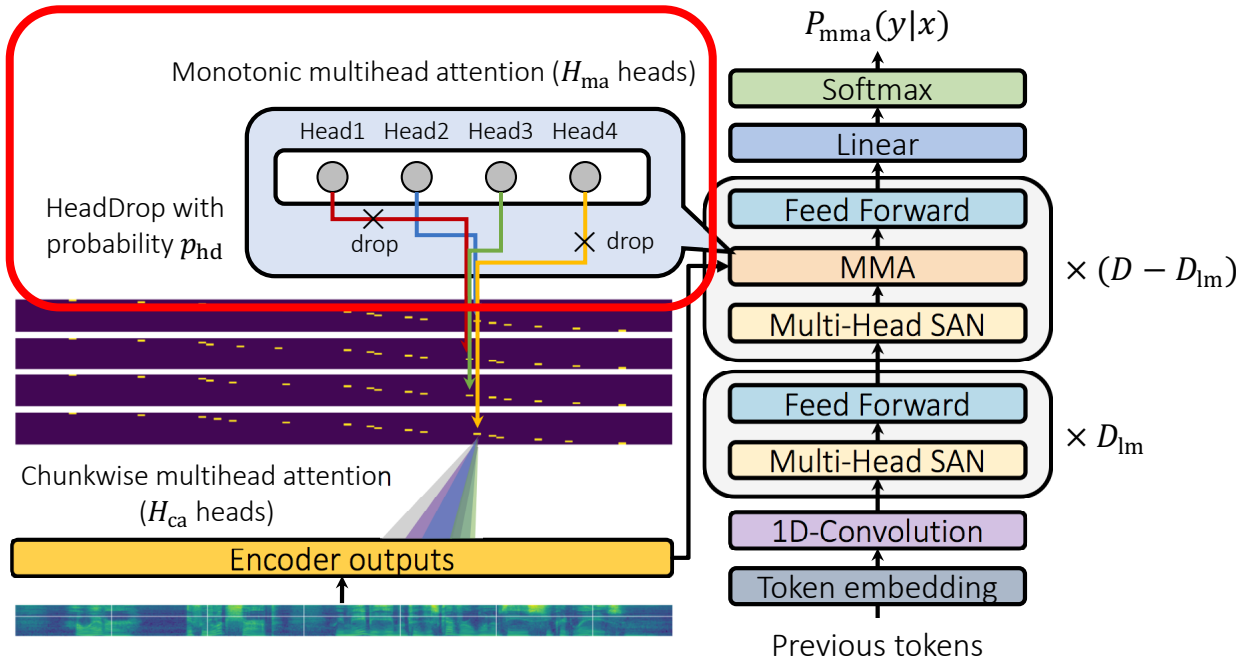
- Stochastically mask all elements in each MA head with a probability p_{hd} during training
- Expect to encourage other unmasked heads to learn proper alignments

$$m_h \sim \mathrm{Bernoulli}(p_{\mathrm{hd}})$$

$$\hat{\alpha}^h_{:,j} = \begin{cases} 0 & (\mathrm{mask}, m_h = 1) \\ \dfrac{H_{\mathrm{ma}}}{H^+_{\mathrm{ma}}} \cdot \alpha^h_{:,j} & (\mathrm{otherwise}, m_h = 0) \end{cases}$$

$$O = [\hat{A}_1 V_1; \ldots; \hat{A}_H V_H] W_o$$

Normalize to keep the scale of outputs
(H^+_{ma}: number of non-masked MA heads)

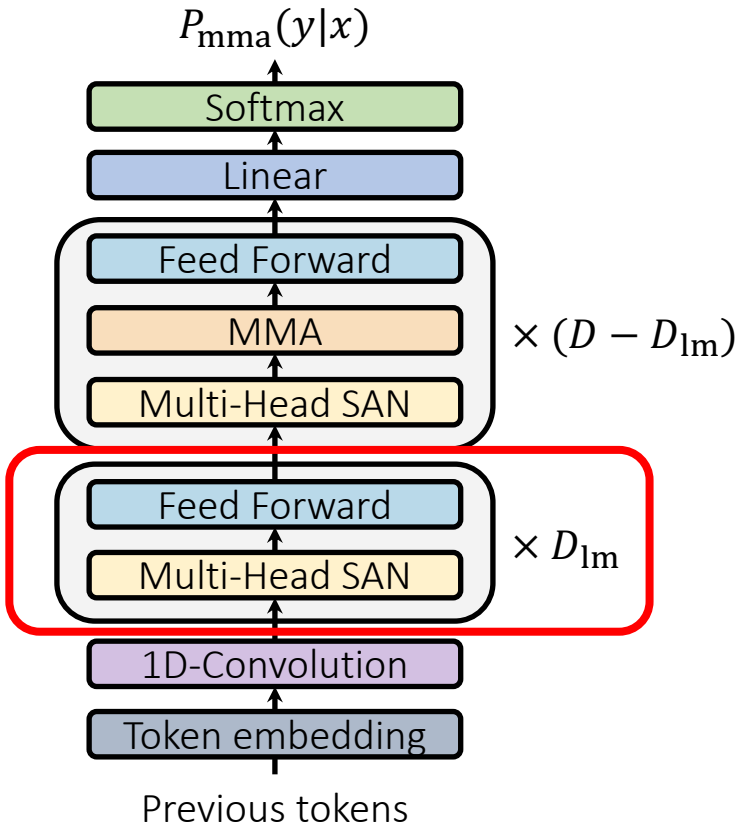


2. Head pruning in lower decoder layers

- Even w/ HeadDrop, some MA heads in lower layers did not learn proper alignments
  - ➢ Cross-attention in lower layers are not responsible for learning diagonal alignments

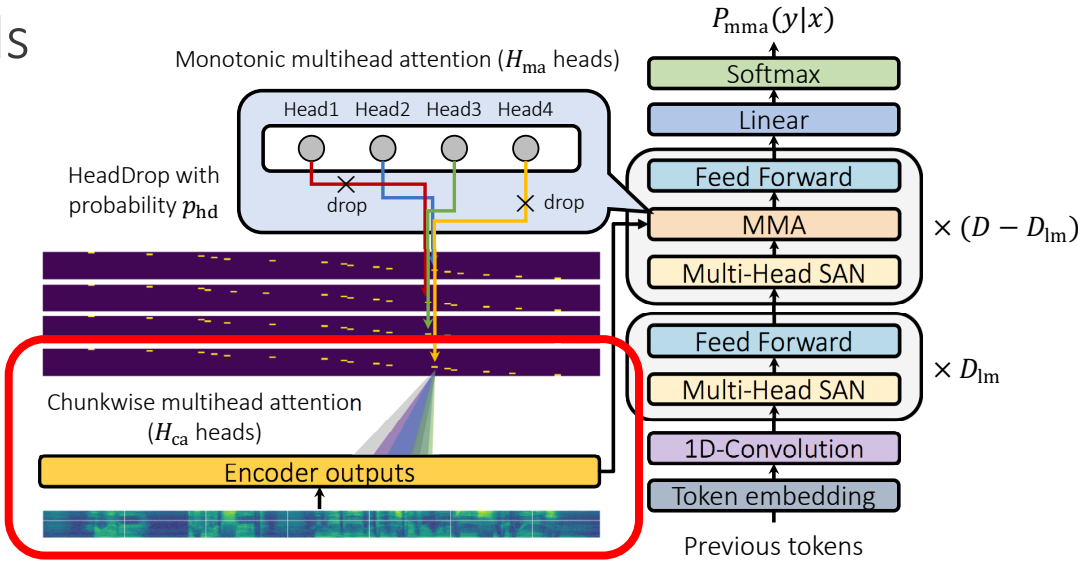
- Remove the MMA function (i.e., cross-attention) in the bottom D_{lm} decoder layers ($1 \leq D_{\mathrm{lm}} \leq D$)
  - ➢ Bottom D_{lm} decoder layers have language model structure

- Total number of MA heads: $D \cdot H_{\mathrm{ma}} \rightarrow (D - D_{\mathrm{lm}}) \cdot H_{\mathrm{ma}}$
  - ➢ Speed up Inference

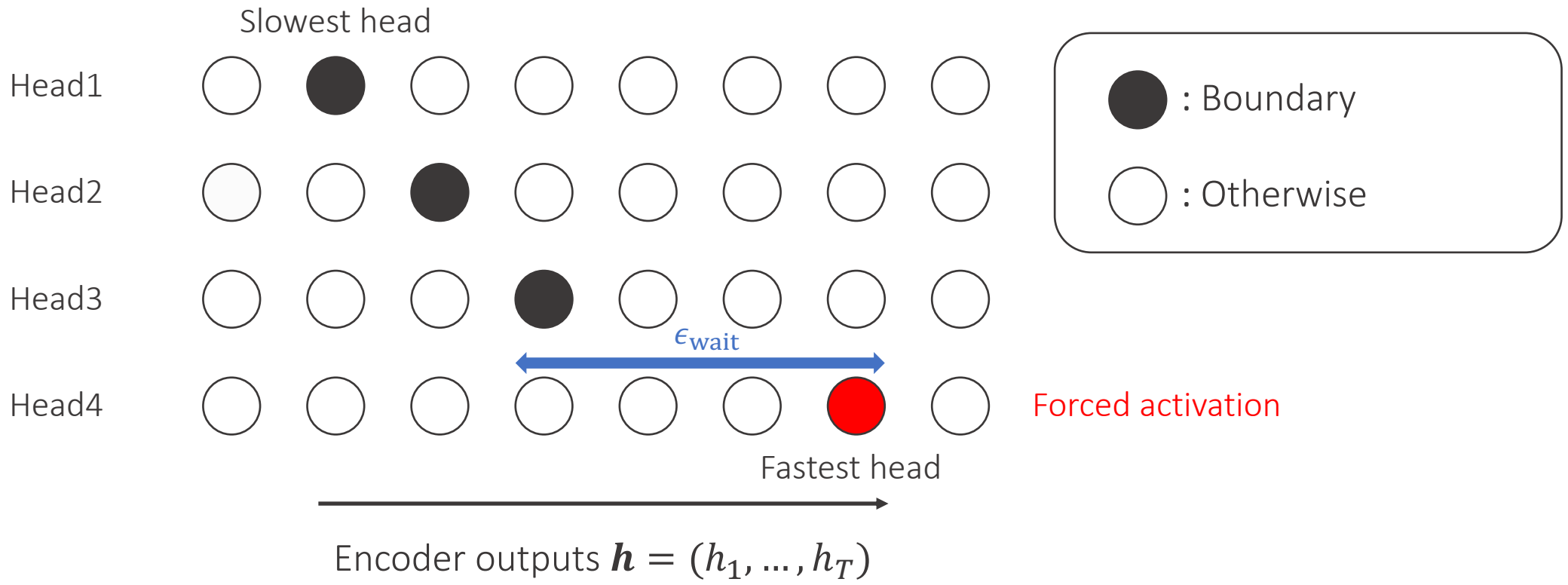


3. Chunkwise multihead attention

- Extend the idea of MoChA to the multi-head version to extract useful representations with multiple views from every token boundary
- Each MA head has H_{ca} (≥ 1) chunkwise attention (CA) head
- Total number of CA heads in a layer: $H_{\mathrm{ma}} \rightarrow H_{\mathrm{ma}} \cdot H_{\mathrm{ca}}$
- Share parameters of CA heads among MA heads in the same layer was effective



4. Head-synchronous beam search decoding



- Force non-activated MA heads to activate after a small delay ϵ_{wait} [frame]
- Latency between fastest and slowest MA heads in the same layer is equal to or less than ϵ_{wait}
- At least one MA head must be activated at every layer

Experimental setting

Corpus	Librispeech (960h), TEDLUM2, AISHELL-1
Feature	80-dim log-mel fbank
Output unit	BPE 10k units
Architecture	Encoder: 6-layer CNN (1/8 downsampling) -> 12-layer Transformer
	Decoder: 6-layer Transformer
Regularization	$p_{\mathrm{hd}} = 0.5$
Optimization	Adam + Noam schedule
Loss weight	$\lambda_{\mathrm{ctc}} = 0.3$
Decoding	Beam width: 10 (no CTC score), shallow fusion w/ 4-layers of LSTM-LM

Evaluation metric for boundary detection

1. **Boundary coverage** (How well each MA head learns monotonic alignments)
   - The ratio of detected boundaries in the best hypothesis to the number of hypothesis length

$$R_{\mathrm{cov}}[\%] = \frac{1}{N}\sum_{n=1}^{N}\frac{Q_i^{n,1}}{|\tilde{y}^{n,1}|}\times 100 \qquad Q_i^{n,k} = \frac{1}{H_{\mathrm{ma}}^{\mathrm{total}}}\sum_{h=1}^{H_{\mathrm{ma}}^{\mathrm{total}}}\sum_{i'=1}^{i}\sum_{j=1}^{|\boldsymbol{h}|}\alpha_{i',j}^{h}$$

2. **Streamability** (How often the model satisfies the streamable condition)
   - The ratio of utterances satisfying the streamiable condition over all candidates in the beam until the best candidate is completely generated

$$R_{\mathrm{str}}[\%] = \frac{1}{N}\sum_{n=1}^{N}\delta_n \times 100$$

Definition: Streamable condition
- All MA heads in the decoder detect the corresponding token boundaries before reaching the last encoder frame

$$\delta_n = \begin{cases} 1 & (Q_i^{n,k} = |\tilde{y}^{n,k}|, 1 \le i^{\forall} \le |\tilde{y}^{n,1}|, 1 \le k^{\forall} \le |\Omega_i^n|) \\ 0 & (otherwise) \end{cases}$$

Offline results: HeadDrop, head pruning

- HeadDrop improved WER
- Pruning MA heads in lower layers improved coverage and streamability

ID	D_{lm}	H_{ma}	$H_{\text{ma}}^{\text{total}}$	HeadDrop	dev-clean/dev-other %WER	dev-clean/dev-other Boundary coverage	dev-clean/dev-other Streamability
A1	0		24		8.6 / 16.5	67.40	0.0
A2	1		20		7.3 / 16.3	79.02	0.0
A3	2	4	16	-	4.7 / 12.6	86.07	0.0
A4	3		12		4.5 / 12.8	83.87	0.0
A5	4		8		3.6 / 10.8	**93.80**	**0.9**
B1	0		24		**3.7** / 11.4	60.59	0.0
B2	1		20		4.0 / 11.9	73.73	0.0
B3	2	4	16	✓	3.9 / **10.8**	98.85	3.7
B4	3		12		4.1 / 11.0	99.36	6.4
B5	4		8		4.1 / 11.3	99.50	15.8

Offline results: Head-synchronous decoding

- Head-synchronous beam search decoding improved WER and streamability

ID	D_{lm}	H_{ma}	w	H_{ca}	dev-clean/dev-other %WER	dev-clean/dev-other Boundary coverage	dev-clean/dev-other Streamability
B3	2	4	4	1	3.9 (-0.1) / 10.7 (-0.1)	99.74 (+0.89)	21.6 (+17.9)
B4	3				3.9 (-0.2) / **10.6 (-0.4)**	99.76 (+0.40)	25.1 (+18.7)
B5	4				**3.8 (-0.3)** / 11.1 (-0.2)	99.84 (+0.34)	40.5 (+24.7)

w/ head-sync (improvement from w/o head sync)

Offline results: Chunkwise multihead attention

- Increasing window size $\boldsymbol{w}$ in chunkwise attention was effective (B* vs. D*)
- Multiple chunkwise attention heads improved WER and streamability (D* vs. E*)

ID	D_{lm}	H_{ma}	w	H_{ca}	dev-clean/dev-other		
					%WER	Boundary coverage	Streamability
B3	2	4	4	1	3.9 / 10.7	99.74	21.6
B4	3				3.9 / 10.6	99.76	25.1
B5	4				3.8 / 11.1	99.84	40.5
D1	2	4	16	1	**3.3 / 9.9**	99.78	37.4
D2	3				3.7 / 10.8	99.83	36.5
D3	4				3.5 / 10.4	**99.93**	**60.4**
E1	2	4	16	2	**3.3** / 10.2	99.78	40.6
E2	3				3.6 / 10.3	99.87	51.2
E3	4				3.5 / 10.7	**99.92**	50.0
E4	2			4	**3.3 / 9.8**	99.91	77.9
E5	3				3.4 / 9.9	99.90	**84.5**
E6	4				3.6 / 10.4	**99.92**	63.2

Offline results: Head number, head place

- Reducing the total number of MA heads was not a solution (vs. C*, F1)
- Placing multiple MA heads in upper layers is important

ID	D_{lm}	H_{ma}	$H_{\text{ma}}^{\text{total}}$	w	H_{ca}	Head-sync	dev-clean/dev-other %WER	dev-clean/dev-other Boundary coverage	dev-clean/dev-other Streamability
C1	0	1	6	4	1	-	4.9 / 11.7	99.38	15.7
C2		1	6				3.7 / **10.4**	**99.86**	**35.9**
C3		2	12				**3.5** / 10.7	72.08	0.0
D1	2	4	20	16	1	✓	**3.3 / 9.9**	99.78	37.4
D2	3		16				3.7 / 10.8	99.83	36.5
D3	4		12				3.5 / 10.4	**99.93**	**60.4**
E1	2	4	20	16	2	✓	**3.3** / 10.2	99.78	40.6
E2	3		16				3.6 / 10.3	99.87	51.2
E3	4		12				3.5 / 10.7	**99.92**	50.0
E4	2		20		4		**3.3 / 9.8**	99.91	77.9
E5	3		16				3.4 / 9.9	99.90	**84.5**
E6	4		12				3.6 / 10.4	**99.92**	63.2
F1	0	1	6	16	4	-	3.5 / 10.5	96.23	40.6

The rest 15.5% was able to continue streaming decoding until **76.9%** of input frames on overage

Streaming results

Model		WER [%] Librispeech test-clean	WER [%] Librispeech test-other	WER [%] TEDLIUM 2	CER [%] AISHELL -1
Offline	Transformer	3.3	9.1	**10.1**	**6.4**
	+ data augmentation	2.8	7.0	-	-
	++ large	**2.5**	**6.1**	-	-
	MMA (E5)	3.4	9.9	10.5	6.5
	Triggered attention [Moritz+ 2020]	2.8	7.2		
	CIF [Dong+ 2020]	3.3	9.7		
	MoChA [Inaguma+ 2020]	4.0	9.5	11.3	
	MMA [Tsunoo+ 2020]				9.7
Streaming	MMA (narrow chunk)	3.5	11.1	11.0	7.5
	MMA (wide chunk)	3.3	10.5	**10.2**	**6.6**
	+ data augmentation	3.0	8.5	-	-
	++ large	**2.7**	**7.1**	-	-

- Streaming encoder: chunk-hopping mechanism (N_l / N_c / N_r)
  - narrow: 960ms/640ms/320ms
  - wide: 640ms/1280ms/640ms
- Data augmentation: speed perturbation + SpecAugment

Conclusion

Observation

- Cross-attention heads in the lower decoder layers do no learn clear alignments

Proposals

- Four methods to stabilize the streaming inference with MMA-based Transformer ASR
- (1) HeadDrop, (2) head pruning, (3) chunkiwe multihead attention, (4) head-synchronous beam search decoding

Future work

- Reduce perceived latency caused by delayed token generation similar to RNN-based methods [Inaguma+ 2020]
- Analyze what lower decoder layers in Transformer ASR learn