Neural Machine Translation with Joint Representation

Yanyang Li[1], Qiang Wang[1], Tong Xiao[1,2*], Tongran Liu[3] and Jingbo Zhu[1,2]
[1]Natural Language Processing Lab., Northeastern University, Shenyang, China
[2]NiuTrans Co., Ltd., Shenyang, China
[3]CAS Key Laboratory of Behavioral Science, Institute of Psychology, CAS, Beijing, China
blamedrlee@outlook.com, wangqiangneu@gmail.com, {xiaotong, zhujingbo}@mail.neu.edu.cn, liutr@psych.ac.cn

Abstract

Though early successes of Statistical Machine Translation (SMT) systems are attributed in part to the explicit modelling of the interaction between any two source and target units, e.g., alignment, the recent Neural Machine Translation (NMT) systems resort to the attention which partially encodes the interaction for efficiency. In this paper, we employ *Joint Representation* that fully accounts for each possible interaction. We sidestep the inefficiency issue by refining representations with the proposed efficient attention operation. The resulting *Reformer* models offer a new Sequence-to-Sequence modelling paradigm besides the Encoder-Decoder framework and outperform the Transformer baseline in either the small scale IWSLT14 German-English, English-German and IWSLT15 Vietnamese-English or the large scale NIST12 Chinese-English translation tasks by about 1 BLEU point. We also propose a systematic model scaling approach, allowing the Reformer model to beat the state-of-the-art Transformer in IWSLT14 German-English and NIST12 Chinese-English with about 50% fewer parameters. The code is publicly available at https://github.com/lyy1994/reformer.

Introduction

To translate one sentence in the source language to its equivalent in the target one, the translation model relies on the bilingual interaction between any two source and target units to select the appropriate hypothesis. Early SMT systems are good examples of this as they use the alignment matrix between the source sentence and the translated part to direct the decoding (Koehn 2009).

When it comes to NMT, the natural idea to explicitly model the interaction is by extending the intrinsic representation to have the size $S \times T \times H$, dubbed *Joint Representation*, where S is the source sentence length, T is the target sentence length and H is the hidden size. Despite that this representation can flexibly learn to encode various types of interaction, it is inefficient as it incurs high computation and storage cost. A practical surrogate is the well-known attention mechanism (Vaswani et al. 2017). It mimics the desired interaction by dynamically aggregating a sequence of representations. Though successful, the receptive field of each position in the attention is restricted to one source or target sequence only instead of the cartesian product of them as required by the joint representation.

In this work, we take one step toward the model family *Reformer*, which built entirely on top of the joint representation. We efficiently adapt the most advanced self-attention module to the joint representation space $\mathbb{R}^{S\times T\times H}$, namely *Separable Attention*. With this building block at hand, we present two instantiations of Reformer. The former one, *Reformer-base*, enjoys the best theoretical effectiveness with the capability that it can access any source or target token with minimum $O(1)$ operations but has higher complexity induced by stacking separable attentions. The latter one, *Reformer-fast*, better trades off the effectiveness and efficiency of the separable attention, achieving comparable results as Reformer-base but 50% faster. As both Reformer variants do not resort to either the encoder or the decoder, they shed light on exploring the new promising Sequence-to-Sequence paradigm in the future.

We additionally show that with proper model scaling, our Reformer models are superior to the state-of-the-art (SOTA) Transformer (Vaswani et al. 2017) with fewer parameters on larger datasets. The proposed model scaling method requires only $O(2)$ runs to generate the enhanced model compared to the common Grid Search, whose cost grows polynomially as the candidates of hyper-parameters increase.

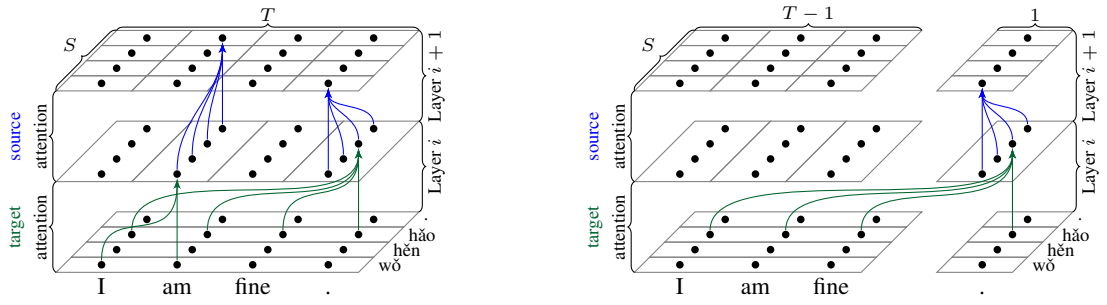
In our experiments, the Reformer models achieve 1.3, 0.8 and 0.7 BLEU point improvement over the Transformer baseline in the small scale IWSLT15 Vietnamese-English and IWSLT14 German-English, English-German datasets as well as 1.9 in the large scale NIST12 Chinese-English dataset. After scaling, it outperforms the SOTA large Transformer counterpart by 0.7 and 2 BLEU point with about 50% parameters in IWSLT14 German-English and NIST12 Chinese-English translations respectively.

Background

Sequence-to-Sequence Learning

Given a sentence pair (x, y), the NMT model learns to maximize its probability $\Pr(y|x)$, which is decomposed into the

*Corresponding Author.



(a) Training (b) Decoding (T-th step)

Figure 1: Separable Attention over representations (Chinese pinyin-English: *"wǒ hěn hǎo ." → "I am fine ."*).

product of the conditional probability of each target token $\Pr(y|x) = \prod_{t=1}^{T} \Pr(y_t|y_{<t}, x)$, where y_t is the target token at position t and $y_{<t}$ is all target tokens before t. Existing Sequence-to-Sequence models for NMT will first use an encoder to map the source sentence x into a sequence of real value vectors h, then a decoder to predict $\Pr(y_t|y_{<t}, x)$ using h and $y_{<t}$.

Transformer

The most successful sequence-to-sequence model is Transformer (Vaswani et al. 2017), which consists of a stack of layers. Each layer first utilizes the self-attention to extract information from the whole sentence, then follows a point-wise feed-forward network to provide non-linearity. These two types of building blocks, self-attention and feed-forward network, are both wrapped by the residual connection (He et al. 2016) to form a sublayer:

$$\text{Sublayer}(x) = \text{Block}(\text{LayerNorm}(x)) + x \tag{1}$$

where x is the input representation, Block is either the self-attention or the feed-forward network and LayerNorm is the layer normalization (Ba, Kiros, and Hinton 2016). The self-attention is formulated as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_\text{m}}})V \tag{2}$$

where d_m is the dimension of the hidden representation and set as the embedding size. For the self-attention inside the encoder, $Q, K, V \in \mathbb{R}^{S \times d_\text{m}}$, while for the self-attention inside the decoder, $Q, K, V \in \mathbb{R}^{T \times d_\text{m}}$. For the attention that bridges the encoder and decoder, $Q \in \mathbb{R}^{T \times d_\text{m}}$ and $K, V \in \mathbb{R}^{S \times d_\text{m}}$. As for the feed-forward network, it consists of two linear projections with ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3}$$

For more details, please refer to Vaswani et al. (2017).

The Reformer Family

Joint Representation

The first problem of designing Reformer is to construct the initial joint representation as the model input. For the encoder or the decoder of a standard NMT model, its input is a sized $S \times E$ or $T \times E$ matrix, where E is the token embedding size. It implies that there are S or T tokens with each represented by a sized E embedding. For the joint representation with the size $S \times T \times H$, it could be interpreted in a similar way: there are $S \times T$ tokens combinations where the first token is from the source sentence and the second one is from the target sentence. Each tokens combination is represented by a sized H embedding.

However, naively assigning a unique embedding to each possible combination is intractable, as it will result in a $V^2 \times H$ embedding matrix, where V is the vocabulary size typically with the scale $10^3 \sim 10^4$. Intuitively, there are only weak connections (e.g., word translation) among tokens from different languages without knowing the context. The independency among tokens can then be safely imposed to factorize their combination. Therefore the combination of each token embedding becomes the embedding of the tokens combination.

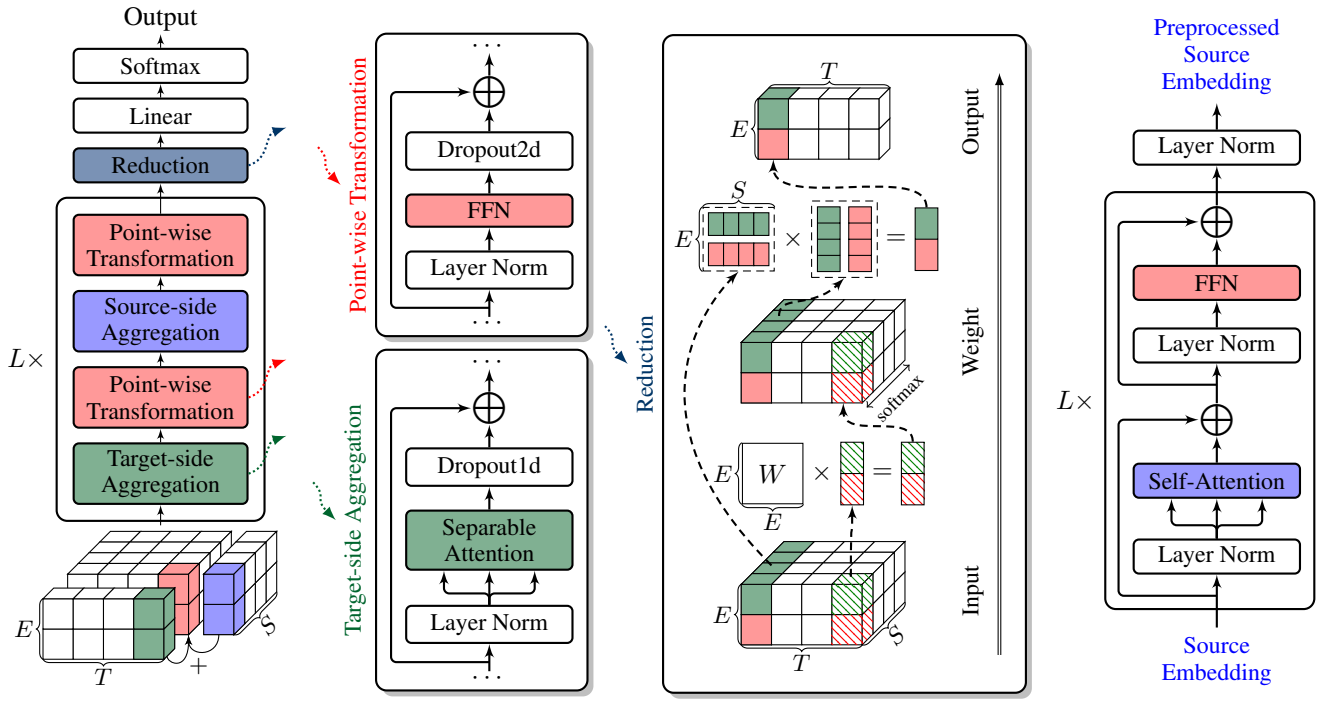
Another issue is to inject the position information to the joint representation for the attention mechanism, as it is unaware of tokens orders. Inspired by the position embedding of Transformer, we similarly use the sinusoidal position embedding to represent the position information of one token and the combination of each token position embedding as the position embedding of tokens combination, because positions are independent of the ones from another axis. Here we choose the simplest addition to combine embedding. Eventually, for the combination of i-th token from the source sentence and j-th token from the target sentence, the embedding of their combination and its position embedding are:

$$\begin{aligned} \text{embed}_{ij} &= \text{embed}_i + \text{embed}_j \\ \text{pos}_{ij} &= \text{pos}_i + \text{pos}_j \end{aligned} \tag{4}$$

where embed is the token/combination embedding, pos is the position embedding. The H of joint representation equals to E according to Eq. (4). The Reformer input is the sum of embed_{ij} and pos_{ij} then multiplied by $\sqrt{E}$.

Separable Attention

The simplest approach to adapt the self-attention to the $\mathbb{R}^{S \times T \times E}$ input space is to collapse the S and T dimensions into a single dimension $J = S \times T$, then perform



(a) Reformer-base (b) Sublayers (c) PreNet

Figure 2: The Reformer Architecture. The green cube in the bottom of Figure. 2(a) is the target token embedding and the blue one is the source token embedding if in Reformer-base otherwise the preprocessed source token embedding from Figure. 2(c).

self-attention to aggregate different combinations information by treating it as a length J sequence. But such an approach suffers from great inefficiency though it can access any combination with $O(1)$ attention, as the operations required for the self-attention grow quadratically with the sequence length, i.e., $J^2 = S^2 \times T^2$ operations.

Inspired by Separable Convolution (Howard et al. 2017) and the fact that combinations are strongly connected along either the S or T dimension, we instead perform the self-attention on these two temporal dimensions separately and sequentially, dubbed *Separable Attention*. Without the lack of theoretical effectiveness, the network output can access the input information in any source or target token through $O(2)$ attentions, as long as we stack one attention operated on S and another one on T alternatively. This way reduces the complexity from $S^2 \times T^2$ to $S^2 \times T + S \times T^2$. Formally, for the separable attention that aggregates information along the dimension $D \in \{S, T\}$, we have:

$$\begin{aligned} \text{SepAttn}(Q, K, V) &= \text{Concat}(\text{split}_1, \cdots, \text{split}_D) \\ \text{where split}_i &= \text{Attention}(Q_i, K_i, V_i) \end{aligned} \tag{5}$$

where Attention is the self-attention from Eq. (2) and $Q, K, V \in \mathbb{R}^{S \times T \times E}$. If $D = S$, then $Q_i, K_i, V_i \in \mathbb{R}^{T \times E}$, otherwise $Q_i, K_i, V_i \in \mathbb{R}^{S \times E}$. In practical implementations, we can perform Eq. (5) efficiently by treating the non-aggregated dimension as an extra batch dimension. Figure. 1(a) shows an example of separable attention during training, where the *target attention* denotes separable attention on T and *source attention* denotes separable attention on S.

Similar to the masked self-attention in Transformer, a future mask is adopted to mask out the illegal softmax inputs in the target attention. This mask, as well as the parallelism among positions, allows the previously generated representations to be reused for computing only the sized $S \times E$ representation which is associated to the latest target tokens at each decoding step. We exemplify it in Figure. 1(b).

Reformer-base

Having the initial joint representation and separable attention, we present *Reformer-base*. As shown in Figure. 2(a), Reformer-base is simply a stack of layers. After computing the input $x \in \mathbb{R}^{S \times T \times E}$ according to Eq. (4), it will pass through four sublayers in each layer: the first one is the target attention that aggregates information along the dimension T, the second one is a point-wise feed-forward network as in Eq. (3), the third one is the source attention that aggregates information along the dimension S and the final one is another feed-forward network. All these sublayers are wrapped by the residual connection similar to Eq. (1). A graphical illustration of these sublayers is in Figure. 2(b).

In the last layer, the network will produce the output with a size of $S \times T \times E$. If a linear projection as well as a softmax is immediately followed, it will result in S distinct predictions for each target position, while we expect only one. Thereby a *Reduction* component is introduced to transform an input $x \in \mathbb{R}^{S \times E}$ to an output $y \in \mathbb{R}^{E}$ before the linear projection and softmax so as to obtain one prediction for one

Layer Type	Complexity	Path Length
Separable Attention	$O(n^3d)$	$O(1)$
Self-Attention	$O(n^2d)$	$O(l)$
Recurrent	$O(nd^2)$	$O(l+n)$
Convolution	$O(knd^2)$	$O(l+n\log_k(n))$

Table 1: The path length required to access any source or target token, n is the sequence length, d is the representation size, k is the kernel size of convolutions and l is the number of layers.

target position. The reduction is defined as:

$$\begin{aligned} \text{Reduction}(x) &= \text{Concat}(\text{head}_1, \cdots, \text{head}_E) \\ \text{where head}_i &= \text{softmax}(W_i x^T) x_i \end{aligned} \tag{6}$$

where W are the learnable parameters and $W_i \in \mathbb{R}^{1\times E}$ and $x_i \in \mathbb{R}^{S\times 1}$. The input and output of the reduction will go through a layer normalization first before any subsequent processing. This simple attention-like mechanism uses all E features of a token to determine the importance of a single feature and normalizes it with other S features in the same position but from different tokens. The rightmost part of Figure. 2(b) demonstrates how reduction runs.

Another important detail of Reformer-base is Dropout. From the initial construction of the joint representation in Eq. (4), we see that the information of a source token is distributed along the dimension T and for the target token along the dimension S. This means that Dropout is no longer able to regularize model by encouraging feature independency within a token representation, as the model can access to the dropped features from representations of other tokens. We introduce Dropout1d and Dropout2d to alleviate this problem (Tompson et al. 2015). In principle, the standard dropout noise is sampled independently for each feature. To prevent the feature dependency along specific dimensions, sharing dropout noise in these dimensions will mask out all potential duplicates that might creep into the model. We apply Dropout2d to the S and T dimensions of the output of the feed-forward network and Dropout1d to the S/T dimension of the output of the target/source attention. We do so because for the feed-forward network it operates on the whole $\mathbb{R}^{S\times T\times E}$ space and we need to prevent feeding dropped features back via S and T dimensions, while for the separable attention it aggregates representations along one temporal dimension thus we only block feature dependency in the other one so as to not affect the aggregation.

Trading-Off the Effectiveness & Efficiency

The Pros & Cons of Reformer-base

As noted in Table 1, Reformer-base can access the information of any source or target token in a path with the minimum $O(1)$ operations. Transformer with the self-attention layer requires $O(l)$ operations as the source side information is only available after passing through l layers in the encoder. The same also happens in other layer types, resulting in the l occurs inside their path lengths. This observation implies that Reformer-base can better capture the dependencies between the source and target tokens, because the path length for propagating the source side information is minimized. Although the separable attention of Reformer-base has higher complexity, it also has a higher degree of parallelism, as it computes all $S \times T$ positions simultaneously.

Despite the theoretical effectiveness of Reformer-base, it suffers from two important issues that hinder its efficiency:

Duplicate Computation When a new target token is fed into the model during decoding, its inner representation is computed only based on the output of the previous layer and starts from the embedding, as depicted in Figure. 1(b). This means that the high-level information about the source sentence and previously generated target tokens is unavailable for the lower layers since it is usually stored in the higher layers representations. In each decoding step, the model will have to recompute this information from scratch, which wastes the model capacity.

Computation Allocation Another less obvious problem of Reformer-base is that it assigns the same amount of computation to both the source and target side, as in each layer the model has one source attention on S and one target attention on T. It is undesirable since the number of source tokens fed into the model is significantly more than the one of target tokens at each decoding step. This observation implies that there should be more computation allocated to process the source side input (Wang et al. 2019). Reformer-base can only stack more separable attention for compensation, where the redundant target attention introduces extra computation cost as noted by the separable attention complexity in Table. 1.

Reformer-fast

Directly feeding the higher layer output from the preceding decoding steps into the lower layer input might resolve these two issues of Reformer-base. But this naive approach is not feasible as its sequential nature breaks the training parallelism by forcing the computation of the representation in one layer and one decoding step to wait for all representation below or before it finished computing, giving the total $O(LT)$ training cost, where L is the total number of layers and T is the target sentence length. Yet for Reformer-base, each of its layers only relies on the previous layer output, thus $O(LT)$ training cost can reduce to $O(L)$ by paralleling the computation over different steps.

The solution we adopted here is that before computing the joint representation in Eq. (4), we introduce a preprocessing network *PreNet* to process the source token embedding embed_i first and the network output is used to replace embed_i in Eq. (4). This network provides not only as much extra computation as required for the source tokens, but also a reusable high-level abstraction of these tokens. It consists of a stack of layers and a layer normalization at the end, where each layer is composed of a self-attention in Eq. (2) and a feed-forward network in Eq. (3) with both wrapped as in Eq. (1). PreNet shares the same hyper-parameters setting as Reformer-base, e.g., the hidden size of the feed-forward network. PreNet is very efficient, as it is computed only once

and reused at each decoding step. Figure. 2(c) shows the PreNet architecture. Reformer-base together with the PreNet forms our new design, *Reformer-fast*.

Reformer-fast has higher efficiency than Reformer-base, cause it avoids stacking many high complexity separable attentions by reusing the high-level abstraction provided in the low complexity PreNet, i.e., from $O(n^3d)$ to $O(n^2d)$. But Reformer-fast scarifies the effectiveness of Reformer-base since it can no longer access any source or target token with $O(1)$ operation but $O(l)$ as the self-attention counterpart.

Discussion Reformer-fast introduces one additional hyper-parameter to be tuned, i.e., the number of layers of PreNet, which denotes how much extra computation is allocated to process the source sentence. Assuming the source sentence length N equals to the target sentence length, there will be N source tokens and t target tokens are fed into the model at the t-th decoding step. This implies that total N^2 source tokens and $\frac{1}{2}N^2 + \frac{1}{2}N$ target tokens are presented in the whole decoding process. Considering only the second-order terms, the ratio of source tokens count to target tokens count $N^2/\frac{1}{2}N^2 = 2$ gives the insight that the computation allocated to the source side should be roughly $2\times$ more than the one of the target side. Therefore, we set the PreNet to have the same height as Reformer-base.

Toward Optimal Model Scaling

Understanding Reformer

To scale Reformer to datasets that are larger than the one that the base setup is optimized for, we need to better understand both the network capacity and parameters. Unfolding the residual connection of the input x_L in the last linear classifier layer, we have:

$$\begin{aligned} x_L &= x_{L-1} + f(x_{L-1}) \\ &= x_0 + f(x_0) + \cdots + f(x_{L-1}) \end{aligned} \tag{7}$$

where x_0 stands for the embedding, x is the layer input and f is the layer. Three factors that determine the network capacity are observed: the number of layers l, the embedding size e and the hidden size of the feed-forward network $h = w \times e$, where larger l and e provide more features for the last layer and higher w strengthens the capacity of each layer thus the output features quality.

l, w, e are also the dominant factors of network parameters. Omitting the bias terms and layer normalization, we have the number of parameters $P(l, w, e) = 2le^2(4 + 2w)$, where we do not consider the embedding matrix here as the involved embedding are negligible on-the-fly. We can observe that P grows quadratically with e, thus enlarging e will overwhelm the contributions of l, w and expose the model to the high risk of overfitting. Therefore we leave e out of consideration for either scaling or calculating parameters. This gives a simplified formulation of parameters:

$$P(l, w) = 2l(4 + 2w) \tag{8}$$

Then the problem of scaling becomes finding l and w that maximize the model performance on the validation set:

$$\max_{l,w} \mathcal{L}(l, w) \tag{9}$$

where $\mathcal{L}$ is any performance measurement metric. In our experiments, we use the cross-entropy loss instead of BLEU as $\mathcal{L}$ because it is more stable.

Discussion Eq. (7) reveals that the superiority of Reformer comes from the fact that it has a larger input space thus richer input features, i.e., from $\mathbb{R}^{l \times e \times S}$ of the encoder or $\mathbb{R}^{l \times e \times (T+S)}$ of the decoder to $\mathbb{R}^{l \times e \times S \times T}$ of Reformer, where S is the source sentence length and T is the target sentence length. This also suggests that Reformer is a special scaled Transformer that scales e dynamically.

Single-Shot Gradient-Based Scaling

Though many research have attempted to find the concrete formulation between the generalization and model complexity (Zhang et al. 2017), there is still no clear conclusion yet. Hence we can only rely on the black-box optimization algorithms to solve Eq. (9), e.g., Grid Search.

However, most black-box optimization algorithms are extremely inefficient in our case despite their superior empirical performances, because they require to evaluate $\mathcal{L}$ with different l, w many times while a single evaluation is a complete training process. An efficient way is to compute the approximate gradient of $\mathcal{L}$ with respect to the current l, w and perform only one step gradient ascent along with a step size α to scale l, w. By the infinitesimal definition of the partial derivative, we have the gradient of $\mathcal{L}$ w.r.t. l:

$$\mathcal{L}'_l = \lim_{\delta \to 0} \frac{\mathcal{L}(l + \delta, w) - \mathcal{L}(l, w)}{\delta} \approx \frac{\mathcal{L}(l + \epsilon, w) - \mathcal{L}(l, w)}{\epsilon} \tag{10}$$

where ϵ is a small number that we manually pick for the approximation. The same method is also adopted for retrieving $\mathcal{L}'_w$. Although the approximated gradients are only useable within a small region around the current l, w since it is only a local direction indicator, it is sufficient when the step size α is small as we need not deviate from the base setup much.

The next problem is to choose a proper step size α. When scaling up a network, we would like to maximize its performance while its risk of overfitting as well as the resource requirement is under controlled. In most cases, the number of parameters P expresses the risk of overfitting, e.g., the use of L_0-Norm regularization term, and the required resources such as space and time also grow with P. Therefore the resource requirement and the risk of overfitting can be jointly represented by the parameters. Choosing a step size α under the parameters constraints is then equivalent to solving:

$$\frac{P(l + \alpha\mathcal{L}'_l, w + \alpha\mathcal{L}'_w)}{P(l, w)} \approx \beta \tag{11}$$

where the manually given constraint β is the ratio we want to scale current parameters, i.e., having $\beta\times$ parameters after scaling. Solving Eq. (11) with P replaced by Eq. (8) will produce a new promising l and w setup, denoted as $\hat{l} = l + \alpha\mathcal{L}'_l$ and $\hat{w} = w + \alpha\mathcal{L}'_w$ respectively.

Discussion Compared to other hyper-parameter tuning algorithms (Golovin et al. 2017), our approach is not directly applicable to hyper-parameters except l, w, because they do not contribute to Eq. (11), therefore finding the optimal step size for them reduces to the Grid Search.

System	Vi-En		De-En		En-De		Zh-En		
	tst2012	tst2013	valid	test	valid	test	MT06	MT05	MT08
baseline	24.70	27.53	34.44	33.63	28.19	27.54	49.63	48.23	43.10
Reformer-base	24.42	27.18	**35.87**	**34.92**	**29.42**	28.32	50.00	48.72	**45.04**
Reformer-fast	**24.98**	**28.26**	**35.87**	34.87	29.31	**28.36**	**50.82**	**49.29**	44.64

Table 2: BLEU results of NMT systems.

System	PPL	BLEU	Params	Speed
baseline	5.39	34.44	16M	$1\times$
Reformer	5.00	35.16	16M	$0.47\times$
+Dropout 1/2d	4.82	35.87	16M	$0.52\times$
+PreNet	4.89	35.87	17M	$0.73\times$

Table 3: Ablation study on De-En validation set.

System	De-En		Zh-En	
	test	Params	MT08	Params
baseline	33.63	16M	43.10	101M
+scaling	34.41	42M	44.60	291M
Reformer-fast	34.87	17M	44.64	105M
+scaling	**35.11**	27M	**46.66**	146M

Table 4: BLEU results of scaled systems.

Experiments

Setup

Dataset We evaluated our approach on IWSLT15 Vietnamese-English (Vi-En), IWSLT14 German-English (De-En), English-German (En-De) and NIST12 Chinese-English (Zh-En) translation tasks. For Vi-En translation, the training set consisted of 130K sentence pairs and we used tst2012 as the validation set and tst2013 as the test set. For De-En and En-De translations, the training set consisted of 160K sentence pairs and we randomly drew 7K samples from the training set as the validation set. We concatenated dev2010, dev2012, tst2010, tst2011 and tst2012 as the test set. For Zh-En translation, We used 1.8M sentence Chinese-English bitext provided within NIST12 OpenMT[1]. We chose the evaluation data of NIST MT06 as the validation set, and MT05, MT08 as the test set. All Chinese sentences were word segmented by a language model-based toolkit.

Model Our baseline systems were based on the open-source implementation of the Transformer model (Vaswani et al. 2017) presented in Ott et al. (2019). The model consisted of the 6-layer encoder and decoder. The size of the embedding, the heads and hidden layer of the feed-forward network were set to 256/512, 4/8 and 1024/2048 for the IWSLT/NIST datasets. Dropout was set to 0.1 for all experiments. For training, we used the Adam optimizer (Kingma and Ba 2015) where the learning rate and batch size were set to 0.0007 and 4096×8 tokens. We applied BPE (Sennrich, Haddow, and Birch 2016) to the De-En, En-De and Zh-En tasks except for the Vi-En one, where the word-based vocabulary achieved better performance.

For both Reformer-base and Reformer-fast, we adopted the same settings as the Transformer baseline, except that Reformer-base consisted of 7 layers and for Reformer-fast it had 5 layers so as to obtain the similar number of parameters as the baseline. In the scaling experiments, we doubled the embedding size and set dropout to 0.3 for Transformer as provided in Ott et al. (2019) for IWSLT14 De-En translation and used Transformer-big setting for NIST12 Zh-En translation. As for Reformer-fast, we used $\beta = 2$ to constraint the added parameters. All experiments were done on 8 Titan V GPUs with the half-precision training.

[1]LDC2000T46, LDC2000T47, LDC2000T50, LDC2003E14, LDC2005T10, LDC2002E18, LDC2007T09, LDC2004T08

Results

As shown in Table 2, both Reformer-base and Reformer-fast significantly outperform the Transformer baseline by a considerable margin in almost all test sets. For Vi-En translation, Reformer-base slightly underperforms the baseline. We attribute this to the symmetric computation allocation for both the source and target side in Reformer-base, as the Vietnamese sentences are typically longer than their English translations thus requesting more source-side operations. For De-En and En-De translations, Reformer-base outperforms Reformer-fast in most cases, where the symmetric computation allocation property better suits these datasets since German sentences are often of similar lengths as their English equivalents. As for Zh-En translation, none of the Reformer models outperforms the other one in all test sets, which indicates that both models behave similarly well.

We perform the ablation study of Reformer models on the IWSLT14 De-En validation set. According to Table 3, we find that dropout 1/2d did better regularize our model with 0.7 BLEU point improvement. With the same amount of parameters, adding PreNet achieves comparable performance but nearly 50% faster than the one without PreNet in decoding, justifying that Reformer-fast did better trade-off the effectiveness and efficiency than Reformer-base.

In the scaling experiments, we obtain $\mathcal{L}'_l = 0.01063$ and $\mathcal{L}'_w = 0.01069$ by picking $\epsilon = 1$, which implies that depth and width are of similar importance to Reformer-fast. Solving Eq. (11) under $\beta = 2$ gives the approximate solution of adding 2 layers and $1/2$ hidden size. Table 4 shows the test set performances after applying the resulting scaling strategy. We observe that our scaling approach indeed generates a stronger model with better generalization ability, which outperforms the SOTA baseline by 0.7/2 BLEU points in the De-En/Zh-En translation with only $\sim$50% parameters.

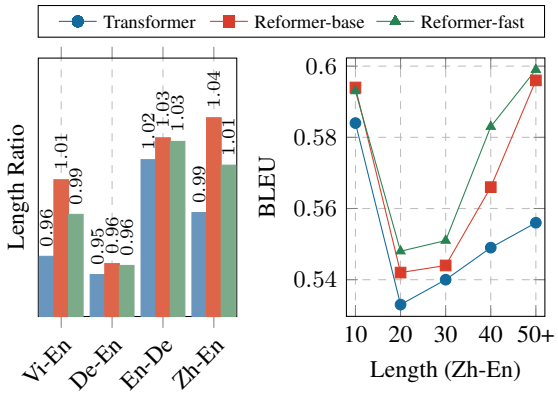


Figure 3: Length statistics.

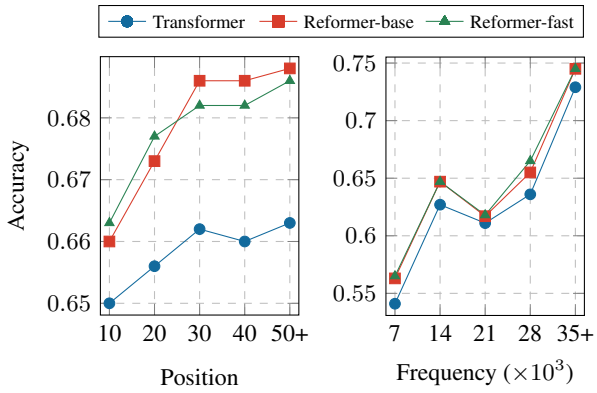


Figure 4: Accuracy statistics (Zh-En).

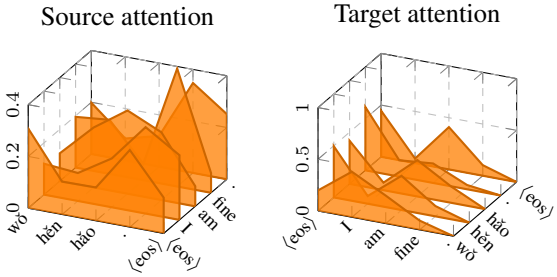


Figure 5: Attention (*"wǒ hěn hǎo ."* → *"I am fine ."*).

Analysis

We investigate the effectiveness of the proposed two Reformer variants. The left of Figure. 3 demonstrates the ratio of the translations lengths to the references lengths in various systems. We observe that Reformer models generate translations with more appropriate lengths than the baseline, i.e., less likely to have under-translation. Interestingly, Reformer-base has higher length ratio than the others, implying it tends to generate longer translations. The right of Figure. 3 shows the BLEU score of translations under source sentences of different lengths. Reformer models outperform the baseline the most in long sentences, which implies that they better capture the long-term dependencies.

In addition to the length perspective, we provide a series of accuracy statistics for the analysis. According to the left of Figure. 4, we find that Reformer models have higher prediction accuracy than the baseline over all target positions within sentences, especially for the distant ones. This observation is on par with the one in the right of Figure. 3 that Reformer outperforms Transformer the most in long sequences. The right of Figure. 4 presents how systems behave on target tokens of different occurrence frequencies. An interesting phenomenon is that Reformer predicts low-frequency tokens more accurate than Transformer, which means that it exploits the context better to infer rare tokens.

Moreover, we illustrate an example of separable attention in Figure. 5. The left part is the source attentions of the source token *"."* to the other ones given different target tokens. These distributions vary when the target token changed, as opposed to a single fixed distribution in the Transformer encoder self-attention. The right part is the target attentions of *"fine"* to previous target tokens given different source tokens. Each source token has its own distribution and it differs from each other, contrary to a single static distribution in the Transformer decoder self-attention. This suggests that Reformer passes more information through more channels.

Related Work

The problem of modelling the interaction in the machine translation systems has long been discussed for years. SMT systems explicitly represent the alignment (Koehn 2009), which directly account for one type of the interaction between any two source and target units. For NMT models, the interaction is only partially modelled via the attention, as it is restricted to one sequence rather than the cartesian product of two sequences. Recent works attempt to alleviate this issue through the joint representation. Kalchbrenner, Danihelka, and Graves (2015) extends LSTM from one dimension to multi-dimensions and Bahar, Brix, and Ney (2018) uses a two dimensions version to iterate through the joint representation. Perhaps the most related work is the pervasive attention model (Elbayad, Besacier, and Verbeek 2018). They first construct a joint representation from the source and target embedding, then stack several convolution layers to produce the final predictions. Our network is also built on top of the joint representation, but each layer can access to any source or target token in a path with the minimum $O(1)$ operation, compared to $O(n)$ for the 2D LSTM and $O(\log_k(n))$ for the pervasive attention, where n is the sequence length and k is the kernel size of the convolution.

Conclusion

We proposed two attention-based networks that use the joint representation to model the interaction. These models achieved significant improvements over four translation tasks, including the small and large scale ones. Despite their successes, we expect more future work on this type of models as they are still in a primitive form.

Acknowledgments

This work was supported in part by the National Science Foundation of China (Nos. 61876035, 61732005 and 61432013), the National Key R&D Program of China (No. 2019QY1801) and the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research. The authors would like to thank anonymous reviewers for their comments.

References

Ba, L. J.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *CoRR* abs/1607.06450.

Bahar, P.; Brix, C.; and Ney, H. 2018. Towards two-dimensional sequence to sequence model in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3009–3015. Brussels, Belgium: Association for Computational Linguistics.

Elbayad, M.; Besacier, L.; and Verbeek, J. 2018. Pervasive attention: 2d convolutional neural networks for sequence-to-sequence prediction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, 97–107.

Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J. E.; and Sculley, D., eds. 2017. *Google Vizier: A Service for Black-Box Optimization*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR* abs/1704.04861.

Kalchbrenner, N.; Danihelka, I.; and Graves, A. 2015. Grid long short-term memory. *CoRR* abs/1507.01526.

Kingma, D. P., and Ba, J. L. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Koehn, P. 2009. *Statistical Machine Translation*. Cambridge, UK: Cambridge University Press.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 48–53. Minneapolis, Minnesota: Association for Computational Linguistics.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. Berlin, Germany: Association for Computational Linguistics.

Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; and Bregler, C. 2015. Efficient object localization using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 648–656.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

Wang, Q.; Li, B.; Xiao, T.; Zhu, J.; Li, C.; Wong, D. F.; and Chao, L. S. 2019. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1810–1822. Florence, Italy: Association for Computational Linguistics.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2017. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.