Modeling Recurrence for Transformer

Jie Hao
Florida State University
haoj8711@gmail.com

Xing Wang
Tencent AI Lab
brightxwang@tencent.com

Baosong Yang
University of Macau
nlp2ct.baosong@gmail.com

Longyue Wang
Tencent AI Lab
vinnylywang@tencent.com

Jinfeng Zhang
Florida State University
jinfeng@stat.fsu.edu

Zhaopeng Tu*
Tencent AI Lab
zptu@tencent.com

Abstract

Recently, the Transformer model (Vaswani et al., 2017) that is based solely on attention mechanisms, has advanced the state-of-the-art on various machine translation tasks. However, recent studies reveal that the lack of recurrence hinders its further improvement of translation capacity (Chen et al., 2018; Dehghani et al., 2019). In response to this problem, we propose to directly model recurrence for Transformer with an additional recurrence encoder. In addition to the standard recurrent neural network, we introduce a novel *attentive recurrent network* to leverage the strengths of both attention and recurrent networks. Experimental results on the widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation tasks demonstrate the effectiveness of the proposed approach. Our studies also reveal that the proposed model benefits from a *short-cut* that bridges the source and target sequences with a single recurrent layer, which outperforms its deep counterpart.

1 Introduction

Recently, Transformer (Vaswani et al., 2017) – a new network architecture based solely on attention mechanisms, has advanced the state-of-the-art on various translation tasks across language pairs. Compared with the conventional recurrent neural network (RNN) (Schuster and Paliwal, 1997) based model that leverages recurrence as the basic building module (Sutskever et al., 2014; Bahdanau et al., 2015; Chen et al., 2018), Transformer replaces RNN with self-attention network (SAN) to model the dependencies among input elements. One appealing strength of SAN is that it breaks down the sequential assumption to obtain the ability of highly parallel computation: input elements interact with each other simultaneously without regard to their distance.

However, prior studies empirically show that the lack of recurrence modeling hinders Transformer from further improvement of translation quality (Dehghani et al., 2019). Modeling recurrence is crucial for capturing several essential properties of input sequence, such as structural representations (Tran et al., 2016) and positional encoding (Shaw et al., 2018), which are exactly the weaknesses of SAN (Tran et al., 2018). Recently, Chen et al. (2018) show that the representations learned by SAN-based and RNN-based encoders are complementary to each other, and merging them can improve translation performance for RNN-based NMT models.

Starting from these findings, we propose to directly model recurrence for Transformer with an additional recurrence encoder. The recurrence encoder recurrently reads word embeddings of input sequence and outputs a sequence of hidden states, which serves as an additional information source to the Transformer decoder. In addition to the standard RNN, we propose to implement recurrence modeling with a novel *attentive recurrent network* (ARN), which combines advantages of both SAN and RNN. Instead of recurring over the individual symbols of sequences like RNN, the ARN recurrently revises its representations over a set of feature vectors, which are extracted by an attention model from the input sequence. Accordingly, ARN combines the strong global modeling capacity of SAN with the recurrent bias of RNN.

We evaluate the proposed approach on widely-used WMT14 English⇒German and WMT17 Chinese⇒English translation tasks. Experimental results show that the additional recurrence encoder, implemented with either RNN or ARN,

* Zhaopeng Tu is the corresponding author of the paper. This work was conducted when Jie Hao and Baosong Yang were interning at Tencent AI Lab.

Proceedings of NAACL-HLT 2019, pages 1198–1207
Minneapolis, Minnesota, June 2 - June 7, 2019.

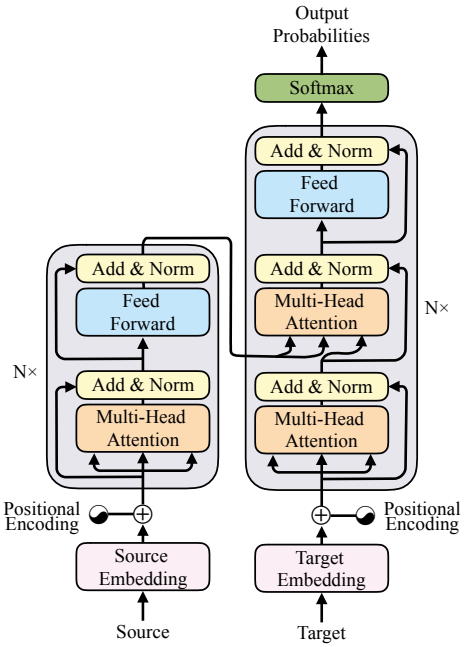

Figure 1: The architecture of Transformer.

consistently improves translation performance, demonstrating the necessity of modeling recurrence for Transformer. Specifically, the ARN implementation outperforms its RNN counterpart, which confirms the strength of ARN.

Further analyses reveal that our approach benefits from a *short-cut* that bridges the source and target sequences with shorter path. Among all the model variants, the implementation with shortest path performs best, in which the recurrence encoder is single layer and its output is only fed to the top decoder layer. It consistently outperforms its multiple deep counterparts, such as multiple-layer recurrence encoder and feeding the output of recurrence encoder to all the decoder layers. In addition, our approach indeed generates more informative encoder representations, especially representative on syntactic structure features, through conducting linguistic analyses on probing tasks (Conneau et al., 2018).

2 Background

Figure 1 shows the model architecture of Transformer. The encoder is composed of a stack of N identical layers, each of which has two sub-layers. The first sub-layer is a self-attention network, and the second one is a position-wise fully connected feed-forward network. A residual connection (He et al., 2016) is employed around each of two sub-layers, followed by layer normalization (Ba et al., 2016). Formally, the output of the first sub-layer $\mathbf{C}_e^n$ and the second sub-layer $\mathbf{H}_e^n$ are sequentially calculated as:

$$\mathbf{C}_e^n = \textsc{Ln}\big(\textsc{Self-Att}(\mathbf{H}_e^{n-1}) + \mathbf{H}_e^{n-1}\big), \quad (1)$$

$$\mathbf{H}_e^n = \textsc{Ln}\big(\textsc{Ffn}(\mathbf{C}_e^n) + \mathbf{C}_e^n\big), \quad (2)$$

where $\textsc{Self-Att}(\cdot)$, $\textsc{Ln}(\cdot)$, and $\textsc{Ffn}(\cdot)$ are respectively self-attention mechanism, layer normalization, and feed-forward network with ReLU activation in between.

In transformer, $\textsc{Self-Att}(\cdot)$ computes attention over the input $\mathbf{H}_e^{n-1}$ as follows:

$$\textsc{Self-Att}(\mathbf{H}_e^{n-1}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \quad (3)$$

where $\{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}$ are query, key and value vectors that are transformed from the input representations $\mathbf{H}_e^{n-1}$. $\sqrt{d_k}$ is the scaling factor where the d_k is the dimension size of the query and key vectors.

The decoder is also composed of a stack of N identical layers. In addition to two sub-layers in each decoder layer, the decoder inserts a third sub-layer $\mathbf{D}_d^n$ to perform attention over the output of the encoder $\mathbf{H}_e^N$:

$$\mathbf{C}_d^n = \textsc{Ln}\big(\textsc{Self-Att}(\mathbf{H}_d^{n-1}) + \mathbf{H}_d^{n-1}\big), \quad (4)$$

$$\mathbf{D}_d^n = \textsc{Ln}\big(\textsc{Att}(\mathbf{C}_d^n, \mathbf{H}_e^N) + \mathbf{C}_d^n\big), \quad (5)$$

$$\mathbf{H}_d^n = \textsc{Ln}\big(\textsc{Ffn}(\mathbf{D}_d^n) + \mathbf{D}_d^n\big), \quad (6)$$

where $\textsc{Att}(\mathbf{C}_d^n, \mathbf{H}_e^N)$ denotes attending the top encoder layer $\mathbf{H}_e^N$ with $\mathbf{C}_d^n$ as query. The top layer of the decoder $\mathbf{H}_d^N$ is used to generate the final output sequence.

3 Approach

In this section, we first describe the architecture of the introduced recurrence encoder and elaborate two types of neural network that are used as recurrence encoder in this work. Then we introduce the integration of recurrence encoder into the Transformer. Specifically, two strategies are presented to fuse the representations produced by the recurrence encoder and the conventional encoder. Finally we present the *short-cut* connection between the recurrence encoder and the decoder that we found very effective to use the learned representation to improve the translation performance under the proposed architecture.

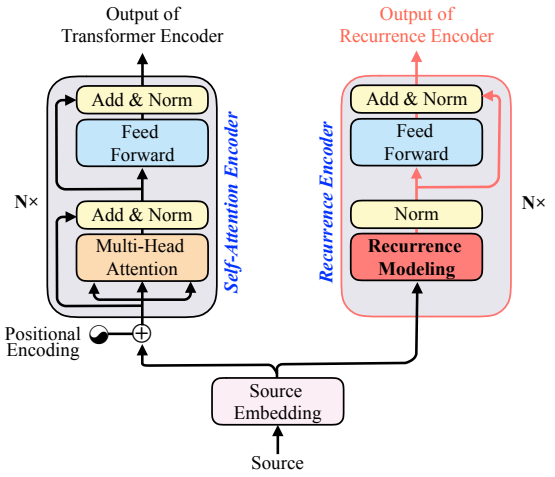

Figure 2: The architecture of Transformer augmented with an additional *recurrence encoder*, the output of which is directly fed to the top decoder layer.

3.1 Recurrence Modeling

Figure 2 shows the architecture of the introduced *recurrence encoder* which reads word embeddings of source words and outputs a sequence of hidden states that embeds recurrent information. Similar to the Transformer encoder, it has a stack of N identical layers, each of which has two sub-layers. The first one is a recurrence modeling network and the second is a fully connected feed-forward network:

$$\mathbf{C}_r^n = \textsc{Ln}(\textsc{Rec}(\mathbf{H}_r^{n-1}) + \mathbf{H}_r^{n-1}), \quad (7)$$
$$\mathbf{H}_r^n = \textsc{Ln}(\textsc{Ffn}(\mathbf{C}_r^n) + \mathbf{C}_r^n), \quad (8)$$

where $\textsc{Rec}(\cdot)$ is the function of recurrence modeling. Note that at the bottom layer of the recurrence encoder (N=1), we do not employ a residual connection on the recurrence sub-layer (i.e. Equation 7), which releases the constraint that $\mathbf{C}_r^1$ should share the same length with input embeddings sequence $\mathbf{E}_{in}$[1]. This offers a more flexible choice of the recurrence functions.

There are many possible ways to implement the general idea of recurrence modeling $\textsc{Rec}(\cdot)$. The aim of this paper is not to explore this whole space but simply to show that some fairly straightforward implementations work well. In this work, we investigate two representative implementations, namely RNN and its variation *attentive current network* that combines advantages of both RNN and attention models, as shown in Figure 3.

[1]The input of the lowest layer in the recurrence encoder is the word embeddings of input sequence $\mathbf{E}_{in}$.

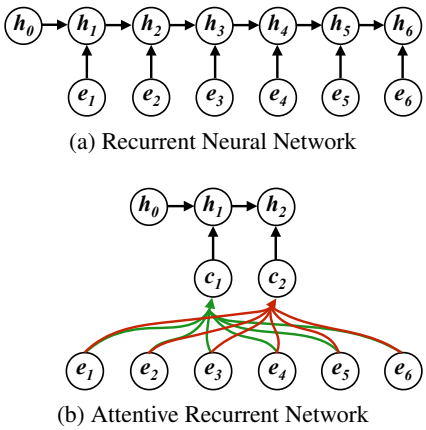

Figure 3: Two implementations of recurrence modeling: (a) standard RNN, and (b) the proposed ARN.

Recurrent Neural Network (RNN) An intuitive choice of recurrence modeling is RNN, which is a standard network to model sequence orders. In this work, we use a bidirectional RNN (BiRNN), which is widely applied in RNN-based NMT models (Bahdanau et al., 2015; Chen et al., 2018). Each hidden state in the output representations $\mathbf{H}_{\text{RNN}}^n = \{\mathbf{h}_1^n, \ldots, \mathbf{h}_J^n\}$ is calculated as

$$\mathbf{h}_j^n = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j], \quad (9)$$
$$\overrightarrow{\mathbf{h}}_j = \overrightarrow{f}(\overrightarrow{\mathbf{h}}_{j-1}, \mathbf{h}_j^{n-1}), \quad (10)$$
$$\overleftarrow{\mathbf{h}}_j = \overleftarrow{f}(\overleftarrow{\mathbf{h}}_{j+1}, \mathbf{h}_j^{n-1}), \quad (11)$$

where $\overrightarrow{f}(\cdot)$ and $\overleftarrow{f}(\cdot)$ are the activation functions of forward and backward RNN respectively, which can be implemented as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014). $\mathbf{h}_0^n$ is the initial state of RNN, which is the mean of $\mathbf{H}_{\text{RNN}}^{n-1}$. $\mathbf{H}_{\text{RNN}}^0$ represents the word embeddings of the input sequence.

Attentive Recurrent Network (ARN) We can also extend RNN by recurring over a set of feature vectors extracted with an attention model, which allows the model to learn a compact, abstractive feature vectors over the input sequence. Specifically, the ARN performs T recurrent steps on the attentive output of the input representation $\mathbf{H}_r^{n-1}$:

$$\mathbf{h}_t^n = f(\mathbf{h}_{t-1}^n, \mathbf{c}_t^n), \quad (12)$$
$$\mathbf{c}_t^n = \textsc{Att}(\mathbf{h}_{t-1}^n, \mathbf{H}_r^{n-1}). \quad (13)$$

The output representations $\mathbf{H}_{\text{ARN}}^n = \{\mathbf{h}_1^n, \ldots, \mathbf{h}_T^n\}$ are fed to the subsequent modules. Analogous to Equations 9-11, ARN can be extended to the bidirectional variant, i.e. BiARN,

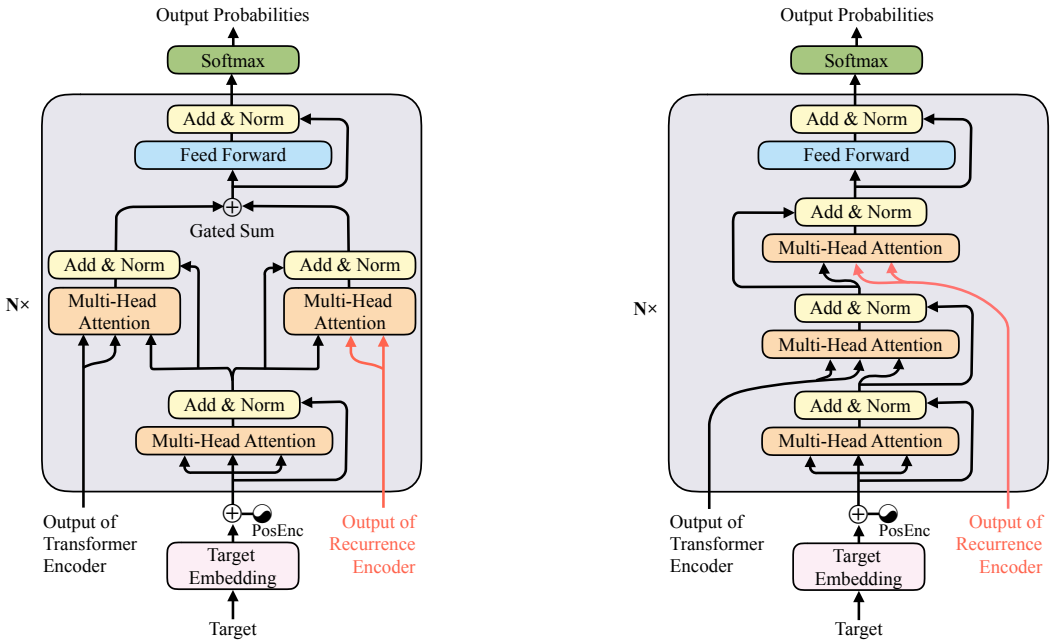

Figure 4: Different strategies to integrate the output of the additional *recurrence encoder* into the decoder.

except that the input is the attentive context vector $\mathbf{c}_t^n$ rather than the individual representation vector of the input sequence.

Note that, the number of recurrence step T is allowed to be unequal to the length of input sequence J. In contrast to RNN which recurs over the individual symbols of the input sequences, ARN recurrently revises its representations of all symbols in the sequence with an attention model.

3.2 Integrating into Transformer

Since the output of recurrence encoder unnecessarily shares the same length with that of Transformer encoder (e.g. when ARN is used as recurrence function), combination strategy on the encoder side, such as concatenating the outputs of both encoders (Chen et al., 2018), is not an universal solution in this scenario. Accordingly, we feed the information of the additional recurrence encoder into the decoder of Transformer. Specifically, we serve an additional attention layer $\mathbf{R}_d^n$ as the fourth sub-layer in each decoder block to perform attention over the output of the recurrence encoder $\mathbf{H}_r^N$. As shown in Figure 4, we present two strategies to integrate $\mathbf{R}_d^n$, namely *gated sum* and *stack*, which differ at how $\mathbf{R}_d^n$ interacts with the output of attention over the Transformer encoder, i.e., $\mathbf{D}_d^n$ in Equation 5.

Gated Sum The first strategy combines the outputs of the two attention sub-layers in a gating fusion (Figure 4(a)), in which the outputs of both encoders are attended simultaneously:

$$\mathbf{R}_d^n = \textsc{Ln}\big(\textsc{Att}(\mathbf{C}_d^n, \mathbf{H}_r^N) + \mathbf{C}_d^n\big), \quad (14)$$

$$\widehat{\mathbf{D}}_d^n = \lambda_n \mathbf{D}_d^n + (1-\lambda_n)\mathbf{R}_d^n, \quad (15)$$

$$\mathbf{H}_d^n = \textsc{Ln}\big(\textsc{Ffn}(\widehat{\mathbf{D}}_d^n) + \widehat{\mathbf{D}}_d^n\big), \quad (16)$$

where λ_n is an interpolation weight calculated by a logistic sigmoid function:

$$\lambda_n = sigmoid(\mathbf{D}_d^n, \mathbf{R}_d^n) \quad (17)$$

As seen, the output of self-attention layer $\mathbf{C}_d^n$ serves as a query to attend the outputs of both encoders (Equations 5 and 14), and the outputs of both attention models $\{\mathbf{D}_d^n, \mathbf{R}_d^n\}$ are combined via a gated sum (Equation 15), which is subsequently fed to the feed-forward layer (Equation 16).

Stack We can also arrange the sub-layers in a *stack* (Figure 4(b)), in which the outputs of both encoders are attended sequentially:

$$\mathbf{R}_d^n = \textsc{Ln}\big(\textsc{Att}(\mathbf{D}_d^n, \mathbf{H}_r^N) + \mathbf{D}_d^n\big), \quad (18)$$

$$\mathbf{H}_d^n = \textsc{Ln}\big(\textsc{Ffn}(\mathbf{R}_d^n) + \mathbf{R}_d^n\big), \quad (19)$$

The decoder first attends the output of Transformer encoder, and the attention output $\mathbf{D}_d^n$ serves as the query to attend the output of recurrence encoder (Equation 18).

3.3 Short-Cut Effect

The introduced recurrence encoder provides an additional computation path ranging from the input sequence to the output sequence. Chung et al. (2017) and Shen et al. (2019) have shown that a shortcut for gradient back-propagation benefits language modeling. Inspired from them, we use a shorter path to transform the learned recurrence. We call this the "*short-cut effect*".

Among all the model variants, we implement shortest path as: the recurrence encoder is single layer and its output is only fed to the top decoder layer while the first $N-1$ decoder layers perform the same as the standard Transformer (e.g. Equations 4-6). Accordingly, the computation path is $\mathbf{E}_{in} \rightarrow \mathbf{H}_r \rightarrow \mathbf{R}_d^N \rightarrow \mathbf{H}_d^N$, then the decoder uses $\mathbf{H}_d^N$ to make a target word prediction. It is much simpler than that of the conventional Transformer, which transfers information learned from input sequences across multiple stacking encoder and decoder layers. We expect it outperforms its multiple deep counterparts, such as multiple-layer recurrence encoder and feeding the output of recurrence encoder to all the decoder layers.

4 Related Work

Improving Transformer Encoder From the perspective of representation learning, there has been an increasing amount of work on improving the representation power of SAN encoder. Bawden et al. (2018) and Voita et al. (2018) exploit external context for SAN encoder, while Yang et al. (2019) leverage the intermediate representations to contextualize the transformations in SAN. A number of recent efforts have explored ways to improve multi-head SAN by encouraging individual attention heads to extract distinct information (Strubell et al., 2018; Li et al., 2018). Concerning multi-layer SAN encoder, Dou et al. (2018, 2019) and Wang et al. (2018) propose to aggregate the multi-layer representations, and Dehghani et al. (2019) recurrently refine these representations. Our approach is complementary to theirs, since they focus on improving the representation power of SAN encoder, while we aim to complement SAN encoder with an additional recurrence encoder.

Along the direction of modeling recurrence for SAN, Vaswani et al. (2017) and Shaw et al. (2018) inject absolute position encoding and relative positional encoding to consider the position information respectively. Shen et al. (2018) introduce a directional self-attention network (DiSAN), which allows each token to attend to previous (or following) tokens only. Both studies verify the necessity of modeling recurrence for SAN. We re-implemented these approaches on top of Transformer, and experimental results show that our approach outperforms them by explicitly augmenting Transformer with an additional recurrence encoder. It should be emphasized that our approach is complementary to theirs, and combining them together is expected to further improve performance, which we leave for future work.

Closely related to our work, Chen et al. (2018) propose to combine SAN encoder with an additional RNN encoder. The main differences between our work and theirs are: 1) we enhance the state-of-the-art Transformer with recurrence information, while Chen et al. (2018) augment RNN-based models with SAN encoder. To this end, we propose a novel attentive recurrent network to implement the additional recurrence encoder in Transformer. We re-implemented the approach proposed by Chen et al. (2018) on top of Transformer. Experimental results indicate the superiority of our approach, which confirms our claim. In addition, we elaborately design the integration strategy to effectively feed the recurrence information to the decoder, and empirically show that the proposed model benefits from the *short-cut* effect.

Comparison to Reviewer Network Attentive recurrent network are inspired by the reviewer network, which is proposed by Yang et al. (2016) for the image caption generation task. There are two key differences which reflect how we have generalized from the original model. First, we perform attention steps over the source embeddings instead of the encoder representations. The main reason is that the Transformer encoder is implemented as multiple layers, and higher layers generally encode global information, as indicated by Peters et al. (2018). Second, we feed the feature vectors together with the original encoder representations to the decoder. In image caption generation, the source side (i.e. image) contains much more information than the target side (i.e. caption) (Tu et al., 2017). Therefore, they aim at learning a compact and abstractive representation from the source information, which serves as the only input to the decoder. In this work, we focus on leveraging the attention model to better learn

the recurrence, which we expect to complement the Transformer model. In our preliminary experiments, attending over the encoder representations does not improve performance, while feeding the feature vectors only to the decoder seriously harms performance.

5 Experiment

5.1 Setup

We conducted experiments on the widely-used WMT14 English-to-German (4.6M sentence pairs, En⇒De) and WMT17 Chinese-to-English (20.6M sentence pairs, Zh⇒En) translation tasks. All the data had been tokenized and segmented into subword symbols using byte-pair encoding (Sennrich et al., 2016) with 32K merge operations[2]. We used case-sensitive NIST BLEU score (Papineni et al., 2002) as the evaluation metric, and *bootstrap resampling* (Koehn et al., 2003) for statistical significance test.

We implemented the proposed approaches on top of the Transformer model (Vaswani et al., 2017). Both in our model and related model of Subsection 5.3, the RNN is implemented with GRU (Cho et al., 2014) for fair comparison. We followed the configurations in Vaswani et al. (2017), and reproduced their reported results on the En⇒De task.

We initialized parameters of the proposed models by the pre-trained baseline model. We have tested both *Base* and *Big* models, which differ at hidden size (512 vs. 1024), filter size (2048 vs. 4096), and number of attention heads (8 vs. 16). In consideration of computation cost, we studied model variations with *Base* model on En⇒De task, and evaluated overall performances with both *Base* and *Big* models on both En⇒De and Zh⇒En translation tasks.

5.2 Impact of Components

In this subsection, we conducted ablation studies to evaluate the different implementations of the proposed model, e.g., recurrence encoder and integration strategy, under the proposed architecture.

Effect of Recurrence Modeling We first investigated the effect of recurrence encoder implementations, as listed in Table 1. We observed that introducing an additional recurrence encoder improves translation performance in all cases.

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

Model	Rec. Encoder	Speed	BLEU
BASE	n/a	1.28	27.31
OURS	6-Layer BIRNN	1.10	27.54
	6-Layer BIARN	1.09	27.72
	3-Layer BIARN	1.15	28.10
	1-Layer BIARN	1.24	**28.21**

Table 1: Evaluation of recurrence encoder implementations. The output of recurrence encoder is fed to the top decoder layer in a stack fusion. "Speed" denotes the training speed (steps/second).

Model	Integration	to Dec.	BLEU
BASE	n/a	n/a	27.31
OURS	Gated Sum	Top	28.12
	Gated Sum	All	28.02
	Stack	Top	**28.21**
	Stack	All	27.93

Table 2: Evaluation of decoder integration strategies.

Among all model variations, BIARN outperforms its BIRNN counterpart.

Concerning BIARN models, reducing the layers consistently improves performance. Specifically, the 1-Layer BIARN achieves the best performances in both translation quality and training speed. This confirms the claim that the proposed approach benefits from a *short-cut* on gradient back-propagation. Accordingly, we adopted 1-Layer BIARN as the default setting in the following experiments.

Effect of Integration Strategies We then tested the effect of different integration strategies, as showed in Table 2. We have two observations. First, feeding only to the top decoder layer consistently outperforms feeding to all decoder layers with different integration strategies. This empirically reconfirms the short-cut effect. Second, the stack strategy marginally outperforms its gated sum counterpart. Therefore, in the following experiments, we adopted the "Stack + Top" model in Table 2 as defaulting setting.

5.3 Results

Performances across Languages Finally, we evaluated the proposed approach on the widely used WMT17 Zh⇒En and WMT14 En⇒De data, as listed in Table 3.

To make the evaluation convincing, we reviewed the prior reported systems, and built strong

System	Architecture	Zh⇒En		En⇒De	
		# Para.	BLEU	# Para.	BLEU
Existing NMT systems					
(Vaswani et al., 2017)	TRANSFORMER-BASE	n/a	n/a	65M	27.3
	TRANSFORMER-BIG	n/a	n/a	213M	28.4
(Hassan et al., 2018)	TRANSFORMER-BIG	n/a	24.2	n/a	n/a
(Chen et al., 2018)	RNMT + SAN Encoder	n/a	n/a	n/a	28.84
Our NMT systems					
this work	TRANSFORMER-BASE	107.9M	24.13	88.0M	27.31
	+ 1-Layer BIARN	+9.4M	24.70⇑	+9.4M	28.21⇑
	TRANSFORMER-BIG	303.9M	24.56	264.1M	28.58
	+ 1-Layer BIARN	+69.4M	25.10⇑	+69.4M	28.98↑

Table 3: Comparing with the existing NMT systems on WMT17 Zh⇒En and WMT14 En⇒De test sets. "↑ / ⇑": significant over the conventional self-attention counterpart ($p < 0.05/0.01$), tested by bootstrap resampling.

Model	BLEU
TRANSFORMER-BASE	27.31
+ RELPOS	27.64
+ DISAN	27.58
+ RNN Encoder	27.47
+ BIARN Encoder (OURS)	**28.21**

Table 4: Comparison with re-implemented related work: "RELPOS": relative position encoding (Shaw et al., 2018), "DISAN": directional SAN (Shen et al., 2018), "RNN Encoder": combining SAN and RNN encoders with multi-column strategy (Chen et al., 2018).

baselines which outperform the reported results on the same data. As seen in Table 3, modeling recurrence consistently improves translation performance across model variations (BASE and BIG models) and language pairs (Zh⇒En and En⇒De), demonstrating the effectiveness and universality of our approach.

Comparison with Previous Work In order to directly compare our approach with the previous work on modeling recurrence, we re-implemented their approaches on top of the TRANSFORMER-BASE in WMT14 En⇒De translation task. For relative position encoding, we used unique edge representations per layer and head with clipping distance $k = 16$. For the DiSAN strategy, we applied a mask to the TRANSFORMER encoder, which constrains the SAN to focus on forward or backward elements. For the multi-column encoder, we re-implemented the additional encoder with six RNN layers.

Table 4 lists the results. As seen, all the recurrence enhanced approaches achieve improvements over the baseline model TRANSFORMER-BASE, which demonstrates the necessity of modeling recurrence for TRANSFORMER. Among these approaches, our approach (*i.e.,* 1-Layer BIARN Encoder) achieves the best performance.

5.4 Analysis

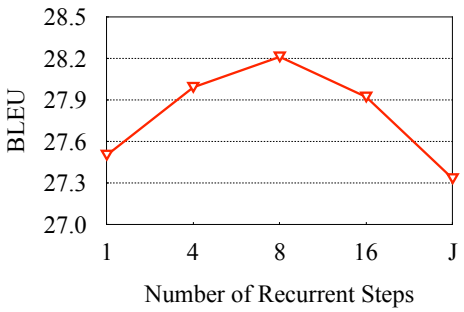

Figure 5: Effect of recurrent steps. The recurrence encoder is implemented as a single-layer BIARN. J denotes the length of the input sequence.

Effect of Recurrent Steps To verify the recurrence effect on the proposed model, we conducted experiments with different recurrent steps on single-layer BIARN model. As shown in Figure 5, the BLEU score typically goes up with the increase of the recurrent steps, while the trend does not hold when $T > 8$. This finding is consistent with Yang et al. (2016), which indicates that conducting too many recurrent steps fails to generate a compact representation. This is exactly one of the ARN's strengths.

Model	Surface		Syntactic			Semantic				
	SeLen	WC	TrDep	ToCo	BShif	Tense	SubN	ObjN	SoMo	CoIn
BASE	**92.20**	63.00	44.74	79.02	71.24	**89.24**	84.69	84.53	52.13	62.47
6-Layer BIRNN	89.90	**77.46**	44.47	79.55	71.53	89.17	**85.99**	84.96	51.75	61.92
6-Layer BIARN	89.78	72.02	44.45	79.21	71.31	88.38	85.64	**85.00**	53.27	62.38
3-Layer BIARN	89.80	72.61	44.28	79.43	71.84	88.93	85.79	84.99	53.30	62.42
1-Layer BIARN	90.91	73.68	**45.15**	**79.62**	**72.21**	89.00	85.54	84.54	**53.44**	**62.71**

Table 5: Classification accuracies on 10 probing tasks of evaluating linguistics embedded in the encoder outputs.

Linguistic Analyses In this section, we conducted 10 probing tasks[3] to study what linguistic properties are captured by the encoders (Conneau et al., 2018). A probing task is a classification problem that focuses on simple linguistic properties of sentences. 'SeLen' predicts the length of sentences in terms of number of words. 'WC' tests whether it is possible to recover information about the original words given its sentence embedding. 'TrDep' checks whether an encoder infers the hierarchical structure of sentences. In 'ToCo' task, sentences should be classified in terms of the sequence of top constituents immediately below the sentence node. 'BShif' tests whether two consecutive tokens within the sentence have been inverted. 'Tense' asks for the tense of the main-clause verb. 'SubN' focuses on the number of the main clause's subject. 'ObjN' tests for the number of the direct object of the main clause. In 'SoMo', some sentences are modified by replacing a random noun or verb with another one and the classifier should tell whether a sentence has been modified. 'CoIn' contains sentences made of two coordinate clauses. Half of sentences are inverted the order of the clauses and the task is to tell whether a sentence is intact or modified.

We used the pre-trained encoders of model variations in Table 1 to generate the sentence representations of input, which are used to carry out probing tasks. For the TRANSFORMER-BASE model, the mean of the encoder top layer representations is used as the sentence representation. For the proposed models, which have two encoders, two sentence representations are generated from the same way in base model. To make full use of the learned representations, we combined these two sentence representations via a gate as the final sentence representation to conduct the experiments.

[3]https://github.com/facebookresearch/SentEval/tree/master/data/probing

Table 5 lists the results. Clearly, the proposed models significantly improve the classification accuracies, although there is still considerable difference among different variants. More specifically,

- Concerning surface properties, among the ARN variants, multi-layer ARN inversely decreases the accuracies, while 1-layer ARN consistently improves the accuracies. Considering the related results presented in Table 1 (Row 3-5), we believe that ARN benefits from the shallow structure.

- ARN tends to capture deeper linguistic properties, both syntactic and semantic. Especially, among these probing tasks, 'TrDep' and 'Toco' tasks are related to syntactic structure modeling. As expected, TRANSFORMER augmented with an additional encoders outperforms the baseline model, which demonstrates that the proposed models successfully model the syntactic structure.

6 Conclusion

In this work, we propose to directly model recurrence for Transformer with an additional recurrence encoder. We implement the recurrence encoder with a novel attentive recurrent network as well as RNN. The recurrence encoder is used to generate recurrence representations for the input sequence. To effectively feed the recurrence representations to the decoder to guide the output sequence generation, we study two strategies to integrate the recurrence encoder into the Transformer. To evaluate the effectiveness of the proposed model, we conduct experiments on large-scale WMT14 EN⇒DE and WMT17 ZH⇒EN datasets. Experimental results on two language pairs show that the proposed model achieves significant improvements over the baseline TRANSFORMER. Linguistic analyses on probing tasks

further show that our model indeed generates more informative representations, especially representative on syntactic structure features.

Future work includes validating the proposed model in other tasks, such as reading comprehension, language inference, and sentence classification. Another promising direction is to directly augment Transformer encoder on recurrence modeling without the additional encoder.

Acknowledgments

J.Z. was supported by the National Institute of General Medical Sciences of the National Institute of Health under award number R01GM126558. We thank the anonymous reviewers for their insightful comments.

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. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating Discourse Phenomena in Neural Machine Translation. In *NAACL*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

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 *EMNLP*.

JunYoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical Multiscale Recurrent Neural Networks. In *ICLR*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $\&!\#*$ vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *ICLR*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *EMNLP*.

Ziyi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic layer aggregation for neural machine translation. In *AAAI*.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *ACL*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: directional self-attention network for RNN/CNN-free language understanding. In *AAAI*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *NAACL*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *EMNLP*.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *TACL*.

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 *NIPS*.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *ACL*.

Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. 2018. Multi-layer representation fusion for neural machine translation. In *COLING*.

Baosong Yang, Jian Li, Derek F. Wong, Lidia S. Chao, Xing Wang, and Zhaopeng Tu. 2019. Context-aware self-attention networks. In *AAAI*.

Zhilin Yang, Ye Yuan, Yuexin Wu, Ruslan Salakhutdinov, and William W Cohen. 2016. Review networks for caption generation. In *NIPS*.