PartialFormer: Modeling Part Instead of Whole for Machine Translation

Tong Zheng[1*], **Bei Li**[1*], **Huiwen Bao**[1,2*], **Jiale Wang**[1], **Weiqiao Shan**[1], **Tong Xiao**[1,2†] **and Jingbo Zhu**[1,2]

[1]School of Computer Science and Engineering, Northeastern University, Shenyang, China
[2]NiuTrans Research, Shenyang, China
{zhengtong12356, goodbaohuiwen}@gmail.com, libei_neu@outlook.com
{xiaotong,zhujingbo}@mail.neu.edu.cn

Abstract

The design choices in Transformer feed-forward neural networks have resulted in significant computational and parameter overhead. In this work, we emphasize the importance of hidden dimensions in designing lightweight FFNs, a factor often overlooked in previous architectures. Guided by this principle, we introduce PartialFormer, a parameter-efficient Transformer architecture utilizing multiple smaller FFNs to reduce parameters and computation while maintaining essential hidden dimensions. These smaller FFNs are integrated into a multi-head attention mechanism for effective collaboration. We also propose a tailored head scaling strategy to enhance PartialFormer's capabilities. Furthermore, we present a residual-like attention calculation to improve depth scaling within PartialFormer. Extensive experiments on 9 translation tasks and 1 abstractive summarization task validate the effectiveness of our PartialFormer approach on machine translation and summarization tasks. Our code would be available at: https://github.com/zhengkid/PartialFormer.

1 Introduction

The Transformer model (Vaswani et al., 2017) has emerged as a cornerstone in the natural language processing (NLP) domain, overshadowing convolutional neural networks (Gehring et al., 2017) and recurrent neural networks (Sutskever et al., 2014) by virtue of its minimal inductive bias, superior scalability, and proficiency in modeling sequences. Nonetheless, its substantial computational and parametric requisites pose significant challenges to its deployment and training, warranting an ongoing trend in the research community toward eliminating redundant parameters and computations (Dehghani et al., 2019; Mehta et al., 2019; Lan et al., 2020; Wu et al., 2020; Mehta et al., 2021; Reid et al., 2021; Li et al., 2022a) in Transformer.

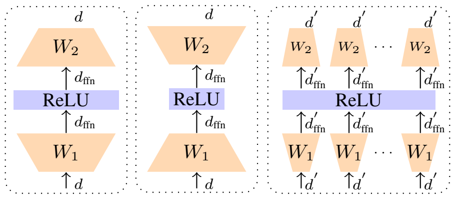


Figure 1: Illustration of our idea.

While these attempts represent significant strides in enhancing the efficiency of the Transformer architecture, they largely neglect an equally critical component: the Feed-Forward Network (FFN) that constitutes a substantial part of the Transformer's computational and parametric footprint, due to the inherent large feature space and hidden dimension. Previous studies (Mehta et al., 2021; Wu et al., 2020; Ge et al., 2022) have simplified FFNs by naively reducing their hidden dimensions, often at the expense of expressive power. This leads to a question: *Is the current formulation of lightweight FFNs truly optimal?*

To answer this concern, we turn to the insights provided by Geva et al. (2021), who depicted FFNs as a collection of key-value memories, where the number of memories is equal to the number of hidden dimensions in FFNs. This finding underscores the significance of hidden dimension in FFNs. Drawing inspiration from this finding and the successful application of large hidden sizes in FFNs as evidenced by Meta's 4B model (Tran et al., 2021)[1], we hypothesize that an efficient lightweight FFN is not merely about parameter reduction. Rather, it should aim to maintain or even increase the hidden dimension while judiciously reducing the number of parameters involved.

The literature on animal cognition provides some

* Equal Contribution.
† Corresponding author.

[1]They have shown enlarging the hidden size of FFNs to 16384 delivers significant BLEU improvements.

clues for designing lightweight and expressive FFNs. Research on animals' behavior has shown that group animals such as insects, fish, and some birds can emerge with some incredible abilities to deal with some complex tasks, though each individual owns poor abilities (Couzin, 2009; Conradt and Roper, 2005). This concept resonates with the AI community's "Swarm Intelligence" paradigm (Bonabeau et al., 1999), which emphasizes the power of collective decision-making. This biological prior motivates us to integrate Swarm Intelligence principles into the FFN design process.

To this end, we propose PartialFormer, an innovative approach to Transformer architecture. At the heart of PartialFormer lies the novel concept of Partial-Level Gated Feed-Forward Networks (PG-FFN). Conceived as an ensemble of streamlined FFNs operating in concert, each PG-FFN produces lower-dimensional hidden features. Despite their reduced individual dimensions, the aggregated output of these PG-FFNs either matches or surpasses the hidden dimensions of traditional, larger FFNs, as empirically substantiated in Figure 1. Moreover, we further equipped PartialFormer with a head scaling strategy tailed for efficiently scaling, and a residual-like attention calculation for stable optimization. These techniques empower PartialFormer to efficiently utilize parameters within the same parameter budget.

Our main contributions are as follows:

- We introduced PG-FFNs, a method that efficiently reduces parameters and computations, and integrated them into the PartialFormer architecture for high performance. Additionally, we introduced an attention calculation method for stable optimization.
- We investigated the scalability of PartialFormer and proposed a head scaling strategy tailored for PartialFormer to efficient scaling.
- Rigorous empirical tests across 9 machine translation tasks and 1 abstractive summarization task confirm the effectiveness and efficiency of PartialFormer on machine translation and summarization tasks.

2 Preliminary: Transformer

In this section, we present some prior knowledge about the Transformer. The Transformer block consists of a multi-head self-attention and a feed-forward network. Let $X \in \mathbb{R}^{T\times d}$ be a $T \times d$ input matrix of T tokens. Each multi-head self-attention component owns H heads. For simplicity, we omit layer normalization and residual connections.

Multi-Head Self-Attention MHSA aims to model the global dependency among tokens. MHSA computes as follows:

$$\begin{aligned} A_i &= \mathrm{Softmax}(\frac{Q_i(K_i)^\mathsf{T}}{\sqrt{d_k}}), && (1)\\ \mathrm{head}_i &= A_iV_i, && (2)\\ X &= \sum_{i=1}^{H}\mathrm{head}_iW_i^O, && (3)\end{aligned}$$

where Q_i, K_i, V_i denote the query, key and value of i-th head, which are derived from input with three learnable matrices $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d_k}$ as follows: $Q_i = XW_i^Q, K_i = XW_i^K, V_i = XW_i^V$, respectively. $W_i^O \in \mathbb{R}^{d_k\times d}$ is a learnable matrix. d_k and d denote the head dimension and embedding dimension, respectively. A_i and head_i denote the attention matrix and representation of i-th head, respectively.

Feed-Forward Network Feed-forward network is responsible for improving the expressiveness of the whole representation space by adopting an "expansion-activation-reduction" mapping strategy. It computes as follows:

$$X = \mathrm{ReLU}(XW_1 + b_1)W_2 + b_2, \qquad (4)$$

where $W_1 \in \mathbb{R}^{d\times d_{\mathrm{ffn}}}, W_2 \in \mathbb{R}^{d_{\mathrm{ffn}}\times d}, b_1 \in \mathbb{R}^{d_{\mathrm{ffn}}}, b_2 \in \mathbb{R}^d$ are learnable matrices and d_{ffn} denotes the hidden dimension in FFN that is usually set to $4d$.

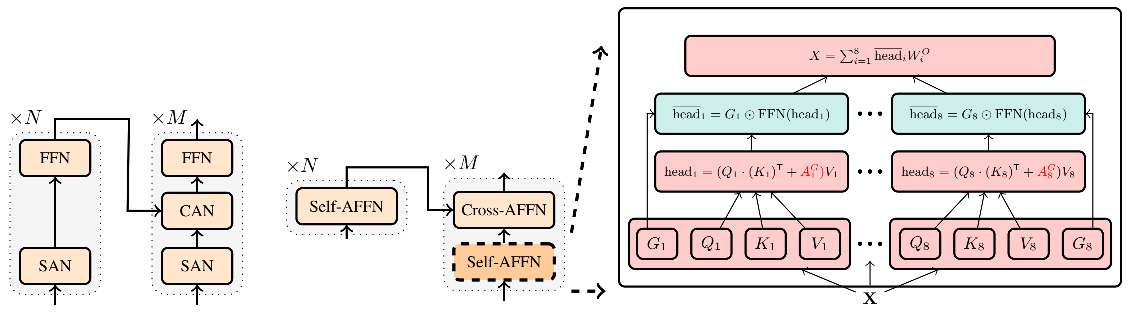
3 PartialFormer

3.1 Overall Architecture

Figure 2 illustrates the overall architecture of PartialFormer, encompassing both an encoder and a decoder. Although the foundational structure adheres to the design of the vanilla Transformer (Vaswani et al., 2017), there are some notable modifications.

Encoder. Different from vanilla Transformer, each encoder layer in PartialFormer consists of a unified sub-layer that integrates the PG-FFNs into the multi-head self-attention mechanism.

Decoder. Each decoder layer is composed of two types of sub-layers, both of which integrate the multi-head attention mechanism with PG-FFNs. The sub-layers differ based on the type of multi-head attention mechanisms employed, specifically


×N
×M
FFN
SAN
FFN
CAN
SAN
×N
×M
Self-AFFN
Cross-AFFN
Self-AFFN
$X = \sum_{i=1}^{8} \overline{\text{head}}_i W_i^O$
$\overline{\text{head}}_1 = G_1 \odot \text{FFN}(\text{head}_1)$
$\overline{\text{head}}_8 = G_8 \odot \text{FFN}(\text{head}_8)$
$\text{head}_1 = (Q_1 \cdot (K_1)^\intercal + A_1^G)V_1$
$\text{head}_8 = (Q_8 \cdot (K_8)^\intercal + A_8^G)V_8$
G_1 Q_1 K_1 V_1
Q_8 K_8 V_8 G_8
x

(a) Transformer (b) PartialFormer (c) PartialFormer Self-AFFN Block

Figure 2: (a) Architecture of Transformer. (b) Architecture of PartialFormer. (c) Details of Self-AFFN Block. All architecture are based on pre-normalization strategy. We omit the layer normalization operation, residual connection, softmax operation and scale coefficient for simplicity.

whether it's a decoder self-attention or an encoder-decoder cross-attention mechanism. Notably, this design is inspired by previous studies (Lu et al., 2019; Gulati et al., 2020), but it differs in that we employ small FFNs, known as PG-FFNs, within each attention head of both the self-attention and cross-attention modules. To reduce computation, we halved the hidden dimension of PG-FFNs. Further decoder comparisons are in Appendix C.

3.2 Partial-Level Gated FFN

Intuition Previous studies (Wu et al., 2020; Mehta et al., 2021; Ge et al., 2022) commonly reduced the parameters in feed-forward networks by decreasing the hidden dimension (e.g., 2048 to 256). Different from them, our key idea involves utilizing a collection of small FFNs to model smaller input features expecting them to collaboratively emerge better performance while consuming fewer parameters, akin to "Swarm Intelligence".

In the concept of "Swarm Intelligence", a vanilla FFN can be viewed as a single large individual, which processes the whole feature input, making it, while effective, very resource-intensive in terms of computing power and memory. Assume a vanilla FFN with mappings of 1024->4096->1024, which consumes around 8.4 million parameters. By contrast, if we utilize multiple smaller FFNs (viewed as multiple weak individuals), each of which processes a subset of the input feature and collaboratively utilizes these outputs to generate the final output, the parameter and computation consumption will be significantly fewer. For example, 8 smaller FFNs with mappings of 128->512->128, we can retain the same hidden dimension, such as 8 * 512, while using only 1.05 million parameters. This approach significantly reduces parameters while maintaining the crucial hidden dimension, as emphasized in Geva et al. (2021); Tran et al. (2021).

Design of PG-FFNs We have observed that the Transformer architecture inherently consists of multiple smaller subspaces, namely "heads" within the multi-head attention (MHA) mechanism. These heads act as sub-components of the original inputs and retain substantial information from the original data. Besides, the fusion mechanism in MHA enables the consolidation of the capabilities of multiple FFNs. As a result, PG-FFNs should naturally be constructed based on the MHA mechanism. More specifically, we insert multiple FFNs into the place between Eq. (2) and Eq. (3), as shown in the blue part of Figure 2(c).

While group transformation operations could be used to instantiate our idea, they are not optimal on GPUs due to their low I/O efficiency (Ma et al., 2018), causing significant inference latency. To address this, we propose sharing parameters across each FFN within different heads, thereby eliminating the need for group transformation operations. However, directly sharing weights may result in homogeneous representations across different heads, which may potentially hinder the performance (Li et al., 2018). To mitigate this, we further introduce a head-specific gated mechanism. The core idea is to use a set of diverse masks to filter the information of different heads so that the head representation will be more diverse.

Formally, given a set of head features $\{\text{head}_i | 1 \leq i \leq H\}$ and diverse masks $\{G_i | 1 \leq i \leq H\}$, the calculation of PG-FFNs is as:

$$\overline{\text{head}}_i = G_i \odot \text{FFN}(\text{head}_i), \tag{5}$$

where $\text{FFN}(\cdot)$ is the same as Eq. (4) and G_i is generated via multiplication between the input feature of the block X and a learnable matrix W_i^G followed by an activation function $\sigma(\cdot)$, e.g., ReLU, Sigmoid and Tanh, as follows: $G_i = \sigma(XW_i^G)$. We compared the choice of $\sigma(\cdot)$ in Table 7.

3.3 Residual-like Attention Calculation

Dong et al. (2021); Wang et al. (2022) have shown that the original location of FFNs plays an essential role in optimizing transformers, e.g., alleviating *Token Uniformity*. Therefore, it's vital to consider the impact of altering the FFN placement. Densely residual connections are effective but typically implemented either at the feature level (e.g., DLCL (Wang et al., 2019)) or integrated into the network structure (e.g., Realformer (He et al., 2021)), which are not flexible.

To this end, we design a new variant of the residual connection integrated into the attention calculation, while also decoupling from the network architecture. Specifically, the calculation of attention maps consists of two parts: 1) A^G, the global part, and 2) A^L, the local part, as shown in Figure 2(c). The calculation of A^L remains the same as in the vanilla Transformer, while A^G is computed once by using the original embedding as input through Eq. (1) (without softmax operation). Inspired by He et al. (2021), to efficiently fuse these components, we add them together and apply a Softmax function, as follows:

$$A_i = \mathrm{Softmax}(A_i^G + A_i^L), \qquad (6)$$

where A_i^G and A_i^L denote the global and local attention map of i-th head.

In addition to the benefit of efficient depth scaling (See Appendix G), this approach provides remarkable flexibility in combining different attention mechanisms, specifically tailored to address specific conditions. For instance, it allows for the utilization of local attention to calculate A^G when dealing with small datasets (see Appendix F).

3.4 Efficient Scaling Strategy

Though PG-FFN offers the advantage of reducing lots of parameters when applied directly to the transformer, it also leads to marginal performance degradation (see Table 10 (a)). Thus, a crucial aspect of this study is to determine how to effectively utilize the spared parameters. In this work, we adopt a hybrid scaling strategy, which has been validated in computer vision, e.g., EfficientNet (Tan and Le, 2019). Note that our approach differs from EfficientNet, as we incorporate a combination of head scaling and depth scaling into our method.

Head Scaling As aforementioned, PartialFormer is guided by "swarm intelligence" and operates with small subspaces. Expanding the number and size of these subspaces intuitively augments PartialFormer's capabilities. In response to this insight, we introduced a head-scaling strategy tailored specifically for PartialFormer, involving the direct addition of more heads and the expansion of their width, effectively bolstering its performance.

To achieve this objective, we decouple the relationship between the number of heads and the embedding size, specifically $d_k \times H \neq d$. This approach shares similarities with methods discussed in Bhojanapalli et al. (2020). However, it differs in its two-step process, which draws inspiration from the inherent redundancy observed in attention maps as discussed in Michel et al. (2019); Clark et al. (2019); Voita et al. (2019); Nguyen et al. (2022); Zheng et al. (2024). Given values for d_k, d, and H, we first create intermediate values for Q and K, and then we expand the attention maps to the desired number of heads using a robust MLP network. In the case of V, we generate them directly. This approach allows for the inclusion of more heads in PartialFormer while maintaining the same parameter budget.

We demonstrate that this scaling strategy is naturally well-suited for PartialFormer (see Section 6.4). Furthermore, it can also be regarded as a variation of width scaling, offering two significant advantages: 1) enabling flexible imbalanced computation distribution in encoder-decoder architecture, and 2) preventing an excessive distribution of parameters in the embedding and output layers.

4 Experimental Setups

We assess PartialFormer's performance across both machine translation and abstractive summarization tasks[2]. More details are given in Appendix A.

Dataset. For the machine translation task, we selected 9 datasets involving WMT'14 English-German (En-De), WMT'14 English-French (En-Fr), WMT'16 English-Romanian (En-Ro), and six translation tasks from WMT'17 benchmark. We preprocessed the raw data following the standard strategy. For the abstractive summarization task, we utilized the widely-used CNN-DailyMail dataset. We followed the same preprocessing approach as described in Ott et al. (2019). We applied

[2]We tested PartialFormer's performance in language modeling, with results in the Appendix.

Type	Model	N-M	d	d_k	H	Param	BLEU	COMET-22	sBLEU
Multi-Branch Architecture	Weighted Transformer (Ahmed et al., 2017)	6-6	1024	-	-	211M	28.90	-	-
	Multi-Unit Transformer (Yan et al., 2020)	6-6	-	-	-	130M	29.30	-	-
	MAT (Fan et al., 2020)	6-6	-	-	-	206M	29.90	-	-
	Multi-Path Transformer (Lin et al., 2022)	6-6	-	-	-	193M	29.68	-	-
Lightweight Architecture	Evolved Transformer (So et al., 2019)	-	-	-	-	64M	28.20	-	-
	Delight (Mehta et al., 2021)	-	640	-	-	54M	28.00	-	-
Weight Sharing	Universal Transformer (Dehghani et al., 2019)	-	1024	-	-	65M	28.90	-	-
	SubFormer (Reid et al., 2021)	-	-	-	-	63M	28.50	-	-
	SubFormer-big (Reid et al., 2021)	-	-	-	-	197M	29.30	-	-
	ODE Transformer (RK4) † (Li et al., 2022a)	6-6	512	-	-	62M	28.88	83.47	27.8
	ODE Transformer (RK2, Learn.) † (Li et al., 2022a)	24-6	512	-	-	118M	29.73	83.94	28.6
Other Comparisons	RealFormer (He et al., 2021)	18-18	512	64	8	151M	29.35	-	-
	DMAN † (Fan et al., 2021)	6-6	512	64	8	62M	27.54	82.27	26.4
	Mega-Softmax † (Ma et al., 2022)	6-6	512	-	1	64M	28.11	82.79	27.0
Our System	Transformer	24-6	512	64	8-8	118M	29.05	83.60	27.9
	PartialFormer (w/o Head Scaling)	24-6	512	64	8-8	66M	28.86	83.35	27.7
	PartialFormer	24-6	512	64	24-16	115M	30.09	84.17	29.0
	Transformer	6-6	512	64	8-8	62M	27.43	82.19	26.4
	PartialFormer (w/o Head Scaling)	6-6	512	64	8-8	42M	27.15	81.75	26.1
	PartialFormer	6-6	512	64	24-16	63M	28.60	83.21	27.5
	Transformer	24-6	360	45	8-8	62M	28.00	82.72	27.0
	PartialFormer (w/o Head Scaling)	24-6	360	45	8-8	36M	27.88	82.49	26.8
	PartialFormer	24-6	360	45	24-16	61M	29.23	83.74	28.1
	PartialFormer	24-6	360	45	30-16	68M	29.56	83.94	28.4

Table 1: Results on the WMT'14 En-De task. For a more fair comparison, we also re-implemented some state-of-the-arts models with same data and training strategy, as indicated by †.

joint byte pair encoding (BPE) (Sennrich et al., 2016) with sizes of 32K for all the tasks except the En-Ro task (20K), and CNN-DailyMail (30K).

Training & Evaluation. We trained models on GeForce RTX 3090 cards via Fairseq (Ott et al., 2019) toolkit primarily following the training strategy in Wang et al. (2019). For machine translation evaluation, we utilized *multi-BLEU* (Papineni et al., 2002), COMET-22 (Rei et al., 2022) and sacreBLEU (Post, 2018) scores. Following Wang et al. (2019), beam sizes were 4, 4, and 5 for En-De, En-Fr, and En-Ro tasks respectively. *Length_penalty* of 0.6, 0.8, and 1.3 were applied to En-De, En-Fr, and En-Ro tasks respectively. For the WMT'17 benchmark, beam size and *Length_penalty* were set to 4 and 1, respectively. We used an ensemble of last 10 checkpoints. For abstractive summarization, we set beam size, *Length_penalty*, minimum length and maximum length to 4, 2.0, 55 and 140, respectively. The evaluation metric was F1-Rouge (Lin, 2004)(Rouge-1, Rouge-2 and Rouge-L).

5 Experiments

5.1 Machine Translation

Table 1 presents the results for the WMT'14 En-De task. $N-M$, d, d_k, H and sBLEU denote encoder-decoder depths, embedding dimension, head dimension, number of heads and SacreBLEU, respectively. We made the following observations:

- PartialFormer achieves BLEU scores of 28.60, 29.56, and 30.09 in three different configurations, surpassing the standard Transformer by 1.17 BLEU points, 1.56 BLEU points, and 1.04 BLEU points with a similar model capacity. These observations are further supported by COMET-22 and sacreBLEU scores.
- Without the head scaling strategy, PartialFormer performs slightly worse than the standard Transformer (27.15 vs. 27.43, 27.88 vs. 28.00, and 28.86 vs. 29.05) but is significantly more parameter-efficient (42M vs. 62M, 36M vs. 62M, 66M vs. 118M). This is due to our PG-FFN structure, which maintains high hidden dimensions while reducing parameter usage.
- PartialFormer surpasses other multi-branch Transformers and state-of-the-art weight-sharing methods like ODE Transformer (Li et al., 2022a), as well as strong baselines such as Mega (Ma et al., 2022). Notably, ODE Transformer and Mega use extra relative position encoding and require more computational resources. Moreover, while Mega and DMAN train for up to 500K updates and 220 epochs, achieving BLEU scores of 29.01 and 29.10, our strategy involves only 50K updates, leading to their sub-optimal scores of 28.11 and 27.54 under similar conditions.

Tables 2, 3, and 4 showcase results for the WMT'14 En-Fr, WMT'16 En-Ro, and WMT'17 benchmarks, respectively. Similar trends are observed in these

Model	N	d	d_k	H	Param	BLEU	COMET-22
Weighted Transformer (2017)	6	-	-	-	211M	41.40	-
Evolved Transformer (2019)	-	-	-	-	64M	40.60	-
Delight (2021)	-	640	-	-	54M	40.50	-
ODE Transformer (RK4) (2022a)	6	-	-	-	69M	42.56	-
ODE Transformer (RK2, Learn.) (2022a)	24	-	-	-	123M	43.48	-
Multi-Path Transformer (2022)	-	-	-	-	168M	42.44	-
Transformer	24	512	64	8-8	120M	42.33	85.62
PartialFormer	24	512	64	24-18	119M	43.10	86.34
PartialFormer	24	512	64	24-24	127M	43.29	86.61
Transformer	6	512	64	8-8	63M	40.79	84.27
Transformer	24	360	45	8-8	64M	40.96	84.42
PartialFormer	24	360	45	24-18	63M	42.16	85.61
PartialFormer	24	360	45	24-24	67M	42.39	85.74

Table 2: Results on the WMT'14 En-Fr task.

Model	N	d	d_k	H	Param	BLEU	COMET-22
Delight (Mehta et al., 2021)	-	640	-	-	53M	34.70	-
Subformer (Reid et al., 2021)	-	-	-	-	48M	34.70	-
ODE Transformer (RK2 γ) † (2022a)	6	1024	64	16-16	192M	35.00	82.63
Transformer	24	512	64	8-8	111M	35.00	82.11
PartialFormer	24	320	40	24-24	48M	35.30	82.52

Table 3: Results on the WMT'16 En-Ro task. †denotes re-implementation with same data and training strategy.

Model	Fi⟷En		De⟷En		Lv⟷En		Avg.
	Fi→ En	En→ Fi	De→ En	En→ De	Lv→ En	En→ Lv	
Transformer	26.07	22.14	35.04	28.59	17.59	16.23	24.27
PartialFormer	**27.48**	**23.35**	**35.60**	**29.91**	**19.65**	**17.37**	**25.56**

Table 4: Results on the WMT'17 benchmark. PartialFormer has the same depth and d as the Transformer but consumes 1M fewer parameters on average.

tasks as in the En-De task.

MACs Comparison. Table 5 displayed the multiplication-addition operations (MACs), a metric for measuring neural network computations, on the En-De task. We made the following observations: 1) A deeper and narrower Transformer architecture consumes fewer computations while exhibiting superior performance (#1 vs. #2), 2) PartialFormer achieves comparable performance to the vanilla Transformer with the same width and depth, while utilizing fewer computations and parameters (#2 vs. #3), and 3) Head scaling is an efficient scaling strategy for PartialFormer to significantly improve its capacity (1.68 BLEU points) by adding 1.7B MACs and 32M parameters (#3 vs. #4).

5.2 Abstractive Summarization

Table 6 exhibited results on the CNN-DailyMail task. We can see that PartialFormer achieves better performance, as evidenced by higher Rough-1, Rough-2, and Rough-L scores, despite having fewer parameters (37M vs. 61M). This highlights the efficiency and effectiveness of the PartialFormer architecture in this task.

#	Model	N-M	d	d_k	H	MACs	Param	BLEU
1	Transformer	6-6	512	64	8-8	9.9B	62M	27.43
2	Transformer	24-6	360	45	8-8	6.3B	62M	28.00
3	PartialFormer (w/o hs)	24-6	360	45	8-8	5.2B	36M	27.88
4	PartialFormer	24-6	360	45	30-16	6.9B	68M	**29.56**

Table 5: MACs denote the multiplication-addition operations. We compute them via 20 source and target tokens following Mehta et al. (2021).

Model	N-M	d	d_k	H	Param	RG-1	RG-2	RG-L
Transformer	6-6	512	64	8-8	61M	41.21	18.32	37.83
PartialFormer	6-6	400	50	24-16	37M	**41.50**	**18.60**	**38.25**

Table 6: Rough-1, Rough-2 and Rough-L comparisons on CNN-DailyMail task.

#	Model	Param	BLEU
1	Transformer ($N = 24$, $d = 360$)	62M	28.00
2	Pure Attention ($N = 24$, $d = 360$)	31M	25.70
3	PartialFormer	68M	**29.56**
4	w/o Partial-level Gated FFN	52M	$\underline{27.51}$
5	w/o Residual-like Attention Calculation	66M	29.26
6	w/o Head Scaling	36M	27.88
7	PartialFormer (encoder only)	67M	29.15
8	PartialFormer (decoder only)	63M	28.80
9	PG-FFNs with Sigmoid activation	68M	29.21
10	PG-FFNs with Tanh activation	68M	29.03

Table 7: Ablation studies on WMT'14 En-De task.

6 Analysis

6.1 Ablation Studies

Table 7 presents an ablation study of PartialFormer on the WMT'14 En-De task, demonstrating the critical role of each component. Omitting any element causes performance decline, underscoring the holistic design. The PG-FFN removal (#3 vs. #4) results in a large performance drop of 2.05 BLEU points, despite a mere 16 million parameters reduction. This evidence corroborates previous findings (Dong et al., 2021) on the subpar performance of pure attention networks sans FFN, highlighting the essential role of PG-FFN in PartialFormer.

Besides, Table 7 shows the results of different PartialFormer configurations on the WMT'14 En-De task. The encoder-decoder PartialFormer achieves the highest performance, reaching 29.56 BLEU points, indicating the effectiveness of our approach in enhancing both the encoder and the decoder. Employing our concept to either the encoder or the decoder individually also improves performance, yet the encoder-decoder configuration persistently surpasses others, marking the greatest performance improvement.

Setting	H	d	d_k	Param	BLEU
Varying Encoder H	30-16	360	45	68M	29.56
	24-16	360	45	61M	29.23
	16-16	360	45	51M	29.02
Varying Decoder H	16-16	360	45	51M	29.02
	16-24	360	45	56M	28.85
	16-30	360	45	60M	29.20
Varying d_k	30-16	360	30	49M	28.70
	30-16	360	60	86M	29.68
	30-16	360	90	124M	30.00
Varying d	30-16	180	45	35M	27.61
	30-16	270	45	51M	28.80
	30-16	450	45	84M	29.41

Table 8: Parameters analysis on WMT'14 En-De task.

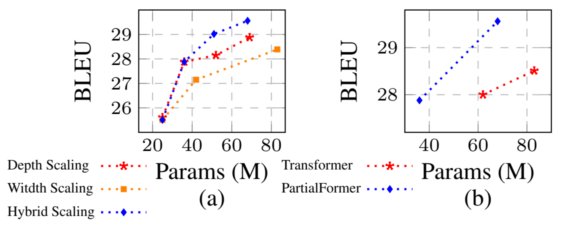


Figure 3: (a) Scaling Up PartialFormer with Different Methods. (b) Scaling Transformer and PartialFormer with Head Scaling.

6.2 Comparison of Gating Strategy

Table 7 (#9 and #10) presents a comparison of various activation functions used in PG-FFN. The results indicate that the default choice, ReLU activation, yields the best performance. One explanation is that the ReLU activation provides hard masks for filtering the information of different heads, compared to other activation functions. Such hard masks can make different heads more diverse.

6.3 Hyper-Parameter Analysis

Since the proposed method relies on multiple parameters, we conducted additional experiments and analyses with different hyper-parameters, including the number of heads, head dimensions, and embedding dimensions, to further strengthen the robustness of our findings. Table 8 presented the results on the WMT'14 En-De task. We can observe that PartialFormer demonstrates strong performance across various choices of H, d_k, and d. This suggests that the superiority of PartialFormer arises from its efficient architecture design rather than hyper-parameter optimization.

6.4 Analysis of Scaling Approaches for PartialFormer

To disentangle the contribution of our proposed scaling method from the PartialFormer architecture, Figure 3 compares the WMT'14 En-De performance of different scaling methods. Specifically, the initial setting is the PartialFormer ($N - M = 6 - 6, H = 8 - 8, d = 360$). It's important to note that our hybrid scaling initially employs depth scaling, followed by head scaling. In general, all scaling methods improve BLEU scores with the cost of more parameters, but our hybrid scaling method can further improve BLEU, by up to 2.3%, than other scaling methods, suggesting the importance of our proposed hybrid scaling.

Head scaling can also improve the vanilla Transformer, though it is not as effective as in PartialFormer. Notably, PartialFormer attains 0.0525 BLEU per million parameters, significantly outperforming the vanilla Transformer (0.0243). This highlights the suitability of head-scaling for PartialFormer's design, a key contribution of this paper.

6.5 Analysis on Behaviours of FFN

Metric. Following Zhang et al. (2022), we examine FFN behaviors across four aspects: activation neuron count (namely $n_{\text{act.}}$), FFNs' hidden dimension, activation-neuron ratio (activations divided by hidden dimension, namely $R_{\text{act.}}$), and FFN efficiency (activations divided by parameters, namely η_{ffn}). Notably, for PartialFormer, the hidden dimension represents the concatenation of hidden dimensions from all smaller FFNs.

Results. Figure 4 (a-c) exhibits the results on the En-De test set. It is evident that PartialFormer has a lower activation ratio than the vanilla Transformer, as shown in Figure 4 (b). This indicates that PG-FFNs present lower utilization of the hidden dimension compared to the vanilla FFNs. However, our PG-FFN is parameter consumption friendly, enabling larger hidden layer dimensions with the same parameter budget (e.g., 5400 vs. 1440). Despite lower utilization of hidden dimension, it can still own more activated neurons, as depicted in Figure 4 (a). Additionally, our PG-FFN exhibits higher efficiency compared to vanilla FFNs, as shown in Figure 4 (c).

6.6 Analysis on Head Diversity

Metric. We select the same metric, namely D_{output}, as that in Li et al. (2018) to measure the diversity among head features. In this metric, a larger value indicates a higher level of diversity.

Results. From Figure 4 (d), we can observe that PartialFormer exhibits more diverse head features compared to the vanilla Transformer. This aligns with previous study (Li et al., 2018), which demonstrates the positive impact of head feature diversity

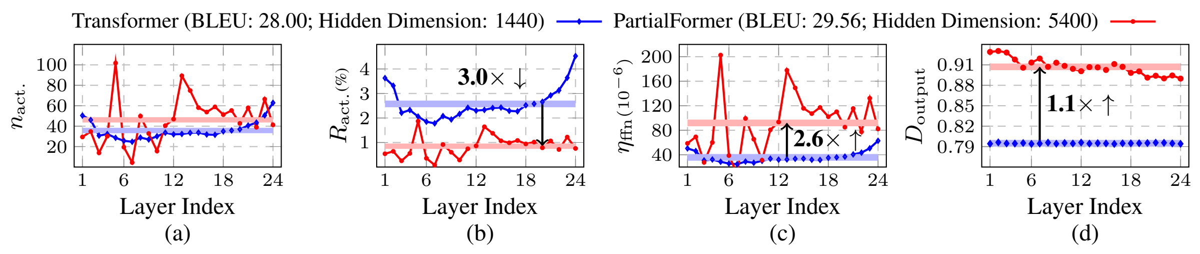


Figure 4: Analysis on behaviours of FFNs and head diversity in Transformer and PartialFormer.

# Model	Batch Size	Total Updates	Training Speed (sec. / 100 updates)	BLEU
1 Vanilla Transformer	8 x 4096 x 2	50K	28	28.00
2 Mega-Softmax	8 x 8192 x 1	500K	37	29.01
3 Mega-Softmax (50k updates)	8 x 8192 x 1	50K	37	28.11
4 ODE Transformer	8 x 4096 x 2	50K	34	29.03
5 ODE Transformer (reproduced)	8 x 4096 x 2	50K	34	28.88
6 SubFormer	8 x 8192 x 2	250K (max)	42	28.50
7 PartialFormer	8 x 4096 x 2	50K	40	29.56

(a) Training Phase

# Model	Param	Speed (Tok./s)	Peak Memory	COMET
PartialFormer vs. vanilla Transformer				
1 Transformer	62M	4325	3.0G	82.72
2 PartialFormer (larger batch)	36M	6579	3.0G	82.49
PartialFormer vs. ODE Transformer				
3 ODE Transformer	118M	3254	8.9G	83.94
4 PartialFormer	68M	3023	3.3G	83.94

(b) Inference Phase

Table 9: Efficiency analysis between PartialFormer and other Transformer variants.

on the Transformer model's performance. Thus, we conclude that the insertion of FFNs into attention mechanism may be a more optimal design.

6.7 Efficiency Analysis

Convergence Analysis Table 9 (a) compared the convergence updates, training speed and BLEU scores of PartialFormer with other methods. We made the following observations:

- PartialFormer and ODE Transformer do not require more training updates to achieve higher performance than vanilla transformer, unlike other strong baselines.
- All improved methods indeed lead to increased running latency.
- PartialFormer achieve highest BLEU scores among all the comparisons.

Overall, we believe PartialFormer can achieve significant performance improvements while maintaining good training efficiency.

Inference Analysis Table 9 (b) exhibits the inference efficiency on the test set of En-De task. We can see following observations: 1) Under the constraints of desired memory and performance, PartialFormer exhibits higher inference efficiency (6579 vs. 4325) when compared to the vanilla Transformer (#1 vs. #2). This revealed that PartialFormer has good practicability, and 2) In comparison to ODE Transformer, PartialFormer achieves similar inference speed and performance while significantly reducing memory consumption. This underscores PartialFormer's superiority over weight-sharing methods by effectively eliminating redundant computations.

6.8 PG-FFNs vs. Vanilla Lightweight FFN

In this section, we further emphasized PG-FFNs' superiority over vanilla lightweight FFN.

Settings. We replaced the Transformer's FFNs with our PG-FFNs. In the decoder, we only integrated PG-FFNs for cross-attention, aligning with the vanilla Transformer. We set Transformer with reduced FFN hidden dimensions (384) as baseline.

Results. Table 10 (a) showcases the superior efficiency of our PG-FFNs. They outperform vanilla lightweight FFNs (26.82 vs. 26.07) with similar computational resources (40M vs. 41M, 7.7B vs. 7.7B). This is attributed to PG-FFNs' ability to maintain a large hidden dimension while using fewer parameters and computations, setting them apart from existing lightweight FFNs.

6.9 Combination with Existing Architectures

We further investigated the adaptability and effectiveness of PartialFormer by applying it to three kinds of existing state-of-the-art architectures: 1) weight sharing methods (Lan et al., 2020), 2) gated linear units (Dauphin et al., 2017) and 3) deep Transformer methods (Wang et al., 2019). We utilized the ODE Transformer (Li et al., 2022a), known for its parameter efficiency. Additionally, we selected Swi-GLU (Shazeer, 2020) and DLCL (Wang et al., 2019).

Table 10 (b) shows the results. PartialFormer-DLCL achieved the highest performance, outper-

Model	N-M	d	d_k	H	MACs	Param	BLEU	COMET
Transformer	6-6	512	64	8-8	9.9B	62M	27.43	82.19
Transformer + LW FFNs	6-6	512	64	8-8	7.7B	41M	26.07	81.13
Transformer + PG-FFNs	6-6	512	64	8-8	7.7B	40M	26.82	81.72

(a) PG-FFNs vs. Vanilla FFNs.

Model	Param	BLEU
PartialFormer	68M	29.56
PartialFormer-ODE	68M	29.71
PartialFormer-GLU	68M	29.67
PartialFormer-DLCL	68M	29.88

(b) Results of PartialFormer variants.

Table 10: (a) PG-FFNs offer a compelling alternative to vanilla FFNs; (b) More results of PartialFormer variants. Metrics are reported on WMT'14 En-De.

forming PartialFormer by 0.32 BLEU points, while PartialFormer-GLU showed the smallest improvement with an increase of 0.11 BLEU points. We attribute this to the fact that DLCL is an architecture-level modification addressing different issues from PartialFormer. In contrast, both GLU and ODE focus on the parameter-efficiency problem. Although ODE is also an architecture-level modification, its goal significantly overlaps with that of PartialFormer, leading to moderate performance improvements when combined. This indicates that PartialFormer already significantly enhances parameter efficiency, as adding ODE and GLU does not yield substantial performance gains.

7 Related Work

Lightweight Transformers Several strands of research have been dedicated to enhancing the parameter efficiency of the Transformer architecture, each taking a distinct approach to the problem at hand. The first category aims to mitigate redundancy directly through architectural innovations, employing more efficient transformation operations (Mehta et al., 2019, 2021), integrating disparate yet synergistic patterns (Wu et al., 2020), or leveraging neural architecture search techniques (So et al., 2019). Another avenue of research explores weight sharing as a means of improving parameter efficiency, exemplified by the Universal Transformer's cross-layer parameter sharing strategy (Dehghani et al., 2019; Reid et al., 2021). Moreover, Li et al. (2022a) introduced an ordinary differential equation-inspired weight-sharing approach to achieve higher performance. Different from them, our study focus on the design of lightweight FFN.

Multi-Branch Transformer The multi-branch strategy is widely used in Transformer design. Weighted Transformer (Ahmed et al., 2017) employs a multi-branch FFN, while Multi-attentive Transformer (Fan et al., 2020), Multi-units Transformer (Yan et al., 2020), and Multi-Path Transformer (Lin et al., 2022; Li et al., 2023) extend this concept to different components of the Transformer. Our PartialFormer can be viewed as a pure multi-branch architecture based on natural subspaces.

Scaling Strategy in Transformer Deepening (Bapna et al., 2018; Wang et al., 2019; Li et al., 2020) and widening (Vaswani et al., 2017; Wu et al., 2021) Transformer have been well-acknowledged as two strategies to improve the capacity of Transformer in literature. In this work, PartialFormer adopts two alternative strategies to improve the capacity: specifically, it enhances both the number of attention heads and the dimensions of each head.

8 Conclusion

In this paper, we present PartialFormer, a new parameter-efficient Transformer architecture that offers an alternative approach to the design of the lightweight FFN. By employing multiple small FFNs and leveraging matrix factorization techniques, PartialFormer effectively reduces the number of parameters in the FFN. Moreover, we propose two innovative operations to further efficiently enhance the model capabilities. Experimental results across various machine translation tasks showcase the significant performance improvements achieved by PartialFormer, while maintaining comparable parameter consumption.

Acknowledgments

This work was supported in part by the National Science Foundation of China (No.62276056), the Natural Science Foundation of Liaoning Province of China (2022-KF-16-01), the Fundamental Research Funds for the Central Universities (Nos. N2216016 and N2316002), the Yunnan Fundamental Research Projects (No. 202401BC070021), and the Program of Introducing Talents of Discipline to Universities, Plan 111 (No.B16009).

Limitations

Despite the potential advantages of Partialformer in terms of parameter utilization and performance within a limited parameter budget, it is important to note that the existing conclusions regarding its effectiveness have not been thoroughly examined in the context of large-scale datasets and a higher number of parameters. Further research is needed to validate the claims and assess the scalability of Partialformer in more challenging scenarios.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *CoRR*.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3028–3033, Brussels, Belgium. Association for Computational Linguistics.

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. 2020. Low-rank bottleneck in multi-head attention models. *ArXiv*, abs/2002.07028.

Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Larissa Conradt and Timothy J Roper. 2005. Consensus decision making in animals. *Trends in ecology & evolution*, 20(8):449–456.

Iain D Couzin. 2009. Collective cognition in animal groups. *Trends in cognitive sciences*, 13(1):36–43.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2021. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 2793–2803.

Yang Fan, Shufang Xie, Yingce Xia, Lijun Wu, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2020. Multi-branch attentive transformer. *ArXiv*, abs/2006.10270.

Zhihao Fan, Yeyun Gong, Dayiheng Liu, Zhongyu Wei, Siyuan Wang, Jian Jiao, Nan Duan, Ruofei Zhang, and Xuanjing Huang. 2021. Mask attention networks: Rethinking and strengthen transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1692–1701, Online. Association for Computational Linguistics.

Tao Ge, Si-Qing Chen, and Furu Wei. 2022. Edge-Former: A parameter-efficient transformer for on-device seq2seq generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10786–10798, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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 *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 5036–5040. ISCA.

Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2021. RealFormer: Transformer likes residual attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 929–943, Online. Association for Computational Linguistics.

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, JingBo Zhu, Xuebo Liu, and Min Zhang. 2022a. ODE transformer: An ordinary differential equation-inspired model for sequence generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8335–8351, Dublin, Ireland. Association for Computational Linguistics.

Bei Li, Yi Jing, Xu Tan, Zhen Xing, Tong Xiao, and Jingbo Zhu. 2023. TranSFormer: Slow-fast transformer for machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6883–6896, Toronto, Canada. Association for Computational Linguistics.

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 the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 995–1005, Online. Association for Computational Linguistics.

Bei Li, Tong Zheng, Yi Jing, Chengbo Jiao, Tong Xiao, and Jingbo Zhu. 2022b. Learning multiscale transformer models for sequence generation. In *International Conference on Machine Learning*, pages 13225–13241. PMLR.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2897–2903, Brussels, Belgium. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ye Lin, Shuhan Zhou, Yanyang Li, Anxiang Ma, Tong Xiao, and Jingbo Zhu. 2022. Multi-path transformer is better: A case study on neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5646–5656, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Understanding and improving transformer from a multi-particle dynamic system point of view. *CoRR*, abs/1906.02762.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part XIV*, page 122–138, Berlin, Heidelberg. Springer-Verlag.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. 2022. Mega: Moving average equipped gated attention. *CoRR*.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. Delight: Deep and light-weight transformer. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari, and Hannaneh Hajishirzi. 2019. Define: Deep factorized input word embeddings for neural sequence modeling. *ArXiv*, abs/1911.12385.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Tan Minh Nguyen, Tam Minh Nguyen, Hai Ngoc Do, Khai Nguyen, Vishwanath Saragadam, Minh Pham, Nguyen Duy Khuong, Nhat Ho, and Stanley Osher. 2022. Improving transformer with an admixture of attention heads. In *Advances in Neural Information Processing Systems*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

David R. So, Quoc V. Le, and Chen Liang. 2019. The evolved transformer. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5877–5886.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. 2021. Facebook AI's WMT21 news translation task submission. In *Proceedings of the Sixth Conference on Machine Translation*, pages 205–215, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Peihao Wang, Wenqing Zheng, Tianlong Chen, and Zhangyang Wang. 2022. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

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 the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy. Association for Computational Linguistics.

Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. 2021. R-drop: Regularized dropout for neural networks. *Advances in Neural Information Processing Systems*, 34:10890–10905.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020. Lite transformer with long-short range attention. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Jianhao Yan, Fandong Meng, and Jie Zhou. 2020. Multi-unit transformers for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1047–1059, Online. Association for Computational Linguistics.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland. Association for Computational Linguistics.

Tong Zheng, Bei Li, Huiwen Bao, Tong Xiao, and Jingbo Zhu. 2024. Eit: Enhanced interactive transformer. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

A Detailed Setups of Experiments

A.1 Dataset

Table 11 displays the statistics of all the 9 translation task.

A.2 Training Details

Table 12 and 13 exhibits the training details on all translation tasks.

B Implementation of Previous State-of-the-art Methods

The accuracy of fairseq-based translation results can vary due to tokenization methods and other factors. To address fairness concerns, we re-implemented three state-of-the-art approaches in our codebase. To ensure absolute fairness, we employed the identical training strategy and data usage as in our PartialFormer model.

Data. The dataset is sourced from Google's open release, featuring BPE operations totaling 32K.

Training Strategy. Our training strategy is the same as that of Wang et al. (2019), where 0.002 learning rate, 16000 warmup steps, pre-norm, relu_dropout=0.1, attention dropout=0.1, 4096 tokens per GPUs (8 GPUs) and update the parameters every 2 steps.

C Ablation on Design of Decoder

The design of the Decoder is a crucial component of the Transformer architecture due to its direct association with decoding. We evaluated three configurations: 1) Integrating PG-FFNs into both the decoder's self-attention and cross-attention, while halving the hidden dimension, 2) Incorporating PG-FFNs solely into the decoder's cross-attention, and 3) Incorporating PG-FFNs solely into the decoder's self-attention.

Table 14 exhibited the results on the WMT'14 En-De task. Our observations are as follows: 1) The first configuration yields the best performance, aligning with the insights from Gulati et al. (2020); Lu et al. (2019), 2) Using a single PG-FFN in each layer also delivers commendable results with a score of 29.21, and 3) Excluding PG-FFNs from the decoder's cross-attention results in erratic training, which is expected since there are no FFNs to handle the cross-attention features.

D Metric Definition

D.1 Measurement of Head Diversity

Following Li et al. (2018), we measure the head diversity as follows:

$$D_{\text{output}} = \exp(-\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}\frac{|O^i \cdot O^j|}{\|O^i\|\|O^j\|}) \quad (7)$$

During evaluation, we calculate the metric on all samples and average the values to obtain the final result.

Dataset	Sentence Train	Sentence Dev	Sentence Test	BPE	Vocab
WMT'14 En-De	4.5M	2999	3003	32K	34040
WMT'14 En-Fr	36M	26815	3003	32K	37288
WMT'16 En-Ro	0.6M	1999	1999	20K†	19064
WMT'17 En-De	5.9M	7998	3004	32K	35488
WMT'17 De-En	5.9M	7998	3004	32K	35448
WMT'17 En-Fi	2.7M	4225	3002	32K	32584
WMT'17 Fi-En	2.7M	4225	3002	32K	32584
WMT'17 En-Lv	4.5M	2003	2001	32K	32368
WMT'17 Lv-En	4.5M	2003	2001	32K	32368

Table 11: The details of datasets of 9 translation tasks.†: we follow the settings in Li et al. (2022b).

	En-De	En-Ro	En-Fr
GPUs	8	4	8
Batch Size	4096	4096	4096
Update Frequency	2	1	8
Optimer	Adam	Adam	Adam
Adam_β	(0.9, 0.997)	(0.9, 0.997)	(0.9, 0.997)
LR	0.0020	0.0020	0.0020
LR scheduler	inverse sqrt	inverse sqrt	inverse sqrt
Initial LR	$1e^{-7}$	$1e^{-7}$	$1e^{-7}$
Total updates	50K	25K	100K
Warmup updates	16000	8000	16000
Weight decay	0.0000	0.0000	0.0000
Label smoothing	0.1	0.1	0.1
Dropout	0.1	0.1	0.1
Attention dropout	0.1	0.1	0.1
ReLU dropout	0.1	0.1	0.1

Table 12: The training setups of WMT'14 En-De, WMT'16 En-Ro and WMT'14 En-Fr tasks.

E More Comparison with Previous Lightweight Transformer

Table 15 presents a comprehensive comparison of previous lightweight Transformer models on the En-De task's test set, with a specific focus on operating within a smaller parameter budget. The

	En-{De, Lv}	{De, Lv}-En	En-Fi	Fi-En
GPUs	8	8	8	8
Batch Size	4096	4096	4096	4096
Update Frequency	2	1	1	4
Optimer	Adam	Adam	Adam	Adam
Adam$_\beta$	(0.9, 0.997)	(0.9, 0.997)	(0.9, 0.997)	(0.9, 0.997)
LR	0.0020	0.0020	0.0020	0.0020
LR scheduler	inverse sqrt	inverse sqrt	inverse sqrt	inverse sqrt
Initial LR	$1e^{-7}$	$1e^{-7}$	$1e^{-7}$	$1e^{-7}$
Total updates	50K/17K	50K/17K	40K	10K
Warmup updates	16000	16000	16000	16000
Weight decay	0.0000	0.0000	0.0000	0.0000
Label smoothing	0.1	0.1	0.1	0.1
Dropout	0.1	0.1	0.1	0.1
Attention dropout	0.1	0.1	0.1	0.1
ReLU dropout	0.1	0.1	0.1	0.1

Table 13: The training setups of WMT'17 benchmark.

Model	Param	BLEU
PartialFormer	68M	29.56
-PGFFNs in decoder self-AFFN	66M	29.21
-PGFFNs in decoder cross-AFFN	66M	Failed

Table 14: **Utilizing PG-FFNs in both the decoder's self-attention and cross-attention mechanisms is a preferable option.** BLEU points are reported in WMT'14 En-De task.

Model	Param	BLEU
DELIGHT (Mehta et al., 2021)	23M	26.70
EdgeFormer (Ge et al., 2022)	-	26.90
Lite Transformer (Wu et al., 2020)	-	26.50
PartialFormer	27M	**27.50**
Evolved Transformer (So et al., 2019)	48M	27.70
DELIGHT (Mehta et al., 2021)	37M	27.60
ODE Transformer (Li et al., 2022a)	37M	28.24
PartialFormer	36M	**28.35**

Table 15: Comparison with state-of-the-art models of smaller capacities on the En-De task.

A_G	A_L	Param	BLEU
RPR	MHSA	62M	35.76

Table 16: Results of several PartialFormer variants on the En-De task.

results prominently showcase the outstanding performance of PartialFormer, even when faced with constraints on model capacity. This outcome further emphasizes the superior capabilities of PartialFormer in scenarios with limited resources.

F PartialFormer with Different A_G for Small Dataset

Table 16 showcases the results of PartialFormer on the WMT'16 En-Ro task, a small-scale translation dataset, specifically when A_G is calculated using local attention (Shaw et al., 2018). Notably, these results reveal that by adopting such an approach, PartialFormer achieves an impressive BLEU score of 35.76. We hope this can shed lights on the area of model integration.

G Analysis on Token Uniformity

Following (Dong et al., 2021; Wang et al., 2022), we measure the token uniformity among token representations. We use pearson correlation to compute it.

From Figure 5, we can observe that PartialFormer owns a lower token uniformity among token representations than the vanilla Transformer, revealing that PartialFormer can benefit from depth scaling efficiently (Dong et al., 2021; Wang et al., 2022).

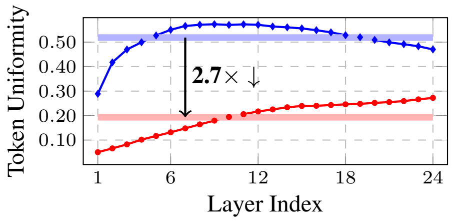


Figure 5: Comparison of token uniformity (lower is better) in Transformer and PartialFormer.

H Preliminary Experiments on Language Modeling

We also evaluate the effectiveness of PartialFormer on the language modeling task.

Dataset. For the language modeling task, we utilized the WikiText-103 dataset for evaluation. The training set comprises 103 million words from 28,000 articles, while the validation and test sets contain 218,000 and 246,000 words, respectively. We followed the data acquisition and preprocessing instructions from Fairseq (Ott et al., 2019).

Training & Evaluation. The training and evaluation settings adhere to the standard guidelines for language modeling in PyTorch (Ott et al., 2019). We trained all models over 286,000 updates.

Results. Table 17 exhibited results on the WikiText-103 task. PartialFormer surpasses the Adaptive Input model (Baevski and Auli, 2019) with a lower test perplexity of 19.87 compared to 21.11. Remarkably, PartialFormer achieves this with slightly fewer parameters (143M vs. 147M), demonstrating its efficiency and effectiveness as a

Model	N	d	d_k	H	**Param**	**Test PPL**
Adaptive Input	8	1024	128	8	147M	21.11
PartialFormer	16	1024	256	4	143M	**19.87**

Table 17: Results on the WikiText-103 dataset.

language model for WikiText-103. We will present more comprehensive experiments in the future.