HOW MUCH SELF-ATTENTION DO WE NEED? TRADING ATTENTION FOR FEED-FORWARD LAYERS

Kazuki Irie[1,*], *Alexander Gerstenberger*[1], *Ralf Schlüter*[1,2], *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, schlueter, ney}@cs.rwth-aachen.de, alexander.gerstenberger@rwth-aachen.de

ABSTRACT

We propose simple architectural modifications in the standard Transformer with the goal to reduce its total state size (defined as the number of self-attention layers times the sum of the key and value dimensions, times position) without loss of performance. Large scale Transformer language models have been empirically proved to give very good performance. However, scaling up results in a model that needs to store large states at evaluation time. This can increase the memory requirement dramatically for search e.g., in speech recognition (first pass decoding, lattice rescoring, or shallow fusion). In order to efficiently increase the model capacity without increasing the state size, we replace the single-layer feed-forward module in the Transformer layer by a deeper network, and decrease the total number of layers. In addition, we also evaluate the effect of key-value tying which directly divides the state size in half. On TED-LIUM 2, we obtain a model of state size 4 times smaller than the standard Transformer, with only 2% relative loss in terms of perplexity, which makes the deployment of Transformer language models more convenient.

Index Terms— language modeling, speech recognition, self-attention, Transformer, LSTM

1. INTRODUCTION

Large scale Transformers [1] have become very popular for different language modeling related tasks [2, 3] for which a large amount of training data is available. While masked-language models [3] are typically used as a pre-trained model to be fine-tuned for the specific downstream natural language processing tasks, the standard (auto-regressive) Transformer language models [2,4–7] can directly be applied for automatic speech recognition (ASR) [8, 9].

However, memory requirements of such large and deep Transformer based ASR language models at evaluation time become very demanding because each self-attention sub-layer in the model stores key and value vectors for all predecessor positions. This is a practical issue, since search algorithms (including lattice rescoring [10] in hybrid HMM-neural network ASR [11] or shallow fusion [12] in end-to-end speech recognition) typically store these large *states* for a large number of hypotheses. Interestingly, the only hyper-parameter in the original Transformer which can increase the parameter count (therefore potentially the model capacity) without affecting the state size is the feed-forward inner dimension.

A natural question which arises out of this observation is whether we can put more parameters in the feed-forward module more efficiently. We investigate the following modifications with the goal of achieving a smaller state but still powerful Transformer: First, we introduce an extra hyper-parameter to specify the number of feed-forward sub-layers in each Transformer layer; thus, replace the feed-forward module by a deep neural network (DNN) with residual connections, which allows us to increase the model capacity efficiently, independent of the state size. Second, we also explore sharing key and value projection matrices which would allow a Transformer to only store key vectors as its states. We present our main results on the TED-LIUM release 2 (200h) dataset [13]. Our experiments have been conducted using the TensorFlow [14] based open-source toolkit RETURNN [15][1].

2. RELATED WORK

Existing works have focused on modifying Transformers to either reduce the model size [16] or the computational complexity [17]. We are interested in reducing the *state size* of Transformers. Notable previous works have proposed models with a limited state size, which make use of some segment level recurrence such as Transformer-XL [7] or Compressive Transformer [18]. Our method can be potentially combined with these techniques. Some quantization (e.g. [19] for model compression) can be an alternative solution for reducing the state size; in this work, we tackle the problem from the modeling perspective. Sharing query and key matrices has been investigated in [17]. However, that does not help in reducing the state size. [20] replaces some feed-forward layers by

*Work conducted while the author was at RWTH Aachen. Now with the Swiss AI Lab, IDSIA, USI & SUPSI, 6928 Manno-Lugano, Switzerland.

[1]Example config files and models are available in `https://github.com/rwth-i6/returnn-experiments/tree/master/2020-lm-small-state-trafo`.

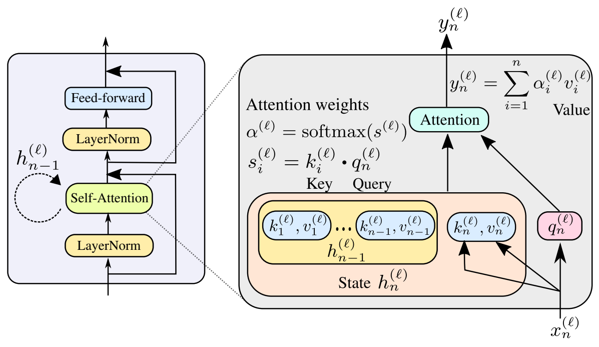
Feed-forward
LayerNorm
$h_{n-1}^{(\ell)}$
Self-Attention
LayerNorm
$y_n^{(\ell)}$
$y_n^{(\ell)} = \sum_{i=1}^{n} \alpha_i^{(\ell)} v_i^{(\ell)}$
Value
Attention weights
$\alpha^{(\ell)} = \mathrm{softmax}(s^{(\ell)})$
$s_i^{(\ell)} = k_i^{(\ell)} \bullet q_n^{(\ell)}$
Key
Query
Attention
$k_1^{(\ell)}, v_1^{(\ell)}$
$k_{n-1}^{(\ell)}, v_{n-1}^{(\ell)}$
$h_{n-1}^{(\ell)}$
$k_n^{(\ell)}, v_n^{(\ell)}$
$q_n^{(\ell)}$
State $h_n^{(\ell)}$
$x_n^{(\ell)}$

Fig. 1. Layer ℓ in the standard Transformer language model.

a more powerful but efficient product-key memory layer, and they also effectively managed to reduce the number of self-attention layers; in principle, our work also follows a similar spirit since we also replace the feed-forward sub-layer by a more powerful DNN. Other works [21, 22] replace the Transformer decoder by an LSTM in the encoder decoder translation model; in this work, we include knowledge distillation [23, 24] from a Transformer to an LSTM model [25].

3. SMALL STATE TRANSFORMERS

3.1. States in standard Transformer language model

A deep Transformer model consists of L *layers*. Each *layer* is defined as a stack of *self-attention* and *feed-forward* modules, as depicted in Figure 1. The *self-attention module* in the l-th layer transforms the input $z_n^{(l-1)}$ at position n as follows:

$$\begin{aligned} x_n^{(l)} &= \mathrm{LayerNorm}(z_n^{(l-1)}) && (1)\\ q_n^{(l)}, k_n^{(l)}, v_n^{(l)} &= Qx_n^{(l)}, Kx_n^{(l)}, Vx_n^{(l)} && (2)\\ h_n^{(l)} &= \left(h_{n-1}^{(l)}, (k_n^{(l)}, v_n^{(l)})\right) && (3)\\ y_n^{(l)} &= \mathrm{Attention}(h_n^{(l)}, q_n^{(l)}) && (4)\\ \tilde{y}_n^{(l)} &= z_n^{(l-1)} + W_0 y_n^{(l)} && (5)\end{aligned}$$

where Q, K, V, respectively denote query, key, value projection matrices, $\mathrm{LayerNorm}$ denotes layer normalization [26], $\mathrm{Attention}$ denotes the scaled multi-head dot product self-attention [1], and W_0 denotes the projection matrix for the residual connection [27]. In our experiments, we use the same dimension for key/query and value dimensions as well as for the residual connection, which we denote as d_{kv}.

As shown in the Eq. (3), each self-attention module stores the **state vector** $h_n^{(l)}$. Its total size is $2 \times n \times L \times d_{\mathrm{kv}}$ which not only grows with the position n but when we make the model deeper (larger L) or wider via self-attention dimensions (d_{kv}).

$\tilde{y}_n^{(l)}$ is then fed to the one-layer *feed-forward* module:

$$\begin{aligned} m_n^{(l)} &= \mathrm{LayerNorm}(\tilde{y}_n^{(l)}) && (6)\\ z_n^{(l)} &= \tilde{y}_n^{(l)} + W_2\,\mathrm{ReLU}(W_1 m_n^{(l)}) && (7)\end{aligned}$$

The outer dimension of W_1 is typically called the feed-forward inner dimension d_{ff}.

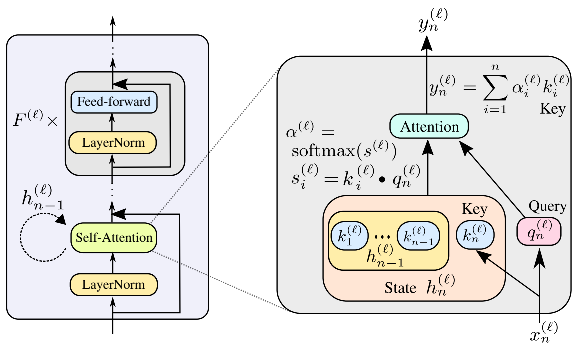

Fig. 2. Layer ℓ in a modified Transformer language model with self-attention-DNN and shared-KV.

3.2. Modifying Transformer layers for smaller state size

For deep Transformer language models, the size of state vectors $(h_n^{(1)}, .., h_n^{(L)})$ can be potentially very large, which is inconvenient, especially for ASR applications, for search where such states must be stored for different hypotheses.

At the same time, we need to provide the model with a large number of parameters for a good performance. The only model hyper-parameter in the original Transformer which can increase the model parameter counts, but does not affect the state size is the feed-forward inner dimension d_{ff}. In order to isolate an increase in the model size from the total state size in a Transformer, we make the feed-forward component in each Transformer layer deeper: using $F^{(\ell)}$ layers for the layer ℓ as indicated in Figure 2 (in experiments we use the same number F for all layers). We carry out experiments with modified Transformers (Figure 2) which:

- Define the Transformer layer as one self-attention sub-layer plus F feed-forward sub-layers (*self-attention-DNN*). Each sub-layer uses the layer normalization and the residual connection as in Eqs. (6, 7).
- Share key and value weight matrices K and V (*shared-KV*), and only store the key vectors as the state:

$$\begin{aligned} q_n^{(l)}, k_n^{(l)} &= Qx_n^{(l)}, Kx_n^{(l)} && (8)\\ h_n^{(l)} &= \left(h_{n-1}^{(l)}, k_n^{(l)}\right) && (9)\end{aligned}$$

With this model, we increase the number of feed-forward sub-layers F while reducing the number of Transformer layers L, as long as it preserves the model performance. Sharing K and V is an extra option for further reduction of state size. We evaluate this model for language modeling in ASR.

4. EXPERIMENTAL SETUPS

4.1. Datasets

Our main experiments are carried out on the TED-LIUM release 2 (200h) dataset [13], on the word level (152K vocabulary). Some extra experiments are also presented on the large LibriSpeech dataset [28]; we refer to [9] for the basic setups. In our previous work [9, 29], we observed that the performance gap between LSTM and Transformer is especially pronounced when the training data is large and/or sequences are long on average[2]. TED-LIUM is a medium-size publicly available dataset (270M running words) and sentences are relatively long (20 words on average for the dev and eval sets).

4.2. Baseline n-gram count language models

The language model training data provided by TED-LIUM release 2 consists of **7 subsets** including the TED-LIUM 2 audio transcriptions [13]. We first train n-gram modified Kneser-Ney language models [30, 31] on each subset of the training data with the discount parameters optimized on the dev set [32]. We linearly interpolate these sub-LMs using the interpolation weights optimized for the dev perplexity; we include a background n-gram model as the 8th component in interpolation, which is trained on all training texts (which gave 5% rel. improvements on the 4-gram before pruning).

The upper block of Table 1 shows perplexities for the count models. First of all, we observe large improvements in development perplexity by increasing the order from 4 to 6 (in contrast to what we typically observe e.g., on LibriSpeech [28]): this is in fact due to some **overlap** between the *common crawl* training subset (16 M words) and the development text[3]. However, once we apply pruning[4] to obtain a reasonably sized LM for the first pass decoding, the improvements from these higher-order n-grams disappear; almost no improvement is obtained by going beyond 4-gram (as is typically the case for a clean dataset without overlap).

4.3. Baseline LSTM and Transformer language models

In order to exploit the mutli-corpus TED-LIUM training data, we train both LSTM and Transformer language models in two steps. We first pre-train the model on the whole training data until convergence. Then we fine-tune the model on the TED-LIUM 2 *transcriptions* (2 M words) and *common crawl* (16 M words) sub-sets which are the top-2 sets with the highest weights for 6-gram interpolation[5].

The perplexities of the LSTM and standard Transformer models are presented in the lower part of Table 1. The input word embedding dimension is 128 for all models. The LSTM model has 4 layers with 2048 nodes, as in [9], except that we apply 20% dropout. For Transformers, the number of attention heads H is always set to 12 and d_{kv} is set to 768, and we apply 10% dropout, unless otherwise specified. The model in Table 1 has 32 layers with $d_{\text{ff}} = 4096$. No positional encoding is used [9]. More than 15% relative improvement in perplexity is obtained by the Transformer over the LSTM baseline.

Table 1. ***Perplexity*** *of the word-level (152K vocab)* ***baseline*** *models on* ***TED-LIUM 2***.

Model	# Param. [M]	Perplexity Dev	Perplexity Test
4-gram	343	105.4	124.7
+ pruning	161	113.2	127.9
5-gram	663	92.3	123.2
+ pruning	169	112.4	127.8
6-gram	1021	86.2	121.3
+ pruning	183	116.2	125.9
LSTM	450	73.5	71.3
Transformer	414	**62.0**	**60.7**

5. EXPERIMENTAL RESULTS

5.1. Effect of DNN inside Transformer layer

We introduce an extra hyper-parameter F to specify the number of feed-forward layers in each Transformer layer. Table 2 shows the perplexity results for TED-LIUM 2. The best numbers in the first block are copied from Table 1 as the baseline 32-layer standard Transformer model. The *8L-3F* model which contains only 8 layers with 3 feed-forward sub-layers per layer (thus, only 8 self-attention and 24 feed-forward sublayers) achieves comparable performance with the 32-layer models; with a degradation in perplexity of about 2% relative, the state size is reduced by a factor of 4.

Table 2. ***Perplexity*** *of the word-level (152K vocab) models on* ***TED-LIUM 2***. $d_{\text{kv}} = 768$ *and* $H = 12$ *for all models. The models with* $F = 1$ *are standard Transformers.*

L	F	d_{ff}	State size / position	#Param. [M]	Perplexity Dev	Perplexity Test
8	1	4096	12,288	206	67.9	64.9
32	1	2048	49,152	313	63.3	61.5
32	1	4096	49,152	414	**62.0**	**60.7**
3	15	2048	4,608	247	69.3	66.0
6	7	2048	9,216	280	64.5	62.6
8	3	4096	12,288	338	63.4	61.7
12	3	4096	18,432	379	62.2	61.0
16	3	4096	24,576	464	**61.4**	**60.7**

We also carry out similar experiments on LibriSpeech. Table 3 presents the perplexities. As an additional engineering improvement from [9], we make use of two speed-up methods for training; the noise contrastive estimation [34, 35] and an improved batch construction: Instead of fully randomizing sentences, we first sort them by the length, create bins (each bin containing as many sentences as the batch size; here 32), and shuffle the bins. We found that using both techniques can result in up to four times speedup in training, with a marginal

[2]In [29] we observed that the Transformer models are only slightly better than LSTM models on Switchboard (small data/short sequences) whereas a large performance gap was observed for Quaero (small data/long sequences) and for LibriSpeech (large data/long sequences) [9].

[3]To the best of our knowledge, this problem has not been reported in previous work on TED-LIUM. In any case, the overall effect seems to be marginal after pruning (shown by perplexities for different orders n), and this problem does not affect the evaluation set.

[4]We note that to avoid the well known negative effect of entropy pruning on Kneser-Ney LMs [33], we trained separate Katz ($n - 1$)-gram LMs [31] to help the pruning process. However, at this pruning ratio (at most factor of 6), the benefit of such extra care was marginal.

[5]We made this selection before getting aware of the overlap problem. The interpolation weights for 6-gram LMs were: 22% for the *common crawl*, 16% for the TED-LIUM 2 *transcriptions* and 60% for the background model, while the weight next in size was only 0.9%.

loss of performance. The *6L-7F model* (6 self-attention and 42 feed-forward sub-layers) is trained using these speed-up tricks. This model has a similar number of parameters as the 32-layer standard Transformer model taken from our previous work [9], and it gives similar perplexities with a much smaller state size[6].

Table 3. ***Perplexity*** *of the word-level (200K vocab) model on* ***LibriSpeech****.* d_{kv} *is* ***512*** *for all models.*

L	F	d_{ff}	H	NCE Train	State size / position	#Param. [M]	Perplexity	
							Dev	Test
32 [9]	1	2048	8	No	32,768	306	56.6	59.5
42 [9]					43,008	338	54.2	56.8
6	7	4096	16	No	6,144	307	55.5	58.1
				Yes			56.8	59.4

5.2. Effect of shared-KV

Sharing K and V matrices reduces the state size by a factor of 2. We evaluate KV-sharing in both the standard Transformer and the one with the proposed self-attention-DNN. Table 4 shows that the degradation is marginal for the baseline Transformer $32L$-$1F$, while a degradation of about 5% relative in perplexity is observed for the $8L$-$3F$ model. The $6L$-$7F$ model without shared-KV from Table 2 outperforms the $8L$-$3F$ model with shared-KV while having less parameters.

Table 4. *Effect of sharing KV for both standard and small state Transformers. Word level* ***perplexity*** *on* ***TED-LIUM 2****.*

Shared-KV	L	F	State size / position	#Param. [M]	Perplexity	
					Dev	Test
No	32	1	49,152	414	**62.0**	**60.7**
Yes			24,576	395	62.7	61.2
No	8	3	12,288	338	**63.4**	**61.7**
Yes			6,144	333	66.3	63.9

5.3. Knowledge distillation

Finally, an alternative approach for obtaining a small state model is to train an LSTM model via knowledge distillation. We use the baseline 32-layer Transformer model as a teacher[7]. Table 5 shows the perplexities. We obtain good improvements over the baseline LSTM (Table 1), while not matching the performance of the large state Transformer teacher model.

Table 5. *Results of* ***knowledge distillation****.* ***Perplexities*** *for the word-level (152K vocab)* ***TED-LIUM 2****.*

Model		State size for n tokens	#Param. [M]	Perplexity	
				Dev	Test
Teacher	Transformer	$n\times$49,152	414	**62.0**	**60.7**
Student	LSTM	16,384	450	66.1	63.0

[6]We note that this 32-layer model from [9] makes use of positional encoding (which is in favor for the new $6L$-$7F$ model); but $6L$-$7F$ is trained using the speed up tricks which instead is in favor of 32-layer model.

[7]In [36], we have shown that it is difficult to transfer performance from a full-context LSTM LM to a short context n-gram feed-forward LM. Here the question is whether we can transfer from a *growing-state* Transformer to a *fixed size state* LSTM RNN.

5.4. Lattice Rescoring ASR experiments

We finally show lattice rescoring ASR experiments for TED-LIUM 2. We refer the readers to the dedicated system paper [37] for details about the baseline hybrid HMM/NN system. Table 6 shows the WERs. The proposed small state $8F$-$3L$ Transformer model (without KV-sharing) gives comparable performance to the standard deep Transformers, with 4 times smaller memory requirement: concretely, the highest requirement for one lattice is reduced from 64 GB to 16 GB.

Table 6. *WERs on* ***TED-LIUM 2****. Perplexities are after* ***interpolation*** *with the 4-gram LM. Lattices are generated by either 4-gram or 4-gram + LSTM LMs in the* ***first pass****.*

Model	L	F	Dev		Eval	
			PPL	WER	PPL	WER
4-gram [37]	-	-	113.2	6.8	127.9	7.3
+ LSTM	-	-	64.4	5.5	69.2	6.0
+ Transformer	32	1	55.3	**5.3**	60.1	5.9
	8	3	56.6	**5.3**	61.1	5.9
	16		**54.9**	**5.3**	**59.8**	**5.8**
4-gram + LSTM [37]	-	-	64.4	5.5	69.2	6.1
+ Transformer	32	1	54.8	**5.1**	59.3	**5.6**
	8	3	56.0	5.2	60.1	5.7
	16		**54.4**	5.3	**58.9**	5.7

6. CONCLUSION

We have shown that the one-to-one ratio between the self-attention and feed-forward sub-layers in the Transformer is sub-optimal for the state size. By increasing the capacity of each feed-forward module, we managed to reduce the number of self-attention layers to a relatively small number such as 6 or 8, with a marginal loss of performance; these small state Transformer language models directly reduced the memory requirement for the ASR application. Further sharing key and value matrices could halve the state size at the cost of 5% rel. degradation in perplexity. In this work, we defined each layer in the model to be of the same type. In future work, based on the visualization we carried out in [9], we could potentially replace the self-attention in the bottom/mid layers by simple weighted bag-of-words layers [38, 39] which can further reduce the state size of the Transformer language model.

7. ACKNOWLEDGEMENT

We thank Albert Zeyer and Pavel Golik for suggestions on Figure 1 and the title, resp. This work has received funding from the European Research Council under the European Unions 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. Experiments were partially performed with computing resources granted by RWTH under project nova0003.

8. REFERENCES

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. NIPS*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, "Language models are unsupervised multitask learners," https://blog.openai.com/better-language-models/, 2019.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, June 2019, pp. 4171–4186.

[4] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer, "Generating wikipedia by summarizing long sequences," in *ICLR*, Vancouver, Canada, Apr. 2018.

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

[6] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones, "Character-level language modeling with deeper self-attention," in *Proc. AAAI*, Honolulu, HI, USA, Jan. 2019.

[7] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. ACL*, Florence, Italy, July 2019.

[8] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde, "Jasper: An End-to-End Convolutional Neural Acoustic Model," in *Proc. Interspeech*, 2019, pp. 71–75.

[9] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "Language modeling with deep Transformers," in *Proc. Interspeech*, Graz, Austria, Sept. 2019, pp. 3905–3909.

[10] Martin Sundermeyer, Zoltán Tüske, Ralf Schlüter, and Hermann Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Interspeech*, Singapore, Sept. 2014, pp. 661–665.

[11] Hervé Bourlard and Nelson Morgan, *Connectionist speech recognition: a hybrid approach*, vol. 247, Springer, 1994.

[12] Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "On using monolingual corpora in neural machine translation," *Computer Speech & Language*, vol. 45, pp. 137–148, Sept. 2017.

[13] Anthony Rousseau, Paul Deléglise, and Yannick Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *LREC*, 2014, pp. 3935–3939.

[14] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, and Matthieu Devin et al., "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, Savannah, GA, USA, Nov. 2016, pp. 265–283.

[15] Albert Zeyer, Tamer Alkhouli, and Hermann Ney, "RETURNN as a generic flexible neural toolkit with application to translation and speech recognition," in *Proc. ACL*, Melbourne, Australia, July 2018.

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut, "ALBERT: A lite bert for self-supervised learning of language representations," in *ICLR*, Addis Ababa, Ethiopia, Apr. 2020.

[17] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya, "Reformer: The efficient transformer," in *ICLR*, Addis Ababa, Ethiopia, Apr. 2020.

[18] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap, "Compressive transformers for long-range sequence modelling," in *ICLR*, Addis Ababa, Ethiopia, Apr. 2020.

[19] Shankar Kumar, Michael Nirschl, Daniel Holtmann-Rice, Hank Liao, Ananda Theertha Suresh, and Felix Yu, "Lattice rescoring strategies for long short-term memory language models in speech recognition," in *Proc. ASRU*, Okinawa, Japan, Dec. 2017.

[20] Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou, "Large memory layers with product keys," in *Proc. NeurIPS*, Vancouver, Canada, Dec. 2019.

[21] Tobias Domhan, "How much attention do you need? a granular analysis of neural machine translation architectures," in *Proc. ACL*, Melbourne, Australia, July 2018, pp. 1799–1808.

[22] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes, "The best of both worlds: Combining recent advances in neural machine translation," in *Proc. ACL*, Melbourne, Australia, July 2018, pp. 76–86.

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, Montreal, Canada, Dec. 2014.

[24] Jimmy Ba and Rich Caruana, "Do deep nets really need to be deep?," in *Proc. NIPS*, Quebec, Canada, Dec. 2014, vol. 27, pp. 2654–2662.

[25] Alexander Gerstenberger, Kazuki Irie, Pavel Golik, and Hermann Ney, "Domain robust, fast, and compact neural language models for automatic speech recognition," in *ICASSP*, Barcelona, Spain, May 2020.

[26] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, "Layer normalization," *Preprint arXiv:1607.06450*, 2016.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Identity mappings in deep residual networks," in *Proc. ECCV*, Amsterdam, Netherlands, Oct. 2016, pp. 630–645.

[28] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*, South Brisbane, Queensland, Australia, Apr. 2015, pp. 5206–5210.

[29] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "Training language models for long-span cross-sentence evaluation," in *Proc. ASRU*, Sentosa, Singapore, Dec. 2019, pp. 419–426.

[30] Reinhard Kneser and Hermann Ney, "Improved backing-off for m-gram language modeling," in *Proc. ICASSP*, Detroit, MI, USA, May 1995, pp. 181–184.

[31] Stanley F Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech & Language*, vol. 13, no. 4, pp. 359–393, 1999.

[32] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "On the estimation of discount parameters for language model smoothing," in *Proc. Interspeech*, Florence, Italy, Aug. 2011, pp. 1433–1436.

[33] Ciprian Chelba, Thorsten Brants, Will Neveitt, and Peng Xu, "Study on interaction between entropy pruning and kneser-ney smoothing," in *Proc. Interspeech*, Makuhari, Japan, Sept. 2010, pp. 2422–2425.

[34] Michael Gutmann and Aapo Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proc. AISTATS*, 2010, pp. 297–304.

[35] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul, "Fast and robust neural network joint models for statistical machine translation," in *Proc. ACL*, Baltimore, Maryland, June 2014, pp. 1370–1380.

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

[37] Wei Zhou, Wilfried Michel, Kazuki Irie, Markus Kitza, Ralf Schlüter, and Hermann Ney, "The RWTH ASR system for TED-LIUM release 2: Improving hybrid-HMM with SpecAugment," in *Proc. ICASSP*, Barcelona, Spain, May 2020.

[38] Kazuki Irie, Ralf Schlüter, and Hermann Ney, "Bag-of-words input for long history representation in neural network-based language models for speech recognition," in *Proc. Interspeech*, Dresden, Germany, Sept. 2015, pp. 2371–2375.

[39] Shiliang Zhang, Hui Jiang, MingBin Xu, JunFeng Hou, and LiRong Dai, "The fixed-size ordinally-forgetting encoding method for neural network language models," in *Proc. ACL*, Beijing, China, July 2015, pp. 495–500.