Towards Linear Time Neural Machine Translation with Capsule Networks

Mingxuan Wang[1] **Jun xie**[2] **Zhixing Tan**[2] **Jinsong Su**[3] **Deyi Xiong**[4] **Lei Li**[1]
[1]ByteDance AI Lab, Beijing, China
{wangmingxuan.89,lilei.lab}@bytedance.com
[2]Mobile Internet Group, Tencent Technology Co., Ltd
[3]Xiamen University, Xiamen, China
[4]Tianjin University, Tianjin, China

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. Different from 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 aggregate 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 Transformer system. To the best of our knowledge, this is the first work that capsule networks have been empirically investigated for sequence to sequence problems.[1]

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 which aggravates learning for long sequences. This results in the computational complexity of decoding process being $\mathcal{O}(|S| + |T|)$, 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 represention, 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 the input source sentence into context vectors which are fixed during decoding. Therefore, a natural question was raised, *Will carefully designed aggregation operations help the Enc-Dec paradigm to achieve the best performance?*

In recent promising work of capsule network, a dynamic routing policy is proposed and proven to

[1]The work is partially done when the first author worked at Tencent.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 803–812,
Hong Kong, China, November 3–7, 2019.

effective (Sabour et al., 2017; Zhao et al., 2018; Gong et al., 2018). Following a similar spirit to use this technique, we present CAPSNMT, which is 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, and then introduces 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 summarized 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 propose several techniques 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

From the perspective of machine learning, the task of linear-time translation can be formalized as learning the conditional distribution $p(\mathbf{y}|\mathbf{x})$ of a target sentence (translation) $\mathbf{y}$ given a source sentence $\mathbf{x}$. Figure 1 gives the framework of our proposed linear time NMT with capsule encoder, which mainly consists of two components: a constant encoder that represents the input source sentence with a fixed-number of vectors, and a decoder which leverages these vectors to generate the target-side translation. Please note that due to the fixed-dimension representation of encoder, the time complexity of our model could be linear in the number of source words.

Constant Encoder with Aggregation Layers Given the input source sentence $\mathbf{x} = x_1, x_2 \cdots, x_L$, Then, we introduce capsule networks to transfer

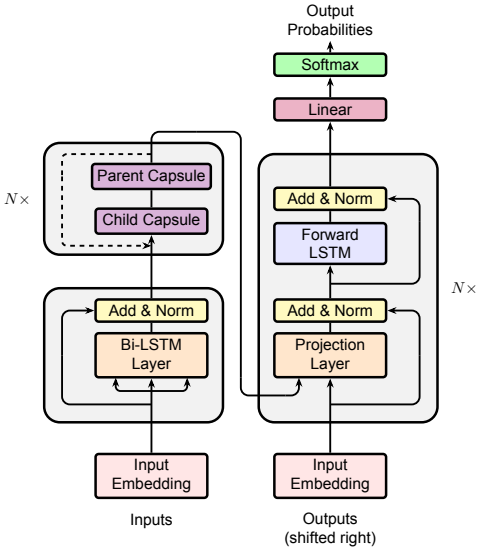

Figure 1: CAPSNMT:Linear time neural machine translation with capsule encoder

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

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 $M < L$ is the pre-determined size of the encoder output, and d_c is the dimension of the hidden states.

We first introduce a bi-directional LSTM (BiLSTM) as the primary-capsule layer to incorporate forward and backward context information of the input sentence:

$$\begin{aligned} \overrightarrow{\mathbf{h}_t} &= \overrightarrow{\mathbf{LSTM}}(\mathbf{h}_{t-1}, \mathbf{x}_t) \\ \overleftarrow{\mathbf{h}_t} &= \overleftarrow{\mathbf{LSTM}}(\mathbf{h}_{t+1}, \mathbf{x}_t) \\ \mathbf{h}_t &= [\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}] \end{aligned} \quad (1)$$

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

On the top of the BiLSTM, we introduce several aggregation layers to map the variable-length inputs into the compressed representation. To demonstrate the effectiveness of our encoder, we

compare it with a more powerful aggregation method, which mainly involves Max and Average pooling.

Both Max and Average pooling are the simplest ways to aggregate information, which require no additional parameters while being computationally efficient. Using these two operations, we perform pooling along the time step as follows:

$$\mathbf{h}^{max} = \max([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]) \quad (2)$$

$$\mathbf{h}^{avg} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{h}_i \quad (3)$$

Moreover, because the last time step state h_L and the first time step state h_1 provide complimentary information, we also exploit them to enrich the final output representation of encoder. Formally, the output representation of encoder consists of four vectors,

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

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 $\mathbf{C}$ is fixed for the subsequent translation generation, therefore its quality directly affects the success of building the Enc-Dec paradigm. In this work, we mainly focus on how to introduce aggregation layers with capsule networks to accurately produce the compressed representation of input sentences, of which details will be provided in Section 3.

LSTM Decoder The goal of the LSTM is to estimate the conditional probability $p(y_t|y_{<t}; x_1, x_2, \cdots, x_T)$, 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):

$$\begin{aligned}\mathbf{s}_t &= \mathbf{LSTM}(\mathbf{s}_{t-1}, \mathbf{u}_t) \\ \mathbf{u}_t &= \mathrm{ATT}(\mathbf{C}, \mathbf{s}_{t-1}) + \mathbf{y}_t\end{aligned} \quad (5)$$

where $\mathbf{y}_t$ is the target word embedding of y_t, $\mathbf{u}_t$ is the inputs of LSTM at time step t, $\mathbf{C}$ is the concatenation of the source sentence representation and $\mathrm{ATT}(\cdot)$ is the traditional attention function.

At inference stage, we only utilize the top-most hidden states $\mathbf{s}_t$ to make the final prediction with a softmax layer:

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

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

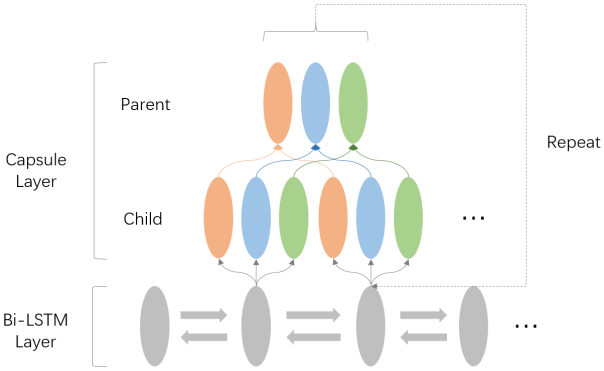

Figure 2: Capsule encoder with dynamic routing by agreement

The aggregation layers with capsule networks play a crucial role in our model. As shown in Figure 2, unlike the traditional linear-time approaches collecting information in a bottom-up approach without considering the state of the whole encoding, our capsule layer is able to iteratively decide the information flow, where the part-whole relationship of the source sentence can be effectively captured.

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 is an ideal solution which is able 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, $\mathbf{u}_{i\to j}$ denotes the information be transferred from the child capsule $\mathbf{h}_i$ into the

parent capsule $\mathbf{c}_j$:

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

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:

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

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

Finally, the parent capsule aggregates all the incoming messages from all the child capsules:

$$\mathbf{v}_i = \sum_{j=1}^{L} \mathbf{u}_{j\to i} \tag{9}$$

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. However we find 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.

$$\begin{aligned}\mathbf{c}_i &= \mathbf{squash}(\mathbf{v}_i) \\ &= \frac{||\mathbf{v}_i||^2}{1+||\mathbf{v}_i||^2}\frac{\mathbf{v}_i}{||\mathbf{v}_i||}\end{aligned} \tag{10}$$

3.2 Dynamic Routing by Agreement

The dynamic routing process is implemented via an EM iterative process of refining the coupling coefficient α_{ij} , which measures 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

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

where $\sum \alpha_j \alpha ij$. This ensures that all the information from the child capsule $\mathbf{h}_i$ will be transferred to the parent.

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

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 the scalar product of $\mathbf{c}_j^{t-1}$ and $\mathbf{f}(\mathbf{h}_j^{t-1}, \boldsymbol{\theta}_j^{t-1})$, which is essentially the similarity between the input to the capsule and the output from the capsule. Likewise, remember from above, the lower level capsule will send its output to the higher level capsule with similar output. Particularly, b_{ij}^0 is initialized with 0. The coefficient depends on the location and type of both the child and the parent capsules, which iteratively refinement of b_{ij}. The capsule network can increase or decrease the connection strength by dynamic routing. It is more effective than the previous routing strategies such as max-pooling which essentially detects whether a feature is present in any position of the text, but loses spatial information about the feature.

Algorithm 1 Dynamic Routing Algorithm

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

When an output capsule $\mathbf{c}_j^t$ receives the incoming messages $\mathbf{u}_{i\to j}^t$, its state will be updated and the coefficient α_{ij}^t is also re-computed for the input capsule $\mathbf{h}_i^{t-1}$. Thus, we iteratively refine the route of information flowing, towards an instance dependent and context aware encoding of a sequence. After the source input is encoded into M capsules, we map these capsules into vector representation by simply concatenating all capsules:

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

Finally matrix $\mathbf{C}$ will then be fed to the final end to end NMT model as the source sentence encoder.

In this work, we also explore three strategies to improve the accuracy of the routing process.

Position-aware Routing strategy The routing process iteratively decides what and how much information is to be sent to the final encoding with considering the state of both the final outputs cap-

sule and the inputs capsule. In order to fully exploit the order of the child and parent capsules to capture the child-parent relationship more efficiently, we add "positional encoding" to the representations of child and parent capsules. In this way, some information can be injected according to the relative or absolute position of capsules in the sequence. In this aspect, there are many choices of positional encoding proposed in many NLP tasks (Gehring et al., 2017b; Vaswani et al., 2017). Their experimental results strongly demonstrate that adding positional information in the text is more effective than in image since there is some sequential information in the sentence. In this work, we follow (Vaswani et al., 2017) to apply sine and cosine functions of different frequencies.

Non-sharing Weight Strategy Besides, we explore two types of transformation matrices to generate the message vector $\mathbf{u}_{i\to j}$ which has been previously mentioned Eq.(7,8). The first one shares parameters $\boldsymbol{\theta}$ 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. We will compare the effects of these two strategies in our experiments. In our preliminary, we found that non-shared weight strategy works slightly better than the shared one which is in consistent with (Liao and Poggio, 2016).

Separable Composition and Scoring strategy The most important idea behind capsule networks is to measure the input and output similarity. It is often modeled as a dot product function between the input and the output capsule, and the routing coefficient is updated correspondingly. Traditional capsule networks often resort to 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, however, is undesired for maintaining the composition functionality of capsules. Here, we propose to employ separate functional networks to release the scoring duty, and let $\boldsymbol{\theta}$ defined in Eq.(7) be responsible for composition. More specifically, we redefined the iteratively scoring function in Eq.(12) as follow,

$$b_{ij}^t = b_{ij}^{t-1} + \mathbf{g}(\mathbf{c}_j^t) \cdot \mathbf{g}(\mathbf{h}_i^t). \qquad (14)$$

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

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 batched sentence pairs by approximate length, and limited input and output tokens per batch to 4096 per GPU.

During training, we employed label smoothing of value $\epsilon = 0.1$(Pereyra et al., 2017). We used a beam width of 8 and length penalty $\alpha = 0.8$ in all the experiments. 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

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
CapsNMT systems				
Base Model	Simple Aggregation	c$\|S\|$+c$\|T\|$	37.1	21.3
Base Model	CapsNMT	c$\|S\|$+c$\|T\|$	39.6	25.7
Big Model	CapsNMT	c$\|S\|$+c$\|T\|$	**40.0**	**26.4**
Base Model	Simple Aggregation + KD	c$\|S\|$+c$\|T\|$	38.6	23.4
Base Model	CapsNMT + KD	c$\|S\|$+c$\|T\|$	40.4	26.9
Big Model	CapsNMT+ KD	c$\|S\|$+c$\|T\|$	**40.6**	**27.6**

Table 1: Case-sensitive BLEU scores on English-German and English-French translation. KD indicates knowledge distillation (Kim and Rush, 2016).

for the big model, the dimensions were set to 1024. The capsule number is set to 6.

Sequence-level knowledge distillation is applied to alleviate multimodality in the training dataset, using the state-of-the-art transformer big models as the teachers(Kim and Rush, 2016). We decode the entire training set once using the teacher to create a new training dataset for its respective student.

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 comparable BLEU score among all the with Big Transform, a relative strong competitor with a gap of only 0.4 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. Knowledge also helps a lot to bridge the performance gap between CapsNMT and the state-of-the-art transformer model.

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 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 conclusions:

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.

- **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 can still achive an increase of +1.2 BLEU.

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.

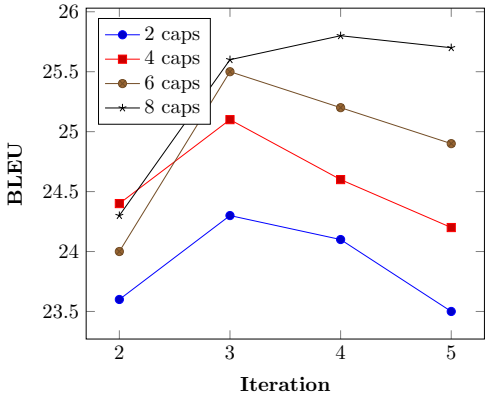

Figure 3: Effects of Iterative Routing with different capsule numbers

Model	Num	Latency(ms)
Transformer	-	225
CAPSNMT	4	146
CAPSNMT	6	153
CAPSNMT	8	168

Table 3: Time required for decoding with the base model. Num indicates the capsule number. 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 CAP-

sNMT in Table 3. 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 degradation 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.

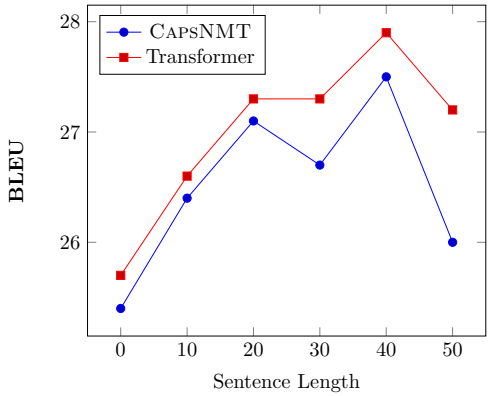

Figure 4: The plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length.

Orlando Bloom and Miranda Kerr still love each other
Orlando Bloom and Miranda Kerr still love each other
Orlando Bloom and Miranda Kerr still love each other
Orlando Bloom and Miranda Kerr still love each other

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

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 was used to encode a source sentence and starting from the last hidden state, to decode a target sentence. Different the RNN based approach, Kalchbrenner et al., (2016) propose ByteNet which makes use of the convolution networks to build the liner-time NMT system. Unlike the previous work, the CAPSNMT encodes the source sentence with an iterative process to decide the credit attribution between nodes from lower and higher layers.

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. 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 for linear-time NMT. Three strategies were proposed to boost the performance of dynamic routing process. The emperical results show that CAPSNMT can achieve state-of-the-art results with better decoding latency in several benchmark datasets.

Acknowledgement

We thank the three anonymous reviewers for their valuable suggestions. Deyi Xiong was supported by the National Natural Science Foundation of China (Grant No. 61622209 and 61861130364)

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.

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*.

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.

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*.

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.

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*.