INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

LVCSR with Transformer Language Models

Eugen Beck, Ralf Schlüter, Hermann Ney

Human Language Technology and Pattern Recognition, Computer Science Department,
RWTH Aachen University, 52074 Aachen, Germany
{beck, schlueter, ney}@cs.rwth-aachen.de

Abstract

Neural network language models (LMs) based on self-attention have recently outperformed the previous state of the art, LSTM LMs. Transformer LMs today are often used as a postprocessing step in lattice or n-best list rescoring. In this work the main focus is on using them in one-pass recognition. We show that by a simple reduction of redundant computations in batched self-attention we can obtain a 15% reduction in overall RTF on a well-tuned system. We also show that through proper initialization the layer normalization inside the residual blocks can be removed, yielding a further increase in forwarding speed. This is done under the constraint of staying close to state-of-the-art in terms of word-error rate (5.4% on LibriSpeech test-other) and achieving a real-time factor of around 1. Last but not least we also present an approach to speed up classic push-forward rescoring by mixing it with n-best list rescoring to better utilize the inherent parallelizability of Transformer language models, cutting the time needed for rescoring in half.

Index Terms: speech recognition, decoding, Transformer language model

1. Introduction

Language Models based on the Transformer architecture [1] have recently become very popular [2, 3, 4] as they outperform the previously state-of-the-art LMs based on Long Short-Term Memory (LSTM) [5]. For LSTM LMs multiple prior works exist on using them in one-pass recognition [6, 7, 8, 9, 10, 11] and most recently also in low-latency streaming [12]. The main focus of these works is on caching as many of the computations as possible or quantizing and recombining state. Another common requirement for real-time performance on large-vocabulary[1] tasks is the avoidance of a naive softmax to compute the final output probabilities as this is computationally too inefficient when only a small fraction is needed during decoding. Often Noise-Contrastive Estimation (NCE) [13, 14] is used to obtain a self-normalized output-layer where word probabilities can be computed independently of each other.

Transformer LMs are now also gaining traction and are for example used in lattices/n-best lists rescoring [15, 16]. One challenge with these new models is the growth of the models internal state with the number of inputs. It is precisely this growth that allows the model to capture longer / more complex dependencies. The linear growth in memory is also accompanied by a quadratic increase in computation time, posing an additional challenge. Apart from using Transformer LMs directly one can argue that an end-to-end trained Transformer acoustic model (AM) is also a Transformer language model as for example in [16, 17]. Especially the label-encoder part of the Transformer-transducer architecture [17] is apart from the missing output layer an Transformer LM. In addition they perform shallow fusion with a grapheme based Transformer LM. To make the model streamable the authors propose to limit the attention of the audio- and label-encoder to fixed size contexts. Other works concerning efficient acoustic Transformer AMs focus on efficient attention computations by using locality sensitive hashing [18] or block-wise attention with context propagation [19]. For Transformer LMs similar optimizations were proposed by e.g. [20], where the attention spans of each attention head are automatically learned and capped. Others propose modifications to the self-attention mechanism itself by using tensor decomposition to reduce the computational effort required [21].

Apart from [21], most works focus on reducing the temporal range over which self-attention is performed. As the number of inputs for our word-level Transformer LM are much fewer than in acoustic models or character language models we focus on training more compact models starting with the work from [22]. In this work the focus was on trading self attention layers for more and wider residual blocks between the layers.

Another important aspect of Transformer LMs is their inability to generalize to sequence lengths far beyond what they have seen in training [23]. This is not a problem in practical systems as even in the case of a mismatch in training and test data the test data can be cut into smaller pieces by using a segmenter.

In light of these problems we will foremost focus on optimizing Transformer LMs for one-pass speech recognition. To that end, we first give a brief overview of the Transformer LM architecture. Then we explain our proposals to improve one- and two-pass performance which we afterwards evaluate on the LibriSpeech corpus. We also show some experiments regarding long-context recognition. The paper ends with conclusions.

2. Transformer LMs

The central component of Transformer neural networks is the self attention mechanism[1]. For a given input x_i at position i a self-attention layer will create 3 values (k_i, v_i, q_i) through an affine transformation of input x_i. These are the key, value and query for the attention mechanism:

$$\alpha_{i,j} = \frac{\exp(q_i \cdot k_j)}{\sum_{\tilde{j} \leq i} \exp(q_i \cdot k_{\tilde{j}})}$$

$$o_i = \sum_{j \leq i} \alpha_{i,j} v_j$$

This attention mechanism is usually performed with multiple independent heads. The residual output o_i is then added to the input x_i. In between two self-attention layers there are a number of wide residual feed-forward blocks[24]. These consist of two linear layers with a nonlinearity in-between. Often dropout is used as well, but following the training configuration of [3] we do not use dropout for our models.

[1] in this case roughly above 100-150k words

http://dx.doi.org/10.21437/Interspeech.2020-1164

3. Improvements to runtime performance

3.1. Better Training Hyperparameters

As stated before our Transformer architecture is carried over from [22]. In their setup, the model is trained with stochastic gradient descent and gradient clipping. The gradient clipping is chosen to be as large as possible while still obtaining model convergence. We noticed (by accident) that pretraining the model with a small gradient clipping limit allows us to use a larger gradient clipping limit later on. This yields a small decrease in perplexity such that the model is even closer to the very large Transformer LM from [3]. Note that the perplexity of the language model not only influences the final WER, but also the runtime performance as more focused LM distributions allow the decoder to prune more hypotheses.

3.2. Quantization of the LM-state

In our search implementation, the language model state is stored in CPU memory and thus needs to be transferred to and from GPU memory before and after each forwarding. To reduce memory consumption (especially for yet untuned systems where the search-space can be very large) we quantize the state and store it in 16-bit integers. To Simplify the quantization / dequantization procedure we perform quantization using a fixed scale (0.001). The effect of this on well tuned systems is rather small, but it is very helpful in early stages of optimization when the search space is rather large due to sub-optimal pruning parameters.

3.3. Common Prefix

As the self-attention mechanism processes the whole sequence at each step, decoding inevitably slows down for longer sequences. One counteracting factor is the fact that after some time all hypotheses will also have a common prefix that also grows over time. While batching multiple histories into one request is more efficient than processing each history individually, it still incurs a runtime-cost compared to just processing a single history. Thus we split the self-attention mechanism into two parts: The first part is the common prefix and operates with a batch size of one, the second is the batch for all individual suffixes. After computing the energies, these two parts are merged and the softmax operation is carried out for all histories in one batch. This also reduces the amount of state-data that needs to be transferred to the GPU.

3.4. Fixup initialization

In the classic Transformer architecture each residual and self-attention layer is preceded by a layer-normalization layer[25]. From a computational point of view layer-normalization consists of the computation of mean and variance and their application to the inputs (subtraction + division). While an individual layer-norm is relatively cheap (compared to e.g. the self-attention mechanism), the frequent use incurs significant computational overhead. In our model around 30% of the time was spent on its computation. To reduce it's impact we use the fixup normalization [26] to remove layer-norm from all residual blocks, keeping it only in front of the self-attention layers. As in [26] we add additional bias layers and a scaling layer to each residual block during training. For decoding, we integrate them into the already existing bias and weight layers to avoid computational overhead.

3.5. Hybrid Lattice/N-best Rescoring

In contrast to LSTM LMs, Transformer LMs do not have an explicit recurrence over the label position, i.e. given a word sequence within each layer, the output at each position can be computed in parallel. This makes it more efficient to process long word sequences within one mini-batch on devices with a high degree of hardware parallelization like GPUs. This can be exploited to do fast rescoring of n-best lists. Unfortunately n-best list rescoring often yields results that are worse than push-forward lattice rescoring, where hypotheses are rescored and recombined within each lattice node. In order to achieve lattice-rescoring level accuracy and still seize the parallelization opportunities presented by Transformer LMs we propose a hybrid lattice/n-best rescoring algorithm. To that end we modify the push-forward lattice rescoring algorithm as follows. Instead of rescoring the hypotheses within one node directly we just collect them and push them on to the next node until the number of hypotheses within one node surpasses a given threshold. The hypotheses within that node can be thought of as a partial n-best list that is rescored in one batch. The then rescored hypotheses are recombined and pruned and the expansion in n-best list style can continue again. A pseudocode formulation is given in algorithm 1.

Algorithm 1 Hybrid Lattice/N-best Rescoring, L is the set of nodes n in the lattice, O_n is the set of outgoing arks of node n, H_n is the set of hypotheses for node n, ρ is the threshold to start rescoring, S_a/S_h is the score for ark a / hypotheses h, W_h/W_a is the word-sequence for hypothesis h / ark a and $n(a)$ is the destination node for ark a

for n ∈ L **do**
  if $|H_n| > \rho$ or $|O_n| = 0$ **then**
    H_n = rescoreWithNewLM(H_n)
  end if
  for $a \in O_n$ **do**
    for $h \in H_n$ **do**
      $\tilde{h} = (W_h + W_a, S_h + S_a)$
      $H_{n(a)} = H_{n(a)} \cup \{\tilde{h}\}$
    end for
  end for
end for

4. Experiments

4.1. Corpus and Models

In this paper we present results on the LibriSpeech corpus [27]. Our acoustic model is the same as in [15]. It is a 6-layer BLSTM model trained on 960 hours of 40 dimensional Gammatone features [28] using the state-level minimum Bayes Risk (sMBR) criterion [29]. The output units of the acoustic models are tied triphone states obtained using a Classification and Regression Tree (CART). The total number of parameters is 152.5 million. The main Transformer LM we use consists of 6 self attention layers. Each self-attention layer except for the first is preceded by 7 residual blocks. The embedding layer contains 128-dimensional vectors and the intermediate representation size between self-attention and residual blocks is 512. Each residual block is 4096 units wide. In total 285.9 million parameters are used within that model.

Table 1: *Effect of LM pretraining*

Model	PPL		WER [%]	
	dev	test	dev-other	test-other
Baseline	56.77	59.39	5.0	5.4
+ Fixup	56.98	59.71	4.8	5.3
+ Pretraining	54.88	57.64	4.6	5.2

4.2. Hardware and Measurement Methodology

As in our previous work [11] each machine used for our experiments has two sockets with Intel Xeon E5-2620 v4 CPUs with a base-clock speed of 2.1Ghz and 4 Nvidia Geforce 1080Ti GPUs. Unless stated otherwise, our decoder ran in a single thread. As we are primarily using the GPU to do computations, we set the Tensorflow intra/inter_op_parallelism_threads to 1. To compute the real time factor (RTF), we measure the total wall-clock time required by the recognizer/rescorer to process all segments within the corpus and divide it by the total duration. This includes loading features from disk, forwarding them through the acoustic model and decoding / rescoring. Startup time is not included. Features are not extracted on the fly as it creates higher load on our fileserver and is not a major part during decoding anyway. In a research context, preextracting features for a common task is useful as they are required for many experiments. In a production streaming system, feature extraction can be offloaded into a separate thread and will only contribute to latency, but not (significantly) to the RTF.

4.3. Parameter tuning

As usual tuning of hyperparameters is very important to obtain good performance. For the training of Transformer LMs we optimized mainly the gradient clipping and model size and left other parameters untouched. For decoding we tune the scale for the language model and the lookahead language model (a bigram count model). We always run recognitions with different beam sizes, but histogram pruning limits[2] are only tuned for the smallest beam size that yields the best WER. We also use forced recombination after 10 words in the one-pass systems to speed up decoding without sacrificing WER.

4.4. LM Training

For the LM training we show in Table 1 the effects of our two proposed methods. The baseline is taken from [22]. First we employ fixup to remove the layer-norm from the residual blocks. This results in very small degradation in perplexity. The WER goes up, but this is most likely an effect of more extensive tuning for the fixup model. The main claim from us at this point is that applying the fixup normalization does not hurt perplexity or WER. Resetting the learning rate to 1.0 and increasing the gradient clipping from 1.0 to 2.0 helped us to decrease the perplexity and WER (on dev-other) by 4% relative. This improvement is not that large, but given that these models are very close to the best systems [15, 16, 30], we think that all improvements are worth the effort.

4.5. One-pass recognition

For our one-pass recognition setups we only provide results for the best set of hyperparameters. In Table 2 results for the dev-other and test-other corpora are reported. We did not tune our system for the clean parts of the LibriSpeech corpus as the lower WER correlates with a much smaller search space and thus even systems with little optimization perform quite well. The first aspect to note is that the one-pass recognition results here are very close to the results in [15] on which our system is based. WER increased by 0.1-0.2% on the dev/test-other corpus respectively while using a much smaller Transformer language model. Decreasing the beam[3] from 16 to 14 results in another 0.2% WER degradation, but also yields a RTF of around 1. Note that the share of the language model score computation of the total runtime is around 20%-30% (depending on beam size). As our acoustic model is a classic 3-state 10ms frame-shift model a large part of the time is spent in the expansion of the HMM.

Table 2: *WER/RTF for the best single-pass system on LibriSpeech*

Corpus	Beam	WER [%]	RTF	
			Total	LM
dev-other	16	4.6	1.83	0.40
	15	4.7	1.34	0.33
	14	4.8	0.93	0.26
test-other	17	5.2	2.56	0.49
	15	5.3	1.48	0.34
	14	5.4	1.04	0.27

Table 3: *Ablation experiments on dev-other*

Model	RTF for beam		
	14	15	16
full	0.93	1.34	1.83
- cache-prefix	1.02	1.37	1.87
- common-prefix	1.19	1.56	2.11
- state quantization	1.17	1.62	2.23
- LM pretraining	1.14	1.62	2.27
- fixup	1.26	1.74	2.37

We also performed ablation experiments to quantify the effect of various improvements to the decoder. As Table 3 show the main contributor to RTF gains is the common prefix optimization which provides up to 0.24 RTF for beam 16. Caching of the prefix is only effective for smaller beam sizes as for larger beams the probability of a change in the prefix increases due to the increase in concurrent histories. State quantization is also more effective for larger beams. Going to the LM without LM training does not change RTF significantly, but increases WER from 4.6% to 4.9%. The last step in our ablation experiment series is the step to remove the fixup initialization and add layer-norm. This again increases RTF for beam 16 by another 0.1. In total we decreased the RTF for the full system by around 25% by LM optimizations alone.

4.6. Long-context recognition

As mentioned before the forwarding time of Transformer LMs is quadratic in the length of the history. To quantify how large this effect is we merged segments from the same speaker+book

[2] 150k state hypotheses, 1000 word-end hypotheses

[3] beam refers here to the acoustic pruning threshold and not as in many "end-to-end" systems to the number of hypotheses.

Table 4: *Long-context recognition experiments on dev-other*

language model	long segments	history pruning	WER [%]	RTF
4-gram	no	no	8.3	1.77
	yes		8.6	2.01
Transformer	no		4.6	1.83
	yes		8.3	2.82
		yes	5.2	2.35

in the dev-other corpus into one contiguous segment. These vary in length from 100 to 2000 words. These do not necessarily form one contiguous excerpt from a book and sentences are sometimes missing in-between. Thus across sentence boundary context might get lost to some extend. As already observed in [23] Transformer LM performance drops significantly in the case of inputs that are significantly longer than seen in training. As can be seen in Table 4 the WER increases from 4.6% to 8.3% on dev-other for Transformer models while e.g. a count-model is hardly affected by the change with only a degradation of 0.3%. To recover most of the loss of the Transformer LM we also added history pruning. If the word history of a hypothesis exceeds a certain length we prune it back to only the last few words. In our experiments the optimum for the dev-other corpus is at 100 words for the pruning threshold and 15 words for the new history length. This shows that while the largest part of the WER loss stems from the length of the context a significant amount of it is also due to the mismatch in sentence boundaries between training and testing.

4.7. Rescoring

Table 5: *WER/RTF for rescoring with different first-pass LMs on the dev-other corpus, LM name format is #self-attention layers x #attention heads _ #residual block x #residual block size, HR = hybrid rescoring, RL=recombination limit, i.e. only this many words in a history are considered during recombination, resc. = recoring, LM=LM calculation in first-pass decoding and rescoring*

First-Pass LM	HR	beam /RL	WER [%]	real-time factor		
				total	resc.	LM
6x16_7x4096	no	15/4	4.7	1.29	0.15	0.40
		14/4	4.8	0.92	0.11	0.32
		14/2	4.9	0.86	0.13	0.31
	yes	15/4	4.7	1.25	0.08	0.32
		14/4	4.8	0.86	0.05	0.26
		14/2	4.9	0.79	0.07	0.25
3x8_5x1024	yes	15/4	4.9	1.18	0.12	0.24
3x8_5x2048			4.8	1.10	0.11	0.23
3x8_5x4096			4.7	1.09	0.11	0.22
3x16_5x4096			4.8	1.10	0.10	0.23
4x4_5x2048			4.8	1.18	0.11	0.25
4x8_5x2048			4.8	1.20	0.11	0.26
4x8_5x4096			4.8	1.12	0.10	0.25

For the rescoring experiments we focus on three aspects: Firstly how much does hybrid rescoring speed up the rescoring step. Secondly whether a two pass system can outperform a one-pass system. Furthermore we check if a smaller Transformer LM in the first-pass can further improve performance. In all experiments we use the first-pass LM from the previous section for rescoring and vary the the first pass LM only. In the first pass we do forced recombination in the first pass to reduce the search space in the first pass as in [11]. This is denoted by recombination limit in Table 5.

In the first 6 rows of Table 5 we see that across different beam sizes the hybrid rescoring is about twice as fast as the regular push-forward lattice rescoring algorithm. Due to the relatively slow first pass the total impact is not that significant. In another setting where the first pass recognizer is faster the relative impact would be more significant.

Comparing to the one-pass system we see that for a given WER the two-pass system is faster by around 7-8% relative, but this comes at the price of not being able to reach the best WER of 4.6% on dev-other. This is due to the reduction of the initial search space by forced recombination of LM contexts. Without it though the two-pass system becomes a one-pass system.

Lastly we want to compare the performance of the 6-layer Transformer LM with smaller LMs. We trained a set of different LMs with 3 and 4 self attention layers and different numbers of attention heads and residual block sizes. The decoding parameters for the smaller LMs are the same as the 4.7% WER system with the 6-layer LM. Indeed we see that most smaller Transformer LMs perform similarly to the full Transformer model in the first pass. The actual difference in most cases is less or equal to 0.03% absolute even if this is not reflected in the single digit precision WER, but there is a limit to how small the initial model can be as we see in the case of the smallest model (3x8+5x1024). For the model 3x8+5x4096 model the WER is the same as for the full model at a 0.16 better RTF. We did not try using count based LMs for the first-pass as we have already shown in [11] that the lattices produced by a count-LM yield worse WER.

5. Conclusions

In this work we have demonstrated how to train and use Transformer LMs efficiently in one- and two-pass LVCSR. A simple pretraining helps to improve the perplexity of the LM and exploiting common prefixes in decoding and quantization of the LM state help to reduce the RTF and memory consumption of the model. In total we have improved the WER by 4% and the RTF by 25% on a strong baseline. We have also shown that utilizing hybrid n-best/lattice rescoring further helps to improve speed in a two-pass system by a factor of 2.

6. Acknowledgements

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694537, project "SEQCLAS") and from a Google Focused Award. The work reflects only the authors' views and none of the funding parties is responsible for any use that may be made of the information it contains. We also want to thank Kazuki Irie for providing us with the baseline compact Transformer LM and many hints on LM training.

7. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 5998–6008.

[2] 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.

[3] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language Modeling with Deep Transformers," in *Proc. Interspeech 2019*, 2019, pp. 3905–3909.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, pp. 4171–4186.

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

[6] Z. Huang, G. Zweig, and B. Dumoulin, "Cache based recurrent neural network language model inference for first pass speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6354–6358.

[7] T. Hori, Y. Kubo, and A. Nakamura, "Real-time one-pass decoding with recurrent neural network language model for speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6364–6368.

[8] K. Lee, C. Park, I. Kim, N. Kim, and J. Lee, "Applying GPGPU to recurrent neural network language model based fast network search in the real-time LVCSR," in *Proc. Interspeech 2015*. ISCA, 2015, pp. 2102–2106.

[9] K. Lee, C. Park, N. Kim, and J. Lee, "Accelerating recurrent neural network language model based online speech recognition system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*. IEEE, 2018, pp. 5904–5908.

[10] J. Jorge, A. Gimnez, J. Iranzo-Snchez, J. Civera, A. Sanchis, and A. Juan, "Real-Time One-Pass Decoder for Speech Recognition Using LSTM Language Models," in *Proc. Interspeech 2019*, 2019, pp. 3820–3824.

[11] E. Beck, W. Zhou, R. Schlüter, and H. Ney, "Lstm language models for lvcsr in first-pass decoding and lattice-rescoring," *arXiv preprint arXiv:1907.01030*, 2019.

[12] J. Jorge, A. Gimnez, J. Iranzo-Snchez, J. A. Silvestre-Cerd, J. Civera, A. Sanchis, and A. Juan, "Lstm-based one-pass decoder for low-latency streaming," in *Proc. of 45th Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2020)*, Barcelona (Spain), 2020.

[13] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, ser. JMLR Proceedings, vol. 9. JMLR.org, 2010, pp. 297–304.

[14] M. Gutmann and A. Hyvarinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *Journal of Machine Learning Research*, vol. 13, pp. 307–361, 2012.

[15] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention," in *Proc. Interspeech 2019*, 2019, pp. 231–235.

[16] G. Synnaeve, Q. Xu, J. Kahn, E. Grave, T. Likhomanenko, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert, "End-to-end asr: from supervised to semi-supervised learning with modern architectures," *arXiv preprint arXiv:1911.08460*, 2019.

[17] 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 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7829–7833.

[18] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020.

[19] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Transformer ASR with contextual block processing," in *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*. IEEE, 2019, pp. 427–433.

[20] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin, "Adaptive attention span in transformers," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 2019, pp. 331–335.

[21] X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, and D. Song, "A tensorized transformer for language modeling," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 2019, pp. 2229–2239.

[22] K. Irie, A. Gerstenberger, R. Schlüter, and H. Ney, "How much self-attention do we need trading attention for feed-forward layers," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6154–6158.

[23] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Training language models for long-span cross-sentence evaluation," in *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*. IEEE, 2019, pp. 419–426.

[24] S. Zagoruyko and N. Komodakis, "Wide residual networks," *Procedings of the British Machine Vision Conference 2016*, 2016.

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

[26] H. Zhang, Y. N. Dauphin, and T. Ma, "Fixup initialization: Residual learning without normalization via better initialization," in *International Conference on Learning Representations*, 2019.

[27] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[28] R. Schlüter, I. Bezrukov, H. Wagner, and H. Ney, "Gammatone features and feature combination for large vocabulary speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, April 2007, pp. IV–649–IV–652.

[29] M. Gibson and T. Hain, "Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition," in *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21, 2006*. ISCA, 2006.

[30] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," *arXiv preprint arXiv:1910.09799*, 2019.