Towards Linear Time Neural Machine Translation with Capsule Networks

Mingxuan Wang[1] **Jun xie**[1] **Zhixing Tan**[1] **Jinsong Su**[2] **Deyi Xiong**[3] **Chao bian**[1]

[1]Mobile Internet Group, Tencent Technology Co., Ltd
{xuanswang,stiffxie,chaobian}@tencent.com

[2]Xiamen University, Xiamen, China
{jssu}@xmu.edu.cn

[3]Soochow University, Suzhou, China
{dyxiong}@suda.edu.cn

Abstract

In this study, we first investigate a novel capsule network with dynamic routing for linear time Neural Machine Translation (NMT), referred as CAPSNMT. CAPsNMT uses an aggregation mechanism to map the source sentence into a matrix with pre-determined size, and then applys a deep LSTM network to decode the target sequence from the source representation. Unlike the previous work (Sutskever et al., 2014) to store the source sentence with a passive and bottom-up way, the dynamic routing policy encodes the source sentence with an iterative process to decide the credit attribution between nodes from lower and higher layers. CAPSNMT has two core properties: it runs in time that is linear in the length of the sequences and provides a more flexible way to select, represent and aggregates the part-whole information of the source sentence. On WMT14 English-German task and a larger WMT14 English-French task, CAPSNMT achieves comparable results with the state-of-the-art NMT systems. To the best of our knowledge, this is the first work that capsule networks have been empirically investigated for sequence to sequence problems.

1 Introduction

Neural Machine Translation (NMT) is an end-to-end learning approach to machine translation which has recently shown promising results on multiple language pairs (Luong et al., 2015; Shen et al., 2015; Wu et al., 2016; Gehring et al., 2017a; Kalchbrenner et al., 2016; Sennrich et al., 2015; Vaswani et al., 2017). Unlike conventional Statistical Machine Translation (SMT) systems (Koehn et al., 2003; Chiang, 2005) which consist of multiple separately tuned components, NMT aims at building upon a single and large neural network to directly map input text to associated output text (Sutskever et al., 2014).

In general, there are several research lines of NMT architectures, among which the Enc-Dec NMT (Sutskever et al., 2014) and the Enc-Dec Att NMT are of typical representation (Bahdanau et al., 2014; Wu et al., 2016; Vaswani et al., 2017). The Enc-Dec represents the source inputs with a fixed dimensional vector and the target sequence is generated from this vector word by word. The Enc-Dec, however, does not preserve the source sequence resolution, a feature that aggravates learning for long sequences. This results in the computational complexity of decoding process is $\mathcal{O}(|S| + |T|)$, with $|S|$ denoting the source sentence length and $|T|$ denoting the target sentence length. The Enc-Dec Att preserves the resolution of the source sentence which frees the neural model from having to squash all the information into a fixed representation, but at a cost of a quadratic running time. Due to the attention mechanism, the computational complexity of decoding process is $\mathcal{O}(|S| \times |T|)$. This drawbacks grow more severe as the length of the sequences increases.

Currently, most work focused on the Enc-Dec Att, while the Enc-Dec paradigm is less emphasized on despite its advantage of linear-time decoding (Kalchbrenner et al., 2016). The linear-time approach is appealing, however, the performance lags behind the Enc-Dec Att. One potential issue is that the Enc-Dec needs to be able to compress all the necessary information of a source sentence into a constant size encoder. Some simple aggregation methods, such as max (or average) pooling, are often used to compress the sentence meaning. The context vectors are fixed during decoding. These methods process the nformation in a bottom-up and passive way and are lack of

child-parent (or part-whole) relationships burdening the model with a memorization step. Therefore, a natural question was raised, *Will carefully designed aggregation operations help the Enc-Dec to achieve the best performance?*

In recent promising work of capsule network, a dynamic routing policy is proposed and proven to be more effective than the simple aggregation method (Sabour et al., 2017; Zhao et al., 2018; Gong et al., 2018). As an outcome, capsule networks could encode the intrinsic spatial relationship between a part and a whole constituting viewpoint invariant knowledge that automatically generalizes to novel viewpoints. Following a similar spirit to use this technique, we present a family of the Enc-Dec approaches, referred as CAPSNMT, that are characterized by capsule encoder to address the drawbacks of the conventional linear-time approaches. The capsule encoder processes the attractive potential to address the aggregation issue. We then introduce an iterative routing policy to decide the credit attribution between nodes from lower (child) and higher (parent) layers. Three strategies are also proposed to stabilize the dynamic routing process. We empirically verify CAPSNMT on WMT14 English-German task and a larger WMT14 English-French task. CAPSNMT achieves comparable results with the state-of-the-art Transformer systems. Our contributions can be summrized as follows:

- We propose a sophisticated designed linear-time CAPSNMT which achieved comparable results with the Transformer framework. To the best of our knowledge, CAPSNMT is the first work that capsule networks have been empirically investigated for sequence-to-sequence problems.

- We also propose several techniques including *position-aware routing strategy, separable composition & scoring strategy and non-sharing weight strategy* to stabilize the dynamic routing process. We believe that these technique should always be employed by capsule networks for the best performance.

2 Linear Time Neural Machine Translation

The task of linear-time translation can be understood from the perspective of machine learning as learning the conditional distribution $p(\mathbf{y}|\mathbf{x})$ of
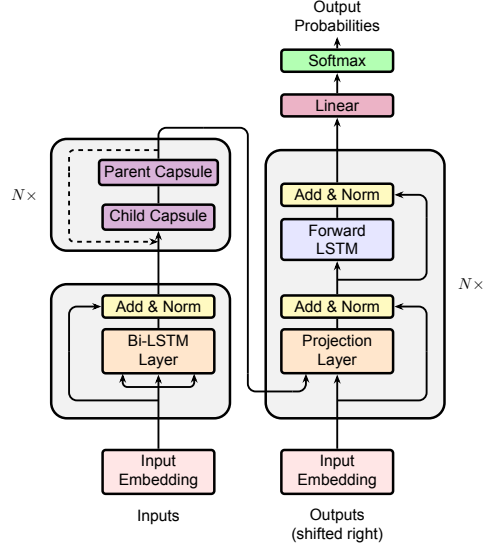

Figure 1: CAPSNMT:Linear time neural machine translation with capsule encoder

a target sentence (translation) $\mathbf{y}$ given a source sentence $\mathbf{x}$. The network that models $p(\mathbf{y}|\mathbf{x})$ is composed of two parts: a encoder that processes the source string into a representation and a decoder that uses the source representation to generate the target string. A crucial feature of the linear-time NMT is that the source sentence representation is of pre-determined size. Due to the constant size representation of the encoder, the running time of the network could be linear in the length of the source tokens. Figure 1 shows the linear time NMT with capsule encoder which encodes the source sentence into the fixed size.

Constant Encoder with Aggregation Layers Given a text sequence with words $\mathbf{x} = x_1, x_2 \cdots, x_L$, Since the words are symbols that could not be processed directly using neural architectures, so we first map each word into a d_v dimensional embedding vector

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L] \in \mathbb{R}^{L \times d_x}.$$

The goal of the constant encoder is to transfer the inputs $\mathbf{X} \in \mathbb{R}^{L \times d_v}$ into a pre-determined size representation

$$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M] \in \mathbb{R}^{M \times d_c}.$$

where L is the length of the input sentence, $M < L$ is the pre-determined length of the encoder output, d_x is the dimension of the word embedding, and d_c is the dimension of the hidden states.

In this work, we first build a bi-directional LSTM (BiLSTM) as the primary-capsule layer to incorporate forward and backward context information of a sequence:

$$\overrightarrow{\mathbf{h}_t} = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$\overleftarrow{\mathbf{h}_t} = \mathbf{LSTM}(\mathbf{h}_{t+1}, \mathbf{x}_t) \qquad (1)$$
$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}]$$

We can get sentence-level encoding $\mathbf{h}_t$ of a word by concatenating forward $\overrightarrow{\mathbf{h}_t}$ and backward output vector $\overleftarrow{\mathbf{h}_t}$. Thus, the outputs of BiLSTM encoder are a sequence of vectors $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]$ correspond to the input sequence.

The encoder consists of several primary-capsule layers to extract the basic features and then followed by several aggregation layer to map the variable-length sequence into the fixed-size representation. Max or Average pooling is the simplest way of aggregating information, which does not require extra parameters and is computationally efficient. In the process of modeling natural language, max or average pooling is performed along the time dimension.

$$\mathbf{h}^{max} = \max([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]) \qquad (2)$$
$$\mathbf{h}^{avg} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{h}_i \qquad (3)$$

We propose a more powerful aggregation methods as our strong basline of Enc-Dec approach. The outputs of encoder is static and consists of $M = 4$ parts which is

$$\mathbf{C} = [\mathbf{h}^{max}, \mathbf{h}^{avg}, \mathbf{h}_1, \mathbf{h}_L]. \qquad (4)$$

The last time step state $\mathbf{h}_L$ and the first time step state $\mathbf{h}_1$ provide complimentary information, thus improve the performance. The compressed represention is fixed once learning, therefore the compression strategy plays a crucial role in the success of building the Enc-Dec NMT model.

LSTM Decoder The goal of the LSTM is to estimate the conditional probability $p(y_t|y_{t-1}; x_1, x_2, \cdots, x_T)$, where $(x_1, x_2, \cdots, x_T)$ is the input sequence and $(y_1, y_2, \cdots, y'_T)$ is its corresponding output sequence.

A simple strategy for general sequence learning is to map the input sequence to a fixed-sized vector, and then to map the vector to the target sequence with a conditional LSTM decoder(Sutskever et al., 2014):

$$\mathbf{s}_t = \mathbf{LSTM}(\mathbf{s}_{t-1}, \mathbf{u}_t)$$
$$\mathbf{u}_t = \mathbf{W}_c[\mathbf{c}_1, \cdots, \mathbf{c}_M] + \mathbf{y}_t \qquad (5)$$

where $\mathbf{y}_t$ is the target word embedding of y_t, $\mathbf{u}_t$ is the inputs of LSTM at time step t, $[\mathbf{c}_1, \cdots, \mathbf{c}_M]$ is the concatenation of the source sentence represention and $\mathbf{W}_c$ is the projection matrix. Since $\mathbf{W}_c[\mathbf{c}_1, \cdots, \mathbf{c}_M]$ is calculated in advance, the decoding time could be linear in the length of the sentence length. At inference stage, we only utilize the top-most hidden states $\mathbf{s}_t$ to make the final prediction with a softmax layer:

$$p(y_t|y_{t-1}, \mathbf{x}) = \mathbf{softmax}(\mathbf{W}_o\mathbf{s}_t). \qquad (6)$$

Similar as (Vaswani et al., 2017), we also employ a residual connection (He et al., 2016) around each of the sub-layers, followed by layer normalization (Ba et al., 2016).

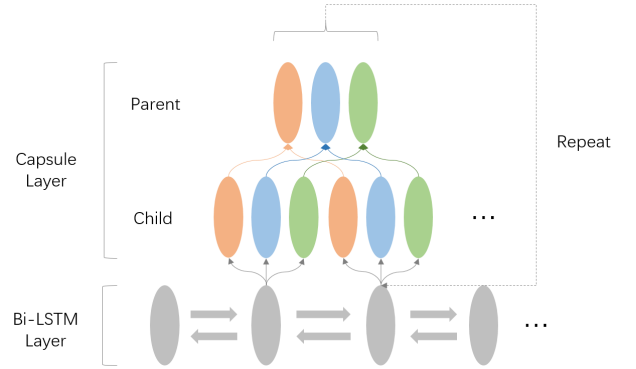## 3 Aggregation layers with Capsule Networks

Figure 2: Capsule encoder with dynamic routing by agreement

The traditional linear-time approaches collect information in a bottom-up way, without considering the state of the whole encoding. Therefore, it is difficult to avoid the problems of information attenuation. As can be seen in Figure 2, we introduce a capsule layer to aggregate the information, which can iteratively decides the information flow and provides a more flexible way to select, represent and synthesize the part-whole relationship of the source sentence.

3.1 Child-Parent Relationships

To compress the input information into the representation with pre-determined size, the central issue we should address is to determine the information flow from the input capsules to the output capsules.

Capsule network tries to address the representational limitation and exponential inefficiencies of the simple aggregation pooling method. It allows the networks to automatically learn child-parent (or part-whole) relationships. Formally, $\mathbf{u}_{i \rightarrow j}$ denotes the information be transferred from the child capsule $\mathbf{h}_i$ into the parent capsule $\mathbf{c}_j$:

$$\mathbf{u}_{i \rightarrow j} = \alpha_{ij}\mathbf{f}(\mathbf{h}_i, \boldsymbol{\theta}_j) \qquad (7)$$

where $\alpha_{ij} \in \mathbb{R}$ can be viewed as the voting weight on the information flow from child capsule to the parent capsule; $\mathbf{f}(\mathbf{h}_i, \boldsymbol{\theta}_j)$ is the transformation function and in this paper, we use a single layer feed forward neural networks:

$$\mathbf{f}(\mathbf{h}_i, \boldsymbol{\theta}_j) = \mathbf{ReLU}(\mathbf{h}_i \mathbf{W}_j) \qquad (8)$$

where $\mathbf{W}_j \in \mathbb{R}^{d_c \times dc}$ is the transformation matrix corresponding to the position j of the parent capsule.

The parent capsule aggregates all the incoming messages from all the child capsule:

$$\mathbf{v}_i = \sum_{j=1}^{L} \mathbf{u}_{i \rightarrow j} \qquad (9)$$

and then squashes $\mathbf{v}_i$ to $||\mathbf{v}_i|| \in (0,1)$ confine. ReLU or similar non linearity functions work well with single neurons. But the paper found that this squashing function works best with capsules. This tries to squash the length of output vector of a capsule. It squashes to 0 if it is a small vector and tries to limit the output vector to 1 if the vector is long.

$$\mathbf{c}_i = \mathbf{squash}(\mathbf{v}_i)$$
$$= \frac{||\mathbf{v}_i||^2}{1 + ||\mathbf{v}_i||^2} \frac{\mathbf{v}_i}{||\mathbf{v}_i||} \qquad (10)$$

3.2 Dynamic Routing by Agreement

The dynamic routing process is implemented by an EM iterative process of refining the coupling coefficient α_{ij}, which define proportionally how much information is to be transferred from $\mathbf{h}_i$ to $\mathbf{c}_j$.

At iteration t, the coupling coefficient α_{ij} is computed by

$$\alpha_{ij}^t = \frac{\exp(b_{ij}^{t-1})}{\sum_k \exp(b_{ik}^{t-1})} \qquad (11)$$

where α_{ij} is computed using a soft-max function across the child capsules. Therefore, $\sum_i \alpha_{ij} = 1$ and all the information from the child capsule h_i will be transferred to the parent. Following (Zhao et al., 2018), We explore Leaky-Softmax in the place of standard soft-max while updating the connection strength. The approach help to route the noise child capsules to extra dimension without any additional parameters and computation consuming.

$$b_{ij}^t = b_{ij}^{t-1} + \mathbf{c}_j^t \cdot \mathbf{f}(\mathbf{h}_i^t, \boldsymbol{\theta}_j^t) \qquad (12)$$

This coefficient b_{ij}^t is simply a temporary value that will be iteratively updated with the value b_{ij}^{t-1} of the previous iteration and, after the procedure is over, its value will be stored in c_{ij}^{t+1}. The agreement is simply the scalar product of $\mathbf{c}_j^{t-1}$ and $\mathbf{f}(\mathbf{h}_i^{t-1}, \boldsymbol{\theta}_j^{t-1})$. The dot product looks at similarity between input to the capsule and output from the capsule. Also, remember from above, the lower level capsule will sent its output to the higher level capsule whose output is similar. This similarity is captured by the dot product. b_{ij}^0 is initialized with 0. The coefficient depend on the location and type of both the child and the parent capsules. With a iteratively refinement of b_{ij}, the capsule network can increase or decrease the connection strength by dynamic routing, which is more effective than the primitive routing strategies such as max-pooling that essentially detects whether a feature is present in any position of the text, but loses spatial information about the feature.

When an output capsule $\mathbf{c}_j^t$ receives the incoming messages $\mathbf{u}_{i \rightarrow j}^t$, its state will be updated and the coefficient α_{ij}^t is also re-computed for each input capsule. Thus, we iteratively refine the route of information flowing, towards an instance dependent and context aware encoding of a sequence. After the text sequence is encoded into M capsules, we map these capsules into vector representation by simply concatenating all capsules:

$$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M] \qquad (13)$$

The matrix $\mathbf{C}$ with pre-determined size will then be fed to the final end to end NMT model as

Algorithm 1 Dynamic Routing Algorithm

1: **procedure** ROUTING($[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]$,$T$)
2:     Initialize $\mathbf{b}_{ij}^0 \leftarrow 0$
3:     **for** each $t \in$ range$(0 : T)$ **do**
4:         Compute the routing coefficients α_{ij}^t for all $i \in [1, L], j \in [1, M]$ ▷ From Eq.(11)
5:         Update all the output capsule $\mathbf{c}_j^t$ for all $j \in [1, M]$ ▷ From Eq.(7,8,9,10)
6:         Update all the coefficient $\mathbf{b}_{ij}^t$ for all $i \in [1, L], j \in [1, M]$ ▷ From Eq.(12)
7:     **end for**
8: **return** $[\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M]$
9: **end procedure**

---

the source sentence encoder. In this paper, we also explore three strategies to improve the accuracy of the routing process.

Position-aware Routing strategy   The routing process can iteratively decide what and how much information is to be sent to the final encoding with considering the state of both the final outputs capsule and the inputs capsule. In order for the model to make use of the order of the child and parent capsules, some information cab be injected about the relative or absolute position of the capsules in the sequence. Adding positional information in the text is more effective than in image since there is some sequential information in the sentence which can help the capsule network model the child-parent relationship more efficiently. There are many choices of positional encodings, learned and fixed (Gehring et al., 2017b). To this end, we add "positional encoding" to the child capsules and the parent capsules. The positional encodings have the same dimension as the corresponding hidden state, so that the two can be summed. Following (Vaswani et al., 2017), we use sine and cosine functions of different frequencies:

$$PE_{pos,2i} = \sin(pos/1000^{2i/d_v})$$
$$PE_{pos,2i+1} = \cos(pos/1000^{2i/d_v}) \qquad (14)$$

Non-sharing Weight Strategy   In this paper, we explore two different types of transformation matrices to generate the message vector $\mathbf{u}_{i \rightarrow j}$. from its child capsule i to the parent capsule j in Eq.(7,8). The first one shares parameters of $\boldsymbol{\theta}$ across different iterations. In the second design, we replace the shared parameters with the non-shared strategy $\boldsymbol{\theta}^t$ where t is the iteration step during the dynamic process. In our preliminary, we found that shared weight strategy works slightly better than the non-shared one which is in consistent with (Liao and Poggio, 2016). They investigated the effect of sharing weight in deep neural networks. The experiments showed that deep networks with weight sharing generally perform worse than using independent parameters.

Separable Composition and Scoring strategy The most important idea behind capsule networks is to mesure the input and output similarity. It is often modeled as dot product between input and output of a capsule and then routing coefficient is updated correspondingly. Traditional capsule networks take a straightforward strategy in which the "fusion" decisions (e.g., deciding the voting weight) are made based on the values of feature-maps. This is essentially a soft template matching (Lawrence et al., 1997), which works for tasks like classification, but undesired for maintaining the composition functionality of capsules. In this paper, we propose to use separate functional networks to release the scoring duty, and let $\boldsymbol{\theta}$ defined in Eq.(7) focus on composition. More specifically, we redefined the iteratively scoring function in Eq.(12).

$$b_{ij}^t = b_{ij}^{t-1} + \mathbf{g}(\mathbf{c}_j^t) \cdot \mathbf{g}(\mathbf{h}_i^t) \qquad (15)$$

where $\mathbf{g}(\cdot)$ is a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

4   Experiments

4.1   Datasets

We mainly evaluated CAPSNMT on the widely used WMT English-German and English-French translation task. The evaluation metric is BLEU. We tokenized the reference and evaluated the performance with multi-bleu.pl. The metrics are exactly the same as in previous work (Papineni et al., 2002).

For English-German, to compare with the results reported by previous work, we used the same subset of the WMT 2014 training corpus that contains 4.5M sentence pairs with 91M English words and 87M German words. The concatenation of news-test 2012 and news-test 2013 is used as the validation set and news-test 2014 as the test set.

To evaluate at scale, we also report the results of English-French. To compare with the results reported by previous work on end-to-end NMT, we used the same subset of the WMT 2014 training corpus that contains 36M sentence pairs. The concatenation of news-test 2012 and news-test 2013 serves as the validation set and news-test 2014 as the test set.

4.2 Training details

Our training procedure and hyper parameter choices are similar to those used by (Vaswani et al., 2017). In more details, For English-German translation and English-French translation, we use $50K$ sub-word tokens as vocabulary based on Byte Pair Encoding(Sennrich et al., 2015).

We initialized parameters by sampling each element from the Gaussian distribution with mean 0 and variance 0.01^2. Parameter optimization was performed using stochastic gradient descent. Adam (Kingma and Ba, 2015) was used to automatically adapt the learning rate of each parameter ($\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-8}$). To avoid gradient explosion, the gradients of the cost function which had ℓ_2 norm larger than a predefined threshold 25 were normalized to the threshold (Pascanu et al., 2013). We batched sentence pairs by approximate length, and limited input and output tokens per batch to 8192 per GPU. Each resulting training batch contained approximately 60,000 source and 60,000 target tokens. We trained our NMT model with the sentences of length up to 150 words in the training data. During training, we employed label smoothing of value $\epsilon = 0.1$(Pereyra et al., 2017).

Translations were generated by a beam search and log-likelihood scores were normalized by the sentence length. We used a beam width of 8 and length penalty $\alpha = 0.8$ in all the experiments. Dropout was applied on each layer to avoid overfitting (Hinton et al., 2012). The dropout rate was set to 0.3 for the English-German task and 0.1 for the English-French task. Except when otherwise mentioned, NMT systems had 4 layers encoders followed by a capsule layer and 3 layers decoders. We trained for 300,000 steps on 8 M40 GPUs, and averaged the last 50 checkpoints, saved at 30 minute intervals. For our base model, the dimensions of all the hidden states were set to 512 and for the big model, the dimensions were set to 1024. The capsule number is set to 6.

4.3 Results on English-German and English-French Translation

The results on English-German and English-French translation are presented in Table 1. We compare CAPSNMT with various other systems including the winning system in WMT'14 (Buck et al., 2014), a phrase-based system whose language models were trained on a huge monolingual text, the Common Crawl corpus. For Enc-Dec Att systems, to the best of our knowledge, GNMT is the best RNN based NMT system. Transformer (Vaswani et al., 2017) is currently the SOTA system which is about 2.0 BLEU points better than GNMT on the English-German task and 0.6 BLEU points better than GNMT on the English-French task. For Enc-Dec NMT, ByteNet is the previous state-of-the-art system which has 150 convolutional encoder layers and 150 convolutional decoder layers.

On the English-to-German task, our big CAPSNMT achieves the highest BLEU score among all the Enc-Dec approaches which even outperform ByteNet, a relative strong competitor, by +3.9 BLEU score. In the case of the larger English-French task, we achieves the highest BLEU score among all the systems which even outperform Big Transform, a relative strong competitor, by +0.6 BLEU score. To show the power of the capsule encoder, we also make a comparison with the simple aggregation version of the Enc-Dec model, and again yields a gain of +1.8 BLEU score on English-German task and +3.5 BLEU score on English-French task for the base model. The improvements is in consistent with our intuition that the dynamic routing policy is more effective than the simple aggregation method. It is also worth noting that for the small model, the capsule encoder approach get an improvement of +2.3 BLEU score over the Base Transform approach on English-French task.

The first column indicates the time complexity of the network as a function of the length of the sequences and is denoted by Time. The ByteNet, the RNN Encoder-Decoder are the only networks that have linear running time (up to the constant c). The RNN Enc-Dec, however, does not preserve the source sequence resolution, a feature that aggravates learning for long sequences. The Enc-Dec Att do preserve the resolution, but at a cost of a quadratic running time. The ByteNet overcomes these problem with the convolutional

SYSTEM	Architecture	Time	EN-Fr BLEU	EN-DE BLEU
Buck et al. (2014)	Winning WMT14	-	35.7	20.7
Existing Enc-Dec Att NMT systems				
Wu et al. (2016)	GNMT + Ensemble	$|S||T|$	40.4	26.3
Gehring et al. (2017a)	ConvS2S	$|S||T|$	40.5	25.2
Vaswani et al. (2017)	Transformer (base)	$|S||T|+|T||T|$	38.1	27.3
Vaswani et al. (2017)	Transformer (large)	$|S||T|+|T||T|$	41.0	27.9
Existing Enc-Dec NMT systems				
Luong et al. (2015)	Reverse Enc-Dec	$|S|+|T|$	-	14.0
Sutskever et al. (2014)	Reverse stack Enc-Dec	$|S|+|T|$	30.6	-
Zhou et al. (2016)	Deep Enc-Dec	$|S|+|T|$	36.3	20.6
Kalchbrenner et al. (2016)	ByteNet	$c|S|+c|T|$	-	23.7
Our Encoder-Decoder based NMT systems				
Base Model	Simple Aggregation	$c|S|+c|T|$	38.6	23.4
Base Model	CAPSNMT	$c|S|+c|T|$	40.4	26.9
Big Model	CAPSNMT	$c|S|+c|T|$	**41.6**	27.6

Table 1: Case-sensitive BLEU scores on English-German and English-French translation. CAPSNMT achieves compariable results with Transformer (Vaswani et al., 2017). We obtained the highest BLEU score among all the competitors on English-French translation.

neural network, however the architecture must be deep enough to capture the global information of a sentence. The capsule encoder makes use of the dynamic routing policy to automatically learn the part-whole relationship and encode the source sentence into fixed size representation. With the capsule encoder, CAPSNMT keeps linear running time and the constant c is the capsule number which is set to 6 in our mainly experiments.

4.4 Ablation Experiments

In this section, we evaluate the importance of our main techniques for training CAPSNMT. We believe that these techniques are universally applicable across different NLP tasks, and should always be employed by capsule networks for best performance. From Table 2 we draw the following con-

Model	BLEU
Base CAPSNMT	24.7
+ Non-weight sharing	25.1
+ Position-aware Routing Policy	25.3
+ Separable Composition and Scoring	25.7
+Knowledge Distillation	26.9

Table 2: English-German task: Ablation experiments of different technologies.

clusions:

- **Non-weight sharing strategy** We observed

that the non-weight sharing strategy improves the baseline model leading to an increase of 0.4 BLEU.

- **Position-aware Routing strategy** Adding the position embedding to the child capsule and the parent capsule can obtain an improvement of 0.2 BLEU score.

- **Separable Composition and Scoring strategy** Redefinition of the dot product function contributes significantly to the quality of the model, resulting in an increase of 0.4 BLEU score.

- **Knowledge Distillation** Sequence-level knowledge distillation is applied to alleviate multimodality in the training dataset, using the state-of-the-art transformer models as the teachers(Kim and Rush, 2016). In addition, we use the same sizes and hyperparameters for the student and its respective teacher. We decode the entire training set once using the teacher to create a new training dataset for its respective student.

4.5 Model Analysis

In this section, We study the attribution of CAPSNMT.

Effects of Iterative Routing We also study how the iteration number affect the performance of aggregation on the English-German task. Figure 3 shows the comparison of $2-4$ iterations in the dynamic routing process. The capsule number is set to $2, 4, 6$ and 8 for each comparison respectively. We found that the performances on several different capsule number setting reach the best when iteration is set to 3. The results indicate the dynamic routing is contributing to improve the performance and a larger capsule number often leads to better results.
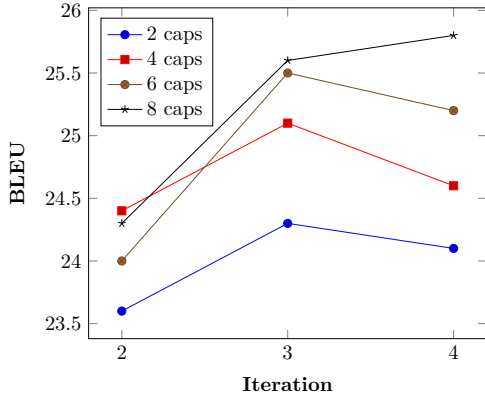

Figure 3: Effects of Iterative Routing with different capsule numbers

Model	Latency(ms)
Transformer	225
CapsNMT	153

Table 3: Time required for decoding. Decoding indicates the amount of time in millisecond required for translating one sentence, which is averaged over the whole English-German newstest2014 dataset.

Analysis on Decoding Speed We show the decoding speed of both the transformer and CapsNMT in Table 3. The capsule number is set to 6. Thg results empirically demonstrates that CapsNMT can improve the decoding speed of the transformer approach by $+50\%$.

Performance on long sentences A more detailed comparison between CapsNMT and Transformer can be seen in Figure 4. In particular, we test the BLEU scores on sentences longer than $\{0, 10, 20, 30, 40, 50\}$. We were surprised to discover that the capsule encoder did well on medium-length sentences. There is no degrada-

tion on sentences with less than 40 words, however, there is still a gap on the longest sentences. A deeper capsule encoder potentially helps to address the degradation problem and we will leave this in the future work.
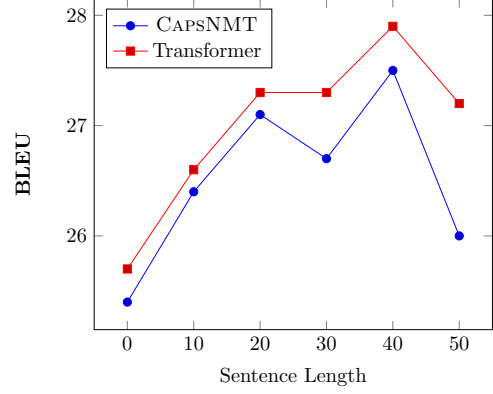

Figure 4: The plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length.

Table 4: A visualization to show the perspective of a sentence from 4 different capsules at the third iteration.

Visualization We visualize how much information each child capsule sends to the parent capsules. As shown in Table 4, the color density of each word denotes the coefficient α_{ij} at iteration 3 in Eq.(11). At first iteration, the α_{ij} follows a uniform distribution since b_{ij} is initialized to 0, and α_{ij} then will be iteratively fitted with dynamic routing policy. It is appealing to find that afte 3 iterations, the distribution of the voting weightsα_{ij} will converge to a sharp distribution and the values will be very close to 0 or 1. It is also worth mentioning that the capsule seems able to capture some structure information. For example, the information of phrase *still love each other* will be sent to the same capsule. We will make further exploration in the future work.

5 Related Work

Neural Machine Translation A number of attention-based neural architectures have proven to be very effective for NMT. For RNMT models,

both the encoder and decoder were implemented as deep Recurrent Neural Networks (RNNs), interacting via a soft-attention mechanism (Bahdanau et al., 2014; Chen et al., 2018). It is the pioneering paradigm achieving the state-of-the-art performance. Following RNMT, convolutional sequence-to-sequence (ConvS2S) models take advantages of modern fast computing devices which outperform RNMT with faster training speed (Kalchbrenner et al., 2016; Gehring et al., 2017a). Most recently, the Transformer model, which is based solely on a self-attention mechanism and feed-forward connections, has further advanced the field of NMT, both in terms of translation quality and speed of convergence (Vaswani et al., 2017; Dehghani et al., 2018). The attention mechanism plays a crucial role in the success of all these models to achieve the state-of-the-art results, as the memory capacity of a single dense vector in the typical encoder-decoder model seems not powerful enough to store the necessary information of the source sentence. Despite the generally good performance, the attention based models have running time that is super-linear in the length of the source sequences, burdening the inference speed as the length of the sequences increases. Different from the attention based approach, CAPSNMT runs in time that is linear in the length of the sequences.

Linear Time Neural Machine Translation Several papers have proposed to use neural networks to directly learn the conditional distribution from a parallel corpus(Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Kalchbrenner et al., 2016). In (Sutskever et al., 2014), an RNN with LSTM units was used to encode a source sentence and starting from the last hidden state, to decode a target sentence. Similarly, the authors of Cho et al., (2014) proposed to use an RNN to encode and decode a pair of source and target phrases. Different the RNN based approach, Kalchbrenner et al., (2016) propose ByteNet which makes use of the convolution networks and successfully build a liner-time NMT system. Unlike the previous work to store the source sentence with a bottom-up way, the CAPSNMT encodes the source sentence with an iterative process to decide the credit attribution between nodes from lower and higher layers. As a result, CAPSNMT achieves the best performance among the linear time NMT systems.

Capsule Networks for NLP Currently, much attention has been paid to how developing a sophisticated encoding models to capture the long and short term dependency information in a sequence. Gong et al., (2018) propose an aggregation mechanism to obtain a fixed-size encoding with a dynamic routing policy. Zhao et al., (2018) explore capsule networks with dynamic routing for multi-task learning and achieve the best performance on six text classification benchmarks. Wang et al., (2018) propose RNN-Capsule, a capsule model based on Recurrent Neural Network (RNN) for sentiment analysis. For a given problem, one capsule is built for each sentiment category. To the best of our knowledge, CAPSNMT is the first work that capsule networks have been empirically investigated for sequence-to-sequence problems.

6 Conclusion

We have introduced CAPSNMT with dynamic routing policy for linear-time NMT. Three strategies were proposed to boost the performance of dynamic routing process. We have shown that CAPSNMT is a state-of-the-art encoder-decoder based NMT that outperforms ByteNet while maintaining linear running time complexity. We have also shown that with a carefully designed encoder, the CAPSNMT can achieve comparable results with the state-of-the-art Transform system. To the best of our knowledge, this is the first work that capsule networks have been empirically investigated for sequence to sequence problems.

In the future, we would like to investigate more sophisticated routing policy for better encoding the long sequence. Besides, dynamic routing should also be useful to the decoder of the NMT.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language

models from the common crawl. In *LREC*, volume 2, page 4. Citeseer.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George F Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *meeting of the association for computational linguistics*, 1:76–85.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

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. *arXiv preprint arXiv:1406.1078*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and ÅĄukasz Kaiser. 2018. Universal transformers. *arXiv: Computation and Language*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017a. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017b. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *international conference on computational linguistics*.

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*, pages 770–778.

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *international conference on learning representations*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. 1997. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113.

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

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. *neural information processing systems*, pages 3856–3866.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

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

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

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.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018. Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1165–1174. International World Wide Web Conferences Steering Committee.

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*.

Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. *arXiv: Computation and Language*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199*.