Multi-layer Representation Fusion for Neural Machine Translation

Qiang Wang[†], **Fuxue Li**[†,‡], **Tong Xiao**[†*], **Yanyang Li**[†], **Yinqiao Li**[†], **Jingbo Zhu**[†]
[†]Natural Language Processing Lab., Northeastern University
[‡]YingKou Institute of Technology
wangqiangneu@gmail.com, lifuxue119@163.com
xiaotong@mail.neu.edu.cn, blamedrlee@outlook.com
li.yin.qiao.2012@hotmail.com, zhujingbo@mail.neu.edu.cn

Abstract

Neural machine translation systems require a number of stacked layers for deep models. But the prediction depends on the sentence representation of the top-most layer with no access to low-level representations. This makes it more difficult to train the model and poses a risk of information loss to prediction. In this paper, we propose a multi-layer representation fusion (MLRF) approach to fusing stacked layers. In particular, we design three fusion functions to learn a better representation from the stack. Experimental results show that our approach yields improvements of 0.92 and 0.56 BLEU points over the strong Transformer baseline on IWSLT German-English and NIST Chinese-English MT tasks respectively. The result is new state-of-the-art in German-English translation.

1 Introduction

Neural models that use the encoder-decoder architecture to capture the translation equivalence relation between languages have been widely adopted over the last few years. The simplest of these relies on one recurrent neural network layer on both the encoder and decoder sides (Bahdanau et al., 2015), whereas others have successfully explored the high-level representation of language via deeper models (Wu et al., 2016; Gehring et al., 2017b; Vaswani et al., 2017). It has been noted that increasing the network depth is one of the factors contributing to the success of neural machine translation (NMT). To this end, one can stack a number of layers for an enriched sentence representation. E.g., popular NMT systems require 4 stacked layers or more for state-of-the-art results on large-scale translation tasks (Wu et al., 2016). Unfortunately, a straightforward implementation of deep neural networks has been found to underperform shallow models due to the poor convergence. One solution to this problem is to add identity mapping (or shortcut connection) between layers. A popular method is the residual network (He et al., 2016a), which has been used in several successful systems, e.g., GNMT and Transformer.

This model offers a good way to ease the information flow in the stack. But they do not make full use of the multi-level representations of the deep networks. E.g., word predictions depend on the sentence representation of the top-most layer with no access to low-level representations. Recent evidence supports that it helps when features from lower-level layers are involved in prediction (Xiong et al., 2017; Gehring et al., 2017b). A good instance is DenseNet that applies a fully-connected convolutional network to computer vision (Huang et al., 2017a). But it is still rare to see studies on this issue in machine translation.

In this paper, we take a further step along the line of this research. Drawing on previous successful attempts to create shortcut connections between adjacent layers, we propose a multi-layer representation fusion (MLRF) approach that connects one layer to all its predecessors. It learns a fused representation by accessing all lower-level representations of sentence, rather than learning from the limited information flow in one residual connection. The contributions of this work are three-fold.

- We propose an approach to learning better sentence representations by accessing lower-level layers. To our knowledge, it is the first time to use densely connected networks in NMT.

* Corresponding author
This work is licensed under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/

Proceedings of the 27th International Conference on Computational Linguistics, pages 3015–3026
Santa Fe, New Mexico, USA, August 20-26, 2018.

- We investigate methods to learn functions that fuse different levels of sentence representations. In particular, we propose to learn a 2D representation by the self-attention mechanism.
- We demonstrate the effectiveness of the approach on top of a strong system (Transformer) which has already used residual networks. It achieves a new state-of-the-art result on the IWSLT German-English MT task.

Our approach is applicable to arbitrary neural architectures and is easy to be implemented. In experiments on IWSLT German-English and NIST Chinese-English translation, it yields improvements of 0.92 and 0.56 BLEU points over the baseline respectively. More interestingly, we find that our approach has a regularizing effect and reduces the risk of over-fitting.

2 The Transformer System

In this work, all discussions and experiments are based on the Transformer system. We choose Transformer because it is one of the most successful NMT systems in recent MT evaluations. Unlike usual NMT models, Transformer does not require any recurrent units for modeling word sequences of arbitrary length. Instead, it resorts to self-attention and standard feed-forward networks for both encoder and decoder.

On the encoder side, there are L identical stacked layers. Each of them is composed of a self-attention sub-layer and a feed-forward sub-layer. The attention model used in Transformer is scaled dot-product attention[1]. Its output is fed into a fully connected feed-forward network. To ease training, the output of each sub-layer is defined as $\text{LayerNorm}(x + \text{sublayer}(x))$, where $\text{LayerNorm}(\cdot)$ is layer normalization (Ba et al., 2016) and $\text{sublayer}(x)$ is the output of the sub-layer. The identical mapping of input x represents the residual connection. To facilitate description, we use $H = \{h^1, \ldots, h^L\}$ to denote the outputs of source-side layers in this paper [2].

Likewise, the decoder has another stack of L identical layers (denoted as the function $\text{Layer}(\cdot)$). It has an encoder-decoder attention sub-layer in addition to the two sub-layers used in each encoder layer. Moreover, because the model is auto-regressive, the decoder attends a target position to all positions up to it. Let z_j^l be the output vector of target position j in the l-th layer, and $z_{\leq j}^{l-1}$ be the vector sequence $\{z_1^{l-1}, \ldots, z_j^{l-1}\}$. The output of the layer can be described as $z_j^l = \text{Layer}(z_{\leq j}^{l-1}, H)$.

The auto-regressive property also persists in the output layer. Given a source sentence $\mathbf{x} = (x_1, \ldots, x_M)$, and a target sentence $\mathbf{y} = (y_1, \ldots, y_N)$, the translation model is defined to be:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{N} \Pr(y_j|y_{<j}, \mathbf{x}) \tag{1}$$

where $\Pr(y_j|y_{<j}, \mathbf{x})$ is the probability of generating a target word y_j given the previously generated words $y_{<j}$ and the source sentence $\mathbf{x}$. To model $\Pr(y_j|y_{<j}, \mathbf{x})$, we have

$$\Pr(y_j|y_{<j}, \mathbf{x}) = \text{Softmax}(W_o \cdot z_j^L + b_o) \tag{2}$$

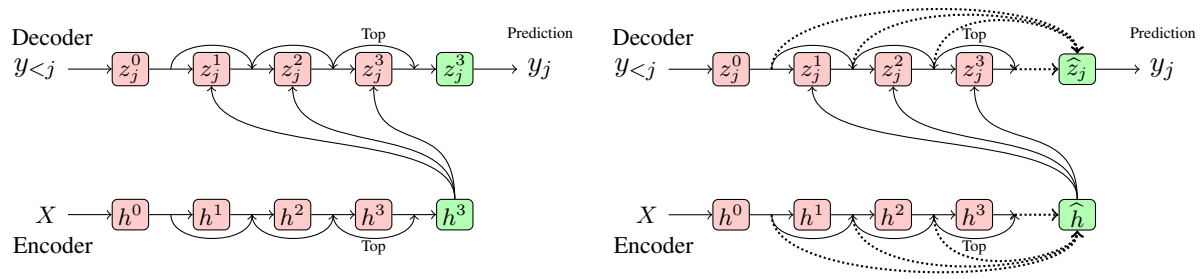
where W_o and b_o are the model parameters of the output layer. Obviously, the word prediction model here depends only on the output of the top-most layer in the stack (i.e., z_j^L).

3 The Approach

The model presented in Section 2 shows a standard way of word prediction where the output layer is connected to the top-most hidden layer only. This is fine if the system is composed of one hidden layer or two. But it may be problematic when the network goes deeper. First, this method has difficulties

[1] Given a sequence of vectors and a position i, the self-attention model computes the dot-product of the input vectors for each pair of positions (i, j), followed by a rescaling operation and Softmax. In this way, we have an attention score (or weight) for each (i, j). It is then used to generate the output by a weighted sum over all input vectors.

[2] We regard the word embedding layer as a special layer at the bottom of the stack, and denote it as h^0.



(a) Standard Architecture (b) The MLRF Architecture

Figure 1: Network Architectures. Dotted lines denote multi-layer fusion connections. Green rectangles denote the representations used in the downstream components.

in training deep models. To learn model parameters, gradients have to be propagated through a single channel of several stacked layers. Previous work has pointed out that it is not easy for lower-level layers to find good parameters due to the vanished or exploded gradients (Srivastava et al., 2015; He et al., 2016a). The situation is even worse when we use many non-linear transformations that are hard to learn. The residual connection between adjacent layers can alleviate the problem, but it is still a long distance for bottom layers to have direct feedbacks from the prediction. Second, there is potential information loss when we feed a single layer to the output layer. Top-level layers may forget previous features, especially when the stacked layers have the same capacity (e.g., the same size of layer output). For a simple solution, one can enlarge the layer size on the top. But it in turn drastically increases the number of parameters and is obviously not efficient for practical systems.

In this work, we propose a multi-layer representation fusion method to address these problems. It learns a fused representation of the sentence by direct access to all the layers in the stack. In this way, the prediction can benefit from the fused representation which has less information loss. As another bonus, this method makes it easier for the deep network to learn parameters because of the efficient information flow in connections from bottom to top.

3.1 Multi-layer Representation Fusion

The idea of representation fusion is pretty simple. We introduce a fusion layer into the network, and connect it to all the stacked layers. This layer can learn a refined representation from different levels of the stack. Let $Z = \{z^1, \ldots, z^L\}$ be the output sequence of the stacked layers on the decoder side. We define $\phi(Z)$ to be the fusion function that fuses $\{z^1, \ldots, z^L\}$ into a single representation. More specifically, given a target word position j, $\widehat{z}_j = \phi(Z_j)$ is defined to be the fused representation of all layer representations $Z_j = \{z_j^1, \ldots, z_j^L\}$ in the stack for position j. Then Eq. (2) can be rewritten as:

$$\Pr(y_j|y_{<j}, \mathbf{x}) = \text{Softmax}(W_o \cdot \phi(Z_j) + b_o) \tag{3}$$

There are many choices to design the fusion function $\phi(Z_j)$. E.g., we can cast the usual method as a special case of our model and do exactly the same thing as in Section 2, i.e., $\phi(Z_j) = z_j^L$. Alternatively, we can fuse more layers with another neural network. Moreover, we can do representation fusion on either the encoder side, the decoder side, or both. See Figure 1 for an illustration of the method.

Another note on representation fusion. The method is applicable to arbitrary neural network architectures with multiple levels of layers. Although we restrict ourselves to Transformer for our experiments, Eq. (3) actually shows a more general case. In a sense, our model is doing something similar to fully connected networks in computer vision (Huang et al., 2017a; Liu et al., 2018). Representation fusion is essentially a process that creates a densely connected network on top of the stack. In this work, we describe the problem in the more general framework, and will show that NMT systems can benefit from better design of fusion functions.

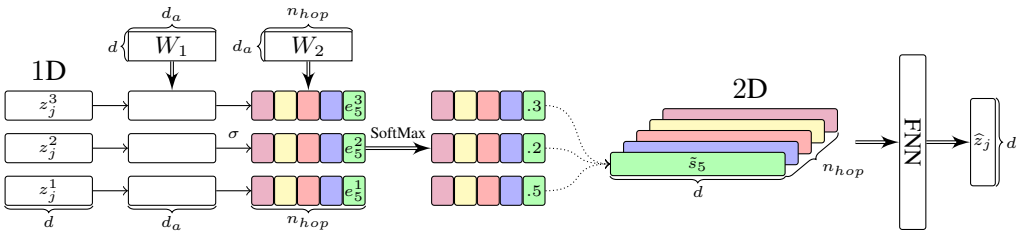


Figure 2: An example of 3-layer representation fusion based on self-attention with 5 hops.

3.2 Fusion Functions

Fusion by Pooling The simplest way of representation fusion is pooling. Here we choose average pooling (avg-pooling) because it shows better results than other pooling methods in our preliminary experiments. Given a sequence of layer outputs at position j, the output of average pooling-based fusion is $\phi(Z_j) = \frac{1}{L}\sum_{l=1}^{L} z_j^l$. In general, this way can be seen as the natural generalization of residual connections (He et al., 2016b; He et al., 2016a). It accumulates all layer features at once when going through the stack, rather than performing identity mapping between adjacent layers at each time. Also, this method is nonparametric and simply assigns an equal weight to each layer in fusion.

Fusion by Feed-forward Neural Network A more sophisticated method is to use feed-forward neural networks (FNNs) for fusion. To this end, we first concatenate all layer outputs to form a big vector. Then, we feed the vector into a single hidden layer FNN with d_f hidden units. Let d be the size of the layer output vector (i.e., $d = |z_j^l| \ \ \forall j, l$). The FNN transforms an $L * d$ dimensional vector to a d dimensional vector. This is good because we do not need to change the size of the downstream components. Moreover, we can learn a better fusion function by the non-linear activation functions used in FNNs. It should be noted that this method can be seen as a special case of dense connection, that is we only concatenate all the features of predecessors in the topmost layer rather than every intermediate layer. As a result, it can reduce the connection complexity from $\mathcal{O}(L^2)$ to $\mathcal{O}(L)$.

Fusion by Multi-hop Self Attention In addition to the pooling and FNN-based methods, we design another fusion function based on the self-attention model with hops [3] (Figure 2). The motivation is straightforward. Different layers reflect different aspects of sentence representation. An ideal way is to take correlations between layers into account, and model the correlation strength (or weight) in generating the fused result. This agrees with the nature of the attention mechanism well (Bahdanau et al., 2015). Beyond this, self-attention with hops can generate a 2D output that explains different meanings of the input vectors. Previous work has reported that 2D matrix-based representations outperform 1D vector-based counterparts in sentence embedding (Lin et al., 2017). So it is worth a study on this issue in NMT.

The idea is that we apply self-attention with multiple hops vertically over the layer sequence (along the axis of depth), and extract a part of layer information by each hop. The new problem that arises here is that we need to perceive the temporal order information in sequence (Shen et al., 2018). Unlike Vaswani et al. (2017)'s work, we do not resort to a fixed positional encoding generated by sine and cosine functions of different frequencies. Instead, we learn an embedding of the layer index, as used in positional embedding (Gehring et al., 2017a). The output of layer embedding is of size d, so it is compatible with the layer representation vector z_j^l. Also, we use a shared layer embedding for the encoder and the decoder for simple implementation.

More formally, let $E^l \in \mathbb{R}^d$ be the embedding of layer l. We inject layer embedding into the original representation of the layer, like this:

$$\tilde{z}^l = z^l + E^l \tag{4}$$

[3] In this work, the term "hop" denotes an independent attention computation, which has the same meaning as Lin et al. (2017). More specifically, one hop can generate a vector as the attention distribution over the layers. And different hops may generate the distinguished distributions.

Entry	Fusion Function	Low-level accessible	Parametric	2D represent.
Baseline	$\phi(Z_j) = z_j^L$	No	No	No
Avg-pooling	$\phi(Z_j) = \frac{1}{L}\sum_{l=1}^{L} z_j^l$	Yes	No	No
FNN-based	$\phi(Z_j) = \text{FNN}([z_j^1, ..., z_j^L])$	Yes	Yes	No
Self-att.-based	$\phi(Z_j) = \text{FNN}(\text{SelfAtt}([z_j^1, ..., z_j^L]))$	Yes	Yes	Yes

Table 1: Comparison of the fusion functions used in this work.

where $\tilde{z}^l \in \mathbb{R}^d$ is the new representation that encodes both the content and index information of layer l. Then the self-attention method is performed over $(\tilde{z}^1, ..., \tilde{z}^L)$. For hop p, the energy of attention weight is computed by an FNN with a single hidden layer,

$$e = W_2(\sigma(W_1 \cdot [\tilde{z}^1, ..., \tilde{z}^L]^T)) \tag{5}$$

where $[\tilde{z}^1, ..., \tilde{z}^L] \in \mathbb{R}^{d \times L}$ is the 2D layout of layer representations, $W_1 \in \mathbb{R}^{d \times d_a}$ and $W_2 \in \mathbb{R}^{d_a \times n_{hop}}$ are model parameters, σ is the activation function, and d_a is the size of inner hidden state. In this model, each hop has an independent parameter vector corresponding to one column of W_2, whereas W_1 is shared across different layers to make efficient use of memory [4]. After that, we obtain the attention weight vector $\mathrm{a}_p = (a_p^1, ..., a_p^L)$ of hop p by Softmax. For each layer index l, we have

$$a_p^l = \frac{\exp(e_p^l)}{\sum_{l'=1}^{L} \exp(e_p^{l'})} \tag{6}$$

The self-attention model generates an intermediate representation by dot-production ($\odot$) of the weights and the layer representations.

$$\tilde{s}_p = a_p \odot [\tilde{z}^1, ..., \tilde{z}^L] \tag{7}$$

In this way, all the hops compose a 2D matrix $\tilde{s} = [\tilde{s}_1, ..., \tilde{s}_{n_{hop}}] \in \mathbb{R}^{d \times n_{hop}}$. This matrix is fed into another single hidden layer FNN with d_f hidden units for the final output. It is worth noting that the model presented here can be enhanced by using an independent weight $a_{p;k}^l$ for each dimension k of $\tilde{z}^l$ (call it feature-wise fusion). But this method is computationally expensive. In our experiments we do not observe improvements by this method. Therefore, we choose the version described in Eqs. (6-7) for the empirical study.

Table 1 shows a comparison of the methods used in this work. For good convergence, we use layer normalization after the fusion layer in this work.

4 Experiments

We evaluate our proposed approach on German-English and Chinese-English translation tasks.

4.1 Setup

For German-English translation, we use the data of the IWSLT 2014 German-English track (Cettolo et al., 2014). We follow Ranzato et al. (2015)'s work for preprocessing. We use a joint source and target byte-pair encoding with 10k merge operations (Sennrich et al., 2016). The source and target vocabulary sizes are 8,389 and 6,428 respectively. We remove the sentences with more than 175 words or 100 sub-word units. This results in 160K sentence pairs for training. We randomly sample 7K sentences from the training data for held-out validation, and concatenate dev2010, dev2012, tst2010, tst2011, and tst2012 for test.

For Chinese-English translation, we use parts of the bitext provided within NIST12 OpenMT[5]. We choose NIST 2006 (MT06) as the validation set, and 2004 (MT04), 2005 (MT05), 2008 (MT08) as the

[4] W_1 can be unique for each layer. We discuss this issue in Section 4.4.

[5] LDC2000T46, LDC2000T47, LDC2000T50, LDC2003E14, LDC2005T10, LDC2002E18, LDC2007T09, LDC2004T08

System		#Param.	Valid.	Test
Existing Systems	RNN-MIXER (Ranzato et al., 2015)	-	-	20.73
	RNN-BSO (Wiseman and Rush, 2016)	-	-	26.36
	RNN-AC(Bahdanau et al., 2016)	-	-	28.53
	RNN-NPMT (Huang et al., 2017b)	-	-	28.96
	RNN-NPMT + LM (Huang et al., 2017b)	-	-	29.16
	ConvSeq2Seq-MLE (Edunov et al., 2017)	-	32.96	31.74
	ConvSeq2Seq-Risk (Edunov et al., 2017)	-	33.91	32.85
Baselines	Transformer-MLE	10.97M	33.58	31.75
	+RestartAdam	10.97M	34.14	32.67
	+RestartAdam-4Layers	12.82M	34.20	32.57
	+RestartAdam-6Layers	16.50M	34.35	32.97
MLRF Systems	Enc-AVG	10.97M	34.34	32.71
	Enc-FNN	11.63M	34.55	33.31
	Enc-SA (n_{hop}=4)	11.90M	34.48	33.06
	Enc-SA (n_{hop}=6)	12.16M	34.61	33.40
	Dec-AVG	10.97M	34.02	32.80
	Dec-FNN	11.63M	34.38	32.93
	Dec-SA (n_{hop}=4)	11.90M	**34.83**	33.29
	Dec-SA (n_{hop}=6)	13.47M	**34.73**	33.54
	Both-FNN	12.29M	34.30	32.66
	Both-SA ($n_{hop} = 4$)	12.82M	34.59	33.46
	Both-FNN-SA (n_{hop}=4)	12.55M	34.55	**33.59**

Table 2: BLEU scores [%] on IWSLT German-English translation.

test sets. All Chinese sentences are word segmented using the tool provided within NiuTrans (Xiao et al., 2012). All sentences of more than 50 words are removed. The resulting training corpus consists of 1.85M sentence pairs, with 39.42M Chinese words and 44.92M English words on each language side. We limit the vocabularies to the most frequent 30K words, covering approximately 98.16% and 99.17% of the Chinese and English words respectively. All out-of-vocabulary words are replaced with <UNK>. We report results without any UNK-replacement techniques (Luong et al., 2015).

4.2 Implementation Details

For German-English systems, the model consists of a 3-layer encoder and a 3-layer decoder with $d = 256$ and 1024 hidden units in the FNN sub-layer. For our approach, we set $d_a = 1024$ and $d_f = 512$. Dropout (rate$= 0.1$) is used for regularization. We initialize all word/sub-word embedding matrices using a normal distribution $\mathcal{N}(0, d^{-0.5})$, while initialize the layer embedding matrix using a uniform distribution $\mathcal{U}(-0.1, 0.1)$. The weight and bias in layer-normalization are initialized to constants 1 and 0. All other parameters are initialized from $\mathcal{U}(-\frac{1}{\sqrt{fan_{in}}}, \frac{1}{\sqrt{fan_{in}}})$, where fan_{in} is the input size of the parameter matrix. We use negative Maximum Likelihood Estimation (MLE) as loss function, and train all the models using Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. We run training for 40 epochs with a mini-batch of 80. The learning rate is scheduled as described in (Vaswani et al., 2017): $lr = d^{-0.5} \cdot min(t^{-0.5}, t \cdot 16\text{k}^{-1.5})$, where t is the step number. After that, we restart Adam and continue the training for additional 20 epochs with a fixed learning rate $5e^{-5}$ and a smaller mini-batch of 32 (Denkowski and Neubig, 2017). At test time, translations are generated by beam search with length normalization. By tuning on the validation set, we use a beam of width 8 and a length normalization weight of 1.6.

For Chinese-English systems, we use a 6-layer encoder and a 6-layer decoder, with $d = 512$ and 2048 hidden units in the FNN sub-layer. We restart Adam after 10 epochs and train the model for 5 additional epochs. A beam of width 12 and a length normalization weight of 1.3 are employed.

Note that the network equipped with our fusion layer increases a fraction of the computation cost than original one. E.g., in Chinese-English translation, the training speed reduces from 2.6 batches/second

System		Valid.	Test			
		MT06	MT04	MT05	MT08	Ave.
Open Source Systems	Nematus-4Layers	40.49	46.83	39.40	32.73	39.65
	NMTTutorial-GNMT	41.29	47.43	40.18	33.67	40.43
	T2T	41.87	47.42	41.09	34.39	40.97
Baselines	Transformer-MLE	40.60	47.70	40.33	34.70	40.91
	+RestartAdam	41.82	48.09	40.76	34.75	41.20
MLRF Systems	Enc-AVG	42.17	48.35	41.24	34.93	41.51
	Enc-FNN	40.81	47.44	40.03	33.55	40.34
	Enc-SA ($n_{hop} = 4$)	40.17	46.79	39.55	32.80	39.71
	Dec-AVG	41.43	48.38	41.19	34.66	41.49
	Dec-FNN	41.97	48.14	41.36	34.38	41.23
	Dec-SA ($n_{hop} = 4$)	42.13	48.48	**41.51**	34.79	41.59
	+ indep. W_1	**42.26**	**48.75**	41.44	**35.08**	**41.76**
	Both-AVG-SA ($n_{hop} = 4$)	41.61	47.65	40.69	34.52	40.95

Table 3: BLEU scores [%] on NIST Chinese-English translation.

(baseline) to 2.4 batches/second (Dec-SA, ref. row 11 in Table 3).

4.3 Results on German-English Translation

Table 2 shows the BLEU scores of various NMT systems[6]. For comparison, we list previous results on the same data set. First of all, our baseline is good. The base setting of our baseline system can lead to a similar score to ConvSeq2Seq which is trained for more epochs (Edunov et al., 2017). The baseline is stronger when Adam restart is adopted. It outperforms most of previous systems on this task (only lower than the best reported system by 0.2 BLEU points).

Then, we test different fusion functions (AVG, FNN and SA) on the encoder side (Enc), the decoder side (Dec) and both of them (Both). We see that avg-pooling does not yield promising improvements due to the relatively low expressive power. In addition, the representation fusion on the decoder side is more effective than that on the encoder side. A possible reason is that the prediction can benefit more from the lower-level layers in the decoder due to the "shorter" distance from fusion to prediction. This agrees with the result that representation fusion on both sides (Both-SA) does not improve Enc-SA and Dec-SA significantly. The best result is achieved when we use FNN-based fusion on the encoder side and self-attention-based fusion on the decoder side (Both-FNN-SA). It outperforms the 3-layer baseline by 0.92 BLEU points. Also, we increase the number of layers to 4 and 6 to examine whether the improvement comes from the additional model parameters introduced by the added fusion layer. It results in stronger baselines, but they still underperform the Both-FNN-SA system which has fewer parameters.

4.4 Results on Chinese-English Translation

For Chinese-English translation, we compare our systems with three open source systems: Nematus (Sennrich et al., 2017), NMTTutorial (Luong et al., 2017) and T2T [7]. The first two are based on RNN, while the last one is based on Transformer. Table 3 shows that our baseline with restarting Adam is a bit better than T2T. Like in German-English translation, fusion on the decoder side is superior than that on the encoder side. But both Enc-FNN and Enc-SA are worse than the baseline. We suspect that adding more layers with non-linear transformations makes it more difficult to train the deep network (6 layers in Chinese-English translation) than the shallow counterpart (3 layers in German-English translation). This problem is more evident here because the Chinese-English systems have many more parameters than the German-English systems and are more difficult to optimize. Interestingly, we see that the simple avg-pooling fusion method works well in both encoder and decoder. In addition, using independent W_1 (in

[6]All BLEU results are case-insensitive. For a fair comparison with previous work, we run *multi-bleu.perl* for German-English translation, and run *mteval-v13a.pl* for Chinese-English translation.

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

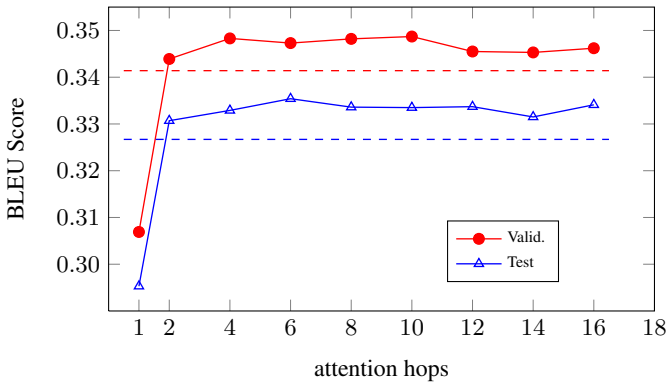


Figure 3: BLEU scores under various attention hops in Dec-SA on German-English translation. Dashed lines denotes the baseline.

System	Word Emb.	Layer Emb.	Test
Enc-FNN	√	√	33.31
	√		33.17
		√	33.07*
Dec-SA	√	√	33.29
	√		33.06*
		√	33.21
Both-FNN-SA	√	√	33.59
	√		33.55
		√	33.16*

Table 4: BLEU scores on German-English translation when fusing with (denoted by √)/without word embedding and layer embedding in encoder, decoder, and both of them. * denotes significant performance reduction (> 0.2 BLEU) than baseline (with both word embedding and layer embedding).

self-attention) for each layer is helpful for better fitting. It indicates that the MLRF method can benefit from more sophisticated design of the self-attention model.

4.5 Effect on Attention Hops

Figure 3 shows the BLEU curves by varying n_{hop} on German-English translation with the Dec-SA approach. Clearly, increasing the number of hops improves the system. The improvement is significant when $n_{hop} < 6$ thanks to the 2D representations extracted by multiple hops. Note that when $n_{hop} = 1$, the 2D sentence representation is degraded into the 1D representation. It results in a dramatic decrease in BLEU. However, larger n_{hop} does not make further improvements due to the redundant information in hops and potential overlaps between them.

4.6 Importance of Word Embedding and Layer Embedding

Then, we study the model behavior with/without word embedding or layer embedding on German-English translation. Table 4 shows that removing either word embedding or layer embedding harms the Enc-FNN system. The impact of word embedding is a bit more than that of layer embedding. As a contrast, Dec-SA is more sensitive to the remove of layer embedding. We attribute it to the unawareness of the temporal order information in the self-attention mechanism. More interestingly, when we fuse representations on both sides, almost no BLEU reduction is observed in Both-FNN-SA. It is true even if layer embedding is removed. This result indicates that the layer indexing information can be delivered from the FNN fusion layer in the encoder to the decoder in an implicit way. Also, a significant improvement can be achieved by fusing with the original word embedding features. It agrees with the result reported in (Xiong et al., 2017).

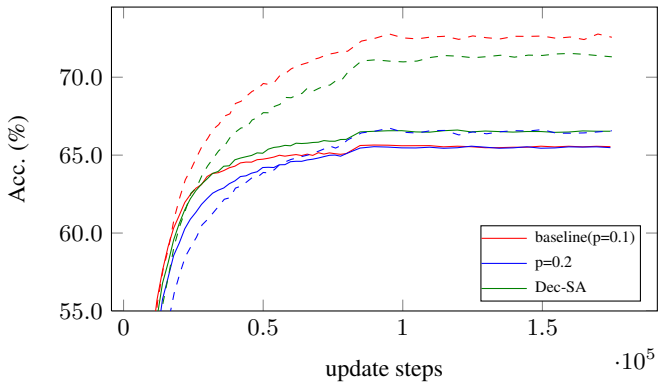
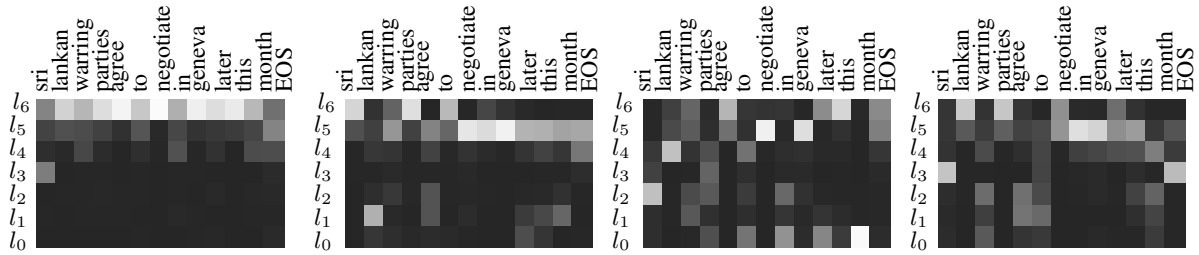


Figure 4: Curve of Accuracy in training-set (dashed lines) and validation-set (solid lines) along with update steps. p is dropout rate.



(a) Hop 1 (b) Hop 2 (c) Hop 3 (d) Hop 4

Figure 5: A visualization example of Dec-SA with 4 hops in Chinese-English translation. Colors change between white (probability of 1) to black (probability of 0). The source sentence (pinyin) is *Sīlǐlánkǎ jiāozhàn shuāngfāng tóngyì běnyuè xiàxún zài rìnèiwǎ tánpàn*.

4.7 Regularization

Also, we plot the curves of accuracy on the training and validation sets in Figure 4 (German-English translation). It is obvious that the baseline with dropout rate $p = 0.1$ (red) has a significant lower BLEU score when we switch from training to validation, implying that the model somehow over-fits the data. When we use a larger $p = 0.2$ (blue) for dropout, the accuracy on the training set decreases dramatically, but no promising improvement is observed on the validation set. On the contrary, Dec-SA seems to have a better regularization effect. Although Dec-SA has lower accuracy in training than the baseline, it outperforms the baseline on the validation set. It might be due to the introduced direct connections from lower-level layers to the top of the stack. These connections make the model easier to fit the data because they have no dependence on the non-linear transformations in the stacked layers.

4.8 Layer Attention Visibility

A visualization example of layer attention with multiple hops is presented in Figure 5. The attention result is generated from an example of Chinese-English translation by Dec-SA with 4 hops. The x-axis is the target word sequence, and the y-axis is the indexing of layers (including the word embedding layer denoted by l_0). We can see that most attention weights focus on the high-level layers like hops 1 and 2, while hops 3 and 4 have more dispersive attentions over different layers. It indicates the different aspects of the sentence encoded in different hops. An interesting finding is that those large probability points in the lower-level layers are easier to appear when a noun or pronoun is fed into the decoder. For example, when we feed *sri* into the decoder to generate *lankan*, the first layer is obviously more important in hop 2. Similar cases can be observed for words *geneva* and *this* in hop 3, where the word embedding layer has a larger weight. This is reasonable because most of these words have specific meanings and do not need high-level representations for modeling large context in disambiguation.

5 Related Work

Training neural networks with multiple stacked layers is challenging. It has been observed that introducing direct connections between layers can drastically improve the performance of deep neural models. Methods include highway networks (Srivastava et al., 2015), residual connections (He et al., 2016a), dense connections (Huang et al., 2017a), and fast-forward connections (Zhou et al., 2016). E.g., in machine translation, residual networks have been a popular way to address the issue due to its simplicity (Wu et al., 2016; Gehring et al., 2017a; Gehring et al., 2017b; Vaswani et al., 2017). Another related study is Wang et al. (2017). They introduce linear associative units to reduce the length of gradient propagation in recurrent neural networks (RNNs), and demonstrate promising improvements on their RNN-based NMT systems. But previous studies all focus on using the top-level sentence representation for prediction, and ignore the access to the representations encoded in lower-level layers.

The next obvious step is toward models that make full use of all stacked layers for prediction (call it representation fusion). Some research groups have been aware of this and explored solutions. E.g., Gehring et al. (2017b) find that NMT systems can benefit from shortcut connections from the source word embedding layer to the attention layer. Perhaps the most related work is Xiong et al. (2017). They propose a multi-channel encoder (MCE) which uses an external memory module (Graves et al., 2014) to compose word embeddings and hidden states in their RNN-based encoder. But this model is applied to the shallow network on the encoder side. In this work we instead propose a more general method to do representation fusion in either the encoder, the decoder, or both. In addition, we present a 2D representation of sentence which has not been well studied in machine translation.

6 Conclusion

We have proposed a multi-layer representation fusion approach that densely connects all the stacked layers to a fusion layer for encoder or/and decoder in NMT. Also, we have developed three fusion functions to learn a better representation of sentence. Experimental results on German-English and Chinese-English translation show that our method outperforms the strong Transformer baseline significantly, and achieves new state-of-the-art on the IWSLT German-English MT task. More interestingly, it is observed that our approach has a regularizing effect and reduces the risk of over-fitting.

Acknowledgements

This work was supported in part by the National Science Foundation of China (No. 61672138, 61432013 and 61572120), the Fundamental Research Funds for the Central Universities. The authors would like to thank anonymous reviewers and Chunliang Zhang for their comments.

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. CoRR, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In In Proceedings of the 3rd International Conference on Learning Representations.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. arXiv e-prints, abs/1607.07086, July.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In Proceedings of the First Workshop on Neural Machine Translation, pages 18–27. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. arXiv preprint arXiv:1711.04956.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017a. A convolutional encoder model for neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 123–135. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017b. Convolutional Sequence to Sequence Learning. ArXiv e-prints, May.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. arXiv preprint arXiv:1410.5401.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In European Conference on Computer Vision, pages 630–645. Springer.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017a. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, volume 1, page 3.

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017b. Neural phrase-based machine translation. arXiv preprint arXiv:1706.05565.

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

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding.

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. 2018. Path aggregation network for instance segmentation. arXiv preprint arXiv:1803.01534.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 11–19. Association for Computational Linguistics.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. https://github.com/tensorflow/nmt.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. arXiv preprint arXiv:1511.06732.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics, pages 65–68, Valencia, Spain, April. Association for Computational Linguistics.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In AAAI Conference on Artificial Intelligence.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2377–2385. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 136–145.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 1296–1306.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. Niutrans: an open source toolkit for phrase-based and syntax-based machine translation. In Proceedings of the ACL 2012 System Demonstrations, pages 19–24. Association for Computational Linguistics.

Hao Xiong, Zhongjun He, Xiaoguang Hu, and Hua Wu. 2017. Multi-channel encoder for neural machine translation. arXiv preprint arXiv:1712.02109.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. Transactions of the Association of Computational Linguistics, 4:371–383.