INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Language Modeling with Deep Transformers

Kazuki Irie[1], *Albert Zeyer*[1,2], *Ralf Schlüter*[1], *Hermann Ney*[1,2]

[1]Human Language Technology and Pattern Recognition Group, Computer Science Department
RWTH Aachen University, 52074 Aachen, Germany
[2]AppTek GmbH, 52062 Aachen, Germany
{irie, zeyer, schlueter, ney}@cs.rwth-aachen.de

Abstract

We explore deep autoregressive Transformer models in language modeling for speech recognition. We focus on two aspects. First, we revisit Transformer model configurations specifically for language modeling. We show that well configured Transformer models outperform our baseline models based on the shallow stack of LSTM recurrent neural network layers. We carry out experiments on the open-source LibriSpeech 960hr task, for both 200K vocabulary word-level and 10K byte-pair encoding subword-level language modeling. We apply our word-level models to conventional hybrid speech recognition by lattice rescoring, and the subword-level models to attention based encoder-decoder models by shallow fusion. Second, we show that deep Transformer language models do not require positional encoding. The positional encoding is an essential augmentation for the self-attention mechanism which is invariant to sequence ordering. However, in autoregressive setup, as is the case for language modeling, the amount of information increases along the position dimension, which is a positional signal by its own. The analysis of attention weights shows that deep autoregressive self-attention models can automatically make use of such positional information. We find that removing the positional encoding even slightly improves the performance of these models.

Index Terms: language modeling, self-attention, Transformer, speech recognition

1. Introduction

Transformer models [1] have become popular in natural language processing. The Transformer architecture allows to successfully train a deep stack of *self-attention* layers [2–4] via residual connections [5] and layer normalization [6]. The positional encodings [1, 7] are used to provide the self-attention with the sequence order information. Across various applications, systematic improvements have been reported over the standard long short-term memory (LSTM) [8] recurrent neural network (RNN) based models. While originally designed as an encoder-decoder architecture in machine translation, the *encoder* (e.g., [9]) and the *decoder* (e.g., [10]) components are also separately used in corresponding problems depending on whether the problem disposes the whole sequence for prediction or not.

A number of recent works have also shown impressive performance in language modeling using the Transformer *decoder* component [10–15]. The earliest example can be found in [10] where such models are investigated for text generation. Recent works on training larger and deeper models [12, 14, 15] have shown further potential of the Transformer in language modeling. On the other hand, an obvious limitation of the Transformers is that their memory requirement linearly increases in terms of number of tokens in the sequence, which requires to work with a limited context window (basically a n-gram model where the typical number for n is 512) for tasks dealing with long sequences such as character-level language modeling [12]. Dai et al. [11] has introduced a segment-level recurrence and relative positional encoding in the Transformer language model (LM) to be able to potentially handle unlimited context.

In this work, we investigate deep autoregressive Transformers for language modeling in speech recognition. First, we revisit the parameter configurations of Transformers, originally engineered for the sequence-to-sequence problem [1], specifically for language modeling. We follow the spirit of the recent work [12, 14, 15] in investigating larger and deeper Transformers. We conduct experiments on the LibriSpeech automatic speech recognition (ASR) task [16] for both word-level conventional speech recognition and byte-pair encoding (BPE) [17] level end-to-end speech recognition [18, 19]. We apply our word-level models to hybrid speech recognition by *lattice rescoring* [20], and the BPE-level models to end-to-end models by *shallow fusion* [21, 22]. We show that deep Transformer LMs can be successfully applied to speech recognition; well configured Transformer LMs outperform our LSTM-RNN baselines in terms of both perplexity and word error rate (WER).

Second, we experimentally show that the positional encoding is not needed for Transformer LMs to give the best performance. Many previous works have investigated positional encoding variants to improve self-attention (e.g., [11, 23–25]). Previous works in Transformer LMs systematically use positional encoding, either jointly learned one or the sinusoidal one (both cases are reported to give similar performance in [12]). However, in the autoregressive problem where a new token is provided to the model at each time step, the amount of information the model has access to strictly increases from left to right at the lowest level of the network, which should provide some positional information by its own. Our analysis of the attention weights shows that Transformer LMs without positional encoding automatically make use of such information, and even give slight improvements over models with positional encodings.

2. Autoregressive Self-Attention

The language model we consider is based on the *decoder component* of the Transformer architecture [1]. Similar to previous work [1, 10–15], we define *layer* as a stack of two components: *self-attention* and *feed-forward* modules.

The autoregressive *self-attention module* in the l-th layer transforms the input $z_t^{(l-1)}$ at position t as follows:

$$\begin{aligned} x_t^{(l)} &= \text{LayerNorm}(z_t^{(l-1)}) \\ q_t^{(l)}, k_t^{(l)}, v_t^{(l)} &= Qx_t^{(l)}, Kx_t^{(l)}, Vx_t^{(l)} \\ h_t^{(l)} &= \big(h_{t-1}^{(l)}, (k_t^{(l)}, v_t^{(l)})\big) \\ y_t^{(l)} &= z_t^{(l-1)} + W_0\, \text{SelfAttention}(h_t^{(l)}, q_t^{(l)}) \end{aligned}$$

http://dx.doi.org/10.21437/Interspeech.2019-2225

where Q, K, V, respectively denote query, key, value projection matrices, $\mathrm{LayerNorm}$ denotes layer normalization [6], $\mathrm{SelfAttention}$ denotes the scaled multi-head dot product self-attention [1], and W_0 denotes the projection matrix for the residual connection [5, 26].

The output $y_t^{(l)}$ is then fed to the *feed-forward* module:

$$\begin{aligned} m_t^{(l)} &= \mathrm{LayerNorm}(y_t^{(l)}) \\ z_t^{(l)} &= y_t^{(l)} + W_2\,\mathrm{Activation}(W_1 m_t^{(l)}) \end{aligned}$$

where $\mathrm{Activation}$ is rectifier [27], Gaussian error linear unit (GELU) [15, 28], or gated linear unit (GLU) [29] in this work. The final model is build by stacking these *layers* multiple times.

The input of the network consists of the sum of the token embedding (word or BPE in this work) and the sinusoidal *positional encoding* as specified in [1]. The output softmax layer gives the probability distribution for the next token. As shown in the equations above, $h_t^{(l)}$ can be seen as *states* of the Transformer model[1] (whose size, as opposed to the RNN states, linearly grows along the position dimension). During inference, these states are stored to avoid redundant computation. During training, the computation along the position dimension is parallelized for speed-up.

3. LibriSpeech Dataset

The LibriSpeech dataset [16] for language modeling consists of 800M-word text only data and 960hr of audio transcriptions which corresponds to 10M-word text data. Based on analysis of count model perplexities, we observe that the audio transcription part does not contain special domain signal which matches the development set. Therefore, we simply merge the two datasets to form a single dataset for language model training. The average sentence length in the resulting training data is 21 words with the maximum length of 600 words. The development and test sets respectively have two parts [16]: dev-clean, dev-other, test-clean, and test-other. This separation is based on the audio-level characteristics, therefore it has no special meaning for language modeling. In the experimental section, we denote by "**Dev**" and "**Test**" the concatenation of *clean* and *other* parts of the respective data. Both datasets consist of about 110K running words with average of 20 words per sentence. The word-level vocabulary contains 200K words. We report all perplexities without making use of contexts beyond the sentence boundary.

We use the official LibriSpeech 4-gram count LM [16]. No improvement in perplexity is observed when going up to 5-grams. For LSTM-RNN LMs [30], we first train our base configuration; the model has 2 LSTM-RNN layers with 2048 nodes and the input projection layer of 128, where the dropout with a rate of 0.2 is applied between each layer. Since we observe that this model underfits the LibriSpeech training set, we remove the dropout and further increase the model size[2], which effectively gives better perplexities as shown in Table 1. We find that improvements from simply stacking layers saturate at 4 layers even without overfitting. Introducing a small linear bottleneck layer (size 512 here) before the output layer can make the models compact but with a loss in performance. The best model we obtain has 2 layers with 4096 nodes. Relative improvements greater than 58% are obtained by the LSTM over the 4-gram language model.

[1] In principle, we could also consider an autoregressive self-attention model which updates states at all predecessor positions for each new input, which would be then much more computationally inefficient.

[2] For training of larger models, we used the sampled softmax [31] with a frequency sorted log-uniform sampler and a sample size of 16384.

Table 1: *Perplexities of the 200K-word level **baseline** models.*

Model	Drop-out	Bottle-beck	Num. units	Num. layers	Params in M	Dev	Test
4-gram	-	-	-	-	230	146.2	151.8
LSTM	0.2	None	2048	2	487	71.3	74.8
	0.0					66.6	69.9
				3	520	64.0	67.2
				4	554	**61.9**	**64.9**
				5	587	62.7	65.9
				6	621	64.5	67.5
				8	688	67.2	70.3
			4096	2	1048	**60.2**	**63.2**
		512			334	63.1	66.3
			2048	4	248	64.5	67.7

4. Text based Experiments

We carry out experiments for both word-level and BPE-level language modeling. We first focus on the word-level one.

4.1. Hyper-parameters in Transformers

The Transformer architecture is a new search space Odyssey [32]. The exhaustive model hyper-parameters for Transformer language models specified by the equations in Sec. 2 are the input token embedding size, the number of layers, the dimension of the residual connection, and for each layer the number of attention heads, the dimension of the key and query, the dimension of the value, and the dimension of the feed-forward layer.

In our experiments, we use the same dimension for key, query and value, as well as the residual connection. We use the same dimensionality across all layers. Therefore, our models can be fully specified by the tuple (**number of layers** L, **feed-forward dimension** d_{ff}, **residual dimension** d_{res}, **number of heads** H). We do not apply any regularization method including dropout. We train all models using the plain stochastic gradient descent and Newbob learning rate tuning on a single GPU. We define our training sub-epoch (for Newbob) as the 10th of the full training data. All our implementations are based on the Tensorflow [33] based open-source toolkit RETURNN [34][3].

4.2. Hyper-parameter tuning

Given the amount of LibriSpeech training data (810M words), it is unreasonable to train all model variants until full convergence. The earlier stage of the training already consistently indicates the potential performance of the models. Therefore, we first carry out comparisons between models with different configuration at the equal, large enough, but reasonable number of updates.

The first set of comparison investigates the effect of depth and width. The perplexity results can be found in Table 2. All models in the table use 8 attention heads. Other parameters are specified in the table. The table is organized in three parts. The upper part of Table 2 shows the effect of number of layers; we observe that increasing number of layers (therefore the number of parameters) from 1 to 42 gradually improves the perplexity. In the middle part of Table 2, we also vary the feed-forward and residual dimensions. First of all, the 12-layer $(12, 4096, 512, 8)$ model outperforms the 6-layer $(6, 8192, 512, 8)$ model, while having similar number of parameters, which seems to indicate that the depth effectively benefits Transformer language models.

[3] Training configuration files and trained models are available at `https://github.com/rwth-i6/returnn-experiments/tree/master/2019-lm-transformers`.

We also train an extreme model which has only 2 layers with wide dimensions $(2, 8192, 2048, 8)$. The number of parameters in fact blows up because of the large value of d_{res} which results in a large matrix in the output softmax layer with 200K vocabulary[4]. We observe that such wide but shallow models do not perform well[5]. Finally, the lower part of Table 2 shows deeper models (up to 112 layers) with a smaller input dimension.

Table 2: *Perplexity **after 2.5 epoch** (25 sub-epochs in our setup; 6.5M updates). The number of heads H is 8 for all models below.*

Input emb.	L	d_{ff}	d_{res}	Params. in M	Perplexity Train	Perplexity Dev
512	1	2048	512	208	108.3	104.9
	6			224	75.7	74.3
	12			243	67.6	67.1
	24			281	62.2	62.3
	32			306	60.1	60.6
	42			338	**59.0**	**59.6**
512	2	8192	2048	536	73.1	73.8
	6		512	262	66.7	66.7
	12	4096		268	**63.5**	**63.8**
	4	16384		277	67.6	67.4
		32768		344	65.4	68.4
128	64	2048	512	330	56.3	57.6
	80			380	53.1	55.5
	96			431	51.9	54.9
	112			481	**51.5**	**54.5**

Table 3 shows the effect of number of attention heads. 16 heads which is the largest number we try in this setup give the best performance. In addition, we examine the type of activation function (Table 4). As opposed to previous work on feed-forward language models using GLUs [29, 36], we do not observe faster convergence. As we observe that the impact of choice of activation functions on the perplexity is overall limited, all our other models use the standard ReLU. As reported in the original Transformer, we confirm that both layer normalization and residual connections are needed for these models for stable training.

Finally, we train models with the best configurations for longer. Table 5 shows the perplexities which are better than those obtained by our LSTM based models (Table 1).

Table 3: *Effect of number of heads. Perplexity **after 2.5 epoch** for (12, 2048, 512, H).*

H	Params. in M	Perplexity Train	Perplexity Dev
1	243	71.9	70.8
4		69.1	68.6
8		67.6	67.1
16		**66.9**	**66.6**

Table 4: *Effect of activation functions. Perplexity **after 1 epoch** (10 sub-epochs in our setup) for (24, 2048, 512, 8).*

Activation	Perplexity Train	Perplexity Dev
ReLU [1, 27]	76.4	72.5
GLU [29]	76.5	72.8
GELU [15, 28]	**75.7**	**72.2**

Motivated by the success of Universal Transformers [37] for language modeling, we also experiment with parameter sharing across all layers. For such models to have comparable number of parameters with the standard Transformers, the dimensions in each layer must be increased, which results in slower training; here we simply investigate the effect of number of recurrence. We train (L, 8192, 1024, 16) models (329M params.) with shared parameters across all layers for three values of L: 3, 6 and 12, which give development perplexities of 79.9, 74.6, and 72.1, after 2.5 epochs. These perplexities are worse than those of the standard Transformers[6] (Table 2). However, increasing the number of layers from 3 to 12 consistently improves the perplexity. This improvement without additional parameters motivates future work in parameter sharing strategies for Transformers.

[4]As is the case for the baseline LSTM models in Table 1.

[5]We also train a (12, 2048, 512, 8) model with an additional projection layer with a large dimension of 2048 before the output layer (for a larger softmax bottleneck [35]; Table 1); no improvement was obtained.

Table 5: *Perplexities **after longer training**.*

Max. Epoch	Converged	L	d_{ff}	d_{res}	Params. in M	Perplexity Train	Perplexity Dev	Perplexity Test
5.5	Yes	12	4096	512	268	57.3	59.9	62.3
5		24	2048		281	55.6	58.0	60.7
		32			306	53.4	56.6	59.5
		42			338	51.2	**55.0**	**57.7**
3	No	80	2048	512	380	51.9	54.3	56.9
		96			431	50.9	**53.7**	**56.3**

5. ASR Experiments

5.1. Lattice rescoring results

We apply our word-level Transformer language models to conventional hybrid speech recognition by lattice rescoring. The standard push-forward lattice rescoring algorithm [20] for long-span language models can be directly applied to self-attention based models. The only modifications from the RNN version is to define the "state" as all hidden states ($h_t^{(l)}$ in Sec.2) in all layers from all predecessor positions and the current position (t; for position encoding). Table 6 shows the WERs and perplexities (PPL). Our baseline acoustic model is based on multi-layer bi-directional LSTM [38]. Further descriptions of our baseline acoustic model can be found in [39]. We obtain consistent improvements in terms of WER over the LSTM baselines.

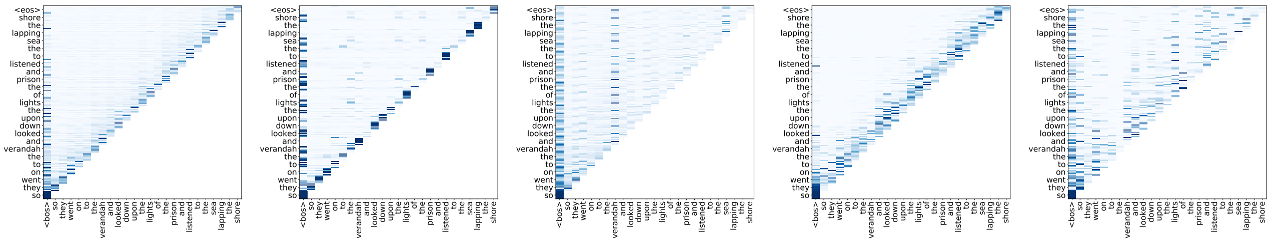
Table 6: *WERs (%) for **hybrid** systems on the LibriSpeech **960hr**. 4-gram model is used in the first pass to generates lattices for rescoring. The row "Lattice" shows oracle WERs of the lattices.*

LM	L	Para. in M	dev clean PPL	dev clean WER	dev other PPL	dev other WER	test clean PPL	test clean WER	test other PPL	test other WER
4-gram	-	230	151.7	3.4	140.6	8.3	158.1	3.8	145.7	8.8
Lattice	-	-	-	1.0	-	2.3	-	1.3	-	2.6
LSTM	2	1048	60.2	2.3	60.2	5.4	64.8	2.6	61.7	5.9
Trans-former	24	281	57.8	2.2	58.3	**5.2**	62.2	**2.5**	59.4	5.7
	42	338	54.5	**2.1**	55.5	**5.2**	59.1	**2.5**	56.4	5.7
	96	431	**53.2**	**2.1**	**54.2**	**5.2**	**57.6**	**2.5**	**55.0**	**5.6**

5.2. End-to-end ASR shallow fusion results

We train 10K BPE-level Transformer language models to be combined with an attention-based encoder-decoder speech model by shallow fusion [21, 22]. The 10K BPE level training data has a longer average length of 24 tokens per sentence with the longest sentence length of 1343, which is still manageable without any truncation for self-attention. We use the Transformer architecture of (24, 4096, 1024, 8). The LSTM model has 4 layers with 2048 nodes. We refer to our previous work [19] for the description of the baseline attention model; the baseline WERs better than

[6]The comparison is not straightforward though. The training hyperparameters tuned for the Transformers could not be applied to Universal Transformers; we found it crucial to reduce the gradient norm clipping threshold from 1 to 0.1, which can slow down the convergence.



(a) *First layer with PE* (b) *First layer without PE* (c) *"Blur" layer* (d) *"Window" layer* (e) *"Structured" layer*

Figure 1: *Layer categories in word-level 24-layer Transformer language models. The x-axis corresponds to the input words. The y-axis shows the target words; each target word position has 8 sub-rows corresponding to 8 heads. "PE" denotes positional encoding.*

our previous work [19] are obtained by improved curriculum learning and longer training. Table 7 shows both perplexities and WERs. Following [40], we introduce an end-of-sentence penalty in shallow fusion to benefit from a large beam size of 64. Again, we obtain consistent improvements over the LSTM baseline. These results are better than previously reported WERs [40–42] for end-to-end models without data augmentation [43].

Table 7: *WERs (%) for* ***attention-based models*** *on LibriSpeech* ***960hr*** *dataset. Perplexities are on the 10K BPE level.*

LM	Beam	dev				test			
		clean		other		clean		other	
		PPL	WER	PPL	WER	PPL	WER	PPL	WER
None	12	-	4.3	-	12.9	-	4.4	-	13.5
LSTM	64	43.7	2.9	46.4	8.9	47.1	3.2	47.2	9.9
Transfo.	64	**35.9**	**2.6**	**38.9**	**8.4**	**38.8**	**2.8**	**39.0**	**9.3**

6. Analysis

We explore attention weights for analysis, with a particular focus on comparing models with and without positional encoding.

6.1. Transformer LMs without positional encoding

In the autoregressive problem where a new token is provided to the model at each time step, the amount of information the model has access to strictly increases from left to right at the lowest level of the network; the deeper layers should be able to recognize this structure which should provide the model with some positional information by its own. To check this hypothesis, we train models without any positional encoding. First, we observe that they give better perplexities than the models with sinusoidal positional encoding (Table 8).

Table 8: *Effect of sinusoidal positional encoding. Perplexity* ***after 5 epochs*** *(13M updates) for (L, 2048, 512, 8) models.*

L	Position. encoding	Params. in M.	Perplexity Train	Dev	Test
12	Sinusoidal	243	61.8	63.1	66.1
	None		58.0	**60.5**	**63.4**
24	Sinusoidal	281	55.6	58.0	60.8
	None		52.7	**56.6**	**59.2**
42	Sinusoidal	338	51.2	55.0	57.7
	None		50.5	**54.2**	**56.8**

6.2. Layer categories

The attention in the first layer is the most straightforward for interpretation because the feature at each position exactly corresponds to the word at the position (while deeper layers can potentially shuffle the feature content). The attention weights in the first layer of 24-layer Transformer language models with and without positional encodings are visualized in Figure 1. We observe that the first layer of the model with positional encoding (Figure 1(a)) learns to create *n-gram features* (roughly 2 or 3-gram), which indicates that the positional information is directly used. In contrast, the first layer of the model without positional encoding learns to focus on the new input token as can be seen as the diagonal in Figure 1(b) (interestingly, we also see that it ignores some functional words such as "the", "and", "to" which might be modeled by some off-set values, therefore attending to the beginning of sentence token instead), which demonstrates that the model is aware of the position of the new input.

We observe that the behavior of other layers are rather similar for both Transformer models with and without positional encoding. We find 3 categories in the other 23 layers; the second and third layers are *"blur"* layers as shown in Figure 1(c), which seems to roughly average over all positions (while we can also see that some heads focus on *difficult* words, here "verandah"). Layer 4 to 9 are *"window"* layers which focus on the local n-gram. A representative example is show in Figure 1(d). Finally, we find the top layers 10 to 24 to be more *"structured"*, attending to some specific patterns; an example is shown in Figure 1(e).

7. Conclusion

We apply deep Transformer language models for speech recognition. We show that such models outperform the shallow stack of LSTM-RNNs on both word-level and BPE-level modeling. Future work investigates application of crucial components of deep Transformers (such as layer normalization) to deeper LSTM models; e.g., the RNMT+ *decoder* architecture [44] for language modeling. Furthermore, we do not apply any regularization on models for the LibriSpeech task, as no overfitting is observed in the range of model sizes we experimented with (for the word-level models). We can possibly still improve our models simply by scaling up their size and using regularization.

8. Acknowledgements

This work has received funding from the European Research Council 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 thanks Liuhui Deng for contributing to our lattice rescoring code, Arne Nix and Julian Schamper for sharing their base Transformer configs, as well as Eugen Beck, Christoph Lüscher and Wei Zhou for help with generating lattices. Experiments were partially performed with computing resources granted by RWTH Aachen University under project nova0003.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[2] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Proc. EMNLP*, Austin, TX, USA, Nov. 2016, pp. 551–561.

[3] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *Int. Conf. on Learning Representations (ICLR)*, Apr. 2017.

[4] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proc. EMNLP*, Austin, TX, USA, Nov. 2016, pp. 2249–2255.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Patt. Recog. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

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

[7] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. ICML*, Sydney, Australia, Aug. 2017, pp. 1243–1252.

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

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, Minneapolis, USA, Jun. 2019.

[10] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, Ł. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," in *ICLR*, Vancouver, Canada, Apr. 2018.

[11] Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *ACL*, Florence, Italy, Jul. 2019.

[12] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *Proc. AAAI Conf. on Artif. Int.*, Honolulu, HI, USA, Jan. 2019.

[13] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," in *ICLR*, New Orleans, LA, USA, May 2019.

[14] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," [Online]. : https://blog.openai.com/language-unsupervised/, 2018.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," [Online]. : https://blog.openai.com/better-language-models/, 2019.

[16] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *ICASSP*, South Brisbane, Queensland, Australia, Apr. 2015, pp. 5206–5210.

[17] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. ACL*, Berlin, Germany, August 2016, pp. 1715–1725.

[18] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: a neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, Shanghai, China, Mar. 2016, pp. 4960–4964.

[19] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," in *Proc. Interspeech*, Hyderabad, India, Sep. 2018, pp. 7–11.

[20] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Interspeech*, Singapore, Sep. 2014, pp. 661–665.

[21] Ç. Gülçehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *Computer Speech & Language*, vol. 45, pp. 137–148, Sep. 2017.

[22] 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 *Proc. SLT*, Athens, Greece, Dec. 2018.

[23] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. NAACL*, New Orleans, LA, USA, Jun. 2018, pp. 464–468.

[24] M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel, "Self-attentional acoustic models," in *Proc. Interspeech*, Hyderabad, India, Sep. 2018, pp. 3723–3727.

[25] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," in *Proc. ICASSP*, Brighton, UK, May 2019, pp. 7115–7119.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. European Conf. on Computer Vision (ECCV)*, Amsterdam, Netherlands, Oct. 2016, pp. 630–645.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. on Machine Learning (ICML)*, Haifa, Israel, Jun. 2010, pp. 807–814.

[28] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," arXiv preprint arXiv:1606.08415, 2018.

[29] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. ICML*, Sydney, Australia, Aug. 2017, pp. 933–941.

[30] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling." in *Proc. Interspeech*, Portland, OR, USA, Sep. 2012, pp. 194–197.

[31] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," in *Proc. ACL*, Beijing, China, Jul. 2015, pp. 1–10.

[32] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, 2017.

[33] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. USENIX Sympo. on Operating Sys. Design and Impl. (OSDI 16)*, Savannah, GA, USA, Nov. 2016, pp. 265–283.

[34] A. Zeyer, T. Alkhouli, and H. Ney, "RETURNN as a generic flexible neural toolkit with application to translation and speech recognition," in *Proc. ACL*, Melbourne, Australia, Jul. 2018.

[35] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, "Breaking the softmax bottleneck: A high-rank RNN language model," in *ICLR*, Vancouver, Canada, Apr. 2018.

[36] K. Irie, Z. Lei, R. Schlüter, and H. Ney, "Prediction of LSTM-RNN full context states as a subtask for N-gram feedforward language models," in *ICASSP*, Calgary, Canada, Apr. 2018, pp. 6104–6108.

[37] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal Transformers," in *Int. Conf. on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[38] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "A comprehensive study of deep bidirectional lstm rnns for acoustic modeling in speech recognition," in *Proc. ICASSP*, New Orleans, LA, USA, Mar. 2017, pp. 2462–2466.

[39] 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 *Interspeech*, Graz, Austria, Sep. 2019.

[40] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," *arXiv preprint arXiv:1904.02619*, 2019.

[41] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully convolutional speech recognition," *arXiv preprint arXiv:1812.06864*, 2018.

[42] K. Irie, R. Prabhavalkar, A. Kannan, A. Bruguier, D. Rybach, and P. Nguyen, "Model unit exploration for sequence-to-sequence speech recognition," *preprint arXiv:1902.01955*, 2019.

[43] 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," *preprint arXiv:1904.08779*, 2019.

[44] M. X. Chen *et al.*, "The best of both worlds: Combining recent advances in neural machine translation," in *Proc. ACL*, Melbourne, Australia, Jul. 2018, pp. 76–86.