INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Leveraging Cross-Utterance Context For ASR Decoding

Robert Flynn, Anton Ragni

Department of Computer Science, University of Sheffield, 211 Portobello, Sheffield S1 4DP, UK
rflynn2@sheffield.ac.uk, a.ragni@sheffield.ac.uk

Abstract

While external language models (LMs) are often incorporated into the decoding stage of automated speech recognition systems, these models usually operate with limited context. Cross utterance information has been shown to be beneficial during second pass re-scoring, however this limits the hypothesis space based on the local information available to the first pass LM. In this work, we investigate the incorporation of long-context transformer LMs for cross-utterance decoding of acoustic models via beam search, and compare against results from n-best rescoring. Results demonstrate that beam search allows for an improved use of cross-utterance context. When evaluating on the long-format dataset AMI, results show a 0.7% and 0.3% absolute reduction on dev and test sets compared to the single-utterance setting, with improvements when including up to 500 tokens of prior context. Evaluations are also provided for Tedlium-1 with less significant improvements of around 0.1% absolute.

Index Terms: speech recognition, language modelling, cross-utterance, beam-search, rescoring

1. Introduction

The decoding stage of end-to-end automated speech recognition (ASR) systems often benefits from the use of an external language model. For Connectionist temporal classification (CTC) [1] based acoustic models (AM) this is often crucial for good performance due to their conditional independence assumption. Traditionally, n-gram models have been used for this task, although due to data scarcity these models are usually restricted to a limited context of 3-4 preceding words. Consequently, neural models are often applied during second pass [2, 3, 4, 5, 6, 7, 8, 9] or first pass decoding [10, 11, 12, 13, 14] stages for further reductions in word error rate (WER).

The advent of the transformer architecture [15] has lead increased performance on many tasks, including ASR. In part, this can be attributed to its ability to effectively propagate gradients across long-distances, enabling the learning of much longer-term dependencies. The use of external transformer language models (TLMs) to improve ASR has been investigated in prior work [3, 16, 2, 4], with results demonstrating an advantage over recurrent-based architectures.

While many realistic use cases for ASR will involve long and continuous conversations or talks, ASR systems are mainly trained over very short utterances with a (often invalid) $i.i.d$ assumption. Likewise decoding/re-scoring is typically performed over individual utterances, which can lead to context fragmentation reductions in performance.

The contributions made as part of this work are as follows: 1. We assess the benefit of cross-utterance information for decoding, including how much context is useful and the impact of errors in the history. 2. We demonstrate that external language model integration via beam search improves the ability to utilise cross-utterance information compared to re-scoring. 3. A set of adaptions from prior work [17, 18] are proposed for use in this task making shallow fusion with TLMs more feasible.

The remainder of the paper is organized in the following manner: Section 2 provides a brief overview of related work, Section 3 details our method including adaptions made for efficient decoding (3.3) and language modelling in a conversational setting (3.4). Experimental details such as model and decoding configuration, and datasets is provided in section 4. Results and key findings are given in Section 5 with our conclusion in Section 6.

2. Related work

Long-range linguistic context has been previously been exploited in ASR through the use of external neural language models. In [2, 5, 8, 9] cross-utterance information is found to be beneficial for perplexity (PPL) and WER during second-pass re-scoring with LSTM and transformer type models. Similar findings are presented in [7] where LSTMs are adapted for a conversational setting and effectively used for re-scoring with an extended history. However it is not clear from these works how much context is useful for re-scoring. First pass cross-utterance decoding using LSTM models has previously been explored [10] using on-the-fly composition with a WSFT [19] based decoder. This work shows some performance benefit for using the LSTM during the initial pass compared to re-scoring. No benefit is found to using greater than 4 sentences of prior context, which may be due limitations of LSTMs use of context [20].

3. Transformer Decoding

3.1. Transformer Language Modelling

While typically a transformer [15] may consist of a bi-directional encoder, followed by a causal decoder, for language modelling we use the decoder-only variant transformer. This consists of alternating multi-headed self-attention with a causal mask, and feed-forward modules.

Given a word sequence $\boldsymbol{w} = (w_1, .., w_T)$ causal language models are trained to estimate the conditional probability of $P(w_t|\boldsymbol{w}_{<t})$. Word sequence probabilities can then be obtained by an expansion resulting in $P(\boldsymbol{w})$, when working with log-likelihoods this equates to: $\sum_{t=1}^{T} \log P(w_t|\boldsymbol{w}_{<t})$. For the purpose of decoding these likelihoods can be treated as scores and combined with the AM through a log-linear interpolation.

10.21437/Interspeech.2023-1941

3.2. Self-Attention

Self attention, which is one of the key components of TLMs, is a *mixing* operation between input tokens that enables the learning of local and long-range statistical patterns. The input X is first transformed to obtain the query, key and value matrices.

$$Q, K, V = XW^Q, XW^K, XW^V \tag{1}$$

In this work we use the key query normalisation variant of attention [21].

$$\hat{q}_i = \frac{q_i}{\|q_i\|}, \hat{k}_i = \frac{k_i}{\|k_i\|} \tag{2}$$

This involves applying L_2 normalisation to the keys and queries, obtaining their dot product, and scaling the result by a learnt parameter g. To encode positional information information an additional matrix P is added as a bias to the similarity matrix, alongside a causal mask M to prevent interactions with future tokens, before the softmax is applied. The resultant distribution is used to obtain a weighting of the value vectors, which is used to produce inputs to the following layer.

$$\text{Attention}(\hat{Q}, \hat{K}, V) = \text{Softmax}(g \cdot \hat{Q}\hat{K}^T + P + M)V \tag{3}$$

3.3. Adaptions for Efficient Decoding

3.3.1. Key-Value Caching

Unlike recurrent-based networks TLMs do not condense the history into a single representation, and instead process their input in parallel. For each token or utterance passed to the model we want to avoid re-computing the previous history as this would be extremely costly. As the language model is causal, and prior states are not updated based on future information, it is possible to cache keys and values from the attention sub-layers to avoid re-computation. This is not an approximation, and can be viewed as computing the attention in a sequential manner which is explored in prior work [18, 22]. This enables efficient cross-utterance decoding using TLMs with a linear increase in memory for each token in the cache.

To achieve this we simply concatenate keys $\hat{K}$ and values V from previous timesteps $t_{0:t-1}$ that are cached during each forward pass with the data from the current timestep.

$$\hat{K}_{0:t}, V_{0:t} = (\hat{K}_{0:t-1}, \hat{K}_t), (V_{0:t-1}, V_t) \tag{4}$$

With $\hat{Q}_t, \hat{K}_{0:t}, V_{0:t}$ acting as input to equation 3 for this method. To limit the sequence length at inference time, the cache can be truncated, when doing this we find it necessary to maintain the beginning of sentence token at the start of the history.

3.3.2. Multi-Query Attention

Additionally, we employ *Multi-Query* Attention [17], which uses only 1 head for both the keys and the values. This reduces slowdown due to memory bandwidth, with negligible performance degradation, and was recently validated in large-scale training of PaLM [23]. When decoding incrementally with a batch size of 25 and cache/history size of 500 tokens, this improves our iteration time by **74%**.

3.4. Conversational Language Modelling

As the language model is operating over series of utterances $U = (u_0, ...u_t)$, where utterance boundaries may indicate speaker changes, or changes in topic we choose to model this with a few modifications to the architecture. These modifications are illustrated in figure 1.

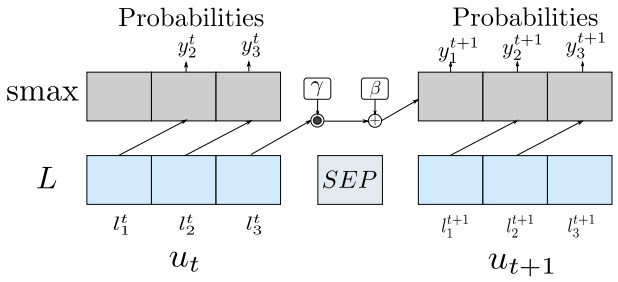

Figure 1: *Modifications for language modelling in a conversational setting.*

A separator token SEP is introduced to denote utterance boundaries, this is passed to the model alongside the input at the end of each utterance. There are no targets for this token and it is simply used to denote a boundary in the history.

To predict the first token in a following utterance, we introduce initial token prediction, where learnt per dimension scalar γ and offset β transformations are applied to the logit predictions from the last token in the current utterance. Specifically, given a sequence of logits $L^t = (l_1^t ... l_n^t)$ from utterance u_t, probabilities for the first token in the following utterance y_0^{t+1} are obtained as follows:

$$y_1^{t+1} = \text{Softmax}(l_n^t \odot \gamma + \beta) \tag{5}$$

Hence the model can learn to modulate probabilities for tokens that may be more, or less likely at the beginning of an utterance. The inclusion of eq. 5 was motivated by the frequent speaker changes in AMI [24], with new utterances often beginning with a hesitation such as "hmm".

3.5. Beam Search

Language models can be combined with the output probabilities of an AM using beam search. Ideally, we would like to find the most probable path across all alignments using all models. However, this is infeasible so we approximate by restricting the search to a set of top paths or beams. For each time-step t we use the AM probabilities and look-ahead probabilities from the TLM to score the vocabulary indices at that time-step i_t.

$$\text{Score}(i_t) = \log P_{AM}(i_t) + \text{Score}_{LM}(i_t) \tag{6}$$

Our AM uses the CTC decoding algorithm [1] which features blank tokens θ as part of the vocabulary, and allows for repetitions which are collapsed as part of decoding. As the TLM does not provide probabilities for repeats $i_t = i_{t-1}$ or blanks θ an insertion bonus β is used to prevent the search favouring these tokens. Additionally, a scaler α is used to weight the TLMs score. Hence the TLMs score is as follows [12]:

$$\text{Score}_{LM}(i_t) = \begin{cases} 0, & \text{if } i_t = \theta \text{ or } i_t = i_{t-1} \\ \alpha \log P_{LM}(i_t) + \beta, & \text{otherwise} \end{cases} \tag{7}$$

4. Experimental Setup

4.1. Model and Training Specifications

The TLMs all have 12 layers with a hidden dimension of 256. For the feedforward blocks, SwiGLU layers [25] are used with

an expansion factor of 4. Key query normalised attention is used in-place of dot product attention as in [21]. 8 heads are used for the queries during attention with 1 head for the keys and values [17]. For positional encoding, the *dynamic position bias* proposed in [26, 27] is used. TLMs are trained without dropout, as they under-fit the pre-training corpus. In total the TLMs amount to around 11M parameters.

During training of the TLM windows of up to 25 prior utterances are used as context. A TLM is pre-trained and fine-tuned for each dataset, in order to match the AMs vocabulary and training takes around 5 days on a GTX 3060 GPU.

For the AM, we use an encoder-only Conformer architecture [28] trained with self-conditioned CTC and intermediate losses [29]. Batch normalisation [30] is swapped out in favour of batch renormalisation [31]. Key query normalised attention and dynamic position biases are also used with the AM. For both AMs a byte pair vocabulary of size 128 is used. For the Tedlium dataset, the model has 12 layers, a hidden dimension of 256, 8 attention heads, and a convolution kernel width of 31. For AMI a smaller hidden dimension of 176 is used with 4 attention heads, 16 layers, and a convolution kernel width of 15. SpecAugment [32] is used during training alongside dropout (0.1 on feedforwards and 0.3 post-attention) for regularisation. Training of the AMs takes around 2 days on AMI and 4 days on Tedlium using a V100 GPU.

Models are trained using the Madgrad optimizer [33] with weight decay of $1e-6$. An EMA of model parameters set to 0.9999 is used for the AM, and during finetuning of the TLM.

4.2. Beam Search

For beam search a beam width of 25 is used. The search is constrained at each time-step to indices within a given threshold of the argmax of the AM probabilities (cut-off threshold). The insertion bonus β, TLM weighting α and the cut-off threshold were trained through a random search. The ranges explored in the search are as follows: For α $[0.0, 1.0]$, β $[-0.1, 0.8]$, and $[-4, -12]$ for the cutt-off threshold. Only the top beam is maintained when transferring across utterance boundaries, this helps reduce the propagation of errors found in low probability beams from previous utterances. The beam search is parallelised over each utterance in the single utterance setting, or over each meeting/talk for cross-utterance evaluation using the ray library [34]. Single-utterance decoding takes around 10 minutes and 40 minutes on Tedlium and AMI, and 35-55 minutes and 2.5-3.5 hours in the cross-utterance setting, however our implementation is not optimized and is implemented in python.

4.3. N-best Rescoring

To obtain the N-best list, beam search with a width of 1000 is used with a n-gram trained with the same BPE vocabulary as the AM, this is then re-ranked through a word-level n-gram model and pruned to an n-best list of 100 hypotheses. The TLM is combined with first-pass models through weighted addition of the log probabilities. Additionally, a length penalty is included. These weights/hyperparameters are trained through a random search. For best performance it was necessary to standardise the log probabilities of the TLM using mean and standard deviation statistics from the top hypotheses.

4.4. Datasets

4.4.1. Tedlium 1

Tedlium [35] is an ASR corpus consisting of single-speaker TED talks. This data is selected as it is an ideal use-case for long-range ASR, with many unique speakers using varied language, talking continuously for periods of up to 18 minutes. Despite this there is, too our knowledge, no work that attempts to fully utilise this longer time-frame for ASR. For this work we use the smaller 1^{st} release of Tedlium. For pre-processing, spaces between the apostrophe in contraction are removed. In this data there are instances of large un-annotated gaps between utterances (this is alot less prevalent in the test and dev data). To address this, when training cross-utterance TLMs the cache is not propagated over gaps larger than ten seconds. Statistics for this corpus are provided in table 1.

	Total Words	Duration (h)	Recordings
Train	764,088 / 1,315,797	80.1 / 118.1	132 / 774
Dev	95,374 / 17,733	9.8 / 1.6	18 / 8
Test	89,978 / 27,500	9.4 / 2.6	16 / 11

Table 1: *Corpus statistics for AMI / Tedlium-1*

4.4.2. AMI

AMI [24] consists of multi-speaker meetings. The individual headphone microphone (IHM) version is used in this work. This data was selected due to it's long-format conversational setting and it's use in similar, prior work. However, features of this dataset such as, very frequent speaker changes and speaker overlaps, may present difficulties for measuring the long-range performance of ASR systems in isolation.

Utterances in the training data longer than 16 seconds are re-segmented using the provided time-stamps, while dev and test splits are kept intact. Statistics are provided in table 1

4.4.3. OpenSubtitles

As a pre-training corpus for the TLMs OpenSubtitles[1] [36], is used. We pretrain on a subset of the corpus for containing 408,985,555 words for one epoch. Numbers and monetary values are converted to their orthographic form, and all text is set to lower case and punctuation excluding apostrophes is removed.

5. Experimental Results

Perplexities for the TLMs are provided in table 2. Here the models show continued decreases in PPL which begins to plateau at 1000 tokens of prior context.

Table 4 presents our baseline results for both greedy decoding, and the initial first-pass which is used in re-scoring. Notably, our baselines achieve good results, performing better on both datasets than other single-domain AMs in the literature which do not incorporate i-vectors [2, 9, 37].

Results for re-scoring and beam search are given in table 3. The presentation of results for **AMI** is split into sub-sections that are based on the key components of this work, lastly we overview and discuss performance on **Tedlium** separately.

5.1. Re-scoring vs Beam Search

For decoding via beam search on AMI we see sizeable improvement over re-scoring, both in the utilisation of context, and more generally in the single-utterance setting. When zero prior context is available we see a 2.4% and 2.5% (dev and test) absolute decrease over the greedy decoding baseline for beam search, compared to 2.0% and 2.1% for re-scoring.

With additional context we find much more consistent improvements with beam search. For example, with the full 500

[1] `http://www.opensubtitles.org/`

Dataset	0	50	100	250	500	1000
AMI	89.77 / 75.49	74.37 / 67.82	71.43 / 66.12	68.18 / 64.47	66.53 / 63.80	**64.89 / 62.54**
Tedlium	148.72 / 136.21	132.47 / 114.60	126.79 / 118.96	119.70 / 108.10	117.86 / 104.63	**116.33 / 101.91**

Table 2: *Perplexity (PPL) for TLMs on dev / splits at (0, 50, 100, 250, 500) tokens of context from preceding utterances*

Dataset	Decoding Method	0	50	100	250	500
AMI	Rescoring	20.96 / 19.41	**20.88** / 19.31	20.88 / 19.31	20.89 / **19.30**	20.88 / 19.30
	Rescoring (GTH)	20.96 / 19.41	20.84 / 19.26	**20.83** / 19.26	20.83 / 19.23	20.83 / **19.21**
	Beam Search	20.52 / 18.98	19.82 / 18.75	**19.78** / 18.72	19.78 / 18.67	19.80 / **18.65**
	Beam Search (GTH)	20.52 / 18.98	19.71 / 18.68	19.70 / 18.65	18.67 / 18.62	**19.66 / 18.57**
Tedlium	Rescoring	9.51 / 8.48	9.52 / 8.44	9.52 / 8.45	**9.50 / 8.44**	9.52 / 8.45
	Rescoring (GTH)	9.51 / 8.48	9.48 / 8.42	9.47 / **8.41**	**9.44** / 8.42	9.50 / 8.42
	Beam Search	9.73 / 8.61	9.69 / 8.54	9.69 / 8.48	**9.67 / 8.47**	9.72 / 8.48
	Beam Search (GTH)	9.73 / 8.61	9.62 / 8.52	9.59 / 8.48	**9.56 / 8.35**	9.61 / 8.41

Table 3: *Results (WER) for each decoding method on AMI and Tedlium with (0, 50, 100, 250, 500) tokens of context from preceding utterances. Results are given for dev / test sets. GTH denotes the use of the Ground Truth transcripts to form the History.*

Dataset	Dev	Test
Tedlium	11.73 / 9.89	10.37 / 8.80
AMI	22.94 / 21.09	21.49 / 19.58

Table 4: *Baselines WERs for each dataset, results are given without/with an n-gram language model*

tokens of prior context beam search shows an absolute decrease over the single-utterance setting of 0.7% and 0.3% for dev and test respectively. For re-scoring 0.1% and 0.1% absolute decreases are attained with the additional context. Hence we find that re-scoring is limiting the use of the context with improvements from beam search becoming more pronounced in the cross-utterance setting.

5.2. How much context is useful?

Unsurprisingly, on the AMI corpus, decoding benefits most from including the recent history of the previous 50 tokens, with gains of 0.7% and 0.2% (dev and test) absolute over the single-utterance setting using beam search. On average there is 23.7 tokens per utterance on AMI, hence this amounts to around 2 prior utterances of context. For the test set we can see continued improvements with up to 500 tokens of prior context with a further 0.1% absolute decrease when over 50 tokens of context. This is on average 21 utterances of context that we are able to benefit from, increasing beyond 500 tokens brought no further improvements. On the dev set while we see significant gains from the cross-utterance context of 0.7% absolute there is no benefit from increasing the context beyond 100 tokens.

5.3. Impact of errors in the context

During the training the model is provided with ground truth transcripts, while during decoding, only the decoded history is available. To examine the effect of this mismatch on the use of the context we present additional results where during decoding the transcripts from previous utterances are used as the context. This is denoted as Ground Truth Histoy (GTH) in table 3.

For both dev and test dataset on AMI when decoding via beams search we see around an additional 0.1% absolute decrease when using the GTH across all context lengths. Additionally, with the GTH we see continued improvements on the dev set until 500 tokens of context, while otherwise this plateaus at 100 tokens. Similar findings are seen for re-scoring. As such, we find that test-time errors have a consistent impact on the use of the history. This does not account for the impact of errors within the current utterance, which is likely more severe.

5.4. Performance on Tedlium

On the Tedlium data re-scoring provides an improvement over beam search of 0.2% and 0.1% absolute in the single utterance setting. During re-scoring the TLM benefits from interpolation with the first-pass models. Consequently, this is likely due to the TLMs much higher perplexity on this corpora, causing greater reliance on the n-gram models.

As with AMI we find that beam search enables better utilisation of the context with around a 0.1% decrease over the single-utterance setting on the test data, compared to fairly insignificant decrease of less than 0.1% for re-scoring. For both decoding methods we see an increase in WER when using 500 tokens of context. Additionally, there is no benefit from the context on the development set when using the model outputs as history. However, for the GTH evaluations there is around a 0.2% absolute decrease over single-utterance when using beam search suggesting that the poor performance on this split may be due to misleading errors in the history.

6. Conclusion

In this work, we examined the benefit of context from prior utterances during the decoding stage of ASR. Up to 500 tokens were found to be helpful when decoding on AMI with reductions in a similar range to prior work. From Tedlium there are much smaller gains from the context. However, it is difficult to tell to what degree this is a result of our method or the prevalence of useful long-range dependencies in this dataset.

It was found that decoding directly via beam search allowed for greater use of the context, with increasing improvements compared to re-scoring as more context was made available. As we find better results on AMI where the TLMs show a much lower PPL, it is reasonable to assume that increasing the scale of our models would provide improved results on both datasets. Additionally, due to our findings that the use of context is limited during re-scoring due to the first-pass models, it is likely also limited by the AMs implicit language model. Ideally, cross-utterance information would be incorporated during both encoding and decoding stages, which can be investigated as part of future work.

7. References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[2] G. Sun, C. Zhang, and P. C. Woodland, "Transformer language models with lstm-based cross-utterance information representation," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7363–7367.

[3] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language modeling with deep transformers," *arXiv preprint arXiv:1905.04226*, 2019.

[4] H. Huang and F. Peng, "An empirical study of efficient asr rescoring with transformers," *arXiv preprint arXiv:1910.11450*, 2019.

[5] S. Parthasarathy, W. Gale, X. Chen, G. Polovets, and S. Chang, "Long-span language modeling for speech recognition," *arXiv preprint arXiv:1911.04571*, 2019.

[6] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5934–5938.

[7] W. Xiong, L. Wu, J. Zhang, and A. Stolcke, "Session-level language modeling for conversational speech," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2764–2768. [Online]. Available: https://aclanthology.org/D18-1296

[8] G. Sun, C. Zhang, and P. C. Woodland, "Cross-utterance language models with acoustic error sampling," *arXiv preprint arXiv:2009.01008*, 2020.

[9] S.-H. Chiu, T.-H. Lo, F.-A. Chao, and B. Chen, "Cross-utterance reranking models with bert and graph convolutional networks for conversational speech recognition," in *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2021, pp. 1104–1110.

[10] X. Chen, S. Parthasarathy, W. Gale, S. Chang, and M. Zeng, "Lstm-lm with long-term history for first-pass decoding in conversational speech recognition," *arXiv preprint arXiv:2010.11349*, 2020.

[11] S. Toshniwal, A. Kannan, C.-C. Chiu, Y. Wu, T. N. Sainath, and K. Livescu, "A comparison of techniques for language model integration in encoder-decoder speech recognition," in *2018 IEEE spoken language technology workshop (SLT)*. IEEE, 2018, pp. 369–375.

[12] T. Zenkel, R. Sanabria, F. Metze, J. Niehues, M. Sperber, S. Stüker, and A. Waibel, "Comparison of decoding strategies for ctc acoustic models," *arXiv preprint arXiv:1708.04469*, 2017.

[13] K. Hwang and W. Sung, "Character-level incremental speech recognition with recurrent neural networks," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 5335–5339.

[14] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[15] 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.

[16] P. Pandey, S. D. Torres, A. O. Bayer, A. Gandhe, and V. Leutnant, "Lattention: Lattice-attention in asr rescoring," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7877–7881.

[17] N. Shazeer, "Fast transformer decoding: One write-head is all you need," *CoRR*, vol. abs/1911.02150, 2019. [Online]. Available: http://arxiv.org/abs/1911.02150

[18] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. [Online]. Available: https://aclanthology.org/P19-1285

[19] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[20] U. Khandelwal, H. He, P. Qi, and D. Jurafsky, "Sharp nearby, fuzzy far away: How neural language models use context," *arXiv preprint arXiv:1805.04623*, 2018.

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

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

[23] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[24] W. Kraaij, T. Hain, M. Lincoln, and W. Post, "The ami meeting corpus," 2005.

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

[26] W. Wang, L. Yao, L. Chen, B. Lin, D. Cai, X. He, and W. Liu, "Crossformer: A versatile vision transformer hinging on cross-scale attention," *arXiv preprint arXiv:2108.00154*, 2021.

[27] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong *et al.*, "Swin transformer v2: Scaling up capacity and resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12 009–12 019.

[28] 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.

[29] 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.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[31] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," *Advances in neural information processing systems*, vol. 30, 2017.

[32] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[33] A. Defazio and S. Jelassi, "Adaptivity without compromise: a momentumized, adaptive, dual averaged gradient method for stochastic optimization," *J Mach Learn Res*, vol. 23, pp. 1–34, 2022.

[34] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan *et al.*, "Ray: A distributed framework for emerging {AI} applications," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 561–577.

[35] A. Rousseau, P. Deléglise, and Y. Esteve, "Ted-lium: an automatic speech recognition dedicated corpus." in *LREC*, 2012, pp. 125–129.

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

[37] M. Yang, I. Lane, and S. Watanabe, "Online continual learning of end-to-end speech recognition models," *arXiv preprint arXiv:2207.05071*, 2022.