Interspeech 2022
18-22 September 2022, Incheon, Korea

Streaming model for Acoustic to Articulatory Inversion with transformer networks

Sathvik Udupa[1], *Aravind Illa*[1], *Prasanta Kumar Ghosh*[1]

[1]Electrical Engineering, Indian Institute of Science (IISc), Bangalore-560012, India

sathvikudupa66@gmail.com, aravindece77@gmail.com, prasantag@gmail.com

Abstract

Estimating speech articulatory movements from speech acoustics is known as Acoustic to Articulatory Inversion (AAI). Recently, transformer-based AAI models have been shown to achieve state-of-art performance. However, in transformer networks, the attention is applied over the whole utterance, thereby needing to obtain the full utterance before the inference, which leads to high latency and is impractical for streaming AAI. To enable streaming during inference, evaluation could be performed on non-overlapping chucks instead of a full utterance. However, due to a mismatch of the attention receptive field during training and evaluation, there could be a drop in AAI performance. To overcome this scenario, in this work we perform experiments with different attention masks and use context from previous predictions during training. Experiments results revealed that using the random start mask attention with the context from previous predictions of transformer decoder performs better than the baseline results.

Index Terms: acoustic to articulatory inversion, Streaming prediction, transformer networks

1. Introduction

Articulatory movements play an important role in speech production as an intermediate representation between neuro-motor planning and speech acoustics [1]. Having information of the articulatory movements along with speech acoustics has shown benefits in different applications such as in speech synthesis [2], speech recognition [3, 4], language learning [5] etc. In the absence of directly measured articulatory trajectories, they are typically estimated from acoustic features, such as Mel-Frequency Cepstral Coefficients (MFCC). This process of estimating articulatory movements from acoustics is referred to as acoustic to articulatory inversion (AAI).

In recent years, progress has been made on AAI using deep neural networks. Different type of models have been applied such as DNNs [6], CNNs [7], LSTMs [2, 8], WaveNet [9], transformers [10] etc. Recently, Transformer based AAI models [10] have shown to achieve state-of-the-art performance. However, there are limitations in using transformer network-based AAI models directly for real-time systems [11]. Since these models are time-dependent on sequence length, longer sequences will bring in more latency, hindering real-time usage. To address this, a modified approach is used, usually referred to as streaming, which is prominently used in some real-time use cases such as speech recognition. Streaming models involve adapting a model to have less real-time latency while trying to retain the performance. In this work, we build a transformer-based streaming model for AAI.

Streaming is used for automatic speech recognition with different models such as transducers [12, 13, 14], attention [15, 16, 17] and CTC based [18] etc. Different memory mechanisms [19, 20] are also used in ASR streaming to learn in such a way that the model can adapt to chunk level decoding. While the problem statement of ASR is similar, AAI is a much simpler problem that does not require alignment between input and output. If we consider frame-level AAI visualisation, we need not modify the previous outputs as we perform streaming. These simplifications allow for a much simpler approach for streaming AAI models. Streaming can be performed by windowing the utterance sequence into chunks of non-overlapping data [21]. This can be improved by utilising previous chunk information as memory [22].

In this work, we first consider if any training approach for streaming the AAI model is necessary for AAI. To evaluate this premise, independent chunk level predictions are performed on a trained full sequence model. We observe the drop in performance in streaming inference when compared to full sequence inference. We propose training approaches to improve the AAI performance by modifying the receptive field in transformers and utilising context from previous predictions. Experimental results revealed that the proposed training approaches improve the performance of AAI while performing inference in a streaming manner.

2. Dataset

We use a total of 460 phonetically balanced English sentences from the MOCHA-TIMIT corpus [23] as the stimuli for acoustic-articulatory data collection. Data from 10 subjects are considered, comprising 6 male (M1, M2, M3, M4, M5, M6) and 4 female (F1, F2, F3, F4) subjects with ages in the range of 20-28 years. All the subjects are native Indians who are proficient in English. We simultaneously recorded audio signals using a microphone and articulatory movement data using Electromagnetic Articulograph (EMA) AG501 [24].

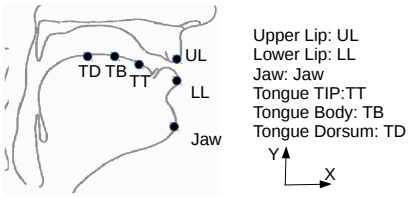

Figure 1: *Schematic diagram indicating the placement of EMA sensors [25]*

As shown in Figure 1, six sensors were glued on different articulators, viz. Upper Lip (UL), Lower Lip (LP), Jaw, Tongue Tip (TT), Tongue Body (TB), and Tongue Dorsum (TD). Two sensors were glued behind the ears for head movement correction. In this work, we only consider the movements in the mid-sagittal plane, corresponding to horizontal and vertical directions. Thus, we have twelve articulatory trajectories denoted by

10.21437/Interspeech.2022-10159

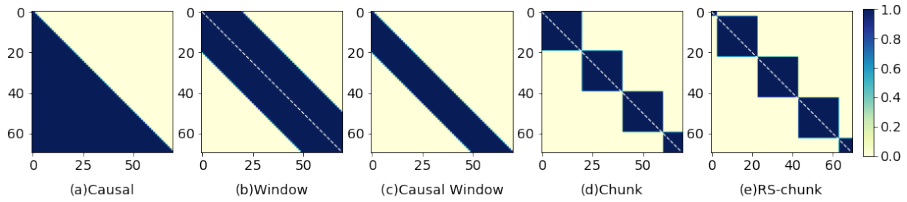

Figure 2: *Different attention masks used are shown in the figure. The x-axis corresponds to transformer key (K) positions and the y axis corresponds to transformer query (Q), both with the same sequence length. In this illustration, a sequence length of 70 is used, with the window size defined as 20. The receptive field of the sequence can be understood by looking at each point on the diagonal and the horizontal points to which it can attend. Different masking offers different control over the receptive field during training.*

ULx, ULy , LLx, LLy , Jawx, Jawy , TTx, TTy , TBx, TBy , TDx, TDy .

The articulatory data is passed through a low-pass filter, with a cut-off frequency 25Hz, to remove high-frequency noise incurred during the measurement process. We then perform sentence-wise mean removal and variance normalisation along each articulator dimension, for all the subjects. The recorded speech is down-sampled from 48kHz to 16kHz, we obtain 13-dimensional MFCC features. We obtain both articulatory trajectories and MFCC at 100Hz and zero pad both the sequences to a maximum length of 400 after removing silence.

3. Streaming Transformer models for AAI

Streaming models are designed to have constant latency irrespective of the sequence length. This can be achieved by passing information from the sequence by non-overlapping windowed chunks. To enable this, modifications are usually necessary during training for the model to achieve comparable performance with the full sequence inference. In this section, we will briefly explain the training approaches with Transformers to improve performance of AAI with streaming inference.

3.1. Transformer Architecture

Transformer [26] neural network is built with a self attention operation. At first, linear projections of the input, Query (Q), Key (K) and Value (V) are obtained through learnable weights W_q, W_k, W_v. Attention matrix A is obtained by the computing dot product of Q and K, followed by a softmax operation. This matrix is then weighted by the value V. This process is known as Self-attention. The self-attention operation is responsible for learning dependencies within a sequence. This allows the model to have a global receptive field over the input, non-autoregressively. The self-attention operation can be repeated multiple times, parallelly. This is known as Multi-Head Attention (MHA).

This is followed by feed-forward networks (FFN), which mixes information learnt over different feature dimensions. Additionally, skip connection from the MHA output to FFN is present and layer norm is used. The MHA and FFN operations constitute a single transformer layer. We stack many such layers sequentially to form out transformer encoder and decoder blocks as shown in Figure 4(a). A transformer model can lose information on the sequence positions due to the self-attention operation. To retain this, sinusoidal positional encoding is used. We use the same transformer model for AAI as utilised in [10].

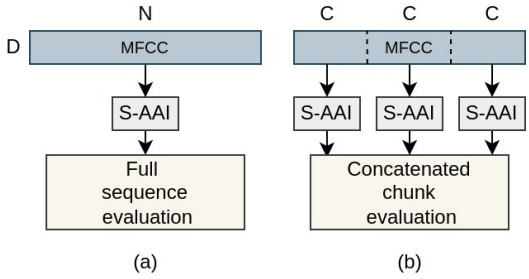

Figure 3: *The figure demonstrates the streaming setup in this work, where N is the sequence length and D is the MFCC dimension and C is the chunk size. (a) refers to a training setup where it is done in full sequence. (b) Decoding is done autoregressively on non-overlapping chunks.*

3.2. Training approaches for streaming

We design our models in such a way that it allows easy transition from the non-streaming model (NS-AAI) to the streaming model (S-AAI). Towards this, we always train on the full sequence by incorporating the different masks to control the attention receptive field in the Transformer. Further to make use of context from previous predictions, we utilize the decoder hidden outputs of transformers. During inference, we perform non-overlapping chunk level evaluation to enable streaming. The details of different attention masks and chunk memory utilized during transformer training are as follows:

Attention mask used during training: Here we present different attention masks used to improve the performance of AAI while streaming with chunks, with no context from any other chunk. Given a sequence of length N, and chunk size C, we can design the different attention masks as follows:

Causal mask: Here, we enforce the criteria that for any point in the sequence n, attention will be computed only on positions in [1, n], while the information from $n+1$ to N positions are not considered. This masking is applied during training since while decoding there is limited context present from the future time steps.
Window mask: This mask is applied such that there is a fixed receptive field on present and future positions in the sequence. We apply the mask such that only [n-C, n+C] positions are considered. Note that in this case, during training every position has a receptive field of $2C+1$.
Causal window mask: This is an extension of the window mask where a causal mask is also applied. The receptive field

will be [n-C, n]. With this masking, the attention on future time steps is removed.

Chunk mask: We use this mask such that the attention receptive field is limited only to the positions inside a chunk. Let C_b be the boundary of any chunk. Then the chunk mask is defined over the interval defined by [$max(C_b,n-C)$, $min(C_b+C,n+C)$].

Random Start Chunk mask (RS-Chunk): This is the same as the previous mask except for that fact that the masking will randomly start from any position i in [$1,C$] and determine the mask as [$max(C_b+i,n-C+i)$, $min(C_b+C+i,n+C+i)$]. This is necessary to capture attention in different parts of the sentence.

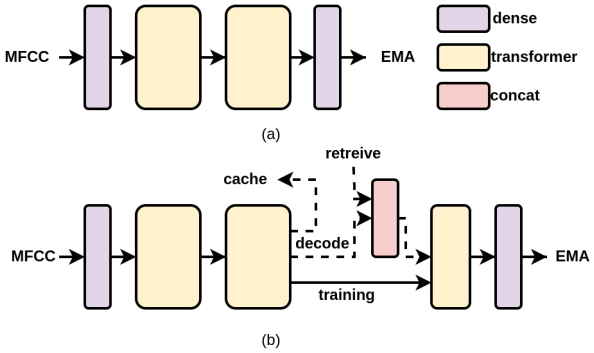

Figure 4: *(a) refers to the standard transformer used. (b) refers to the model with an additional transformer for memory, where chunks can be cached and used during decoding*

Training without attention mask: This is the default case where no changes are made. Transformers are trained in full sequence and decoding is done in chunks. This approach is performed for comparisons.

Utilizing context with an additional memory at decoder: In this approach along with the attention masks, we further utilize the context from the decoder's previous hidden states. This is achieved with a simple memory mechanism to consider the previous chunk context. The decoder output is cached for the next chunk prediction. After the transformer decoder, a small (with two attention heads) additional transformer is used which takes input from the (concatenation of) previous and current decoder outputs, as shown in Figure 4.

We apply the same attention masking procedure at the encoder and decoder of the transformer during training as described earlier. For a small transformer at the end of the decoder, the attention receptive field is applied with a duration double the size C, since there are two chunks (previous and current hidden state outputs from the decoder) of information to be learnt.

3.3. Unidirectional LSTM (UL-AAI)

For a baseline comparison, we use uni-directional LSTM which provides predictions in a streaming manner. We use a 3 layer LSTM with 256 hidden units [27]. We compare the performance of LSTM with Transformer based AAI models.

4. Experimental setup

We pool data from 10 subjects data and use 80% of the data for training, 10% for validation and the remaining 10% for testing, uniformly across subjects. We perform training on the full sequence, as it is done in non-streaming models (NS-AAI). We apply techniques to the model architecture so that it will allow chunk level decoding (S-AAI) with good performance.

Table 1: *The table represents the different models with their configurations. Note that unidirectional LSTM is autoregressive and have memory.*

	model config	sentence level	chunk level	attention masking	memory
T-AAI	Transformers	✓			
SIC-AAI	Transformers		✓	✓	
SM-AAI	Transformers		✓	✓	✓
UL-AAI	LSTM				✓

We experiment with different chunk sizes starting from 200ms to 1sec with increments of 100ms. We evaluate our models with Correlation coefficient (CC) [28] to access the performance of AAI. In Table 1, we use the sentence-level decoded transformer (T-AAI) and unidirectional LSTM (UL-AAI) as our baselines. We further train our masking based transformer models for isolated chunk decoding (SIC-AAI) and chunk memory decoding (SM-AAI).

We also look at Real-time factor (RTF), latency to measure the prediction time in the different models. RTF is a common metric used for measuring latency in speech recognition streaming models. It is defined as the time taken to obtain the prediction divided by the utterance duration. All codes to replicate the experiments are available at https://github.com/bloodraven66/streaming_aai_codes.

5. Results

In this section, we will first go through the baseline transformer (T-AAI) and LSTM (UL-AAI) results. We then talk about streaming results with SIC-AAI and SM-AAI models for different chunk sizes, followed by articulator specific results. We will then analyse the predicted trajectories for streaming, and look at RTF.

5.1. Baselines

We obtain a correlation coefficient of 0.746 for UL-AAI. Whereas for T-AAI, we obtain a correlation coefficient of 0.853. We can observe the drop in performance by using a unidirectional LSTM for streaming. In next sections, we will discuss results with training approaches to improve the streaming performance.

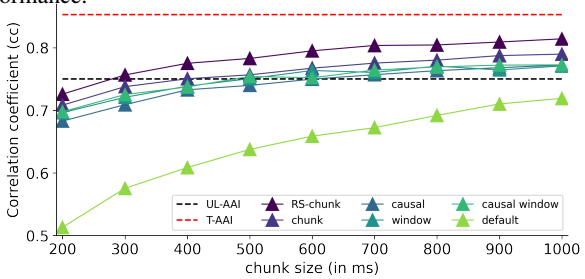

Figure 5: *Correlation coefficient for different SIC-AAI models at chunk sizes ranging from 200ms to 1s. The full sequence evaluation by the baselines UL-AAI and T-AAI is also shown in the figure.*

5.2. SIC-AAI

Figure 5 refers to the isolated chunk transformer decoding of transformer based AAI trained with different attention masks. We can also observe the baselines UL-AAI and reference T-AAI in Figure 5. We notice that default transformer decoding

Table 2: *We report Articulator to specify correlation coefficient values for the upper bound, lower bound models along with the best performing SM-AAI with RS-chunk mask. We report the standard deviation across subjects inside brackets for the average results*

Model	ULx	ULy	LLx	LLy	Jawx	Jawy	TTx	TTy	TBx	TBy	TDx	TDy	Average
UL-AAI	0.64	0.612	0.688	0.751	0.746	0.736	0.792	0.801	0.801	0.796	0.802	0.784	0.745(0.014)
SM-AAI (300ms)	0.693	0.61	0.736	0.808	0.794	0.786	0.852	0.87	0.862	0.851	0.86	0.842	0.797(0.015)
SM-AAI (600ms)	0.701	0.617	0.754	0.818	0.812	0.791	0.863	0.877	0.873	0.861	0.87	0.855	0.807(0.016
SM-AAI (1s)	0.722	0.666	0.774	0.838	0.833	0.813	0.878	0.893	0.887	0.878	0.884	0.873	0.828(0.013
T-AAI	0.758	0.708	0.801	0.863	0.858	0.842	0.898	0.909	0.909	0.897	0.906	0.892	0.853(0.014

performs poorly and the performance drastically increases on applying to mask. We observe that the RS-Chunk mask works best on all chunk sizes. It obtains better performance than UL-AAI from a chunk size of 300ms. Further, we observe that there is a considerable gap between the masking and T-AAI.

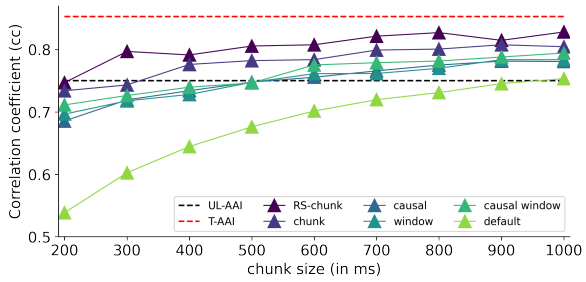

Figure 6: *Correlation coefficient for different SM-AAI models at chunk sizes ranging from 200ms to 1s. The full sequence evaluation by the baselines UL-AAI and T-AAI are also shown in the figure.*

5.3. SM-AAI

Here, we look at how memory improves the performance as shown in Figure 6. We observe that RS-chunk mask performance is close to UL-AAI even at 200ms chunk. We also notice at 600ms chunk, all masking methods perform better than the UL-AAI baseline. Overall we find that there is an improvement with all masking methods, at all chunks compared to SIC-AAI. Finally, we still observe a gap vs the T-AAI.

5.4. Articulator specific results

From Table 2, we observe the performance of the baselines along with the best performing model - RS-chunk mask transformer with memory, at 300ms, 600ms and 1s chunk. The correlation coefficient is noted for each articulator along with the average scores. Though UL_y performance is comparable to that of UL-AAI, there is a significant increase in the results on all

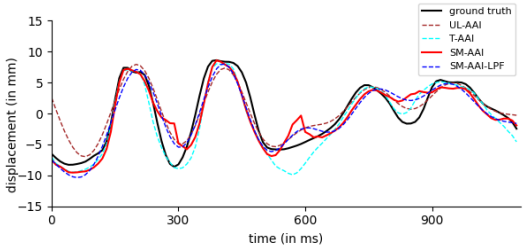

Figure 7: *We show the ground truth as well as estimated articulatory trajectories for the different models, for the sentence "alfalfa is healthy for you" uttered by subject* M1. *We show the baseline UL-AAI and T-AAI trajectories along with the SM-AAI trajectory at 300ms chunking. We also show that low pass filtering decoded output will remove any discontinuity between chunks.*

Table 3: *Real time factor (RTF) for the baselines and best performing SM-AAI models.*

	UL-AAI	SM-AAI (300ms)	SM-AAI (600ms)	SM-AAI (1s)	T-AAI
RTF	0.0028	0.0294	0.0157	0.0103	0.0051

other articulators. The SM-AAI values keep increasing with the increase in chunk size, as noted from the Figures 5, 6. We also note that the best SM-AAI model does not reach the performance of the baseline transformer. This could be due to absence of context from the neighbouring chunks.

5.5. Trajectory analysis

In this section, we observe the ground truth and estimated articulator trajectories as shown in Figure 7. We have the predicted trajectory from transformers, LSTM and 20ms chunk SM-AAI for the same utterance. We observe that while the transformer's trajectory closely resembles that of ground truth, on analysing the SM-AAI trajectory, we notice that there are discontinuities at chunk boundaries. This is expected since we make predictions on non-overlapping chunks. We apply a low pass filter and show that it can smoothen the transition between chunks and can resemble the transformer baseline trajectory.

5.6. RTF

Here, we observe the RTF values for the different models. We notice that UL-AAI has the least score, this is expected since LSTMs have *O(N)* time complexity. The SM-AAI models have higher RTF than T-AAI. This is due to the additional masking operation applied at each attention layer along with the additional transformer block. Even though the baselines have better RTF values, note that they will continue to degrade on much longer sequences, whereas SM-AAI will have the same value.

6. Conclusions

In this work, we investigate the use of the transformer streaming model for AAI. We compare against the baseline streaming model (UL-AAI) and standard transformer model (T-AAI). We use a training regime where the AAI mapping is learnt on a sequence and can be evaluated for streaming with chunk level autoregressive decoding. We apply different attention masks in the transformer and utilise chunk memory, to achieve better performance, and evaluate at different chunk sizes. With our best SM-AAI model, we show that it can beat the baseline UL-AAI, even at a 200ms chunk. Although SM-AAI could not match the baseline transformer (T-AAI) results, we have showed that SM-AAI models are a better alternative for streaming when compared to UL-AAI. We also report the RTF of the decoding and analyse the predicted articulatory trajectories. In the future, we will look at better masking, memory techniques suited for this task and alleviate the use of post-processing for the chunk boundaries.

7. References

[1] P. Denes and E. Pinson, *The speech chain*. Macmillan, 1993.

[2] A. Illa and P. K. Ghosh, "An investigation on speaker specific articulatory synthesis with speaker independent articulatory inversion," in *Proc. Interspeech*, 2019, pp. 121–125. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2664

[3] J. Frankel, K. Richmond, S. King, and P. Taylor, "An automatic speech recognition system using neural networks and linear dynamic models to recover and model articulatory traces," *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China (CD-ROM) 2000.

[4] K. Kirchhoff, "Robust speech recognition using articulatory information," Ph.D. dissertation, University of Bielefeld, 1999.

[5] C. S, C. Yarra, R. Aggarwal, S. K. Mittal, K. N K, R. K T, A. Singh, and P. K. Ghosh, "Automatic visual augmentation for concatenation based synthesized articulatory videos from real-time MRI data for spoken language training," in *Proc. Interspeech*, 2018, pp. 3127–3131.

[6] Z. Wu, K. Zhao, X. Wu, X. Lan, and H. Meng, "Acoustic to articulatory mapping with deep neural network," *Multimedia Tools and Applications*, vol. 74, no. 22, pp. 9889–9907, 2015.

[7] A. Illa and P. K. Ghosh, "Representation learning using convolution neural network for acoustic-to-articulatory inversion," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5931–5935.

[8] P. Liu, Q. Yu, Z. Wu, S. Kang, H. Meng, and L. Cai, "A deep recurrent approach for acoustic-to-articulatory inversion," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4450–4454.

[9] N. Bozorg and M. T. Johnson, "Acoustic-to-articulatory inversion with deep autoregressive articulatory-wavenet," in *INTERSPEECH*, 2020.

[10] S. Udupa, A. Roy, A. Singh, A. Illa, and P. K. Ghosh, "Estimating articulatory movements in speech production with transformer networks," in *Interspeech*, 2021.

[11] ——, "Web-interface for articulatory movement estimation in speech production," in *Interspeech*, 2021.

[12] B. Li, S.-y. Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards fast and accurate streaming end-to-end asr," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6069–6073.

[13] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," 2020. [Online]. Available: https://arxiv.org/abs/2010.11395

[14] B. Li, A. Gulati, J. Yu, T. N. Sainath, C.-C. Chiu, A. Narayanan, S.-Y. Chang, R. Pang, Y. He, J. Qin, W. Han, Q. Liang, Y. Zhang, T. Strohman, and Y. Wu, "A better and faster end-to-end model for streaming asr," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5634–5638.

[15] N. Moritz, T. Hori, and J. Le Roux, "Triggered attention for end-to-end speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5666–5670.

[16] H. Inaguma, Y. Gaur, L. Lu, J. Li, and Y. Gong, "Minimum latency training strategies for streaming sequence-to-sequence asr," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6064–6068.

[17] H. Miao, G. Cheng, C. Gao, P. Zhang, and Y. Yan, "Transformer-based online ctc/attention end-to-end speech recognition architecture," 2020. [Online]. Available: https://arxiv.org/abs/2001.08290

[18] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6074–6078.

[19] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6783–6787.

[20] S. Zhang, Z. Gao, H. Luo, M. Lei, J. Gao, Z. Yan, and L. Xie, "Streaming chunk-aware multihead attention for online end-to-end speech recognition," 2020. [Online]. Available: https://arxiv.org/abs/2006.01712

[21] B. Zhang, D. Wu, Z. Yao, X. Wang, F. Yu, C. Yang, L. Guo, Y. Hu, L. Xie, and X. Lei, "Unified streaming and non-streaming two-pass end-to-end model for speech recognition," 2020. [Online]. Available: https://arxiv.org/abs/2012.05481

[22] S. Zhang, Z. Gao, H. Luo, M. Lei, J. Gao, Z. Yan, and L. Xie, "Streaming chunk-aware multihead attention for online end-to-end speech recognition," 2020. [Online]. Available: https://arxiv.org/abs/2006.01712

[23] A. Wrench, "MOCHA-TIMIT," Department of Speech and Language Sciences, Queen Margaret University College, Edinburgh, speech database, 1999. [Online]. Available: http://sls.qmuc.ac.uk

[24] "3d electromagnetic articulograph," available online: http://www.articulograph.de/, last accessed: 4/2/2020.

[25] A. Illa and P. K. Ghosh, "Low resource acoustic-to-articulatory inversion using bi-directional long short term memory," in *Proc. Interspeech*, 2018, pp. 3122–3126.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[27] A. Illa and P. K. Ghosh, "Closed-set speaker conditioned acoustic-to-articulatory inversion using bi-directional long short term memory network," *The Journal of the Acoustical Society of America*, vol. 147, no. 2, pp. EL171–EL176, 2020. [Online]. Available: https://doi.org/10.1121/10.0000738

[28] P. K. Ghosh and S. Narayanan, "A generalized smoothness criterion for acoustic-to-articulatory inversion," *The Journal of the Acoustical Society of America*, vol. 128, no. 4, pp. 2162–2172, 2010. [Online]. Available: https://asa.scitation.org/doi/abs/10.1121/1.3455847