Efficient Transformer for Mobile Applications

Anonymous authors
Paper under double-blind review

Abstract

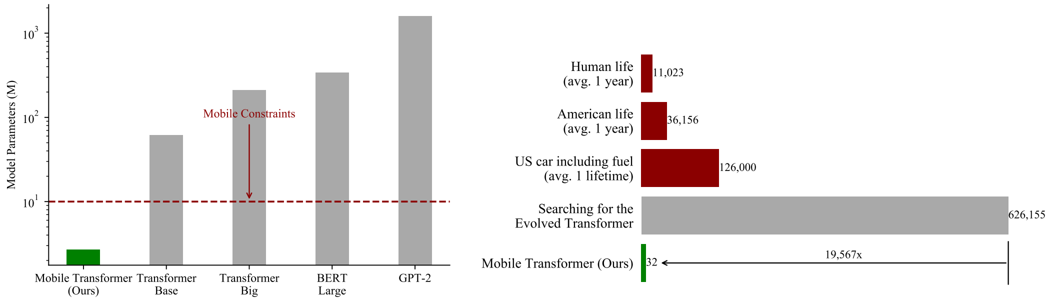
Transformer has become ubiquitous in natural language processing (*e.g.*, machine translation, question answering); however, it requires enormous amount of computations to achieve high performance, which makes it not suitable for real-world mobile applications since mobile phones are tightly constrained by the hardware resources and battery. In this paper, we investigate the mobile setting (under 500M Mult-Adds) for NLP tasks to facilitate the deployment on the edge devices. We present *Long-Short Range Attention* (LSRA), where some heads specialize in the local context modeling (by convolution) while the others capture the long-distance relationship (by attention). Based on this primitive, we design *Mobile Transformer* (MBT) that is tailored for the mobile NLP application. Our MBT demonstrates consistent improvement over the transformer on two well-established language tasks: IWSLT 2014 German-English and WMT 2014 English-German. It outperforms the transformer by 0.9 BLEU under 500M Mult-Adds and 1.1 BLEU under 100M Mult-Adds on WMT'14 English-German. Without the costly architecture search that requires more than 250 GPU **years**, our manually-designed MBT achieves 0.4 higher BLEU than the AutoML-based Evolved Transformer under the extremely efficient mobile setting (*i.e.*, 100M Mult-Adds).

1 Introduction

Transformer (Vaswani et al., 2017) has prevailed among various areas of natural language processing due to its high training efficiency and superior capability in capturing long-distance dependencies. Building on top of them, modern state-of-the-art models, such as BERT (Devlin et al., 2019), are able to learn powerful language representations from unlabeled text and even surpass the human performance on the challenging question answering task.

However, the good performance come at high computational cost. For example, a single transformer deep model executes more than 10G Mult-Adds in order to translate a sentence of only 30 words. Such extremely high computational resources requirement is beyond the capabilities of many edge devices such as smartphones and IoTs. Therefore, it is of great importance to design efficient and fast transformer architecture specialized for real-time NLP applications on the edge. Automatic neural architecture search (Zoph & Le, 2017; So et al., 2019) is a choice for high accuracy model design, but the massive searching cost raises much concern (Figure 1b).

In this paper, we mainly focus on the efficient inference scenario for mobile devices where the total number of Mult-Adds is constrained to be lower than 500M. A straightforward way to reduce the computation of transformer is to directly shrink the embedding size. Although it can effectively reduce both model size and computation, it also weakens the model capaciy capturing the long and short distance relationship at the same time. To this end, we systematically studied the computation breakdown of the transformer and observe that the computation is dominated by the feed forward network (FFN), not the attention. We discovered that the prevailing bottleneck-structured transformer block is not efficient. We then present a novel Long-Short Range Attention (LSRA) primitive. LSRA trades off the computation in FFN for wider attention layers. It stretches the bottleneck to introduce more dependency capturing capability for the attention layer, and then shrink the embedding size to reduce the total computation amount while maintaining the same performance. Instead of having one module for "general" information, LSRA dedicates *specialized* heads to model long and short distance contexts. Inspired by Wu et al. (2019b), LSRA introduces convolution in a parallel branch to



(a) Model sizes of modern NLP models.

(b) The design cost measured in pounds of CO_2 emission.

Figure 1: Left: the size of recent NLP models grows rapidly and exceeds the mobile constraints by a large extend. Right: the search cost of AutoML-based NLP models is considerable, which emits carbon dioxide nearly five times the lifetime emissions of the average car.

extract local dependencies so that the attention branch can focus on long-distance context modeling. By stacking this primitive, we build Mobile Transformer (MBT) for mobile NLP applications.

Extensive experiments demonstrate that our MBT model offers significant improvement over the transformer on machine translation tasks. On IWSLT 2014 German-English dataset, it outperforms the transformer by 1.5 BLEU under 100M Mult-Adds, while on WMT 2014 English-German dataset, MBT surpasses the transformer by 0.9 BLEU under 500M Mult-Adds and 1.1 BLEU under 100M Mult-Adds. Guided by our design insights, our manually-designed MBT also achieves 0.4 higher BLEU than the AutoML-based Evolved Transformer (So et al., 2019) which requires more than 250 GPU years to search, emitting as much carbon as five cars in their lifetimes (see Figure 1b). It indicates that AutoML is not panacea: careful analysis and design insights (*i.e.*, removing bottleneck, specialized heads) can effectively prune the search space and improve the sample efficiency.

The contribution of this paper has four aspects:

1. We systematically analyze the commonly used computation bottleneck structure in modern neural networks and argue that the bottleneck design is not optimal for 1-D attention.
2. We propose a novel specialized multi-branch feature extractor, Long-Short Range Attention (LSRA), as the basic building block of our transformer, where convolution helps capture the local context and attention concentrates on global context feature.
3. We build Mobile Transformer (MBT) based on LSRA. Under mobile computation resource constraints (500M Mult-Adds), MBT demonstrates coherent improvement on two widely used machine translation datasets.
4. Even compared to AutoML-searched Evolved Transformer, MBT offers 0.4 more BLEU score on WMT En-De dataset, saving the design cost by 20000× in CO_2 emission. It alerts us to rethink the practicality of AutoML in terms of design cost.

2 Related Work

RNNs and CNNs. Recurrent neural networks (RNNs) have prevailed various sequence modeling tasks for a long time (Sutskever et al., 2014; Luong et al., 2015; Bahdanau et al., 2015; Wu et al., 2016). However, RNNs cannot be run parallelly across the sequence due to its temporal dependency. Recently, some work has demonstrated that RNN is not an essential component to achieve the state-of-the-art performance. For instance, researchers have proposed highly-efficient convolution-based models (Kalchbrenner et al., 2016; Gehring et al., 2017; Kaiser et al., 2018; Wu et al., 2019b). Convolution is an ideal primitive to model the local context information; however, it lacks the ability to capture the long-distance relationship, which is critical in many sequence modeling tasks.

Transformers. As an alternative, attention is able to capture global-context information by pairwise correlation. Transformer (Vaswani et al., 2017) has demonstrated that it is possible to stack the self-attentions to achieve the state-of-the-art performance. Recently, there have been a lot of variants to the transformer (Ahmed et al., 2017; Ott et al., 2018; Chen et al., 2018; Paulus et al., 2018; Shaw et al., 2018). Among them, Ott et al. (2018) proposed to scale up the batch size; Shaw et al. (2018) leverages the relative position representations; Ahmed et al. (2017) introduces the weighted multi-head attention. All these attempts are orthogonal to our work, as we focus on improving the neural network architecture itself, and not the general techniques used for improving overall performance.

Automated Model Design. Due to the vast architecture design space, there has been an increasing trend of automating the compact model design using the neural architecture search (NAS) (Zoph & Le, 2017; Zoph et al., 2018; Pham et al., 2018) and integrating the hardware resource constraints into the optimization loop, such as MnasNet (Tan et al., 2018), ProxylessNAS (Cai et al., 2019) and FBNet (Wu et al., 2019a). In the NLP community, the evolved transformer (So et al., 2019) adopts the neural architecture search (Zoph & Le, 2017) for basic blocks and finds a better #parameter-BLEU trade-off for the transformer. However, all these AutoML-based model design requires significant amount of GPU hours to find the 'best' model, which is not affordable for most researchers.

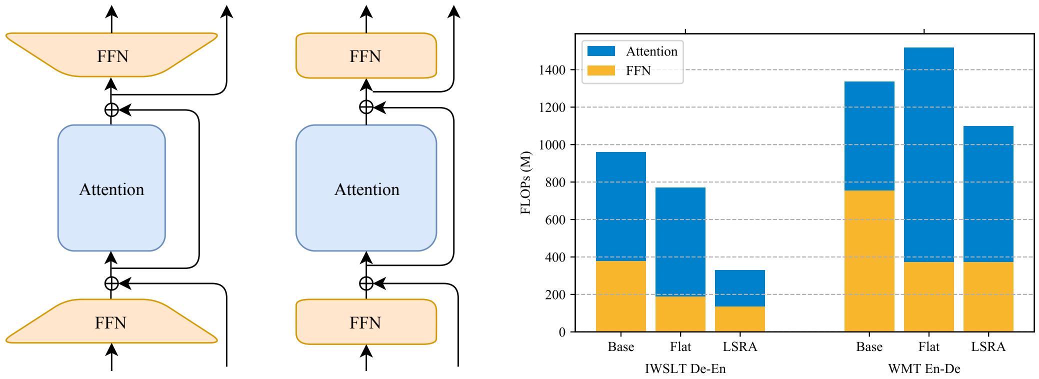
Model Acceleration. Apart from designing efficient models directly, another approach to achieve the efficient inference is to compress and accelerate the existing large models. For instance, some have proposed to prune the separate neurons (Han et al., 2015; 2016) or the entire channels (He et al., 2017; Liu et al., 2017; He et al., 2018); others have proposed to quantize the network (Courbariaux et al., 2016; Zhu et al., 2017; Krishnamoorthi, 2018; Wang et al., 2019) to accelerate the model inference. Recently, AutoML has also been used to automate the model compression and acceleration (Yang et al., 2018; He et al., 2018; Wang et al., 2019; Liu et al., 2019). All these techniques are effective for arbitrary models and are therefore orthogonal to our approach, while we aim to explore how to make use to the domain knowledge of natural language processing.

3 Is Bottleneck Effective for 1-D Attention?

Attention mechanism has been widely used in various applications, including 1-D (language processing (Vaswani et al., 2017)), 2-D (image recognition) and 3-D (video recognition (Wang et al., 2018)). It computes pairwise dot-product between all the input elements to model both short-term and long-term relationships. Despite its effectiveness, the operation introduces massive computation. Assume the number of elements (*e.g.*, length of tokens in language processing, number of pixels in image, *etc.*) fed to attention layer is N, and the dimension of features (*i.e.*, channels) is d, the computation needed for the dot-product is N^2d. For images and videos, N is usually very large. For example, the intermediate feature map in a video network (Wang et al., 2018) has 16 frames, each with 112×112 resolution, leading to $N = 2 \times 10^5$. The computation of convolution and fully-connected layers grows linearly w.r.t. N, while the computation of attention layers grows quadratically w.r.t. N. The computation of attention module will soon overwhelm with a large N.

To address the dilemma, a common practice is to first reduce the number of channels d using a linear projection layer before applying attention, and increase the dimension afterwards (as shown in Figure 2a). In the original design of transformer (Vaswani et al., 2017), the channel dimension in the attention module is 4 times smaller than that in the FFN layer. Similarly, in non-local video network (Wang et al., 2018), the channel number is first reduced by half before applying non-local attention module. This practice saves the computation by 16 or 4 times. Nevertheless, it also *decreases* the feature modeling ability of attention layers with a smaller feature dimension. The situation could be even worse for language processing, as attention is the major module for feature modeling (unlike images and videos where convolutions conduct the major modeling).

For tasks like translation, the length of the input sequence N tends to be small, which is around 20-30 in common cases. A transformer block consists of an attention (or two for decoder), followed by a feed-forward network (FFN). For the attention layer, the FLOPs would be $O(4Nd^2 + N^2d)$; for FFN, the FLOPs is $O(2 \times 4Nd^2)$. Given a small N, it is doubtful if the bottleneck design is a good trade-off between computation and accuracy on 1D attention. To verify the idea, we first profile the computation makeup in the transformer in Figure 2b. Surprisingly, for the original transformer (denoted as 'base' in the figure), FFN layer actually consumes much of the computation especially on WMT En-De task. This is not desirable since FFN itself cannot perform any context modeling. In conclusion, due to the small N, bottleneck design cannot significantly reduce the computation in 1D attention, while the limited benefit is further compromised by the FFN layer; it also harms the



(a) Vanilla and flattened transformer block.

(b) Mult-Adds breakdown for attention and FFN.

Figure 2: Flattening the bottleneck of transformer blocks increases the proportion of the attention versus the FFN, which is good for further optimization for attention.

	IWSLT De-En			WMT En-De			
	Embedding	Mult-Adds	BLEU	Embedding	Mult-Adds	Attention	BLEU
Vaswani et al. (2017)	512-1024	959M	34.4	512-2048	1.3G	44%	27.3
Our Reimplementation	512-1024	959M	34.5	512-2048	1.3G	44%	27.7
Transformer (*Flat*)	512-512	460M	34.5	720-720	1.5G	75%	27.8

Table 1: Bottleneck design is not optimal for 1-D attention. A 'flat' version of transformer block (slightly increase the embedding size when necessary) does not reduce the BLEU score, but makes the attention part take the major computation for further optimization.

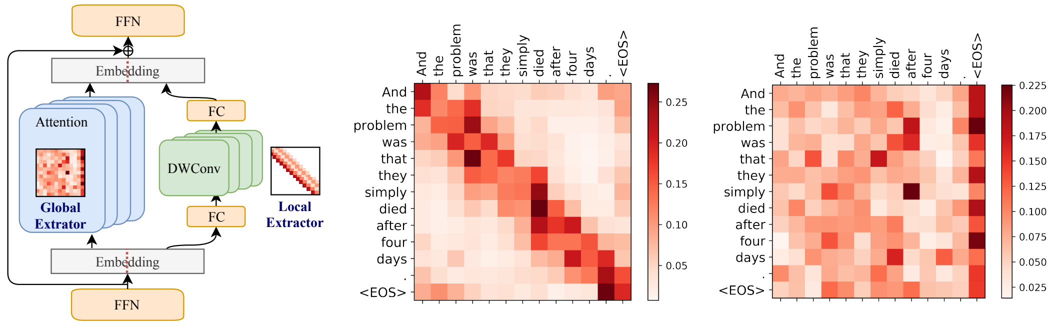
capacity of attention layer due to smaller dimension, which is the major context modeling unit in the transformer.

Therefore, we argue that the bottleneck design is not optimal for 1-D attention. We instead design a 'flat' version of transform block that does not reduce and increase the channel dimension. With the new design, the attention part now takes up the major computation in the flat transformer model ('Flat' in Figure 2b), leaving a larger space for further optimization. We also test the performance change of such modification on IWSLT and WMT datasets (Table 1). On IWSLT De-En task, our Flatten Transformer can achieve similar BLEU score using less than half computation compared to the original transformer; on WMT, we can achieve comparable performance at a slightly larger computation, which can be easily reduced with further optimization on the attention heads.

4 Long-Short Range Attention (LSRA)

Since the attention-based network made a great progress in NLP, researchers have tried to understand what the exact information an attention captures. Kovaleva et al. (2019) and Clark et al. (2020) visualized the attention weights from different layers in BERT. As shown in Figure 3b, the weight w illustrate the relationships between the words from the source sentence and the target sentence (the same for self-attention). With a greater weight w_{ij} (darker color), the i-th word in the source has more attention focus on the j-th word in the target. And the focus of an attention module typically has some strong patterns: sparse points, vertical lines and diagonal groups. The three of them respectively represent the relationships between some particular words, the long term information and the correlation in small neighborhoods. We denotes the former two as "global" relationships and the last one as "local".

In order to provide both the global and local information for a translation task, the attention modules have to extract all of those features with exactly the same architecture, requiring a large capacity. That is not an optimal case since, taking hardware design as an example, devices for general purpose,



(a) Mobile Transformer block

(b) Self-Attention

(c) Long-Short Range Attention

Figure 3: Model architecture (a) and the visualization of attention weights. Conventional attention (b) put too much emphasizes on short-range relationship modeling (diagonal structure). We filtered the short-range relationship modeling by a convolutional branch which efficiently models locality, so that the attention branch can evenly focus on short-range and long-range modeling (c).

like CPUs, tend to be more redundant than specialized ones, like FPGAs. When the model capacity is relatively large, the redundancy can be tolerated and may even provides a better performance. However, when it comes to mobile applications, a model should be more efficient due to the computation and power constraints. To tackle the problem, instead of having one module for "general" information, we propose a more specialized architecture, *Long-Short Range Attention (LSRA)*, that captures the global and local information separately.

As shown in Figure 3a, our LSRA module follows a two-branch design. The left branch captures global context, while the right branch models local information. Instead of feeding the whole input to both branch, we split it into two parts along the channel dimension, which will be mixed by the following FFN layer. Such a practice reduces the overall computation by 2 times. The upper branch is a normal attention model as in (Vaswani et al., 2017), while the channel dimension is reduced by half. For the right branch of local relationships, one natural idea is to apply convolution over sequence. With a sliding window, the diagonal groups can be easily covered by the module. To further reduce the computation, we replace the normal convolution with a lighter version Wu et al. (2019b) consisting of linear layers and depth-wise convolution. In this manner, we place the attention and the convolutional module side by side, encouraging them to have different perspective of the sentence, globally and locally, so that the architecture can then benefit from the specialization and achieve a better efficiency.

To have a better insight, we visualized the attention weights for a fully trained basic Transformer and our Mobile Transformer Figure 3. It can be easily distinguished that instead of attempting to model both the global and local feature, the attention module in Long-Short Range Attention only focus on the global, leaving the local context modeling to the convolution branch.

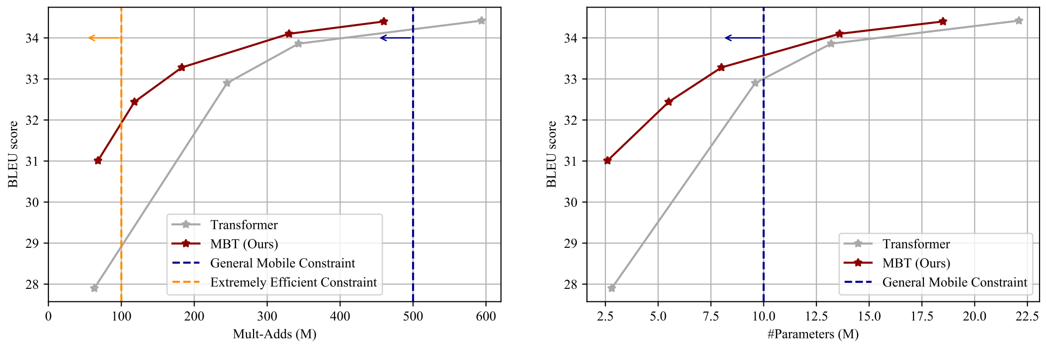
5 Experimental Setup

Mobile Settings. Most of machine translation architectures benefit from the large model size and computational complexity. For example, the transformer architecture gains 1.1 BLEU score improvement on WMT'14 En-De datasets from the base to the big model and 3.7 on WMT'14 En-Fr (Vaswani et al., 2017). Empirically, however, edge devices, such as mobile phones and IoTs, is highly computationally limited. Those massive architectures are no more suitable for real-world mobile applications. Therefore, comparing to the typical mobile constraints in computer vision (Sandler et al., 2018), we propose mobile settings for machine translation tasks: a general mobile device the constraints should be under 500M Mult-Adds (or 1G FLOPs) and for an extremely efficient setting the computation should be less than 100M Mult-Adds (or 200M FLOPs). And the general limit for the number of parameters is set to be 10M.

Datasets. The results are based on two machine translation benchmarks: For IWSLT'14 German-English (De-En), we follow the settings in Grave et al. (2017) with 160K training sentence pairs and 10K joint BPE vocabulary in lower case. For WMT English to German (En-De) we train the

	#Parameters	Mult-Adds	BLEU	ΔBLEU
Transformer (Vaswani et al., 2017)	2.8M	63M	27.9	–
LightConv (Wu et al., 2019b)	2.5M	52M	28.5	+0.6
Mobile Transformer (Ours)	2.6M	48M	**29.4**	**+1.5**
Transformer (Vaswani et al., 2017)	5.7M	139M	31.4	–
LightConv (Wu et al., 2019b)	5.1M	115M	31.6	+0.2
Mobile Transformer (Ours)	5.5M	118M	**32.4**	**+1.0**
Transformer (Vaswani et al., 2017)	9.6M	245M	32.9	–
LightConv (Wu et al., 2019b)	8.4M	204M	32.9	+0.0
Mobile Transformer (Ours)	9.1M	183M	**33.3**	**+0.4**

Table 2: Quantitative results on IWSLT'14 De-En. Our MBT outperforms the transformer (Vaswani et al., 2017) as well as the Lightweight convolution network (Wu et al., 2019b) especially in mobile settings.



(a) BLEU score vs. model Mult-Adds

(b) BLEU score vs. #Parameters

Figure 4: Trade-off curve on IWSLT'14 De-En. Both curves show that our Mobile Transformer achieves better performance on mobile settings. With tighter constraints, the improvement is more significant.

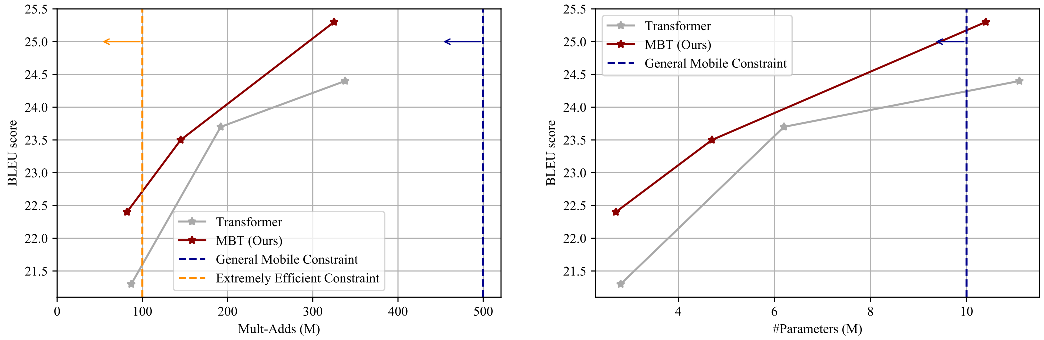
model on WMT'16 training data with 4.5M sentence pairs and validate on newstest2013 and test on newstest2014, the same as Wu et al. (2019b). Moreover, the vocabulary used is a 32K joint source and target byte pair encoding (Sennrich et al., 2016).

Architecture. The model architecture is based on the sequence to sequence learning encoder-decoder (Sutskever et al., 2014). Our baseline model is based on the one proposed by Vaswani et al. (2017) for WMT. And for IWSLT we follow the settings in Wu et al. (2019b). We adopt fairseq's re-implementation (Ott et al., 2019) of transformer base model as the backbone. Transformer blocks form both the encoder and the decoder. In the encoder, the transformer block consists of a self-attention module followed by a feed-forward network (FFN) module. The block in the decoder is the same, except for an additional encoder-decoder attention module in the middle of the self-attention and FFN. Each module is surrounded by the residual connection (He et al., 2016) and followed by a layer normalization.

In our architecture, we first flatten the bottleneck from the transformer base model and then replace the self-attention with the LSRA. More specifically, the LSRA module for IWSLT contains four specialized modules, two for attention and two for depth-wise convolution (to get a higher compression ratio); for WMT, we use two specialized modules, an attention and a depth-wise convolution. Both the input and the output of the convolution are transformed by fully connected layers and the kernel is dynamically calculated from the input using a fully connected layer. The kernel sizes are [3, 5, 7, 31×3] for both the encoder and the decoder (Wu et al., 2019b), and the number of heads are tuned for each module from [2, 4, 8].

	#Parameters	Mult-Adds	BLEU	ΔBLEU
Transformer (Vaswani et al., 2017)	2.8M	87M	21.3	–
Mobile Transformer (Ours)	2.7M	82M	**22.4**	**+1.1**
Transformer (Vaswani et al., 2017)	11.1M	338M	24.4	–
Mobile Transformer (Ours)	10.4M	325M	**25.3**	**+0.9**
Transformer (Vaswani et al., 2017)	17.3M	527M	26.1	–
Mobile Transformer (Ours)	18.4M	561M	**26.3**	**+0.2**

Table 3: Quantitative results on WMT'14 En-De. Our MBT improves the BLEU score with comparable Mult-Adds with the Transformer.



(a) BLEU score vs. model Mult-Adds

(b) BLEU score vs. #Parameters

Figure 5: Trade-off curve on WMT'14 En-De dataset. Both curves illustrate that our Mobile Transformer outperform the basic transformer on mobile settings.

Training Settings. All of our training settings are in line with Wu et al. (2019b). We use a dropout of 0.3 for WMT En-De and IWSLT De-En and linearly scale down the dropout ratio when shrinking the dimension of the embeddings. Same as Wu et al. (2019b), we apply Adam optimizer and a cosine learning rate schedule (Kingma & Ba, 2015; Loshchilov & Hutter, 2017) for the WMT models, where the learning rate is first linearly warm up from 10^{-7} to 10^{-3} and then annealed following a cosine rate with a single cycle. For IWSLT De-En, we use inverse square root learning rate scheduling (Vaswani et al., 2017) with linear warm-up.

We train WMT models on 8 NVIDIA RTX 2080Ti GPUs for a total of 50K steps. For IWSLT De-En, we train for 50K steps on a single GPU. Mixed-precision training with floating point 16 is used for WMT models, when applicable. We also accumulate the gradients for 16 batches before each model update (Ott et al., 2018). The gradients of IWSLT models are not accumulated. Particularly, the maximum number of tokens in a batch is 4K for all the models. Label smooth of 0.1 is applied for the prior distribution over the vocabulary (Szegedy et al., 2016; Pereyra et al., 2017).

Evaluation. For evaluation, we use the same beam decoding configuration used by Vaswani et al. (2017), where there is a beam size of 4 and a length penalty of 0.6. All BLEUs are calculated with case-sensitive tokenization[*], but for WMT En-De, we also use the compound splitting BLEU[†], the same as Vaswani et al. (2017). When testing, we average the last 10 model checkpoints for IWSLT'14 De-En and take the model with the lowest perplexity on the validation set for the WMT'14 En-De. We omit the word embedding lookup table from the model parameters since the number of entries in the table would highly differ for various tasks using transformer. For the Mult-Adds, we calculate the total number of multiplication-addition pairs for a model translating a sequence with the length of 30 to a sequence with the same length, which is the common length for sentence-level machine translation tasks.

[*] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

[†] https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/get_ende_bleu.sh

	#Parameters	Mult-Adds	BLEU	GPU Hours	CO_2e (lbs)	Cloud Computation Cost
Transformer	2.8M	87M	21.3	8×12	26	\$68 - \$227
ET (So et al., 2019)	3.0M	94M	22.0	8×274K	626K	\$1.6M - \$5.5M
Mobile Transformer	2.7M	82M	**22.4**	8×14	32	\$83 - \$278

Table 4: Performance and training cost of an NMT model in terms of CO_2 emissions (lbs) and cloud compute cost (USD). The training cost estimation is adapted from Strubell et al. (2019). The training time for transformer and our Mobile Transformer is measured on NVIDIA V100 GPU. The cloud computation price is based on the AWS.

6 Results

6.1 Results on IWSLT

We first report the result on IWSLT'14 De-En dataset. The baseline model is the same as the one used in Wu et al. (2019b), which provides the best results in the literature with 512 embedding dimension, 1024 FFN hidden dimension and 4 heads for all the attentions. Figure 4 illustrate the trade-off of BLEU score and model computation as well as model size. Our Mobile Transformer generally outperforms the base transformer on mobile constraints. With tighter computation limitation, our model achieves more significant improvement. That is because, when the dimension of the embedding decreases, it becomes much harder for the "general" attention to extract both the global and local feature from the rather more compact information within the embedding. On the contrary, with the specialized LSRA, our model can capture the information from the embedding more effectively.

In Table 2, we present the quantitative results of our Mobile Transformer on IWSLT'14 De-En dataset, comparing to the transformer baseline as well as the LightConv (Wu et al., 2019b). In the extremely efficient settings, our model even achieves 1.5 BLEU score improvement than the transformer.

6.2 Results on WMT

We also show the result on the WMT'14 En-De dataset. Similar to the IWSLT, our Mobile Transformer achieves a better trade-off with regard to transformer (Vaswani et al., 2017) against the total computation and the number of model parameters Figure 5 on mobile settings. The quantitative results in Table 3 indicates that our specialized MBT has 1.1 BLEU score improvement under the extremely efficient settings with 100M Mult-Adds and 0.9 under 300M Mult-Adds.

6.3 Comparison with Automated Design

Comparing with the AutoML-based Evolved Transformer (ET) (So et al., 2019), our Mobile Transformer also shows a significant improvement in mobile settings. Moreover, within the extremely efficient mobile constraints, the MBT outperforms the ET by 0.4 BLEU scores, as shown in Table 4. Our architecture design is different from ET's design: ET stacks attentions and convolutions sequentially, while our MBT puts them in a parallel manner; also, ET does not flatten the FFN.

Though nowadays, neural architecture search has been proved to be very powerful for searching in a large design space, the huge cost, more than 626155 lbs CO_2 emissions and more than 250 GPU years, cannot be ignored. Instead, careful human design with intuitions for specific tasks can also be a great choice in practice to save a large number of resources for the earth.

7 Conclusion

In this paper, we presented *Long-Short Range Attention* (LSRA), where some heads focus on the local context modeling while the others capture the long-distance relationship. Based on this primitive, we design *Mobile Transformer* (MBT) that is specialized for the mobile setting (under 500M Mult-Adds) to facilitate the deployment on the edge devices. Our MBT demonstrates consistent improvement over the transformer on multiple datasets. It also surpasses the Evolved Transformer that requires costly architecture search under the mobile setting.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted Transformer Network for Machine Translation. *arXiv*, 2017. 3

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*, 2015. 2

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *ICLR*, 2019. 3

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *ACL*, 2018. 3

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What Does BERT Look At? An Analysis of BERT's Attention. In *BlackboxNLP*, 2020. 4

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv*, 2016. 3

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019. 1

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. Convolutional Sequence to Sequence Learning. In *ICML*, 2017. 2

Edouard Grave, Armand Joulin, Moustapha Cissé, Hervé Jégou, et al. Efficient softmax approximation for GPUs. In *ICML*, 2017. 5

Song Han, Jeff Pool, John Tran, and William Dally. Learning both Weights and Connections for Efficient Neural Networks. In *NIPS*, 2015. 3

Song Han, Huizi Mao, and William Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *ICLR*, 2016. 3

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

Yihui He, Xiangyu Zhang, and Jian Sun. Channel Pruning for Accelerating Very Deep Neural Networks. In *ICCV*, 2017. 3

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *ECCV*, 2018. 3

Lukasz Kaiser, Aidan N Gomez, and François Chollet. Depthwise Separable Convolutions for Neural Machine Translation. In *ICLR*, 2018. 2

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural Machine Translation in Linear Time. *arXiv*, 2016. 2

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 7

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the Dark Secrets of BERT. In *EMNLP*, 2019. 4

Raghuraman Krishnamoorthi. Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper. *arXiv*, 2018. 3

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning. *arXiv*, 2019. 3

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning Efficient Convolutional Networks through Network Slimming. In *ICCV*, 2017. 3

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*, 2017. 7

Minh-Thang Luong, Hieu Pham, and Christopher Manning. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*, 2015. 2

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling Neural Machine Translation. In *WMT*, 2018. 3, 7

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *NAACL Demo*, 2019. 6

Romain Paulus, Caiming Xiong, and Richard Socher. A Deep Reinforced Model for Abstractive Summarization. In *ICLR*, 2018. 3

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR Workshop*, 2017. 7

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient Neural Architecture Search via Parameter Sharing. In *ICML*, 2018. 3

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*, 2018. 5

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *ACL*, 2016. 6

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-Attention with Relative Position Representations. In *NAACL*, 2018. 3

David So, Quoc Le, and Chen Liang. The Evolved Transformer. In *ICML*, 2019. 1, 2, 3, 8

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP. In *ACL*, 2019. 8

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*, 2014. 2, 6

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 7

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *arXiv*, 2018. 3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *NeurIPS*, 2017. 1, 2, 3, 4, 5, 6, 7, 8

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-Aware Automated Quantization with Mixed Precision. In *CVPR*, 2019. 3

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local Neural Networks. In *CVPR*, 2018. 3

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In *CVPR*, 2019a. 3

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay Less Attention with Lightweight and Dynamic Convolutions. In *ICLR*, 2019b. 1, 2, 5, 6, 7, 8

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. 2016. 2

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications. In *ECCV*, 2018. 3

Chenzhuo Zhu, Song Han, Huizi Mao, and William Dally. Trained Ternary Quantization. In *ICLR*, 2017. 3

Barret Zoph and Quoc V Le. Neural Architecture Search with Reinforcement Learning. In *ICLR*, 2017. 1, 3

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. In *CVPR*, 2018. 3