XLSR-Transducer: Streaming ASR for Self-Supervised Pretrained Models

Shashi Kumar[1,2], Srikanth Madikeri[1,3], Juan Zuluaga-Gomez[1,2], Esaú Villatoro-Tello[1], Iuliia Thorbecke[1,3]
Petr Motlicek[1,4], Manjunath K E[5] and Aravind Ganapathiraju[5]

[1]*Idiap Research Institute, Switzerland*; [2]*EPFL, Switzerland*; [3]*University of Zurich, Switzerland*;
[4]*Brno University of Technology, Czech Republic*; [5]*Uniphore, India*

Abstract—Self-supervised pretrained models exhibit competitive performance in automatic speech recognition (ASR) on finetuning, even with limited in-domain supervised data. However, popular pretrained models are not suitable for streaming ASR because they are trained with full attention context. In this paper, we introduce XLSR-Transducer, where the XLSR-53 model is used as encoder in transducer setup. Our experiments on the AMI dataset reveal that the XLSR-Transducer achieves 4% absolute WER improvement over Whisper large-v2 and 8% over a Zipformer transducer model trained from scratch. To enable streaming capabilities, we investigate different attention masking patterns in the self-attention computation of transformer layers within the XLSR-53 model. We validate XLSR-Transducer on AMI and 5 languages from CommonVoice under low-resource scenarios. Finally, with the introduction of attention sinks, we reduce the left context by half while achieving a relative 12% improvement in WER.

Index Terms—streaming ASR, self-supervised learning, XLSR, transformer transducer

I. INTRODUCTION

End-to-End (E2E) modeling approaches for automatic speech recognition (ASR) have become ubiquitous in literature. There are three popular architecture for achieving E2E models: encoder-decoder (AED) based models [1]–[4], using Connectionist Temporal Classification (CTC) loss [5], and neural Transducer [6], [7] based models. The transducer models, shown in Figure 1a, consist of an encoder, predictor and joint networks. Using Transformers [8] encoder, termed as Transformer Transducer (TT) [9], [10], is a popular choice for streaming ASR [11]–[13] because of its inherently streaming nature. It is common to train TT models from scratch which requires large amount of supervised data [11], [12]. Self-supervised (SSL) models have shown strong ASR performance when trained with small amount of in-domain supervised data [14]–[18]. Most of the recent works using SSL models for ASR use encoder-decoder setup [19]–[21] or CTC loss [14], [15], [22], [23]. In this paper, we present XLSR-Transducer where we use a pretrained XLSR-53 model as encoder in the TT architecture. This opens the door to streaming TT systems for low-resource applications.

In streaming ASR, partial hypotheses are generated for each audio chunk sequentially [12], [24] to produce the transcript for the full audio, whereas the entire audio segment is available for non-streaming decoding. Depending on the latency requirements, the chunk size may vary from few hundred milliseconds to few seconds [24]. Typically, a drastic degradation in word error rate (WER) is observed when non-streaming models are decoded in streaming fashion [13], because only a limited context is available. In this work, we propose a variety of attention masking patters that enables streaming training and decoding of our XLSR-Transducer model. We also study the importance of chunk sizes and past context by varying them during inference. Increasing past context typically enhances ASR performance [24], at the expense of increased latency. Recently, it was shown that the transformer layers learn to assign
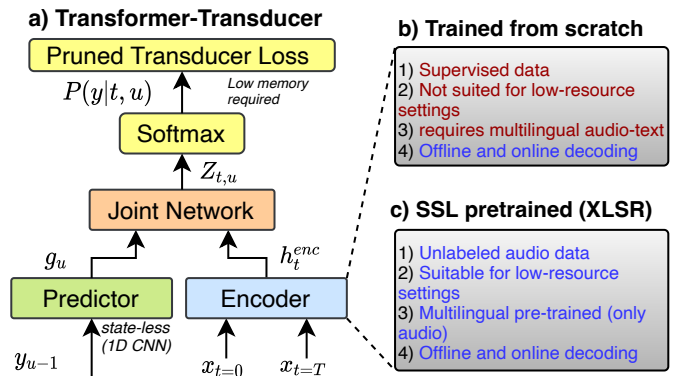

Fig. 1: Current state-of-the-art a) Transducer ASR includes state-less predictor, pruned transducer loss and b) Transformer-based encoder, trained from scratch. We replace the encoder by XLSR-53, an SSL model suitable for low-resource applications. Our contributions lead to the c) XLSR-Transducer.

disproportionate attention scores to few initial tokens for streaming language models [25], termed as attention sinks. We study the effects of attention sinks for the first time in streaming ASR. Formally, at decoding time, we allow the transformer layers in XLSR to attend to few initial frames in addition to designated frames in chunk and past context. In theory, this reduces total computation required for processing an audio chunk during streaming decode. Although SSL encoder-based transducer models have been explored in [26], detailed results for varying chunk sizes and past context are not available. The trade-off between increased latency due to larger chunk or past context sizes and performance is not immediately clear, especially when the SSL model is not pre-trained in a streaming fashion. In this work, we present a comprehensive analysis that can better inform the choice of these hyperparameters based on specific requirements. Moreover, our work explores attention sinks, which can reduce overall computation while improving performance.

Our contributions are covered below:

- Introduction of the XLSR-Transducer, a multilingual SSL encoder based transducer model, demonstrating significant WER improvement on the AMI dataset compared to large speech foundational models and other open-source ASR models;
- extension to streaming XLSR-Transducer and a systematic study of chunk size and past context on training and inference;
- to author's knowledge, this is the first work that explores the attention sink [25] phenomenon for streaming ASR which leads to improved WER; and
- Evaluation of the XLSR-Transducer on AMI and five languages of CommonVoice dataset in low resource settings.
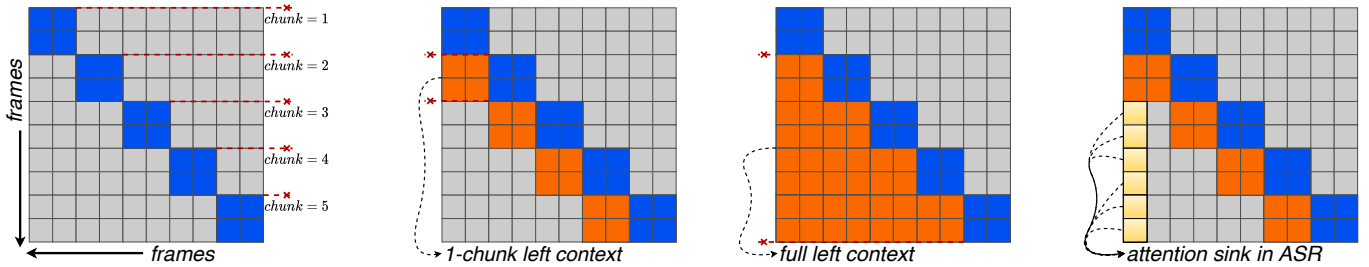
Fig. 2: Masking strategies for streaming XLSR-Transducer. Multi-chunk training allows decoding with variable chunk size (blue) and left context (orange). Each square denotes "n" frames. Attention sink (yellow) allows context from the first n frames. Our results show that **attention sink frames offer a better trade-off w.r.t increasing left context alone**, leading to lower WERs.

II. XLSR-TRANSDUCER

In a typical Transformer-Transducer (TT) ASR model (Figure 1a), there are three networks: the encoder, predictor, and joiner. The encoder processes audio frames to produce acoustic embeddings. The predictor generates token embeddings in an auto-regressive manner, taking previous non-blank tokens as input. Lastly, the joiner combines the outputs from the encoder and predictor to predict a probability distribution over the tokens in the vocabulary. In this work, we utilize a stateless predictor [27] composed of an embedding layer and one 1-D CNN layer, and the joiner network consists of one linear layer. Typically, the encoders [28]–[31] are trained from scratch and require a large amount of in-domain supervised data to achieve decent WER, which may not always be feasible. We train the TT model using the pruned-transducer loss [7] from k2[1] toolkit.

A. Non-Streaming XLSR-Transducer

In contrast to encoders trained from scratch, recent advancements in SSL pretrained models demonstrate competitive performance [14], [15] when fine-tuned with a limited amount of labeled data for ASR. Previously, the popular ASR systems employing SSL pretrained models have utilized CTC loss [5], encoder-decoder based architecture [2], [3], and Lattice-Free MMI loss [32] (hybrid approach) for training. In this paper, we integrate pretrained models as encoders in the TT setup, as illustrated in Figure 1c. One notable advantage is the ability to achieve strong ASR performance with relatively low amounts of training data. We select XLSR-53 [14] as our encoder model, which takes raw audio as input and outputs audio frames with a frame duration of 25ms and a stride of 20ms. The selection of XLSR-53 is driven by its large-scale pre-training on multilingual audio, which has demonstrated competitive ASR performance [14] in the low-resource across multiple languages.

B. Streaming XLSR-Transducer

XLSR is typically trained and decoded using entire audio sample. This makes the proposed XLSR-Transducer non-streaming despite the use of stateless predictor and linear joiner that are inherently streaming. The main challenge to port SSL models to streaming is the use of self-attention in the transformer layers [8], i.e., computed over entire acoustic frames of an utterance. In this paper, we present multiple masking patterns [24] to limit the frame context over which self-attention is computed, simulating streaming ASR within the XLSR-Transducer setup.

Chunked masking   In streaming ASR, decoding partial hypotheses begins after receiving a few audio frames, known as the chunk size. As depicted in Figure 2a, we implement chunk-wise decoding by

masking frames outside a specific chunk during the forward pass from the XLSR model. The mask is applied after dot-product computation during self-attention, ensuring that each frame inside a chunk has access to all the frames within that chunk. Note that the XLSR model also includes a CNN front-end, which takes raw audio as input. Thus, we feed chunk-size equivalent raw waves to the CNN front-end in a sequential manner and concatenate them across the time dimension to obtain all the frames for an utterance. In this paper, we explore chunk sizes of 16, 32, 64, and 128, translating to approximately 320ms, 640ms, 1280ms, and 2560ms, respectively, for XLSR.

Chunked masking with variable left context chunks   In practice, when decoding chunk "n", we have access to all the previous chunks, which can be utilized as left context. As illustrated in Figures 2b and 2c, a variable number of left context chunks can be utilized during the self-attention computation of a chunk, with the possibility of using the full left context. The number of frames in the left context is a multiple of the chunk size, as this can be efficiently implemented to store past chunks in the cache.

Streaming training and decoding   The use of non-streaming XLSR-Transducer for streaming decoding with the described masking patterns presents a challenge. The model has been trained on full context, creating a train-test mismatch. To address this challenge, we train the model in a streaming fashion using a fixed chunk size and left context. Flexibility in our chunked mask implementation allow us to perform both streaming and non-streaming decoding using a single model. The advantage of our method is that it only affects the fine-tuning stage and we can avoid the computationally prohibitive pre-training of the SSL model.

Multi-chunk Training   In many practical use-cases of streaming ASR, varying the chunk size at inference is often desirable depending upon latency requirements. However, a streaming XLSR-Transducer model trained with a fixed chunk size may not yield optimal WERs when decoded with different chunk sizes. Also, training multiple models for varying chunk sizes may be infeasible. To address this limitation, we propose randomly selecting the chunk size from the predefined list mentioned above for each batch during training.

C. Attention sinks for streaming ASR

In a recent work on streaming language models (LM) [25], [33], [34], it was shown that a surprisingly large amount of attention scores during self-attention computation inside transformer layers is directed towards the initial tokens, termed as *attention sinks*. This was attributed to the Softmax operation, which mandates attention scores to sum up to one and in autoregressive LMs, all subsequent tokens have access to the initial tokens. Consequently, the model may find it easier to learn to assign large scores to these initial tokens. In our streaming model training, where we utilize full left

context, we employ a similar setup. This leads us to introduce the first utilization of the attention sinks in the context of streaming ASR during inference. Specifically, as depicted in Figure 2d, we enable self-attention to focus on not only frames within a chunk and left context chunks, but also on the initial few frames.

III. EXPERIMENTAL SETUP

A. Datasets

We benchmark our proposed model on two datasets—AMI and CommonVoice. We choose AMI, instead of Librispeech [35], because it involves natural, multi-party, conversational speech with acoustic complexity and overlapping dialogue, whereas LibriSpeech contains read speech, making AMI more relevant for real-world scenarios.
AMI We train and evaluate the XLSR-Transducer model on the individual head microphone (IHM) split from the AMI dataset [36] containing audios with a sampling rate of $16\,\mathrm{kHz}$. We use the default recipe for AMI from lhotse[2] toolkit to prepare the train, dev and eval sets containing 80hr, 8.8hr, 8.5hr of audios respectively. In all our experiments on the AMI dataset, we use WER on the dev set to select the best epoch and report the results on the eval set.
CommonVoice We validate XLSR-Transducer on five non-English languages from CommonVoice-v11 [37].[3] This includes Catalan (CA), Belarusian (BE), Spanish (ES), French (FR) and Italian (IT). To keep experimentation under the low-resource domain, we extract randomly a 100-hr subset from the training data per language. Later, we train streaming and non-streaming models. We report WERs on the full official test sets.

B. Zipformer-Transducer Baseline

We establish strong baselines by training non-streaming and streaming Zipformer transducer models [30] from scratch, following the AMI recipe[4] from Icefall toolkit (we only use the IHM set). The state-less Predictor [27] consists of an embedding layer followed by one 1-D CNN layer and joiner consists of 1 linear layer. We use the default hyper-parameters from the recipes and train for 30 epochs. See more training details in [30]. Beam search with width of 4 is used across all ASR decoding in this paper.

C. XLSR-Transducer Training

The XLSR-transducer model is constructed from the Icefall's Transducer recipe for AMI dataset adapted with the XLSR model from fairseq [38]. The fine-tuning uses Scaled Adam [39] and a learning rate scheduler that consists of a 500-step warmup phase [8] followed by a decay phase directed by number of steps and epochs. Furthermore, the model is optimized with pruned rnn-t loss [6], [7]. The learning rate is set to $lr = 1.25e^{-3}$ for AMI and $lr = 5.0e^{-3}$ for CommonVoice. We train AMI and CommonVoice models for 10 and 20 epochs, respectively.

IV. RESULTS & DISCUSSION

Non-streaming ASR We begin by benchmarking the XLSR-Transducer model for non-streaming ASR on the AMI dataset. Results are reported in the Table I (*full-attn*). We compare against large open source foundational speech models. It can be seen that the proposed XLSR-Transducer model achieves significant improvement in WERs. Note that we do not finetune these open-source models on the AMI dataset because the primary goal of our work is to demonstrate best practices for utilizing SSL models in streaming

[2]https://github.com/lhotse-speech/lhotse
[3]CommonVoice-v11: cv-corpus-11.0-2022-09-21.
[4]github.com/k2-fsa/icefall/tree/master/egs/ami

TABLE I: WERs on the AMI eval set. On non-streaming decoding[†] XLSR-Transducer yields significant WER reduction. On streaming decoding, the multi-chunk training (mutiple) provides significant gain w.r.t encoders trained from scratch with minimal degradation in non-streaming performance. [‡]encoder-decoder model. [¶]decoding chunk size 2000ms.

Encoder	Chunk Size	Chunk Size decoding		
	training	320 ms	1280 ms	full-att[†]
decoding: non-streaming ASR				
Whisper large-v2 (1.6B)[‡] [1]	-	-	-	16.9
FastConformer (1.1B) [31]	-	-	-	15.6
Zipformer (70M)	-	-	-	21.0
decoding: streaming ASR				
FastConformer (114M)[¶] [12]	-	-	24.2	-
Zipformer (70M)	multiple	28.5	24.6	23.2
XLSR (300M)	full-att	35.3	17.8	**12.7**
XLSR (300M)	320 ms	**17.1**	15.0	14.2
XLSR (300M)	1280 ms	19.7	14.5	13.1
XLSR (300M)	multiple	17.7	**14.2**	**12.9**

ASR. Next, we train a *Zipformer* encoder based transducer model from scratch and observe that XLSR-Transducer yields 39% relative improvement in WER. It is clear that there are significant advantages of using pretrained encoders in TT setup for low resource ASR.
Streaming ASR First, we decode the non-streaming trained XLSR-Transducer model in streaming fashion by applying different masks (see §II-B). Results are reported in the Table I, where full left context is used during decoding. The XLSR-Transducer achieves significant improvement over *Zipformer* and *FastConformer* based transducer models. Despite the improvements, there is a significant degradation from non-streaming performance because the model was not trained for streaming. When model was trained for streaming with full left context and decoded using chunk size of 320ms, the performance improves (35.3% → 17.1% WER) because of the train-test matched chunk size setting. However, training with larger chunk sizes and decoding at 320ms degrades performance, showcasing the importance of context during self attention computation inside a chunk which the model may have learned during training.

Increasing the chunk duration during decoding (320ms → 1280ms) improves the performance monotonically [13], which is expected as larger context available for frame attention score computation. Now, the streaming trained models are decoded in non-streaming which can serve as performance upper bound. We observe that increasing chunk size during training improves the non-streaming performance and even when chunk size of 320ms is used during training, the results only degrade by 1.5% in absolute WER. When random chunk sizes are used during training, the gap is 0.2% when compared with the best non-streaming ASR performance. Overall, a streaming trained XLSR-Transducer model using random chunk sizes shows best WER when decoded in streaming fashion with 1280ms chunk duration and performance gap for non-streaming decoding is minimal. Thus, a single model can be used for both streaming and non-streaming ASR.
Streaming ASR with variable left context Using full left context during training and decoding will incur additional computation. Limiting left context during training of streaming models resulted in significant WER degradation; therefore, we use full left context for all streaming XLSR-Transducer training. Figure 3 list WERs when the number of left context chunks are varied during decoding. Note that the left context duration is in multiple of chunk size. Increasing the left context improves the performance for all training scenarios
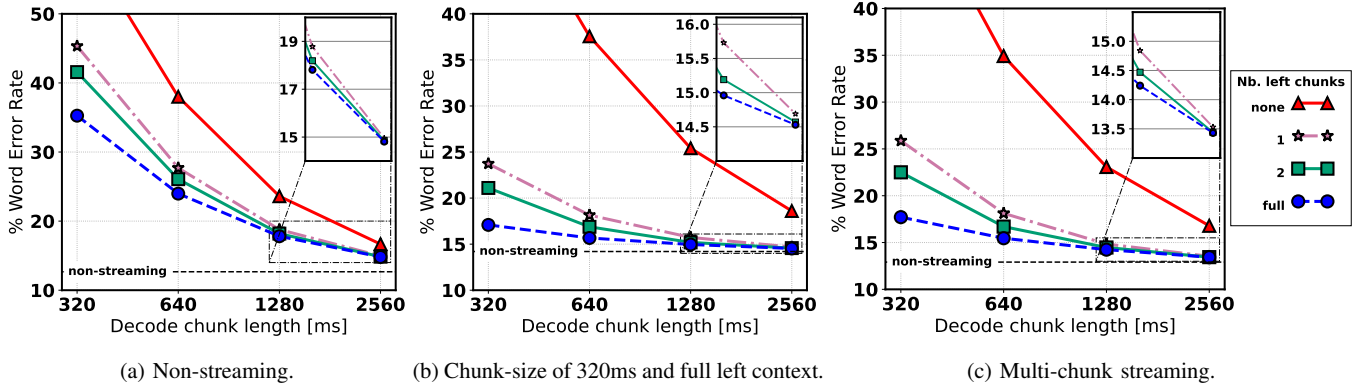
Fig. 3: Plots of WERs on AMI eval set for XLSR-Transducer trained on three configurations (a, b and c) and decoded on multiple streaming scenarios. Note that adding one or more left-context chunks at decoding time reduces WERs dramatically.

TABLE II: WERs of streaming XLSR-Transducer on five Common-Voice languages. Models are fine-tuned on random 100hr train subset and with multi-chunk training. full-att: non-streaming decoding. [†]CA is 28h long, while the rest 26h. [‡]median (duration) in seconds. [¶]non-streaming training and decoding.

Lang[†]	Test Set		Streaming model - chunk size [ms]				full-att[¶]
	#utt	[50%-dur][‡]	320	640	1280	2560	full-att
CA	16k	6.1	17.5	15.2	13.9	12.9	12.0	10.7
BE	15.8k	5.7	20.0	17.5	15.9	14.8	13.8	13.7
ES	15.5k	6.1	17.7	15.0	13.5	12.2	11.3	10.8
FR	16k	5.7	24.3	21.6	20.0	18.7	17.6	17.1
IT	15k	6.3	18.5	15.9	14.3	13.1	12.1	11.5

and chunk duration. At the same time, a significant improvement is observed using just one chunk of left context. Further increase in left context improves the performance but the reduction in WER per left context chunk is lower. When the XLSR-Transducer is trained with multi-chunk streaming strategy and decoded with 1280ms chunk size, a relative improvement of only 4% in WER is observed using full left context instead of one left context chunk. Thus, a limited number of left context chunks should be enough for most real-world streaming ASR with XLSR-Transducer model.

XLSR-Transducer on multiple languages We also train the proposed model on five non-English languages of CommonVoice [37]. WERs are listed in Table II for multi-chunk streaming and full-attention non-streaming models. We see competitive WERs for models evaluated under different streaming conditions, with constant WERs improvement as chunk size increases; similar behavior is reported in [13]. The upper-bound WERs are obtained with a model trained and evaluated in non-streaming fashion, i.e., last column of Table II. We note negligible WER degradation (up to 1.5% absolute WER, worse CA; best BE) for full-attention decoding (*full-att*) on streaming models vs. their non-streaming counterparts. This confirms the robustness of XLSR-Transducer on multiple languages.

Improving streaming ASR with attention sinks In theory, restricting left context chunks should lead to overall latency improvements for streaming ASR. The recent observation of attention sinks phenomena [25], [40], where the transformer models learn to assign relatively higher attention scores to initial tokens, may help in reducing the overall computation required to decode one chunk of audio in streaming ASR. WERs on the AMI dataset are reported in Table III for the XLSR-Transducer model trained in multi-chunk setting. For different chunk sizes and left contexts, we observe that increased frames for attention sinks improve the performance

TABLE III: WERs on AMI eval set for varied decoding settings, including attention sink. Adding attention sinks offer a better trade off than larger left context. (blue) denotes relative WER reduction w.r.t no attention sink within same chunk and left context. [†]Number of chunks. [‡]Number attention sink frames.

Decoding settings		Decoding chunk-size	
Left-context[†]	attn-sink[‡]	320ms	640ms
full	none	17.7	15.5
1	none	25.9 –	18.1 –
	1	22.9 (+11.6)	17.4 (+4.1)
	4	21.4 (+17.4)	16.8 (+7.1)
	16	19.7 (+23.7)	16.3 (+10.0)
2	none	22.5 –	16.7 –
	1	20.9 (+7.1)	16.4 (+1.8)
	4	20.0 (+11.1)	16.2 (+3.0)
	16	18.9 (+16.0)	15.9 (+4.8)
4	none	19.8	15.9

monotonically. Specifically, for a smaller chunk of 320ms, using 1 left context chunk and 16 frames (i.e., 320ms) for attention sinks performs better than using 2 left context chunks by 12% in relative terms despite attending over same number of frames. We do not observe a significant reduction in WERs beyond a chunk size of 640ms. Overall, our results show that it is better to use attention sinks than increasing left context chunks beyond 1 for improved performance. We also run decoding with attention sinks on the CommonVoice dataset and observe similar trends but do not include a results table for brevity.

V. Conclusions

In this paper, we demonstrate that using a self-supervised trained model as encoder in the transducer framework, termed as XLSR-Transducer, leads to significant improvement in WER on AMI dataset. We explore various chunked masks and left context configurations to enable streaming decoding in XLSR. Our findings across 2 datasets and 6 languages shows that the proposed model achieves streaming performance comparable to non-streaming ASR.

REFERENCES

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28 492–28 518.

[2] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[3] F. Mai, J. Zuluaga-Gomez, T. Parcollet, and P. Motlicek, "HyperConformer: Multi-head HyperMixer for Efficient Speech Recognition," in *Proc. INTERSPEECH 2023*, 2023, pp. 2213–2217.

[4] R. Prabhavalkar, T. Hori, T. N. Sainath, R. Schlüter, and S. Watanabe, "End-to-end speech recognition: A survey," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[5] A. Graves and A. Graves, "Connectionist temporal classification," *Supervised sequence labelling with recurrent neural networks*, pp. 61–93, 2012.

[6] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[7] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and D. Povey, "Pruned rnn-t for fast, memory-efficient asr training," *arXiv preprint arXiv:2206.13236*, 2022.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] 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:1910.12977*, 2019.

[10] 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 *ICASSP*. IEEE, 2020, pp. 7829–7833.

[11] B. Li, A. Gulati, J. Yu, T. N. Sainath, C.-C. Chiu, A. Narayanan, S.-Y. Chang, R. Pang, Y. He, J. Qin *et al.*, "A better and faster end-to-end model for streaming asr," in *ICASSP*. IEEE, 2021, pp. 5634–5638.

[12] V. Noroozi, S. Majumdar, A. Kumar, J. Balam, and B. Ginsburg, "Stateful fastconformer with cache-based inference for streaming automatic speech recognition," *arXiv preprint arXiv:2312.17279*, 2023.

[13] 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*. IEEE, 2020, pp. 6069–6073.

[14] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979*, 2020.

[15] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi *et al.*, "Scaling speech technology to 1,000+ languages," *arXiv preprint arXiv:2305.13516*, 2023.

[16] B. Khonglah, S. Madikeri, S. Dey, H. Bourlard, P. Motlicek, and J. Billa, "Incremental semi-supervised learning for multi-genre speech recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7419–7423.

[17] M. Kleinert, H. Helmke, G. Siol, H. Ehr, A. Cerna, C. Kern, D. Klakow, P. Motlicek, Y. Oualil, M. Singh, and A. Srinivasamurthy, "Semi-supervised adaptation of assistant based speech recognition models for different approach areas," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, 2018, pp. 1–10.

[18] S. Dey, P. Motlicek, T. Bui, and F. Dernoncourt, "Exploiting semi-supervised training through a dropout regularization in end-to-end speech recognition," in *Proc. of Interspeech 2019*, 2019.

[19] J. Ao, Z. Zhang, L. Zhou, S. Liu, H. Li, T. Ko, L. Dai, J. Li, Y. Qian, and F. Wei, "Pre-training transformer decoder for end-to-end asr model with unpaired speech data," *arXiv preprint arXiv:2203.17113*, 2022.

[20] Y.-A. Chung, Y. Zhang, W. Han, C.-C. Chiu, J. Qin, R. Pang, and Y. Wu, "W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 244–250.

[21] J. Ao, R. Wang, L. Zhou, C. Wang, S. Ren, Y. Wu, S. Liu, T. Ko, Q. Li, Y. Zhang *et al.*, "Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing," *arXiv preprint arXiv:2110.07205*, 2021.

[22] T. Javed, S. Doddapaneni, A. Raman, K. S. Bhogale, G. Ramesh, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Towards building asr systems for the next billion users," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 10 813–10 821.

[23] A. Bapna, C. Cherry, Y. Zhang, Y. Jia, M. Johnson, Y. Cheng, S. Khanuja, J. Riesa, and A. Conneau, "mSLAM: Massively multilingual joint pre-training for speech and text," *arXiv preprint arXiv:2202.01374*, 2022.

[24] P. Swietojanski, S. Braun *et al.*, "Variable attention masking for configurable transformer transducer speech recognition," in *ICASSP*. IEEE, 2023, pp. 1–5.

[25] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, "Efficient streaming language models with attention sinks," *arXiv preprint arXiv:2309.17453*, 2023.

[26] C.-C. Chiu, J. Qin, Y. Zhang, J. Yu, and Y. Wu, "Self-supervised learning with random-projection quantizer for speech recognition," in *International Conference on Machine Learning*. PMLR, 2022, pp. 3915–3924.

[27] M. Ghodsi, X. Liu, J. Apfel, R. Cabrera, and E. Weinstein, "Rnn-transducer with stateless prediction network," in *ICASSP*. IEEE, 2020, pp. 7049–7053.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[30] Z. Yao, L. Guo, X. Yang, W. Kang, F. Kuang, Y. Yang, Z. Jin, L. Lin, and D. Povey, "Zipformer: A faster and better encoder for automatic speech recognition," *arXiv preprint arXiv:2310.11230*, 2023.

[31] D. Rekesh, N. R. Koluguri, S. Kriman, S. Majumdar, V. Noroozi, H. Huang, O. Hrinchuk, K. Puvvada, A. Kumar, J. Balam *et al.*, "Fast conformer with linearly scalable attention for efficient speech recognition," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2023, pp. 1–8.

[32] A. Vyas, S. Madikeri, and H. Bourlard, "Comparing ctc and lfmmi for out-of-domain adaptation of wav2vec 2.0 acoustic model," in *Proceedings of Interspeech*, 2021. [Online]. Available: https://arxiv.org/abs/2104.02558

[33] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in neural information processing systems*, vol. 33, pp. 17 283–17 297, 2020.

[34] C. Han, Q. Wang, H. Peng, W. Xiong, Y. Chen, H. Ji, and S. Wang, "Lm-infinite: Zero-shot extreme length generalization for large language models," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 3991–4008.

[35] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[36] J. Carletta, S. Ashby *et al.*, "The ami meeting corpus: A pre-announcement," in *International workshop on machine learning for multimodal interaction*. Springer, 2005, pp. 28–39.

[37] R. Ardila, M. Branson, K. Davis, M. Kohler, J. Meyer, M. Henretty, R. Morais, L. Saunders, F. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020, pp. 4218–4222.

[38] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 2019, pp. 48–53.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] G. Attanasio, B. Savoldi, D. Fucci, and D. Hovy, "Multilingual speech models for automatic speech recognition exhibit gender performance gaps," *arXiv preprint arXiv:2402.17954*, 2024.