EdgeFormer: A Parameter-Efficient Transformer for On-Device Seq2seq Generation

Tao Ge **Si-Qing Chen** **Furu Wei**
Microsoft
{tage,sqchen,fuwei}@microsoft.com

Abstract

We introduce EDGEFORMER – a parameter-efficient Transformer for on-device seq2seq generation under the strict computation and memory constraints. Compared with the previous parameter-efficient Transformers, EDGEFORMER applies two novel principles for cost-effective parameterization, allowing it to perform better given the same parameter budget; moreover, EDGEFORMER is further enhanced by layer adaptation innovation that is proposed for improving the network with shared layers. Extensive experiments show EDGEFORMER can effectively outperform previous parameter-efficient Transformer baselines and achieve competitive results under both the computation and memory constraints. Given the promising results, we release EDGELM[1] – the pretrained version of EDGEFORMER, which is the first publicly available pretrained on-device seq2seq model that can be easily fine-tuned for seq2seq tasks with strong results, facilitating on-device seq2seq generation in practice.

1 Introduction

On-device modeling draws increasing attention for its unique advantages (Dhar et al., 2019). On the other hand, strict resource constraints prevent many neural networks performing well in the on-device setting. In Natural Language Processing (NLP), on-device sequence-to-sequence (seq2seq) generation remains challenging, especially for the Transformer (Vaswani et al., 2017) under strict resource constraints in both computation and memory.

To customize the Transformer for seq2seq tasks in the on-device setting, we propose EDGEFORMER – a novel parameter-efficient Transformer of the encoder-decoder architecture. EDGEFORMER is structurally similar to the standard Transformer with a deep encoder and shallow decoder, but with a modification that it uses an interleaved decoder with shared lightweight feed-forward networks, as shown in Figure 1. The modified decoder architecture allows EDGEFORMER to apply two novel principles that we propose for cost-effective parameterization: 1) encoder-favored parameterization that suggests we parameterize the encoder using as many parameters as possible; 2) load-balanced parameterization that suggests we balance the load of model parameters to avoid them being either underused or overused in a neural network with shared parameterization.

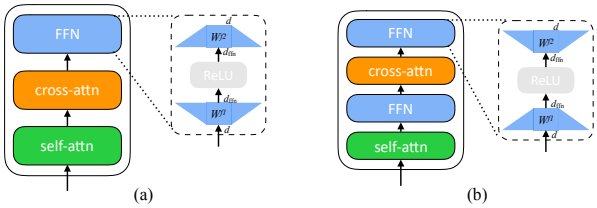


Figure 1: **(a)** Vanilla Transformer decoder layer in which $d_{\text{ffn}} > d$; **(b)** Interleaved Transformer decoder layer with shared lightweight FFNs in which $d_{\text{ffn}} < d$.

In addition to cost-effective parameterization, EDGEFORMER proposes and applies layer adaptation to further improve the model with tied layers, as Figure 2 shows. Inspired by parameter-efficient task transfer, we investigate 3 efficient layer adaptation approaches for improving the performance with negligible cost. We show EDGEFORMER (with fewer than 10 million model parameters) largely outperforms the strong UNIVERSAL TRANSFORMER baselines in the on-device setting with competitive results, and the int8-quantized EDGEFORMER can perform high-quality on-device seq2seq generation within around 100ms latency (20-30 sequence length on average) using two mid-to-high-end CPU cores and less than 50MB RAM.

The contributions of this work are three-fold:

- This paper is one of the earliest work that formally studies on-device seq2seq generation by discussing its challenges and defining

[1] https://github.com/microsoft/unilm/tree/master/edgelm

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 10786–10798
December 7-11, 2022

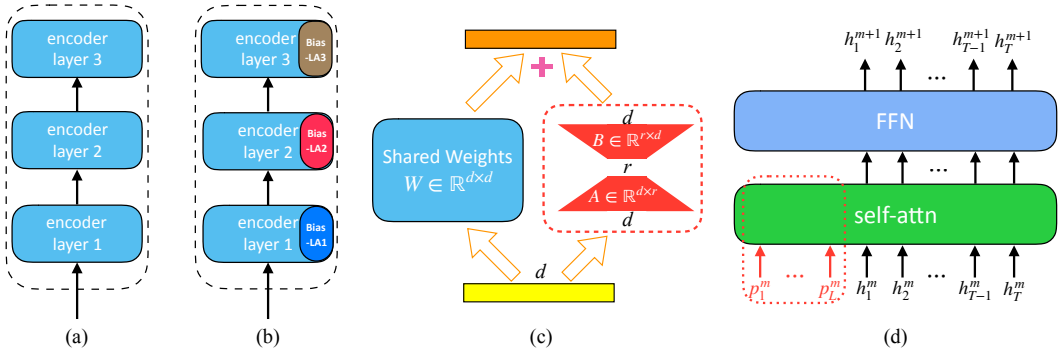


Figure 2: **(a)** Encoder layers with shared weights (the same color) without layer adaptation: the tied weights undermine the specialities of encoder layers to process their specific inputs; **(b)** Bias-based Layer Adaptation (Bias-LA) employs free bias terms to adapt layers with tied weights to fit their specific roles well; **(c)** Adapter-LA uses a layer-specific LoRA adaptation block with rank $r < d$ for layer adaptation; **(d)** Prefix-LA uses L layer-specific tokens (i.e., learnable parameters) as the prefix (dotted square) to adapt the mth layer.

a practical setting with appropriate resource constraints for the evaluation.

- We propose EDGEFORMER, a parameter-efficient Transformer with novel cost-effective parameterization and layer adaptation, achieving the state-of-the-art result in the on-device seq2seq generation setting under strict computing and memory resource constraints.

- We introduce and release EDGELM (the pre-trained EDGEFORMER) – the first publicly available pretrained on-device seq2seq model that can be easily fine-tuned for seq2seq tasks with strong results, which can largely reduce the effort for delivering a powerful on-device seq2seq model in practice.

2 Background: Transformer

2.1 Architecture

The Transformer follows the encoder-decoder architecture. The Transformer encoder consists of a stack of encoder layers, each of which has a self-attention module parameterized by projection matrices for the query, key, value and output: $[W^Q, W^K, W^V, W^O]$ whose shapes are all $d \times d$, followed[2] by a feed-forward network (FFN) parameterized by $W^{f1} \in \mathbb{R}^{d\times d_{ffn}}$ and $W^{f2} \in \mathbb{R}^{d_{ffn}\times d}$.

The Transformer decoder consists of a stack of decoder layers whose architecture is similar to an encoder layer except for an additional cross attention module between self-attention and FFN.

[2]For simplicity, we omit discussing the layer normalization and residual connection that are not related with this work.

In summary, we understand that the main parameters in an encoder layer i are:

$$\mathbf{\Phi}_{e_i} = [W_{e_i}^{\{Q,K,V,O\}}, W_{e_i}^{f1}, W_{e_i}^{f2}]$$

$|\mathbf{\Phi}_{e_i}| = 4d^2 + 2d \times d_{\text{encffn}}$. For a decoder layer j, its main parameters are:

$$\mathbf{\Phi}_{d_j} = [W_{d_j}^{\{Q,K,V,O\}}, W_{d_j}^{\{\mathcal{Q},\mathcal{K},\mathcal{V},\mathcal{O}\}}, W_{d_j}^{f1}, W_{d_j}^{f2}]$$

where $W_{d_j}^{\{\mathcal{Q},\mathcal{K},\mathcal{V},\mathcal{O}\}}$ is the cross-attention module. $|\mathbf{\Phi}_{d_j}| = 8d^2 + 2d \times d_{\text{decffn}}$.

2.2 Parameterization: Full vs Shared

Full Parameterization Full parameterization is a common parameterization approach for Transformer, meaning that each model parameter (excluding embedding) is independent without being shared by multiple modules in the network. In a forward pass, each parameter is used only once. Full parameterization allows parameters to be flexible to fit their roles well during model training.

Shared Parameterization Despite the advantages of full parameterization, it is criticized to use large numbers of parameters inefficiently, motivating shared parameterization where multiple modules in a network share parameters. For a model with shared parameterization (e.g., ALBERT), each model parameter is exploited more than once in a forward pass. The efficient parameterization can lead to a better result given the same parameterization budget, despite slowing down inference. On the other hand, given a fixed architecture (i.e., the same network depth and width), shared parameterization usually underperforms full parameterization because it has much fewer free model parameters.

Layer	Module	#params	$d = 512$ #params / FLOPS	$d = 384$ #params / FLOPS	$d = 768$ #params / FLOPS
encoder layer	self-attn FFN	$4d^2$ $8d^2$	3.15M / 95.4M	1.77M / 53.9M	7.08M / 214M
vanilla decoder layer	self-attn cross-attn FFN	$4d^2$ $4d^2$ $8d^2$	4.20M / 128M	2.37M / 72.3M	9.45M / 286M
interleaved decoder layer	self-attn cross-attn 2 shared lightweight FFNs	$4d^2$ $4d^2$ $d^2/2$	2.23M / 72.9M	1.25M / 41.3M	5.01M / 162M

Model	$d = 512$ #params / FLOPS	$d = 384$ #params / FLOPS	$d = 768$ #params / FLOPS
6+6 Transformer (full parameterization)	44M / 1.84G	25M / 1.13G	99M / 3.76G
12+2 Transformer (full parameterization)	46M / 1.90G	26M / 1.17G	104M / 3.89G
12+2 UNIVERSAL TRANSFORMER (shared parameterization)	7.4M / 1.90G	4.1M / 1.17G	16.5M / 3.89G
EdgeFormer (Ours)	8.6M / 1.79G	4.8M / 1.11G	19.2M / 3.65G

Table 1: **Top:** #params and FLOPS for Transformer layers. For the encoder and vanilla decoder layer, $d_{\text{ffn}} = 4d$; while for the interleaved decoder layer, $d_{\text{ffn}} = d/4$. **Bottom:** #params and FLOPS of whole models, where #params excludes embedding lookup, and FLOPS is measured on a sample with src/tgt length of 30 and 32K vocabulary.

3 Constraints for On-device Seq2seq

Computation On-device computer vision (CV) models tend to use 1G FLOPS (0.5G MACS) as a constraint, which is directly followed by previous work on on-device translation (Wu et al., 2020). In our work, however, we propose to relax the FLOPS constraint for typical seq2seq tasks to **2G FLOPS (1G MACS)** because the latency requirement for on-device seq2seq generation is not so rigid as CV tasks and it is uncommon for an on-device seq2seq model to handle too many concurrent requests in practice. The relaxed constraint allows better prediction quality that strongly correlates with user experience for seq2seq tasks, but still ensure the CPU on edge devices to process tens of sentences per second, which is more than sufficient for an on-device seq2seq model. In addition to FLOPS that is a theoretical hardware-independent measurement for computational cost, we also require the runtime latency for an input sentence (typically $20 \sim 30$ tokens on average) to be within **around 100ms** using two mid-to-high-end CPU cores.

Memory In contrast to deploying a model on a cloud server without caring about memory cost much, there is a very strict memory constraint for an on-device model in practice, because a user's edge device (e.g., PC) is not only for model hosting; instead, it usually runs many other (background) apps and programs at the same time besides the model. To ensure moderate memory cost, we limit the number of model parameters (excluding word embedding lookup table) up to **10 million**, following previous work (Wu et al., 2020), and require the runtime memory footprint to be **less than 50MB**.

4 EdgeFormer

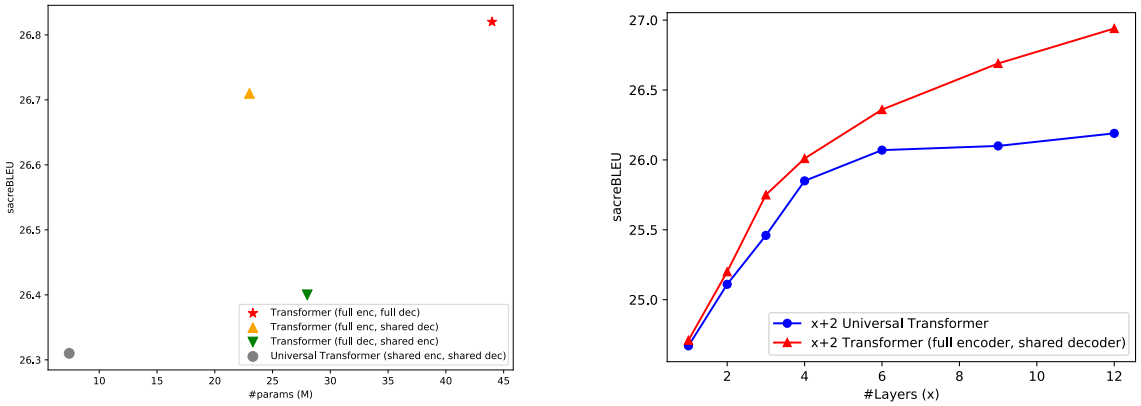
4.1 Architecture

The biggest challenge for an on-device seq2seq model is regarding the model size and memory cost. As shown in Table 1, the number of parameters of a standard Transformer-base model (d=512) is about 45 million (excluding the embedding parameters), which is far beyond the parameterization budget (10 million) and unavoidably leads to massive memory cost despite acceptable FLOPS.

EDGEFORMER is proposed to address the challenge. Instead of disruptive architectural changes[3] as previous research (Wu et al., 2020; Mehta et al., 2020; Panahi et al., 2021), EDGEFORMER's architecture basically follows the standard Transformer consisting of a 12-layer encoder and 2-layer[4] decoder, which is efficient in decoding. We mainly discuss the model with d=512 in this paper since it can achieve good performance in the on-device setting as long as it can be appropriately parameterized. The minor architectural modification we propose for EDGEFORMER is using an interleaved

[3]We basically follow the standard Transformer without major architectural changes because a standard Transformer should be more widely compatible and supported than a customized model architecture in user devices with various environments (e.g., hardware and runtime libraries).

[4]We do not use 1-layer decoder because it does not consistently perform well (Sun et al., 2021).



(a) (b)

Figure 3: **(a)** Performance of 6+6 Transformer ($d = 512$) on the newstest2013 English-German (En-De) translation dataset (dev set): densely parameterizing the decoder is uneconomical and much less beneficial than parameterizing the encoder; **(b)** Comparison of x+2 Transformer with full-/shared-parameterized x encoder layers on newstest2013 En-De dataset: when $x > 6$, the performance of the Transformer with shared parameterization only improves marginally even if x continues to increase.

decoder where attention modules are interleaved with shared lightweight[5] FFNs ($d_{\text{decffn}} < d$; in this work, $d_{\text{decffn}} = d/4$) in each decoder layer (shown in Figure 1). The modification is helpful for cost-effective parameterization (Section 4.2):

- The interleaved structure makes the architecture of encoder and decoder layers consistent (Ma et al., 2021), facilitating shared parameterization of attention modules throughout the encoder and decoder.
- As shown in Table 1, the lightweight FFNs that interleave attention modules in the decoder reduce FLOPS and save a large number of parameters for decoder FFNs' parameterization that is very uneconomical.

4.2 Cost-effective Parameterization

Due to the tight parameterization budget (i.e., 10 million), EDGEFORMER cannot be fully parameterized as in the standard way; instead, it has to adopt shared parameterization.

As a strong baseline for shared parameterization, UNIVERSAL TRANSFORMER lets all its M encoder layers share 1 group of encoder layer parameters and all its N decoder layers share 1 group of decoder layer parameters:

$$\mathbf{\Phi}_{e_1} \overset{\text{tied}}{=} \mathbf{\Phi}_{e_2} \overset{\text{tied}}{=} \cdots \overset{\text{tied}}{=} \mathbf{\Phi}_{e_M}$$

$$\mathbf{\Phi}_{d_1} \overset{\text{tied}}{=} \mathbf{\Phi}_{d_2} \overset{\text{tied}}{=} \cdots \overset{\text{tied}}{=} \mathbf{\Phi}_{d_N}$$

[5] As observed by Kasai et al. (2020), reducing d_{decffn} does not hurt the result much, as shown in Table 8 in Appendix A.

Although UNIVERSAL TRANSFORMER is a popular solution to shared parameterization, it is still not cost-effective for two reasons:

First, UNIVERSAL TRANSFORMER uses (over) half of total parameters to parameterize the decoder, which is uneconomical. As shown in Figure 3a, given a fixed architecture (6+6 Transformer, $d = 512$), densely parameterizing the decoder results in much less performance gain than parameterizing the encoder. This suggests we use as many parameters as possible to parameterize the encoder for the performance.

Second, UNIVERSAL TRANSFORMER does not consider load balance of model parameters, which was a rarely discussed problem until the recent emergence of Mixture-of-Expert models (Fedus et al., 2021). For the Transformers with a deep encoder and shallow decoder, UNIVERSAL TRANSFORMER's parameterization method will overburden parameters in the encoder but underutilize parameters in the decoder. For example, for a 12+2 UNIVERSAL TRANSFORMER, a parameter in the encoder is used 12 times, while a parameter in the decoder is used only twice in a forward pass. As shown in Figure 3b, moderately reusing parameters (e.g., when $x \leq 4$) helps better utilize the parameters, resulting in significant performance gain without increasing parameters. However, as the shared parameters are overused (when $x > 6$), the performance gain will become marginal, which is intuitive because a parameter's capability is limited. This suggests we balance the load of parameters to

avoid them being either overused or underused.

Based on the above insights, we parameterize EDGEFORMER in the following two novel principles for cost-effective parameterization:

Encoder-favored Parameterization For EDGEFORMER, we parameterize its encoder using as many parameters as possible: except a small number of parameters ($d^2/2$) for all lightweight FFNs in the decoder, we use almost all parameters in our budget to parameterize the encoder. For attention modules in the decoder, we let them reuse (i.e., share) parameters with the attention modules in the encoder since attention modules in both the encoder and decoder work in the same mechanism and can be effectively shared (Dong et al., 2019). Thanks to the interleaved decoder architecture that makes the structure of encoder and decoder layers consistent, we let the self-attention module in a decoder layer share parameters with its corresponding odd layer in the encoder, and let its cross-attention module share with the corresponding even layer in the encoder, inspired by Ma et al. (2021):

$$W_{d_j}^{[Q,K,V,O]} \overset{\text{tied}}{=} W_{e_{2j-1}}^{[Q,K,V,O]} \quad (1 \le j \le 2)$$

$$W_{d_j}^{[\mathcal{Q},\mathcal{K},\mathcal{V},\mathcal{O}]} \overset{\text{tied}}{=} W_{e_{2j}}^{[Q,K,V,O]} \quad (1 \le j \le 2)$$

Load-balanced Parameterization We try parameterizing EDGEFORMER with a balanced load for each model parameter so that each parameter could be as equally exploited as possible in a forward pass. Given the parameterization budget and the load balance principle, we create 2 groups of encoder FFN parameters equally shared by all encoder layers, 1 group of decoder FFN parameters is shared by light FFNs in the decoder, and 4 groups of attention parameters are shared throughout the encoder and decoder. Except for parameters in the encoder FFNs that are used 6 times, other parameters are all used 4 times in a forward pass, resulting in a load balanced parameterization:

$$W_{e_i}^{[Q,K,V,O]} \overset{\text{tied}}{=} W_{e_{i+4}}^{[Q,K,V,O]} \quad (1 \le i < 9)$$

$$W_{e_i}^{[f_1,f_2]} \overset{\text{tied}}{=} W_{e_{i+2}}^{[f_1,f_2]} \quad (1 \le i < 11)$$

$$W_{d_j}^{[f_1,f_2]} \overset{\text{tied}}{=} W_{d_1}^{[f_1,f_2]} \quad (1 \le j \le 2)$$

4.3 Layer Adaptation

Shared parameterization causes layers with tied weights to become less specialized, as discussed in Section 1. To allow tied layers to be better adapted to their corresponding roles, we propose layer adaptation to further enhance EDGEFORMER. Inspired by parameter-efficient task transfer methods, we investigate three efficient layer adaption approaches:

Bias-based Layer Adaptation (Bias-LA) Inspired by BitFit (Ben Zaken et al., 2021) fine-tuning with only bias terms, we untie all bias terms of each layer and use them to specialize the layers with tied weights, as shown in Figure 2(b). As BitFit, bias-based layer adaptation introduces very few additional parameters without inference overhead.

Adapter-based Layer Adaptation (Adapter-LA) Adapter-based approaches (Houlsby et al., 2019) introduce adapter modules for NLP task transfer without full fune-tuning. We borrow this idea for layer adaptation by introducing an independent adapter module for each layer. Specifically, we adopt the recently proposed LoRA (Hu et al., 2021) as our layer adapter, as Figure 2(c) shows. In our experiments, we apply the layer adapter to W^Q and W^V, as the original paper of LoRA suggests.

Prefix-based Layer Adaptation (Prefix-LA) Inspired by recent work (Li & Liang, 2021; Lester et al., 2021) using a prefix/prompt for task transfer, we introduce L tokens with learnable parameters as a specific prefix for each layer to adapt layers with tied weights, as shown in Figure 2(d). The prefixs are only used for keys and values in attention modules, which will not introduce much inference overhead as long as L is moderately set.

Following the encoder-favored principle in Section 4.2, we only apply LA to encoder layers.

5 Experiments

5.1 Experimental Setting

We mainly evaluate our approach in Machine Translation (MT). We select the most popular MT benchmark – WMT14 English-German (En-De) translation task, which is also a touchstone for seq2seq evaluation, as our main test bed. To compare with previous work, we also evaluate WMT14 English-French (En-Fr) translation. We follow the standard way to train and evaluate evaluate WMT14 En-De and En-Fr. As Ott et al. (2018), we use a joint source-target dictionary of 32K Byte Pair Encoding (BPE) for En-De, and 40K BPE for En-Fr. We mainly use sacreBLEU (Post, 2018) for evaluation.

We select UNIVERSAL TRANSFORMER which is the most popular and a strong baseline of parameter-

Model	#Params	FLOPS	sacreBLEU
Teacher	176M	6.7G	29.3
6+6 Transformer (full enc, full dec)	44M	1.8G	28.5
6+6 Transformer (full enc, shared dec)	23M	1.8G	28.2
6+6 Transformer (full dec, shared enc)	28M	1.8G	27.3
12+2 Transformer (full enc, full dec)	46M	1.9G	28.5
12+2 Transformer (full enc, shared dec)	42M	1.9G	28.4
12+2 Transformer (full dec, shared enc)	12M	1.9G	27.2
12+2 UT	7.4M	1.9G	27.0
12+2 UT ($d_{\text{ffn}} = 2560$)	8.5M	2.1G	27.2
12+2 UT ($d_{\text{encffn}} = 3072$)[1]	8.5M	2.3G	27.4
12+2 UT ($d_{\text{decffn}} = 3072$)	8.5M	2.0G	27.0
EDGEFORMER w/o LA[2]	8.6M	1.8G	$27.7^{\dagger(1)}$
EDGEFORMER (Bias-LA)	8.6M	1.8G	27.8
EDGEFORMER (Adapter-LA) ($r = 32$)	9.4M	1.8G	$28.0^{\dagger(2)}$
EDGEFORMER (Prefix-LA) ($L = 8$)	8.6M	1.9G	$28.0^{\dagger(2)}$

Table 2: WMT14 En-De results. To fairly compare with UNIVERSAL TRANSFORMER (UT) that is originally smaller than EDGEFORMER, we also test UT with larger FFNs to make its model size comparable to EDGEFORMER. $^{\dagger(i)}$ denotes $p < 0.05$ in significance test compared with the model marked with ^{i}.

FFNs	Load	#Params	FLOPS	sacreBLEU
2 FFNs ($d_{\text{ffn}} = 2048$)[1]	6-6	8.6M	1.8G	27.7
3 FFNs ($d_{\text{ffn}} = 1536$)	4-4-4	9.1M	1.6G	$27.4^{\dagger(1)}$
4 FFNs ($d_{\text{ffn}} = 1024$)	3-3-3-3	8.6M	1.4G	$27.2^{\dagger(1)}$
2 FFNs ($d_{\text{ffn}} = 2048$)	1-11	8.6M	1.8G	$27.5^{\dagger(1)}$
2 FFNs ($d_{\text{ffn}} = 2048$)	11-1	8.6M	1.8G	$27.4^{\dagger(1)}$

Table 3: Performance of EDGEFORMER with various encoder FFN parameterization on WMT14 En-De. Load 6-6 means the 2 groups of FFN parameters are used 6 times each, while Load 1-11 means 1 group of FFN is used once, and the other is used 11 times.

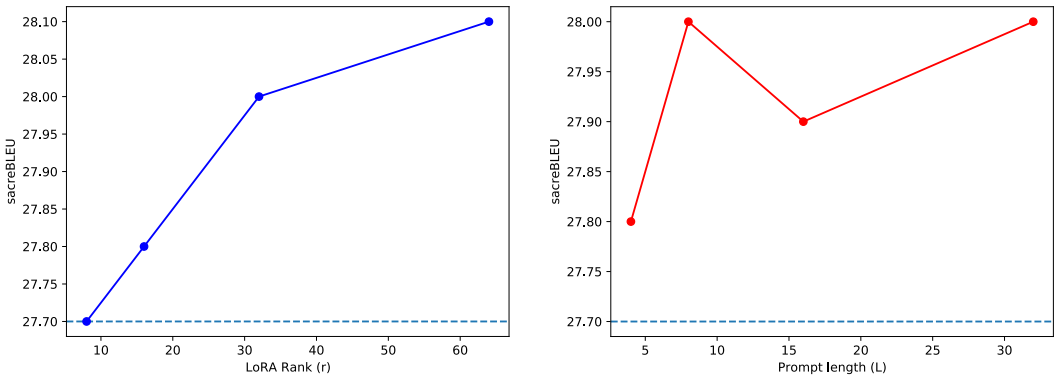
efficient Transformer for fair comparison. By default, we apply Seq-KD (Kim & Rush, 2016) to train models and use the full-parameterized 6+6 Transformer-big ($d = 1,024$) model (Vaswani et al., 2017; Ott et al., 2018) as the teacher.

By default, for each experiment, we train 5 models with different initializations and report their average evaluation results for Table 2, 3 and 6 with significance test. For inference, we use beam search (beam=5).

5.2 Offline Evaluation

We evaluate EDGEFORMER and compare it with UNIVERSAL TRANSFORMER (UT) on WMT14 En-De. According to Table 2, the EDGEFORMER without layer adaptation (LA) largely outperforms UTs. Among the LA approaches, both Adapter-LA and Prefix-LA are clear to benefit the result with marginal computational or parameterization cost, while Bias-LA does not show significant performance gain though it is the cheapest.

As discussed in Section 4.2, the advantage of EDGEFORMER over UT comes from its cost-effective parameterization. The encoder-favored principle is again supported by comparing 6+6 Transformers' results in Table 2, which is consistent with the observation on the dev set in Figure 3a. To further understand the effectiveness of load-balanced parameterization principles, we conduct an ablation study by adjusting encoder FFNs in EDGEFORMER. Table 3 shows the results of EDGEFORMER with various FFN parameterization. As we reduce d_{ffn} (e.g., to 1536 or 1024), we can increase the group of encoder FFN parameters and reduce their load given a fixed parameterization budget. However, such a strategy leads to a clear degradation of sacreBLEU. One reason is that the FFN parameters of a reduced load (3-4 times) are not so fully utilized as the baseline (6 times) despite other reasons such as the differences of network shape (e.g., d_{ffn}). To minimize the effects of other factors, we compare the first group with a balanced



(a) (b)

Figure 4: The effects of **(a)** rank r in Adapter-LA, and **(b)** prefix length L in Prefix-LA on the performance in WMT14 En-De. Note that $r = 64$ will lead to exceed our parameterization budget despite better performance.

Model	#params	FLOPS	En-De	En-Fr
6+6 Transformer	44M	1.8G	28.3	41.0
12+2 Transformer	46M	1.9G	28.4	41.4
12+2 UT	7.4M	1.9G	26.2	39.2
DeLighT	31.4M	-	**27.6**	39.6
Shapeshifter	8.2M	-	26.6	**40.8**
Lite Transformer (small)	2.9M	0.2G	22.5	35.3
Lite Transformer (medium)	11.7M	0.7G	25.6	39.1
Lite Transformer (big)	17.3M	1.0G	26.5	39.6
EdgeFormer w/o LA	8.6M	1.8G	26.5	39.8
EdgeFormer (Adapter-LA)	9.4M	1.8G	**26.9**	**40.5**
EdgeFormer (Prefix-LA)	8.6M	1.9G	26.8	40.3

Table 4: Result comparison to previous parameter-efficient Transformers that have fewer parameters than the baseline Transformer (around 45M parameters). "-" means that the metrics are unavailable or not comparable in the original paper. The underlines denote that the metrics cannot meet the on-device requirement. Note that all the models in this table do not apply Seq-KD.

parameter load (i.e., 6-6) and the last group with a imbalanced parameter load (1-11 or 11-1), showing load-balanced parameterization is consistently better than the imbalanced counterparts.

After discussing parameterization, we then analyze the effects of layer adaptation on the results by mainly focusing on Adapter-LA and Prefix-LA that both show performance gain. Figure 4 shows the effects of the rank r in Adapter-LA and prefix length L in Prefix-LA. As r increases, the model performance will gradually improve. However, when r becomes large (e.g., $r \geq 64$), it will exceed our parameterization budget and thus the gain will become meaningless. As for prefix length L in Prefix-LA, it is different from r that it will not keep improving the results as it increases: the gain can hardly be observed after some length (e.g., $L = 8$), which is similar to the observation in prefix-tuning (Li & Liang, 2021). Therefore, we use $r = 32$ and $L = 8$ as the default setting to report the results of Adapter-LA and Prefix-LA.

Finally, we compare EDGEFORMER with recent work on parameter-efficient Transformer modeling. To keep consistency of the training and evaluation protocols with previous work, we here give up using Seq-KD to train the models, and report BLEU (Papineni et al., 2002) for comparison. Specifically, we compare with DeLighT (Mehta et al., 2020), Shapeshifter (Panahi et al., 2021) and Lite Transformer (Wu et al., 2020), and show the results in Table 4. However, it is notable that the results are not strictly comparable because the previous studies have their own focus and setting, which are different from ours. For example, DeLighT and Lite Transformer focus much more on FLOPS than the model size, thus they do not a desirable tradeoff between the model quality and size; while Shapeshifter's goal is minimizing the model size despite an additional $10\% \sim 20\%$ inference overhead. Regardless of these factors that

WMT14 En-De					
Model	**Disk size (7zip)**	**Peak Memory**	**Latency 1**	**Latency 2**	**sacreBLEU**
EdgeFormer (Adapter-LA, 32k vocab)	28MB	60MB	65ms	114ms	27.2
EdgeFormer (Adapter-LA, 8k vocab)	15MB	47MB	59ms	101ms	27.1
CoNLL-14					
Model	**Disk size (7zip)**	**Peak Memory**	**Latency 1**	**Latency 2**	$F_{0.5}$
EdgeFormer (Adapter-LA, 2k vocab)	11MB	42MB	51ms	98ms	50.8

Table 5: Runtime results for int8-quantized EDGEFORMER, in which Latency1 and Latency 2 denote the average latency per sentence measured on the Intel® Xeon® E-2288G CPU and Qualcomm SM8150 Snapdragon 855 CPU, respectively. We run through the test set with batch size=1, and use greedy decoding instead of beam search.

Model	**#Param**	**FLOPS**	**CoNLL14** $F_{0.5}$	**XSum** RG-1	RG-2	RG-L	**SQuAD-NQG** B4	MTR	RG-L
Transformer-base	44M	1.8G	50.1	31.2	10.7	24.9	2.6*	9.0*	26.0*
Pretrained 12+2 UT ($d_{ffn} = 2048$)	7.4M	1.4G	50.8	36.0	14.5	29.2	19.8	22.2	46.9
Pretrained 12+2 UT ($d_{ffn} = 3072$)[1]	9.4M	1.9G	51.1	36.7	14.9	29.7	20.1	22.4	47.1
EDGELM	9.4M	1.3G	**52.0**$^{(1)}$	**37.2**$^{(1)}$	**15.4**$^{(1)}$	**30.3**$^{(1)}$	**20.6**$^{(1)}$	**23.0**$^{(1)}$	**47.4**$^{(1)}$

Table 6: The performance of EDGELM in comparison with the baselines. $*$ denotes that the results are from Chen et al. (2019).

prevent fair comparison, EDGEFORMER achieves 26.9 BLEU in En-De under the strict on-device resource constraints, which outperforms the state-of-the-art Shapeshifter with the similar model size despite. It is notable that EDGEFORMER here uses the same model architecture and training configuration for both En-De and En-Fr, while Shapeshifter uses different model architecture configurations specific for En-De and En-Fr, which may account for its better performance in En-Fr.

5.3 Runtime Evaluation

We conduct experiments in WMT14 En-De translation and CoNLL-14 Grammatical Error Correction[6] (GEC) benchmark for runtime latency and memory evaluation using onnxruntime[7] that supports efficient seq2seq decoding. We apply int8-quantization to EDGEFORMER and test latency on 2 devices: a 2-core Intel® Xeon® E-2288G CPU (in PC), and a 2-core Qualcomm SM8150 Snapdragon 855 CPU (in Pixel 4), which are both current mid-to-high end CPUs launched 2-3 years ago.

Table 5 shows runtime evaluation results. With int8-quantization and smaller vocabulary, EDGEFORMER can not only meet the on-device seq2seq requirements but also maintain its good performance, demonstrating its practical values.

[6]We include experiments details of GEC in Appendix A.

[7]https://github.com/microsoft/onnxruntime

6 EdgeLM – The Pretrained EdgeFormer

Given the promising results, we introduce EDGELM – the pretrained[8] EDGEFORMER (Adapter-LA) with 8K sentenpiece vocabulary with factorized embedding ($d_{\text{embed}} = 128$) through the same self-supervised task (i.e., masked span infilling) as T5 (Raffel et al., 2019) and make it publicly available for downstream on-device seq2seq task fine-tuning.

We evaluate EDGELM in the benchmarks of three popular seq2seq tasks: CoNLL-14 for Grammatical Error Correction (GEC), XSum (Narayan et al., 2018) for Abstractive Summarization, and SQuAD-NQG (Du et al., 2017) for Question Generation. According to Table 6, EDGELM achieves significantly better performance than the pretrained UT models as well as the Transformer-base model trained from scratch. We believe that EDGELM, as the first publicly released on-device seq2seq pretrained model, can largely facilitate on-device seq2seq generation in practice.

7 Related Work

On-device seq2seq generation in NLP is a research area that has been less explored than on-device CV and NLU (Tambe et al., 2021). Besides the general techniques like pruning, compression, quantization and knowledge distillation (Fan et al., 2019;

[8]We include pretraining details in the Appendix B.

Xu et al., 2020; Li et al., 2022) that are orthogonal to our effort, parameter-efficient Transformer-based seq2seq modeling is the most related research branch to ours. In this branch, UNIVERSAL TRANSFORMER (Dehghani et al., 2018) uses cross-layer sharing method, which is the most popular solution to parameter efficiency. Takase & Kiyono (2021) extends UNIVERSAL TRANSFORMER by studying different ways for layer sharing, and Reid et al. (2021) proposes to free the first and last encoder layer and widen the intermediate layers for better performance. However, both the approaches consider parameter-efficiency only without caring about latency becoming worse.

In addition to work improving parameter efficiency by weight sharing, there is research that studies lightweight model architecture for seq2seq learning where early work (Gehring et al., 2017; Wu et al., 2019) mainly focuses on CNNs, while recent efforts have tended to switch to attention-based models such as Mehta et al. (2020). Also, low-rank factorization has been studied intensively to make the model tiny (Zhang et al., 2021; Panahi et al., 2021); and hardware-aware network architecture search with elastic modeling (Wang et al., 2020) has been proposed recently for facilitating deployment of seq2seq models on various devices. Among previous studies, the work of Wu et al. (2020) is the most related to ours, which studies seq2seq generation in an on-device setting. However, it sets the computational constraint for on-device seq2seq to be the same with the CV tasks, which is too strict and unnecessary, as discussed in Section 3. As a result, their models focus on FLOPS optimization much more than memory, leading to an undesirable tradeoff between the quality and model size for the practical on-device seq2seq setting which should care about memory much more than latency. In contrast, our work carefully evaluates bottleneck constraints, and proposes appropriate models with parameterization and layer adaptation innovations, largely improving the results for practical on-device seq2seq generation.

8 Conclusion and Future Work

We formally study on-device seq2seq generation, including defining its practical resource constraint setting and proposing an appropriate modeling technology EDGEFORMER. The cost-effective parameterization and layer adaptation innovations in EDGEFORMER both prove effective to improve the results with negligible computation and memory cost, achieving state-of-the-art results in the on-device seq2seq generation setting. Our released pretrained EDGEFORMER – EDGELM can be easily fine-tuned for downstream seq2seq tasks, largely facilitating on-device seq2seq generation in practice.

For future work, we plan to further study load-balanced parameterization for parameter-efficient models, which is an interesting and new but seemingly profound machine learning research problem: instead of naively assuming that all the parameters are equal in this preliminary study, we suspect that parameters in different modules (e.g., parameters in the self-attn and FFN; or parameters in different layers) should be under different amounts of load. We look forward to in-depth research on this problem, which might be helpful to deepen our understanding of neural networks.

9 Limitations

EDGEFORMER is a preliminary model proposed for on-device seq2seq generation setting, which still has much room for improvement. For example, as mentioned in Section 8, the current load balance mechanism naively assumes that the number of times that a parameter is used in a forward pass is equal to its load, which may not be always true because parameters in different moduels are different: some parameters may be effectively used more times than others, which requires deeper understanding of neural network and the Transformer.

Acknowledgments

We thank all the anonymous reviewers for their valuable comments. We thank Xiaohu Tang, Fucheng Jia, Yifan Yang and Huiqiang Jiang for their help in runtime evaluation, and thank Shuming Ma, Ting Cao, Fan Yang, Qiufeng Yin, Yuqing Yang and Lidong Zhou in Microsoft Research Asia for the discussion and helpful comments. We also appreciate the support from Wenbing Li, Yufeng Li, Bowen Bao, Ye Wang, Sunghoon Choi, Scott McKay and Emma Ning in Microsoft AI Frameworks for onnxruntime, and appreciate the feedback and valuable suggestions from Joshua Burkholder, Xun Wang, Weixin Cai and Zhang Li in Microsoft Office Intelligence regarding detailed constraints for on-device seq2seq generation in real-word applications.

References

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv e-prints*, pp. arXiv–2106, 2021.

Christopher Bryant, Mariano Felice, Øistein E Andersen, and Ted Briscoe. The bea-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 52–75, 2019.

Mengyun Chen, Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. Improving the efficiency of grammatical error correction with erroneous span detection and correction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7162–7169. Association for Computational Linguistics, 2020.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942*, 2019.

Daniel Dahlmeier and Hwee Tou Ng. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 568–572, 2012.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Sauptik Dhar, Junyao Guo, Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. On-device machine learning: An algorithms and learning theory perspective. *arXiv preprint arXiv:1911.00623*, 2019.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*, 2019.

Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*, 2017.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2019.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pp. 1243–1252. PMLR, 2017.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. *arXiv preprint arXiv:2006.10369*, 2020.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Zheng Li, Zijian Wang, Ming Tan, Ramesh Nallapati, Parminder Bhatia, Andrew Arnold, Bing Xiang, and Dan Roth. Dq-bart: Efficient sequence-to-sequence model via joint distillation and quantization. *arXiv preprint arXiv:2203.11239*, 2022.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, Alexandre Muzio, Saksham Singhal, Hany Hassan Awadalla, Xia Song, and Furu Wei. Deltalm: Encoder-decoder pre-training for language generation and translation by augmenting pretrained multilingual encoders. *arXiv preprint arXiv:2106.13736*, 2021.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*, 2020.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pp. 147–155, 2011.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pp. 1–12, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-3601.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pp. 1–14, 2014.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*, 2018.

Aliakbar Panahi, Seyran Saeedi, and Tom Arodz. Shapeshifter: a parameter-efficient transformer using factorized reshaped matrices. *Advances in Neural Information Processing Systems*, 34, 2021.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Matt Post. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. *arXiv preprint arXiv:2101.00234*, 2021.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5937–5947, 2021.

Sho Takase and Shun Kiyono. Lessons on parameter sharing across layers in transformers. *arXiv preprint arXiv:2104.06022*, 2021.

Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M Rush, David Brooks, et al. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 830–844, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*, 2020.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1901.10430.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. Bert-of-theseus: Compressing bert by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7859–7869, 2020.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 180–189, 2011.

Aston Zhang, Yi Tay, Shuai Zhang, Alvin Chan, Anh Tuan Luu, Siu Cheung Hui, and Jie Fu. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. *arXiv preprint arXiv:2102.08597*, 2021.

Wangchunshu Zhou, Tao Ge, Canwen Xu, Ke Xu, and Furu Wei. Improving sequence-to-sequence pre-training via sequence span rewriting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 571–582, 2021.

A Details of Evaluations for MT and GEC

For MT, we follow the setting of Ott et al. (2018) to train the model on the WMT14 datasets. For En-De, the training set contains 4.5M parallel sentence pairs. We use newstest2013 as our dev set. For En-Fr, there are 36M parallel sentence pairs for training, and we use newstest2012+2013 as the dev set.

In GEC evaluation, we follow previous work (Chen et al., 2020; Zhou et al., 2021) to use the BEA-19 restricted setting, training with Lang-8 (Mizumoto et al., 2011), FCE (Yannakoudakis et al., 2011) and WI+LOCNESS (Bryant et al., 2019). We validate on CoNLL-13 shared task dataset (Ng et al., 2013), and test on CoNLL-14 shared task dataset (Ng et al., 2014). After de-duplicating, we have around 900K sentence pairs for training. Both the dev (CoNLL-13) and test (CoNLL-14) have 1.3K sampales. We train a sentencepiece (Kudo & Richardson, 2018) model of 2K vocabulary for tokenization, and evaluate with the metric of Max-Match (Dahlmeier & Ng, 2012) $F_{0.5}$.

For model training configuration, we show in Table 7; The ablation study into the reduction of d_{decffn} is presented in Table 8.

Configurations	Values
Number of epochs	1000
Devices	8 Nvidia V100 GPU
Max tokens per GPU	20,000
Update Frequency	4
Optimizer	Adam
	(β_1=0.9, β_2=0.99, ϵ=1 $\times$ 10^{-8})
Learning rate	1×10^{-3}
Learning rate scheduler	inverse sqrt
Warmup	4000
Weight decay	0.00001
Loss Function	label smoothed cross entropy (label-smoothing=0.1)
Dropout	[0.1, 0.2] for MT, [0.3, 0.4, 0.5] for GEC

Table 7: Training details for EDGEFORMER for NMT and GEC.

d_{decffn}	#Param	sacreBLEU
2048	46M	26.5
512	37M	26.4
256	35M	26.4
128	34M	26.4

Table 8: The ablation study into the reduction of d_{decffn} on a standard 6+6 Transformer on the dev set.

Configurations	Values
Total updates	250,000
Devices	8 $\times$ 8 Nvidia V100 GPU
Batch size per GPU	128
Sample length	512
Optimizer	Adam
	(β_1=0.9, β_2=0.98 ϵ=1 $\times$ 10^{-6})
Learning rate	5×10^{-4}
Learning rate scheduler	polynomial
clip norm	2.0
Warmup	10,000
Loss Function	cross entropy
Weight decay	0.0
Dropout	0.1

Table 9: Pretraining details for EDGELM.

Configurations	Values
Total updates	100,000
Devices	8 Nvidia V100 GPU
Max tokens per GPU	20,000
Optimizer	Adam
	(β_1=0.9, β_2=0.98 ϵ=1 $\times$ 10^{-6})
Learning rate	Vary for different downstream tasks
Learning rate scheduler	polynomial
clip norm	1.0
Warmup	8,000
Loss Function	cross entropy
Weight decay	0.0
Dropout	0.1

Table 10: Fine-tuning details for EDGELM.

B Configurations of Pretraining and Fine-tuning

We pretrain EDGEFORMER with the same pretrain data as RoBERTa (Liu et al., 2019), through the same pretrain task as T5 (Raffel et al., 2019). The detailed configuration of pretraining is shown in Table 9.

For downstream task fine-tuning, we present the configuration details in Table 10.

C Detailed Evaluation of EdgeLM

In addition to the evaluation results presented in Table 6, we present more detailed results in Table 11 and 12 to show the performance of EDGELM in different sizes and with different vocabularies with factorized embedding parameterization respectively. According to Table 11, EDGELMs consistently outperform the pretrained UTs in the downstream tasks under various model sizes with only 70% computation cost of UTs.

According to Table 12, we show that enlarging the vocabulary size with factorized embedding is effective to improve abstractive summarization and question generation tasks, while it appears to have an adverse effect for GEC. One reason for the performance degradation is that GEC is more sensitive

Model	#Param	FLOPS	CoNLL-14 $F_{0.5}$	Xsum RG-1	Xsum RG-2	Xsum RG-L	QG B4	QG MTR	QG RG-L
Transformer	44M	1.8G	50.1	31.2	10.7	24.9	2.6	9.0	26.0
Pretrained UT ($d = 512$, $d_{\text{ffn}} = 3072$)	9.4M	1.9G	51.1	36.7	14.9	29.7	20.1	22.4	47.1
EDGELM ($d = 512$)	9.4M	1.3G	**52.0**	**37.2**	**15.4**	**30.3**	**20.6**	**23.1**	**47.4**
Pretrained UT ($d = 384$, $d_{\text{ffn}} = 2432$)	5.5M	1.1G	50.1	31.9	11.5	25.7	17.9	21.0	45.9
EDGELM ($d = 384$)	5.5M	0.8G	50.1	**32.3**	**11.7**	**26.0**	**18.9**	**21.2**	**45.9**
Pretrained UT ($d = 768$, $d_{\text{ffn}} = 4608$)	21.2M	4.2G	52.8	37.3	15.6	30.4	20.9	23.5	47.5
EDGELM ($d = 768$)	21.4M	2.9G	**53.1**	**37.9**	**15.9**	**30.8**	**21.0**	**23.6**	**47.8**

Table 11: The comparison between pretrained Universal Transformer (UT) and EdgeLM with Adapter-based layer adaptation ($r = d/8$). We enlarge UT's d_{ffn} to let its model size comparable with its EdgeFormer counterpart with the same d for fair comparison given the same parameter budget though introducing additional cost.

Vocab	d_{embed}	**#Param (including embedding)**	**CoNLL-14** $F_{0.5}$	**Xsum** RG-1	**Xsum** RG-2	**Xsum** RG-L	**QG** B4	**QG** MTR	**QG** RG-L
spm2k	512	10.4M	**52.7**	36.3	14.8	29.5	19.0	21.7	46.3
spm8k	128	10.5M	52.0	**37.2**	**15.4**	**30.3**	**20.6**	**23.1**	**47.4**
spm16k	64	10.5M	51.6	37.1	15.1	30.0	19.6	22.3	46.7

Table 12: EDGELM ($d = 512$) with Adapter-LA ($r = 32$) using different vocabulary and d_{embed}. Except the model with spm2k whose $d_{\text{embed}} = d$, the models with spm8k and spm16k use factorized embedding to prevent increasing the total parameters.

to morphological and syntactic information of a token; when d_{embed} becomes small with factorized embedding, it may not accurately capture the morphological and syntactic information of the token.