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 (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 deeper 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].

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].

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
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:

\[
x_{n}^{(l)} = \text{LayerNorm}(z_{n}^{(l-1)})
\]

\[
q_{n}^{(l)}, k_{n}^{(l)}, v_{n}^{(l)} = Q x_{n}^{(l)}, K x_{n}^{(l)}, V x_{n}^{(l)}
\]

\[
h_{n}^{(l)} = (h_{n-1}^{(l-1)}, q_{n}^{(l)}, v_{n}^{(l)})
\]

\[
y_{n}^{(l)} = \text{Attention}(h_{n}^{(l)}, q_{n}^{(l)})
\]

\[
y_{n}^{(l)} = y_{n}^{(l-1)} + W_{0}y_{n}^{(l)}
\]

where \( Q, K, V \), respectively denote query, key, value projection matrices, \( \text{LayerNorm} \) denotes layer normalization [26], \( \text{Attention} \) 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 \( 2 \times n \times L \times d_{kv} \) which not only grows with the position \( n \) but when we make the model deeper (larger \( L \)) or wider via self-attention dimensions (\( d_{kv} \)). \( y_{n}^{(l)} \) is then fed to the one-layer feed-forward module:

\[
m_{n}^{(l)} = \text{LayerNorm}(y_{n}^{(l)})
\]

\[
z_{n}^{(l)} = \tilde{y}_{n}^{(l)} + W_{2} \text{ReLU}(W_{1}m_{n}^{(l)})
\]

The outer dimension of \( W_{1} \) is typically called the feed-forward inner dimension \( d_{f} \).

3.2. Modifying Transformer layers for smaller state size

For deep Transformer language models, the size of state vectors \( (h_{n}^{(1)}, \ldots, h_{n}^{(L)}) \) 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 \( d_{f} \). 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 \( \ell \)-th 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:

\[
q_{n}^{(l)}, k_{n}^{(l)} = Q x_{n}^{(l)}, K x_{n}^{(l)}
\]

\[
h_{n}^{(l)} = (h_{n-1}^{(l-1)}, q_{n}^{(l)}, v_{n}^{(l)})
\]

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[5]. 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[5]. However, once we apply pruning[5] 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 multi-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 perplexities 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 $d_{ff} = 4096$. 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

<table>
<thead>
<tr>
<th>Model</th>
<th># Param. [M]</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dev</td>
</tr>
<tr>
<td>4-gram</td>
<td>343</td>
<td>105.4</td>
</tr>
<tr>
<td>+ pruning</td>
<td>161</td>
<td>113.2</td>
</tr>
<tr>
<td>5-gram</td>
<td>663</td>
<td>92.3</td>
</tr>
<tr>
<td>+ pruning</td>
<td>169</td>
<td>112.4</td>
</tr>
<tr>
<td>6-gram</td>
<td>1021</td>
<td>86.2</td>
</tr>
<tr>
<td>+ pruning</td>
<td>183</td>
<td>116.2</td>
</tr>
<tr>
<td>LSTM</td>
<td>450</td>
<td>73.5</td>
</tr>
<tr>
<td>Transformer</td>
<td>414</td>
<td>62.0</td>
</tr>
</tbody>
</table>

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 sub-layers) 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. $d_{kv} = 768$ and $H = 12$ for all models. The models with $F = 1$ are standard Transformers.

<table>
<thead>
<tr>
<th>$L$</th>
<th>$F$</th>
<th>$d_{ff}$</th>
<th>State size</th>
<th>#Param. [M]</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dev</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4096</td>
<td>12,288</td>
<td>206</td>
<td>67.9</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>2048</td>
<td>49,152</td>
<td>313</td>
<td>63.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4096</td>
<td>49,152</td>
<td>414</td>
<td><strong>62.0</strong></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>2048</td>
<td>4,608</td>
<td>247</td>
<td>69.3</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>2048</td>
<td>9,216</td>
<td>280</td>
<td>64.5</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>4096</td>
<td>12,288</td>
<td>338</td>
<td>63.4</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>4096</td>
<td>18,432</td>
<td>379</td>
<td>62.2</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>4096</td>
<td>24,576</td>
<td>464</td>
<td><strong>61.4</strong></td>
</tr>
</tbody>
</table>

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
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.

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

<table>
<thead>
<tr>
<th>$L/F$</th>
<th>$d_{a}$</th>
<th>$H$</th>
<th>NCE Train / position</th>
<th>State size / position</th>
<th>#Param. [M]</th>
<th>Perplexity</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 [9]</td>
<td>1</td>
<td>2048</td>
<td>No</td>
<td>32,768</td>
<td>306</td>
<td>56.6</td>
<td>59.5</td>
<td></td>
</tr>
<tr>
<td>42 [9]</td>
<td>2</td>
<td>43,008</td>
<td>No</td>
<td>338</td>
<td>54.2</td>
<td>56.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>4096</td>
<td>No</td>
<td>6,144</td>
<td>307</td>
<td>55.5</td>
<td>58.1</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Shared-KV</th>
<th>$L/F$</th>
<th>State size / position</th>
<th>#Param. [M]</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>32</td>
<td>1</td>
<td>49,152</td>
<td>414</td>
</tr>
<tr>
<td>Yes</td>
<td>8</td>
<td>3</td>
<td>12,288</td>
<td>338</td>
</tr>
<tr>
<td>No</td>
<td>32</td>
<td>1</td>
<td>49,152</td>
<td>414</td>
</tr>
<tr>
<td>Yes</td>
<td>8</td>
<td>3</td>
<td>12,288</td>
<td>338</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>State size for n tokens</th>
<th>#Param. [M]</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher</td>
<td>Transformer</td>
<td>LSTM</td>
<td>$n \times 49,152$</td>
</tr>
<tr>
<td>Student</td>
<td>LSTM</td>
<td>16,384</td>
<td>450</td>
</tr>
</tbody>
</table>

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


