TokenFormer: Rethinking Transformer Scaling with Tokenized Model Parameters

Haiyang Wang[1,3], Yue Fan[1], Muhammad Ferjad Naeem[2], Yongqin Xian[2], Jan Eric Lenssen[1], Liwei Wang[3], Federico Tombari[2], Bernt Schiele[1]

[1]Max Planck Institute for Informatics [2]Google [3]Peking University

Large Scale Training

- Simple Architecture + Large Model Size (param) + Large Dataset => good performance!
  - Often outperforms the most complex algorithm
- Scale to Larger Dataset?
  - Fine-tuning
- Scale to Larger Model?
  - Retraining => incurs extremely high costs!

Existing Solutions: Model Reuse

- Why not reuse existing model
  - Initialize larger models with pre-trained smaller models by
    - Duplication
    - Stacking
    - Combining model weights
- Cons
  - Risks Losing pre-trained knowledge and slowing convergence

TokenFormer

- Allows for parameter scaling in a natural and seamless manner while preserving the integrity of the existing model.

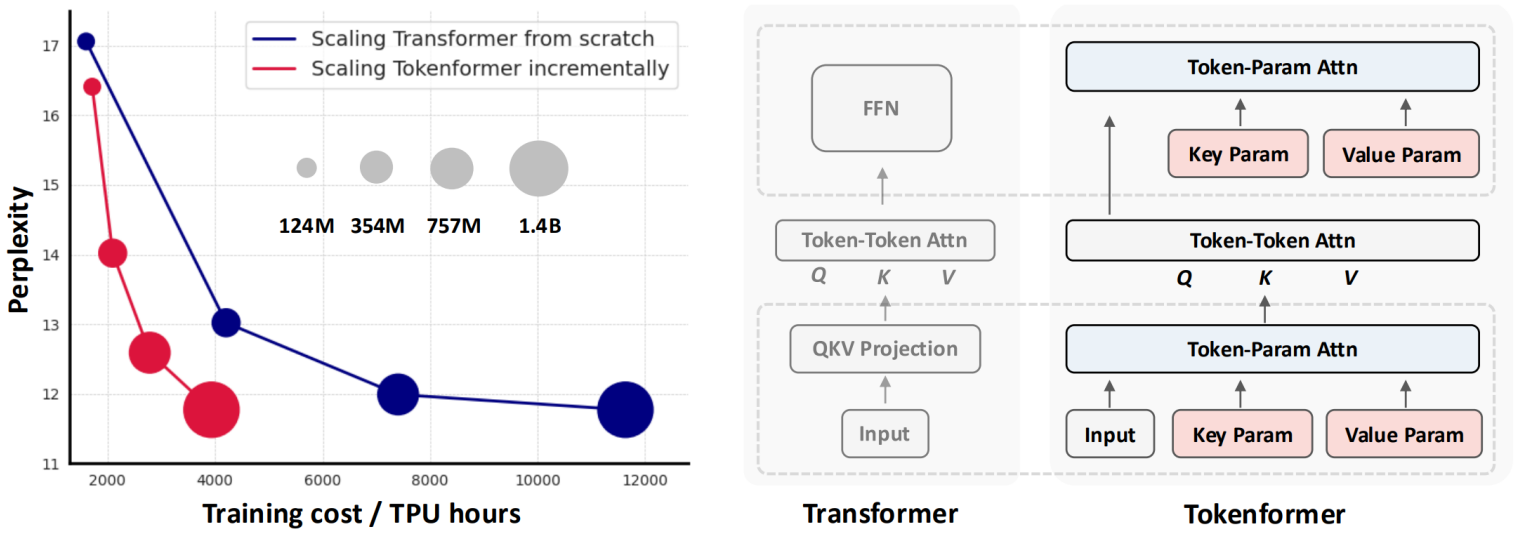


Figure 1: Traditionally, large transformer architectures are trained from scratch without reusing previous smaller-scale models (represented by blue dots on the left). In this paper, we propose a novel fully attention-based architecture that allows scaling model incrementally, thus greatly reducing the overall cost of training large transformer architectures (depicted by red dots on the left). The right panel delineates a comparison between conventional Transformer and our Tokenformer.

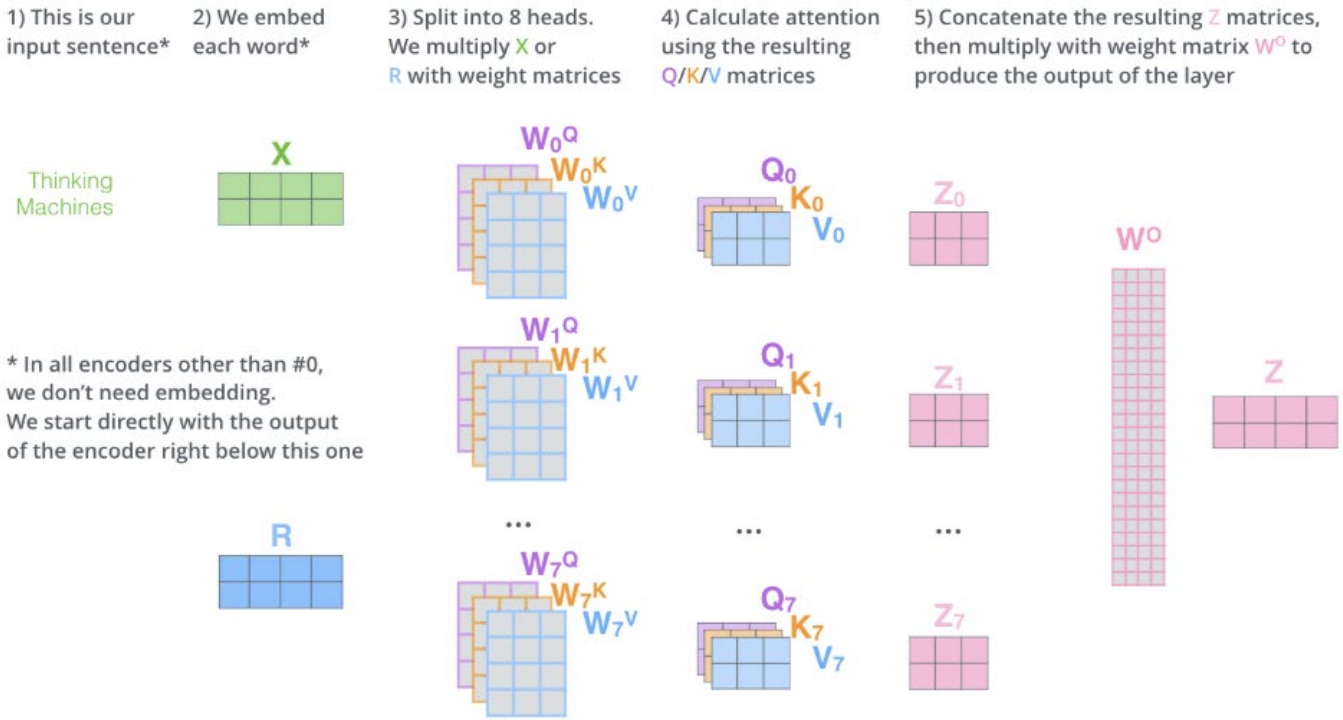
Methodology

- Token-Parameter Attention Layer (Pattention Layer)
- Progressive Model Scaling

Transformer (Original)

- Linear Projection limits scalability!

$$Q = X \cdot W^Q, \qquad K = X \cdot W^K, \qquad V = X \cdot W^V,$$



Token-**P**arameter **Attention** Layer (Pattention Layer)

- Treat parameters as tokens

$$\text{Pattention}(X, K_P, V_P) = \Theta\left(X \cdot K_P^\top\right) \cdot V_P,$$

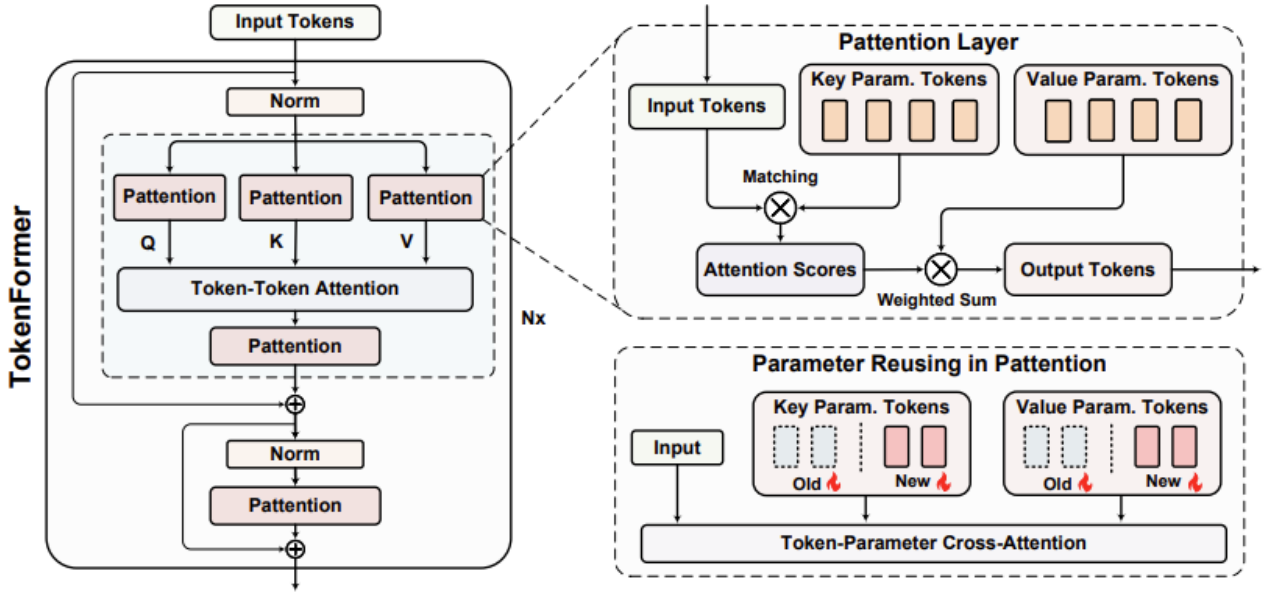


Figure 2: Tokenformer is a fully attention-driven architecture featuring a new token-**P**arameter **attention** (Pattention) layer. The Pattention uses a set of learnable tokens to represent model parameters and lets the input tokens attend to them. As the model scales, Tokenformer adds new learnable tokens to expand the existing key-value parameter sets, while keeping the feature dimension constant and leaving the rest of the computation unaffected.

Token-**P**arameter **Attention** Layer (Pattention Layer)

- Θ is a modified softmax operation
- f is the GeLU function

$$\text{Pattention}(X, K_P, V_P) = \Theta\left(X \cdot K_P^\top\right) \cdot V_P,$$

$$S_i = \frac{\exp(A_i/\sqrt{d})}{\sum_{j=1}^{n} \exp(A_j/\sqrt{d})}, \quad \forall\, i \in 1...n,$$

Softmax Function

$$\hat{S}_i = f(Z_i) = f\left(\frac{A_i \times \sqrt{n}}{\sqrt{\sum_{j=1}^{n} |A_j|^2}}\right), \quad \forall\, i \in 1...n,$$

Modified Softmax Function

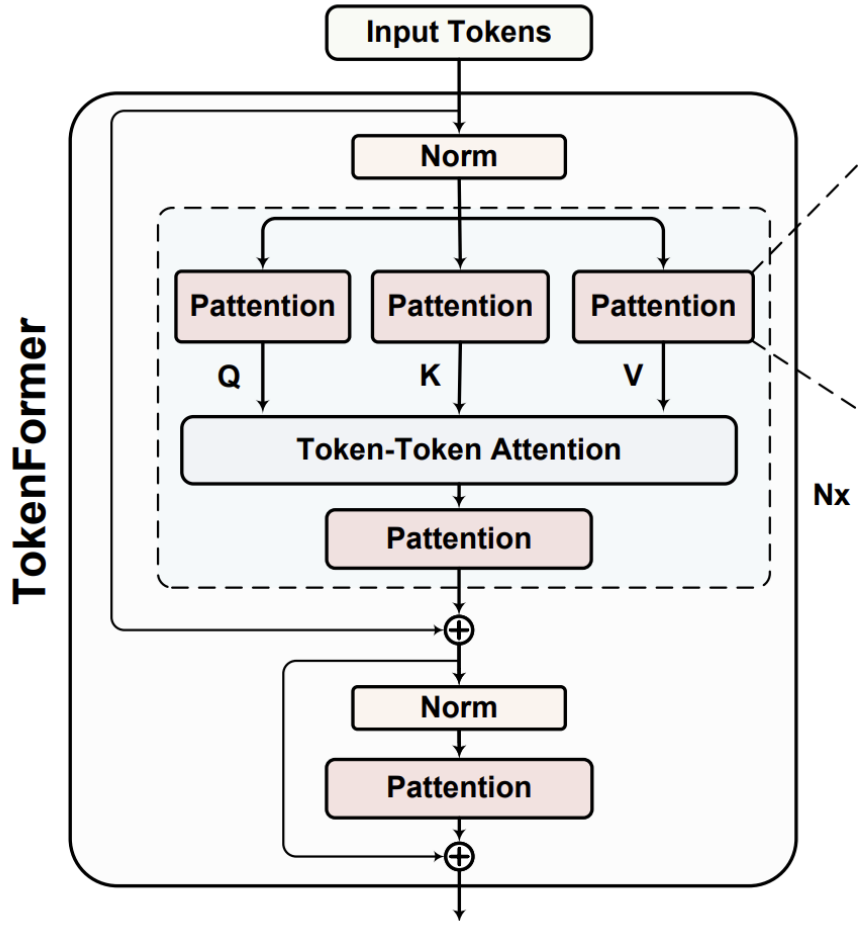
Applying Pattention Layer

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

$$X_{\text{att}} = \text{softmax}\left[\frac{Q \cdot K^\top}{\sqrt{d}}\right] \cdot V,$$

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

$$O_{\text{ffn}} = \text{Pattention}\left(X_{\text{ffn}}, K_P^{\text{ffn}}, V_P^{\text{ffn}}\right),$$

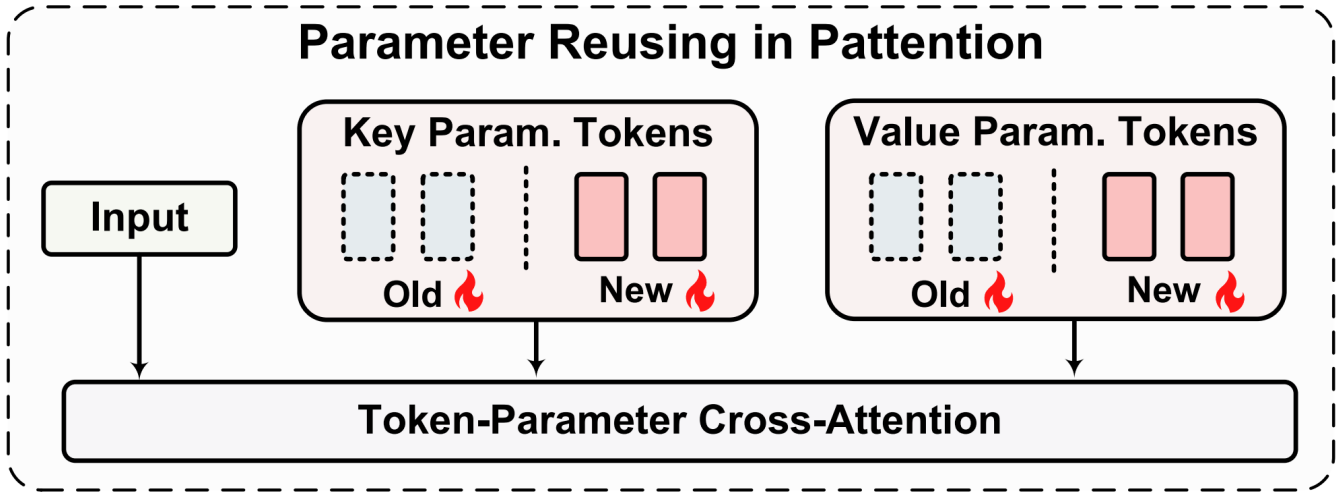


Progressive Model Scaling

- Progressive Model Scaling is done by concatenating new parameter tokens to old
  - New tokens are initialized at 0, model can perfectly resume the model state from the pre-training phase

$$K_P^{\text{scale}} = \left[K_P^{\text{old}}, K_P^{\text{new}}\right], \quad V_P^{\text{scale}} = \left[V_P^{\text{old}}, V_P^{\text{new}}\right],$$

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



Experiments

- Continual Expansion Capability
- Efficacy (Vision / Language)
- Comparison vs Transformer
- Ablation Study

Progressive Model Scaling

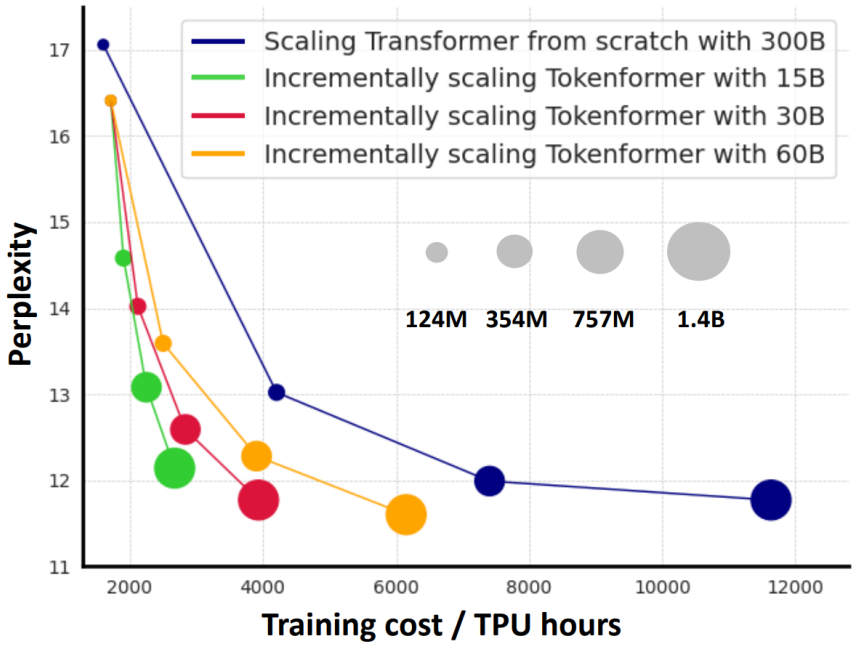


Figure 3: Evaluating model scaling costs through cumulative computational budgets. The Transformer baseline incurs expenses for each individual scaling step performed independently from scratch, whereas Tokenformer aggregates costs across all scaling stages, including training a 124M model initially, progressively scaling to 354M, 757M, and 1.4B parameters.

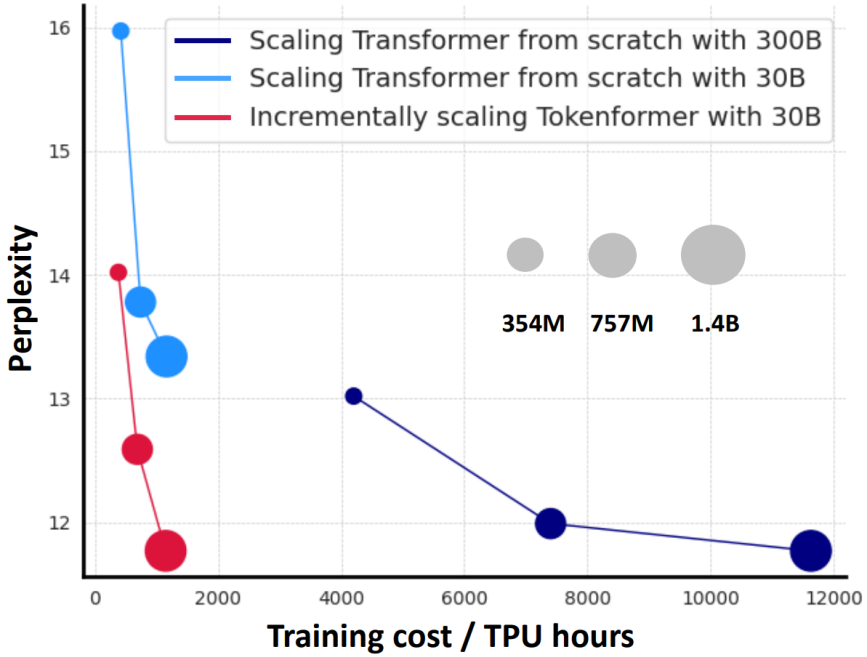


Figure 4: Evaluating model scaling costs by measuring the budget required at each scaling stage. The Transformer baselines used are consistent with those depicted in Figure 3, trained with 30B and 300B tokens. Similarly, for Tokenformer, the cost is the budget required for each incremental scaling step from a smaller one. All the experiments were conducted on TPU v4 hardware.

Efficacy Language

Model	#Param	Pile ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc ↑	WinoGrande acc ↑	Average acc ↑
Pythia-160M (Biderman et al., 2023)	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 (Biderman et al., 2023)	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 (Biderman et al., 2023)	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 (Black et al., 2021)	1.3B	-	7.50	57.2	48.9	71.1	56.2	25.9	54.9	52.4
OPT-1.3B (Zhang et al., 2022)	1.3B	-	6.64	58.0	53.7	72.4	56.7	29.6	59.5	55.0
Pythia-1.3B (Biderman et al., 2023)	1.3B	7.51	6.08	61.7	52.1	71.0	60.5	28.5	57.2	55.2
Ours (TokenFormer-1.5B)	1.5B	**6.91**	5.24	**64.7**	60.0	**74.8**	**64.8**	32.0	59.7	**59.3**

Table 1: (**Zero-shot Evaluations.**) The best performance for each model size is highlighted in bold. Our comparisons are made with publicly available transformer-based LMs with various tokenizers. Following Pythia (Biderman et al., 2023), our model is trained for up to 300B tokens on pile dataset.

Efficacy Image Classification

Method	Image Size	#Param	Top-1 acc
ViT-B/16 (Dosovitskiy et al., 2021)	384^2	86M	77.9
DeiT-B/16 (Touvron et al., 2021)	224^2	86M	81.8
ViT-B/16 (MAE) (He et al., 2022)	224^2	86M	82.3
Ours-B/16†	224^2	86M	82.1
Ours-B/16	224^2	109M	**82.5**
ViT-L/16 (Dosovitskiy et al., 2021)	384^2	307M	76.5
ViT-L/16 (MAE) (He et al., 2022)	224^2	307M	82.6
Ours-L/16†	224^2	307M	83.0
Ours-L/16	224^2	407M	**83.1**

Table 2: (**Image Classification.**) Comparison of standard vision transformer on ImageNet-1K. The training hyperparameters are completely consistent (batch size, learning rate, etc.) with He et al. (2022). † denotes models where the parameter size has been matched to that of the standard ViT.

Comparison vs Transformer

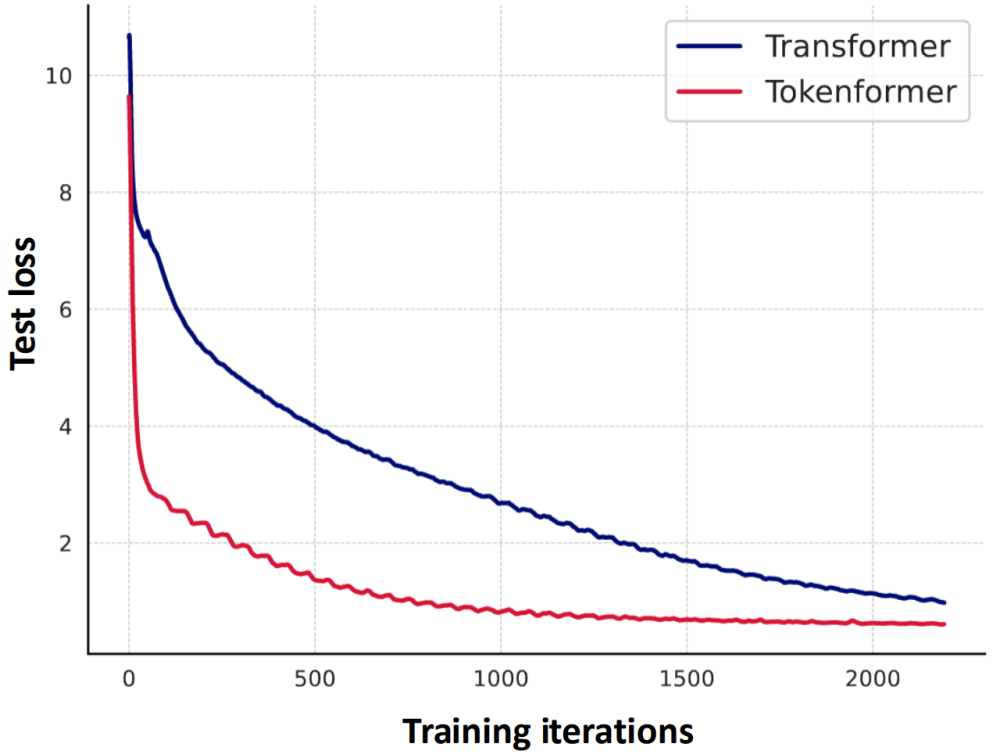


Figure 6: Loss curves comparing pre-trained Transformer and Tokenformer as their parameters are scaled during continued training on enwik8.

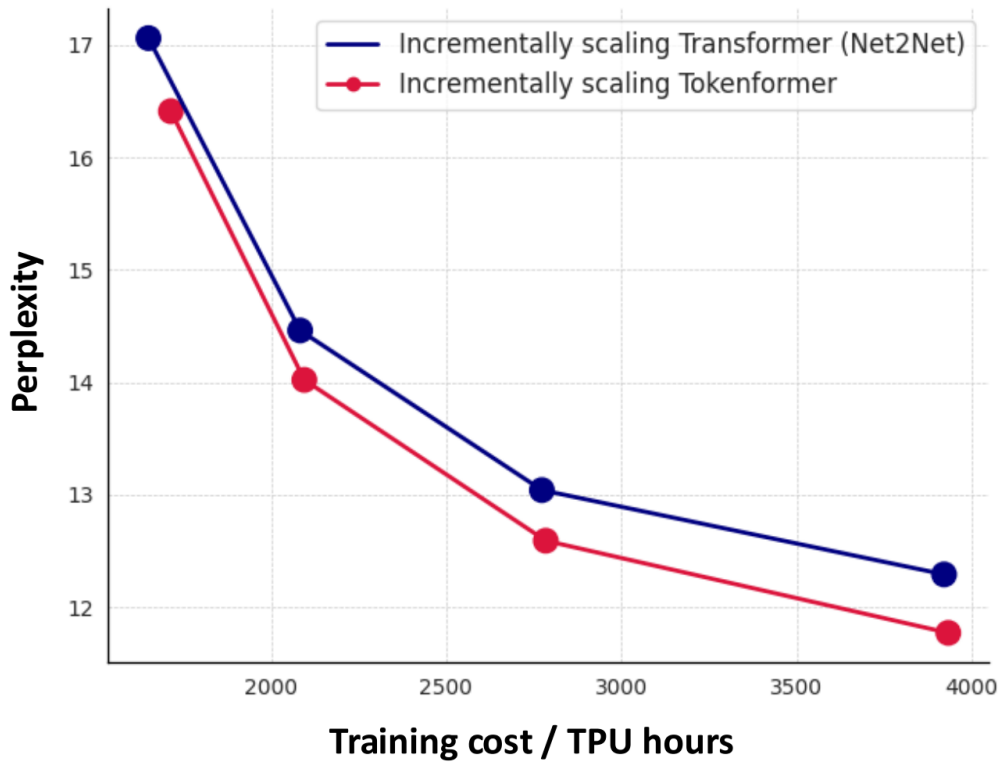


Figure 7: Performance benchmarking on incremental model scaling between Transformer with Net2Net scheme and our Tokenformer.

Ablation Study

Nonlinear Function	Normalization	Top-1 acc
e^x	L_1 Norm	79.6
GeLU	L_1 Norm	81.7
GeLU	L_2 Norm	82.5

Table 4: Ablation of Softmax part on ImageNet classification with base model.

Learnable Weight (γ)	Learnable Bias (β)	Top-1 acc
✓	✓	82.6
-	✓	82.5
-	-	82.5

Table 5: Ablation of non-parametric layer normalization on ImageNet classification with base model.

Conclusion

- TokenFormer
  - Leverages the attention mechanism to facilitate interaction between tokens and model parameters
  - All linear projection layers are replaced with the **pattention layer**.
  - TokenFormer offers great flexibility than traditional transformers