Lessons on Parameter Sharing across Layers in Transformers

Sho Takase* **Shun Kiyono**
LINE Corporation
{sho.takase, shun.kiyono}@linecorp.com

Abstract

We propose a novel parameter sharing method for Transformers (Vaswani et al., 2017). The proposed approach relaxes a widely used technique, which shares the parameters of one layer with all layers such as Universal Transformers (Dehghani et al., 2019), to improve the efficiency. We propose three strategies: SEQUENCE, CYCLE, and CYCLE (REV) to assign parameters to each layer. Experimental results show that the proposed strategies are efficient in terms of the parameter size and computational time in the machine translation task. We also demonstrate that the proposed strategies are effective in the configuration where we use many training data such as the recent WMT competition. Moreover, we indicate that the proposed strategies are also more efficient than the previous approach (Dehghani et al., 2019) on automatic speech recognition and language modeling tasks.

1 Introduction

Transformer-based methods have achieved notable performance in various NLP tasks (Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020). In particular, Brown et al. (2020) indicated that the larger parameter size we prepare, the better performance the model achieves. However, the model which is composed of many parameters occupies a large part of a GPU memory capacity. Thus, it is important to explore a parameter efficient way, which achieves better performance than a basic model with the same parameter size.

Parameter sharing is a widely used technique as a parameter efficient way (Dehghani et al., 2019; Dabre and Fujita, 2019; Lan et al., 2020). Dehghani et al. (2019) proposed Universal Transformer which consists of parameters for only one layer of a Transformer-based encoder-decoder, and uses these parameters N times for an N-layered encoder-decoder. Dabre and Fujita (2019) and Lan et al. (2020) also used such parameter sharing across layers for their Transformers.

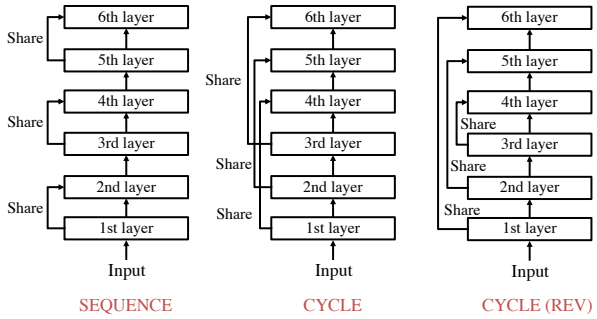

Figure 1: Examples of three parameter assignment strategies proposed in this study when we set $M = 3$ and $N = 6$.

Dehghani et al. (2019) reported that Universal Transformer achieved better performance than the vanilla Transformer in machine translation if the parameter sizes of both models are (almost) the same. However, when we prepare the same number of parameters for Universal Transformer and vanilla Transformer, the dimension sizes of each layer in Universal Transformer are much larger than ones in the vanilla Transformer. Thus, Universal Transformer requires much more computational time since its weight matrices are larger. For example, Universal Transformer requires twice as much training time as the vanilla Transformer in WMT English-to-German dataset, which is a widely used machine translation dataset (see Table 1).

In this paper, we propose a new parameter sharing method that is faster than using the same parameters for all layers such as Universal Transformers. Universal Transformers raise their expressiveness power by increasing the size of weight matrices for each layer. On the other hand, stacking (more) layers is another promising approach to raise expressiveness power of neural methods (He et al., 2016). Thus, the most straight-forward way to

* A part of this work was done when the author was at Tokyo Institute of Technology.

Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), pages 78–90

make Universal Transformers faster is stacking layers with smaller weight matrices for each layer. However, the approach using the same parameters for all layers limits the improvement of stacking layers (Dabre and Fujita, 2019). Therefore, instead of preparing parameters for only one layer, we prepare parameters for M layers to construct an N-layered encoder-decoder, where $1 \leq M \leq N$. In other words, the proposed method relaxes the parameter sharing strategy in previous studies (Dehghani et al., 2019; Dabre and Fujita, 2019; Lan et al., 2020). Because this relaxation addresses the above limitation of improvement by stacking layers, the proposed method can be fast by stacking layers with using small weight matrices for each layer. For the actual parameter assignment strategies, we provide several simple examples (Figure 1) and investigate their performance empirically. The main focus of this study is to demonstrate that such simple strategies can be a better alternative to the existing parameter sharing strategy used in Universal Transformers.

We mainly conduct experiments on machine translation datasets. Experimental results show that the proposed method achieves slightly better scores to the previous method, that assigns parameters of one layer to all layers, with smaller computational time. In addition, we indicate that the proposed method outperforms the previous parameter sharing method when we spend almost the same training time. Moreover, we conduct experiments on automatic speech recognition and language modeling tasks (Section 4 and Appendix A). Experimental results on these tasks also indicate that the proposed method are also efficient in these situations.

2 Proposed Method

As described in Section 1, we use parameters for M layers in the construction of an N-layered Transformer-based encoder-decoder. We provide three examples for the parameter assignment: SEQUENCE, CYCLE, and CYCLE (REV). This section describes these parameter assignment strategies.

Figure 1 shows examples of three parameter assignment strategies for an encoder side when we set $M = 3$ and $N = 6$. Let enc_i be the i-th layer of an encoder. Figure 2 describes the algorithm to assign each parameter to each layer of the encoder. For the decoder side, we assign each parameter with the same manner.

Algorithm Encoder Construction

Input: the total number of layers N, number of independent layers M, sharing strategy TYPE $\in$ {SEQUENCE, CYCLE, CYCLE (REV)}

Output: enc_1, ..., enc_N

for i in $[1, ..., N]$ **do**
  if $i == 1$ **then**
    $\text{enc}_i \leftarrow$ CreateNewLayer
  else if TYPE == SEQUENCE **then**
    if $(i-1) \bmod \lfloor N/M \rfloor == 0$ **then**
      $\text{enc}_i \leftarrow$ CreateNewLayer
    else
      $\text{enc}_i \leftarrow \text{enc}_{i-1}$
  else if TYPE == CYCLE **then**
    if $i \leq M$ **then**
      $\text{enc}_i \leftarrow$ CreateNewLayer
    else
      $\text{enc}_i \leftarrow \text{enc}_{((i-1) \bmod M)+1}$
  else if TYPE == CYCLE (REV) **then**
    if $i \leq M$ **then**
      $\text{enc}_i \leftarrow$ CreateNewLayer
    else if $i \leq (M \times (\lceil N/M \rceil - 1))$ **then**
      $\text{enc}_i \leftarrow \text{enc}_{((i-1) \bmod M)+1}$
    else
      $\text{enc}_i \leftarrow \text{enc}_{M-((i-1) \bmod M)}$

Figure 2: Proposed parameter assignment strategies for encoder construction. CreateNewLayer is a function that creates a new encoder layer.

2.1 SEQUENCE

The simplest strategy is to assign the same parameters to sequential $\lfloor N/M \rfloor$ layers. We name this strategy SEQUENCE. For example, when we set $M = 3$ and $N = 6$, two sequential layers share their parameters as illustrated in Figure 1.

2.2 CYCLE

In CYCLE, we stack M layers whose parameters are independent from each other. Then, we repeat stacking the M layers with the identical order to the first M layers until the total number of layers reaches N. When we set $M = 3$ and $N = 6$, we stack 3 layers twice as illustrated in Figure 1.

2.3 CYCLE (REV)

Liu et al. (2020) and Takase et al. (2022) reported that higher decoder layers tends to obtain larger

gradient norms[1]. Their report implies that higher layers require more degrees of freedom than lower layers for their expressiveness. In other words, lower layers probably have redundant parameters compared to higher layers. Thus, we propose the CYCLE (REV) strategy reusing parameters of lower layers in higher layers.

In this strategy, we repeat stacking M layers in the same manner as CYCLE until $M * (\lceil N/M \rceil - 1)$ layers. For the remaining layers, we stack M layers in the reverse order. When we set $M = 3$ and $N = 6$, we stack 3 layers and then stack the 3 layers in the reverse order as in Figure 1. Thus, the lowest layer and highest layer share parameters.

3 Experiments on Machine Translation

We investigate the efficiency of the proposed parameter sharing strategies. In detail, we indicate that our proposed strategies are faster than Universal Transformers while achieving comparable (or better) performance when we use the same parameter size. In this section, we conduct experiments on machine translation datasets. First, we focus on the English-to-German translation task because this task is widely used in the previous studies (Vaswani et al., 2017; Ott et al., 2018; Dehghani et al., 2019; Kiyono et al., 2020). We conduct comparisons based on following aspects: (i) comparison with Universal Transformers in terms of efficiency and (ii) comparison with models without parameter sharing across layers to investigate whether our proposed strategies can achieve comparable (or better) performance to the models with larger memory footprint.

In addition to the widely used training data, we conduct experiments on a large amount of training dataset in the English-to-German translation task. Then, we investigate if our findings are consistent in other language direction (i.e., German-to-English) and other language pair (i.e., English-to-French and French-to-English). We describe details in the following subsections.

3.1 Standard Setting

3.1.1 Datasets

We used the WMT 2016 training dataset, which is widely used in previous studies (Vaswani et al., 2017; Ott et al., 2018; Takase and Kiyono, 2021). This dataset contains 4.5M English-German sentence pairs. Following previous studies, we constructed a vocabulary set with BPE (Sennrich et al., 2016b) in the same manner. We set the number of BPE merge operations at 32K and shared the vocabulary between the source and target languages. We measured case-sensitive detokenized BLEU with SacreBLEU (Post, 2018)[2].

3.1.2 Methods

For the proposed parameter assignment strategies, we fixed $M = 6$ and set $N = 12, 18$ based on the Vanilla configuration below. We compare the proposed strategies with the following baselines.

Vanilla: This is the original Transformer (base) setting in (Vaswani et al., 2017). To stabilize the training, we applied Admin (Liu et al., 2020). See Section 5 for more details of Admin.

Universal: As the parameter sharing strategy in previous studies such as Universal Transformers (Dehghani et al., 2019), we set $M = 1$[3]. In this setting, we increased the dimensions of each layer for a fair comparison in terms of the number of parameters. This configuration corresponds to the Universal Transformer base setting in (Dehghani et al., 2019). Moreover, we prepared the model using twice as many layers to investigate the effect of stacking many layers in Universal Transformers. We call this setting **Universal (deep)**. In addition, we prepared **Universal (small)** whose dimension sizes are the identical to ones of Transformer (base).

Furthermore, we prepare two models that consist of a large number of parameters for reference.

Vanilla (big): This is the original Transformer (big) setting in (Vaswani et al., 2017).

Vanilla (deep): We stacked layers until $N = 18$ in the Vanilla configuration.

[1] In particular, this property is observed during warm-up when we use the post layer normalization (Post-LN) setting, which is originally used in Vaswani et al. (2017) and widely used in machine translation.

[2] The BLEU score computed by SacreBLEU is often lower than the score obtained by the procedure of Vaswani et al. (2017) as reported in Ott et al. (2018). In fact, when we used the same procedure as Vaswani et al. (2017), SEQUENCE of $M = 6, N = 12$ in Table 1 achieved 29.40 in the averaged BLEU score in newstest2014 and the best model in Table 2 achieved 35.14 in the averaged BLEU score in newstest2014. However, since Post (2018) encouraged using SacreBLEU for the compatibility of WMT results, we used SacreBLEU.

[3] The original Universal Transformers (Dehghani et al., 2019) use the sinusoidal positional encoding for each layer and adaptive computation time technique (Graves, 2017) but we omitted them in this study to focus on the difference among parameter sharing strategies.

Method	M	N	#Params	Speed	2010	2011	2012	2013	2014	2015	2016	Avg.
Vanilla	6	6	61M	×2.02	24.14	21.93	22.25	26.14	27.05	29.59	34.23	26.48
Universal	1	6	63M	×1.00	24.37	22.33	22.70	26.40	27.65	30.24	34.60	26.90
Universal (deep)	1	12	63M	×0.52	24.42	22.30	22.61	26.52	27.76	29.75	34.01	26.77
Universal (small)	1	6	24M	×2.52	22.89	21.11	21.29	24.75	24.71	28.16	32.81	25.10
SEQUENCE	6	12	61M	×1.31	24.65	22.32	22.83	**26.98**	27.88	30.27	**34.99**	**27.13**
CYCLE	6	12	61M	×1.31	24.51	22.43	22.69	26.61	**27.91**	**30.37**	34.77	27.04
CYCLE (REV)	6	12	61M	×1.31	**24.66**	**22.47**	**22.87**	26.68	27.72	**30.37**	34.81	27.08
SEQUENCE	6	18	61M	×0.98	24.53	22.44	22.73	26.59	27.73	30.30	34.80	27.02
CYCLE	6	18	61M	×0.98	24.74	22.60	23.04	**26.89**	**28.14**	30.54	34.79	27.25
CYCLE (REV)	6	18	61M	×0.98	**24.93**	**22.77**	**23.09**	26.88	28.09	**30.60**	**34.84**	**27.31**
Methods consisting of a large number of parameters for reference												
Vanilla (big)	6	6	210M	×0.81	24.31	22.21	22.75	26.39	28.28	30.35	33.40	26.81
Vanilla (deep)	18	18	149M	×0.96	24.54	22.30	22.75	26.57	28.03	30.24	34.19	26.94

Table 1: The number of layers, number of parameters, computational speeds based on the Universal configuration, BLEU scores on newstest2010-2016, and averaged scores when we trained each method on widely used WMT 2016 English-to-German training dataset. Scores in bold denote the best results for each set. The results of our proposed strategies are statistically significant ($p < 0.05$) in comparison with Universal. The lowest part indicates results of methods consisting of a large number of parameters for reference.

3.1.3 Results

Table 1 shows BLEU scores on newstest2010-2016 for each method. We trained three models with different random seeds, and reported the averaged scores. Table 1 also shows the total number of parameters and computational speeds[4]. The computational speed is based on the speed of Universal.

(i) Comparison with Universal in terms of efficiency In the comparison between Universal and Vanilla, Universal achieved better scores although their parameter sizes are almost the same. This result is consistent with the report in (Dehghani et al., 2019). However, the training time of Universal is more than twice as much as the one of Vanilla. In addition, Universal (deep) didn't improve the performance from Universal, and thus stacking many layers have small effect on BLEU scores when the model shares parameters of one layer with all layers.

In contrast, the proposed strategies (SEQUENCE, CYCLE, and CYCLE (REV)) were faster and achieved slightly better scores than Universal when we set $M = 6$ and $N = 12$. Thus, our proposed parameter sharing strategies are more efficient than Universal in terms of the parameter size and computational time.

In comparison among Universal (small) and the proposed strategies, Universal (small) was faster[5] but the configuration drastically sacrificed BLEU scores. These results imply that the strategy in Universal Transformer, which shares parameters of one layer with all layers, damages computational time or the quality of output sequences. In comparison with those Universal configurations, our proposed strategies improved both of the computational speed and BLEU scores.

Figure 3 illustrates the negative log-likelihood (NLL) values on newstest2013 for each training step. In this figure, we used $M = 6$ and $N = 12$ for our proposed strategies. This figure shows that Universal achieved better NLL values in the beginning of the training but the proposed strategies outperformed others when the training step is larger than 15,000. When we have finished training, the proposed strategies achieved better NLL values than Universal (and Vanilla). This result also indicates that the proposed strategies achieved better performance. We emphasize that the proposed strategies reached this better performance with small computational time in comparison with Universal because the proposed strategies are faster as in Table 1.

(ii) Comparison with models without parameter sharing across layers The lowest part of Table 1 indicates results when we prepared more parameters. We trained these models to investigate the performance of models without parameter sharing across layers. In other words, the purpose of these settings are comparison with models using larger memory footprint. As shown in Table 1, the proposed strategies achieved better performance than

[4]We regard processed tokens per second during the training as the computational speed.

[5]We used the same dimension sizes for Vanilla and Universal (small) but their training speeds are different from each other. Since Universal (small) consists of small parameters, the computational time for updating is smaller than Vanilla.

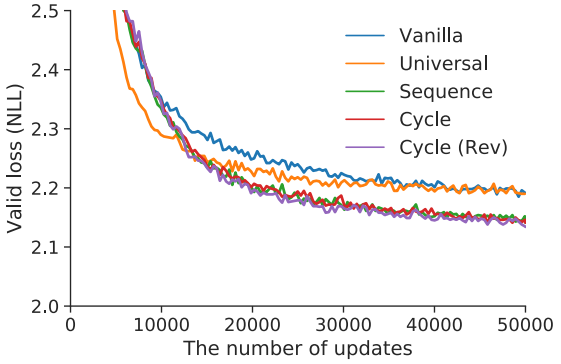

Figure 3: Negative log-likelihood (NLL) of each method on newstest2013. For our proposed parameter sharing strategies, we used $M = 6$ and $N = 12$.

models consisting of a large number of parameters in the averaged BLEU scores of newstest2010-2016. This result implies that the proposed parameter sharing strategies are not only efficient but also effective in constructing better encoder-decoder models.

3.2 High Resource Setting

3.2.1 Datasets

In the high resource setting, we constructed 44.2M translation sentence pairs as a training dataset with the procedures of (Kiyono et al., 2020) which achieved the best result in the WMT 2020 news translation task. In addition, we augmented the training data by using the back-translation technique (Sennrich et al., 2016a) in the same manner as (Kiyono et al., 2020). We obtained 284.3M pairs as synthetic training data. For evaluation, we add newstest2018 and 2019 to the set used in Section 3.1 to because (Kiyono et al., 2020) used these two test sets. In the same as Section 3.1, we measured case-sensitive detokenized BLEU with SacreBLEU.

3.2.2 Methods

We used the original Transformer (big) setting (Vaswani et al., 2017) as our baseline in using genuine training data. We call this setting **Vanilla** in this experiment. Moreover, we also prepared **Universal**, which shares the parameters with all layers, namely, $M = 1, N = 6$. We increased the dimensions of each layer in Universal to make their parameter size almost the same as others. For the proposed strategies, we used $M = 6$ and $N = 12$.

In using both of the genuine and synthetic (back-translated) datasets, we applied CYCLE (REV) to the BASE setting in (Kiyono et al., 2020) because CYCLE (REV) achieved the best BLEU scores on most test sets in Table 1. We also used $M = 6$ and $N = 12$ in this configuration. We compare the reported scores of the best model in (Kiyono et al., 2020). Their model is composed of 9 layers (i.e., $M = 9$ and $N = 9$); thus, it contains considerably more parameters than ours.

3.2.3 Results

Table 2 shows BLEU scores of each method on each test set. Similar to the experiments in Section 3.1, we reported the averaged scores of three models trained with different random seeds. Table 2 also shows the total number of parameters[6].

Table 2 shows that the proposed strategies achieved better BLEU scores than Vanilla and Universal when we prepared almost the same number of parameters. This result indicates that the proposed strategies are also parameter efficient in the high resource setting. In addition, since we used $M = 6$ and $N = 12$ for proposed strategies, they are also more efficient than Universal in terms of computational time (see Table 1).

When we used additional synthetic data for training in the same manner as (Kiyono et al., 2020), CYCLE (REV) achieved comparable BLEU scores to the best system of (Kiyono et al., 2020) except for newstest2019[7] even though the parameter size of CYCLE (REV) was smaller than theirs. This result indicates that CYCLE (REV) is also efficient in the construction of models for recent competitive tasks. In addition, this result implies that our proposed strategies can be used in the configuration where we train many parameters with a tremendous amount of data such as recent pre-trained language models, e.g., GPT series (Brown et al., 2020). We investigate the effect of the proposed strategies on language models in Appendix A.

3.3 Other Direction and Language Pair

3.3.1 Datasets

We conduct experiments on the other direction and language pair. For the German-to-English training dataset, we used the identical data in Section 3.1. For English-to-French and French-to-English, we

[6] The parameter sizes of Vanilla (big) in Table 1 and Vanilla in Table 2 are different from each other due to the difference of sharing embeddings. Following (Kiyono et al., 2020), we did not share embeddings in the high resource setting.

[7] For newstest2019, synthetic data might harm the quality of a model because models trained with only genuine data outperformed those trained with both data.

Method	#Params	2010	2011	2012	2013	2014	2015	2016	2018	2019	Avg.
Genuine training data											
Vanilla	242M	26.53	24.09	24.51	28.51	31.40	33.52	39.08	47.11	42.80	33.06
Universal	249M	27.00	24.20	24.96	28.94	31.73	33.53	39.38	47.54	43.11	33.38
SEQUENCE	242M	27.31	24.24	24.86	29.15	31.90	33.84	39.93	48.15	43.12	33.61
CYCLE	242M	27.23	24.45	25.13	29.12	**32.10**	**34.04**	39.82	48.11	43.19	33.69
CYCLE (REV)	242M	**27.37**	**24.46**	**25.14**	**29.16**	32.06	33.98	**40.28**	**48.34**	**43.43**	**33.80**
+ Synthetic (back-translated) data											
Kiyono et al. (2020)	514M	-	-	-	-	33.1	-	-	**49.6**	**42.7**	-
CYCLE (REV)	343M	**28.29**	**24.99**	**25.98**	**30.01**	**33.54**	**34.93**	**41.37**	**49.55**	42.18	**34.54**

Table 2: BLEU scores on newstest2010-2016, 2018, and 2019. We add newstest2018 and 2019 to the set in the standard setting to compare the top system on WMT 2020 (Kiyono et al., 2020).

Method	M	N	German-to-English 2013	German-to-English 2014	English-to-French 2013	English-to-French 2014	French-to-English 2013	French-to-English 2014
Vanilla	6	6	30.48	30.96	33.41	38.41	33.48	36.06
Universal	1	6	31.06	31.32	33.58	38.84	33.83	37.11
SEQUENCE	6	18	31.31	31.97	34.49	**40.18**	**34.26**	37.45
CYCLE	6	18	**31.46**	**32.18**	34.50	40.17	33.97	**37.59**
CYCLE (REV)	6	18	31.32	32.12	**34.67**	40.13	34.16	37.32

Table 3: The number of layers and BLEU scores on each dataset. Each method is composed of almost the same number of parameters.

used the WMT 2014 training dataset. We applied the same pre-processing as in (Ott et al., 2018), and used 35.8M English-French sentence pairs. Each configuration, we used newstest2013 and newstest2014 as valid and test sets, respectively. We also measured case-sensitive detokenized BLEU with SacreBLEU in these experiments.

3.3.2 Methods

We compare our proposed strategies with baselines used in Section 3.1. We used the Transformer (base) setting with Admin as **Vanilla** and prepared **Universal** which is $M = 1, N = 6$ with large dimension sizes for each internal layer. For the proposed strategies, we used $M = 6$ and $N = 18$. In these configurations, the training time of proposed strategies are almost the same as one of Universal as described in Table 1.

3.3.3 Results

Table 3 shows BLEU scores of each method on each dataset. This table indicates that Universal outperformed Vanilla in all datasets. The proposed parameter sharing strategies (SEQUENCE, CYCLE, and CYCLE (REV)) achieved better scores than Universal in all datasets. These results are consistent with results in Table 1. These results also indicate that the proposed strategies are more efficient than Universal, which shares parameters of one layer with all layers, because they achieved better performance with almost the same parameter size and computational time.

In the comparison among the proposed strategies, CYCLE and CYCLE (REV) outperformed SEQUENCE on German-to-English but it is difficult to conclude that CYCLE and CYCLE (REV) are superior to SEQUENCE on English-to-French and French-to-English. This result implies that the best strategy might depend on a language pair[8]. However, we emphasize that our proposed strategies outperformed Universal. For applying our proposed parameter sharing strategies to other datasets, we recommend using SEQUENCE as a first step because it is the easiest to implement.

4 Experiments on Automatic Speech Recognition

4.1 Datasets

To investigate the effect of our proposed strategies on other modality, we conduct comparisons on the automatic speech recognition (ASR) task. We used the de-facto standard English ASR benchmark dataset: LibriSpeech (Panayotov et al., 2015). The dataset contains 1,000 hours of English speech from audiobooks. We used the standard splits of LibriSpeech; used all available training data for training and two configurations (clean and other) of development and test sets for evaluation. We

[8] Section 4 and Appendix A imply that a sort of task and Transformer architectures also have an influence on the performance of proposed strategies.

Method	Enc M	Enc N	Dec M	Dec N	#Params	Speed	Dev clean	Dev other	Test clean	Test other
Vanilla	6	6	6	6	52M	×2.94	3.98	9.06	4.18	9.18
Universal	1	6	1	6	54M	×1.00	3.73	8.85	4.14	8.80
SEQUENCE	8	16	4	8	50M	×1.41	3.16	**7.84**	**3.32**	**7.71**
CYCLE	8	16	4	8	50M	×1.41	3.28	7.86	3.57	7.97
CYCLE (REV)	8	16	4	8	50M	×1.41	**3.11**	8.10	3.60	8.11

Table 4: The parameter sizes, computational speeds based on the Universal configuration, and word error rates of each method. For word error rates, lower is better. Scores in bold denote the best results for each set.

applied the same pre-processing as in (Wang et al., 2020). We measured word error rate on each set.

4.2 Methods

We also compare our proposed strategies with baselines in Section 3. As the base architecture, we used Transformer based speech-to-text model (T-Md) described in (Wang et al., 2020). In contrast to the Post-LN architecture, which is the original Transformer architecture (Vaswani et al., 2017), the Transformer in T-Md consists of the Pre-LN configuration. We prepared 6 layers for the encoder and decoder in **Vanilla** and **Universal**. For proposed strategies, we stacked more layers for the encoder side in the same as in (Wang et al., 2020). We prepared $N = 16$ and $M = 8$ for the encoder side, and $N = 8$ and $M = 4$ for the decoder side.

4.3 Results

Table 4 shows word error rates of each method on each dataset. This table indicates that Universal outperformed Vanilla in all sets. The proposed parameter sharing strategies (SEQUENCE, CYCLE, and CYCLE (REV)) achieved better scores than Universal in all sets even though they are faster than Universal. These results are consistent with results in machine translation experiments in Section 3. Thus, the proposed strategies are also more efficient in the ASR task.

In contrast to machine translation experiments, SEQUENCE outperformed CYCLE and CYCLE (REV) in the ASR task. We consider that this result might be caused by the difference of tasks. In addition, the cause might be the difference of layer normalization positions in the Transformer architecture. We used Post-LN based method (Admin) (Liu et al., 2020) in machine translation experiments, but Pre-LN based method in this ASR task. Liu et al. (2020) and Takase et al. (2022) demonstrated that the position of the layer normalization has a strong effect on the property of Transformers. The experimental results in language modeling (Appendix A) also imply that SEQUENCE is more appropriate when we use the Pre-LN based Transformer. The main focus of this study is empirical comparisons to the widely used parameter sharing strategy, Universal (Dehghani et al., 2019), but we will address theoretical analyses on the training dynamics in the future to understand the relation between parameter sharing strategies and Transformer architectures.

5 Related Work

Parameter Sharing In the past decade, various studies reported that a large amount of training data improve the performance in NLP tasks (Suzuki and Isozaki, 2008; Brants et al., 2007; Mikolov et al., 2013; Sennrich et al., 2016a; Edunov et al., 2018). Moreover, recent studies indicated that the larger parameter size we prepare, the better performance the model achieves when we have a large amount of training data (Devlin et al., 2019; Brown et al., 2020). In fact, the best system on the WMT 2020 news translation task is composed of about 10 times as many parameters as the widely used Transformer (base) setting (Kiyono et al., 2020). However, due to the limitation on a GPU memory capacity, we have to explore a parameter efficient way, which achieves better performance while saving the parameter size.

Parameter sharing is a widely used technique as a parameter efficient way (Dehghani et al., 2019; Dabre and Fujita, 2019; Xia et al., 2019; Lan et al., 2020). Dehghani et al. (2019) proposed Universal Transformer. Their method requires parameters for only one layer (i.e., $M = 1$) of a Transformer-based encoder-decoder, and shares these parameters with N layers. Dabre and Fujita (2019) investigated the effectiveness of Transformer sharing parameters of one layer across all layers on various

translation datasets. Lan et al. (2020) used this parameter sharing strategy to construct a parameter efficient model. As reported in these studies, we can achieve better performance by the Transformer sharing parameters of one layer across all layers when we use the same parameter size as the original Transformer. However, this strategy requires much more computational time as described in Table 1 because weight matrices for each layer are much larger. To solve this problem, we propose a new parameter sharing strategies that prepare parameters for M layers and assign them into N layers, where $1 \leq M \leq N$. Experimental results show that our proposed strategies are more efficient than the method sharing parameters of one layer with across layers (Dehghani et al., 2019; Dabre and Fujita, 2019; Lan et al., 2020). In addition, experimental results imply that the proposed parameter sharing strategies are effective to improve the performance. In fact, in language modeling, previous studies demonstrated that the parameter sharing is useful to improve the performance (Melis et al., 2018; Merity et al., 2018; Takase et al., 2018),

Xia et al. (2019) proposed an encoder-decoder which shares parameters of the encoder part and decoder part. Xiao et al. (2019) proposed the method to share the attention weights to make the computation of Transformers fast. These techniques are orthogonal to our proposed method. Thus, we can combine them to improve the efficiency of parameters and computational time.

Training Acceleration In this study, we explore a parameter efficient method. On the other hand, recent studies proposed method to accelerate the training. Li et al. (2020) proposed a training strategy for a deep Transformer. Their strategy trains a shallow model and then stacks layers to construct a deep model. They repeat this procedure until the desired deep model. They indicated that their strategy was faster than the training of whole parameters of a deep Transformer. Takase and Kiyono (2021) compared regularization methods in terms of training time. Their experimental results show that the simple regularizations such as word dropout are more efficient than complex ones such as adversarial perturbations. We can use those findings to accelerate the training of our proposed strategies.

Deep Transformers To raise expressiveness power of Transformers, we stack many layers in the proposed method. The stability of training deep Transformers depends on their architectures (Nguyen and Salazar, 2019; Xiong et al., 2020; Liu et al., 2020). Transformer architectures can be categorized into two types based on the position of layer normalizations: Post-LN and Pre-LN. Most of recent studies used the Pre-LN setting when they stacked many layers (Wang et al., 2019; Brown et al., 2020) because Pre-LN makes the training process more stable than the Post-LN setting, which is used in the original Transformer (Nguyen and Salazar, 2019; Xiong et al., 2020). On the other hand, several studies proposed methods to stabilize the training of Post-LN based Transformers (Liu et al., 2020; Takase et al., 2022). In this study, we used Admin (Liu et al., 2020) in machine translation experiments because it stabilizes the training of Post-LN based Transformers while keeping the advantages of Post-LN in the machine translation task. For other experiments, we used the Pre-LN configuration based on the implementations of baselines. These experiments show that our proposed strategies are effective in major two architectures: Post-LN and Pre-LN.

6 Conclusion

We proposed three parameter sharing strategies: SEQUENCE, CYCLE, and CYCLE (REV), for the internal layers in Transformers. In contrast to the previous strategy, which prepares parameters for only one layer and shares them across layers such as Universal Transformers (Dehghani et al., 2019), the proposed strategies prepare parameters for M layers to construct N layers. The proposed strategies stack layers whose weight matrices are smaller than ones of Universal Transformers to raise expressiveness power while saving computational time.

Experimental results in the standard machine translation setting show that the proposed strategies achieved slightly better BLEU scores to those of Universal with a small computational time when we prepared almost the same parameters for each method ($M = 6$ and $N = 12$). In addition, the proposed strategies outperformed Universal under the same computational budgets ($M = 6$ and $N = 18$). Thus, the proposed strategies are efficient in terms of the parameter size and computational time. Through additional experiments, we indicated that the proposed strategies are also more efficient than Universal in the high resource setting, other language pairs, and another modality (speech-to-text).

Limitations

As described in Section 1, the purpose of this study is to relax the existing parameter sharing strategy which shares the parameters of one layer with all layers (Dehghani et al., 2019; Dabre and Fujita, 2019; Lan et al., 2020). Experimental results indicate that the proposed simple parameter sharing strategies can be a better alternative to the existing method. As many studies on neural methods, this study also depend on empirical observations. In other words, this study lacks theoretical justifications for proposed parameter sharing strategies.

We conducted experiments on various situations. We mainly focused on sequence-to-sequence tasks and trained each model from scratch. Our conducted experiments indicated the efficiency of the proposed strategies but we did not conduct experiments on the pre-training and then fine-tuning configuration such as comparison with BERT (Devlin et al., 2019) due to the limitation of our computational budgets. Thus, it is difficult to claim that the proposed strategies are also more efficient in such configuration. In addition, we have to investigate the effectiveness in a more realistic situation. For example, we will investigate the performance of the combination of our proposed method, which is the parameter efficient way for internal layers, and a parameter efficient embedding such as Takase and Kobayashi (2020).

Through experiments in various configurations, it is difficult to conclude which strategy is the best. Experimental results imply that the best strategy depends on the task and Transformer architecture (Post-LN or Pre-LN). Such phenomena are reported in previous studies (Press et al., 2020; Gulati et al., 2020). In fact, the architecture explored by Press et al. (2020) is better in the language modeling task but ineffective in the machine translation task. Since it is intractable to investigate a tremendous amount of possible parameter assignment way due to the limitation of computational budgets, there might be a superior way to three simple strategies proposed in this paper. However, we emphasize that all our proposed strategies are more efficient than the Universal configuration. Because the purpose of our experiments is not to detect the best parameter sharing strategy but to indicate that our proposed parameter sharing strategies are more efficient than the Universal configuration, we consider that our conducted experiments are sufficient to verify our claims.

Ethics Statement

As discussed in Strubell et al. (2019), recent neural models require substantial energy consumption. To address this issue, we explore a parameter efficient way for Transformers in this study. We believe that our proposed strategies are effective to reduce the energy consumption.

On the other hand, we spent a large amount of computational costs to investigate the usefulness of our proposed strategies in various situations. Appendix B indicates our used GPUs and the number of updates that correspond to the computational costs.

Acknowledgements

We thank the anonymous reviewers for their insightful suggestions. A part of this work was supported by JSPS KAKENHI Grant Number JP21K17800 and JST ACT-X Grant Number JPMJAX200I.

References

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *Proceedings of ICLR*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-CoNLL*, pages 858–867.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901.

Raj Dabre and Atsushi Fujita. 2019. Recurrent stacking of layers for compact neural machine translation models. *Proceedings of AAAI*, 33:6292–6299.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of EMNLP*, pages 489–500.

Alex Graves. 2017. Adaptive computation time for recurrent neural networks.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 5036–5040.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778.

Shun Kiyono, Takumi Ito, Ryuto Konno, Makoto Morishita, and Jun Suzuki. 2020. Tohoku-AIP-NTT at WMT 2020 news translation task. In *Proceedings of WMT*, pages 145–155.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite bert for self-supervised learning of language representations. In *Proceedings of ICLR*.

Bei Li, Ziyang Wang, Hui Liu, Yufan Jiang, Quan Du, Tong Xiao, Huizhen Wang, and Jingbo Zhu. 2020. Shallow-to-deep training for neural machine translation. In *Proceedings of EMNLP*, pages 995–1005.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020. Understanding the difficulty of training transformers. In *Proceedings of EMNLP*, pages 5747–5763.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. *Proceedings of ICLR*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *Proceedings of ICLR*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, volume 26.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of IWSLT*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of WMT*, pages 1–9.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *ICASSP*, pages 5206–5210.

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

Ofir Press, Noah A. Smith, and Omer Levy. 2020. Improving transformer models by reordering their sublayers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2996–3005.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of ACL*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3645–3650.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL*, pages 665–673.

Sho Takase and Shun Kiyono. 2021. Rethinking perturbations in encoder-decoders for fast training. In *Proceedings of NAACL-HLT*, pages 5767–5780.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2022. B2t connection: Serving stability and performance in deep transformers. *arXiv preprint arXiv:2206.00330*.

Sho Takase and Sosuke Kobayashi. 2020. All word embeddings from one embedding. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 3775–3785.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of EMNLP*, pages 4599–4609.

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

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq S2T: Fast speech-to-text modeling with fairseq. In *Proceedings of AACL-IJCNLP*, pages 33–39.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning deep transformer models for machine translation. In *Proceedings of ACL*, pages 1810–1822.

Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. 2019. Tied transformers: Neural machine translation with shared encoder and decoder. *Proceedings of AAAI*, 33(01):5466–5473.

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

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of ICML*.

A Experiments on Language Modeling

A.1 Dataset

We focused Transformer-based encoder-decoders in the main experiments of this paper. However, recent studies often employed the decoder side only as a pre-trained model. Thus, we conduct experiments on the language modeling task to investigate the efficiency of our proposed strategies when we use the decoder side only. We used Wikitext-103 (Merity et al., 2017) which contains a large amount of training data. We measured perplexity of validation and test sets.

A.2 Methods

We used the Transformer with adaptive inputs (Baevski and Auli, 2019) as the base architecture. In the same as in Baevski and Auli (2019), the Transformer in the language modeling consists of the Pre-LN configuration. We set $N = 6$ for **Vanilla** and **Universal**. For the proposed strategies, we set $N = 12$ and $M = 6$.

A.3 Results

Table 5 shows perplexities of each method. This table indicates that Vanilla achieved better performance than Universal. Thus, the sharing parameters of one layer with all layers might not be suitable for a large-scaled language modeling task. In contrast, the proposed strategies outperformed Vanilla. This result indicates that our proposed strategies are also more efficient than Universal in the language modeling.

Through the comparison among proposed strategies, SEQUENCE achieved the best perplexity. As described in Section 4, SEQUENCE might be more appropriate to the Transformer with the Pre-LN configuration. To explore the reason, we believe that we have to conduct the theoretical analysis of the Transformer during its training. We address this issue in the future study.

The lower part of Table 5 shows the reported score of Baevski and Auli (2019), our reproduced score, and SEQUENCE with more parameters. This part indicates that SEQUENCE achieved better perplexities than others even though the parameter size of SEQUENCE is smaller. Therefore, SEQUENCE is also efficient when we prepare a large amount of parameters for a language model.

Method	#Params	Valid	Test
Vanilla	121M	20.39	21.13
Universal	121M	22.75	23.84
SEQUENCE	121M	**18.97**	**19.69**
CYCLE	121M	19.00	**19.69**
CYCLE (REV)	121M	19.60	20.24
Models with more parameters			
Baevski and Auli (2019)†	247M	18.53	19.24
Baevski and Auli (2019)	247M	-	18.7
SEQUENCE	234M	**17.71**	**18.55**

Table 5: The parameter sizes and perplexities of each method. The lower part indicates scores reported in Baevski and Auli (2019) and the score of SEQUENCE with more parameters. Scores in bold denote the best results for each set. † represents our re-run of Baevski and Auli (2019).

B Details of Experimental Settings

We used NVIDIATesla V100 GPUs for all experiments. Table 6 shows the hyper-parameters for training in each task. The descriptions in our code also help to understand configurations in this study.

Params	Machine Translation	ASR	Language Model
Leaning rate	0.001	0.001	0.001
Scheduler	inverse sqrt	inverse sqrt	inverse sqrt
Adam β	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.98)
Warmup updates	4k	4k	2k
Max updates	50k	150k	50k

Table 6: Hyper-parameters used in our experiments.