Learning Light-Weight Translation Models from Deep Transformer

Bei Li[1*], Ziyang Wang[1*], Hui Liu[1*], Quan Du[1,2], Tong Xiao[1,2†], Chunliang Zhang[1,2], Jingbo Zhu[1,2]

[1]NLP Lab, School of Computer Science and Engineering, Northeastern University, Shenyang, China
[2]NiuTrans Research, Shenyang, China
{libei_neu,duquanneu}@outlook.com, {wangziyang,huiliu}@stumail.neu.edu.cn,
{xiaotong,zhangcl,zhujingbo}@mail.neu.edu.cn

Abstract

Recently, deep models have shown tremendous improvements in neural machine translation (NMT). However, systems of this kind are computationally expensive and memory intensive. In this paper, we take a natural step towards learning strong but light-weight NMT systems. We proposed a novel group-permutation based knowledge distillation approach to compressing the deep Transformer model into a shallow model. The experimental results on several benchmarks validate the effectiveness of our method. Our compressed model is 8 times shallower than the deep model, with almost no loss in BLEU. To further enhance the teacher model, we present a Skipping Sub-Layer method to randomly omit sub-layers to introduce perturbation into training, which achieves a BLEU score of 30.63 on English-German newstest2014. The code is publicly available at https://github.com/libeineu/GPKD.

Introduction

Neural machine translation (NMT) has advanced significantly in recent years (Bahdanau, Cho, and Bengio 2015). In particular, the Transformer model has become popular for its well-designed architecture and the ability to capture the dependency among positions over the entire sequence (Vaswani et al. 2017). Early systems of this kind stack 4-8 layers on both the encoder and decoder sides (Wu et al. 2016; Gehring et al. 2017), and the improvement often comes from the use of wider networks (a.k.a., Transformer-Big). More recently, researchers try to explore deeper models for Transformer. Encouraging results appeared in architecture improvements by creating direct pass from the low-level encoder layers to the decoder (Bapna et al. 2018; Wang et al. 2019; Wei et al. 2020; Wu et al. 2019b; Li et al. 2019), and proper initialization strategies (Zhang, Titov, and Sennrich 2019; Xu et al. 2020; Liu et al. 2020; Huang et al. 2020).

Despite promising improvements, problems still remain in deep NMT. Deep Transformer stacked by dozens of encoder layers always have a large number of parameters, which are computationally expensive and memory intensive. For example, a 48-layer Transformer is $3\times$ larger than a 6-layer system and $1.5\times$ slower for inference. It is difficult to deploy such models on resource-restricted devices, such as mobile phones. Therefore, it is crucial to compress such heavy systems into light-weight ones while keeping their performance.

Knowledge distillation is a promising method to address the issue. Although several studies (Sun et al. 2019; Jiao et al. 2020) have attempted to compress the 12-layer BERT model through knowledge distillation, effectively compressing extremely deep Transformer NMT systems is still an open question in the MT community. In addition, these methods leverage sophisticated layer-wise distillation loss functions to minimize the distance between the teacher and the student models, which requires huge memory consumption and enormous training cost.

In this paper, we investigate simple and efficient compression strategies for deep Transformer. We propose a novel Transformer compression approach (named as group-permutation based knowledge distillation method (GPKD)) to transfer the knowledge from an extremely deep teacher model into a shallower student model. We disturb the computation order among each layer group during the teacher training phase, which is easy to implement and memory friendly. Moreover, to further enhance the performance of the teacher network, we introduce a vertical "dropout" (named as skipping sub-layer method) into training by randomly omitting sub-layers to prevent co-adaptations of the over-parameterized teacher network. Although similar technique has been discussed in Fan, Grave, and Joulin (2020)'s work, we believe that the finding here is complementary to theirs. Both GPKD and regularization training methods can be well incorporated into the teacher training process, which is essential for obtaining a strong but light-weight student model.

We ran experiments on the WMT16 English-German, NIST OpenMT12 Chinese-English and WMT19 Chinese-English translation tasks. The GPKD method compressed a 48-layer Transformer into a 6-layer system with almost no loss in BLEU. It outperformed the baseline with the same depth by +2.46 BLEU points. Through skipping sub-layer method, the teacher network achieved a BLEU score of 30.63 BLEU on the *newstest2014* English-German task, and the student obtains additional improvements of 0.50 BLEU points.

Compression of Deep Transformer

In this section, we first introduce the formulation of knowledge distillation (KD), then present the group-permutation

*Equal contribution
†Corresponding author
Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

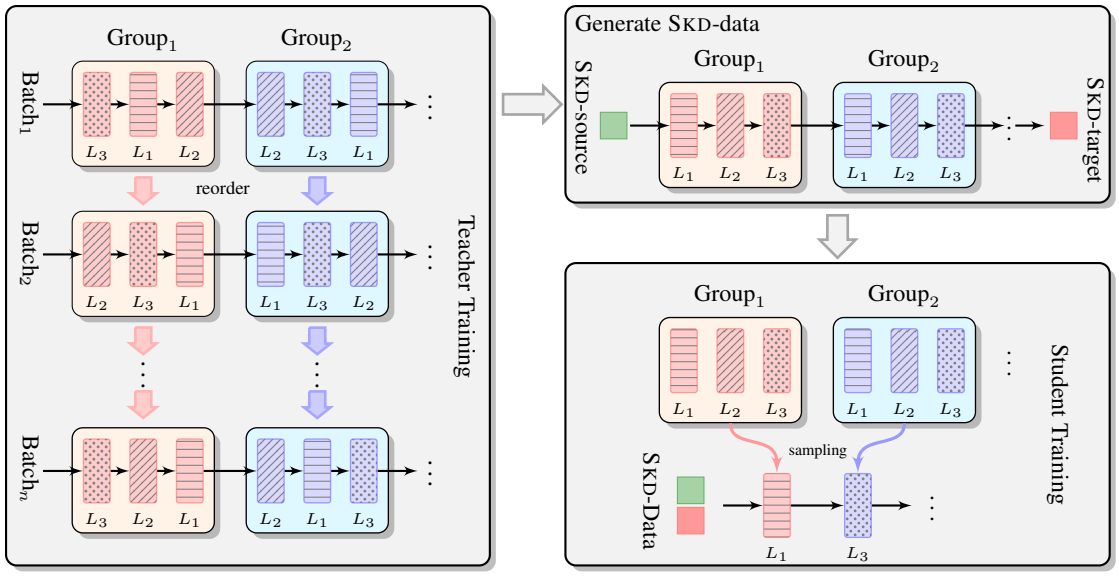


Figure 1: An overview of the GPKD method including three stages. Group_1 and Group_2 correspond to different groups of the stacking layers. L_1, L_2 and L_3 denote the layers in each group.

based knowledge distillation (GPKD) approach to compressing deep Transformer.

Knowledge Distillation

The purpose of KD is to transfer knowledge from a complex teacher network to a light-weight student network by encouraging the student network reproducing the performance of the teacher network. Let $\mathcal{F}^T(x, y)$ and $\mathcal{F}^S(x, y)$ represent the predictions of the teacher network and the student network, respectively. Then KD can be formulated as follows:

$$\mathcal{L}_{\text{KD}} = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} L\left(\mathcal{F}^T(x, y), \mathcal{F}^S(x, y)\right) \quad (1)$$

where $L(\cdot)$ is a loss function to evaluate the distance between $\mathcal{F}^T(x, y)$ and $\mathcal{F}^S(x, y)$. x and y represent the source inputs and target inputs, respectively. $(\mathcal{X},\mathcal{Y})$ denotes the whole training dataset. The objective can be seen as minimizing the loss function to bridge the gap between the student and its teacher.

To advance the student model, a promising method is to learn from the intermediate layer representations of the teacher network via additional loss functions (Sun et al. 2019; Jiao et al. 2020). However, the additional loss functions require large memory footprint due to the logits computation of both the teacher and student networks in each mini-batch training phase. This is quite challenging when the teacher is extremely deep. Alternatively, we choose sequence-level knowledge distillation (SKD), proposed in Kim and Rush (2016)'s work to simplify the training procedure. Through their results, SKD achieves comparable or even higher translation performance than word-level KD method. Concretely, SKD uses the translation results of the teacher network as the gold instead of the ground truth. In this work, we build our student systems upon SKD method.

Group-Permutation Based Knowledge Distillation

Through preliminary experimental results, the student network trained with the SKD data still significantly underperforms its teacher. A possible explanation is directly shrinking the encoder depth is harmful to the translation performance. To further bridge the gap between the teacher network and the student work, we propose a group-permutation based knowledge distillation method, including three stages: (i) group-permutation training strategy which rectifies the information flow of the teacher network during training phase. (ii) generate the SKD data through the teacher network. (iii) train the student network with the SKD data. Instead of randomly initializing the parameters of the student network, we selected layers from the teacher network to form the student network, which provides a better initialization. Figure 1 exhibits the whole training process.

Group-Permutation Training Assuming that the teacher model has N Transformer layers, we aim to extract M layers to form a student model. This can be characterized as learning the mapping between $layer_m$ and $layer_n$, where $m \in \{1, 2..., M\}$ and $n = N/M * m$. To achieve this goal, the stacking layers are first divided into M groups and every adjacent $h = N/M$ layers form a group. The core idea of the proposed method is to make the selected single layer mimic the behavior of its group output. Instead of employing additional loss functions to reduce the distance of the intermediate layer representations between the student network and the teacher network, we simply disturbing the computation order in each group when training the teacher network.

In each mini-batch of training, we randomly disturb the order of the layers in each group. Suppose that $G_i = \{L_1, L_2, L_3\}(i \in M)$ is the set of intermediate layers in each group. The left part of Figure 1 shows the computation order of the layers during the teacher training phase. Note

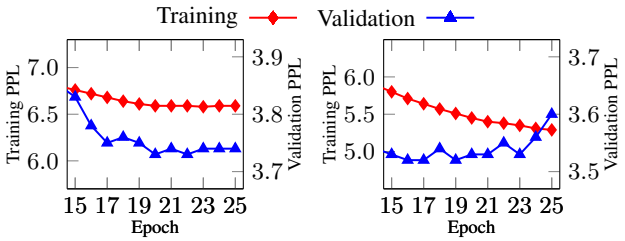


Figure 2: The comparison of training and validation PPL on the shallow (left) and the deep (right) models.

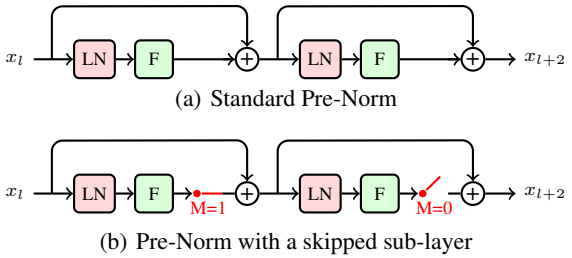


Figure 3: The sub-layer information flow of (a) Standard Pre-Norm, (b) Pre-Norm with a skipped bub-layer.

that each group is independent with others in ordering the layers. In this work, we sample the layer order uniformly from all the permutation choices[1].

Generating SKD Data As shown in Stage 2 (the top right part in Figure 1), given the training dataset $\{\mathcal{X}, \mathcal{Y}\}$, the teacher network translates the source inputs into the target sentences $\mathcal{Z}$. Then the SKD data is the collection of $\{\mathcal{X}, \mathcal{Z}\}$.

Student Training After obtaining the teacher network and the SKD data, we begin optimizing the student network with the supervision of the teacher. As illustrated in Stage 1, each single layer imitates the logits of its group output and every layer in each group behave similar with each other. Then we randomly choose one layer from each group to form a "new" encoder which the encoder depth is reduced from N to M. It can be regarded as the student model with fewer layers. The compression rate is controlled by the hyper-parameter h. One can compress a 48-layer teacher into a 6-layer network by setting $h = 8$, or progressively achieve this goal with $h = 2$ for three times of compression.

The DESDAR Architecture

The GPKD method also fits into the decoder. Hence we design a heterogeneous NMT model consisting of a deep encoder and a shallow decoder, abbreviated as (DESDAR). Such an architecture can enjoy high translation quality due to the deep encoder and fast inference due to the light decoder. Similar findings were observed in previous work (Zhang, Titov, and Sennrich 2019; Xiao et al. 2019), which could speed up the Transformer inference by simplifying the decoder architecture. This is due to the fact that the heavy use of dot-product attention in the decoder and the nature of auto-regressive decoding slows down the system.

Moreover, it offers a way of balancing the translation quality and the inference. This is promising for industrial applications. For example, one can maximize the speedup by using one decoder layer or two, or can yield further BLEU improvements by enlarging the decoder depth. The experimental results in the following sections show the effectiveness of the DESDAR architecture.

[1]For a 3-layer group, the computation order includes $\{L_1, L_2, L_3\}$, $\{L_1, L_3, L_2\}$, $\{L_2, L_1, L_3\}$, $\{L_2, L_3, L_1\}$, $\{L_3, L_1, L_2\}$ and $\{L_3, L_2, L_1\}$

Skipping Sub-Layers for Deep Transformer

It is well known that stronger teacher networks (deeper or wider) can bring better supervision signals (Hinton, Vinyals, and Dean 2015). However, these over-parameterized networks often face the overfitting problem (Fan, Grave, and Joulin 2020). In this section, we introduce a regularization training method to alleviate the overfitting problem which can further improve the extremely deep Transformer systems. Here we start with the sub-layer co-adaptation problem, followed by the *Skipping Sub-Layer* method to address it.

Co-adaptation of Sub-Layers

In Transformer, a stack of sub-layers (or layers) are used between the input and the output. The relationship between the input and the output is complex if the model goes deeper. There will typically be co-adaptation of the sub-layers, and a sub-layer will operate based on the states of other sub-layers. This property is helpful for training because sub-layers are learned to work well together on the training data. But it prevents the model from generalizing well at test time. This is similar to the case in that too many hidden units of a layer lead to the co-adaptation of feature detectors (Hinton et al. 2012). See Figure 2 for perplexities on the training and validation data at different training steps. Clearly, the deep model (48-layer encoder) appears to overfit.

The Skipping Sub-Layer Method

To address the overfitting problem, we drop either the self-attention sub-layer or the feed-forward sub-layer of the Transformer encoder for robust training (call it the *Skipping Sub-Layer* method). Both types of the sub-layer follow the Pre-Norm architecture of deep Transformer (Wang et al. 2019):

$$x_{l+1} = \mathrm{F}(\mathrm{LN}(x_l)) + x_l \tag{2}$$

where $\mathrm{LN}(\cdot)$ is the layer normalization function, x_l is the output of sub-layer l and $\mathrm{F}(\cdot)$ is either the self-attention or feed-forward function. We use variable $M \in \{0, 1\}$ to control how often a sub-layer is omitted. Then, we re-define the sub-layer as:

$$x_{l+1} = M \cdot \mathrm{F}(\mathrm{LN}(x_l)) + x_l \tag{3}$$

where $M = 0$ (or $= 1$) means that the sub-layer is omitted (or reserved). See Figure 3 for comparison of two sub-layer networks.

In addition, Greff, Srivastava, and Schmidhuber (2017) have shown that the lower-level sub-layers of a deep neural network provide the core representation of the input and the subsequent sub-layers refine that representation. Therefore, it is natural to skip fewer sub-layers if they are close to the input, instead of using the same dropping rate for each single layer in Fan, Grave, and Joulin (2020). To this end, we design a method that makes the lower-level sub-layers seldom to be dropped. Let L be the number of layers of the stack[2] and l be the current sub-layer. Then, we define M as

$$M = \begin{cases} 0, & P \leq p_l \\ 1, & P > p_l \end{cases} \tag{4}$$

where

$$p_l = \frac{l}{2L} \cdot \phi, \quad \text{for } 1 \leq l \leq 2L \tag{5}$$

In this model, ϕ is a hyper-parameter that is set to 0.4 in our experiments. p_l is the rate of omitting the sub-layer. For sub-layer l, we first draw a variable P from the uniform distribution in $[0, 1]$. Then, M is set to 1 if $P > p_l$, and 0 otherwise. Thus the lower-level sub-layers are more likely to survive.

Note that the *Skipping Sub-Layer* method is doing something like sampling a sub-network from a full network. For a model with $2L$ sub-layers, it encodes 2^{2L} sub-networks and each configuration of sub-layer omission represents a sub-network. These sub-models are learned efficiently because they share the parameters. For inference, all these sub-models behave like an ensemble model. Following the work in Hinton et al. (2012), we rescale the output representation of each sub-layer by the survival rate $1 - p_l$, like this:

$$x_{l+1} = (1 - p_l) \cdot \mathrm{F}(\mathrm{LN}(x_l)) + x_l \tag{6}$$

Factor $1 - p_l$ is used to scale-down the output of the sub-layer, so the expected output of the sub-layer is the same as the actual output at test time. Then, the final model can make a more accurate prediction by averaging the predictions from 2^{2L} sub-models.

Two-stage Training

Our *Skipping Sub-Layer* method is straightforwardly applicable to the training phase of Transformer. However, we found in our preliminary experiments that the learned model even underperformed the baseline if we introduced sub-layer omission into training from the beginning. This might be due to the fact that deep Transformer is complex and the training is fragile to the perturbation if the model does not get to the smoothed region of the error surface.

Here, we instead adopt a two-stage training method to learn the deep Transformer model with omitting sub-layers. First, we train the model as usual but early stop it when the model converges on the validation set. Then, we apply our *Skipping Sub-Layer* method to the model and continue training until the model converges again. As is shown in Table 4, the two-stage training is helpful for making better use of random sub-layer omission and producing better results. To our knowledge, we are the first to emphasize the importance of the two-stage training in building deep Transformer with omitting layers or sub-layers.

[2]There are $2L$ sub-layers for L layers.

Experiments

We conducted experiments on the WMT'16 English-German, NIST'12 Chinese-English and WMT19' Chinese-English tasks.

Experimental Setups

The bilingual and evaluation data mainly came from three sources:

- WMT'16 English-German (En-De). We used the same datasets as in (Vaswani et al. 2017; Wu et al. 2019a; Wang et al. 2019). They consisted of approximately 4.5M tokenized sentence pairs. All sentences were segmented into sequences of sub-word units (Sennrich, Haddow, and Birch 2016) with 32K merge operations using a vocabulary shared by the source and target sides. *newstest2016* and *newstest2014* was the validation and test data, respectively.
- NIST'12 Chinese-English (NIST Zh-En). We randomly extracted nearly 1.9M bilingual corpus from NIST'12 OpenMT[3]. MT06 was the validation set and the concatenation of MT04 and MT08 was the test set.
- WMT'19 Chinese-English (WMT Zh-En). For more convincing results, we also experimented on a larger dataset extracted from the mixture of the CWMT and UN corpora, provided by Wang et al. (2019). We selected *newstest2017* as the validation data and reported the BLEU scores on *newstest2018* and *newstest2019*.

We adopted the compound split strategy for En-De, which was a common post-processing step used in previous work (Vaswani et al. 2017; Wang et al. 2019; Wu et al. 2019b). For Zh-En tasks, all the sentences were segmented by the tool provided within NiuTrans (Xiao et al. 2012). Translation quality was measured by case-sensitive tokenized BLEU for En-De task, and case-insensitive tokenized BLEU for NIST Zh-En task. The BLEU script was *multi-bleu.perl*. We also reported the sacrebleu[4] results on the En-De and the WMT Zh-En tasks, respectively.

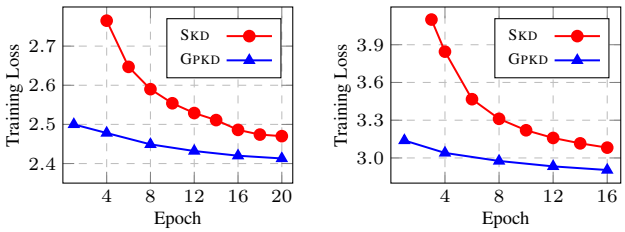
For training, we used Adam optimizer (Kingma and Ba 2015), and followed the hyper-parameters uesd in Wang et al. (2019). As suggested in Shaw, Uszkoreit, and Vaswani (2018), we incorporated the relative position representation into the self-attention mechanism to enhance the positional information. This is quite crucial when building extremely deep Transformer. Then, we batched sentence pairs by approximate length, and limited input/output tokens per batch to $4,096$/GPU and updated the parameters every two steps. The hidden size of Base and Deep models was 512, and 1024 for

[3]LDC Number: LDC2000T46, LDC2000T47, LDC2000T50, LDC2003E14, LDC2005T10, LDC2002E18, LDC2007T09, and LDC2004T08

[4]BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a +version.1.2.12

Systems	Depth	WMT En-De			NIST Zh-En					WMT Zh-En					
		Params	BLEU	Δ	Params	MT06	MT04	MT08	Avg	Δ	Params	Test17	Test18	Test19	Avg	Δ
Deep-RPR-24L	24-6	118M	29.39	-	141M	53.56	56.31	48.15	52.67	-	142M	26.50	27.00	28.30	27.26	-
Base-RPR-6L	6-6	62M	27.60	ref	84M	51.63	55.06	46.17	50.95	ref	85M	25.90	26.00	27.60	26.50	ref
+SKD	6-6	62M	28.50	+0.90	84M	52.60	55.16	46.99	51.58	+0.63	85M	26.10	26.20	27.80	26.70	+0.20
+GPKD	6-6	62M	29.33	+1.73	84M	53.32	56.24	47.65	52.40	+1.45	85M	26.40	26.90	28.30	27.20	+0.70
Deep-RPR-48L	48-6	194M	30.03	-	217M	54.00	56.40	48.21	52.87	-	218M	26.80	27.30	28.60	27.56	-
Base-RPR-6L	6-6	62M	27.60	ref	84M	51.63	55.06	46.17	50.95	ref	85M	25.90	26.00	27.60	26.50	ref
+SKD	6-6	62M	29.01	+1.39	84M	52.55	55.15	46.92	51.54	+0.59	85M	26.40	26.40	28.00	26.93	+0.43
+GPKD	6-6	62M	29.68	+2.08	84M	53.63	56.19	48.11	52.64	+1.69	85M	26.70	27.10	28.50	27.43	+0.93
Big-RPR-12L	12-6	286M	29.91	-	332M	54.19	56.89	49.29	53.45	-	335M	27.00	27.50	28.70	27.73	-
Big-RPR-6L	6-6	211M	29.21	ref	256M	52.80	55.57	47.54	51.97	ref	258M	26.50	27.00	28.20	27.23	ref
+SKD	6-6	211M	29.47	+0.26	256M	53.73	55.80	47.78	52.43	+0.46	258M	26.80	27.30	28.50	27.53	+0.30
+GPKD	6-6	211M	29.88	+0.67	256M	54.03	56.55	49.16	53.24	+1.27	258M	26.90	27.40	28.70	27.66	+0.43

Table 1: The results of the GPKD method applied in encoder side on En-De and Zh-En tasks. We set $h = 4$ and $h = 8$ to compress the 24-layer and 48-layer systems, respectively. RPR denotes the Transformer incorporating byrelative positional information.



(a) WMT En-De (b) NIST Zh-En

Figure 4: The training loss of applying the GPKD (blue) and SKD (red) methods on the WMT En-De, NIST Zh-En tasks, respectively.

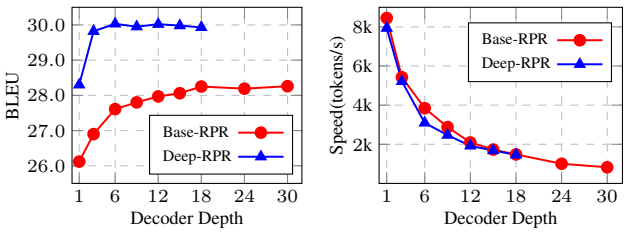


Figure 5: BLEU scores [%] and translation speed [tokens/sec] against decoder depth on the En-De task.

big counterparts. The Base/Big/Deep models were updated for 50k/150k/50k steps on the En-De task, 25k/50k/25k steps on the NIST Zh-En task and 100k/200k/100k on the WMT Zh-En task. The beam size and length penalty were set to 4/0.6 and 6/1.3 for En-De and Zh-En tasks, respectively.

Experimental Results

The Effect of GPKD Method We successfully trained 24-layer/48-layer Transformer-Deep systems and 12-layer Transformer-Big systems incorporating the relative position representation (RPR) on three tasks. Table 1 shows the results

System	BLEU	ΔBLEU		Speedup	
		Base	Deep	Base	Deep
Zhang, Xiong, and Su (2018)	27.33	-0.11	N/A	1.34×	N/A
Xiao et al. (2019)	27.69	+0.17	N/A	1.39×	N/A
Base	27.60	N/A	N/A	N/A	N/A
Deep	30.03	N/A	N/A	N/A	N/A
DESDAR 48-3	29.82	+2.22	-0.21	1.28×	1.52×
DESDAR 48-1	28.31	+0.71	-1.72	1.97×	2.17×
DESDAR 48-1 (SKD)	29.70	+2.10	-0.27	1.99×	2.18×
DESDAR 48-1 (GPKD)	30.01	+2.41	-0.02	1.99×	2.18×

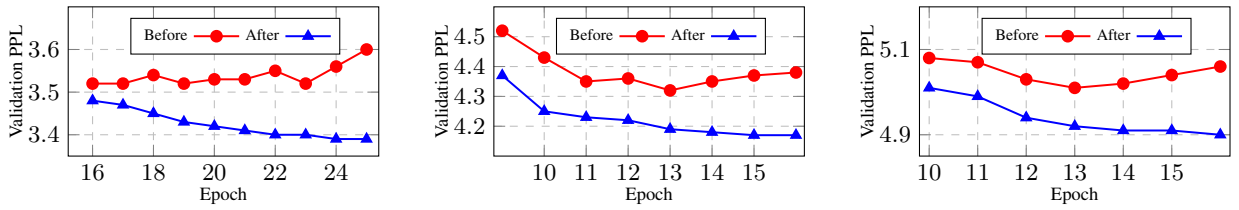
Table 2: BLEU scores [%], inference speedup [×] on the WMT En-De task.

when applying the GPKD method to the encoder side. Deep Transformer systems outperform the shallow baselines by a large margin, but the model capacities are 2 or 3 times larger. And 6-layer models trained through SKD outperform the shallow baselines by 0.63-1.39 BLEU scores, but there is still a nonnegligible gap between them and their deep teachers. As we expect, our GPKD method can enable the baselines to perform similarly with the deep teacher systems, and outperforms SKD by 0.41-1.10 BLEU scores on three benchmarks. Note that, although the compressed systems are 4 or 8 × shallower, they only underperform the deep baselines by a small margin. Similar phenomenon is observed when switching to a wide network, that 6-layer RPR-Big systems match with its teacher with almost no loss in BLEU, indicating the GPKD method is applicable in different model capacities. Moreover, Figure 4 plots the training loss of SKD and GPKD methods. We observe that GPKD obtains a much lower training loss than SKD on the WMT En-De and NIST Zh-En tasks, which further verifies the effectiveness of GPKD.

Figure 5 plots the BLEU scores and the translation speeds of different decoder depths. We can see that deeper decoders yield modest BLEU improvements but slow down the inference significantly based on a shallow encoder. While no

System	WMT En-De					NIST Zh-En				WMT Zh-En		
	Params	Updates	Time	BLEU	SBLEU	MT06	MT04	MT05	MT08	Test17	Test18	Test19
Fan, Grave, and Joulin (2020)	286M	16×13K	N/A	30.20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Li et al. (2020)	194M	50K	11.75h	30.21	29.0	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Wu et al. (2019b)	270M	800K	N/A	29.92	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Ott et al. (2018)	210M	16×13K	N/A	29.30	28.3	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Wang et al. (2019)	137M	50K	N/A	29.30	N/A	53.57	55.91	52.30	48.12	26.9	27.4	N/A
Base-RPR	62M	100K	5.02h	27.60	26.5	51.63	55.06	50.38	46.17	25.9	26.0	27.6
Deep-RPR-48L	194M	50K	16.87h	30.03	28.8	54.00	56.40	52.98	48.21	26.8	27.3	28.6
+ *Skipping Sub-Layer*	194M	50K	15.41h	**30.63**$^{\diamond}$	**29.4**	**54.76**	**57.01**	**53.52**	**49.16**	**27.2**	**27.7**	**29.0**

Table 3: BLEU scores [%], parameters and training time on three language pairs. Note that SBLEU represents the sacrebleu score and the p_l (omission rate) was set to 0.4. Note that $\diamond$ presents the result of beam 8. The BLEU of beam 4 is 30.49.



(a) WMT En-De (b) NIST Zh-En (c) WMT Zh-En

Figure 6: The validation PPL of employing the Skipping Sub-Layer method before (red) and after (blue) on En-De, NIST Zh-En and WMT Zh-En tasks, respectively.

improvement is observed when we switch to a strong Deep-RPR-48L model.

Table 2 exhibits several DESDAR systems with different settings. DESDAR 48-3 achieves comparable performance with the 48-6 baseline, but speeds up the inference by 1.52×. However, a shallower decoder makes a great decrease compact on BLEU, though it obtains a 1.97× speedup. Through the SKD method, the DESDAR 48-1 system can even outperform the RPR-Base by 2.10 BLEU scores and speeds up the inference by 2.18×. Moreover, our GPKD method can enable the DESDAR 48-1 system to perform similarly with the deep baseline, outperforms SKD by nearly +0.31 BLEU scores. Interestingly, after knowledge distillation, the beam search seems like to be not important for the DESDAR systems, which can achieve a 3.2× speedup with no performance sacrifice with the greedy search. This may be due to the fact that the student network learns the soft distribution generated by the teacher network, which has already limited the search space to the max beam margin (Kim and Rush 2016).

The Effect of Skipping Sub-Layer Method The red curves in Figure 6 show that the 48-layer RPR model converges quickly on three tasks, and the validation PPL goes up later. At the same time, the training PPL is still going down (see Figure 2). As we expect, the *Skipping Sub-Layer* method reduces the overfitting problem and thus achieves a lower PPL (3.39) on the validation set. The similar phenomena are observed on the other two tasks. In addition, the last row of Table 3 shows that the strong Deep-RPR model trained through the *Skipping Sub-Layer* approach obtains +0.40-0.72 BLEU improvements on three benchmarks.

Comparison with Related Methods Table 4 exhibits the BLEU scores and the validation PPL of several related systems trained through two optimization ways. Interestingly, all these systems underperform the deep baseline when we trained them from scratch. This is reasonable because the skipped connections make disturbances to the optimization when the model is still in the early stage of training, and the parameters are in the non-smoothed region of the loss function. The phenomena here verify the importance of the two-stage training strategy.

On the other hand, our *Skipping Sub-Layer* method and *Stochastic Layers* (Pham et al. 2019) trained by the finetuning schema both beat the strong baseline. This confirms that dropping sub-layers randomly is helpful for reducing the overfitting when we train deep Transformer models. However, the deep models trained with *LayerDrop* method cannot gain more benefit and we attribute this to the fact that *LayerDrop* uses the same probability to omit sub-layers throughout the stack. This is harmful to the performance because omitting many lower-level layers reduces the representation ability of the deep model significantly (Huang et al. 2016; Greff, Srivastava, and Schmidhuber 2017).

Ablation Study Table 5 shows the ablation study of omitting different components, including randomly skipping the feed-forward (FFN) sub-layer, the self-attention (SAN) sub-layer, all sub-layers and the whole layer. As shown in Table 5, the performance of the single checkpoint and the checkpoint averaging model is reported. First, we can see that all these systems obtain lower validation PPLs and higher BLEU scores for the single model than the baseline. And our default

	System	PPL	BLEU
	Deep-RPR-48L	3.52	30.03
Scratch	*Skipping Sub-Layer*	3.41	29.82
Scratch	*LayerDrop* (Fan, Grave, and Joulin 2020)	3.42	29.65
Scratch	*Stochastic Layers* (Pham et al. 2019)	3.44	29.75
Finetune	*Skipping Sub-Layer*	**3.39**	**30.49**
Finetune	*LayerDrop* (Fan, Grave, and Joulin 2020)	3.47	29.77
Finetune	*Stochastic Layers* (Pham et al. 2019)	3.44	30.12

Table 4: Comparison of training from scratch and finetune on the WMT En-De task.

System	PPL	Single	Avg
Deep-RPR-48L	3.52	29.19	30.03
Skip FFN	3.45	29.61	30.22
Skip SAN	3.45	29.71	30.26
Skip FFN or SAN	**3.39**	**29.95**	**30.49**
Skip Layer	3.46	29.45	29.92

Table 5: Ablation results on the WMT En-De task.

strategy beats the baseline by a larger margin in terms of both PPL and BLEU. There is no significant difference when we skip FFN or SAN only, and they all surpass the system that randomly omits the entire layer. This is mainly due to the fact that we can sample more diverse sub-networks in training. Note that the results here were mainly experimented on deep Transformer, rather than a shallow but wide counterpart reported in (Fan, Grave, and Joulin 2020), which is complementary to the community.

The Overall Results Table 6 shows the results of incorporating both the GPKD and *Skipping Sub-Layer* approaches. Note that, these systems are obtained upon the strong Deep-RPR-48L system. As we can see that a 6-6 system achieves comparable performance with the state-of-the-art, though the parameter is only 4 times less than theirs. In addition, it beats the shallow baseline by +2.56 BLEU scores at the same scale. This offers a way of selecting the proper system considering the trade-off between the translation performance and the model storage. For example, one can choose GPKD 6-3 system with satisfactory performance and fast inference speed, or GPKD 24-3 system with both high translation quality and competitive inference speed. Another interesting finding here is that shrinking the decoder depth may hurt the BLEU score when the encoder is not strong enough.

Related Work

Deep neural networks play an important role in the resurgence of deep learning. It has been observed that increasing the depth of neural networks can drastically improve the performance of convolutional neural network-based systems (He et al. 2016). The machine translation communities follow this trend. For example, Bapna et al. (2018) and Wang et al. (2019) shortened the path from upper-level layers to lower-level layers so as to avoid gradient vanishing/exploding. Wu et al. (2019b) designed a two-stage approach with three specially designed components to build a 8-layer Transformer-Big system. Zhang, Titov, and Sennrich (2019) successfully trained a deep Post-Norm Transformer with carefully designed layer-wise initialization strategy. More attempts on initialization strategy emereged recently (Xu et al. 2020; Liu et al. 2020; Huang et al. 2020). Perhaps the most relevant work with us is Fan, Grave, and Joulin (2020)'s work. They employed *LayerDrop* mechanism to train a 12-6 Transformer-Big and pruned sub-networks during inference without finetuning. Here we address a similar issue in deep Transformer, which has not been discussed yet. Beyond this, we present a new training strategy that can boost the deep system in a robust manner.

System	Params	Depth	BLEU	Speed
Fan, Grave, and Joulin (2020)	286M	12-6	30.20	2534
Wu et al. (2019b)	270M	8-8	29.92	2044
Wei et al. (2020)	512M	18-6	30.56	N/A
Skipping Sub-Layer + GP	194M	48-6	30.59	3092
GPKD 24-3	106M	24-3	30.40	5237
GPKD 24-1	97M	24-1	30.05	8116
GPKD 6-6	62M	6-6	30.16	3817
GPKD 6-3	48M	6-3	29.71	5460

Table 6: The overall results of BLEU scores [%] and translation speed [tokens/sec] on the WMT En-De task.

For model compression, there are many successful methods, such as quantization (Gong et al. 2014), knowledge distillation (KD) (Kim and Rush 2016), weight pruning (Han et al. 2015) and efficient Transformer architecture (Mehta et al. 2020a,b). For Transformer models, Sun et al. (2019) proposed a novel approach to compressing a large BERT model into a shallow one via the Patient Knowledge Distillation method. Jiao et al. (2020) achieved a better compression rate by richer supervision signals between the teacher network and the student network. However, these methods are not straightforwardly applicable to machine translation, they need simultaneously compute the logits of each layer in both the teacher and student networks, which consumes large GPU memory. In this work, we propose the GPKD method to compress an extremely deep model into a baseline-like system, without any additional computation cost.

Conclusions

Our contributions in this work are two folds. (i) We propose a GPKD method to compress the deep model into a shallower one with minor performance sacrifice, which outperforms the SKD method by a large margin. (ii) The proposed *Skipping Sub-Layer* method reduces the overfitting problem when training extremely deep encoder systems by randomly omitting sub-layers during training phase. The experimental results on three widely-used benchmarks validate the effectiveness of the proposed methods. After the incorporating of two methods, the strong but light-weight student models show competitive performance which is application friendly.

Acknowledgments

This work was supported in part by the National Science Foundation of China (Nos. 61876035 and 61732005), the National Key R&D Program of China (No. 2019QY1801). The authors would like to thank anonymous reviewers for their valuable comments. And thank Qiang Wang and Yufan Jiang for their helpful advice to improve the paper. Appreciate Tao Zhou for his suggestion to beautify the figures.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *In Proceedings of the 3rd International Conference on Learning Representations*.

Bapna, A.; Chen, M.; Firat, O.; Cao, Y.; and Wu, Y. 2018. Training Deeper Neural Machine Translation Models with Transparent Attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3028–3033. Brussels, Belgium: Association for Computational Linguistics.

Fan, A.; Grave, E.; and Joulin, A. 2020. Reducing Transformer Depth on Demand with Structured Dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. 2017. Convolutional Sequence to Sequence Learning. In *ICML*.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. D. 2014. Compressing Deep Convolutional Networks using Vector Quantization. *CoRR* abs/1412.6115.

Greff, K.; Srivastava, R. K.; and Schmidhuber, J. 2017. Highway and Residual Networks learn Unrolled Iterative Estimation. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. J. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 1135–1143.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep Networks with Stochastic Depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, 646–661.

Huang, X. S.; Perez, F.; Ba, J.; and Volkovs, M. 2020. Improving Transformer Optimization Through Better Initialization. In *Proceedings of Machine Learning and Systems 2020*, 9868–9876.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4163–4174. Online: Association for Computational Linguistics.

Kim, Y.; and Rush, A. M. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1317–1327. Austin, Texas: Association for Computational Linguistics.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Li, B.; Li, Y.; Xu, C.; Lin, Y.; Liu, J.; Liu, H.; Wang, Z.; Zhang, Y.; Xu, N.; Wang, Z.; Feng, K.; Chen, H.; Liu, T.; Li, Y.; Wang, Q.; Xiao, T.; and Zhu, J. 2019. The NiuTrans Machine Translation Systems for WMT19. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, 257–266. Florence, Italy: Association for Computational Linguistics.

Li, B.; Wang, Z.; Liu, H.; Jiang, Y.; Du, Q.; Xiao, T.; Wang, H.; and Zhu, J. 2020. Shallow-to-Deep Training for Neural Machine Translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 995–1005. Online: Association for Computational Linguistics.

Liu, L.; Liu, X.; Gao, J.; Chen, W.; and Han, J. 2020. Understanding the Difficulty of Training Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 5747–5763. Online: Association for Computational Linguistics.

Mehta, S.; Ghazvininejad, M.; Iyer, S.; Zettlemoyer, L.; and Hajishirzi, H. 2020a. DeLighT: Very Deep and Light-weight Transformer. *arXiv preprint arXiv:2008.00623* .

Mehta, S.; Koncel-Kedziorski, R.; Rastegari, M.; and Hajishirzi, H. 2020b. DeFINE: Deep Factorized Input Token Embeddings for Neural Sequence Modeling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ott, M.; Edunov, S.; Grangier, D.; and Auli, M. 2018. Scaling Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation, Volume 1: Research Papers*, 1–9. Belgium, Brussels: Association for Computational Linguistics.

Pham, N.; Nguyen, T.; Niehues, J.; Müller, M.; and Waibel, A. 2019. Very Deep Self-Attention Networks for End-to-End

Speech Recognition. In Kubin, G.; and Kacic, Z., eds., *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, 66–70. ISCA.

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.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 464–468. New Orleans, Louisiana: Association for Computational Linguistics.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 2019. Patient Knowledge Distillation for BERT Model Compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4323–4332. Hong Kong, China: Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 6000–6010.

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.

Wei, X.; Yu, H.; Hu, Y.; Zhang, Y.; Weng, R.; and Luo, W. 2020. Multiscale Collaborative Deep Models for Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 414–426. Online: Association for Computational Linguistics.

Wu, F.; Fan, A.; Baevski, A.; Dauphin, Y. N.; and Auli, M. 2019a. Pay Less Attention with Lightweight and Dynamic Convolutions. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Wu, L.; Wang, Y.; Xia, Y.; Tian, F.; Gao, F.; Qin, T.; Lai, J.; and Liu, T.-Y. 2019b. Depth Growing for Neural Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5558–5563. Florence, Italy.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* .

Xiao, T.; Li, Y.; Zhu, J.; Yu, Z.; and Liu, T. 2019. Sharing Attention Weights for Fast Transformer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 5292–5298.

Xiao, T.; Zhu, J.; Zhang, H.; and Li, Q. 2012. NiuTrans: An Open Source Toolkit for Phrase-based and Syntax-based Machine Translation. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, 19–24. Stroudsburg, PA, USA: Association for Computational Linguistics.

Xu, H.; Liu, Q.; van Genabith, J.; Xiong, D.; and Zhang, J. 2020. Lipschitz Constrained Parameter Initialization for Deep Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 397–402. Online: Association for Computational Linguistics.

Zhang, B.; Titov, I.; and Sennrich, R. 2019. Improving Deep Transformer with Depth-Scaled Initialization and Merged Attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 898–909. Hong Kong, China: Association for Computational Linguistics.

Zhang, B.; Xiong, D.; and Su, J. 2018. Accelerating Neural Transformer via an Average Attention Network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1789–1798. Melbourne, Australia: Association for Computational Linguistics.