Recurrent Stacking of Layers for Compact Neural Machine Translation Models

Raj Dabre, Atsushi Fujita
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
firstname.lastname@nict.go.jp

Abstract

In encoder-decoder based sequence-to-sequence modeling, the most common practice is to stack a number of recurrent, convolutional, or feed-forward layers in the encoder and decoder. While the addition of each new layer improves the sequence generation quality, this also leads to a significant increase in the number of parameters. In this paper, we propose to share parameters across all layers thereby leading to a recurrently stacked sequence-to-sequence model. We report on an extensive case study on neural machine translation (NMT) using our proposed method, experimenting with a variety of datasets. We empirically show that the translation quality of a model that recurrently stacks a single-layer 6 times, despite its significantly fewer parameters, approaches that of a model that stacks 6 different layers. We also show how our method can benefit from a prevalent way for improving NMT, i.e., extending training data with pseudo-parallel corpora generated by back-translation. We then analyze the effects of recurrently stacked layers by visualizing the attentions of models that use recurrently stacked layers and models that do not. Finally, we explore the limits of parameter sharing where we share even the parameters between the encoder and decoder in addition to recurrent stacking of layers.

Introduction

Sequence-to-sequence (seq2seq) learning (Sutskever, Vinyals, and Le 2014) allows for end-to-end training of a single neural model that can transform an input sequence to another. A typical seq2seq model consists of an encoder and a decoder where the encoder converts the input sequence into a sequence of continuous space vectors and the decoder converts these vectors into a target sequence. One of the most commonly followed practices in sequence-to-sequence modeling is the stacking of multiple recurrent,[1] convolutional, or self-attentional feed-forward layers in the encoders and decoders with each layer having its own parameters. It has been empirically shown that such stacking leads to an improvement in performance, especially in resource-rich scenarios. However, it also increases the size of the model by a significant amount. The search for compact seq2seq models with performance comparable to their bulkier counterparts has led to the development of knowledge distillation approaches (Hinton, Vinyals, and Dean 2015; Freitag, Al-Onaizan, and Sankaran 2017). In this approach, teacher models with a large number of parameters are first trained and then a child model with significantly fewer parameters is trained to mimic the teachers. As a result, a single child model has been shown to perform as well as the ensemble of many parent models.

We propose an alternative solution where we attempt to reduce the sizes of the parent models themselves without an appreciable loss of performance. We thus argue the following research question:

> *Is it really necessary to introduce new parameters when stacking layers in an sequence-to-sequence model?*

In this paper, we try to answer this question by proposing to reduce the number of model parameters via *parameter sharing* across stacked layers. To this end, we take up a case study of neural machine translation (NMT) (Cho et al. 2014; Bahdanau, Cho, and Bengio 2015), which is one of the most popular applications of seq2seq modeling. NMT allows for end-to-end training of a translation system without needing to deal with word alignments, translation rules, and complicated decoding algorithms, that are integral to statistical machine translation (SMT) (Koehn et al. 2007).

We modify the original NMT architecture so that the same parameters are used across all stacked layers, i.e., recurrently stacking (RS)[2] of layers, as illustrated in Figure 1. As a result, our recurrently stacked NMT (RS-NMT) model has the same size of a single-layer NMT model. This paper reports on our extensive study on our RS-NMT model. To show the effectiveness of RS-NMT, we conducted translation experiments using five different datasets, including four for the Japanese–English (ALT, KFTT, GCP, and ASPEC) and one for the Turkish–English (WMT) language pairs, and have observed, for example, that our RS-NMT model with 6 times of recurrence gives results approaching that of a 6-layer vanilla NMT model which does not use any recurrences. We expand our experimental settings to include back-translated corpora (Sennrich, Haddow, and Birch 2016a) and show that our method further benefits from such additional data, demonstrating the potential capacity

[1] Recurrent across time-steps.

[2] RS can mean Recurrently Stacked or Recurrent Stacking both of which have the same implication.

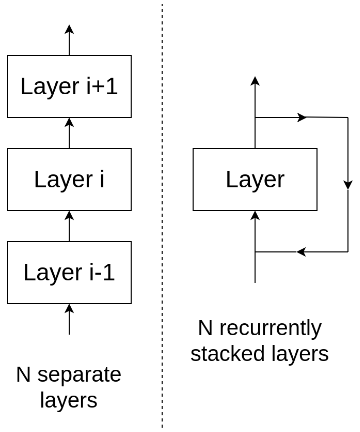


Figure 1: Vanilla layer stacking (left) vs recurrent layer stacking (right). In case of the Transformer NMT model, a layer comprises self-attention, cross-attention (decoder only), and feed-forward sub-layers which can include residual connections and layer normalization.

of RS-NMT models. We also make an analysis on the behavior of the attention mechanism of RS-NMT models; our visualization reveals intriguing characteristics of RS-NMT models. Furthermore, we explore limits of parameter sharing and show that sharing the RS layer parameters between the encoder and the decoder negatively affects the translation quality. However, without RS, such sharing does not have any negative effects on translation quality.

The contributions of this paper are as follows:

Primary Contribution We propose a novel modification to the NMT architecture where parameters are shared across layers, i.e., RS-NMT.

Secondary Contribution Through visualizing the attentions across different layers of RS-NMT models, we analyze them to gain insight into the working of RS-NMT. In doing so, we also gain an understanding of the adaptive power of the attention mechanism.

To the best of our knowledge, this is the first work that shows that it is possible to reduce the NMT model size by sharing parameters across layers and yet achieve results approaching that of a model that does not share parameters across layers. Although we have so far used the *Transformer* (Vaswani et al. 2017) architecture, our method is theoretically architecture independent. Our results will have a direct impact on general sequence-to-sequence (seq2seq) modeling and knowledge distillation approaches because it can lead to extremely compact seq2seq models.

Parameter Sharing in NMT

NMT models usually consist of an encoder and a decoder. The encoder comprises an embedding layer for the source language and N stacked transformation layers. The decoder consists of an embedding layer and a softmax layer for the target language along with M stacked transformation layers, where M is often set to N. The most common way of sharing parameters in NMT models is to use a shared source-target vocabulary and then use the same parameters for the encoder-decoder embeddings and the decoder softmax. On top of them, we propose to perform an additional level of sharing: using the same parameters across the stacked layers in a recurrent fashion.

Figure 1 illustrates our approach. The left-hand side shows the vanilla stacking of N layers where each layer in the neural network has its own parameters. The right-hand side shows our approach of stacking N layers where the same parameters are recurrently used for all layers. In the case of the Transformer model, each layer consists of self-attention, cross-attention (decoder only), and feed-forward sub-layers with layer normalization and residual connections for each sub-layer. Note that in RS-NMT we simply feed the output of a layer to itself. As such, the first input to the RS-layer is the word embedding and all subsequent inputs are the outputs of the RS-layer. This means that an RS-NMT model has the same number of parameters as a 1-layer vanilla NMT model.

Assume that X is the input to the encoder or the decoder consisting of N layers. Let Y be the output of the topmost encoder or decoder layers. Let L_i be the i-th layer and $L_i(X)$ indicates that L_i processes X and produces a hidden representation. Eq. (1) shows the hidden layer computation of a 6-layer vanilla NMT model. In contrast, as shown in Eq. (2), a 6-layer RS-NMT model uses only one layer.

$$Y = L_6(L_5(L_4(L_3(L_2(L_1(X)))))) \tag{1}$$

$$Y = L_1(L_1(L_1(L_1(L_1(L_1(X)))))) \tag{2}$$

As a result of this parameter sharing, the resultant neural model is technically a single-layer model in which the same layer is recurrently stacked N times. This leads to a massive reduction in the size of the model. Recurrently stacking of layers causes the model to revise the hidden states and leads to more complex representations that should help in improving translation quality.

Experimental Settings

In this section we explain all our experimental settings that we followed in order to explore the outcome of RS-NMT with the Transformer model. Specifically, we trained and evaluated the following two types of NMT models.

Vanilla NMT: 2-layer and 6-layer models without any shared parameters across layers.

Recurrently Stacked NMT (RS-NMT): 1, 2, 3, 4, 5, and 6-layer models with parameters shared across all layers.

Datasets and Languages

We evaluated the impact of our proposed method using five datasets. For our Japanese–English (Ja-En) translation for both directions, we used the Asian Language Treebank (ALT) parallel corpus[3] (Thu et al. 2016), the Global Communication Plan (GCP) corpus[4] (Imamura and

[3] http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/

[4] The splits provided by Imamura, Fujita, and Sumita (2018).

Langs	Dataset	Train	Dev	Test	Vocab	Steps
Ja-En	**ALT**	18K	1000	1018	8k	40k
	GCP	400K	2000	2000	16k	60k/120k
	KFTT	440K	1166	1160	8k	160k
	ASPEC	1.50M	1790	1812	32k	400k
Tr-En	**WMT**	208K	3000	3007 3010	16k	50k 4 GPUs

Table 1: Datasets and model settings.

Sumita 2018), the Kyoto free translation task (KFTT) corpus,[5] and the Asian Scientific Paper Excerpt Corpus (ASPEC)[6] (Nakazawa et al. 2016). We also experimented with Turkish–English (Tr-En) language pair using the WMT 2018 corpus.[7] We used newstest2016 for development, and newstest2017 (test17) and newstest2018 (test18) for testing. Table 1 gives the number of parallel sentences for all the datasets.

We tokenized the Japanese sentences in ALT, KFTT, and ASPEC corpora using the JUMAN[8] (Kurohashi et al. 1994) morphological analyzer. We tokenized and lowercased the English sentences of these corpora using the *tokenizer.perl* and *lowercase.perl* scripts in Moses.[9] The GCP corpus was available to us in a pre-tokenized and lowercased form. We did not tokenize the Turkish–English data in any way.[10]

Implementation and Model Settings

We implemented our method on top of an open-source implementation of the Transformer model (Vaswani et al. 2017) in the version 1.6 branch of *tensor2tensor*.[11] We chose the Transformer because it is the current state-of-the-art NMT model but our approach of sharing parameters across layers is independent of implementation and model. For training, we used the default model settings corresponding to *transformer_base_single_gpu* (Vaswani et al. 2017), except the number of sub-words, training iterations, and number of GPUs. These numbers vary as we train the models to convergence.

We used the tensor2tensor internal sub-word segmenter for simplicity. The details of sub-word vocabularies and training iterations are in Table 1. Note that for GCP task, we chose the vocabulary size used by Imamura and Sumita (2018), and trained English-to-Japanese model for 60k iterations whereas we trained for 120k steps for the reverse direction. We used a joint[12] vocabulary and 4 GPUs for training the models only for the WMT dataset.

We averaged the last 10 check-points and decoded the test set sentences with a beam size of 4 and length penalty of $\alpha = 0.6$ for the KFTT Japanese-to-English experiments and $\alpha = 1.0$ for the rest. We evaluated our models using the BLEU metric (Papineni et al. 2002) implemented in tensor2tensor as *t2t_bleu*: case-insensitive and tokenized BLEU for the Japanese–English tasks, and case-sensitive and detokenized BLEU for the Turkish–English tasks.

Main Results

Recurrently Stacked Models vs Vanilla Models

Table 2 presents the results of the experiments using up to 6-layers of RS-NMT for all the datasets. In general, no matter which the dataset was used, the translation quality improved as the same parameters were recurrently used in a depth-wise fashion. In 9 out of 12 translation tasks, the 6-layer RS-NMT models were better than the 2-layer vanilla NMT models that have more parameters, meaning that our RS-NMT models can compensate for the lack of parameters. Furthermore, the performance of a 6-layer RS-NMT model approaches that of the vanilla 6-layer NMT model. For the resource-richest ASPEC dataset, the 6-layer RS-NMT models were better than the 1-layer vanilla NMT models by 3.92 and 7.85 BLEU points, respectively, and they were worse than the 6-layer vanilla NMT models by only 1.57 and 1.28 BLEU points, respectively. Similar observations hold for the WMT, KFTT, and GCP datasets.

For the resource-poorest ALT Japanese-to-English translation, however, we did not observe much difference among the 1-layer, 6-layer vanilla NMT, and 6-layer RS-NMT models. Although our method does not seem to work well for ALT, we believe that this is related to the poor translation performance of NMT in such low-resource scenarios.

Effect of Corpora Sizes on Translation Quality

Figure 2 shows the difference in the performance between the RS-NMT and the vanilla NMT when we vary the size of the training data for the same datasets. We plot the BLEU scores for GCP English-to-Japanese, ASPEC English-to-Japanese, and WMT Turkish-to-English translation. While it is obvious that reducing the size of training data deteriorates the BLEU scores, the trends in terms of difference in performance between the RS-NMT and vanilla NMT remain almost the same. As such, we can safely say that our proposed method is independent of corpus size and domain.

Analyzing Recurrently Stacked Models

In this section, we answer the following four questions to better understand the RS-NMT models.

(a) Does RS-NMT memorize the number of recurrence?

(b) Is RS-NMT complementary with the back-translation approach?

(c) Does RS-NMT behave differently from vanilla NMT internally?

(d) Can we further push the limits of parameter sharing?

[5] http://www.phontron.com/kftt

[6] We used the cleaner half of this corpus.

[7] http://www.statmt.org/wmt18/translation-task.html

[8] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

[9] http://www.statmt.org/moses

[10] tensor2tensor has an internal tokenization mechanism which was used for this language pair.

[11] https://github.com/tensorflow/tensor2tensor

[12] We did this to exploit cognates across Turkish and English.

#recurrently stacked layers	ALT		GCP		KFTT		ASPEC		WMT			
	Ja-En	**En-Ja**	**Ja-En**	**En-Ja**	**Ja-En**	**En-Ja**	**Ja-En**	**En-Ja**	**Tr-En test17**	**Tr-En test18**	**En-Tr test17**	**En-Tr test18**
1	7.59	10.59	21.95	23.89	21.64	25.00	23.28	32.19	13.08	13.75	12.45	11.94
2	7.60	10.92	23.24	24.47	24.50	28.53	27.84	38.54	15.19	15.95	15.07	14.62
3	7.99	11.14	23.42	25.02	25.84	29.90	28.05	39.26	15.80	16.39	15.98	14.68
4	7.91	11.30	24.33	25.28	26.23	30.36	**28.08**	39.31	16.38	17.05	16.52	14.93
5	**8.28**	11.30	23.95	25.38	26.42	30.78	28.02	38.86	16.63	17.12	**16.60**	15.51
6	8.26	**11.37**	**24.36**	**25.84**	**26.51**	**30.83**	27.20	**40.04**	**16.68**	**17.31**	16.59	**15.77**
2-layer model	8.35	11.90	24.23	25.62	24.14	30.05	28.06	38.91	16.61	17.17	16.55	15.37
6-layer model	8.47	13.21	24.67	26.22	27.19	32.72	28.77	41.32	17.80	18.36	17.99	16.29

Table 2: Results (BLEU scores) for our experiments. The last two rows indicate the vanilla NMT models without any parameter sharing. The results in bold indicate the best values among the diverse models of the proposed RS-NMT.

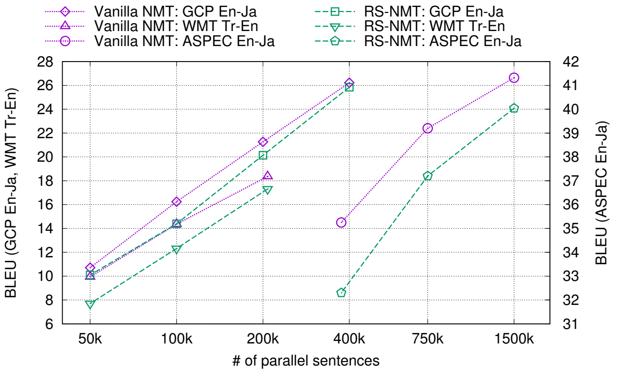


Figure 2: BLEU scores of the 6-layer vanilla NMT and RS-NMT models trained on different sizes of data. WMT Tr-En shows the results for test18.

(a) #Recurrence Memorized by RS-NMT

Referring to Table 2, the performance of RS-NMT models seems to stabilize with the increasing depth of RS. As such, we expected the representations generated by deep RS will be robust and thus enable us to use fewer layers of RS during decoding as compared to during training. To confirm whether this holds or not, we trained N-layer RS models and used M times of recurrence during testing, taking GCP English-to-Japanese task as an example. Here, we evaluated two configurations: (1) one where we performed the same depth of RS for both the encoder and decoder, and (2) one where the depth of RS for the encoder was the same as for training while the depth of RS for the decoder was varied. For example, a trained 6-layer RS-NMT model was tested with (1) do 1 to 8 times of RS for both the encoder and decoder and (2) do 6 times of RS for the encoder (same as training) but do 1 to 8 times of RS for the decoder.

Figure 3 summarizes the BLEU scores obtained with the above two configurations. Once the NMT model has been trained to use N RS layers, it is unable to perform optimally if N RS layers are not used for decoding. Beside that, there are three crucial observations. Although they are applicable to all N-layer RS-NMT models, we explain them, taking our 6-layer model as an example.

First, the computation of the most useful and hence the most reliable features takes place at the deepest layer of recurrence. For a 6-layer RS-NMT model, using both the encoder and the decoder just once during decoding, gave a BLEU of 2.56. This went up to 4.45 when using 6 times of recurrence for the encoder (same as during training) and the layer of the decoder just once. However, as we performed more times of recurrence, the BLEU jumped drastically. This could imply that the NMT model avoids the learning of extremely reliable/complex features at shallower layers. At the very least, there are no detrimental effects of RS. Furthermore, RS of the encoder for the same number of times as during training is also important.

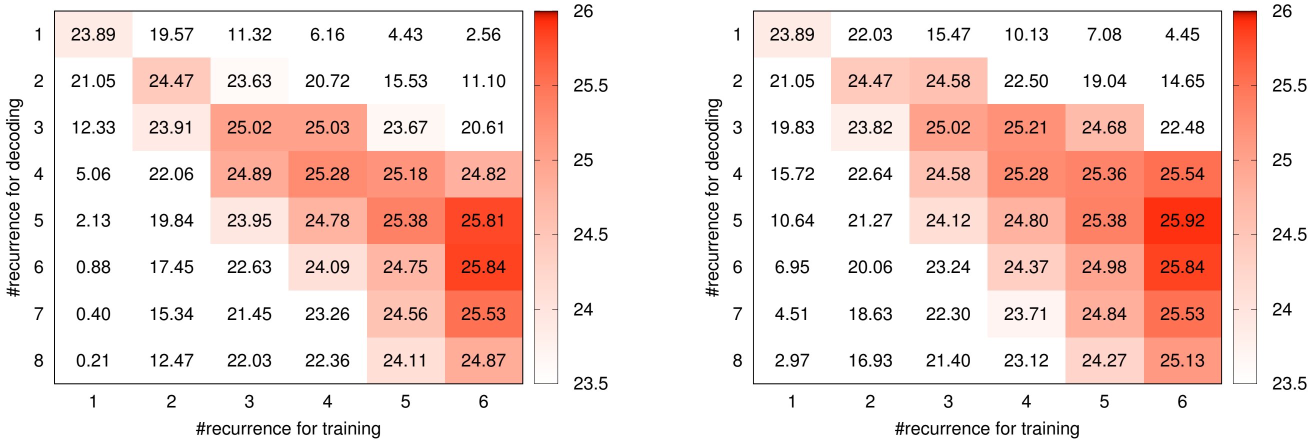
Second, for a 6-layer RS model, the difference between decoding using 6-layer RS and 5-layer RS was not significant. Thus, deep RS leads to robustness during decoding and this means that as we train models using a large number of RS, it should be possible to use fewer RS during decoding.

Third, when we used deeper RS than what had been used for training, the BLEU score started dropping again. This indicates that the model has not learned to extract complex features beyond what it has been trained for. However, once the model has been trained for non-zero number of recurrences, the drop in quality is less severe as can be seen for a 3-layer RS-NMT model being decoded for more than 3 times of recurrence. Eventually, using a different layer stacking configuration for training and decoding leads to only sub-optimal results and thus we conclude that in its current form, the RS-NMT is unable to generalize the variation in the number of recurrence between training and decoding, even though it can use fewer number of recurrences during decoding for reasonable translation quality.

In the future, we will see what happens when we train deeper RS models to identify the limit of recurrence. In this way, we can identify the zone of tolerance and thus use different depths of RS layers depending on the use case. One obvious idea is to use shallower RS models for back-translation experiments (Sennrich, Haddow, and Birch 2016a), where we can afford to sacrifice the back-translation quality for processing speed because each additional layer demands more computing time.

(b) RS-NMT and Back-Translation

Since back-translation is one of the most reliable ways to boost translation quality (Sennrich, Haddow, and Birch 2016a), we evaluated its impact on our proposed method. We experimented with the GCP and WMT tasks, using 1.55M



(1) Varying #recurrence for both encoder and decoder.

(2) Varying #recurrence for decoder only.

Figure 3: BLEU scores of the RS-NMT models for the GCP English-to-Japanese task with diverse depths of RS layers during decoding.

#recurrently stacked layers	**GCP, Ja-En**		**GCP, En-Ja**		**WMT, Tr-En**				**WMT, En-Tr**			
					test17		**test18**		**test17**		**test18**	
	No	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**
1	21.95	**23.90**	23.89	**25.47**	13.08	**16.36**	13.75	**16.52**	12.45	**17.54**	11.94	**15.02**
2	23.24	**24.79**	24.47	**26.56**	15.19	**18.07**	15.95	**18.42**	15.07	**19.34**	14.62	**16.64**
3	23.42	**24.79**	25.02	**26.66**	15.80	**19.05**	16.39	**19.48**	15.98	**20.30**	14.68	**17.24**
4	24.33	**25.17**	25.28	**27.31**	16.38	**19.24**	17.05	**19.67**	16.52	**20.55**	14.93	**17.56**
5	23.95	**24.92**	25.38	**27.08**	16.63	**19.17**	17.12	**20.00**	16.60	**20.55**	15.51	**17.97**
6	24.36	**25.82**	25.84	**27.55**	16.68	**19.55**	17.31	**20.41**	16.59	**20.89**	15.77	**17.78**
6-layer model	24.67	**25.91**	26.22	**28.75**	17.80	**21.19**	18.36	**21.95**	17.99	**22.22**	16.29	**19.18**

Table 3: Results (BLEU scores) of using back-translated data for the GCP and WMT tasks. The "Yes" and "No" columns indicate the involvement of back-translated data. Higher BLEU scores are in bold.

lines of Japanese and English monolingual corpora for GCP (Imamura, Fujita, and Sumita 2018) and 2.63M lines of Turkish and English monolingual corpora for WMT, separately for each translation direction. We generated pseudo-parallel corpora by back-translating the monolingual sentences using the 1-layer models for the opposite translation direction,[13] regarding their speed. We then trained from scratch the 6-layer vanilla NMT and up to 6-layer RS-NMT models on the mixture of the pseudo-parallel and the original parallel corpora. To compensate for the additional data, we trained both the GCP Japanese-to-English and the English-to-Japanese models for 200k iterations on 1 GPU. Similarly, we trained the WMT English-to-Turkish and Turkish-to-English models for 150k iterations on 4 GPUs.

Table 3 provides the results. Despite no increase in the number of parameters, the presence of back-translated data improved the translation quality for all the translation tasks. The 2-layer RS-NMT trained using additional back-translated data already outperformed the 6-layer vanilla NMT models trained only on the original parallel corpus. It is clear that the gains using additional layers of recurrence in a low-resource scenario is much higher than the gains in a resource-rich scenario, despite the potentially lower quality of back-translated data.

(c) Visualizing Recurrently Stacked Models

To acquire a deeper understanding of what happens in RS models, we visualize the attentions across all attention heads for each stacked layer. In particular, we consider the GCP Japanese-to-English vanilla NMT and RS-NMT models trained using back-translated data and visualize[14] their encoder's self-attentions and the decoder's self- and cross-attentions. Due to lack of space we only show the visualizations for attentions of a target word with the entire source sentence. See several sentence-level self- and cross-attention visualizations in our supplementary material.[15]

Figure 4 displays the heat-maps for the 8-head encoder-decoder cross-attention across all 6-layers when generating the first translated word, "how," for an input Japanese sentence consisting of nine tokens, "*dono-youni choushi ga sugure nai no desu ka ?*" (How don't you feel well?). The x-axis shows the input tokens with a prefix indicating an attention head represented by a different color. The y-axis shows the

[13] For example, we used the 1-layer Japanese-to-English model in order to translate the Japanese monolingual sentences for training English-to-Japanese NMT.

[14] We used the Bokeh Python library.

[15] https://github.com/prajdabre/RSNMT

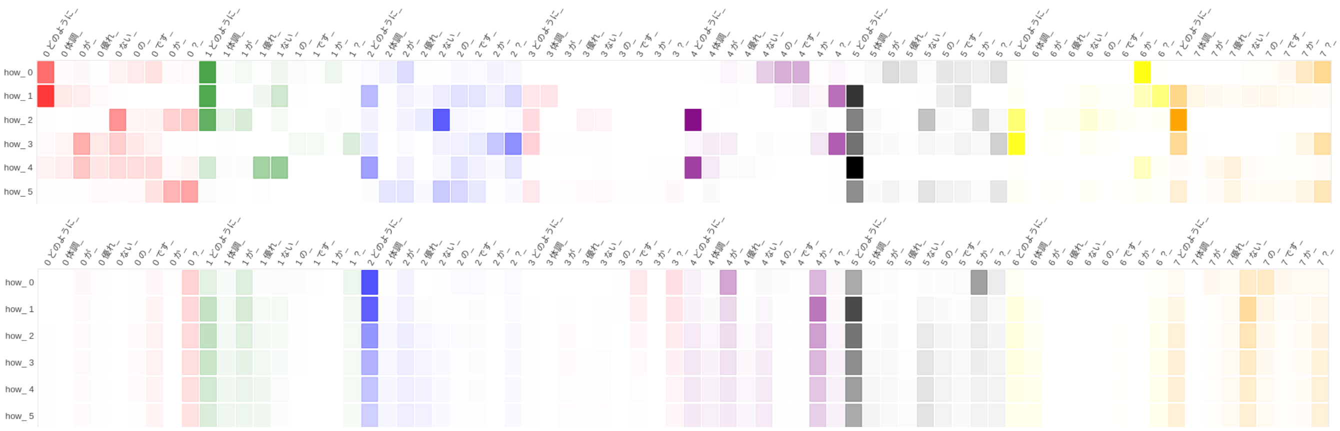


Figure 4: Visualization of the cross-attention of vanilla NMT (top) and RS-NMT (bottom) models trained on the Japanese-to-English GCP corpus and back-translated data when generating a single word in the target language for an input sentence. The x-axis indicates the input sentence and the attention heads (marked by different colors). The y-axis indicates the depth of the layer.

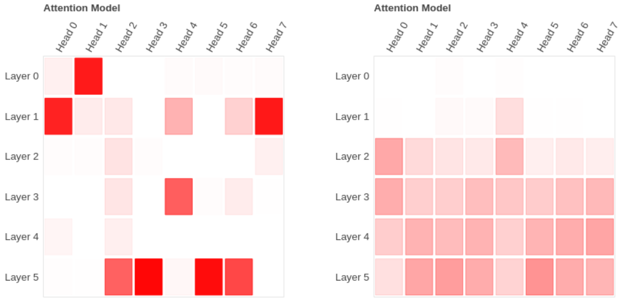


Figure 5: Visualization of attention entropies of the same vanilla (left) and recurrent stacking (right) models as in Figure 4 for another input sentence, "*o tsuri no 200 yen desu .*" (Here is your 200 yen change.). The x-axis indicates the attention heads. The y-axis indicates the depth of the layer. A darker color means higher value of entropy.

6 layers with 0-to-5 in suffix representing 1st-to-6th layer, respectively. Each cell shows the cross-attention given by a head at a layer to a source word, when generating the target word, where darker color means stronger attention.

As shown in the figure, the attention mechanism behaves differently for the vanilla NMT and RS-NMT models. While there is no consistency in the sharpness of attention as we go deeper in the vanilla NMT model, the attention tends to become less sharp as we go deeper in the RS-NMT model. For a further understanding, we produced a heat-map (Figure 5) that displays the entropies of the attentions for each attention head in each layer. While we do not know what this means in the context of the vector space representations of an NMT model, we can say that the attention entropy tends to grow as we move towards deeper layers in RS-NMT models. An increase in attention entropy means that the attention mechanism considers the contribution from more words by giving them similar weights. Consider the work on average attention networks (Xiong, Zhang, and Su 2018) where the self-attention weights were statically set as $1/L$ where L is the length of the sequence against which the attention is computed for a given word. Average attention has the highest entropy which is similar to the case of the attention computed in RS networks. As such, our observation might be an explanation of why average attention networks work well despite using a trivial self-attention mechanism.

Another probable explanation lies in the understanding of what Recurrent Neural Networks (RNN), such as LSTMs (Hochreiter and Schmidhuber 1997) do. At each time-step, an RNN refines the representation of a sequence as it processes each new word with the same parameters. As such, we think that the deeper layers of the RS-NMT model are forced to learn more abstract representations through refinement of the previous ones. The parameter sharing (and thus the reduction in representation power) most likely causes all artificial neurons to work in unison and uniformly distributes the load of generating reliable representations. This could mean that the representation for each word is rather generic. And this in turn causes the attention mechanism to seek out more work for the decoder to compute reliable representations. Since the attentions for RS-NMT and vanilla NMT differ greatly, they might have different applications. For example, RS-NMT attentions might be more suited for phrase-to-phrase matching due to the wide attention spans it tends to exhibit. It would be worthwhile to visualize the word- and sentence-level representations to explore our claims.

(d) Limits of Parameter Sharing

So far, we have only shared parameters across layers but have kept the parameter of the encoder and decoder layers separate. We also explored what happens when we further share parameters between the encoder and the decoder. Note that we can only share the parameters for self-attention and the feed-forward layers because the cross-attention layers are specific to the decoder. To this end, we trained two ad-

#recurrently stacked layers	GCP, En-Ja		WMT, Tr-En test18	
	w/o	w/	w/o	w/
1	**23.89**	23.19	**13.75**	13.15
2	**24.47**	23.29	**15.95**	15.18
3	**25.02**	24.03	**16.39**	16.14
4	**25.28**	24.42	**17.05**	16.42
5	**25.38**	24.57	**17.12**	16.55
6	**25.84**	24.98	**17.31**	16.67
6-layer model	26.22	**26.32**	18.36	**18.48**

Table 4: A comparison of models with and without encoder-decoder parameter sharing. Higher BLEU scores are in bold.

ditional types of model along with the ones described in the beginning of the experimental section:

Shared-EncDec: 6-layer model without any shared parameters across layers but shared self-attention and feed-forward layer parameters across the encoder and decoder.

Recurrently Stacked & Shared-EncDec 1, 2, 3, 4, 5, and 6-layer models with parameters shared across all layers and across the encoder and decoder.

Table 4 compares RS-NMT and vanilla NMT models for GCP English-to-Japanese and WMT Turkish-to-English tasks. These models either do or do not share parameters between the encoder and the decoder. Once again, as we increased the number of RS, the translation quality improved for both translation tasks. However, encoder-decoder parameter sharing had a negative effect on RS-NMT models, although the reduction was within 1.0 BLEU points. Surprisingly, this encoder-decoder parameter sharing had a slightly positive effect on vanilla NMT models. Currently, we are not sure why this happens and leave this for future work.

Table 5 gives the cost-benefit analysis for model size versus loss in terms of BLEU for GCP English-to-Japanese and WMT Turkish-to-English tasks. The number of parameters in a 6-layer vanilla NMT model for Turkish-to-English task is 158.8M which drops to 102.1M by sharing the self-attention and feed-forward layer parameters between the encoder and the decoder. The number of parameters for the RS-NMT models is 48.6M no matter how many layers are in the stack and this number further drops to 39.1M when sharing parameters between the encoder and the decoder. Recurrently stacking of layers and sharing them between the encoder and decoder leads to a loss of 1.66 BLEU which is significant but given that we eliminate about 75% of the parameters such a loss is quite acceptable. Similarly, for GCP English-to-Japanese task, the RS-NMT models are approximately 53–57% smaller than the 6-layer vanilla NMT models and lose only about 0.38–1.24 BLEU points.[16]

If one considers a resource-rich scenario like ASPEC English-to-Japanese translation where the BLEU scores are in the lower 40's, losing 2–3 BLEU points is not a major loss given that the parameter sharing models are significantly smaller than their vanilla counterparts.

Related Work

The most prominent way of reducing the size of a neural model is knowledge distillation (Hinton, Vinyals, and Dean 2015; Freitag, Al-Onaizan, and Sankaran 2017), which requires training one or more parent models and thus is a time-consuming task. Our approach is orthogonal to knowledge distillation and proposes to shrink the sizes of parent models. It is possible for both these methods could be combined to yield extremely compact NMT models. The work on zero-shot NMT (Johnson et al. 2017) shows that it is possible for multiple language pairs to share a single encoder and decoder without an appreciable loss in translation quality. As a result, multiple language pairs can share the same encoder and decoder parameters without significant loss in the translation quality, thereby avoiding the need to train separate models per language pair. For languages that share orthographies (Sennrich, Haddow, and Birch 2016b) or have orthographies that can be mapped to a common script, using a shared embedding layer can help reduce the model size significantly and enable cognate sharing. However, these approaches do not consider the effect of sharing the parameters across the stacked layers in the encoder or the decoder.

The work on Universal Transformer (Dehghani et al. 2018) shows that feeding the output of the multi-layer encoder (and decoder) to itself repeatedly leads to an improvement in quality for English-to-German translation. Our method is similar to this, except that our RS-NMT model has the same size as that of a 1-layer NMT model and yet manages to approach the translation quality achieved by a 6-layer vanilla NMT model. As such, the focus of our work is on training models with significantly fewer parameters whereas Dehghani et al. (2018) have focused on improving the state-of-the-art. We additionally show that the RS-NMT can benefit from back-translated data.

Conclusion

In this paper, we have proposed a novel modification to the NMT architecture where we share parameters across the layers of a N-layer model leading to a recurrently stacked NMT (RS-NMT) model. As a result, our model has the same size as that of a single-layer NMT model and gives performance approaching that of a 6-layer vanilla NMT model where the parameters across layers are not shared. This shows that it is possible to train compact NMT models without a large loss in translation quality. We also showed that our approach is complementary with the back-translation approach. We also visualized the attentions of our models and showed that the internal working of RS-NMT is quite different from vanilla NMT models. We believe that our work will promote the research of techniques that rely on reusability of parameters and hence simplify the existing NMT architectures.

In the future, we will perform an in-depth analysis of the limits of recurrent stacking of layers in addition to combining our methods with knowledge distillation approaches for high-performance and compact NMT modeling. We also plan to experiment with more complex mechanisms to compute the recurrent information during stacking for improving NMT performance.

[16] Note that the Turkish-to-English models have a shared matrix for encoder embedding, decoder embedding and softmax and thus will have greater savings in terms of parameters.

Recurrent Stacking	Shared EncDec	GCP, En-Ja			WMT, Tr-En			
		#Params	%Params Reduced	BLEU	#Params	%Params Reduced	BLEU test17	BLEU test18
		207.9M	0	26.22	158.8M	0	17.80	18.36
	✓	151.2M	37.50	26.32	102.1M	35.68	17.51	18.48
✓		97.6M	53.02	25.84	48.6M	69.38	16.68	17.31
✓	✓	88.2M	57.57	24.98	39.1M	75.33	16.14	16.67

Table 5: Reduction in the number of parameters and BLEU scores by recurrently stacking of layers or encoder-decoder parameter sharing for GCP English-to-Japanese and WMT Turkish-to-English tasks. All results are for 6-layer models.

Acknowledgments

One of the corpora used in our study, GCP, was created under a MIC program "Promotion of Global Communications Plan: Research, Development, and Social Demonstration of Multilingual Speech Translation Technology."

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Cho, K.; van Merriënboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, Ł. 2018. Universal transformers. *CoRR* abs/1807.03819.

Freitag, M.; Al-Onaizan, Y.; and Sankaran, B. 2017. Ensemble distillation for neural machine translation. *CoRR* abs/1702.01802.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Imamura, K., and Sumita, E. 2018. Multilingual parallel corpus for global communication plan. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, 3453–3458.

Imamura, K.; Fujita, A.; and Sumita, E. 2018. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, 55–63.

Johnson, M.; Schuster, M.; Le, Q.; Krikun, M.; Wu, Y.; Chen, Z.; Thorat, N.; Viégas, F. a.; Wattenberg, M.; Corrado, G.; Hughes, M.; and Dean, J. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics* 5:339–351.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; Dyer, C.; Bojar, O.; Constantin, A.; and Herbst, E. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, 177–180.

Kurohashi, S.; Nakamura, T.; Matsumoto, Y.; and Nagao, M. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, 22–28.

Nakazawa, T.; Yaguchi, M.; Uchimoto, K.; Utiyama, M.; Sumita, E.; Kurohashi, S.; and Isahara, H. 2016. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, 2204–2208.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, 311–318.

Sennrich, R.; Haddow, B.; and Birch, A. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, 86–96.

Sennrich, R.; Haddow, B.; and Birch, A. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, 1715–1725.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th Neural Information Processing Systems Conference (NIPS)*, 3104–3112.

Thu, Y. K.; Pa, W. P.; Utiyama, M.; Finch, A.; and Sumita, E. 2016. Introducing the Asian language treebank (ALT). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, 1574–1578.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of the 30th Neural Information Processing Systems Conference (NIPS)*, 5998–6008.

Xiong, D.; Zhang, B.; and Su, J. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Long Papers*, 1789–1798.