Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation

Jie Zhou Ying Cao Xuguang Wang Peng Li Wei Xu
Baidu Research - Institute of Deep Learning
Baidu Inc., Beijing, China
{zhoujie01,caoying03,wangxuguang,lipeng17,wei.xu}@baidu.com

Abstract

Neural machine translation (NMT) aims at solving machine translation (MT) problems using neural networks and has exhibited promising results in recent years. However, most of the existing NMT models are shallow and there is still a performance gap between a single NMT model and the best conventional MT system. In this work, we introduce a new type of linear connections, named fast-forward connections, based on deep Long Short-Term Memory (LSTM) networks, and an interleaved bi-directional architecture for stacking the LSTM layers. Fast-forward connections play an essential role in propagating the gradients and building a deep topology of depth 16. On the WMT'14 English-to-French task, we achieve BLEU=37.7 with a single attention model, which outperforms the corresponding single shallow model by 6.2 BLEU points. This is the first time that a single NMT model achieves state-of-the-art performance and outperforms the best conventional model by 0.7 BLEU points. We can still achieve BLEU=36.3 even without using an attention mechanism. After special handling of unknown words and model ensembling, we obtain the best score reported to date on this task with BLEU=40.4. Our models are also validated on the more difficult WMT'14 English-to-German task.

1 Introduction

Neural machine translation (NMT) has attracted a lot of interest in solving the machine translation (MT) problem in recent years (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). Unlike conventional statistical machine translation (SMT) systems (Koehn et al., 2003; Durrani et al., 2014) which consist of multiple separately tuned components, NMT models encode the source sequence into continuous representation space and generate the target sequence in an end-to-end fashon. Moreover, NMT models can also be easily adapted to other tasks such as dialog systems (Vinyals and Le, 2015), question answering systems (Yu et al., 2015) and image caption generation (Mao et al., 2015).

In general, there are two types of NMT topologies: the encoder-decoder network (Sutskever et al., 2014) and the attention network (Bahdanau et al., 2015). The encoder-decoder network represents the source sequence with a fixed dimensional vector and the target sequence is generated from this vector word by word. The attention network uses the representations from all time steps of the input sequence to build a detailed relationship between the target words and the input words. Recent results show that the systems based on these models can achieve similar performance to conventional SMT systems (Luong et al., 2015; Jean et al., 2015).

However, a single neural model of either of the above types has not been competitive with the best conventional system (Durrani et al., 2014) when evaluated on the WMT'14 English-to-French task. The best BLEU score from a single model with six layers is only 31.5 (Luong et al., 2015) while the conventional method of (Durrani et al., 2014) achieves 37.0.

We focus on improving the single model perfor-

Transactions of the Association for Computational Linguistics, vol. 4, pp. 371–383, 2016. Action Editor: Holger Schwenk.
Submission batch: 1/2016; Revision batch: 4/2016; Published 7/2016.

mance by increasing the model depth. Deep topology has been proven to outperform the shallow architecture in computer vision. In the past two years the top positions of the ImageNet contest have always been occupied by systems with tens or even hundreds of layers (Szegedy et al., 2015; He et al., 2016). But in NMT, the biggest depth used successfully is only six (Luong et al., 2015). We attribute this problem to the properties of the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) which is widely used in NMT. In the LSTM, there are more non-linear activations than in convolution layers. These activations significantly decrease the magnitude of the gradient in the deep topology, especially when the gradient propagates in recurrent form. There are also many efforts to increase the depth of the LSTM such as the work by Kalchbrenner et al. (2016), where the shortcuts do not avoid the nonlinear and recurrent computation.

In this work, we introduce a new type of linear connections for multi-layer recurrent networks. These connections, which are called fast-forward connections, play an essential role in building a deep topology with depth of 16. In addition, we introduce an interleaved bi-directional architecture to stack LSTM layers in the encoder. This topology can be used for both the encoder-decoder network and the attention network. On the WMT'14 English-to-French task, this is the deepest NMT topology that has ever been investigated. With our deep attention model, the BLEU score can be improved to 37.7 outperforming the shallow model which has six layers (Luong et al., 2015) by 6.2 BLEU points. This is also the first time on this task that a single NMT model achieves state-of-the-art performance and outperforms the best conventional SMT system (Durrani et al., 2014) with an improvement of 0.7. Even without using the attention mechanism, we can still achieve 36.3 with a single model. After model ensembling and unknown word processing, the BLEU score can be further improved to 40.4. When evaluated on the subset of the test corpus without unknown words, our model achieves 41.4. As a reference, previous work showed that oracle re-scoring of the 1000-best sequences generated by the SMT model can achieve the BLEU score of about 45 (Sutskever et al., 2014). Our models are also validated on the more difficult WMT'14 English-to-German task.

2 Neural Machine Translation

Neural machine translation aims at generating the target word sequence $\boldsymbol{y} = \{y_1, \ldots, y_n\}$ given the source word sequence $\boldsymbol{x} = \{x_1, \ldots, x_m\}$ with neural models. In this task, the likelihood $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$ of the target sequence will be maximized (Forcada and Ñeco, 1997) with parameter $\boldsymbol{\theta}$ to learn:

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) = \prod_{j=1}^{m+1} p(y_j \mid \boldsymbol{y}_{0:j-1}, \boldsymbol{x}; \boldsymbol{\theta}) \qquad (1)$$

where $\boldsymbol{y}_{0:j-1}$ is the sub sequence from y_0 to y_{j-1}. y_0 and y_{m+1} denote the start mark and end mark of target sequence respectively.

The process can be explicitly split into an encoding part, a decoding part and the interface between these two parts. In the encoding part, the source sequence is processed and transformed into a group of vectors $\boldsymbol{e} = \{e_1, \cdots, e_m\}$ for each time step. Further operations will be used at the interface part to extract the final representation $\boldsymbol{c}$ of the source sequence from $\boldsymbol{e}$. At the decoding step, the target sequence is generated from the representation $\boldsymbol{c}$.

Recently, there have been two types of NMT models which are different in the interface part. In the encoder-decoder model (Sutskever et al., 2014), a single vector extracted from $\boldsymbol{e}$ is used as the representation. In the attention model (Bahdanau et al., 2015), $\boldsymbol{c}$ is dynamically obtained according to the relationship between the target sequence and the source sequence.

The recurrent neural network (RNN), or its specific form the LSTM, is generally used as the basic unit of the encoding and decoding part. However, the topology of most of the existing models is shallow. In the attention network, the encoding part and the decoding part have only one LSTM layer respectively. In the encoder-decoder network, researchers have used at most six LSTM layers (Luong et al., 2015). Because machine translation is a difficult problem, we believe more complex encoding and decoding architecture is needed for modeling the relationship between the source sequence and the target sequence. In this work, we focus on enhancing the complexity of the encoding/decoding architecture by increasing the model depth.

Deep neural models have been studied in a wide range of problems. In computer vision, models with more than ten convolution layers outperform shallow ones on a series of image tasks in recent years (Srivastava et al., 2015; He et al., 2016; Szegedy et al., 2015). Different kinds of shortcut connections are proposed to decrease the length of the gradient propagation path. Training networks based on LSTM layers, which are widely used in language problems, is a much more challenging task. Because of the existence of many more nonlinear activations and the recurrent computation, gradient values are not stable and are generally smaller. Following the same spirit for convolutional networks, a lot of effort has also been spent on training deep LSTM networks. Yao et al. (2015) introduced depth-gated shortcuts, connecting LSTM cells at adjacent layers, to provide a fast way to propagate the gradients. They validated the modification of these shortcuts on an MT task and a language modeling task. However, the best score was obtained using models with three layers. Similarly, Kalchbrenner et al. (2016) proposed a two dimensional structure for the LSTM. Their structure decreases the number of nonlinear activations and path length. However, the gradient propagation still relies on the recurrent computation. The investigations were also made on question-answering to encode the questions, where at most two LSTM layers were stacked (Hermann et al., 2015).

Based on the above considerations, we propose new connections to facilitate gradient propagation in the following section.

3 Deep Topology

We build the deep LSTM network with the new proposed linear connections. The shortest paths through the proposed connections do not include any nonlinear transformations and do not rely on any recurrent computation. We call these connections fast-forward connections. Within the deep topology, we also introduce an interleaved bi-directional architecture to stack the LSTM layers.

3.1 Network

Our entire deep neural network is shown in Fig. 2. This topology can be divided into three parts: the encoder part (P-E) on the left, the decoder part (P-D) on the right and the interface between these two parts (P-I) which extracts the representation of the source sequence. We have two instantiations of this topology: Deep-ED and Deep-Att, which correspond to the extension of the encoder-decoder network and the attention network respectively. Our main innovation is the novel scheme for connecting adjacent recurrent layers. We will start with the basic RNN model for the sake of clarity.

Recurrent layer: When an input sequence $\{x_1, \ldots, x_m\}$ is given to a recurrent layer, the output h_t at each time step t can be computed as (see Fig. 1 (a))

$$\begin{aligned} h_t &= \sigma(W_f x_t + W_r h_{t-1}) \\ &= \mathrm{RNN}\ (W_f x_t, h_{t-1}) \\ &= \mathrm{RNN}\ (f_t, h_{t-1}), \end{aligned} \tag{2}$$

where the bias parameter is not included for simplicity. We use a red circle and a blue empty square to denote an input and a hidden state. A blue square with a "-" denotes the previous hidden state. A dotted line means that the hidden state is used recurrently. This computation can be equivalently split into two consecutive steps:

- Feed-Forward computation: $f_t = W_f x_t$. Left part in Fig. 1 (b). "f" block.
- Recurrent computation: RNN (f_t, h_{t-1}). Right part and the sum operation (+) followed by activation in Fig. 1 (b). "r" block.

For a deep topology with stacked recurrent layers, the input of each block "f" at recurrent layer k (denoted by f^k) is usually the output of block "r" at its previous recurrent layer $k-1$ (denoted by h^{k-1}). In our work, we add **fast-forward connections** (F-F connections) which connect two feed-forward computation blocks "f" of adjacent recurrent layers. It means that each block "f" at recurrent layer k takes both the outputs of block "f" and block "r" at its previous layer as input (Fig. 1 (c)). F-F connections are denoted by dashed red lines in Fig. 1 (c) and Fig. 2. The path of F-F connections contains neither nonlinear activations nor recurrent computation. It provides a fast path for information to propagate, so we call this path fast-forward connections.

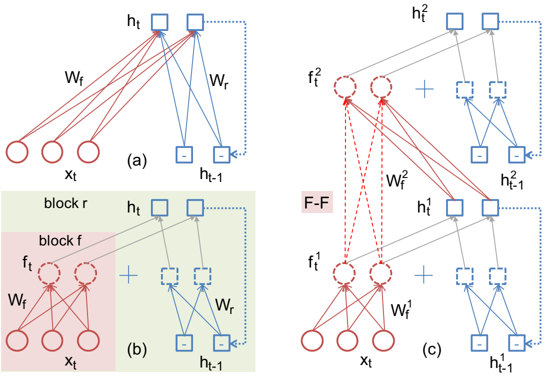

Figure 1: RNN models. The recurrent use of a hidden state is denoted by dotted lines. A "-" mark denotes the hidden value of the previous time step. (a): Basic RNN. (b): Basic RNN with intermediate computational state and the sum operation (+) followed by activation. It consists of block "f" and block "r", and is equivalent to (a). (c):Two stacked RNN layers with F-F connections denoted by dashed red lines.

Additionally, in order to learn more temporal dependencies, the sequences can be processed in different directions at each pair of adjacent recurrent layers. This is quantitatively expressed in Eq. 3:

$$\begin{aligned} f_t^k &= W_f^k \cdot [f_t^{k-1}, h_t^{k-1}], \quad k > 1 \\ f_t^k &= W_f^k x_t \qquad\qquad\qquad k = 1 \\ h_t^k &= \text{RNN}^k\left(f_t^k, h_{t+(-1)^k}^k\right) \end{aligned} \tag{3}$$

The opposite directions are marked by the direction term $(-1)^k$. At the first recurrent layer, the block "f" takes x_t as the input. [,] denotes the concatenation of vectors. This is shown in Fig. 1 (c). The two changes are summarized here:

- We add a connection between f_t^k and f_t^{k-1}. Without f_t^{k-1}, our model will be reduced to the traditional stacked model.

- We alternate the RNN direction at different layers k with the direction term $(-1)^k$. If we fix the direction term to -1, all layers work in the forward direction.

LSTM layer: In our experiments, instead of an RNN, a specific type of recurrent layer called LSTM (Hochreiter and Schmidhuber, 1997; Graves et al., 2009) is used. The LSTM is structurally more complex than the basic RNN in Eq. 2. We define the computation of the LSTM as a function which maps the input f and its state-output pair (h, s) at the previous time step to the current state-output pair. The exact computations for $(h_t, s_t) = \text{LSTM}(f_t, h_{t-1}, s_{t-1})$ are the following:

$$\begin{aligned} [z, z_\rho, z_\phi, z_\pi] &= f_t + W_r h_{t-1} \\ s_t &= \sigma_i(z) \circ \sigma_g(z_\rho + s_{t-1} \circ \theta_\rho) + \\ &\quad\ \sigma_g(z_\phi + s_{t-1} \circ \theta_\phi) \circ s_{t-1} \\ h_t &= \sigma_o(s_t) \circ \sigma_g(z_\pi + s_t \circ \theta_\pi) \end{aligned} \tag{4}$$

where $[z, z_\rho, z_\phi, z_\pi]$ is the concatenation of four vectors of equal size, $\circ$ means element-wise multiplication, σ_i is the input activation function, σ_o is the output activation function, σ_g is the activation function for gates, and W_r, θ_ρ, θ_ϕ, and θ_π are the parameters of the LSTM. It is slightly different from the standard notation in that we do not have a matrix to multiply with the input f in our notation.

With this notation, we can write down the computations for our deep bi-directional LSTM model with F-F connections:

$$\begin{aligned} f_t^k &= W_f^k \cdot [f_t^{k-1}, h_t^{k-1}], \quad k > 1 \\ f_t^k &= W_f^k x_t, \qquad\qquad\qquad k = 1 \\ (h_t^k, s_t^k) &= \text{LSTM}^k\left(f_t^k, h_{t+(-1)^k}^k, s_{t+(-1)^k}^k\right) \end{aligned} \tag{5}$$

where x_t is the input to the deep bi-directional LSTM model. For the encoder, x_t is the embedding of the t^{th} word in the source sentence. For the decoder x_t is the concatenation of the embedding of the t^{th} word in the target sentence and the encoder representation for step t.

In our final model two additional operations are used with Eq. 5, which is shown in Eq. 6. $\text{Half}(f)$ denotes the first half of the elements of f, and $\text{Dr}(h)$ is the dropout operation (Hinton et al., 2012) which randomly sets an element of h to zero with a certain probability. The use of $\text{Half}(\cdot)$ is to reduce the parameter size and does not affect the performance. We observed noticeable performance degradation when using only the first third of the elements of "f".

$$f_t^k = W_f^k \cdot [\text{Half}(f_t^{k-1}), \text{Dr}(h_t^{k-1})], \quad k > 1 \tag{6}$$

With the F-F connections, we build a fast channel to propagate the gradients in the deep topology. F-F connections can accelerate the model convergence and while improving the performance. A similar idea was also used in (He et al., 2016; Zhou and Xu, 2015).

Encoder: The LSTM layers are stacked following Eq. 5. We call this type of encoder **interleaved bi-directional encoder**. In addition, there are two similar columns (a_1 and a_2) in the encoder part. Each column consists of n_e stacked LSTM layers. There is no connection between the two columns. The first layers of the two columns process the word representations of the source sequence in different directions. At the last LSTM layers, there are two groups of vectors representing the source sequence. The group size is the same as the length of the input sequence.

Interface: Prior encoder-decoder models and attention models are different in their method of extracting the representations of the source sequences. In our work, as a consequence of the introduced F-F connections, we have 4 output vectors ($h_t^{n_e}$ and $f_t^{n_e}$ of both columns). The representations are modified for both Deep-ED and Deep-Att.

For Deep-ED, e_t is static and consists of four parts. 1: The last time step output $h_m^{n_e}$ of the first column. 2: Max-operation $\mathrm{Max}(\cdot)$ of $h_t^{n_e}$ at all time steps of the second column, denoted by $\mathrm{Max}(h_t^{n_e,a_2})$. $\mathrm{Max}(\cdot)$ denotes obtaining the maximal value for each dimension over t. 3: $\mathrm{Max}(f_t^{n_e,a_1})$. 4: $\mathrm{Max}(f_t^{n_e,a_2})$. The max-operation and last time step state extraction provide complimentary information but do not affect the performance much. e_t is used as the final representation c_t.

For Deep-Att, we do not need the above two operations. We only concatenate the 4 output vectors at each time step to obtain e_t, and a soft attention mechanism (Bahdanau et al., 2015) is used to calculate the final representation c_t from e_t. e_t is summarized as:

$$\begin{aligned}
&\text{Deep-ED: } e_t \\
&\quad [h_m^{n_e,a_1}, \mathrm{Max}(h_t^{n_e,a_2}), \mathrm{Max}(f_t^{n_e,a_1}), \mathrm{Max}(f_t^{n_e,a_2})] \\
&\text{Deep-Att: } e_t \\
&\quad [h_t^{n_e,a_1}, h_t^{n_e,a_2}, f_t^{n_e,a_1}, f_t^{n_e,a_2}] \qquad (7)
\end{aligned}$$

Note that the vector dimensionality of f is four times larger than that of h (see Eq. 4). c_t is summarized as:

$$\begin{aligned}
\text{Deep-ED: } & c_t = e_t, \quad (\text{const}) \\
\text{Deep-Att: } & c_t = \sum_{t'=1}^{m} \alpha_{t,t'} W_p e_{t'} \qquad (8)
\end{aligned}$$

$\alpha_{t,t'}$ is the normalized attention weight computed by:

$$\alpha_{t,t'} = \frac{\exp(a(W_p e_{t'}, h_{t-1}^{1,dec}))}{\sum_{t''} \exp(a(W_p e_{t''}, h_{t-1}^{1,dec}))} \qquad (9)$$

$h_{t-1}^{1,dec}$ is the first hidden layer output in the decoding part. $a(\cdot)$ is an alignment model described in (Bahdanau et al., 2015). For Deep-Att, in order to reduce the memory cost, we linearly project (with W_p) the concatenated vector e_t to a vector with $1/4$ dimension size, denoted by the (fully connected) block "fc" in Fig. 2.

Decoder: The decoder follows Eq. 5 and Eq. 6 with fixed direction term -1. At the first layer, we use the following x_t:

$$x_t = [c_t, y_{t-1}] \qquad (10)$$

y_{t-1} is the target word embedding at the previous time step and y_0 is zero. There is a single column of n_d stacked LSTM layers. We also use the F-F connections like those in the encoder and all layers are in the forward direction. Note that at the last LSTM layer, we only use h_t to make the prediction with a softmax layer.

Although the network is deep, the training technique is straightforward. We will describe this in the next part.

3.2 Training technique

We take the parallel data as the only input without using any monolingual data for either word representation pre-training or language modeling. Because of the deep bi-directional structure, we do not need to reverse the sequence order as Sutskever et al. (2014).

The deep topology brings difficulties for the model training, especially when first order methods such as stochastic gradient descent (SGD) (LeCun et al., 1998) are used. The parameters should be properly initialized and the converging process can

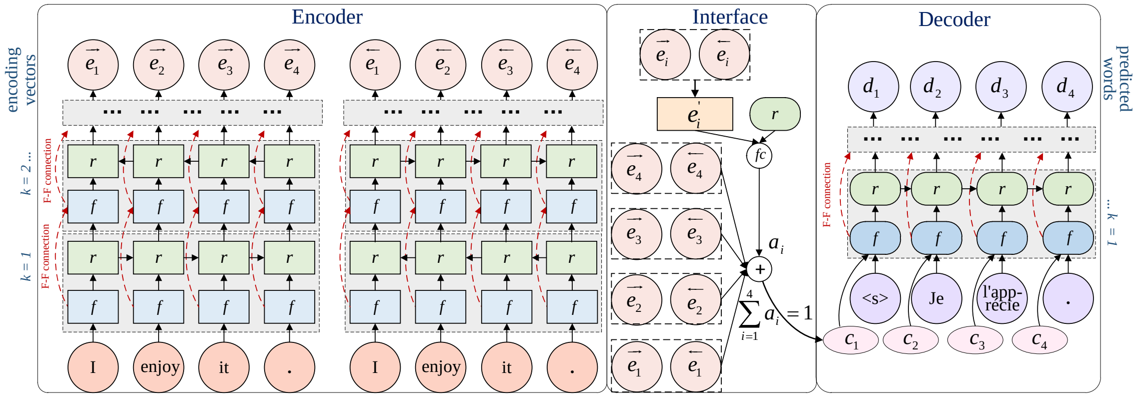

Figure 2: The network. It includes three parts from left to right: encoder part (P-E), interface (P-I) and decoder part (P-D). We only show the topology of Deep-Att as an example. "f" and "r" blocks correspond to the feed-forward part and the subsequent LSTM computation. The F-F connections are denoted by dashed red lines.

be slow. We tried several optimization techniques such as AdaDelta (Zeiler, 2012), RMSProp (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2015). We found that all of them were able to speed up the process a lot compared to simple SGD while no significant performance difference was observed among them. In this work, we chose Adam for model training and do not present a detailed comparison with other optimization methods.

Dropout (Hinton et al., 2012) is also used to avoid over-fitting. It is utilized on the LSTM nodes h_t^k (See Eq. 5) with a ratio of p_d for both the encoder and decoder.

During the whole model training process, we keep all hyper parameters fixed without any intermediate interruption. The hyper parameters are selected according to the performance on the development set. For such a deep and large network, it is not easy to determine the tuning strategy and this will be considered in future work.

3.3 Generation

We use the common left-to-right beam-search method for sequence generation. At each time step t, the word y_t can be predicted by:

$$\hat{y}_t = \arg\max_{y} \mathrm{P}(y|\hat{\boldsymbol{y}}_{0:t-1}, \boldsymbol{x}; \boldsymbol{\theta}) \tag{11}$$

where $\hat{y}_t$ is the predicted target word. $\hat{\boldsymbol{y}}_{0:t-1}$ is the generated sequence from time step 0 to $t-1$. We keep n_b best candidates according to Eq. 11 at each time step, until the end of sentence mark is generated. The hypotheses are ranked by the total likelihood of the generated sequence, although normalized likelihood is used in some works (Jean et al., 2015).

4 Experiments

We evaluate our method mainly on the widely used WMT'14 English-to-French translation task. In order to validate our model on more difficult language pairs, we also provide results on the WMT'14 English-to-German translation task. Our models are implemented in the PADDLE (PArallel Distributed Deep LEarning) platform.

4.1 Data sets

For both tasks, we use the full WMT'14 parallel corpus as our training data. The detailed data sets are listed below:

- English-to-French: Europarl v7, Common Crawl, UN, News Commentary, Gigaword
- English-to-German: Europarl v7, Common Crawl, News Commentary

In total, the English-to-French corpus includes 36 million sentence pairs, and the English-to-German corpus includes 4.5 million sentence pairs. The news-test-2012 and news-test-2013 are concatenated as our development set, and the news-test-2014 is the test set. Our data partition is consistent with previous works on NMT (Luong et al., 2015; Jean et al., 2015) to ensure fair comparison.

For the source language, we select the most frequent 200K words as the input vocabulary. For the target language we select the most frequent 80K French words and the most frequent 160K German words as the output vocabulary. The full vocabulary of the German corpus is larger (Jean et al., 2015), so we select more German words to build the target vocabulary. Out-of-vocabulary words are replaced with the unknown symbol $\langle \text{unk} \rangle$. For complete comparison to previous work on the English-to-French task, we also show the results with a smaller vocabulary of 30K input words and 30K output words on the sub train set with selected 12M parallel sequences (Schwenk, 2014; Sutskever et al., 2014; Cho et al., 2014).

4.2 Model settings

We have two models as described above, named Deep-ED and Deep-Att. Both models have exactly the same configuration and layer size except the interface part P-I.

We use 256 dimensional word embeddings for both the source and target languages. All LSTM layers, including the $2 \times n_e$ layers in the encoder and the n_d layers in the decoder, have 512 memory cells. The output layer size is the same as the size of the target vocabulary. The dimension of c_t is 5120 and 1280 for Deep-ED and Deep-Att respectively. For each LSTM layer, the activation functions for gates, inputs and outputs are sigmoid, $\tanh$, and $\tanh$ respectively.

Our network is narrow on word embeddings and LSTM layers. Note that in previous work (Sutskever et al., 2014; Bahdanau et al., 2015), 1000 dimensional word embeddings and 1000 dimensional LSTM layers are used. We also tried larger scale models but did not obtain further improvements.

4.3 Optimization

Note that each LSTM layer includes two parts as described in Eq. 3, feed-forward computation and recurrent computation. Since there are non-linear activations in the recurrent computation, a larger learning rate $l_r = 5 \times 10^{-4}$ is used, while for the feed-forward computation a smaller learning rate $l_f = 4 \times 10^{-5}$ is used. Word embeddings and the softmax layer also use this learning rate l_f. We refer all the parameters not used for recurrent computation as non-recurrent part of the model.

Because of the large model size, we use strong L_2 regularization to constrain the parameter matrix v in the following way:

$$v \leftarrow v - l \cdot (g + r \cdot v) \qquad (12)$$

Here r is the regularization strength, l is the corresponding learning rate, g stands for the gradients of v. The two embedding layers are not regularized. All the other layers have the same $r = 2$.

The parameters of the recurrent computation part are initialized to zero. All non-recurrent parts are randomly initialized with zero mean and standard deviation of 0.07. A detailed guide for setting hyper-parameters can be found in (Bengio, 2012).

The dropout ratio p_d is 0.1. In each batch, there are $500 \sim 800$ sequences in our work. The exact number depends on the sequence lengths and model size. We also find that larger batch size results in better convergence although the improvement is not large. However, the largest batch size is constrained by the GPU memory. We use $4 \sim 8$ GPU machines (each has 4 K40 GPU cards) running for 10 days to train the full model with parallelization at the data batch level. It takes nearly 1.5 days for each pass.

One thing we want to emphasize here is that our deep model is not sensitive to these settings. Small variation does not affect the final performance.

4.4 Results

We evaluate the same way as previous NMT works (Sutskever et al., 2014; Luong et al., 2015; Jean et al., 2015). All reported BLEU scores are computed with the *multi-bleu.perl*[1] script which is also used in the above works. The results are for tokenized and case sensitive evaluation.

4.4.1 Single models

English-to-French: First we list our single model results on the English-to-French task in Tab. 1. In the first block we show the results with the full

[1] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

corpus. The previous best single NMT encoder-decoder model (Enc-Dec) with six layers achieves BLEU=31.5 (Luong et al., 2015). From Deep-ED, we obtain the BLEU score of 36.3, which outperforms Enc-Dec model by 4.8 BLEU points. This result is even better than the ensemble result of eight Enc-Dec models, which is 35.6 (Luong et al., 2015). This shows that, in addition to the convolutional layers for computer vision, deep topologies can also work for LSTM layers. For Deep-Att, the performance is further improved to 37.7. We also list the previous state-of-the-art performance from a conventional SMT system (Durrani et al., 2014) with the BLEU of 37.0. This is the first time that a single NMT model trained in an end-to-end form beats the best conventional system on this task.

We also show the results on the smaller data set with 12M sentence pairs and 30K vocabulary in the second block. The two attention models, RNNsearch (Bahdanau et al., 2015) and RNNsearch-LV (Jean et al., 2015), achieve BLEU scores of 28.5 and 32.7 respectively. Note that RNNsearch-LV uses a large output vocabulary of 500K words based on the standard attention model RNNsearch. We obtain BLEU=35.9 which outperforms its corresponding shallow model RNNsearch by 7.4 BLEU points. The SMT result from (Schwenk, 2014) is also listed and falls behind our model by 2.6 BLEU points.

Methods	Data	Voc	BLEU
Enc-Dec (Luong,2015)	36M	80K	31.5
SMT (Durrani,2014)	36M	Full	37.0
Deep-ED (Ours)	36M	80K	36.3
Deep-Att (Ours)	36M	80K	**37.7**
RNNsearch (Bahdanau,2014)	12M	30K	28.5
RNNsearch-LV (Jean,2015)	12M	500K	32.7
SMT (Schwenk,2014)	12M	Full	33.3
Deep-Att (Ours)	12M	30K	35.9

Table 1: English-to-French task: BLEU scores of single neural models. We also list the conventional SMT system for comparison.

Moreover, during the generation process, we obtained the best BLEU score with beam size $= 3$ (when the beam size is 2, there is only a 0.1 difference in BLEU score). This is different from other works listed in Tab. 1, where the beam size is 12 (Jean et al., 2015; Sutskever et al., 2014). We attribute this difference to the improved model performance, where the ground truth generally exists in the top hypothesis. Consequently, with the much smaller beam size, the generation efficiency is significantly improved.

Next we list the effect of the novel F-F connections in our Deep-Att model of shallow topology in Tab. 2. When $n_e = 1$ and $n_d = 1$, the BLEU scores are 31.2 without F-F and 32.3 with F-F. Note that the model without F-F is exactly the standard attention model (Bahdanau et al., 2015). Since there is only a single layer, the use of F-F connections means that at the interface part we include f_t into the representation (see Eq. 7). We find F-F connections bring an improvement of 1.1 in BLEU. After we increase our model depth to $n_e = 2$ and $n_d = 2$, the improvement is enlarged to 1.4. When the model is trained with larger depth without F-F connections, we find that the parameter exploding problem (Bengio et al., 1994) happens so frequently that we could not finish training. This suggests that F-F connections provide a fast way for gradient propagation.

Models	F-F	n_e	n_d	BLEU
Deep-Att	No	1	1	31.2
Deep-Att	Yes	1	1	32.3
Deep-Att	No	2	2	33.3
Deep-Att	Yes	2	2	34.7

Table 2: The effect of F-F. We list the BLEU scores of Deep-Att with and without F-F. Because of the parameter exploding problem, we can not list the model performance of larger depth without F-F. For $n_e = 1$ and $n_d = 1$, F-F connections only contribute to the representation at interface part (see Eq. 7).

Removing F-F connections also reduces the corresponding model size. In order to figure out the effect of F-F comparing models with the same parameter size, we increase the LSTM layer width of Deep-Att without F-F. In Tab. 3 we show that, after using a two times larger LSTM layer width of 1024, we can only obtain a BLEU score of 33.8, which is still worse than the corresponding Deep-Att with F-F.

We also notice that the interleaved bi-directional encoder starts to work when the encoder depth is larger than 1. The effect of the interleaved bi-directional encoder is shown in Tab. 4. For our largest model with $n_e = 9$ and $n_d = 7$, we compared

Models	F-F	n_e	n_d	width	BLEU
Deep-Att	No	2	2	512	33.3
Deep-Att	No	2	2	1024	33.8
Deep-Att	Yes	2	2	512	34.7

Table 3: BLEU scores with different LSTM layer width in Deep-Att. After using two times larger LSTM layer width of 1024, we can only obtain BLEU score of 33.8. It is still behind the corresponding Deep-Att with F-F.

the BLEU scores of the interleaved bi-directional encoder and the uni-directional encoder (where all LSTM layers work in forward direction). We find there is a gap of about 1.5 points between these two encoders for both Deep-Att and Deep-ED.

Models	Encoder	n_e	n_d	BLEU
Deep-Att	Bi	9	7	37.7
Deep-Att	Uni	9	7	36.2
Deep-ED	Bi	9	7	36.3
Deep-ED	Uni	9	7	34.9

Table 4: The effect of the interleaved bi-directional encoder. We list the BLEU scores of our largest Deep-Att and Deep-ED models. The encoder term Bi denotes that the interleaved bi-directional encoder is used. Uni denotes a model where all LSTM layers work in forward direction.

Next we look into the effect of model depth. In Tab. 5, starting from $n_e = 1$ and $n_d = 1$ and gradually increasing the model depth, we significantly increase BLEU scores. With $n_e = 9$ and $n_d = 7$, the best score for Deep-Att is 37.7. We tried to increase the LSTM width based on this, but obtained little improvement. As we stated in Sec.2, the complexity of the encoder and decoder, which is related to the model depth, is more important than the model size. We also tried a larger depth, but the results started to get worse. With our topology and training technique, $n_e = 9$ and $n_d = 7$ is the best depth we can achieve.

Models	F-F	n_e	n_d	Col	BLEU
Deep-Att	Yes	1	1	2	32.3
Deep-Att	Yes	2	2	2	34.7
Deep-Att	Yes	5	3	2	36.0
Deep-Att	Yes	9	7	2	37.7
Deep-Att	Yes	9	7	1	36.6

Table 5: BLEU score of Deep-Att with different model depth. With $n_e = 1$ and $n_d = 1$, F-F connections only contribute to the representation at interface part where f_t is included (see Eq. 7).

The last line in Tab. 5 shows the BLEU score of 36.6 of our deepest model, where only one encoding column (Col = 1) is used. We find a 1.1 BLEU points degradation with a single encoding column. Note that the uni-directional models in Tab. 4 with uni-direction still have two encoding columns. In order to find out whether this is caused by the decreased parameter size, we test a wider model with 1024 memory blocks for the LSTM layers. It is shown in Tab. 6 that there is a minor improvement of only 0.1. We attribute this to the complementary information provided by the double encoding column.

Models	F-F	n_e	n_d	Col	width	BLEU
Deep-Att	Yes	9	7	2	512	37.7
Deep-Att	Yes	9	7	1	512	36.6
Deep-Att	Yes	9	7	1	1024	36.7

Table 6: Comparison of encoders with different number of columns and LSTM layer width.

English-to-German: We also validate our deep topology on the English-to-German task. The English-to-German task is considered a relatively more difficult task, because of the lower similarity between these two languages. Since the German vocabulary is much larger than the French vocabulary, we select 160K most frequent words as the target vocabulary. All the other hyper parameters are exactly the same as those in the English-to-French task.

We list our single model Deep-Att performance in Tab. 7. Our single model result with BLEU=20.6 is similar to the conventional SMT result of 20.7 (Buck et al., 2014). We also outperform the shallow attention models as shown in the first two lines in Tab. 7. All the results are consistent with those in the English-to-French task.

4.4.2 Post processing

Two post processing techniques are used to improve the performance further on the English-to-French task.

First, three Deep-Att models are built for ensemble results. They are initialized with different random parameters; in addition, the training corpus

Methods	Data	Voc	BLEU
RNNsearch (Jean,2015)	4.5M	50K	16.5
RNNsearch-LV (Jean,2015)	4.5M	500K	16.9
SMT (Buck,2014)	4.5M	Full	20.7
Deep-Att (Ours)	4.5M	160K	20.6

Table 7: English-to-German task: BLEU scores of single neural models. We also list the conventional SMT system for comparison.

for these models is shuffled with different random seeds. We sum over the predicted probabilities of the target words and normalize the final distribution to generate the next word. It is shown in Tab. 8 that the model ensemble can improve the performance further to 38.9. In Luong et al. (2015) and Jean et al. (2015) there are eight models for the best scores, but we only use three models and we do not obtain further gain from more models.

Methods	Model	Data	Voc	BLEU
Deep-ED	Single	36M	80K	36.3
Deep-Att	Single	36M	80K	37.7
Deep-Att	Single+PosUnk	36M	80K	39.2
Deep-Att	Ensemble	36M	80K	38.9
Deep-Att	Ensemble+PosUnk	36M	80K	**40.4**
SMT	Durrani, 2014	36M	Full	37.0
Enc-Dec	Ensemble+PosUnk	36M	80K	37.5

Table 8: BLEU scores of different models. The first two blocks are our results of two single models and models with post processing. In the last block we list two baselines of the best conventional SMT system and NMT system.

Second, we recover the unknown words in the generated sequences with the Positional Unknown (PosUnk) model introduced in (Luong et al., 2015). The full parallel corpus is used to obtain the word mappings (Liang et al., 2006). We find this method provides an additional 1.5 BLEU points, which is consistent with the conclusion in Luong et al. (2015). We obtain the new BLEU score of 39.2 with a single Deep-Att model. For the ensemble models of Deep-Att, the BLEU score rises to 40.4. In the last two lines, we list the conventional SMT model (Durrani et al., 2014) and the previous best neural models based system Enc-Dec (Luong et al., 2015) for comparison. We find our best score outperforms the previous best score by nearly 3 points.

4.5 Analysis

4.5.1 Length

On the English-to-French task, we analyze the effect of the source sentence length on our models as shown in Fig. 3. Here we show five curves: our Deep-Att single model, our Deep-Att ensemble model, our Deep-ED model, a previously proposed Enc-Dec model with four layers (Sutskever et al., 2014) and an SMT model (Durrani et al., 2014). We find our Deep-Att model works better than the previous two models (Enc-Dec and SMT) on nearly all sentence lengths. It is also shown that for very long sequences with length over 70 words, the performance of our Deep-Att does not degrade, when compared to another NMT model Enc-Dec. Our Deep-ED also has much better performance than the shallow Enc-Dec model on nearly all lengths, although for long sequences it degrades and starts to fall behind Deep-Att.

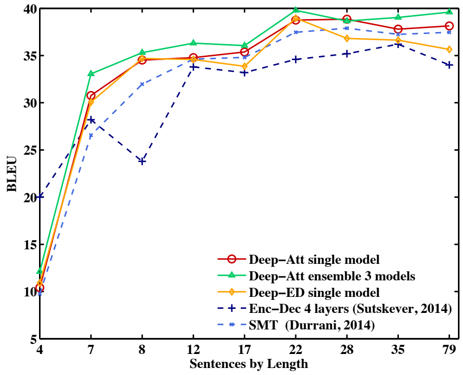

Figure 3: BLEU scores vs. source sequence length. Five lines are our Deep-Att single model, Deep-Att ensemble model, our Deep-ED model, previous Enc-Dec model with four layers and SMT model.

4.5.2 Unknown words

Next we look into the detail of the effect of unknown words on the English-to-French task. We select the subset without unknown words on target sentences from the original test set. There are 1705 such sentences (56.8%). We compute the BLEU scores on this subset and the results are shown in Tab. 9. We also list the results from SMT model (Durrani et al., 2014) as a comparison.

We find that the BLEU score of Deep-Att on this subset rises to 40.3, which has a gap of 2.6 with

Model	Test set	Ratio(%)	BLEU
Deep-Att	Full	100.0	37.7
Ensemble	Full	100.0	38.9
SMT(Durrani)	Full	100.0	37.0
Deep-Att	Subset	56.8	40.3
Ensemble	Subset	56.8	41.4
SMT(Durrani)	Subset	56.8	37.5

Table 9: BLEU scores of the subset of the test set without considering unknown words.

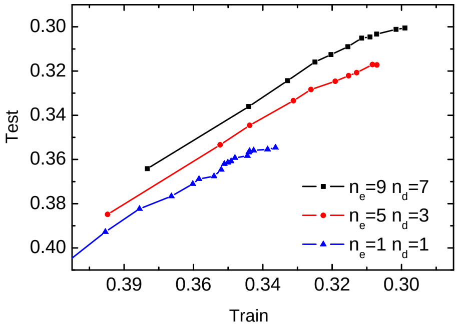

Figure 4: Token error rate on train set vs. test set. Square: Deep-Att ($n_e = 9$, $n_d = 7$). Circle: Deep-Att ($n_e = 5$, $n_d = 3$). Triagle: Deep-Att ($n_e = 1$, $n_d = 1$).

the score 37.7 on the full test set. On this subset, the SMT model achieves 37.5, which is similar to its score 37.0 on the full test set. This suggests that the difficulty on this subset is not much different from that on the full set. We therefore attribute the larger gap for Deep-att to the existence of unknown words. We also compute the BLEU score on the subset of the ensemble model and obtain 41.4. As a reference related to human performance, in Sutskever et al. (2014), it has been tested that the BLEU score of oracle re-scoring the LIUM 1000-best results (Schwenk, 2014) is 45.

4.5.3 Over-fitting

Deep models have more parameters, and thus have a stronger ability to fit the large data set. However, our experimental results suggest that deep models are less prone to the problem of over-fitting.

In Fig. 4, we show three results from models with a different depth on the English-to-French task. These three models are evaluated by token error rate, which is defined as the ratio of incorrectly predicted words in the whole target sequence with correct historical input. The curve with square marks corresponds to Deep-Att with $n_e = 9$ and $n_d = 7$. The curve with circle marks corresponds to $n_e = 5$ and $n_d = 3$. The curve with triangle marks corresponds to $n_e = 1$ and $n_d = 1$. We find that the deep model has better performance on the test set when the token error rate is the same as that of the shallow models on the training set. This shows that, with decreased token error rate, the deep model is more advantageous in avoiding the over-fitting phenomenon. We only plot the early training stage curves because, during the late training stage, the curves are not smooth.

5 Conclusion

With the introduction of fast-forward connections to the deep LSTM network, we build a fast path with neither non-linear transformations nor recurrent computation to propagate the gradients from the top to the deep bottom. On this path, gradients decay much slower compared to the standard deep network. This enables us to build the deep topology of NMT models.

We trained NMT models with depth of 16 including 25 LSTM layers and evaluated them mainly on the WMT'14 English-to-French translation task. This is the deepest topology that has been investigated in the NMT area on this task. We showed that our Deep-Att exhibits 6.2 BLEU points improvement over the previous best single model, achieving a 37.7 BLEU score. This single end-to-end NMT model outperforms the best conventional SMT system (Durrani et al., 2014) and achieves a state-of-the-art performance. After utilizing unknown word processing and model ensemble of three models, we obtained a BLEU score of 40.4, an improvement of 2.9 BLEU points over the previous best result. When evaluated on the subset of the test corpus without unknown words, our model achieves 41.4. Our model is also validated on the more difficult English-to-German task.

Our model is also efficient in sequence generation. The best results from both a single model and model ensemble are obtained with a beam size of 3, much smaller than previous NMT systems where beam size is about 12 (Jean et al., 2015; Sutskever et al., 2014). From our analysis, we find that deep models are more advantageous for learning for long sequences and that the deep topology is resistant to the over-fitting problem.

We tried deeper models and did not obtain further improvements with our current topology and training techniques. However, the depth of 16 is not very deep compared to the models in computer vision (He et al., 2016). We believe we can benefit from deeper models, with new designs of topologies and training techniques, which remain as our future work.

References

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Yoshua Bengio, 2012. *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing*.

Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. 2014. Edinburgh's phrase-based machine translation systems for WMT-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Mikel L. Forcada and Ramón P. Ñeco. 1997. Recursive hetero-associative memories for translation. In *Biological and Artificial Computation: From Neuroscience to Technology*, Berlin, Heidelberg. Springer Berlin Heidelberg.

Alex Graves, Marcus Liwicki, Santiago Fernandez, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for 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*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the Empirical Methods in Natural Language Processing*.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2016. Grid long short-term memory. In *Proceedings of International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the North American Chapter of the Association of Computational Linguistics on Human Language Technology*.

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*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L. Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-RNN). In *Proceedings of International Conference on Learning Representations*.

Holger Schwenk. 2014. http://www-lium.univ-lemans.fr/∼schwenk/cslm_joint_paper [online; accessed 03-september-2014]. *University Le Mans*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the 32nd International Conference on Machine Learning, Deep Learning Workshop*.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the 32nd International Conference on Machine Learning, Deep Learning Workshop*.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated LSTM. *arXiv:1508.03790*.

Yang Yu, Wei Zhang, Chung-Wei Hang, Bing Xiang, and Bowen Zhou. 2015. Empirical study on deep learning models for QA. *arXiv:1510.07526*.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.