Dense Information Flow for Neural Machine Translation

Yanyao Shen[1], Xu Tan[2], Di He[3], Tao Qin[2], and Tie-Yan Liu[2]

[1]University of Texas at Austin
[2]Microsoft Research, Asia
[3]Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
shenyanyao@utexas.edu, {xuta,taoqin,tie-yan.liu}@microsoft.com,di_he@pku.edu.cn

Abstract

Recently, neural machine translation has achieved remarkable progress by introducing well-designed deep neural networks into its encoder-decoder framework. From the optimization perspective, residual connections are adopted to improve learning performance for both encoder and decoder in most of these deep architectures, and advanced attention connections are applied as well. Inspired by the success of the DenseNet model in computer vision problems, in this paper, we propose a densely connected NMT architecture (DenseNMT) that is able to train more efficiently for NMT. The proposed DenseNMT not only allows dense connection in creating new features for both encoder and decoder, but also uses the dense attention structure to improve attention quality. Our experiments on multiple datasets show that DenseNMT structure is more competitive and efficient.

1 Introduction

Neural machine translation (NMT) is a challenging task that attracts lots of attention in recent years. Starting from the encoder-decoder framework (Cho et al., 2014), NMT starts to show promising results in many language pairs. The evolving structures of NMT models in recent years have made them achieve higher scores and become more favorable. The attention mechanism (Bahdanau et al., 2015) added on top of encoder-decoder framework is shown to be very useful to automatically find alignment structure, and single-layer RNN-based structure has evolved into deeper models with more efficient transformation functions (Gehring et al., 2017; Kaiser et al., 2017; Vaswani et al., 2017).

One major challenge of NMT is that its models are hard to train in general due to the complexity of both the deep models and languages. From the optimization perspective, deeper models are hard to efficiently back-propagate the gradients, and this phenomenon as well as its solution is better explored in the computer vision society. Residual networks (ResNet) (He et al., 2016) achieve great performance in a wide range of tasks, including image classification and image segmentation. Residual connections allow features from previous layers to be accumulated to the next layer easily, and make the optimization of the model efficiently focus on refining upper layer features.

NMT is considered as a challenging problem due to its sequence-to-sequence generation framework, and the goal of comprehension and reorganizing from one language to the other. Apart from the encoder block that works as a feature generator, the decoder network combining with the attention mechanism bring new challenges to the optimization of the models. While nowadays best-performing NMT systems use residual connections, we question whether this is the most efficient way to propagate information through deep models. In this paper, inspired by the idea of using dense connections for training computer vision tasks (Huang et al., 2016), we propose a densely connected NMT framework (DenseNMT) that efficiently propagates information from the encoder to the decoder through the attention component. Taking the CNN-based deep architecture as an example, we verify the efficiency of DenseNMT. Our contributions in this work include: (i) by comparing the loss curve, we show that DenseNMT allows the model to pass information more efficiently, and speeds up training; (ii) we show through ablation study that dense con-

Proceedings of NAACL-HLT 2018, pages 1294–1303
New Orleans, Louisiana, June 1 - 6, 2018. ©2018 Association for Computational Linguistics

nections in all three blocks altogether help improve the performance, while not increasing the number of parameters; (iii) DenseNMT allows the models to achieve similar performance with much smaller embedding size; (iv) DenseNMT on IWSLT14 German-English and Turkish-English translation tasks achieves new benchmark BLEU scores, and the result on WMT14 English-German task is more competitive than the residual connections based baseline model.

2 Related Work

ResNet and DenseNet. ResNet (He et al., 2016) proposes residual connections, which directly add representation from the previous layer to the next layer. Originally proposed for image classification tasks, the residual structure have proved its efficiency in model training across a wide range of tasks, and are widely adopted in recent advanced NMT models (Wu et al., 2016; Vaswani et al., 2017; Gehring et al., 2017). Following the idea of ResNet, DenseNet (Huang et al., 2016) further improves the structure and achieves state-of-the-art results. It allows the transformations (e.g., CNN) to be directly calculated over all previous layers. The benefit of DenseNet is to encourage upper layers to create new representations instead of refining the previous ones. On other tasks such as segmentation, dense connections also achieve high performance (Jégou et al., 2017). Very recently, (Godin et al., 2017) shows that dense connections help improve language modeling as well. Our work is the first to explore dense connections for NMT tasks.

Attention mechanisms in NMT. The attention block is proven to help improve inference quality due to existence of alignment information (Bahdanau et al., 2015). Traditional sequence-to-sequence architectures (Kalchbrenner and Blunsom, 2013; Cho et al., 2014) pass the last hidden state from the encoder to the decoder; hence source sentences of different length are encoded into a fixed-size vector (i.e., the last hidden state), and the decoder should catch all the information from the vector. Later, early attention-based NMT architectures, including (Bahdanau et al., 2015), pass all the hidden states (instead of the last state) of the last encoder layer to the decoder. The decoder then uses an attention mechanism to selectively focus on those hidden states while generating each word in the target sentence. Latest architecture (Gehring et al., 2017) uses multi-step attention, which allows each decoder layer to acquire separate attention representations, in order to maintain different levels of semantic meaning. They also enhance the performance by using embeddings of input sentences. In this work, we further allow every encoder layer to directly pass the information to the decoder side.

Encoder/decoder networks. RNNs such as long short term memory (LSTM) are widely used in NMT due to their ability of modeling long-term dependencies. Recently, other more efficient structures have been proposed in substitution for RNN-based structures, which includes convolution (Gehring et al., 2017; Kaiser et al., 2017) and self-attention (Vaswani et al., 2017). More specifically, ConvS2S (Gehring et al., 2017) uses convolution filter with a gated linear unit, Transformer (Vaswani et al., 2017) uses self-attention function before a two-layer position-wise feed-forward networks, and SliceNet (Kaiser et al., 2017) uses a combination of ReLU, depthwise separable convolution, and layer normalization. The advantage of these non-sequential transformations is the significant parallel speedup as well as more advanced performances, which is the reason we select CNN-based models for our experiments.

3 DenseNMT

In this section, we introduce our DenseNMT architecture. In general, compared with residual connected NMT models, DenseNMT allows each layer to provide its information to all subsequent layers directly. Figure 1-3 show the design of our model structure by parts.

We start with the formulation of a regular NMT model. Given a set of sentence pairs $S = \{(x^i, y^i)|i=1,\cdots,N\}$, an NMT model learns parameter θ by maximizing the log-likelihood function:

$$\sum_{i=1}^{N} \log \mathcal{P}(y^i|x^i;\theta). \tag{1}$$

For every sentence pair $(x, y) \in S$, $\mathcal{P}(y|x;\theta)$ is calculated based on the decomposition:

$$\mathcal{P}(y|x;\theta) = \prod_{j=1}^{m} \mathcal{P}(y_j|y_{<j}, x;\theta), \tag{2}$$

where m is the length of sentence y. Typically, NMT models use the encoder-attention-decoder

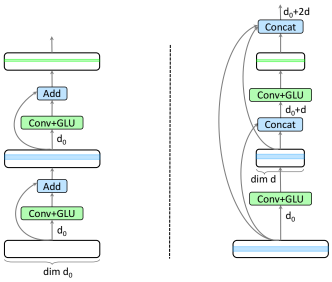

Figure 1: Comparison of dense-connected encoder and residual-connected encoder. Left: regular residual-connected encoder. Right: dense-connected encoder. Information is directly passed from blue blocks to the green block.

framework (Bahdanau et al., 2015), and potentially use multi-layer structure for both encoder and decoder. Given a source sentence x with length n, the encoder calculates hidden representations by layer. We denote the representation in the l-th layer as h^l, with dimension $n \times d^l$, where d^l is the dimension of features in layer l. The hidden representation at each position h^l_j is either calculated by:

$$h^l_j = \mathcal{H}^{\texttt{rec}}(h^{l-1}_j, h^l_{j-1}) \tag{3}$$

for recurrent transformation $\mathcal{H}^{\texttt{rec}}(\cdot)$ such as LSTM and GRU, or by:

$$h^l_j = \mathcal{H}^{\texttt{par}}(h^{l-1}) \tag{4}$$

for parallel transformation $\mathcal{H}^{\texttt{par}}(\cdot)$. On the other hand, the decoder layers $\{z^l\}$ follow similar structure, while getting extra representations from the encoder side. These extra representations are also called *attention*, and are especially useful for capturing alignment information.

In our experiments, we use convolution based transformation for $\mathcal{H}^{\texttt{par}}(\cdot)$ due to both its efficiency and high performance, more formally,

$$h^l_j = \texttt{GLU}([h^{l-1}_{j-r}, \cdots, h^{l-1}_{j+r}]W^l + b^l) \triangleq \mathcal{H}(h^{l-1}). \tag{5}$$

GLU is the gated linear unit proposed in (Dauphin et al., 2017) and the kernel size is $2r + 1$. DenseNMT is agnostic to the transformation function, and we expect it to also work well combining with other transformations, such as LSTM, self-attention and depthwise separable convolution.

3.1 Dense encoder and decoder

Different from residual connections, later layers in the dense encoder are able to use features from all previous layers by concatenating them:

$$h^{l+1} = \mathcal{H}([h^l, h^{l-1}, \cdots, h^0]). \tag{6}$$

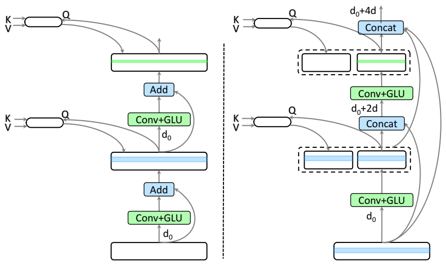

Figure 2: Comparison of dense-connected decoder and residual-connected decoder. Left: regular residual-connected decoder. Right: dense-connected decoder. Ellipsoid stands for attention block. Information is directly passed from blue blocks to the green block.

Here, $\mathcal{H}(\cdot)$ is defined in Eq. (5), $[\cdot]$ represents concatenation operation. Although this brings extra connections to the network, with smaller number of features per layer, the architecture encourages feature reuse, and can be more compact and expressive. As shown in Figure 1, when designing the model, the hidden size in each layer is much smaller than the hidden size of the corresponding layer in the residual-connected model.

While each encoder layer perceives information from its previous layers, each decoder layer z^{l+1} has two information sources: previous layers $z^i, i \leq l$, and attention values $a^i, i \leq l$. Therefore, in order to allow dense information flow, we redefine the generation of $(l+1)$-th layer as a nonlinear function over all its previous decoder layers and previous attentions. This can be written as:

$$z^{l+1} = \mathcal{H}([z^l, a^l, z^{l-1}, a^{l-1}, \cdots, z^1, a^1, z^0]), \tag{7}$$

where a^i is the attention value using i-th decoder layer and information from encoder side, which will be specified later. Figure 2 shows the comparison of a dense decoder with a regular residual decoder. The dimensions of both attention values and hidden layers are chosen with smaller values, yet the perceived information for each layer consists of a higher dimension vector with more representation power. The output of the decoder is a linear transformation of the concatenation of all layers by default. To compromise to the increment of dimensions, we use summary layers, which will be introduced in Section 3.3. With summary layers, the output of the decoder is only a linear transformation of the concatenation of the upper few layers.

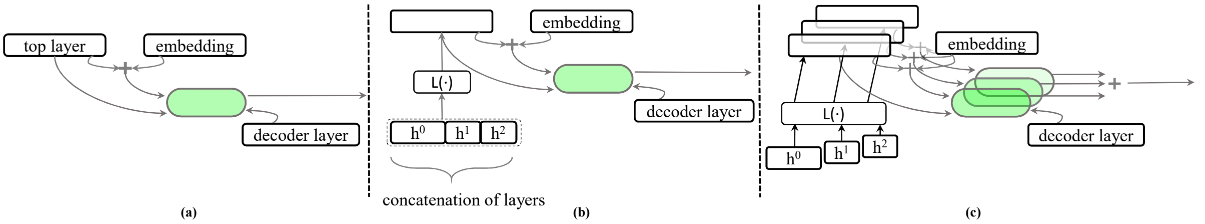

Figure 3: Illustration of DenseAtt mechanisms. For clarity, We only plot the attention block for a single decoder layer. (a): multi-step attention (Gehring et al., 2017), (b): DenseAtt-1, (c): DenseAtt-2. $\mathcal{L}(\cdot)$ is the linear projection function. The ellipsoid stands for the core attention operation as shown in Eq. (8).

3.2 Dense attention

Prior works show a trend of designing more expressive attention mechanisms (as discussed in Section 2). However, most of them only use the last encoder layer. In order to pass more abundant information from the encoder side to the decoder side, the attention block needs to be more expressive. Following the recent development of designing attention architectures, we propose DenseAtt as the dense attention block, which serves for the dense connection between the encoder and the decoder side. More specifically, two options are proposed accordingly. For each decoding step in the corresponding decoder layer, the two options both calculate attention using multiple encoder layers. The first option is more compressed, while the second option is more expressive and flexible. We name them as DenseAtt-1 and DenseAtt-2 respectively. Figure 3 shows the architecture of (a) multi-step attention (Gehring et al., 2017), (b) DenseAtt-1, and (c) DenseAtt-2 in order. In general, a popular multiplicative attention module can be written as:

$$\mathcal{F}\left(Q,K,V\right)=\texttt{Softmax}\left(Q\times K\right)\times V, \quad (8)$$

where Q,K,V represent query, key, value respectively. We will use this function $\mathcal{F}$ in the following descriptions.

DenseAtt-1 In the decoding phase, we use a layer-wise attention mechanism, such that each decoder layer absorbs different attention information to adjust its output. Instead of treating the last hidden layer as the encoder's output, we treat the concatenation of all hidden layers from encoder side as the output. The decoder layer multiplies with the encoder output to obtain the attention weights, which is then multiplied by a linear combination of the encoder output and the sentence embedding. The attention output of each layer a^l can be formally written as:

$$a^l=\mathcal{F}\left(\mathcal{L}(z^l),\mathcal{L}\left([\{h^i\}]\right),\mathcal{L}\left([\{h^i\}]\right)+\mathcal{L}(h^0)\right), \quad (9)$$

where $\mathcal{F}(\cdot,\cdot,\cdot)$ is the multiplicative attention function, $[\cdot]$ is a concatenation operation that combines all features, and $\mathcal{L}(\cdot)$ is a linear transformation function that maps each variable to a fixed dimension in order to calculate the attention value. Notice that we explicitly write the $\mathcal{L}(h^0)$ term in (9) to keep consistent with the multi-step attention mechanism, as pictorially shown in Figure 3(a).

DenseAtt-2 Notice that the transformation $\mathcal{L}([\{h^i\}])$ in DenseAtt-1 forces the encoder layers to be mixed before doing attention. Since we use multiple hidden layers from the encoder side to get an attention value, we can alternatively calculate multiple attention values before concatenating them. In another word, the decoder layer can get different attention values from different encoder layers. This can be formally expressed as:

$$a^l=\sum_{i=1}^{L}\mathcal{F}\left(\mathcal{L}(z^l),\mathcal{L}(h^i),\mathcal{L}([h^i,h^0])\right), \quad (10)$$

where the only difference from Eq. (9) is that the concatenation operation is substituted by a summation operation, and is put after the attention function $\mathcal{F}$. This method further increases the representation power in the attention block, while maintaining the same number of parameters in the model.

3.3 Summary layers

Since the number of features fed into nonlinear operation is accumulated along the path, the parameter size increases accordingly. For example, for the L-th encoder layer, the input dimension of features is $(L-1)d+d_0$, where d is the feature dimension in previous layers, d_0 is the embedding size. In order to avoid the calculation bottleneck for later layers due to large L, we introduce the *summary layer* for deeper models. It summarizes the features for all previous layers and projects back to the embedding size, so that later layers of both the encoder and the decoder side do not need to look back further. The summary layers can be considered as contextualized word vectors in a given sentence (McCann et al., 2017). We add one summary layer after every $(\texttt{sumlen} - 1)$ layers, where sumlen is the hyperparameter we introduce. Accordingly, the input dimension of features is at most $(\texttt{sumlen} - 1) \cdot d + d_0$ for the last layer of the encoder. Moreover, combined with the summary layer setting, our DenseAtt mechanism allows each decoder layer to calculate the attention value focusing on the last few encoder layers, which consists of the last contextual embedding layer and several dense connected layers with low dimension. In practice, we set sumlen as 5 or 6.

3.4 Analysis of information flow

Figure 1 and Figure 2 show the difference of information flow compared with a residual-based encoder/decoder. For residual-based models, each layer can absorb a single high-dimensional vector from its previous layer as the only information, while for DenseNMT, each layer can utilize several low-dimensional vectors from its previous layers and a high-dimensional vector from the first layer (embedding layer) as its information. In DenseNMT, each layer directly provides information to its later layers. Therefore, the structure allows feature reuse, and encourages upper layers to focus on creating new features. Furthermore, the attention block allows the embedding vectors (as well as other hidden layers) to guide the decoder's generation more directly; therefore, during back-propagation, the gradient information can be passed directly to all encoder layers simultaneously.

4 Experimental Setup

4.1 Datasets

We use three datasets for our experiments: IWSLT14 German-English, Turkish-English, and WMT14 English-German.

We preprocess the IWSLT14 German-English dataset following byte-pair-encoding (BPE) method (Sennrich et al., 2015b)[1]. We learn 25k BPE codes using the joint corpus of source and target languages. We randomly select 7k from IWSLT14 German-English as the development set , and the test set is a concatenation of dev2010, tst2010, tst2011 and tst2012, which is widely used in prior works (Ranzato et al., 2015; Bahdanau et al., 2017; Huang et al., 2017).

For the Turkish-English translation task, we use the data provided by IWSLT14 (Cettolo et al., 2014) and the SETimes corpus (Cettolo et al., 2014) following (Sennrich et al., 2015a). After removing sentence pairs with length ratio over 9, we obtain 360k sentence pairs. Since there is little commonality between the two languages, we learn 30k size BPE codes separately for Turkish and English. In addition to this, we give another preprocessing for Turkish sentences and use word-level English corpus. For Turkish sentences, following (Gulcehre et al., 2015; Sennrich et al., 2015a), we use the morphology tool Zemberek with disambiguation by the morphological analysis (Sak et al., 2007) and removal of non-surface tokens[2]. Following (Sennrich et al., 2015a), we concatenate tst2011, tst2012, tst2013, tst2014 as our test set. We concatenate dev2010 and tst2010 as the development set.

We preprocess the WMT14 English-German[3] dataset using a BPE code size of 40k. We use the concatenation of newstest2013 and newstest2012 as the development set.

4.2 Model and architect design

As the baseline model (*BASE-4L*) for IWSLT14 German-English and Turkish-English, we use a 4-layer encoder, 4-layer decoder, residual-connected model[4], with embedding and hidden size set as 256 by default. As a comparison, we design a densely connected model with same number of layers, but the hidden size is set as 128 in order to keep the model size consistent. The models adopting DenseAtt-1, DenseAtt-2 are named as *DenseNMT-4L-1* and *DenseNMT-4L-2* respectively. In order to check the effect of dense connections on deeper models, we also construct a series of 8-layer models. We set the hidden number to be 192, such that both 4-layer models and 8-layer models have similar number of parameters.

[1] https://github.com/rsennrich/subword-nmt

[2] github.com/orhanf/zemberekMorphTR

[3] https://nlp.stanford.edu/projects/nmt/

[4] https://github.com/facebookresearch/fairseq

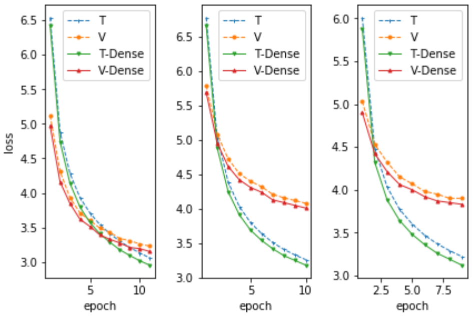

Figure 4: Training curve (T) and validation curve (V) comparison. Left: IWSLT14 German-English (De-En). Middle: Turkish-English, BPE encoding (Tr-En). Right: Turkish-English, morphology encoding (Tr-En-morph).

For dense structured models, we set the dimension of hidden states to be 96.

Since NMT model usually allocates a large proportion of its parameters to the source/target sentence embedding and softmax matrix, we explore in our experiments to what extent decreasing the dimensions of the three parts would harm the BLEU score. We change the dimensions of the source embedding, the target embedding as well as the softmax matrix simultaneously to smaller values, and then project each word back to the original embedding dimension through a linear transformation. This significantly reduces the number of total parameters, while not influencing the upper layer structure of the model.

We also introduce three additional models we use for ablation study, all using 4-layer structure. Based on the residual connected *BASE-4L* model, (1) *DenseENC-4L* only makes encoder side dense, (2) *DenseDEC-4L* only makes decoder side dense, and (3) *DenseAtt-4L* only makes the attention dense using DenseAtt-2. There is no summary layer in the models, and both *DenseENC-4L* and *DenseDEC-4L* use hidden size 128. Again, by reducing the hidden size, we ensure that different 4-layer models have similar model sizes.

Our design for the WMT14 English-German model follows the best performance model provided in (Gehring et al., 2017). The construction of our model is straightforward: our 15-layer model *DenseNMT-En-De-15* uses dense connection with DenseAtt-2, `sumlen` $= 6$. The hidden number in each layer is $1/4$ that of the original model, while the kernel size maintains the same.

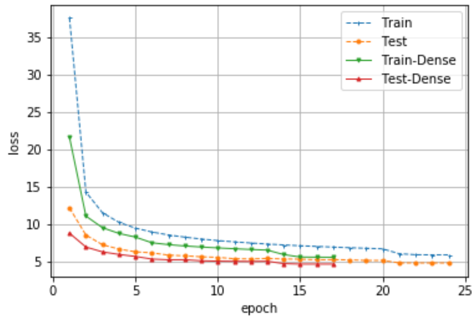

Figure 5: Training curve and test curve comparison on WMT14 English-German translation task.

4.3 Training setting

We use Nesterov Accelerated Gradient (NAG) (Nesterov, 1983) as our optimizer, and the initial learning rate is set to 0.25. For German-English and Turkish-English experiments, the learning rate will shrink by 10 every time the validation loss increases. For the English-German dataset, in consistent with (Gehring et al., 2017), the learning rate will shrink by 10 every epoch since the first increment of validation loss. The system stops training until the learning rate is less than 10^{-4}. All models are trained end-to-end without any warmstart techniques. We set our batch size for the WMT14 English-German dataset to be 48, and additionally tune the length penalty parameter, in consistent with (Gehring et al., 2017). For other datasets, we set batch size to be 32. During inference, we use a beam size of 5.

5 Results

5.1 Training curve

We first show that DenseNMT helps information flow more efficiently by presenting the training loss curve. All hyperparameters are fixed in each plot, only the models are different. In Figure 4, the loss curves for both training and dev sets (before entering the finetuning period) are provided for De-En, Tr-En and Tr-En-morph. For clarity, we compare *DenseNMT-4L-2* with *BASE-4L*. We observe that DenseNMT models are consistently better than residual-connected models, since their loss curves are always below those of the baseline models. The effect is more obvious on the WMT14 English-German dataset. We rerun the best model provided by (Gehring et al., 2017) and compare with our model. In Figure 5, where train/test loss curve are provided, DenseNMT-En-

		De-En			Tr-En			Tr-En-morph		
	Embed size	64	128	256	64	128	256	64	128	256
	Model size (M)	8 ± 1	11 ± 1	17 ± 1	11 ± 1	17 ± 1	28 ± 1	13 ± 1	21 ± 1	36 ± 1
4L	BASE-4L	28.97	29.99	30.43	**19.80**	20.26	20.99	18.90	18.81	20.08
	DenseNMT-4L-1	**30.11**	**30.80**	31.26	19.21	20.08	21.36	18.83	20.16	21.43
	DenseNMT-4L-2	29.77	30.01	**31.40**	19.59	**20.86**	**21.48**	**19.04**	**20.19**	**21.57**
8L	BASE-8L	30.15	30.91	31.51	20.40	21.60	21.92	20.21	20.76	22.62
	DenseNMT-8L-1	**30.91**	**31.54**	32.08	21.82	**22.20**	23.20	21.20	21.73	22.60
	DenseNMT-8L-2	30.70	31.17	**32.26**	**21.93**	21.98	**23.25**	**21.73**	**22.44**	**23.45**

Table 1: BLEU score on IWSLT German-English and Turkish-English translation tasks. We compare models using different embedding sizes, and keep the model size consistent within each column.

De-15 reaches the same level of loss and starts finetuning (validation loss starts to increase) at epoch 13, which is 35% faster than the baseline.

Adding dense connections changes the architecture, and would slightly influence training speed. For the WMT14 En-De experiments, the computing time for both DenseNMT and the baseline (with similar number of parameters and same batch size) tested on single M40 GPU card are 1571 and 1710 word/s, respectively. While adding dense connections influences the per-iteration training slightly (8.1% reduction of speed), it uses many fewer epochs, and achieves a better BLEU score. In terms of training time, DenseNMT uses 29.3%(before finetuning)/22.9%(total) less time than the baseline.

5.2 DenseNMT improves accuracy with similar architectures and model sizes

	De-En	Tr-En	Tr-En-morph
BASE	30.43	20.99	20.08
DenseENC-4L	30.72	21.32	21.24
DenseDEC-4L	31.23	21.04	21.06
DenseAtt-4L	31.05	21.35	21.08
DenseNMT-4L-1	31.26	21.36	21.43
DenseNMT-4L-2	31.40	21.48	21.57

Table 2: Ablation study for encoder block, decoder block, and attention block in DenseNMT.

Table 1 shows the results for De-En, Tr-En, Tr-En-morph datasets, where the best accuracy for models with the same depth and of similar sizes are marked in boldface. In almost all genres, DenseNMT models are significantly better than the baselines. With embedding size 256, where all models achieve their best scores, DenseNMT outperforms baselines by 0.7-1.0 BLEU on De-En, 0.5-1.3 BLEU on Tr-En, 0.8-1.5 BLEU on Tr-En-morph. We observe significant gain using other embedding sizes as well.

Furthermore, in Table 2, we investigate DenseNMT models through ablation study. In order to make the comparison fair, six models listed have roughly the same number of parameters. On De-En, Tr-En and Tr-En-morph, we see improvement by making the encoder dense, making the decoder dense, and making the attention dense. Fully dense-connected model *DenseNMT-4L-1* further improves the translation accuracy. By allowing more flexibility in dense attention, *DenseNMT-4L-2* provides the highest BLEU scores for all three experiments.

From the experiments, we have seen that enlarging the information flow in the attention block benefits the models. The dense attention block provides multi-layer information transmission from the encoder to the decoder, and to the output as well. Meanwhile, as shown by the ablation study, the dense-connected encoder and decoder both give more powerful representations than the residual-connected counterparts. As a result, the integration of the three parts improve the accuracy significantly.

5.3 DenseNMT with smaller embedding size

From Table 1, we also observe that DenseNMT performs better with small embedding sizes compared to residual-connected models with regular embedding size. For example, on Tr-En model, the 8-layer *DenseNMT-8L-2* model with embedding size 64 matches the BLEU score of the 8-layer BASE model with embedding size 256, while the number of parameter of the former one is only 40% of the later one. In all genres, DenseNMT model with embedding size 128 is comparable or even better than the baseline model with embedding size 256.

While overlarge embedding sizes hurt accuracy because of overfitting issues, smaller sizes are not

	Test Set				
	tst2011	**tst2012**	**tst2013**	**tst2014**	**total**
RNN (Gulcehre et al., 2015)	18.40	18.77	19.86	18.64	/
BASE	21.66	22.45	23.76	22.59	22.62
DenseNMT-8L-2	22.52	23.81	23.91	23.68	23.45
DenseNMT-8L-2(embed 256, hid 128)	23.33	24.65	24.92	24.54	24.36

Table 3: Accuracy on Turkish-English translation task in terms of BLEU score.

preferable because of insufficient representation power. However, our dense models show that with better model design, the embedding information can be well concentrated on fewer dimensions, e.g., 64. This is extremely helpful when building models on mobile and small devices where the model size is critical. While there are other works that stress the efficiency issue by using techniques such as separable convolution (Kaiser et al., 2017), and shared embedding (Vaswani et al., 2017), our DenseNMT framework is orthogonal to those approaches. We believe that other techniques would produce more efficient models through combining with our DenseNMT framework.

	Greedy	**Beam**
MIXER (Ranzato et al., 2015)	20.73	21.83
AC (Bahdanau et al., 2017)	27.49	28.53
NPMT (Huang et al., 2017)	27.83	28.96
NPMT+LM (Huang et al., 2017)	/	29.16
DenseNMT-8L-2 (word)	29.11	30.33
DenseNMT-8L-1 (BPE)	30.50	32.08
DenseNMT-8L-2 (BPE)	30.80	32.26

Table 4: Accuracy on IWSLT14 German-English translation task in terms of BLEU score.

5.4 DenseNMT compares with state-of-the-art results

For the IWSLT14 German-English dataset, we compare with the best results reported from literatures. To be consistent with prior works, we also provide results using our model directly on the dataset without BPE preprocessing. As shown in Table 4, DenseNMT outperforms the phrase-structure based network NPMT (Huang et al., 2017) (with beam size 10) by 1.2 BLEU, using a smaller beam size, and outperforms the actor-critic method based algorithm (Bahdanau et al., 2017) by 2.8 BLEU. For reference, our model trained on the BPE preprocessed dataset achieves 32.26 BLEU, which is 1.93 BLEU higher than our word-based model. For Turkish-English task, we compare with (Gulcehre et al., 2015) which uses the same morphology preprocessing as our Tr-En-morph. As shown in Table 3, our baseline is higher than the previous result, and we further achieve new benchmark result with 24.36 BLEU average score. For WMT14 English-German, from Table 5, we can see that DenseNMT outperforms ConvS2S model by 0.36 BLEU score using 35% fewer training iterations and 20% fewer parameters. We also compare with another convolution based NMT model: SliceNet (Kaiser et al., 2017), which explores depthwise separable convolution architectures. SliceNet-Full matches our result, and SliceNet-Super outperforms by 0.58 BLEU score. However, both models have 2.2x more parameters than our model. We expect DenseNMT structure could help improve their performance as well.

	BLEU score
GNMT (Wu et al., 2016)	24.61
ConvS2S (Gehring et al., 2017)	25.16
SliceNet-Full (Kaiser et al., 2017)	25.5
SliceNet-Super (Kaiser et al., 2017)	26.1
DenseNMT-En-De-15	25.52

Table 5: Accuracy on WMT14 English-German translation task in terms of BLEU score.

6 Conclusion

In this work, we have proposed DenseNMT as a dense-connection framework for translation tasks, which uses the information from embeddings more efficiently, and passes abundant information from the encoder side to the decoder side. Our experiments have shown that DenseNMT is able to speed up the information flow and improve translation accuracy. For the future work, we will combine dense connections with other deep architectures, such as RNNs (Wu et al., 2016) and self-attention networks (Vaswani et al., 2017).

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. *5th International Conference on Learning Representations, ICLR, 2017* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR, 2015*.

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*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*. pages 933–941.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia.* .

Fréderic Godin, Joni Dambre, and Wesley De Neve. 2017. Improving Language Modeling using Densely Connected Recurrent Neural Networks. *arXiv preprint arXiv:1707.06130* .

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. *Computer Speech and Language, 2016* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2016*. pages 770–778.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2016. Densely Connected Convolutional Networks. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2017* .

Po-Sen Huang, Chong Wang, Zhou Dengyong, and Deng Li. 2017. Toward Neural Phrase-based Machine Translation. *arXiv preprint arXiv:1706.05565* .

Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. 2017. The One Hundred Layers Tiramisu: Fully Convolutional Densenets for Semantic Segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, pages 1175–1183.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2017. Depthwise Separable Convolutions for Neural Machine Translation. *arXiv preprint arXiv:1706.03059* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*. volume 3, page 413.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. *arXiv preprint arXiv:1708.00107* .

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*. volume 27, pages 372–376.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence Level Training with Recurrent Neural Networks. *4th International Conference on Learning Representations, ICLR, 2016* .

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological Disambiguation of Turkish Text with Perceptron Algorithm. *Computational Linguistics and Intelligent Text Processing* pages 107–118.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving Neural Machine Translation Models with Monolingual Data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016* .

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.

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* .