Recurrent Attention for Neural Machine Translation

Jiali Zeng[1]*, **Shuangzhi Wu**[1], **Yongjing Yin**[2], **Yufan Jiang**[1], **Mu Li**[1]
[1]Tencent Cloud Xiaowei, Beijing, China
[2]Zhejiang University, Westlake University, Zhejiang, China
{lemonzeng,garyyfjiang,frostwu,ethanlli}@tencent.com
yinyongjing@westlake.edu.cn

Abstract

Recent research questions the importance of the dot-product self-attention in Transformer models and shows that most attention heads learn simple positional patterns. In this paper, we push further in this research line and propose a novel substitute mechanism for self-attention: **R**ecurrent **A**tte**N**tion (RAN). RAN directly learns attention weights without any token-to-token interaction and further improves their capacity by layer-to-layer interaction. Across an extensive set of experiments on 10 machine translation tasks, we find that RAN models are competitive and outperform their Transformer counterpart in certain scenarios, with fewer parameters and inference time. Particularly, when apply RAN to the decoder of Transformer, there brings consistent improvements by about +0.5 BLEU on 6 translation tasks and +1.0 BLEU on Turkish-English translation task. In addition, we conduct extensive analysis on the attention weights of RAN to confirm their reasonableness. Our RAN is a promising alternative to build more effective and efficient NMT models.

1 Introduction

Transformer models have achieved remarkable success in Neural Machine Translation (NMT) (Vaswani et al., 2017; Freitag and Firat, 2020; Fan et al., 2020). One of the most crucial component of Transformer is the dot-product multi-head self-attention, which is essential to learn relationships between words as well as complex structural representations. However, many studies have shown that the pairwise self-attention is over-parameterized and leads to a costly inference (Sanh et al., 2019; Correia et al., 2019; Xiao et al., 2019). Based on these observations, various improved networks are proposed by either pruning negligible heads (Voita et al., 2019; Michel et al., 2019) or replacing self-attention with more efficient one (Xu et al., 2019; Wu et al., 2019; Kitaev et al., 2020; Beltagy et al., 2020).

More recently, several researches take this direction to an extreme by replacing dot-product self-attention with fixed or trainable global position-based attentions (Zhang et al., 2018; Tay et al., 2020; You et al., 2020; Raganato et al., 2020). For example, You et al. (2020) roughly modeled attention weights as hard-coded Gaussian distributions, based on the observation that most heads only focus their attention on a local neighborhood.

Another more representative method is Random Synthesizer proposed by Tay et al. (2020). Different from You et al. (2020), they simply treat attention weights of all heads in each layer as trainable parameters. At inference time, the attention weights are directly retrieved based on the index of the query without dot-product operation. However, these variants are not the ideal alternatives of self-attention due to the unsatisfactory performance.

In this paper, we go further along this research line and show that self-attention is empirically replaceable. We propose a novel attention mechanism: **R**ecurrent **A**tte**N**tion (RAN). Specifically, RAN starts with an unnormalized *Initial Attention Matrix* for each head, which is randomly initialized and trained together with other model parameters. Then we introduce a *Recurrent Transition Module*, which takes the Initial Attention Matrices as the input and refines them by layer-wise interaction between adjacent layers. The motivation of Recurrent Transition Module is based on the observation that attention weights show a regular pattern and have certain correlation across layers (Xiao et al., 2019; He et al., 2020). Our RAN not only discards the expensive pairwise dot-product of self-attention but also exploit correlation between attention weights of different layers, achieving a more efficient and

*Corresponding author.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3216–3225
November 7–11, 2021.

compact NMT model.[1]

To verify the effectiveness of RAN, we conduct experiments on a wide range of translation tasks involving 10 language pairs. Compared with a vanilla Transformer, our RAN shows competitive or better performance with lower latency and fewer parameters. We conduct extensive analysis on the learned RAN weights showing that the learned attention pattern are reasonable and explainable, which gives credit for the improvement.

2 Our Method

In this section, we first give a brief introduction of self-attention and we refer readers to the original paper (Vaswani et al., 2017) for details. Then, we introduce the proposed recurrent attention mechanism in detail.

2.1 Multi-Head Attention

Figure 1 depicts the scaled dot-product self-attention which only details the computation of the k-th head in the l-th encoder layer. Given a sequence of token representations with a length of n, the self-attention model first converts the representations into three matrices $Q_l^k \in \mathbb{R}^{n\times d_k}$, $K_l^k \in \mathbb{R}^{n\times d_k}$ and $V_l^k \in \mathbb{R}^{n\times d_k}$, representing queries, keys, and values, respectively, where d_k is the dimensionality of the vector in the k-th head. Then, the attention matrix is calculated via the dot product of queries and keys followed by rescaling:

$$A_l^k = \frac{Q_l^k \cdot (K_l^k)^T}{\sqrt{d_k}}, \tag{1}$$

where A_l^k is an $n \times n$ matrix. Finally, a softmax operation is applied on this unnormalized attention matrix and then the output is used to compute a weighted sum of values:

$$H_l^k = \text{Softmax}(A_l^k) \cdot V_l^k, \tag{2}$$

where H_l^k is new contextual representations of the l-th layer. This procedure can be implemented with multi-head mechanism by projecting the input into different subspaces which requires extra splitting and concatenation operations. The output is fed into a position-wise feed-forward network to get the final representations of this layer.

While flexible, it has been proven that there exists redundant information with pair-wise calculation, which can be replaced by simpler positional attention patterns. In the next section, we propose an extreme version of multi-head self-attention by totally removing the dot-product.

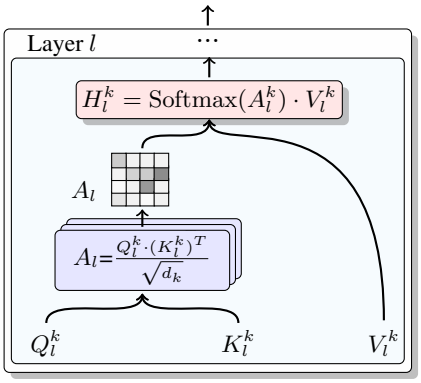

Figure 1: An overview of standard self-attention.

2.2 RAN: Recurrent Attention

We propose a **R**ecurrent **A**tte**N**tion (RAN) as an alternative of multi-head self-attention. RAN consists of a set of global *Initial Attention Matrices* and a *Recurrent Transition Module*. The original self-attention derives key, query from the same token representations and compute attention weights on the fly following Eq. (1), which is repeated in each layer. Our RAN instead learns a set of learnable global attention matrices $\mathbf{A}_0 = \{A_0^1, .., A_0^k, .., A_0^h\}, A_0^k \in \mathbb{R}^{n\times n}$, where h denotes the total number of heads. We denotes $\mathbf{A}_0$ as the *Initial Attention Matrices*, which are initialized together with other parameters. Then, we propose a simple but effective *Recurrent Transition Module*. This module takes $\mathbf{A}_0$ as the input and recursively updated the attention matrices layer by layer. During training, we jointly optimize $\mathbf{A}_0$, the recurrent transition module, and other components. During inference, the attention matrices are completely agnostic to the input representations, and can be retrieved directly without recomputation.

For easier understanding, we illustrate the computation of the k-th head in the l-th encoder layer in Figure 2. Instead of producing attention weights by the dot-product of Q_l^k and K_l^k, we generate the attention matrix A_l^k by the recurrent transition module $\text{Rec}(*)$ with the attention matrix A_{l-1}^k from the previous layer. After obtaining A_l^k, we generate the weighted sum of values by Eq. (2).

We introduce RAN to the encoder self-attention, the decoder self-attention and both of them in our experiment, respectively. We do not consider the cross-attention between encoder and decoder be-

[1]We release source code at https://github.com/lemon0830/RAN.

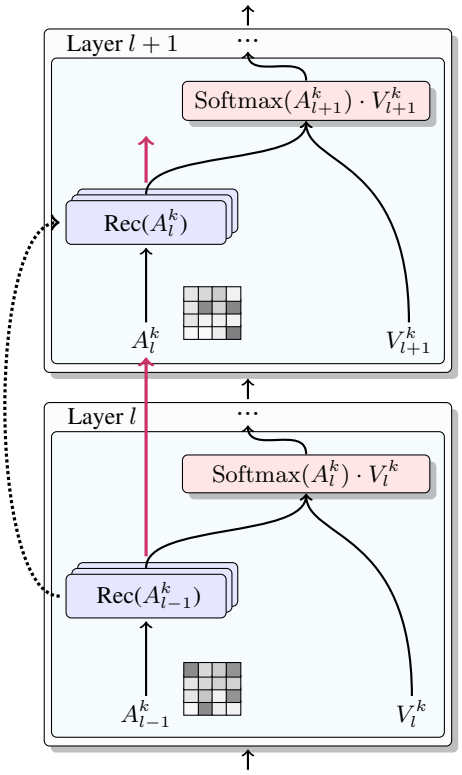

Figure 2: Model architecture of our proposed RAN. Dotted lines indicate parameter sharing.

cause of the poor performance of applying fixed positional attention patterns shown in previous work (You et al., 2020; Tay et al., 2020). When applying RAN to the decoder self-attention, the modeling process is identical to that of the encoder except that only the lower triangular matrix of each *Initial Attention Matrix* is leveraged due to the causal language modeling objective.

2.3 Recurrent Transition Module

We detailedly introduce the composition of the *Recurrent Transition Module* in this section. The transition module can be implemented in various ways such as position-wise feed-forward networks (FFN) (Vaswani et al., 2017), GRU (Cho et al., 2014) or LSTM (Hochreiter and Schmidhuber, 1997). In this paper, we simply use a single feed-forward network with *tanh* as its activation function followed by a layer normalization and a residual connection:

$$\hat{A}^k_l = \text{LN}(\tanh(W \cdot A^k_{l-1} + b)) \quad (3)$$

$$A^k_l = \hat{A}^k_l + A^k_{l-1}. \quad (4)$$

Notably, we share the parameters of the transition module across all heads and all layers.

It is obvious that our RAN has no interaction between queries and keys and thus is more efficient than the dot-product self-attention. In contrast to fixed attention patterns (You et al., 2020; Raganato et al., 2020), the learnable *Initial Attention Matrices* and *Recurrent Transition Module* make the proposed RAN more flexible to learn different attention distribution for different translation tasks. Compared to Random Synthesizer (Tay et al., 2020), our RAN is more likely to learn better context representations thanks to the Recurrent Transition. In terms of parameters, RAN only needs h attention matrices and a linear layer, however, Synthesizer has $h \times L$ attention matrices. Thus RAN is superior in reducing the overall parameters.

3 Experiment

In this section, we evaluate RAN on WMT and NIST translation tasks including 10 different language pairs altogether. We apply RAN to the encoder (*RAN-E*), the decoder (*RAN-D*), or both of them (*RAN-ALL*), respectively. For baselines, we compare against the standard Transformer (*TransF* for short) (Vaswani et al., 2017), and two most related work that are Hard-coded Transformer (*HC-SA*) (You et al., 2020) and Random Synthesizer (*Syn-R*) (Tay et al., 2020).

3.1 Settings

Our corpora come from three sources, and the scales of bilingual corpus range from 210K to 36M:

- WMT2014 (En⇔De, En⇒Fr). We use English-German and English-French corpus, which are comprised of 4.5 and 36 million sentence pairs. We choose newstest 2013 as the valid set and newstest 2014 as the test set.
- WMT2017 (En⇔Lv, En⇔Fi, En⇔Tr). The bidirectional translation tasks of English-Latvian, English-Finnish, English-Turkish consist of 4.46M, 2.63M, and 210K sentence pairs. The setting follows Zhang et al. (2018).
- NIST12 (Zh⇒En). We use parts of the bitext of NIST OpenMT12 [2] as the training set which consists of 1.9 M sentence pairs. The valid data is MT02, and the test sets are MT03, MT04, MT05, MT06, and MT08. We report the average score over all the test sets.

[2]LDC2000T46, LDC2000T47, LDC2000T50, LDC2003E14, LDC2005T10, LDC2002E18, LDC2007T09 and LDC2004T08

Lang.	Matrix	TransF	HC-SA	Syn-R	RAN-E	RAN-D	RAN-ALL
En⇒De	BLEU	27.70	26.94 (-0.76)	27.17 (-0.53)	27.42 (-0.28)	**28.18 (+0.48)**†	27.66 (-0.04)
	SacreBLEU	27.00	26.30 (-0.70)	26.50 (-0.50)	26.80 (-0.20)	**27.50 (+0.50)**†	27.00 (+0.00)
De⇒En	BLEU	**30.97**	29.82 (-1.15)	30.26 (-0.71)	30.74 (-0.23)	30.95 (-0.02)	30.84 (-0.13)
	SacreBLEU	**31.00**	29.80 (-1.20)	30.20 (-0.80)	30.70 (-0.30)	**31.00 (+0.00)**	30.80 (-0.20)
En⇒Fr	BLEU	**42.40**	41.26 (-1.14)	41.33 (-1.07)	42.26 (-0.14)	42.39 (-0.01)	42.05 (-0.25)
	SacreBLEU	**39.40**	38.30 (-1.10)	38.30 (-1.10)	39.30 (-0.10)	**39.40 (+0.00)**	39.10 (-0.30)
En⇒Lv	BLEU	16.78	16.11 (-0.67)	16.67 (-0.11)	16.66 (-0.12)	**17.39 (+0.61)**†	17.00 (+0.22)
	SacreBLEU	16.30	15.60 (-0.70)	16.30 (+0.00)	16.20 (-0.10)	**16.90 (+0.60)**†	16.50 (+0.20)
Lv⇒En	BLEU	18.74	18.41 (-0.33)	18.56 (-0.18)	18.78 (+0.04)	**18.84 (+0.10)**	18.81 (+0.07)
	SacreBLEU	17.40	17.10 (-0.30)	17.30 (-0.10)	17.40 (+0.00)	**17.50 (+0.10)**	**17.50 (+0.10)**
En⇒Fi	BLEU	21.96	21.36 (-0.60)	22.08 (+0.12)	22.46 (+0.50)†	**22.89 (+0.93)**†	22.45 (+0.49)†
	SacreBLEU	20.00	19.30 (-0.70)	20.10 (+0.10)	20.50 (+0.50)†	**20.70 (+0.70)**†	20.50 (+0.50)†
Fi⇒En	BLEU	26.05	25.07 (-0.98)	25.69 (-0.36)	25.96 (+0.09)	**26.55 (+0.50)**†	26.10 (+0.05)
	SacreBLEU	24.10	23.30 (-0.80)	23.80 (-0.30)	24.10(+0.00)	**24.70 (+0.60)**†	24.30 (+0.20)
En⇒Tr	BLEU	16.45	16.35 (-0.10)	16.19 (-0.26)	**17.61 (+1.16)**†	17.22 (+0.77)†	17.23 (+0.78)†
	SacreBLEU	12.20	12.00 (-0.20)	11.80 (-0.40)	**13.20 (+1.00)**†	12.70 (+0.50)†	13.00 (+0.80)†
Tr⇒En	BLEU	17.65	17.89 (+0.24)	17.18 (-0.47)	18.39 (+0.74)†	**18.62 (+0.97)**†	18.41 (+0.76)†
	SacreBLEU	16.60	16.90 (+0.30)	16.10 (-0.50)	17.30 (+0.70)†	**17.60 (+1.00)**†	17.30 (+0.70)†
Zh⇒En	BLEU	48.17	47.02 (-1.15)	47.56 (-0.61)	47.95 (-0.22)	**48.69 (+0.52)**†	47.91 (-0.26)
Average	BLEU	26.69	26.02 (-0.67)	26.27 (-0.42)	26.82 (+0.13)	**27.17 (+0.48)**†	26.85 (+0.16)
	SacreBLEU	22.67	22.07 (-0.60)	22.27 (-0.40)	22.83 (+0.16)	**23.11 (+0.44)**†	22.89 (+0.22)

Table 1: Experimental results on WMT14, WMT17 and NIST12 translation tasks. †means RAN is significantly better than TransF on each test set ($p < 0.01$).

In terms of data preprocessing, for Chinese, we segment all sentences with the word segmentation toolkit THULAC.[3] For the other languages, we run the official script of WMT for tokenization. All sentences of more than 256 words are removed and are encoded using byte-pair encoding.[4] We use a joint vocabulary of 40K tokens for En-De, En-Fr language pairs and 32K tokens for the others, and a separate vocabulary of 32K tokens for Zh-En.

We use standard BASE implementation of Transformer which consists of a 6-layer encoder and a 6-layer decoder. By default, we set d_k=d_v=512 and use 2,048 hidden units in the FFN sub-layers. The residual dropout is 0.1. As for *RAN*s, we set the dropout of attention as 0.2 to avoid over-fitting except on En-Fr. For *HC-SA*, we follow the setting of You et al. (2020) to replace the encoder self-attention with distributions centered around $i-1$ and $i+1$ and the decoder self-attention with distributions centered around $i-1$ and i, and set the standard deviation as 1.0. All models are trained for 150k steps except WMT14 (250k steps) and En-Tr (20k steps). Training is performed using 8 x V100 GPUs for all language pairs except En-Tr and Zh-En which use 2. When decoding, we use a beam width of 4 and a length penalty of 0.6 for the WMT tasks and a length penalty of 1.0 for the Zh-En task. We report the case-sensitive BLEU (Papineni et al., 2002) with *Multi-bleu.perl* [5] and detokenized BLEU score with *SacreBLEU* [6] (Post, 2018) of the best checkpoint in the validation set.

3.2 Main Result

First, we leverage RAN to replace the self-attention of encoder or decoder, respectively. Table 1 shows the overall results on the 10 language pairs. Compared with *TransF*, our RAN models consistently yield competitive or even better results against *TransF* on all datasets. Concretely, 0.13/0.16, 0.48/0.44 and 0.16/0.22 more average BLEU/SacreBLEU are achieved by *RAN-E*, *RAN-D* and *RAN-ALL*, respectively. Although different languages have different linguistic and syntactic structures, RAN can learn reasonable global atten-

[3]https://github.com/thunlp/THULAC-Python
[4]https://github.com/rsennrich/subword-nmt
[5]https://github.com/moses-smt
[6]https://github.com/mjpost/sacrebleu

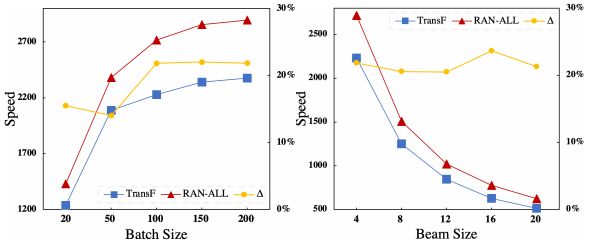

Figure 3: Translation speed (token/sec) varying batch sizes and beam sizes. We set the beam size as 4 when investigating effect of batch sizes, and set the batch size as 100 when explore beam size's influence.

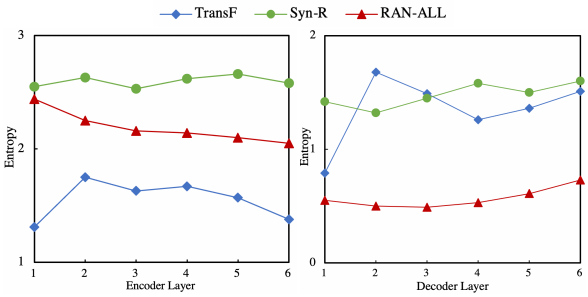

Figure 4: Attention entropy of each encoder or decoder layer.

tion patterns over the whole training corpus.

It is interesting to see that *RAN-D* performs best, which significantly outperforms the *TransF* on most of the language pairs. The biggest performance gain comes from the low resource translation task Tr⇒En where *RAN-D* outperform *TransF* by 0.97/1.0 BLEU/SacreBLEU points. We conjecture that the position-based attention without token-wise interaction is easier to learn and our RAN is able to capture more generalized attention patterns. By contrast, the dot-product self-attention is forced to learn semantic relationship between tokens, and may fall into sub-optimal local minima especially when the training scale is low. This observation is consistent with that in (Raganato et al., 2020). In brief, the improvement indicates that NMT systems can benefit from simplified decoders when training data is insufficient. Besides, although both *RAN-E* and *RAN-D* are effective, we find that their effects can not be accumulated.

Next, we compare RAN with two related methods. To be fair, we only compare *RAN-ALL* to them, where both encoder and decoder self-attention are replaced as done in the two papers. From the table, we can see the two methods significantly decrease the performance over *TransF*, while our model bridges the performance gap between Transformer and the models without the dot-product self-attention, demonstrating the effectiveness of RAN.

3.3 Decoding Speedups

We plot the decoding speed as functions of batch size and beam size in Figure 3. Each experiment is conducted on the same hardware environment and the numbers come from the average of 3 individual runs. To maximize the speedup, we consider *RAN-ALL* setting where both encoder and decoder are accelerated. We can see that *RAN-ALL* speedups the decoding by up to 23.6% with a batch size of 100. In terms of beam size, *RAN-ALL* shows consistent improvement about 1.2x. Note that the previous studies of simplifying attention mechanisms (You et al., 2020; Wu et al., 2019; Michel et al., 2019) also report efficiency improvement of similar magnitudes.

4 Analysis

In order to better understand RAN, we conduct comprehensive empirical studies on its behavior on the WMT14 En⇒De test set.

4.1 Distribution of RAN Weights

In this experiment, we investigate the difference between the learned attention distribution of the different models. To this end, first, we follow Tang et al. (2019) to measure the concentration of attention distribution with attention entropy (Ghader and Monz, 2017):

$$E_A(x_t) = -\sum_{i=1}^{|x|} \mathrm{A}(x_t, x_i) log \mathrm{A}(x_t, x_i), \quad (5)$$

where x_i denotes the i-th token and $\mathrm{A}(x_t, x_i)$ represents the attention distribution at timestep t. Then, we average the attention entropy over all timesteps and then average the attention entropy over all heads in each layer.

Figure 4 displays the entropy of attention distribution. As for encoder, the attention distribution of the *TransF* has the lowest entropy, which gets distributed first and then becomes concentrated again. The attention entropy of *Syn-R* is clearly higher and the attention distribution is uniform. In contrast, the attention distribution of *RAN-ALL* is uniform in the first layer and becomes increasingly concentrated, indicating that the RAN encoder extracts more local information in the higher layers. The

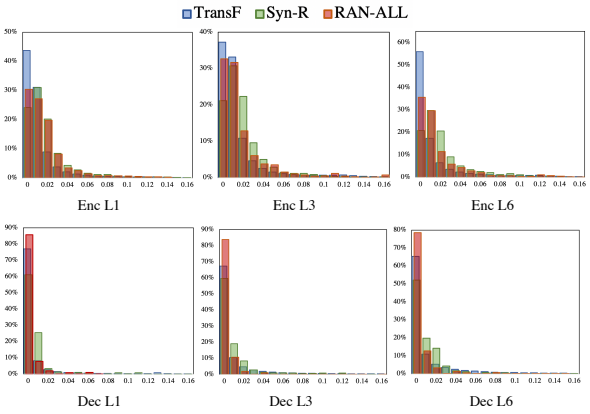

Figure 5: Histogram of encoder and decoder attention weights. *Enc*/*Dec* denotes the encoder or decoder, and L denotes the layer number.

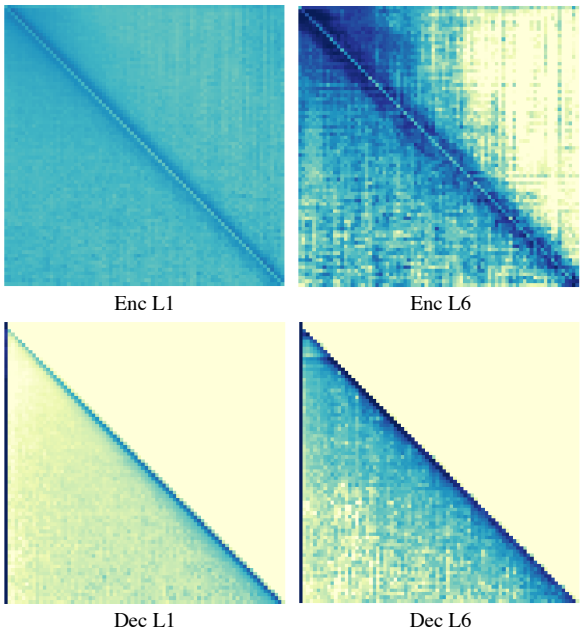

Figure 6: Visualization of RAN Weights.

phenomena of *TransF* and *Syn-R* hold in the decoder, while *RAN-ALL* shows clearly low entropy in all decoder layers.

Moreover, we show each model's attention histograms at layer 1, 3, and 6 in Figure 5. In the encoder, the weights of *Syn-R* and *RAN-ALL* tend to be distributed. In the decoder, the weights of *TransF* and *RAN-ALL* stay near 0 and have smaller variance, while *Syn-R* 's weights are still distributed.

4.2 Visualization of RAN Weights

Since the learned attention weight matrices of RAN are independent of input tokens, we can easily visualize the attention patterns of RAN over positions.[7] In Figure 6, we find that in the encoder, RAN focuses their attention on a local neighborhood around each position. Specifically, in the last layer of the encoder, the weights become more concentrated, potentially due to the hidden representations being contextualized. Interestingly, except attending local windows to the current position, the weights of the decoder are most concentrated in the first token of target sequences. This may demonstrate the mechanism of decoder self-attention that the RAN decoder attends to source-side hidden states based on global source sentence representations aggregated by the start tokens.

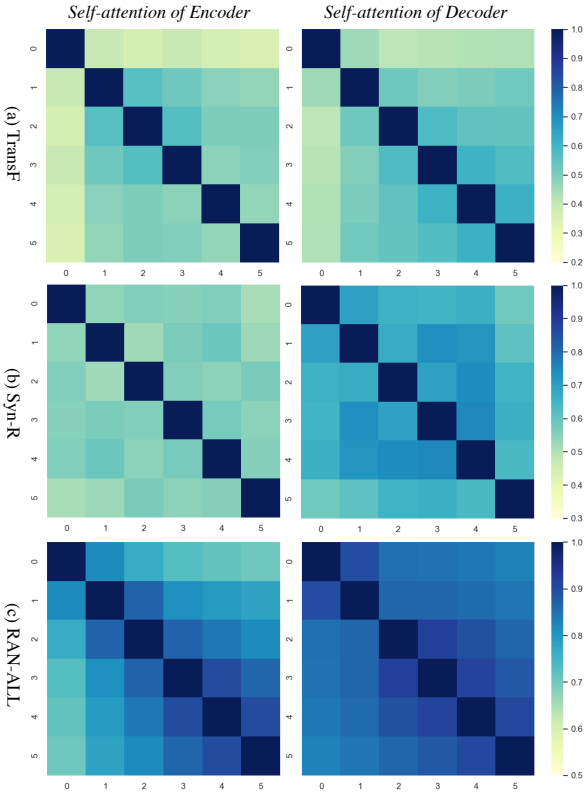

Figure 7: Similarity of attention weights for each pair of layers (WMT14 En⇒De). A dark cell means the distributions are similar.

4.3 Analysis of Attention Weights across Layers

To explore similarity of the attention weights under the different attention mechanisms, we display the Jensen-Shannon divergence (Lin, 1991) of attention between each pair of layers in Figure 7. The conclusions are as follows: First, the attention similarity in *TransF* is not salient but the attention distribution of adjacent layers are similar to some

[7] Each matrix is 80 × 80 since most sentences are not longer than 80 tokens.

Model	En⇒De	
	BLEU	SacreBLEU
TransF	27.70	27.12
−Pos-E	13.14	12.90
−Pos-E+RAN-E	27.48	26.80
−Pos-D	27.66	27.07
−Pos-D+RAN-D	28.02	27.40

Table 2: Experimental results for ablating position embeddings and RAN.

Model	En⇒De		En⇒Fr	
	BLEU	SacreBLEU	BLEU	SacreBLEU
RAN-ALL	27.66	27.00	42.05	39.10
fixed	27.52	26.80	41.93	38.90
w/o LN&RES	27.15	26.50	41.14	38.20

Table 3: Experimental results of ablation study. RES and LN denotes residual connection and layer normalization, respectively.

extent. Second, there are no noticeable patterns found in *Syn-R*. Third, as for *RAN-ALL*, the attention similarity is high especially in the decoder (the JS-divergence ranges from 0.08 to 0.2), and is remarkable between adjacent layers.

4.4 RAN vs. Positional Embedding

The positional embedding is very important to Transformer, and lets the model be aware of word orders. Our RAN learns the input-agnostic global attentions which actually involves the positional information. To verify this point, in this section, we compare several variants of RAN by removing positional embeddings on EN⇒DE translation task, as shown in Table 2. Removing the encoder positional embeddings leads to a catastrophic performance degradation over 14 points. This gap can be recovered by replacing multi-head attention with RAN. *TransF* is merely affected marginally by removing the decoder position embeddings. After applying RAN to decoder, we obtain even better performance than *TransF*. This demonstrates that our RAN indeed captures positional information.

4.5 Ablation Study

To analyze the impact of different components of RAN, we investigate two variants: (1) *RAN-ALL fixed*, where we fixed *Initial Attention Matrices* by random initialization without training; (2) *RAN-ALL w/o LN&RES*, where we removed layer normalization and residual connection in *Recurrent Transition Module*. The results on En⇒De and En⇒Fr translation tasks are listed in Table 3. Surprisingly, we find that the fixed *Initial Attention Matrices* does not lead to significant performance degradation (-0.1 ∼ -0.2 BLEU). This shows we can further reduce the parameters by fixing the *Initial Attention Matrices*. Moreover, removing layer normalization and residual connection leads to a performance drop, which illustrates their effectiveness.

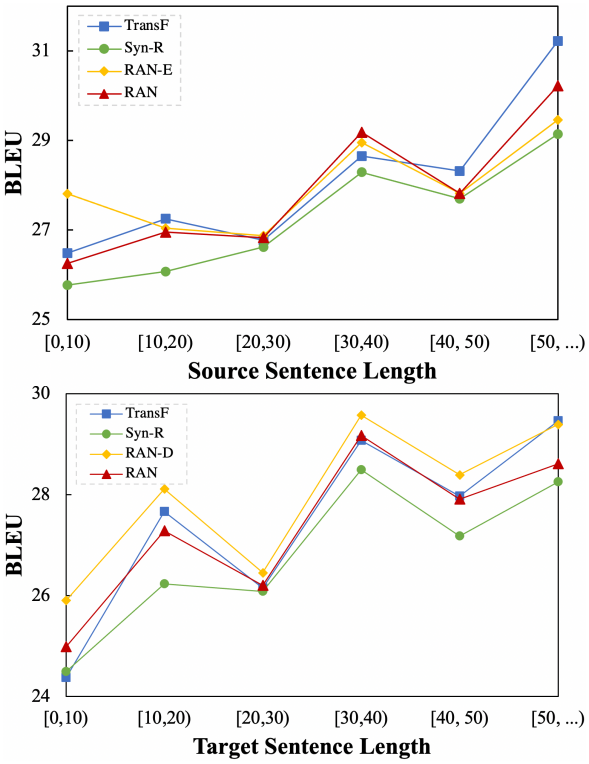

Figure 8: Translation statistics on WMT14 English-German with respect to lengths of source or target sentences.

4.6 Effects on Sentence Length

We divide the WMT14 En⇒De test set into seven bins by source sentence lengths and target sentence lengths, respectively, and plot the performance of each model in BLEU for each bin in Figure 8. We observe that *RAN*s yield better performance on the short and medium-length sentences, while is not good at processing long sentences. Specifically, *RAN-E* performs worse on long source sentences than *TransF*. The improvement of *RAN-D* mainly comes from the performance improvement in translation of the sentences shorter than 50 and promising performance on the long sentences.

Model	Summarization		
	Rouge-1	Rouge-2	Rouge-L
TransF	39.07	17.30	36.00
Syn-R	36.99	15.78	34.07
RAN-E	**39.55**	**17.73**	**36.62**
RAN-D	37.59	16.41	34.73
RAN-ALL	38.25	17.10	35.31
Model	**Dialogue**		
	BLEU-4	Rouge-L	Meteor
TransF	3.12	16.54	8.05
Syn-R	2.87	14.82	7.74
RAN-E	3.47	16.54	**8.33**
RAN-D	3.30	16.32	8.03
RAN-ALL	**3.50**	**16.66**	8.21

Table 4: Experimental results on abstractive summarization (CNN/Dailymail) and dialogue generation (PersonaChat).

5 Application to Other Generation Tasks

To investigate the generalization of RAN, we conduct additional experiments on abstractive summarization using the CNN/Dailymail dataset and dialogue generation using the PersonaChat dataset.[8] On summarization task, all of the models are trained for 300k steps on 2 GPUs with a batch size of 128 sentences. For dialog generation, we segment all dialog with BERT tokenizer[9] and train a SMALL Transformer for 20K steps. We use NLG_Eval[10] for evaluation and report the results in Table 4. *RAN*s achieve competitive results compared to *TransF*, which demonstrates the generalization of RAN on other generation tasks.

6 Related Work

The introduction of attention mechanisms into NMT can be traced back to Bahdanau et al. (2015) and Luong et al. (2015), which are used to learn soft word alignments between language pairs. Due to the significant improvements in translation quality, the attention models have become an critical component of NMT models. More recently, Vaswani et al. (2017) proposed Transformer that achieved the state-of-the-art and soon becomes the most popular NMT architecture. The Self-Attention Network (SAN), playing an important role in the Transformer, has been investigated and analyzed by a number of recent studies (Sanh et al., 2019; Correia et al., 2019; Voita et al., 2019; Michel et al., 2019). These studies have shown that Transformer models are over-parametrized and the self-attention models learn redundant information that can be pruned in various ways.

The observations motivate lots of attempts in improvement of SAN, including 1) improving its computation efficiency and 2) completely replacing it with fixed or learnable global attention patterns. For the former thread, several studies bias attention distributions towards more local areas (Yang et al., 2018; Xu et al., 2019; Cui et al., 2019) or replace SAN with convolutional modules (Yang et al., 2019; Wu et al., 2019), which are more in line with the linguistic expectation. Xiao et al. (2019) share attention weights in adjacent layers and enable efficient re-use of hidden states in a vertical manner.

On the other hand, given that most attention heads learn simple, and often positional patterns, many researchers turn to substitute instance-wise self-attention with global position-based attention patterns. Concretely, Zhang et al. (2018) use average attention models in the decoder of Transformer. You et al. (2020) model the attention distribution as hard-coded Gaussian ones, and Raganato et al. (2020) also replace all but one attention head of each encoder layer with totally position based attentive patterns. More recently, Tay et al. (2020) propose Random Synthesizer in which the attention matrices as trainable parameters that are random initialized and trained with other model parameters.

Overall, our work is related to the second type of approaches and most related to You et al. (2020) and Tay et al. (2020). Unlike You et al. (2020) applying hard-coded Gaussian attention focusing on local windows, the RAN can learn more flexible attention distribution. Tay et al. (2020) allocate different learnable attention matrix for every head in each layer. In addition, so many individual matrices are hard to train and do not reduce the overall parameters at all. In contrast, our RAN uses the recurrent mechanism to refine the learnable attention matrices layer by layer to improve the model capacity, and has the advantages of saving parameters and modeling relationships of attention between adjacent layers.

7 Conclusion

In this paper, we considered a simpler Transformer architecture for NMT without costly dot-product self-attention. For this goal, a novel recurrent atten-

[8] we directly use the dataset from https://github.com/PaddlePaddle/Research/tree/master/NLP/Dialogue-PLATO

[9] https://github.com/google-research/bert

[10] https://github.com/Maluuba/nlg-eval

tion mechanism (RAN) is proposed, which takes the *Initial Attention Matrices* as a whole and update it by a *Recurrent Transition Module* recurrently. Experiments on 10 representative translation tasks show effectiveness of RAN. In the future, we will explore the application of RAN on cross-attention.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014*, pages 103–111.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of EMNLP-IJCNLP 2019*, pages 2174–2184.

Hongyi Cui, Shohei Iida, Po-Hsuan Hung, Takehito Utsuro, and Masaaki Nagata. 2019. Mixed multi-head self-attention for neural machine translation. In *Proceedings of EMNLP-IJCNLP 2019*, pages 206–214.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. *CoRR*, abs/2010.11125.

Markus Freitag and Orhan Firat. 2020. Complete multilingual neural machine translation. In *In Proceedings of WMT@EMNLP 2020*, pages 550–560.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to ? In *Proceedings of IJCNLP 2017*, pages 30–39.

Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2020. Realformer: Transformer likes residual attention. *CoRR*, abs/2012.11747.

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

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proceedings of ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theory*, 37(1):145–151.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*, pages 1412–1421.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Proceedings of NeurIPS 2019*, pages 14014–14024.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of WMT 2018*, pages 186–191.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 556–568. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2019. Understanding neural machine translation by simplification: The case of encoder-free models. In *Proceedings of RANLP 2019*, pages 1186–1193.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2020. Synthesizer: Rethinking self-attention in transformer models. *CoRR*, abs/2005.00743.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS 2017*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of ACL 2019*, pages 5797–5808.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *Proceedings of ICLR 2019*.

Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. 2019. Sharing attention weights for fast transformer. In *Proceedings of IJCAI 2019*, pages 5292–5298.

Mingzhou Xu, Derek F. Wong, Baosong Yang, Yue Zhang, and Lidia S. Chao. 2019. Leveraging local and global patterns for self-attention networks. In *Proceedings of ACL 2019*, pages 3069–3075.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of EMNLP 2018*, pages 4449–4458.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Convolutional self-attention networks. *CoRR*, abs/1904.03107.

Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded gaussian attention for neural machine translation. In *Proceedings of ACL 2020*, pages 7689–7700.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of ACL 2018*, pages 1789–1798.