Residual Stacking of RNNs for Neural Machine Translation

Raphael Shu
The University of Tokyo
shu@nlab.ci.i.u-tokyo.ac.jp

Akiva Miura
Nara Institute of Science and Technology
miura.akiba.lr9@is.naist.jp

Abstract

To enhance Neural Machine Translation models, several obvious ways such as enlarging the hidden size of recurrent layers and stacking multiple layers of RNN can be considered. Surprisingly, we observe that using naively stacked RNNs in the decoder slows down the training and leads to degradation in performance. In this paper, We demonstrate that applying residual connections in the depth of stacked RNNs can help the optimization, which is referred to as residual stacking. In empirical evaluation, residual stacking of decoder RNNs gives superior results compared to other methods of enhancing the model with a fixed parameter budget. Our submitted systems in WAT2016 are based on a NMT model ensemble with residual stacking in the decoder. To further improve the performance, we also attempt various methods of system combination in our experiments.

1 Introduction

Recently, the performance of machine translation is greatly improved by applying neural networks partially in a Statistical Machine Translation (SMT) pipeline (Zou et al., 2013) or training a end-to-end neural network based machine translation model (Sutskever et al., 2014). The latter approach, or Neural Machine Translation (NMT), possess several advantages compared to conventional SMT approaches. Firstly, as the translation process is done with a single network, the pipeline of NMT training is simpler. This advantage also indicates a much lower implementation cost. Developing a SMT decoder will cost one to three months for a graduate student, while training a NMT model requires merely a script that contains the network definition. Secondly, the memory consumption is vastly reduced when translating with a NMT model. While a conventional phrase-based model trained on a large bilingual corpus can easily consumes over 100GB memory space, a trained NMT model typically requires less than 1GB GPU on-board memory in test time.

The most significant difference of NMT approach in contrast with SMT is that the information for producing the translation is held in continuous space but not discrete values. While this characteristics of neural network introduces huge difficulties in debugging, this allows the model to learn translation knowledge beyond the conventional hand-crafted MT framework. Recently, even the word embeddings obtained from NMT training demonstrates advantages over specific tasks (Hill et al., 2014).

The network architectures of NMT models are simple but effective. It produces a sentence representation with the encoder, then the decoder generates the translation from the vector of sentence representation. Generally the encoder is composed of a single recurrent neural network (RNN). The decoder reads the vector representations created by the encoder and produce a series of output vectors. A linear transformation is applied to each vector in order to create a large softmax layer in size of the whole vocabulary. Soft attention mechanism introduced in (Bahdanau et al., 2014) further boosted the performance by increasing the computational complexity.

With the advance of deep learning, deeper network architectures are favored as the models become more expressive with more layers. An absolute way to deepening NMT models is to stack multiple layers of RNN in either encoder or decoder side. However, our experiments show that stacking RNNs naively in the decoder will cause significant slowdown in training and results in degraded performance.

Proceedings of the 3rd Workshop on Asian Translation,
pages 223–229, Osaka, Japan, December 11-17 2016.

In this paper, we explores the effects of applying residual connection (He et al., 2015) between stacked RNNs in the decoder. We found the residual connection successfully helps the deepened NMT model, which leads to a performance gain of evaluation scores.

In our submitted systems for English-Japanese translation task in WAT2016 (Nakazawa et al., 2016a), we also attempts to combination the advantages of Tree-to-String (T2S) SMT systems and NMT models. Specifically, we experimented combination methods with both simple heuristics and Minimum Bayesian Risk (MBR) based approach (Duh et al., 2011).

2 Background

2.1 Neural Machine Translation

In this paper, we adapt the same network architecture of (Bahdanau et al., 2014), in which a bi-directional RNN is used as the encoder. Let $\boldsymbol{e_1}, ..., \boldsymbol{e_N}$ be the word embeddings of input words. Note these embeddings are initialized randomly and optimized simultaneously with other parameters. The hidden states of the bi-directional RNN are computed as:

$$\overrightarrow{\boldsymbol{h}}_{\boldsymbol{i}} = f(\boldsymbol{e_i}, \overrightarrow{\boldsymbol{h}}_{\boldsymbol{i-1}}; \theta_r) \tag{1}$$

$$\overleftarrow{\boldsymbol{h}}_{\boldsymbol{i}} = f(\boldsymbol{e_i}, \overleftarrow{\boldsymbol{h}}_{\boldsymbol{i+1}}; \theta_l) \tag{2}$$

$$\bar{\boldsymbol{h}}_{\boldsymbol{i}} = concat[\overrightarrow{\boldsymbol{h}}_{\boldsymbol{i}}; \overleftarrow{\boldsymbol{h}}_{\boldsymbol{i}}] \tag{3}$$

Where f is the RNN computation, $\overrightarrow{\boldsymbol{h}}_{\boldsymbol{i}}$ and $\overleftarrow{\boldsymbol{h}}_{\boldsymbol{i}}$ are the hidden states of the RNNs in two directions in time step t. The two RNNs possess different parameters: θ_r and θ_l. The final output of the encoder in each time step is a concatenated vector of the hidden states in two networks.

The decoder firstly computes attention weights for each $\bar{\boldsymbol{h}}_{\boldsymbol{i}}$ in each time step t as:

$$a_t(i) = \frac{\text{score}(\bar{\boldsymbol{h}}_{\boldsymbol{i}}, \boldsymbol{h_{t-1}})}{\sum_j \text{score}(\bar{\boldsymbol{h}}_{\boldsymbol{j}}, \boldsymbol{h_{t-1}})} \tag{4}$$

$$\text{score}(\bar{\boldsymbol{h}}_{\boldsymbol{i}}, \boldsymbol{h_{t-1}}) = \boldsymbol{v_a}^\top \tanh(\boldsymbol{W_a} concat[\bar{\boldsymbol{h}}_{\boldsymbol{i}}; \boldsymbol{h_{t-1}}]) \tag{5}$$

Where the attentional weight $a_t(i)$ is determined by a score function, which receives one encoder state $\bar{\boldsymbol{h}}_{\boldsymbol{i}}$ and the previous decoder state $\boldsymbol{h_t}$ as input. Several possible implementation of this score function is discussed in (Luong et al., 2015) in detail. In this paper, the score function in the original paper of (Bahdanau et al., 2014) is adopted, which has two parameters: $\boldsymbol{v_a}$ and $\boldsymbol{W_a}$. In the decoder part, a context vector, which is a weighted summarization of encoder states is calculated as:

$$\boldsymbol{c_t} = \sum_i a_t(i) \bar{\boldsymbol{h}}_{\boldsymbol{i}} \tag{6}$$

The computation of each hidden state $\boldsymbol{h_t}$ of the decoder RNN is shown as follows:

$$\boldsymbol{h_t} = f(\boldsymbol{c_t}, \boldsymbol{e_{t-1}}, \boldsymbol{h_{t-1}}; \theta_d) \tag{7}$$

$$\boldsymbol{o_t} = \boldsymbol{W_o h_t} + \boldsymbol{b_o} \tag{8}$$

Where $\boldsymbol{e_{t-1}}$ is the word embedding of the previous generated word. A large softmax layer $\boldsymbol{o_t}$ is then computed based on the decoder state $\boldsymbol{h_t}$, which is used to compute the cross-entropy cost. Notably, although $\boldsymbol{e_{t-1}}$ shall be the embedding of previous generated word, this word is directly drawn from the target translation during training time. This aims to speed up the training but also introduces exposure bias, which is further discussed in a recent paper (Shen et al., 2015).

2.2 Generalized Minimum Bayes Risk System Combination

In this section, we briefly introduce Generalized Minimum Bayes Risk (GMBR) system combination (Duh et al., 2011), which is used in our evaluations, more details can be found in the original paper.

The objective of the system combination is to find a decision rule $\delta(f) \rightarrow e'$, which takes f as input and generates a e' as output. MBR system combination search for an optimized decision with:

$$\arg\min_{\delta(f)} \sum_{e} L\left(\delta\left(f\right)|e\right) p\left(e|f\right) \tag{9}$$

$$\approx \arg\min_{e' \in N(f)} \sum_{e \in N(f)} L\left(e'|e\right) p\left(e|f\right) \tag{10}$$

Where L is a loss function for scoring $\delta(f)$ given a reference e. $p(e|f)$ is a posterior for e to be translated from f. In Equation 10, the true set of translation is approximated by N-best list. GMBR method does not directly take the scores of candidate systems to compute the loss as they are not comparable. Instead, it substitutes $L(e'|e)$ with $L(e'|e;\theta)$, where θ is a parameter. In (Duh et al., 2011), the loss function is computed based on several features including n-gram precision and brevity penalty. The parameters are trained on a development dataset.

3 Residual Stacking of decoder RNNs

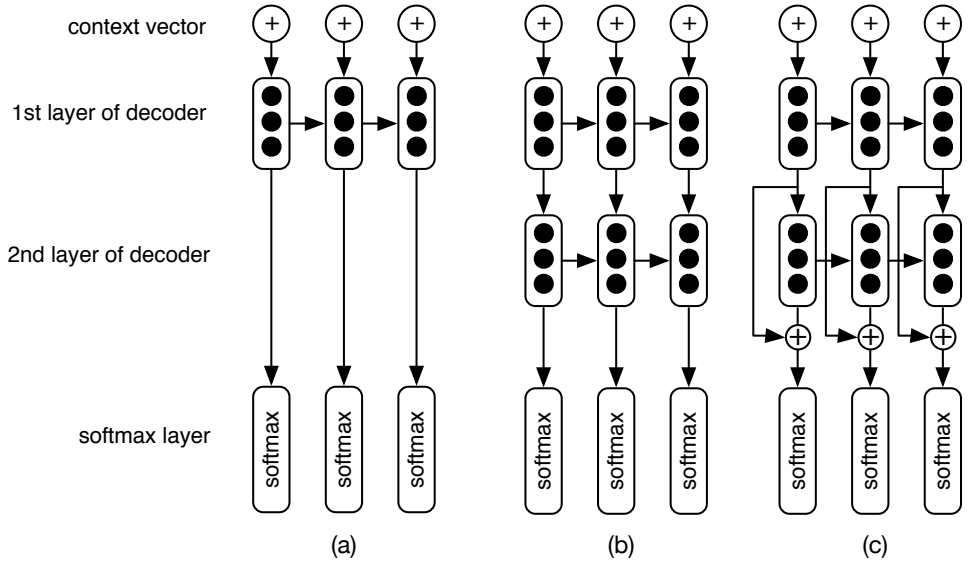

Figure 1: A comparison of three kinds of decoders. (a) A single-layer RNN (b) A stacked two-layer RNN (c) A two-layer residual stacking of RNNs

In this sections, we consider several possible ways to enhance the decoder of NMT models. Two obvious approaches is to enlarge the hidden size of the decoder RNN and to stack more RNNs. Unfortunately, in our experiments we found deepening the decoder slows down the training and finally degrades the final performance. In Figure 1, a decoder with stacked multi-layer RNNs is shown in (b), in comparison with the normal decoder shown in (a).

Recently, several techniques are proposed mainly in computer vision to help the training of very deep networks, such as Highway Networks (Srivastava et al., 2015) and Residual Learning (He et al., 2015). In this paper, we examine the effect of applying Residual Learning to RNN in sequence-to-sequence learning task, where we refer to as residual stacking of RNNs. The implementation of residual stacking is simple and does not involve extra parameters compared to stacked RNNs. The computation of a decoder with Residual RNN is shown as follows:

$$\boldsymbol{h_t} = f(\boldsymbol{c_t}, \boldsymbol{e_{t-1}}, \boldsymbol{h_{t-1}}; \theta_d) \tag{11}$$
$$\boldsymbol{h'_t} = f(\boldsymbol{h_t}, \boldsymbol{h'_{t-1}}; \theta_{d'}) \tag{12}$$
$$\boldsymbol{o_t} = \boldsymbol{W_o}(\boldsymbol{h_t} + \boldsymbol{h'_t}) + \boldsymbol{b_o} \tag{13}$$

As demonstrated in Figure 1(c), another RNN is stacked upon the original decoder RNN, whose hidden states are computed based on the original decoder states. In Equation 13, similar to Residual Learning, instead directly compute the softmax layer based on the outputs of the second RNN, a summation of the states in two RNNs is used. This simple technique is expected to shorten the back-propagation path of the deepened NMT model, which eventually helps the optimization algorithms to train the network.

4 Experiments of residual decoder RNNs

4.1 Settings

In our experimental evaluations, we adopt the fore-mentioned architecture of NMT models described in (Bahdanau et al., 2014). The baseline model contains a single RNN for the decoder. We use LSTM instead of GRU in our experiments. All RNNs have 1000 hidden units for recurrent computation. We then make three model variations and test them in our experiments:

1. Enlarged decoder RNN: the decoder RNN has 1400 hidden units
2. Stacked decoder RNN: the decoder has two-layer RNN stacked, corresponding to Figure 1(b)
3. Residual decoder RNN: the decoder is a two-layer residual stacking of RNNs, corresponding to Figure 1(c)

We design these three model variations in purpose to keep the same number of overall parameters. However, training speed may be vary due to the difference of implementations.

All experiments are performed on ASPEC English-Japanese translation dataset(Nakazawa et al., 2016b). The pre-processing procedure for English-Japanese task contains three steps. Firstly, we tokenize English-side corpus, while Japanese sentences are separated in *character level*. We filter out all sentences that contain more than 50 tokens in either side. Secondly, we make vocabulary sets for both languages. The vocabulary size for English is limited to 200k, no limitation is applied to Japanese data. Out-of-vocabulary tokens are converted to "UNK". Finally, The sentences are sorted according to their lengths. We group them to mini-batches, each batch is composed of 64 bilingual sentences. The order of mini-batches is further randomly shuffled before use.

We adopt Adam optimizer (Kingma and Ba, 2014) to train our end-to-end models with an initial learning rate of 0.0001. Then training ends at the end of 6th epoch, and the learning rate halves at the beginning of last 3 epochs. We keep track of the cross-entropy loss of both training and validation data. Trained models are then evaluated with automatic evaluation metrics.

4.2 Evaluation Results

In Table 1, we show the empirical evaluation results of the fore-mentioned models. Enlarging the single-layer decoder RNN to 1400 hidden units boosted BLEU by 0.92%. Surprisingly, stacking another layer of decoder RNN degraded the performance in automatic evaluation. This is also confirmed by a significant slowdown of the decreasing of the validation loss. With the residual stacking, we get the best performance in all four model variations.

5 Submitted systems and results in WAT2016

5.1 Submitted systems

Our submitted systems for WAT2016 are based an ensemble of two same NMT models with different weight initialization. The decoder of the NMT models are composed by two-layer residual RNNs, which

Model	RIBES(%)	BLEU(%)
Baseline model	79.49	29.32
Enlarged decoder RNN	79.60	30.24
Stacked decoder RNN	79.25	29.07
Residual decoder RNN	**79.88**	**30.75**

Table 1: Automatic evaluation results in English-Japanese translation task on ASPEC corpus. Both baseline model and enlarged decoder RNN has a single-layer RNN with 1000 and 1400 hidden units respectively in decoder. Stacked decoder RNN and residual decoder RNN are both composed of two-layer RNNs with 1000 hidden units each.

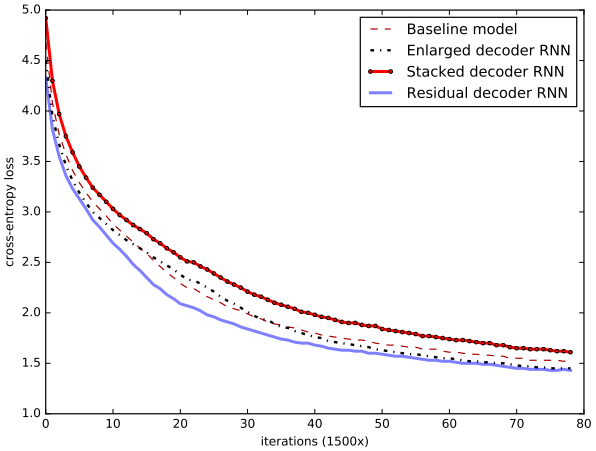

Figure 2: Validation loss (cross-entropy) of the first three epochs. The validation loss of the model with residual decoder RNNs is constantly lower than any other model variations, while stacking decoder RNNs naively slows down the training significantly.

is described in Section 3. *All RNNs in the network* are LSTM with 1200 hidden units. The pre-processing and training procedure is exactly the same as that in Section 4.2.

In order to test the performance with system combination, we trained a T2S transducer with Travatar (Neubig, 2013) on WAT2016 En-Ja dataset, but without forest inputs in the training phase. We used 9-gram language model with T2S decoding. We also evaluated the results of the T2S system with NMT reranking, where the reranking scores are given by the fore-mentioned NMT ensemble.

We tested performance of system combination in two approaches: GMBR system combination and a simple heuristics. The simple heuristics performs the system combination by choosing the result of NMT model ensemble when the input has fewer than 40 words, otherwise the result of reranked T2S system is chosen.

For GMBR system combination, we used 500 sentences from the development data for GMBR training. We obtain a 20-best list from each system, so our system combination task involves hypothesis selection out of 40 hypotheses. The loss function used for GMBR, the sub-components of the loss function are derived from n-gram precisions, brevity penalty, and Kendall's tau.

5.2 Official Evaluation Results in WAT2016

In Table 2, we show the automatic evaluation scores together with pairwise crowdsourcing evaluation scores for our submitted systems [1] in WAT2016. The first submission, which is a two-model ensemble of NMT models with residual decoder RNNs, achieved a comparably high RIBES (Isozaki et al., 2010) score.

For system combination, although we gained improvement by applying GMBR method[2], the naive method of system combination based on the length of input sentence works better in the evaluation with test data. Considering the scores of human evaluation, the system combination does make significant difference compared to the NMT model ensemble.

[1]We found a critical bug in our implementation after submitting the results. However, the implementations evaluated in Section 4.2 are bug-free. Hence, the results of NMT ensemble in Section 4.2 and the results of NMT model with residual decoder RNN in Section 5.2 are not comparable.

[2]We did not submit the result of GMBR system combination to human evaluation as the implementation is unfinished before the submission deadline.

Model	RIBES(%)	BLEU(%)	HUMAN
Online A	71.52	19.81	49.750
T2S system 1-best	77.87	32.32	-
T2S neural reranking	78.29	33.57	-
Ensemble of 2 NMT models with residual decoder RNNs (submission 1)	81.72	33.38	30.500
+ GMBR system combination with T2S results	80.90	34.25	-
+ System combination with a simple heuristics (submission 2)	81.44	34.77	29.750

Table 2: Official evaluation results of WAT2016

6 Related Works

Deep residual learning proposed in (He et al., 2015) learns a residual representation with a deep neural network. As stacking new layers does not lengthen the backprop path of early layers, residual learning enables the training of very deep networks, such as those with 1000 layers. Deep residual nets won the 1st place in ILSVRC 2015 classification task. The success of deep residual learning gives the insight of a better deep architecture of neural nets.

Beyond the success of residual learning, applying this technique to recurrent nets is a promising direction, which is researched in several previous works. Recurrent Highway Networks (Srivastava et al., 2015) enhance the LSTM by adding an extra residual computation in each step. The experiments show the Recurrent Highway Networks can achieve better perplexity in language modeling task with a limited parameter budget. (Liao and Poggio, 2016) achieved similar classification performance when using shared weights in a ResNet, which is exactly a RNN. Pixel Recurrent Neural Networks (van den Oord et al., 2016) demonstrates a novel architecture of neural nets with two-dimensional recurrent neural nets using residual connections. Their models achieved better log-likelihood on image generation tasks. Remarkably, the neural network architecture described in a lecture report [3] is similar to our models in spirit, where they applied stochastic residual learning to both depth and horizontal timesteps, which leads to better classification accuracy in Stanford Sentiment Treebank dataset.

7 Conclusion

In this paper, we demonstrates the effects of several possible approaches of enhancing the decoder RNN in NMT models. Surprisingly, Stacking multi-layer LSTMs in the decoder hinders the training and results in low performance in our experiments. Through empirical evaluation of several decoder architectures, we show that applying residual connections in the deep recurrent nets leads to superior results with same parameters as Stacked LSTMs. The advantage of the using of residual RNN in the decoder provides insights on the correct ways of enhancing NMT models.

8 Acknowledgment

We thank Weblio Inc. for financial support of this research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kevin Duh, Katsuhito Sudoh, Xianchao Wu, Hajime Tsukada, and Masaaki Nagata. 2011. Generalized minimum bayes risk system combination. In *IJCNLP*, pages 1356–1360.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

[3] https://cs224d.stanford.edu/reports/PradhanLongpre.pdf

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Qianli Liao and Tomaso Poggio. 2016. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*.

Thang Luong, Hieu Pham, and D. Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Toshiaki Nakazawa, Hideya Mino, Chenchen Ding, Isao Goto, Graham Neubig, Sadao Kurohashi, and Eiichiro Sumita. 2016a. Overview of the 3rd workshop on asian translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, Osaka, Japan, December.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016b. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208, Portoro, Slovenia, may. European Language Resources Association (ELRA).

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proceedings of the ACL Demonstration Track*, Sofia, Bulgaria, August.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *ICML*.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.