INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Flexi-Transducer: Optimizing Latency, Accuracy and Compute for Multi-Domain On-Device Scenarios

Jay Mahadeokar, Yangyang Shi, Yuan Shangguan*, Chunyang Wu*, Alex Xiao, Hang Su, Duc Le, Ozlem Kalinli, Christian Fuegen, Michael L. Seltzer*

Facebook AI, USA

jaym@fb.com

Abstract

Often, the storage and computational constraints of embedded devices demand that a single on-device ASR model serve multiple use-cases / domains. In this paper, we propose a *Flexible Transducer* (FlexiT) for on-device automatic speech recognition to flexibly deal with multiple use-cases / domains with different accuracy and latency requirements. Specifically, using a single compact model, FlexiT provides a fast response for *voice commands*, and accurate transcription but with more latency for *dictation*. In order to achieve flexible and better accuracy and latency trade-offs, the following techniques are used. Firstly, we propose using domain-specific altering of segment size for Emformer encoder that enables FlexiT to achieve flexible decoding. Secondly, we use Alignment Restricted RNNT loss to achieve flexible fine-grained control on token emission latency for different domains. Finally, we add a domain indicator vector as an additional input to the FlexiT model. Using the combination of techniques, we show that a single model can be used to improve WERs and real time factor for dictation scenarios while maintaining optimal latency for voice commands use-cases.

Index Terms: Speech recognition, RNN-T, Transformers

1. Introduction

On-device automatic speech recognition (ASR) models have been enabled on many embedded devices, including mobile phones, smart speakers, and watches [1–3]. On one hand, on-device ASR eliminates the need to transfer audio and recognition results between devices and a server, thus enabling fast, reliable, and privacy-preserving speech recognition experiences. On the other hand, these devices operate with significant hardware constraints: e.g., memory, disk space, and battery consumption. Moreover, the embedded ASR models often serve multiple applications: e.g., video transcription, dictation, and voice commands. Each of these applications has its latency and accuracy requirements. For example, voice assistants demand an ASR model with low latency to respond to user queries as fast as possible. While server-based ASR might rely on running different models for different applications – more compact models for voice assistants and big, semi-streaming models [4,5] for dictation– the on-device environment prohibits such practice. Due to hardware constraints, and varied requirements of different applications optimizing the model size, compute and accuracy of one single ASR model becomes challenging. In this paper, we take a close look at the scenario where device constrained ASR model needs to be optimized for two different use-cases.

The first use case is voice commands, where the latency requirement is strict. Users expect immediate device responses when they ask the speech assistant to turn on the lights or play a song. The second application uses the ASR model for dictation or audio transcription, where accuracy is more important than the model's latency. Recurrent Neural Network Transducer (RNN-T) framework [1,6] is widely adopted to provide streaming ASR transcriptions for both voice commands [7,8] and dictation applications [9,10].

We focus on Emformer model [11] as an audio encoder for RNN-T, which uses both contextual audio information (in the form of an audio chunk) and future audio context (in the form of model look-ahead). A larger model look-ahead permits the model to access more future context and optimize ASR accuracy but hurts model latency.

This paper proposes a Flexi-Transducer (FlexiT) model that answers the requirements of two streaming ASR use-cases while still staying as one compact model. Further we also show that larger look-ahead enables improve the compute / real time factor trade-offs which help battery consumption.

2. Related Work

Inspired by the successful application of transformer [12], many works in ASR also adopted transformer across different model paradigms, such as the hybrid systems [13–16], the encoder-decoder with attention [17–21] and the sequence transducers [5, 22, 23]. In this work, we follow the neural transducer paradigm using Emformer [11, 16] and alignment restricted transducer loss [24].

Many ASR applications demand real-time low latency streaming. The block processing method [11,16,25] with attention mask modifies the transformer to support streaming applications. In the block processing, self-attention's receptive field consists of one fixed-size chunk of speech utterance and its historical context and a short window of future context. However, the fixed chunk size limits the encoder's flexibility to trade-off latency, real-time factor, and accuracy.

A unified framework is proposed in [26] to train one single ASR model for both streaming and non-streaming speech recognition applications. In [4] cascaded encoders are used to build a single ASR model that operates in streaming and non-streaming mode. These approaches support one fixed latency for the streaming use-case, and the other use-case is strictly non-streaming. In this paper, we tackle scenarios to support two different streaming ASR use-cases with different latency constraints.

More flexible latency is achieved by [27–29] where, in the training phase, the model is exposed to audio context variants up to the whole utterance length. In [27], the authors show that asynchronous revision during inference with convolutional encoders can be used to achieve dynamic latency ASR. The context size selection proposed in these works is purely random during training. In this work, besides random selection, we also

*Equal Contribution

http://dx.doi.org/10.21437/Interspeech.2021-1921

explore context size selection based on a priori knowledge about the targeted use cases (domains). The use of domain information to improve the ASR performance of a single model being used to serve different dialects, accents, or use-cases has been studied previously in [10, 30, 31]

For RNN-T models, both the encoder's context size and the potential delay of token emissions contribute to model latency. It is well known that streaming RNN-T models tend to emit ASR tokens with delay. Techniques like Ar-RNN-T [24], Fast-emit [32], constrained alignment approach [33, 34] or late alignment penalties [35] are used to mitigate token delays. This work further extends the alignment restricted transducer [24] with task-dependent right buffer restriction to control the token emission latency for different use-cases.

3. Methodology

In this section, we outline the three proposed techniques for FlexiT. Note that we focus on demonstrating these techniques on an Emformer [11], but the techniques could be extended to other encoder architectures that support dynamic segment size selection.

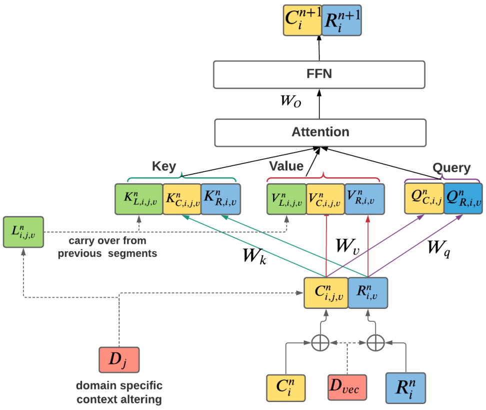

Figure 1: *Illustration for domain-specific context size and the injection of domain vectors in an Emformer layer*

3.1. Domain Specific Segments in Emformer Encoder

FlexiT encoder is shown in Figure 1, with notations similar to those in [11]. Input to the Emformer layer concatenates a sequence of audio features into segments $C_i^n, \ldots C_{I-1}^n$, where i is the index of a segment and n the layer's index. The corresponding left and right contextual blocks L_i^n and R_i^n are concatenated together with C_i^n to form contextual segment $X_i^n = [L_i^n, C_i^n, R_i^n]$. In FlexiT, we do not use the memory vector in the original Emformer layers. We propose to dynamically alter contextual block L_i^n, C_i^n by selecting the number of segments I dynamically. We explore both random context segment selection, as well as domain-dependent context segment selection during training.

During training, we ensure every input batch consists of utterances from the same domain. Let D_j denote the domain being used for training the current batch. A domain-specific context altering operation is performed such that the vector X_i^n is modified to $[L_{i,j}^n, C_{i,j}^n, R_i^n]$, conditioned on D_j. More concretely, according to the desired domain specific context size, C_i^n is split into $[C_{iLeft}^n, C_{iRight}^n]$. We then set $L_{i,j}^n = [L_i^n, C_{iLeft}^n]$, and $C_{i,j}^n = C_{iRight}^n$. The domain-specific selection of $C_{i,j}^n$ provides flexible latency for decoding and, at the same time, as suggested by our results in Section 5.3, helps to improve the speech recognition model's robustness.

3.2. Adding Domain Vector in Emformer Encoder

To improve the ASR model's capability to learn domain-specific features, we append a domain vector to inputs of each layer of the Emformer encoder. The domain vector is simply represented by using 1-hot representation [30], the value of which depends on whether the training sample comes from V_{Cmd} or Dictation domain. More concretely, as illustrated in Figure 1 let D_{vec} denote the 1-hot domain vector representation. We concatenate D_{vec} to all components of X_i^n, to obtain concatenated input vectors $[L_{i,j,v}^n, C_{i,j,v}^n, R_{i,v}^n]$. These are used as inputs to the n^{th} Emformer layer while training.

3.3. Domain Specific Alignment Restrictions Loss

In [24] using pre-computed token level alignment information, configurable thresholds b_l, b_r are used to restrict the alignment paths used for RNN-T loss computation during training. Note that the right-buffer b_r can be made stricter to ensure earlier token emissions, but stricter b_r also leads to increased WER. In this work, our goal is to optimize the WERs for the dictation domain while maintaining low latency for the V_{Cmd} domain. Therefore, we propose using domain-specific alignment restriction thresholds while optimizing the loss. We analyze domain-specific WER and token emission latency in Section 5.

4. Experimental Setup

4.1. Datasets

4.1.1. Training Data

We run our experiments on data-sets that contains 20K hours of human-transcribed data from 2 different domains.

Voice Commands (V_{Cmd}) dataset combines two sources. The first source is in-house, human transcribed data recorded via mobile devices by 20k crowd-sourced workers. The data is deidentified and aggregated with personally identifiable information (PII) removed. We distort the collected audio using simulated reverberation and add randomly sampled additive background noise extracted from publicly available videos. The second source came from 1.2 million voice commands (1K hours), sampled from production traffic with PII removed, audio deidentified and aggregated, and morphed. Speed perturbations [36] are applied to this dataset to create two additional training data at 0.9 and 1.1 times the original speed. We applied distortion and additive noise to the speed perturbed data. From the corpus, we randomly sampled 10K hours.

Dictation (open-domain) dataset consists of 13K hours of data sampled from English public videos that are deidentified and aggregated with PII removed and annotator transcribed. We first apply the same above-mentioned distortions and then randomly sample 10k hours of the resultant data.

4.1.2. Evaluation Datasets

For evaluation, we use the following datasets, representing two different domains:

Voice Commands evaluation set consists of 15K hand-transcribed deidentified and aggregated utterances from volunteer participants as part of an in-house pilot program.

Dictation evaluation set consists of 66K hand-transcribed deidentified and aggregated utterances from vendor collected data where speakers were asked to record unscripted open domain dictation or voice conversations.

4.2. Evaluation Metrics

To measure the model's performance and analyze trade-offs, we track the following metrics:

Accuracy: We use word-error-rate (WER) to measure model accuracy on evaluation sets. Note that we measure the WERs for dictation domain without end-pointer and the V_{Cmd} domain with end-pointer. We also keep track of the deletion errors (DEL), which are proportional to early cutoffs in V_{Cmd}.

Latency: We measure model latency on V_{Cmd} domain using following metrics:

1. *Token Finalization Delay (FD):* as defined in [24] is the audio duration between the time when user finished speaking the ASR token, and the time when the ASR token was surfaced as part of 1-best partial hypothesis, also referred as emission latency in [26], or user-perceived latency in [37].
2. *Endpointing Latency (L):* [24, 38] is defined as the audio time difference between the time end-pointer makes end-pointing decision and the time user stops speaking.

We track the Average Token Finalization Delay and Average Endpointing latency (L_{Avg}) metrics. In all experiments, we use a fixed neural end-pointer (NEP) [38], running in parallel to ASR being evaluated every 60ms to measure V_{Cmd} domain metrics. A detailed study with other end-pointing techniques besides NEP (static, E2E [39]) is beyond the scope of this paper.

ASR Compute: On device power consumption / battery usage is usually well co-related with the amount of compute being used by the ASR model. We use the real time factor (RTF) measured on a real android device as an indirect indicator of the of the model's compute usage.

4.3. Experiments

We use the RNN-T model with Emformer encoder [40], LSTM with layer norm as predictor, and a joiner with 45M total model parameters. As inputs, we use 80-dim log Mel filter bank features at a 10 ms frame rate. We also apply SpecAugment [28] without time warping to stabilize the training. We use a stride of 6 and stack 6 continuous vectors to form a 480 dim vector projected to a 512 dim vector using a linear layer. The model has 10 Emformer layers, each with eight self-attention heads and 512 dimension output. The inner-layer has a 2048 dimension FFN with a dropout of 0.1. We use Alignment Restricted RNN-T loss [24] using a fixed left-buffer b_l of 300ms, while varying right-buffer b_r parameter. All models are trained for 45 epochs using a tri-stage LR scheduler with ADAM optimizer and base LR of 0.005. We study the best way to integrate domain information into FlexiT by experimenting with how domain-conditioned Ar-RNN-T buffer sizes, Emformer context sizes (*Emf Ctx*), the domain one-hot vector input, and their combined interactions work to improve the model's performance in V_{Cmd} and dictation domains.

1. **Fixed Emformer Context:** We first fix the *Emf Ctx* per experiment and analyze how Ar-RNN-T right buffer size r_b and Domain Vector impact model performances.

(a) *Fixed Ar-RNN-T without Domain Vector:* As baselines (B_1-B_3), we train models using 120, 300 and 600ms *Emf Ctx*. We sweep the range of Ar-RNN-T b_r with (120, 300, 420) ms, (300, 600, 900) ms, (600, 900, 1200) ms respectively.

(b) *Domain Ar-RNN-T without Domain Vector:* In this experiment, we study if domain specific Ar-RNN-T b_r sizes help to improve the WER-latency trade-off. We train models C_2,C_3 similar to B_2,B_3 but also adding domain specific Ar-RNN-T b_r of (420, 600) while for B_3 we use b_r of (420, 900) for voice commands / dictation domains.

(c) *Fixed Ar-RNN-T with Domain Vector:* We also train models ($D_1 - D_3$) where we concatenate domain vector to Emformer layer inputs. The models are trained using 120, 300 and 600ms *Emf Ctx* and Ar-RNN-T b_r of 420ms, 600ms and 900ms respectively.

(d) *Domain Ar-RNN-T with Domain Vector:* To analyze if domain vector enables the model to learn to emit tokens with different latency, we also train a models E_2,E_3 using similar configuration as D_2,D_3 but also adding domain specific Ar-RNN-T threshold b_r of (420, 600) for E_2 (420, 900) for E_3.

2. **Random / Domain Specific Emformer Context:** We analyze randomly selected *Emf Ctx* during training as described in 3.1. Context size is randomly selected from 120ms to 1200ms. As shown later in Section 5 we find domain specific Ar-RNN-T b_r of 420, 900ms achieves best results. Therefore, to reduce the number of combinations in this experiment, we fix b_r of 420, 900ms and run experiments R_1 and R_2 without and with use of Domain vector respectively. Lastly, we analyze using domain specific *Emf Ctx* during training using domain-specific Ar-RNN-T b_r of 420, 900ms. Corresponding experiments S_1 and S_2 without and with Domain vector respectively.

Inference: We always evaluate models such that for each domain, the training time *Emf Ctx* matches the context provided to the encoder while doing inference. Only exception being Random Emformer Context experiments R_1 and R_2 where we use inference context size of 120ms, 600ms for V_{Cmd} and dictation domains respectively.

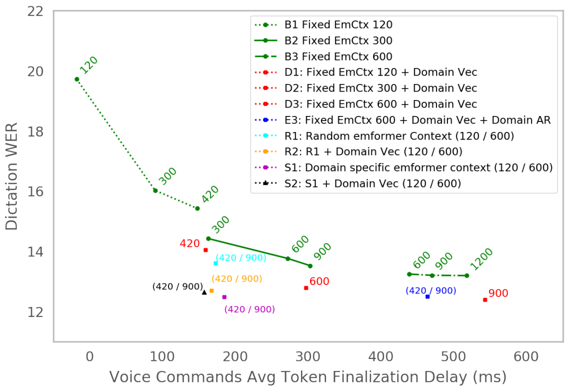

Figure 2: *Dictation WERs and V_{Cmd} token finalization delay tradeoffs for various experiments. The labels in the plot show the Ar-RNN-T b_r used for the experiment. Increasing b_r as well as Emformer context improves WER, but degrades latency. Methods (R_2, S_2) achieve best trade-offs.*

5. Results and Analysis

5.1. Emformer Context and Ar-RNN-T Thresholds

Figure 2 demonstrates the trade-offs as we vary Emformer context and the Ar-RNN-T right-buffer b_r. Dictation WER improves as we increase the Emformer context in experiments B_1-B_3. Similarly, for a fixed Emformer context, WERs improve with larger Ar-RNN-T b_r parameter. Note that only hand-picked variants are detailed in Table 1.

Table 1: *Fixed Emformer Context without Domain Vector: Dictation WERs improve with larger Emf Ctx. V_{Cmd} WERs and Deletions with neural endpointer, also increase due to increased Avg(FD).*

	Emf	**AR** b_r	**Dict**	V_{Cmd}	V_{Cmd}	**Avg**	L_{Avg}
	Ctx		**WER**	**WER**	**DEL**	**FD**	
B_1	120	420	15.4	6.7	2.8	148	449
B_2	300	600	13.8	7.4	3.6	272	463
B_3	600	900	13.2	9.7	5.4	470	505
C_2	300	420/600	13.7	7.5	3.6	271	482
C_3	600	420/900	13.2	10.8	6.4	483	548

On the other hand, V_{Cmd} domain's L_{avg} also increases. To achieve low L_{Avg} throughout experiments, which is important for a better user experience, we use a fixed NEP. Delays in token emissions result in more early cuts and increased deletion errors as shown in Table 1,2. To achieve best latency and simultaneously reduce early cuts, we must maintain a smaller Emformer context and maintain a strict Ar-RNN-T b_r parameter. Therefore, experiments for random and domain-specific Emformer context are performed with b_r 420 and context 120.

5.2. Domain Vector and Domain specific Ar-RNN-T

Table 2: *Fixed Emformer Context with Domain Vector: With Domain vector D_1-D_3 achieve better dictation WERs compared to B_1-B_3. Further, with Domain specific Ar-RNN-T, E_2,E_3 achieve better latency compared to D_2,D_3.*

	Emf	**AR** b_r	**Dict**	V_{Cmd}	V_{Cmd}	**Avg**	L_{Avg}
	Ctx		**WER**	**WER**	**DEL**	**FD**	
D_1	120	420	14.0	6.8	2.9	159	441
D_2	300	600	12.8	7.7	3.8	297	464
D_3	600	900	12.4	10.5	6.2	543	509
E_2	300	420/600	12.8	7.23	3.28	263	457
E_3	600	420/900	12.5	9.6	5.7	464	476

We observe that providing domain vector in encoder improves the WERs for dictation domain in general for all experiments as shown in Table 2, which is consistent with previous works [10,30]. Unfortunately, the WER improvements come in tandem with an increased average FD of V_{Cmd} when the models are trained with domain vector (comparing experiments B_1-B_3 and D_1-D_3). We hypothesize that this is because in the absence of stricter AR-RNNT r_b restrictions for the V_{Cmd} domain, the model learns to delay token emissions to improve accuracy.

Ar-RNN-T helps achieve fine grained control on token delays [24]. However, in multi-domain setting, comparing C_2,C_3 to B_2,B_3, we observe that simply imposing domain specific Ar-RNN-T thresholds does not improve V_{Cmd} FD. The use of domain vector, alongside domain specific Ar-RNN-T thresholds, enables us to achieve a more refined control over V_{Cmd} domain's FD. This is demonstrated in Table 2 where D_1-D_3 have larger Avg(FD) compared to B_1-B_3, but models E_2-E_3 learn to explicitly emit V_{Cmd} domain tokens earlier than C_2-C_3.

5.3. Random / Domain Specific Emformer Context

Results from random context training, R_1 suggests that in the absence of domain information, the model achieves worse trade-offs than V_{Cmd} FD of B_1 and Dictation WER of $B3$. In R_2 adding domain vector, improves the WER for dictation domain significantly which is consistent with Section 5.2. Overall, R_2 achieves better trade-offs for optimizing both use-cases.

Table 3: *Random and Domain Specific Emformer Context: Experiments use 420 / 900 Ar-RNN-T r_b parameters and 120 / 600 $EmCtx$ during inference.*

	Emf	D_{vec}	**Dict**	V_{Cmd}	V_{Cmd}	**Avg**	L_{Avg}
	Ctx		**WER**	**WER**	**DEL**	**FD**	
R_1	Random	No	13.6	7.2	3.1	173	440
R_2		Yes	12.7	7.1	3.1	167	440
S_1	120/600	No	12.5	7.7	3.3	185	458
S_2		Yes	12.6	7.0	2.9	157	450

Experiment S_1, which combines domain-specific *Emf Ctx* and uses domain-specific Ar-RNN-T achieves dictation WERs comparable to D_3 while enabling reasonable FD for V_{Cmd} domain, which improves on the results of experiment R_1 with random *Emf Ctx*. Therefore, we argue that domain-specific *Emf Ctx*, which enables dynamic attention masking per domain, already helps the model learn more robust domain-specific features, even in the absence of a domain vector.

Finally, similar to Section 5.2 further addition of domain vector in experiment S_2 also enables the model to achieve better FD, thus improving the deletion errors from model S_1. This achieves the best tradeoffs in dictation domain WER, and V_{Cmd} domain FD and WER with endpointing enabled.

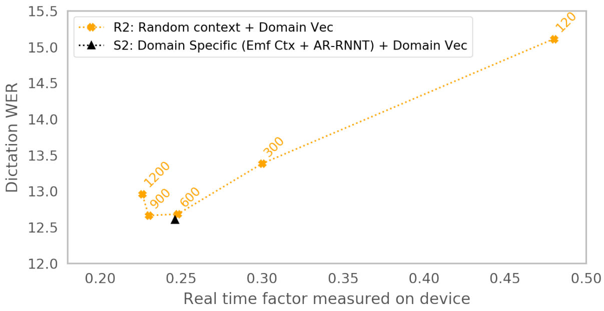

Figure 3: *Dictation WERs and Real Time Factor measured on an android device. The labels show chunk size (ms) used during inference, S_2 was evaluated with 600ms chunk size.*

In Figure 3 we analyze the RTF of FlexiT models. For R_2 we evaluate the RTF and WERs while varying inference context size. We observe that the RTF reduces with larger context size, which is mainly because of improved batching inside Emformer layers across time dimension.

6. Conclusion

This paper proposed a single *Flexi-Transducer* (FlexiT) model that supports domain-dependent trade-off of latency and accuracy. The domain-specific or random context modeling is achieved jointly via segment size altering operation for encoder and the domain vector. Ar-RNN-T loss imposes a domain-specific constraint to limit the token emission latency for different domains. Using the combination of techniques, we achieve better WER, RTF and latency trade-offs when a single model supports multiple streaming ASR use-cases.

7. Acknowledgment

We'd like to thank Rohit Prabhavalkar and Jiatong Zhou for valuable discussions and advice.

8. References

[1] Y. He, T. N. Sainath, R. Prabhavalkar, and Others, “Streaming End-to-end Speech Recognition for Mobile Devices,” in *Proc. ICASSP*, 2019.

[2] K. Kim, K. Lee, D. Gowda, and Others, “Attention based on-device streaming speech recognition with large speech corpus,” in *Proc. ASRU*, 2020.

[3] G. Venkatesh, A. Valliappan, J. Mahadeokar, and Other, “Memory-efficient Speech Recognition on Smart Devices,” *arXiv preprint arXiv:12102.11531*, 2021.

[4] A. Narayanan, T. N. Sainath, R. Pang, and Others, “Cascaded encoders for unifying streaming and non-streaming ASR,” *arXiv preprint arXiv:2010.14606*, 2020.

[5] C. F. Yeh, Y. Wang, Y. Shi, C. Wu, F. Zhang, and Others, “Streaming attention-based models with augmented memory for end-to-end speech recognition,” in *Proc. SLT*, 2020.

[6] A. Graves, “Sequence Transduction with Recurrent Neural Networks,” *arXiv preprint arXiv:1211.3711*, 2012.

[7] Y. Shangguan, J. Li, Q. Liang, R. Alvarez, and I. McGraw, “Optimizing speech recognition for the edge,” in *MLSys On-device Intelligence Workshop*, 2020.

[8] J. Guo, G. Tiwari, J. Droppo, M. Van Segbroeck, C.-W. Huang, A. Stolcke, and R. Maas, “Efficient minimum word error rate training of rnn-transducer for end-to-end speech recognition,” in *Proc. of Interspeech*, 2020.

[9] Y. Shangguan, K. Knister, Y. He, I. McGraw, and F. Beaufays, “Analyzing the Quality and Stability of a Streaming End-to-End On-Device Speech Recognizer,” in *Proc. of Interspeech*, 2020.

[10] T. N. Sainath, Y. He, B. Li, and Others, “A streaming on-device end-to-end model surpassing server-side conventional model quality and latency,” 2020.

[11] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, and Others, “Emformer: Efficient Memory Transformer Based Acoustic Model For Low Latency Streaming Speech Recognition,” in *Proc. ICASSP*, 2021.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. NIPS*, 2017.

[13] D. Povey, H. Hadian, P. Ghahremani, and Others, “A time-restricted self-attention layer for asr,” in *Proc. ICASSP*, 2018.

[14] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang, C. Fuegen, G. Zweig, and M. L. Seltzer, “Transformer-Based Acoustic Modeling for Hybrid Speech Recognition,” in *Proc. ICASSP*, 2019.

[15] F. Zhang, Y. Wang, X. Zhang, C. Liu, Y. Saraf, and G. Zweig, “Fast, Simpler and More Accurate Hybrid ASR Systems Using Wordpieces,” *InterSpeech*, 2020. [Online]. Available: http://arxiv.org/abs/2005.09150

[16] Y. Wang, Y. Shi, F. Zhang, and Others, “Transformer in action: a comparative study of transformer-based acoustic models for large scale speech recognition applications,” *Proc. ICASSP*, 2020.

[17] L. Dong, S. Xu, and B. Xu, “Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition,” in *Proc. ICASSP*, 2018.

[18] S. Karita, N. Chen, T. Hayashi, and Others, “A Comparative Study on Transformer vs RNN in Speech Applications,” *arXiv preprint arXiv:1909.06317*, 2019.

[19] M. Sperber, J. Niehues, G. Neubig, and Others, “Self-attentional acoustic models,” *arXiv preprint arXiv:1803.09519*, 2018.

[20] S. Zhou, L. Dong, S. Xu, and B. Xu, “Syllable-based sequence-to-sequence speech recognition with the transformer in mandarin Chinese,” *arXiv preprint arXiv:1804.10752*, 2018.

[21] C. Wang, Y. Wu, S. Liu, J. Li, L. Lu, G. Ye, and M. Zhou, “Low Latency End-to-End Streaming Speech Recognition with a Scout Network,” *arXiv preprint arXiv:12003.10369*, 2020.

[22] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, “Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss,” in *Proc. ICASSP*, 2020.

[23] C.-F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen, and M. L. Seltzer, “Transformer-Transducer: End-to-End Speech Recognition with Self-Attention,” *arXiv preprint arXiv:11910.12977*, 2019.

[24] J. Mahadeokar, Y. Shangguan, D. Le, and Others, “Alignment restricted streaming recurrent neural network transducer,” in *Proc. SLT*, 2021.

[25] L. Dong, F. Wang, and B. Xu, “Self-attention aligner: A latency-control end-to-end model for asr using self-attention network and chunk-hopping,” in *Proc. of ICASSP*, 2019.

[26] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, “Dual-mode asr: Unify and improve streaming asr with full-context modeling,” 2021.

[27] M. Huang, M. Cai, J. Zhang, Y. Zhang, Y. You, Y. He, and Z. Ma, “Dynamic latency speech recognition with asynchronous revision,” 2020. [Online]. Available: http://arxiv.org/abs/2011.01570

[28] A. Tripathi, J. Kim, Q. Zhang, H. Lu, and H. Sak, “Transformer transducer: One model unifying streaming and non-streaming speech recognition,” *arXiv*, 2020.

[29] B. Zhang, D. Wu, C. Yang, X. Chen, and Others, “WeNet: Production First and Production Ready End-to-End Speech Recognition Toolkit,” *arXiv preprint arXiv:2102.01547*, 2021.

[30] B. Li, T. N. Sainath, K. C. Sim, M. Bacchiani, E. Weinstein, P. Nguyen, Z. Chen, Y. Wu, and K. Rao, “Multi-dialect speech recognition with a single sequence-to-sequence model,” 2017.

[31] S. Zhou, S. Xu, and B. Xu, “Multilingual end-to-end speech recognition with a single transformer on low-resource languages,” 2018.

[32] J. Yu, C.-C. Chiu, B. Li, S. yiin Chang, T. N. Sainath, Y. He, A. Narayanan, W. Han, A. Gulati, Y. Wu, and R. Pang, “Fastemit: Low-latency streaming asr with sequence-level emission regularization,” 2021.

[33] T. N. Sainath, R. Pang, D. Rybach, B. Garcıa, and T. Strohman, “Emitting word timings with end-to-end models,” in *Proc. of Interspeech*, 2020.

[34] H. Sak, A. Senior, K. Rao, and F. Beaufays, “Fast and accurate recurrent neural network acoustic models for speech recognition,” 2015.

[35] B. Li, S. yiin Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, “Towards fast and accurate streaming end-to-end asr,” 2020.

[36] T. Ko, V. Peddinti, D. Povey, and Others, “Audio augmentation for speech recognition,” in *Proc. INTERSPEECH*, 2015.

[37] V. Pratap, Q. Xu, J. Kahn, G. Avidov, T. Likhomanenko, A. Hannun, V. Liptchinsky, G. Synnaeve, and R. Collobert, “Scaling up online speech recognition using convnets,” *arXiv preprint arXiv:2001.09727*, 2020.

[38] M. Shannon, G. Simko, S. yiin Chang, and C. Parada, “Improved end-of-query detection for streaming speech recognition,” in *Proc. of Interspeech*, 2017.

[39] S. Chang, R. Prabhavalkar, Y. He, T. N. Sainath, and G. Simko, “Joint endpointing and decoding with end-to-end models,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5626–5630.

[40] Y. Shi, Y. Wang, C. Wu, and Others, “Weak-Attention Suppression For Transformer Based Speech Recognition,” in *Proc. INTERSPEECH*, 2020.