HOW MUCH CONTEXT DOES MY ATTENTION-BASED ASR SYSTEM NEED?

Robert Flynn, Anton Ragni

Department of Computer Science, The University of Sheffield, United Kingdom
{rjflynn2, a.ragni}@sheffield.ac.uk

ABSTRACT

For the task of speech recognition, the use of more than 30 seconds of acoustic context during training is uncommon, and under-investigated in literature. In this work, we examine the effect of scaling the sequence length used to train/evaluate (dense-attention based) acoustic and language models on speech recognition performance. For these experiments a dataset of roughly 100,000 pseudo-labelled Spotify podcasts is used, with context lengths of 5 seconds to 1 hour being explored. Zero-shot evaluations on long-format datasets Earnings-22 and Tedlium demonstrate a benefit from training with around 80 seconds of acoustic context, showing up to a 14.9% relative improvement from a limited context baseline. Furthermore, we perform a system combination with long-context transformer language models via beam search for a fully long-context ASR system, with results that are competitive with the current state-of-the-art.

Index Terms— speech recognition, long-context, cross-utterance, self-attention

1. INTRODUCTION

Performance on sequence based tasks, has seen a consistent benefit from the introduction of methods that enable the modelling of longer range dependencies [1, 2]. The transformer architecture [2] is a distinct example of this, demonstrating benefits from training on sequences of 1000s of tokens on language modelling tasks [3]. However, for the task of automatic speech recognition (ASR) there is limited work exploring the effect of using longer acoustic sequences. In part, this may be due to the format of many academic datasets, which are typically provided as a series of short (typically 1-20s) utterances. This therefore hurts the development of methods that aim to utilise larger amounts of context or learn to segment a recording in a purely end-to-end fashion.

Previous work on utilising cross-utterance acoustic context still deals with fairly short sequences of 20-30s [4, 5, 6]. The often stated reason for limiting the context window used by self-attention based models is their quadratic complexity with respect to the sequence length. However, it is also not clear whether this current modelling paradigm is capable of utilising truly long sequences of minutes or hours in duration. For instance, in the task of language modelling, [3] finds that transformers struggle gaining any benefit from sequences longer than 1024, when trained on their target dataset.

While long-context acoustic models (AMs) are fairly under-investigated, there is ample work [7, 8, 9] on utilising cross-utterance context within the language modelling component of ASR systems.

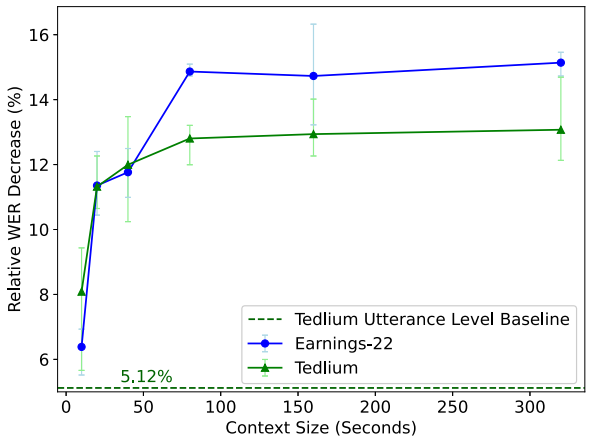

Fig. 1. Relative WER reduction from a baseline with 5 seconds of context, for AMs with 10-320s of context

However, we argue this is not optimal, as an assumption of independence between utterances is still made at the acoustic-level. Consequently, the long-context language model may not recover information lost by the utterance-level AM, and the system is only able to adapt to linguistic aspects of the data based on the context.

As such, in this work, we investigate the benefit of extending the context length of both the AM and language model (LM) components of transformer based speech recognition systems. A breakdown of our contributions is given as follows: **1.** An investigation is conducted on the effect of training/evaluation context length on speech recognition, with results demonstrating an optimal context length of around 80s. **2.** We demonstrate training of dense-attention based AMs with maximum context lengths of up to **1 hour** through the application of a sequence length warmup and various efficiency adaptions from prior work. **3.** An overlapping decoding scheme is introduced to reduce context fragmentation, and the optimal amount of overlap to use is investigated.

The rest of this work is ordered as follows: Section 2 details the training and evaluation adaptions that were made in order to fairly investigate and compare a range of context sizes for both the AM and LM components of the ASR system. Section 3 overviews the experimental details. Section 4 presents and analyses the results, with the conclusion given in section 5.

Finally, we release all trained model checkpoints and code.[1]

This work was supported by the CDT in Speech and Language Technologies (SLT) and their Applications funded by UKRI [grant number EP/S023062/1]. This research was supported by funding from Meta.

[1] https://github.com/robflynnyh/long-context-asr

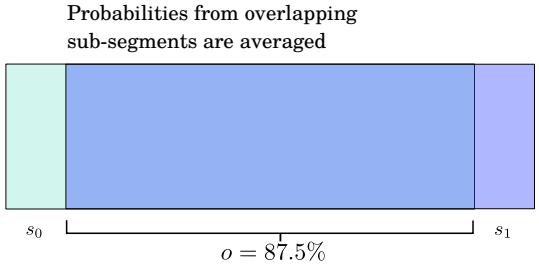

Fig. 2. Depiction of overlapping window inference.

2. ADAPTIONS FOR LONG-CONTEXT ASR

2.1. Architecture

Model	Duration (min)	Runtime (tokens/s)
Conformer	9 / 23	30,200 / 57,300
FastConformer	18 / 70	57,650 / 80,900

Table 1. Maximum context possible on 1 A100 during **training** with batch size of 1. Without/With Flash Attention.

Conformer [10] based AMs, trained with connectionist temporal classification (CTC) [11], are used as the basis for this investigation. These architectures typically utilise some form of subsampling. Recent work [12] explores increasing the level of subsampling from $4\times$ to $8\times$, as a simple method of decreasing the sequence length and hence reducing the compute and memory complexity of the model. Additionally, the standard convolution blocks in the subsampling module are replaced with depthwise separable convolutions with a smaller feature dimension than the rest of the model. This configuration is referred to as FastConformer, with results demonstrating similar or improved results over the standard Conformer setup. As shown in table 1, when pairing the FastConformer with flash-attention [13] (an efficient algorithm for computing attention on the GPU without approximations), it becomes possible to train on recordings of over 1 hour in length on 1 80GB A100 GPU.

2.2. Context Fragmentation

Typically, in ASR, a long-format dialogue is segmented into a set of utterances based on silences, and these utterances are transcribed independently. As a consequence, frames near the start and end of an utterance have a fragmented context. For the case of autoregressive language modelling, [3] demonstrated that many long-context LMs benefit from a reduction in the positions where the context is fragmented as the context window is enlarged, rather than from the utilisation of longer distance dependencies. We posit that this is also the case for many long-context methods in ASR.

2.3. Overlapping Window Inference

To reduce the impact of context fragmentation and enable fairer comparison between different context lengths, recordings are processed using an overlapping decoding scheme. Specifically, the input is processed in segments $S \ni (s_0 ... s_N)$ and for a given context length c and overlap percentage o i.e 87.5% the starting position of the i^{th} segment s_i is given by: $i \cdot c \cdot (1 - {}^{o}/_{100})$. Probabilities from overlapping sub-segments are averaged to obtain predictions for the overall sequence. A depiction of two overlapping segments is shown in figure 2.

2.4. Beam Search Decoding

Due to their independence assumption, CTC based AMs are typically combined with an external LM. For this system combination, the beam search algorithm can be used with a neural LM. This is preferred to rescoring in terms of use of context [8], as it does not rely on the use of limited context n-gram model. We use the method described in [8] for combining the ASR CTC and LM probabilities, which defines the overall score (used to rank a hypothesis) of a given vocabulary index i at time-step t as $\text{Score}(i_t) = \text{Score}_{AM}(i_t) + \text{Score}_{LM}(i_t)$. Due to the overlapping decoding scheme, there exists k instances of AM probabilities for a given timestep from each overlapping segment s, hence as discussed, the AMs score/contribution is attained through an average over segments as depicted in Eq. 1.

$$\text{Score}_{\text{AM}}(i_t) = \log \sum_{s}^{k} \frac{P_{AM}(i_t^s)}{k} \tag{1}$$

2.5. Sequence Length Warmup

Gradually increasing the sequence length throughout training has been investigated in prior work for the purposes of reducing training time [3], and training instability caused by gradient variance [14]. This can be seen as a form of curriculum learning [15]. In this work, we find this gradient variance to be particularly destructive for the AM when the context is greater than 40s, with these models often failing to train without a sequence length warmup.

For this method several hyperparameters are employed, namely: a minimum sequence length s_0 that is used at the start of training, which is then doubled every n recordings/steps r, until a maximum sequence length s_m is reached. Hence, the sequence length at a given recording s_r is shown as follows: $s_r = \min(s_0 + s_0 \cdot \lfloor {}^{r}/_{n} \rfloor, s_m)$

3. EXPERIMENTAL CONFIGURATION

3.1. Data

As investigating long-context models may require larger amounts of long-format data than typical ASR datasets provide, the collection of Spotify podcasts provided in [16] is selected for the AMs training data. Podcasts in this dataset last on average 33 minutes, with many going over one hour. In total, this amounts to 58,000 hours of training data. This data is not human-labeled and instead is provided with pseudo-labels produced using Google's speech API. Tedlium [17] and Earnings-22 [18] are used as evaluation datasets, which were selected due to their long-format. Tedlium is composed of single-speaker TED talks lasting around 14 minutes in duration. Tedlium's dev and test sets total 1.6 and 2.6 hours respectively. As Tedlium contains segments of untranscribed speech such as adverts, these portions of the spectrogram are set to zero for overlapping inference. Earnings-22 consists of earning report meetings lasting up to 2 hours with multiple speakers and a diverse range of accented speech. As no official dev/test splits are provided, the partitions given in [19] are used, which feature 5.5/5.6 hours for dev/test splits.

All audio data is converted to 16khz, and 80-band Mel spectrograms (with a window length of 400 and hop length of 160) are used to train the model. Mean and standard deviation statistics across each audio file i.e. podcast/meeting/talk are used for spectrogram normalisation. For text tokenization, the sentencepiece tokenizer is used with the "nmt_nfkc_cf" normalisation rule. As the model is not able to adapt to dataset specific transcription styles, normalisation is applied to any model outputs and the reference transcript. The text normaliser from Whisper [6] is used.

For language model training data, subsets of the following datasets are used: OpenSubtitles [20], OpenWebText [21], YouTube Subtitles [21], PG-19 [22], Books3 [21] and Spotify Podcasts [16]. These subsets were filtered to remove documents that contain more than 1% OOV tokens according to a tokenizer trained on the Spotify corpus. In total, this amounts to around 4.5 billion words.

3.2. Model Configuration

The AM uses the FastConformer [12] subsampling configuration with 8x downsampling using depthwise separable convolutions with a hidden dimension of 256, followed by 6 Conformer layers [10] with a hidden dimension of 768. The model is trained using SCCTC [23], without intermediate losses. Batch normalisation [24] is swapped out with batch renormalisation [25], and Convolutional modules feature a reduced kernel size of 9. Rotary embeddings [26] are used as the positional encoding method. Each attention layer features 6 heads, and the flash attention algorithm [13] is used to compute attention. In total, the AM has 90 million parameters.

For beam search decoding, a transformer is used as the external LM. This model features 6 layers with a hidden dimension of 1024. QKNorm [27] multi-query attention [28] is used with 16 heads for the queries. RMSnorm [29] is used in-place of Layer norm [30]. For positional encoding, the dynamic position bias proposed in [31] is used. The LM has 72M parameters.

All models share the same vocabulary composed of 4095 BPE tokens learnt from the Spotify corpus using the sentencepiece tokenizer with an additional blank token for the AM. Additionally, GEGLU [32] layers with an expansion factor of 4 are used for feedforward modules. We make use of fused kernels for these modules.

3.3. Training Configuration

During training of the AMs, to ensure all context sizes receive the same number of optimization steps, the total duration of each minibatch is kept fixed at around 1 hour of audio. As the training data is provided with word level timesteps, the podcasts can be chunked into inputs of arbitrary length and the text corresponding to each chunk can be retrieved for training. A separate AM is trained for each sequence length reported on, which is repeated 3 times using different random seeds.

The LM is trained using Transformer-XL style key-value caching [33, 3]. Here, documents are processed sequentially in chunks of a given sequence length, while attending to a key-value cache from prior sequences. A model trained with a sequence length of 512, and a maximum cache of 1280 tokens is used for all evaluations. There was no degradation observed from evaluating the model at sequences shorter than seen during training, compared to models trained at that length for the LM, hence the same model is used for varying context sizes for consistency.

The Madgrad optimizer [34] is used for all training runs with gradient clipping, and a learning rate warmup followed by a cosine annealing schedule. For models trained with a sequence length warmup, 5s is used as the initial sequence length s_0, which is doubled every 5K recordings. All models are trained for one epoch only unless otherwise specified, and no regularisation is used.

Training of the AMs is performed on 1 A100 GPU taking around 15–24 hours, for context lengths below 3 minutes. Training with a maximum context length of 1 hour took 65 hours. The LM is trained on 4 A4500s, taking 2 days.

3.4. Beam Search

A beam width of 25 is used for beam search. The search is constrained to vocabulary indices within a threshold c of the argmax of the AM probabilities for each timestep. Beams with a score p less than the top ranked beam are pruned at each time-step. Hyperparameters: α, β, c and p are tuned on the combined evaluation development sets. Additionally, a document from the Spotify podcast data is included as the initial context/prompt, priming the LM for the AMs transcription style. Our method is implemented in Python, with decoding taking roughly 8 and 40 minutes for Tedlium and Earnings-22, when parallelising over each meeting/talk.

4. EXPERIMENTAL RESULTS

All experiments are repeated 3 times unless otherwise specified and mean and standard deviation (s.t.d) statistics are reported. The main results for varying AM context size can be found in table 2. Excluding table 6, all reported results use greedy decoding, and any results reported on in the tables for context lengths over 40s utilise a sequence length warmup.

4.1. Overlapping Window Inference

A comparison of varying percentages of overlap is given in table 4. As expected, longer context models benefit less from increasing the overlap percentage, as the increase in context size already reduces the amount of context fragmentation. Interestingly, an overlap percentage of 50% is harmful to performance, which may be due to small variances in predicted alignments resulting in duplicate outputs at overlap boundaries. No meaningful improvements are seen from extending the overlap percentage beyond 87.5%, hence this setting is used for all the other investigations. Notably, the use of the overlapping window scheme with a context length of 20s results in a 6.5% relative word error rate (WER) reduction from using the utterance boundaries provided as part of Tedlium, demonstrating the impact of context fragmentation, and the efficacy of the method.

4.2. Acoustic Model Context Size

From the experiments given in table 2 and illustrated in figure 1, it can be observed that it is important to have at least 20s of acoustic context, with an optimal context length of 80s. Increasing from 5s to 10s of context brings a 6.5% and 8.1% relative improvement on Earnings-22 and Tedlium test splits. When increasing from 10s to 20s, we see a further 6.4% and 2.9% improvement. Tedlium benefitted less from the increased context than Earnings-22, with limited improvements extending past 20s of context, and no meaningful gains past 40s. While on Earnings-22, there is a sizeable improvement of 3.6% when increasing from 40s to 80s. This discrepancy may be due to the difficulty of Earning-22, with a higher WER and a broader range of accents, enabling the model to adapt based on the additional context information. No meaningful gains are seen from extending beyond 80s.

Results when using Tedlium's provided utterance boundaries (table 3) illustrate the models have difficulty working with sequence lengths not seen during training. When evaluating each model at varying context lengths, we find that this is most problematic when increasing or decreasing the context size by more than a factor of 2. Varying the context sizes used during training may help alleviate this form of overfitting.

4.3. 1 Hour of Context

As a curiosity, we train a model with up to 1 hour of context and include it in the results. This model attends over a maximum se-

Dataset	Metric	5s	10s	20s	40s	80s	160s	320s	3600s*
Earnings-22	WER	28.7/21.9	27.0/20.5	25.5/19.4	25.2/19.4	24.4/18.7	24.2/18.7	**24.1/18.6**	24.3/18.8
	s.t.d	0.2/0.4	0.1/0.2	0.3/0.2	0.2/0.2	0.1/0.1	0.4/0.3	0.2/0.1	0.1/0.2
Tedlium	WER	8.4/7.4	7.8/6.8	7.4/6.6	**7.2**/6.5	**7.2**/6.5	7.3/6.5	7.3/6.5	**7.2/6.4**
	s.t.d	0.2/0.1	0.1/0.2	0.1/0.1	0.1/0.1	0.0/0.1	0.2/0.1	0.1/0.1	0.2/0.1

Table 2. (Dev/Test) Results for each AM context length. *Evaluation context may be shorter depending on the recording duration

Metric	5s	10s	20s	40s
WER	8.7/7.7	8.0/7.2	**7.8/7.0**	8.0/7.6
s.t.d	0.2/0.1	0.3/0.1	0.1/0.1	0.1/0.0

Table 3. (Dev/Test) Standard utterance-level performance (Tedlium)

Context	0%	25%	50%	75%	**87.5%**	**93.75%**
5s	13.7/12.9	11.3/10.3	12.1/11.3	8.5/7.6	8.4/7.4	8.4/7.4
20s	8.7/8.0	8.1/7.4	8.4/7.6	7.4/6.7	7.4/6.6	7.4/6.6
80s	7.5/6.8	7.5/6.7	7.5/6.8	7.2/6.5	7.2/6.5	7.2/6.5

Table 4. (Dev/Test) WERs for different levels of overlap (Tedlium)

quence length of 45K tokens during training. There is no significant change in results compared to the 80s model, with a small improvement on Tedlium and a small degradation on Earnings-22. Interestingly, when evaluating this model at different context lengths, there is no meaningful change in WER from 320s up to 1 hour (around 14–20 minutes for Tedlium). From inspecting the attention weights, the model does attend to the full context, but does not benefit from it. Overall, this result demonstrates the robustness of our method to work for arbitrary context sizes without degradation, and highlights the need for alternative methods of learning from very long contexts.

4.4. Sequence Length Warmup

The use of the sequence length warmup enabled successful training of models with context windows greater than 40s, where previously training would not converge. Additionally, we train a model with a maximum context of 40s using a sequence length warmup, to compare with the results when using a constant sequence length. For this there is a slight improvement on Earnings-22 with a 0.6% relative WER reduction, and a small degradation of Tedlium with 1.8% relative WER increase. Due to the variance in the results, we conclude that the sequence length warmup did not significantly impact performance, but enables training of longer sequence lengths through an improvement in stability.

4.5. Beam Search Decoding

Dataset	64	128	256	512	1024
Earnings-22	83.4	69.4	62.3	57.6	54.0
Tedlium	98.1	83.1	75.6	71.6	69.7

Table 5. (Test) perplexity for the LM at varying context lengths

Results for the fully long-context ASR system are given in table 6, with language model perplexities provided in table 5. When decoding the 80s AM with 1024 tokens of LM context, there are large relative WER reductions compared to the greedy decoding baseline of 22.5% and 25.8% on Earning-22 and Tedlium test splits. Similarly to the results in table 2 we find that Earnings-22 benefits more from extending the LM context than Tedlium, with improvements when using up to 1024 tokens of context compared to 128 for Tedlium.

		LM Context (Tokens)				
Dataset	AM Context	64	128	256	512	1024
Earnings-22	5s	22.9/17.1	22.6/17.0	**22.5**/16.9	**22.5**/16.8	**22.5/16.7**
	80s	19.6/14.8	19.3/14.7	19.2/14.6	19.2/14.6	**19.1/14.5**
Tedlium	5s	5.9/5.4	**5.8/5.3**	**5.8/5.3**	**5.8/5.3**	**5.8/5.3**
	80s	5.2/4.9	**5.1/4.8**	**5.1/4.8**	**5.1/4.8**	**5.1/4.8**

Table 6. (Dev/Test) WERs when decoding with LM via beam search for various AM (s) and LM context sizes (number of tokens)

While on both datasets there is around a 1.5-2.5% relative improvement from increasing the LM context from 64 to 1024 tokens. On average, 1024 tokens will correspond to around 4.5 minutes of audio, showing that the LM benefits from a longer context than the AM.

When decoding the 5s AM, it can be observed that the LM is not able to recover information lost due to the limited context of the AM. This demonstrates the importance of increasing the context for all components of the ASR system.

Finally, training the 80s context AM for a second epoch (1 repeat) and decoding with the LM using 1024 tokens of context reduces WERs further to **13.6%** and **4.4%** on Earnings-22 and Tedlium, which is competitive with the current state-of-the-art [35, 6, 36, 19]. Notably, this exceeds the *Whisper small.en* [6] models long-form transcription performance on Tedlium of 4.6%, while using 66% of the parameters and less data/compute.

5. CONCLUSION

Many use-cases for ASR involve long-format data i.e. meetings or lectures, consequently there is a demand for models that can utilise the large amount of context information present in these formats. To better understand the capabilities of current approaches, this work analysed the effect of altering the amount of context used during training/evaluation of attention-based ASR systems. We demonstrate a benefit from training with up to 80s of acoustic context, a magnitude larger than what is used in literature. Typically, long-format data is segmented into utterances, which we find harmful to performance due to context fragmentation, and propose an alternative scheme, which rectifies this leading to consistent WER improvements. Results for the full long-context system demonstrate the advantage of increasing the context for all components of the ASR system, with improvements when using up to 1024 tokens of LM context with 80s of acoustic context. Our best results achieved through this system combination are competitive with the current state-of-the-art, while using a fraction of the compute, demonstrating the benefit of accounting for long-format data more appropriately.

While this work investigates a range of context sizes, these results are attained from a fixed model architecture. Altering various factors such as the number of layers could potentially affect the use of context, which we plan to explore in future work. Finally, results exploring the use of an entire hour of context illustrate a potential limit to performance gains from increasing the context for attention based ASR models. Different modelling paradigms may need to be investigated or developed in order to benefit from truly long-contexts of entire meetings/talks.

6. REFERENCES

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

[2] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A N Gomez, L Kaiser, and I Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[3] O Press, N A Smith, and M Lewis, "Shortformer: Better language modeling using shorter inputs," *arXiv preprint arXiv:2012.15832*, 2020.

[4] T Hori, N Moritz, C Hori, and J L Roux, "Advanced long-context end-to-end speech recognition using context-expanded transformers," *arXiv preprint arXiv:2104.09426*, 2021.

[5] Zhiyun L, Y Pan, T Doutre, P Haghani, L Cao, R Prabhavalkar, C Zhang, and T Strohman, "Input length matters: Improving rnn-t and mwer training for long-form telephony speech recognition," *arXiv preprint arXiv:2110.03841*, 2021.

[6] A Radford, J W Kim, T Xu, G Brockman, C McLeavey, and I Sutskever, "Robust speech recognition via large-scale weak supervision," in *ICML*. PMLR, 2023, pp. 28492–28518.

[7] G Sun, C Zhang, and P C Woodland, "Transformer language models with lstm-based cross-utterance information representation," in *ICASSP*. IEEE, 2021, pp. 7363–7367.

[8] R Flynn and A Ragni, "Leveraging cross-utterance context for asr decoding," *arXiv preprint arXiv:2306.16903*, 2023.

[9] SH Chiu, TH Lo, FA Chao, and B Chen, "Cross-utterance reranking models with bert and graph convolutional networks for conversational speech recognition," in *APSIPA ASC*. IEEE, 2021, pp. 1104–1110.

[10] A Gulati, J Qin, CC 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.

[11] A Graves, S Fernández, F Gomez, and J Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[12] D Rekesh, S Kriman, S Majumdar, V Noroozi, H Huang, O Hrinchuk, A Kumar, and B Ginsburg, "Fast conformer with linearly scalable attention for efficient speech recognition," *arXiv preprint arXiv:2305.05084*, 2023.

[13] T Dao, D Y Fu, S Ermon, A Rudra, and C Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," 2022.

[14] C Li, M Zhang, and Y He, "The stability-efficiency dilemma: Investigating sequence length warmup for training gpt models," *NeurIPS*, vol. 35, pp. 26736–26750, 2022.

[15] Y Bengio, J Louradour, R Collobert, and J Weston, "Curriculum learning," in *ICML*, 2009, pp. 41–48.

[16] A Clifton, S Reddy, Y Yu, A Pappu, R Rezapour, H Bonab, M Eskevich, G Jones, J Karlgren, B Carterette, and R Jones, "100,000 podcasts: A spoken English document corpus," in *COLING*. Dec. 2020, pp. 5903–5917, ICCL.

[17] F Hernandez, V Nguyen, S Ghannay, N Tomashenko, and Y Esteve, "Ted-lium 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *SPECOM*. Springer, 2018, pp. 198–208.

[18] M Del Rio, P Ha, Q McNamara, C Miller, and S Chandra, "Earnings-22: A practical benchmark for accents in the wild," *arXiv preprint arXiv:2203.15591*, 2022.

[19] S Gandhi, P Von Platen, and A M Rush, "Esb: A benchmark for multi-domain end-to-end speech recognition," *arXiv preprint arXiv:2210.13352*, 2022.

[20] P Lison and J Tiedemann, "Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles," 2016.

[21] L Gao, S Biderman, S Black, L Golding, T Hoppe, C Foster, J Phang, H He, A Thite, N Nabeshima, S Presser, and C Leahy, "The Pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[22] J W Rae, A Potapenko, S M Jayakumar, C Hillier, and T P Lillicrap, "Compressive transformers for long-range sequence modelling," *arXiv preprint*, 2019.

[23] J Nozaki and T Komatsu, "Relaxing the conditional independence assumption of ctc-based asr by conditioning on intermediate predictions," *arXiv preprint arXiv:2104.02724*, 2021.

[24] S Ioffe and C Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*. pmlr, 2015, pp. 448–456.

[25] S Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," *NeurIPS*, 2017.

[26] J Su, Y Lu, S Pan, A Murtadha, B Wen, and Y Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint arXiv:2104.09864*, 2021.

[27] A Henry, P R Dachapally, S Pawar, and Y Chen, "Query-key normalization for transformers," *arXiv preprint arXiv:2010.04245*, 2020.

[28] N Shazeer, "Fast transformer decoding: One write-head is all you need," *CoRR*, vol. abs/1911.02150, 2019.

[29] B Zhang and R Sennrich, "Root mean square layer normalization," *NeurIPS*, vol. 32, 2019.

[30] J L Ba, J R Kiros, and G E Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[31] Wenxiao Wang, Wei Chen, Qibo Qiu, Long Chen, Boxi Wu, Binbin Lin, Xiaofei He, and Wei Liu, "Crossformer++: A versatile vision transformer hinging on cross-scale attention," *arXiv preprint arXiv:2303.06908*, 2023.

[32] N Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[33] Z Dai, Z Yang, Y Yang, J Carbonell, Q Le, and R Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *ACL*. July 2019, pp. 2978–2988, Association for Computational Linguistics.

[34] A Defazio and S Jelassi, "Adaptivity without compromise: a momentumized, adaptive, dual averaged gradient method for stochastic optimization," *J Mach Learn Res*, vol. 23, 2022.

[35] V Srivastav, S Majumdar, N Koluguri, A Moumen, S Gandhi, et al., "Open automatic speech recognition leaderboard," `https://huggingface.co/spaces/hf-audio/open_asr_leaderboard`, 2023.

[36] T Likhomanenko, Q Xu, V Pratap, P Tomasello, J Kahn, G Avidov, R Collobert, and G Synnaeve, "Rethinking evaluation in asr: Are our models robust enough?," *arXiv preprint arXiv:2010.11745*, 2020.