TOKENFORMER: RETHINKING TRANSFORMER SCALING WITH TOKENIZED MODEL PARAMETERS

——Wang et al.,2024

Li Weizhen

School of Mathematical Sciences, Zhejiang University

2024.12.18

Contents

Introduction

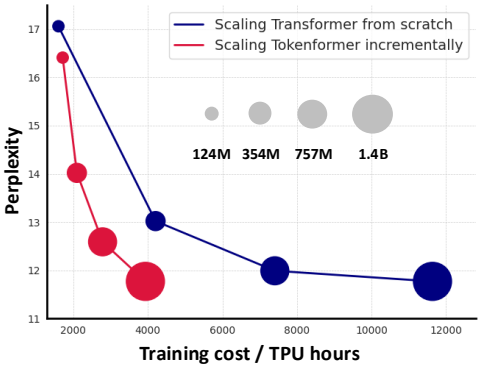
Transformers have become the predominant architecture in foundation models due to their excellent performance across various domains. However,

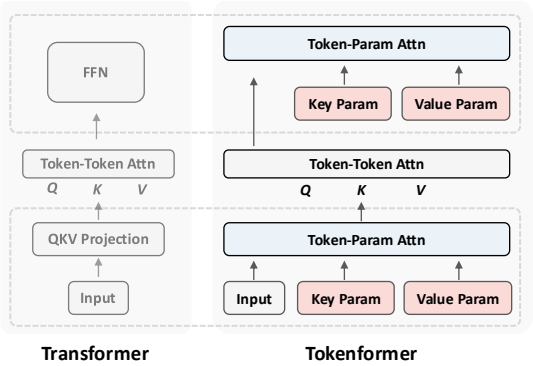
- scaling these models requires substantial costs;
- architectural modifications typically require the entire model to be retrained from scratch.

The **Tokenformer**, a natively scalable architecture that totally leverages the attention mechanism, was proposed.

- Treat model parameters as tokens and replace all linear projections in Transformers with token-parameter attention layers;
- Input tokens act as queries and model parameters as keys and values;
- Allow for progressive and efficient scaling without retraining from scratch;
- Achieve performance comparable to Transformers while greatly reducing the training costs.

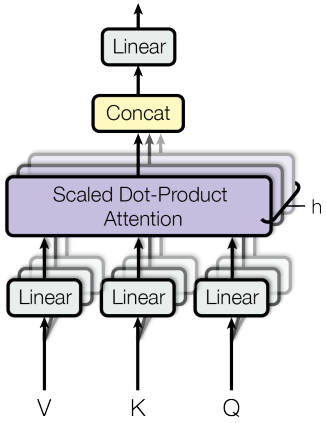
Transformer vs Tokenformer





A simple comparison between Transformers and Tokenformers.

Transformer



Input tokens: $X \in \mathbb{R}^{T \times d}$

$$Q = X \cdot W^Q, \quad K = X \cdot W^K, \quad V = X \cdot W^V; \quad (1)$$

$W^Q, W^K \in \mathbb{R}^{d \times d_k}$, $W^V \in \mathbb{R}^{d \times d_v}$.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \quad (2)$$

Output:

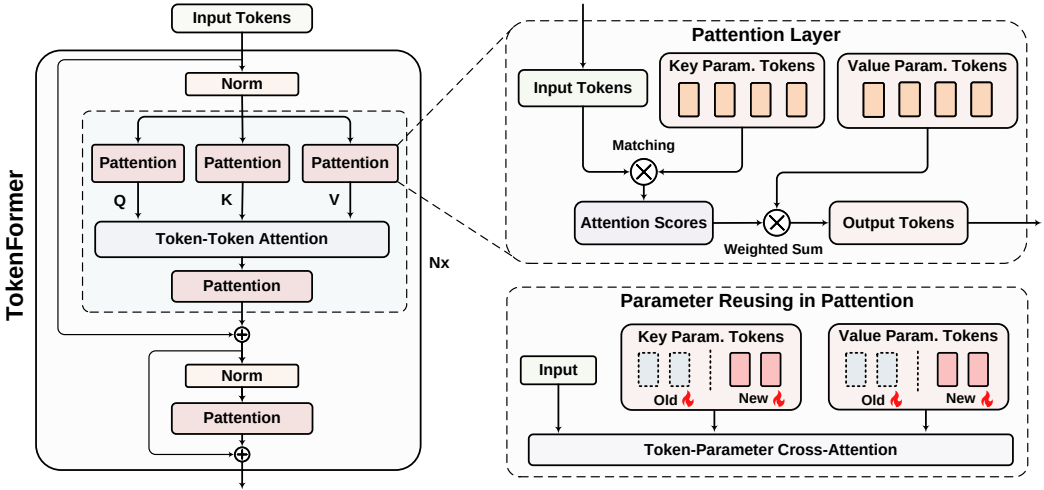
$$O_{\text{att}} = X_{\text{att}} \cdot W^O, \quad W^O \in \mathbb{R}^{d_v \times d}.$$

FFN:

$$O_{\text{ffn}} = \text{Activation}(O_{\text{att}} \cdot W_1 + b_1)W_2 + b_2.$$

$W_1 \in \mathbb{R}^{d \times d_{\text{ffn}}}$, $W_2 \in \mathbb{R}^{d_{\text{ffn}} \times d}$.

TokenFormer



Pattention Layer

Let $K_P \in \mathbb{R}^{n \times d_1}$ and $V_P \in \mathbb{R}^{n \times d_2}$ represent the learnable parameter tokens.(n is the number of key-value pairs)

$$\text{Pattention}(X, K_P, V_P) = \theta(X \cdot K_P^T) \cdot V_P, \tag{3}$$

where θ is the modified softmax function. The output Pattention scores, $S \in \mathbb{R}^{n \times n}$, are computed as:

$$S_{ij} = \text{GeLU}\left(\frac{A_{ij} \times \tau}{\sqrt{\sum_{k=1}^{n} |A_{ik}|^2}}\right), \quad \forall\ i, j \in 1...n, \tag{4}$$

where $A = X \cdot K_P^T$ and τ is a scale factor.

In transformer, we have $Q = X \cdot W_Q$.
In tokenformer, we have $Q = \text{Pattention}(X, K_P^Q, V_P^Q)$.

Overall Architecture

- QKV Pattention:

$$Q = \text{Pattention}(X, K_P^Q, V_P^Q), \quad K = \text{Pattention}(X, K_P^K, V_P^K), \quad V = \text{Pattention}(X, K_P^V, V_P^V).$$

- Token-Token Attention:

$$X_{\text{att}} = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \cdot V,$$

$$O_{\text{att}} = \text{Pattention}(X_{\text{att}}, K_P^O, V_P^O).$$

- FFN:

$$O_{\text{ffn}} = \text{Pattention}(O_{\text{att}}, K_P^{\text{ffn}}, V_P^{\text{ffn}}).$$

Progressive Model Scaling

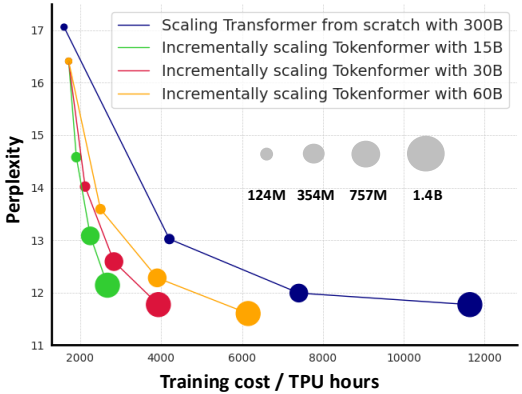
Consider an existing Tokenformer model equipped with a set of pre-trained key-value parameter tokens, denoted as $K_P^{\text{old}}, V_P^{\text{old}} \in \mathbb{R}^{n \times d}$. To scale the model, just augment this set by appending new key-value parameter tokens $K_P^{\text{new}}, V_P^{\text{new}} \in \mathbb{R}^{m \times d}$ as

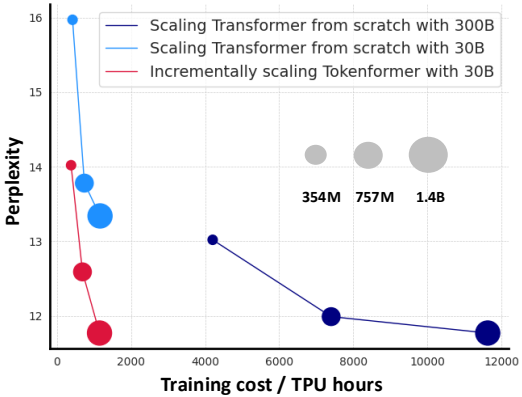
$$K_P^{\text{scale}} = \left[K_P^{\text{old}}, K_P^{\text{new}}\right], \qquad V_P^{\text{scale}} = \left[V_P^{\text{old}}, V_P^{\text{new}}\right]. \tag{5}$$

$$O = \text{Pattention}\left(X, K_P^{\text{scale}}, V_P^{\text{scale}}\right). \tag{6}$$

Importantly, by initializing K_P^{new} with zero, the model can perfectly resume the model state from the pre-training phase without losing the well-learned knowledge, facilitating faster convergence and accelerating the overall scaling process.

Experiments—Progressive Modeling Scaling





Experiments—Benchmarking

Model	#Param	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	Average acc ↑
Pythia-160M	160M	29.64	37.25	35.4	30.3	62.3	43.6	23.6	**51.3**	40.1
Ours (TokenFormer-150M)	150M	**10.45**	**16.38**	**45.0**	**35.5**	**64.9**	**47.3**	**24.9**	50.4	**44.7**
Pythia-410M	410M	9.95	10.84	51.4	40.6	66.9	52.1	24.6	53.8	48.2
Ours (TokenFormer-450M)	450M	**8.28**	**7.69**	**57.3**	**47.5**	**69.5**	**56.2**	**26.7**	**54.6**	**52.0**
Pythia-1B	1B	7.82	7.92	56.1	47.2	70.7	57.0	27.1	53.5	51.9
Ours (TokenFormer-900M)	900M	**7.38**	**5.46**	**64.0**	**55.3**	**72.4**	**59.9**	**30.6**	**56.4**	**56.4**
GPT-Neo 1.3B	1.3B	-	7.50	57.2	48.9	71.1	56.2	25.9	54.9	52.4
OPT-1.3B	1.3B	-	6.64	58.0	53.7	72.4	56.7	29.6	59.5	55.0
Pythia-1.3B	1.3B	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	55.2
GPT-Neo 2.7B	2.7B	-	5.63	62.2	55.8	71.1	61.1	30.2	57.6	56.5
OPT-2.7B	2.7B	-	5.12	63.6	**60.6**	74.8	60.8	31.3	**61.0**	58.7
Pythia-2.8B	2.8B	-	**5.04**	64.7	59.3	74.0	64.1	**32.9**	59.7	59.1
Ours (TokenFormer-1.5B)	1.5B	**6.91**	5.24	**64.7**	60.0	**74.8**	**64.8**	32.0	59.7	**59.3**

Zero-shot Evaluations.

Experiments—Benchmarking

Method	Image Size	#Param	Top-1 acc
ViT-B/16	384^2	86M	77.9
DeiT-B/16	224^2	86M	81.8
ViT-B/16 (MAE)	224^2	86M	82.3
Ours (TokenFormer-B/16†)	224^2	86M	82.1
Ours (TokenFormer-B/16)	224^2	109M	**82.5**
ViT-L/16	384^2	307M	76.5
ViT-L/16 (MAE)	224^2	307M	82.6
Ours (TokenFormer-L/16†)	224^2	307M	83.0
Ours (TokenFormer-L/16)	224^2	407M	**83.1**

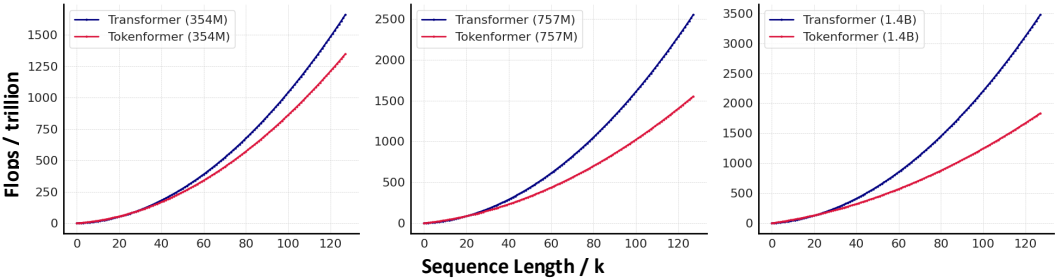
Image Classification.

Experiments—Comparison with Transformer

Operation	Parameter		Training FLOPs	
	Transformer	Ours	Transformer	Ours
Embed	$n_{\text{vocab}} d_{\text{model}}$	$n_{\text{vocab}} d_{\text{model}}$	-	-
Attention: QKV Project	$3n_{\text{layer}} d_{\text{model}}^2$	$n_{\text{layer}} d_{\text{token}} (n_{\text{q}} + n_{\text{k}} + n_{\text{v}})$	$6n_{\text{layer}} d_{\text{model}}^2 T$	$2n_{\text{layer}} d_{\text{token}} (n_{\text{q}} + n_{\text{k}} + n_{\text{v}}) T$
Attention: Token-Token	-	-	$4n_{\text{layer}} d_{\text{model}} T^2$	$4n_{\text{layer}} d_{\text{token}} T^2$
Attention: Output Project	$n_{\text{layer}} d_{\text{model}}^2$	$n_{\text{layer}} d_{\text{token}} n_{\text{o}}$	$2n_{\text{layer}} d_{\text{model}}^2 T$	$2n_{\text{layer}} d_{\text{token}} n_{\text{o}} T$
Feedforward	$8n_{\text{layer}} d_{\text{model}}^2$	$2n_{\text{layer}} d_{\text{token}} n_{\text{ff}}$	$16n_{\text{layer}} d_{\text{model}}^2 T$	$4n_{\text{layer}} d_{\text{token}} n_{\text{ff}} T$
De-embed	-	-	$2n_{\text{vocab}} d_{\text{model}}$	$2n_{\text{vocab}} d_{\text{model}}$
Total (Non-Embedding)	$N = 12n_{\text{layer}} d_{\text{model}}^2$	$N = n_{\text{layer}} d_{\text{token}} (n_{\text{q}} + n_{\text{k}} + n_{\text{v}} + n_{\text{o}} + 2n_{\text{ff}})$	$2NT + 4n_{\text{layer}} d_{\text{model}} T^2$	$2NT + 4n_{\text{layer}} d_{\text{token}} T^2$

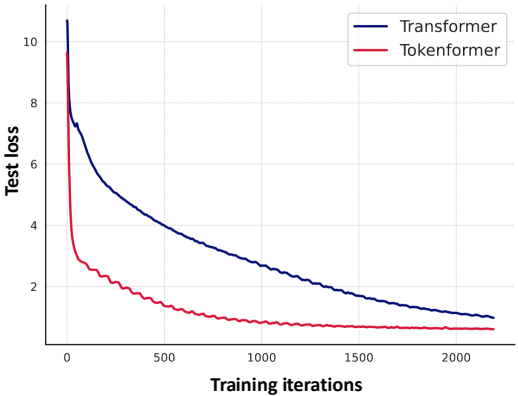
Parameter counts and training compute estimates for Transformer and Tokenformer.

Experiments—Comparison with Transformer

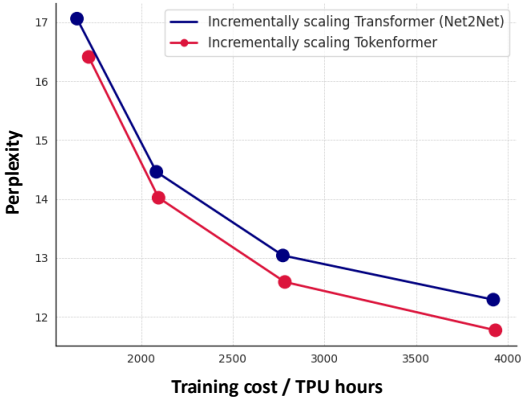


The relationship between FLOPs and text length for both Transformer and Tokenformer.

Experiments—Comparison with Transformer



Loss curves comparing pre-trained Transformer and Tokenformer as their parameters are scaled during continued training on enwik8.



Performance benchmarking on incremental model scaling between Transformer with Net2Net scheme and Tokenformer.

Future Work

- **Advancing Parameter-Efficient Tuning.** When confronted with new tasks or datasets, the model can augment its pre-trained parameters by incorporating these new parameter tokens, thereby adapting to specific task requirements quickly.
- **Integrating Vision and Language Models.** Unify the key-value parameter tokens derived from pre-trained visual Tokeformer and language Tokenformer into a single parameter set. Then, introduce new learnable tokens to perform vision-language alignment and instruction tuning.