Singularformer: Learning to Decompose Self-Attention to Linearize the Complexity of Transformer

Yifan Wu, **Shichao Kan**, **Min Zeng** and **Min Li***

The School of Computer Science and Engineering, Central South University, Changsha, China

{wuyifan, kanshichao, zengmin}@csu.edu.cn

Abstract

Transformers achieve excellent performance in a variety of domains since they can capture long-distance dependencies through the self-attention mechanism. However, self-attention is computationally costly due to its quadratic complexity and high memory consumption. In this paper, we propose a novel Transformer variant (Singularformer) that uses neural networks to learn the singular value decomposition process of the attention matrix to design a linear-complexity and memory-efficient global self-attention mechanism. Specifically, we decompose the attention matrix into the product of three matrix factors based on singular value decomposition and design neural networks to learn these matrix factors, then the associative law of matrix multiplication is used to linearize the calculation of self-attention. The above procedure allows us to compute self-attention as two-dimensional reduction processes in the first and second token dimensional spaces, followed by a multi-head self-attention computational process on the first dimensional reduced token features. Experimental results on 8 real-world datasets demonstrate that Singularformer performs favorably against the other Transformer variants with lower time and space complexity. Our source code is publicly available at https://github.com/CSUBioGroup/Singularformer.

1 Introduction

Transformers [Vaswani *et al.*, 2017] are a class of powerful and efficient deep learning models that achieved excellent performance in many fields, including natural language processing [Kenton and Toutanova, 2019; Liu *et al.*, 2019], computer vision [Dosovitskiy *et al.*, 2020; Liu *et al.*, 2021], bioinformatics [Jumper *et al.*, 2021; Zhang *et al.*, 2021]. Compared to CNN/RNN, Transformers have superior feature capture capabilities, particularly for long-distance dependencies. However, to capture the features of the long-distance dependencies, Transformers need to learn an $n \times n$ (n is equal to the input length) attention matrix $\boldsymbol{A}$ to characterize the relationship between any two tokens in the input sequence. However, the attention matrix $\boldsymbol{A}$ leads to a quadratic time and space complexity in computing. The quadratic complexity limits the computational efficiency and the performance of Transformers. To train or fine-tune a Transformer-based model, researchers have to limit the length of the input sequence, use a large number of TPUs or GPUs, and spend lots of time. For example, pre-training the $\text{BERT}_{\text{LARGE}}$ model on 16 cloud TPUs (64 TPU chips total) takes 4 days while the sequence length is restricted to 512 [Kenton and Toutanova, 2019]. Therefore, it is crucial to optimize the computational efficiency and reduce the complexity of Transformers.

At present, the core idea of speeding up Transformers is to reduce redundant information in the $n \times n$ attention matrix $\boldsymbol{A}$. Actually, we seldom require such a precise or complete attention $\boldsymbol{A}$ which represents the relationship between tokens. It is sufficient to describe the relationship between tokens and extract the semantic information for downstream tasks using a moderately precise or complete attention matrix. Recent works focus on computing the attention matrix $\boldsymbol{A}$ approximately [Katharopoulos *et al.*, 2020; Choromanski *et al.*, 2020; Shen *et al.*, 2021; Xiong *et al.*, 2021; Wang *et al.*, 2020] or introducing sparsity into the attention matrix $\boldsymbol{A}$ [Child *et al.*, 2019; Beltagy *et al.*, 2020; Zaheer *et al.*, 2020; Kitaev *et al.*, 2020]. However, these methods still have some limitations. First, they usually sacrifice some precision or stability in exchange for reducing complexity. Krzysztof et al. pointed out Reformer and Linformer significantly reduce the classification accuracy on the protein dataset [Choromanski *et al.*, 2020]. Additionally, the squared ReLU [So *et al.*, 2021] used in FLASH [Hua *et al.*, 2022] leads to unstable model training [Ma *et al.*, 2022]. Second, the reduction of space and time complexity is only marginally efficient while solving some short sequences. BigBird [Zaheer *et al.*, 2020] only shows the advantages of memory consumption while processing sequences with length over 700. Third, some methods need highly sophisticated implementations that introduce additional hyper-parameters that need to be tuned. For example, Transformer-LS [Zhu *et al.*, 2021] introduces the local window segment size and the rank of dynamic projection to adjust the attention intensity. Last, most of them are intuitive and lack solid theoretical foundation.

In this paper, we propose a linear Transformer vari-

*Corresponding Author

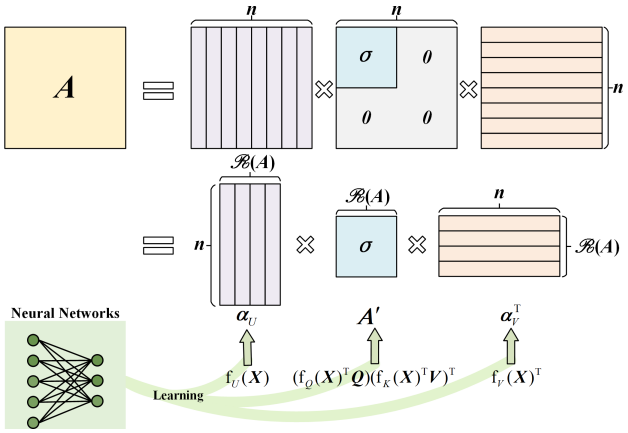

Figure 1: Basic idea: learn to decompose the attention matrix $\boldsymbol{A}$ to accelerate self-attention by using neural networks.

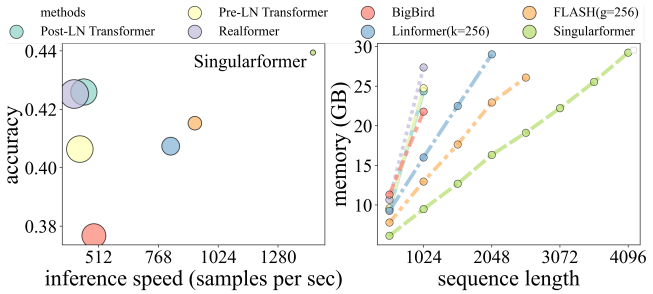

Figure 2: Comparison of Singularformer with other methods on MIMIC-III 50 dataset: (left) accuracy (vertical axis), inference speed (horizontal axis), and memory consumption (size of the circles) of different methods. (right) memory consumption versus sequence lengths of different methods.

ant called Singularformer that addresses the above limitations by being performance-stable, linear-complexity, easy-to-implement, and theory-supported. Specifically, our theoretical analysis shows that the $n \times n$ attention matrix $\boldsymbol{A}$ is singular and can be decomposed into linear combinations of the cross products of the singular vectors weighted by its singular values. The singularity of $\boldsymbol{A}$ indicates that many singular vectors are weighted by zero singular values and can be filtered out. As a result, we decompose $\boldsymbol{A}$ into the product of three matrix factors $\boldsymbol{\alpha}_U, \boldsymbol{A}', \boldsymbol{\alpha}_V^{\mathrm{T}}$ without loss of accuracy using the singular value decomposition (SVD) theorem. Considering that direct decomposition $\boldsymbol{A}$ requires a lot of computation [Xiong *et al.*, 2021], we therefore leverage neural networks $(\mathrm{f}_U(\boldsymbol{X}), \mathrm{f}_Q(\boldsymbol{X}), \mathrm{f}_K(\boldsymbol{X}), \mathrm{f}_V(\boldsymbol{X}))$ to learn this process[1], as shown in Figure 1.

In addition, we record the accuracy versus inference speed and GPU memory consumption versus sequence lengths (with automatic mixed precision [Micikevicius *et al.*, 2018], batch size of 64, hidden size of 384, 6 layers, 6 heads) for different Transformer variants on the MIMIC-III 50 dataset, as shown in Figure 2. From Figure 2(left), we can see that Singularformer has the highest inference speed while maintaining the best accuracy. From Figure 2(right), we can see that Singularformer has lower memory consumption than the other methods as the sequence length increases. In the case of sequence length reaching 1024, the memory consumption of Singularformer is 26.9%, 40.7%, 56.4% lower than FLASH, Linformer, BigBird, respectively. Additionally, on a single NVIDIA TESLA V100 32GB card, Singularformer can train 64 sequences with the length of 4096 in parallel. These results show that Singularformer significantly outperforms other Transformers in both inference speed and memory consumption while maintaining high performance.

The main contributions of this work are summarized as follows:

- We propose a novel Transformer model called Singularformer based on the SVD theorem. Singularformer does not introduce additional tunable hyper-parameters and achieves lower space-time complexity and faster inference speed compared to other linear Transformers.
- Singularformer is built on a rigorous theoretical analysis, with detailed theoretical derivation and space-time complexity analysis proving its advantages. Both the theoretical analysis and experimental results show that Singularformer almost has no accuracy loss when compared with the standard Transformer.
- Extensive experiments on 8 real-world datasets show that Singularformer not only achieves comparable performance, but also has extremely low space and time consumption. Compared to the standard Transformer and 5 Transformer variants, Singularformer achieves favorable performances in terms of accuracy, memory consumption and inference speed.

2 Backgrounds and Related Work

2.1 Transformer

Since 2017, Transformers [Vaswani *et al.*, 2017] have been applied in an increasing number of tasks with amazing results. These achievements are inseparable from the self-attention layer and the feed-forward network layer. The self-attention layer provides a message passing mechanism between different parts of the input, while the feed-forward network layer offers a powerful feature learning capability. In short, the self-attention layer is based on Formulas 1 to 3:

$$\boldsymbol{A}^i = \mathrm{softmax}(\frac{\boldsymbol{Q}^{i\mathrm{T}}\boldsymbol{K}^i}{\sqrt{\frac{d}{m}}}) \tag{1}$$

$$\boldsymbol{H}^i = \boldsymbol{A}^i\boldsymbol{V}^i \tag{2}$$

$$\boldsymbol{Z} = \mathrm{concat}(\boldsymbol{H}^1, \boldsymbol{H}^2, ..., \boldsymbol{H}^m)\boldsymbol{W}_Z + \boldsymbol{b}_Z \tag{3}$$

where $\boldsymbol{Q}^i, \boldsymbol{K}^i, \boldsymbol{V}^i \in \mathbb{R}^{n\times\frac{d}{m}}$ are the query, key, value vectors for head i, n is the length of $\boldsymbol{x}$, d is the hidden size, m is the number of head, $\boldsymbol{W}_Z \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_Z \in \mathbb{R}^{1\times d}$ are learnable weight matrix and bias. Additionally, the feed-forward network layer is to further extract information from $\boldsymbol{Z}$ through

[1]This is different from Nystromformer because it is based on CUR decomposition and implemented by non-learnable iterative methods.

some linear layers with non-linear activation functions. We can see that the quadratic complexity of n mainly comes from Formulas 1 and 2. Optimizing the two formulas is the key to developing efficient Transformer variants.

2.2 Related Work

In recent years, lots of efficient Transformer variants have been proposed. For convenience, we summarize formulas 1 and 2 to $\mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})\boldsymbol{V}$. Shen et al. [Shen *et al.*, 2021] split the $\mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})$ into $\mathrm{softmax}(\boldsymbol{Q})\mathrm{softmax}(\boldsymbol{K}^{\mathrm{T}})$ to get an approximation, and then computed $\mathrm{softmax}(\boldsymbol{K}^{\mathrm{T}})\boldsymbol{V}$ first, where the front $\mathrm{softmax}$ normalizes $\boldsymbol{Q}$ along its second dimension (d), the back $\mathrm{softmax}$ normalizes $\boldsymbol{K}$ along its first dimension (n). In this way, the complexity can be reduced from $O(n^2d)$ to $O(nd^2)$. Katharopoulos et al. [Katharopoulos *et al.*, 2020] approximated $\mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})V$ by $(\mathrm{elu}(\boldsymbol{Q})+1)(\mathrm{elu}(\boldsymbol{K})+1)\boldsymbol{V}$, and calculated the product of the last two terms to reduce the complexity. Similarly, Choromanski et al. [Choromanski *et al.*, 2020] disassembled $\mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}})$ based on the random projection approximation. These variants approximate the Formula 1 using two separate terms, but often bring some loss of accuracy. Wang et al. [Wang *et al.*, 2020] introduced two learnable projection matrices to decompose $\boldsymbol{V}$ and $\boldsymbol{K}$, and implemented a linear-complexity Transformer. However, the learned projection matrices are inflexible to some complex datasets. Child et al. [Child *et al.*, 2019] proposed the sparse self-attention layer which integrates atrous attention and local attention into the self-attention layer. To reduce the quadratic complexity to linear, Zaheer et al. [Zaheer *et al.*, 2020] designed a sparse attention mechanism made up of global attention, sliding attention and random attention. These two methods tend to superpose multiple different sparse patterns, but because of bringing additional computation, they have poor optimization results for short sequences. Nikita et al. [Kitaev *et al.*, 2020] applied local sensitive hash [Gionis *et al.*, 1999] and reversible feed forward network to reduce complexity, which is very complex and requires to reimplement the gradient back propagation. Moreover, Hua et al. [Hua *et al.*, 2022] proposed a linear variant called FLASH, which applies the gated linear unit into self-attention layer and uses the mixed chunk attention to accelerate computing. But some researchers pointed out that training FLASH is unstable [Ma *et al.*, 2022]. In summary, each of these variations has its limitations and often lacks solid theoretical foundation.

3 Methodology

3.1 Singularformer

Singularformer implements a linear Transformer variant (as shown in Figure 3) by applying the attention mechanism to $\boldsymbol{A}^i$'s singular vectors whose corresponding singular value is not zero. The core idea of Singularformer is based on the following facts:

Fact 1: The attention matrix $\boldsymbol{A}^i$ (obtained by the Formula 1) is singular or non-full-rank. And the upper bound on the rank of $\boldsymbol{A}^i$ is $\frac{d}{m}$.

Proof of the Fact 1: According to the rank theorem for matrix products, the rank of the products is lower or equal to the ranks of the matrix factors. Noticing that $\boldsymbol{A}^i = {\boldsymbol{Q}^i}^{\mathrm{T}}\boldsymbol{K}^i$, we can get:

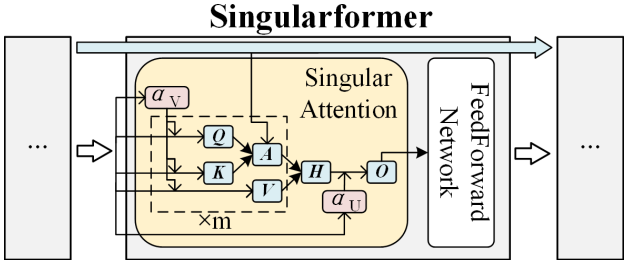

Figure 3: Illustration of Singularformer. From an intuitive perspective, we use $\boldsymbol{\alpha}_V$ to merge the n tokens of input into r tokens ($r << n$), and use $\boldsymbol{\alpha}_U$ to unfold the r tokens back to n tokens. While from a theoretical perspective, $\boldsymbol{\alpha}_U$ and $\boldsymbol{\alpha}_V$ are the singular vector matrices of the original attention matrix, and this is essentially the SVD process of the original attention matrix.

$$\mathscr{R}(\boldsymbol{A}^i) \leq \min\{\mathscr{R}(\boldsymbol{Q}^i), \mathscr{R}(\boldsymbol{K}^i)\} \tag{4}$$

where $\mathscr{R}(\cdot)$ is the rank of a matrix. And the $\boldsymbol{Q}^i, \boldsymbol{K}^i$ are all in $\mathbb{R}^{n\times\frac{d}{m}}$. For most cases, $\frac{d}{m}$ is less than n. Therefore, we can derive that:

$$\min\{\mathscr{R}(\boldsymbol{Q}^i), \mathscr{R}(\boldsymbol{K}^i)\} \leq \frac{d}{m} \tag{5}$$

$$\Rightarrow \mathscr{R}(\boldsymbol{A}^i) \leq \frac{d}{m} \tag{6}$$

Fact 2: The attention matrix $\boldsymbol{A}^i$ can be decomposed into $\boldsymbol{\alpha}_U{\boldsymbol{A}^i}'\boldsymbol{\alpha}_V^{\mathrm{T}}$, where $\boldsymbol{\alpha}_U \in \mathbb{R}^{n\times\mathscr{R}(\boldsymbol{A}^i)}$ is an orthogonal matrix with the left singular vector of A^i as the column vector, $\boldsymbol{\alpha}_V^{\mathrm{T}} \in \mathbb{R}^{\mathscr{R}(\boldsymbol{A}^i)\times n}$ is an orthogonal matrix with the right singular vector of $\boldsymbol{A}^i$ as the row vector, ${\boldsymbol{A}^i}' \in \mathbb{R}^{\mathscr{R}(\boldsymbol{A}^i)\times\mathscr{R}(\boldsymbol{A}^i)}$ is a diagonal matrix formed by the singular values of $\boldsymbol{A}^i$.

Proof of the Fact 2: According to the SVD theorem, any matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ must have a singular value decomposition $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ with singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ and $r = \mathscr{R}(\boldsymbol{M})$. As a result, for $\boldsymbol{A}^i \in \mathbb{R}^{n\times n}$, we have:

$$\begin{aligned}\boldsymbol{A}^i &= \begin{pmatrix}\boldsymbol{u}_1 & ... & \boldsymbol{u}_n\end{pmatrix}\begin{pmatrix}\sigma_1 & & \\ & ... & \\ & & \sigma_r \\ & & \end{pmatrix}\begin{pmatrix}\boldsymbol{v}_1 & ... & \boldsymbol{v}_n\end{pmatrix}^{\mathrm{T}} \\ &= \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}} + ... + \sigma_r\boldsymbol{u}_r\boldsymbol{v}_r^{\mathrm{T}} + 0\boldsymbol{u}_{r+1}\boldsymbol{v}_{r+1}^{\mathrm{T}} + ... + 0\boldsymbol{u}_n\boldsymbol{v}_n^{\mathrm{T}} \\ &= \begin{pmatrix}\boldsymbol{u}_1 & ... & \boldsymbol{u}_r\end{pmatrix}\begin{pmatrix}\sigma_1 & & \\ & ... & \\ & & \sigma_r\end{pmatrix}\begin{pmatrix}\boldsymbol{v}_1 & ... & \boldsymbol{v}_r\end{pmatrix}^{\mathrm{T}}\end{aligned} \tag{7}$$

Here, $\boldsymbol{u}_i, \boldsymbol{v}_i \in \mathrm{R}^{n\times 1}$ are the singular vectors, $r = \mathscr{R}(\boldsymbol{A}^i)$ is the number of nonzero singular values. Finally, we write the $\begin{pmatrix}\boldsymbol{u}_1 & ... & \boldsymbol{u}_r\end{pmatrix}$ as $\boldsymbol{\alpha}_U$, $\begin{pmatrix}\sigma_1 & & \\ & ... & \\ & & \sigma_r\end{pmatrix}$ as ${\boldsymbol{A}^i}'$, $\begin{pmatrix}\boldsymbol{v}_1 & ... & \boldsymbol{v}_r\end{pmatrix}^{\mathrm{T}}$ as $\boldsymbol{\alpha}_V^{\mathrm{T}}$ to get $\boldsymbol{A}^i = \boldsymbol{\alpha}_U{\boldsymbol{A}^i}'\boldsymbol{\alpha}_V^{\mathrm{T}}$.

Singular attention layer. ***Fact 1*** and ***Fact 2*** inspire us to design a new computation mode for Formulas 1 and 2:

$$\boldsymbol{H}^i = \boldsymbol{\alpha}_U \boldsymbol{A}^{i'} \boldsymbol{\alpha}_V^{\mathrm{T}} \boldsymbol{V}^i \tag{8}$$

where $\boldsymbol{\alpha}_U \in \mathbb{R}^{n\times r}$, $\boldsymbol{A}^{i'} \in \mathbb{R}^{r\times r}$, $\boldsymbol{\alpha}_V^{\mathrm{T}} \in \mathbb{R}^{r\times n}$, r is the rank of $\boldsymbol{A}^i$, i.e. $\mathscr{R}(\boldsymbol{A}^i)$. By this computation mode, if we calculate it as the following order $\boldsymbol{\alpha}_U(\boldsymbol{A}^{i'}(\boldsymbol{\alpha}_V^{\mathrm{T}}\boldsymbol{V}^i))$, the complexity will be in $O(2\frac{rd}{m}n + r^2\frac{d}{m})$. In addition, ***Fact 1*** tells us $r = \mathscr{R}(\boldsymbol{A}^i) \leq \frac{d}{m}$, therefore, the complexity can be written as $O(2(\frac{d}{m})^2 n + (\frac{d}{m})^3)$, which is in linear time about n.

The next problem is how to determine $\boldsymbol{\alpha}_U, \boldsymbol{A}^{i'}, \boldsymbol{\alpha}_V^{\mathrm{T}}$. An intuitive idea is to use the neural network to learn these representations. In particular, given that the original $\boldsymbol{A}^i$ is computed from $\boldsymbol{Q}^i, \boldsymbol{K}^i$, or the embedded sequence input $\boldsymbol{X}$ with $\mathrm{softmax}$ activation, we can also assume that the $\boldsymbol{\alpha}_U, \boldsymbol{A}^{i'}, \boldsymbol{\alpha}_V^{\mathrm{T}}$ are all computed from them. We define:

$$\boldsymbol{A}^{i'} = \mathrm{softmax}((\boldsymbol{\alpha}_Q^{\mathrm{T}}\boldsymbol{Q}^i)(\boldsymbol{\alpha}_K^{\mathrm{T}}\boldsymbol{K}^i)^{\mathrm{T}}) \tag{9}$$

where $\boldsymbol{\alpha}_Q^{\mathrm{T}}, \boldsymbol{\alpha}_K^{\mathrm{T}} \in \mathbb{R}^{r\times n}$, $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is the embedded input sequence, $(\cdot)^i$ is to take the $(i-1)\frac{d}{m} \sim i\frac{d}{m}$ columns. Then, we utilize four learnable layers of $\boldsymbol{X}$ to fit $\boldsymbol{\alpha}_U, \boldsymbol{\alpha}_Q^{\mathrm{T}}, \boldsymbol{\alpha}_K^{\mathrm{T}}, \boldsymbol{\alpha}_V^{\mathrm{T}}$:

$$\boldsymbol{\alpha}_U = \mathrm{softmax}(\mathrm{f}_U(\boldsymbol{X})) \tag{10}$$
$$\boldsymbol{\alpha}_Q^{\mathrm{T}} = \mathrm{softmax}(\mathrm{f}_Q(\boldsymbol{X})^{\mathrm{T}}) \tag{11}$$
$$\boldsymbol{\alpha}_K^{\mathrm{T}} = \mathrm{softmax}(\mathrm{f}_K(\boldsymbol{X})^{\mathrm{T}}) \tag{12}$$
$$\boldsymbol{\alpha}_V^{\mathrm{T}} = \mathrm{softmax}(\mathrm{f}_V(\boldsymbol{X})^{\mathrm{T}}) \tag{13}$$

Among them, $\mathrm{f}_U(\cdot), \mathrm{f}_Q(\cdot), \mathrm{f}_K(\cdot), \mathrm{f}_V(\cdot)$ are the projection layers to transform the $n \times d$ matrix $\boldsymbol{X}$ into matrices with shape $n \times r$.

In our implementation, we share the four linear layers[1] $\mathrm{f}_U(\cdot), \mathrm{f}_Q(\cdot), \mathrm{f}_K(\cdot), \mathrm{f}_V(\cdot)$ by introducing a set of linear-layer parameters $\boldsymbol{W}_a \in \mathbb{R}^{d\times r}, \boldsymbol{b}_a \in \mathbb{R}^{1\times r}$. We define:

$$\mathrm{f}_U(\boldsymbol{X}) = \mathrm{f}_Q(\boldsymbol{X}) = \mathrm{f}_K(\boldsymbol{X}) = \mathrm{f}_V(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W}_a + \boldsymbol{b}_a \tag{14}$$

Finally, we can compute $\boldsymbol{Z}$ by the Formula 3.

Value of the parameter r. In order to avoid adding adjustable parameters, it is a good choice to fix r to a fixed value. According to ***Fact 1***, we know that $r \leq \frac{d}{m}$. When $\boldsymbol{Q}^i, \boldsymbol{K}^i$ are of full rank, we have $r = \frac{d}{m}$. Otherwise, we can use a smaller r if $\boldsymbol{Q}^i, \boldsymbol{K}^i$ are also singular. Instead of exploring the lower limit of r, it is better to decrease d or increase m. Therefore, we fix the value of r in $\frac{d}{m}$ in Singularformer.

Feed-forward network layer and multi-layer Singularformer. The feed-forward network layer of Singularformer is similar to the standard Transformer. In the multi-layer Singularformer implementation, we introduce the residual addition from the previous layer for $\boldsymbol{A}^{i'}$ by borrowing the idea of Realformer [He *et al.*, 2021] to improve the convergence performance.

Orthogonal constraint and diagonal constraint. Based on the above computational procedure, we can obtain a fast and efficient self-attention layer. However, according to ***Fact 2***, we need to ensure that $\boldsymbol{\alpha}_U$ is column orthogonal, $\boldsymbol{\alpha}_V$ is row orthogonal, and $\boldsymbol{A}^{i'}$ is diagonal. Considering that directly constraining $\boldsymbol{\alpha}_U, \boldsymbol{\alpha}_V, \boldsymbol{A}^{i'}$ to orthogonal or diagonal matrices may restrict the capabilities of neural networks, we use the idea of regularization to implement a soft constraint. Specifically, we introduce the following two loss items to our methods:

$$L_{\mathrm{orthogonal}} = \frac{1}{r^2}||(\boldsymbol{\alpha}_U^{\mathrm{T}}\boldsymbol{\alpha}_U) \circ (\boldsymbol{1} - \boldsymbol{I})||_F^2 + \frac{1}{r^2}||(\boldsymbol{\alpha}_V\boldsymbol{\alpha}_V^{\mathrm{T}}) \circ (\boldsymbol{1} - \boldsymbol{I})||_F^2 \tag{15}$$

$$L_{\mathrm{diagonal}} = \frac{1}{r^2}||\boldsymbol{A}^{i'} \circ (\boldsymbol{1} - \boldsymbol{I})||_F^2 \tag{16}$$

where r is the size of $\boldsymbol{A}^{i'}$, $\boldsymbol{1}$ is a $r \times r$ matrix with all elements one, $\boldsymbol{I}$ is a $r \times r$ identity matrix, $\circ$ is the element-wise product, $||\cdot||_F^2$ is the square of the Frobenius norm which is equal to the sum of the squares of the elements in the matrix. The final loss function is defined as:

$$L = L_{\mathrm{task}} + \gamma_1 L_{\mathrm{orthogonal}} + \gamma_2 L_{\mathrm{diagonal}} \tag{17}$$

where γ_1, γ_2 are the scaling factors of $L_{orthogonal}, L_{diagonal}$ to adjust the strength of the constraint, L_{task} is the original loss term of a specific task.

3.2 A High-Level Perspective of Singularformer

In brief, Singularformer rewrites formulas 1 and 2 of the standard Transformer as formulas 9 and 8, and they can be summarized as:

$$\boldsymbol{H}^i = \boldsymbol{\alpha} \times \mathrm{softmax}(\hat{\boldsymbol{\alpha}}\boldsymbol{Q}^i(\hat{\boldsymbol{\alpha}}\boldsymbol{K}^i)^{\mathrm{T}}) \times \hat{\boldsymbol{\alpha}}\boldsymbol{V}^i \tag{18}$$

where $\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{X}\boldsymbol{W}_a + \boldsymbol{b}_a) \in \mathbb{R}^{n\times r}$, $\hat{\boldsymbol{\alpha}} = \mathrm{softmax}((\boldsymbol{X}\boldsymbol{W}_a + \boldsymbol{b}_a)^{\mathrm{T}}) \in \mathbb{R}^{r\times n}$. From a mathematical perspective, $\boldsymbol{W}_a$ decomposes $\boldsymbol{X}$ in the second token dimension space to obtain $\boldsymbol{\alpha}$ and $\hat{\boldsymbol{\alpha}}$, while $\hat{\boldsymbol{\alpha}}$ decomposes $\boldsymbol{Q}^i, \boldsymbol{K}^i, \boldsymbol{V}^i$ in their first token dimension space, and $\boldsymbol{\alpha}$ restores the change of first token dimension space. From an intuitive perspective, $\hat{\boldsymbol{\alpha}}$ tends to aggregate the n query vectors of $\boldsymbol{Q}^i$, key vectors of $\boldsymbol{K}^i$, value vectors of $\boldsymbol{V}^i$ into r pseudo query, key, value vectors. The pseudo vectors can be seen as some high-level representations of the original vectors $\boldsymbol{Q}^i, \boldsymbol{K}^i, \boldsymbol{V}^i$. After message passing between the pseudo vectors, $\boldsymbol{\alpha}$ is trying to learn the inverse of the aggregation. In addition, the orthogonal constraint also has its intuitive meaning. The orthogonal constraint tries to orthogonalize $\hat{\boldsymbol{\alpha}}$ along its row dimension, and each row of $\hat{\boldsymbol{\alpha}}$ corresponds to an aggregation mode for the original vectors. Therefore, the orthogonal constraint can

[1] We tried the model without sharing the four linear layers but didn't get ideal results. In BBC dataset, there is only a 0.03% improvement in performance but a 117.6% increase in model scale and a 37.3% decrease in inference speed.

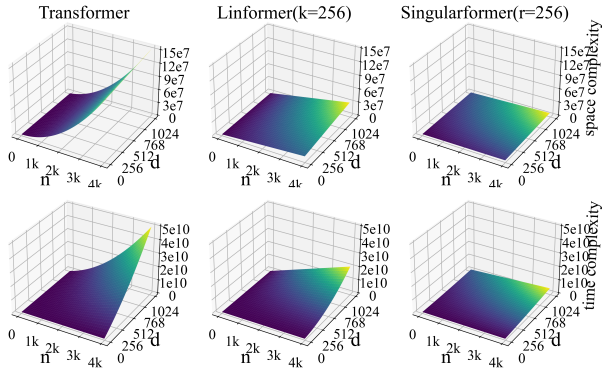

Figure 4: Visualization of the theoretical space-time complexity.

Methods	space complexity	time complexity
Transformer	$mn^2 + 5dn$	$2dn^2 + 4d^2n$
Linformer	$(5d + 2k + mk)n + 2kd$	$(4d^2 + 4kd)n$
Singularformer	$(d + r)n + 6rd + mr^2$	$3rdn + 4rd^2 + 2r^2d$

Table 1: The results of space-time complexity.

drive the model capture differentiated aggregation modes during training process, which reduces redundancy and obtains more powerful performance.

3.3 Analysis of Space-Time Complexity

To analyze the space-time complextiy of Singularformer, we define the number of multiplications as the metric of time complexity, and the number of elements in the matrix produced during the computing steps as the metric of space complexity. We compute the complexity results of the attention layer in standard Transformer, Linformer and Singularformer, as shown in Table 1. In order to better illustrate the comparison between them, we visualize the results in Figure 4. m is set to 8. k of Linformer is set to 256, which is the default value in their paper [Wang *et al.*, 2020]. r of Singularformer is set to 256, which is an adequate number for $\frac{d}{m}$. We can see that when n is extremely large, Linformer has obvious advantages compared to the standard Transformer. However, when d is extremely large and n is extremely small (the case of short sequences), Linformer's complexity may even be more than that of the standard Transformer. By contrast, Singularformer performs better in the scenario with small n and large d. Moreover, with the increase of n and d, it will have more advantages than the standard Transformer and Linformer.

4 Experiments

4.1 Datasets and Evaluation Metrics

In order to demonstrate the effectiveness of Singularformer, we choose R8 [Debole and Sebastiani, 2005], BBC [Greene and Cunningham, 2006], 20NG [Lang, 1995], IMDb [Maas *et al.*, 2011], MIMIC-III 50 [Johnson *et al.*, 2016], SCOPe95/40 [Fox *et al.*, 2014], CIFAR 100 [Krizhevsky and Hinton, 2009] datasets from different fields to conduct the experiments. Among them, R8, BBC, 20NG are three news topic classification datasets. IMDb collects a lot of movie reviews, which are divided into positive and negative reviews. MIMIC-III 50 is a multi-label long document classification dataset about medical coding. SCOPe95 and SCOPe40 are two datasets for protein folding prediction. The SCOPe40 means the proteins are less than 40% identity to each other, while the SCOPe95 means that the proteins share less than 95% identity to each other. CIFAR 100 is a dataset from the computer vision field. It includes lots of 32×32 RGB images from 100 categories. We take each 2×2 pixel in the image as a token in sequence. By this way, each image can be seen as a sequence with $\frac{32}{2} \times \frac{32}{2} = 256$ tokens. To evaluate model performance on the 8 datasets, we select accuracy (ACC) as the evaluation metric, which is defined as the proportion of samples that are correctly predicted.

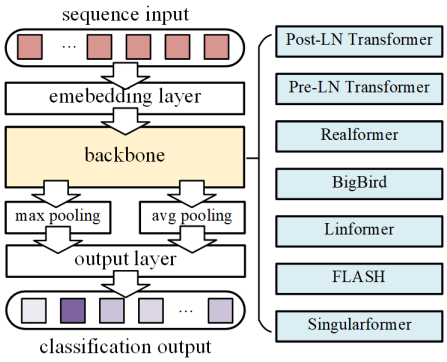

Figure 5: Model framework. Firstly, the input sequence is embedded into a sequence of vectors by the embedding layer. Then we feed the embedded sequence to the backbone to capture semantic features of the sequence. After that, the max pooling and the avg pooling are used to summarize the obtained semantic features. Finally, the summarized features are concatenated and fed into the output layer to get the final classification output. The output layer is composed by two dense layers with a ReLU activation between them.

Hyper-parameters	Values	Hyper-parameters	Values
embedding size	384	scaling factor of $L_{orthogonal}$	0.01
number of layers	6	scaling factor of $L_{diagonal}$	0.01
hidden size	384	batch size	64
number of heads	6	learning rate	0.0001
dropout in embedding layer	0.2	epochs	256
dropout in hidden layers	0.1	warm up epochs	4

Table 2: Hyper-parameter settings of experiments.

Methods	Sequence length thresholds of	
	memory consumption	inference speed
BigBird	704	848
Linformer($k = 256$)	448	368
FLASH(g=256)	432	400
Singularformer	**192**	**144**

Table 3: Sequence length thresholds of the linear Transformer.

4.2 Baselines and Implementation Details

To demonstrate the performance of Singularformer, we choose Post-LN Transformer [Vaswani *et al.*, 2017], Pre-

Methods	R8	BBC	20NG	IMDb	MIMIC-III 50	SCOPe95	SCOPe40	CIFAR 100
quadratic Transformers								
Post-LN Transformer	**0.97963±0.003**	0.97545±0.006	0.87374±0.001	0.87750±0.002	**0.42655±0.012**	0.66923±0.004	0.29350±0.005	**0.53128±0.009**
Pre-LN Transformer	0.97844±0.003	**0.97635±0.003**	0.86660±0.003	0.87627±0.005	0.40886±0.000	**0.68865±0.004**	**0.31334±0.006**	0.52128±0.008
Realformer	0.97899±0.003	**0.97635±0.003**	**0.87642±0.003**	**0.88538±0.001**	0.42376±0.009	0.64721±0.003	0.25889±0.006	0.52070±0.008
linear Transformers								
BigBird	0.97917±0.003	0.97665±0.004	0.86155±0.004	0.87618±0.003	0.37672±0.008	0.66864±0.004	0.28802±0.006	0.46790±0.015
Linformer($k = 256$)	0.97953±0.001	0.96916±0.005	0.86803±0.002	0.87637±0.005	0.40737±0.006	0.56476±0.003	0.23275±0.003	0.43428±0.007
FLASH($g = 256$)	0.97798±0.002	**0.97814±0.003**	0.86880±0.003	0.87731±0.005	0.41524±0.003	**0.67562±0.006**	0.30809±0.003	0.35194±0.007
Singularformer	**0.98145±0.001**	0.97754±0.003	**0.87756±0.003**	**0.88665±0.005**	**0.43942±0.005**	0.65674±0.005	**0.30886±0.006**	**0.52230±0.009**

Table 4: Comparison of Singularformer and other competing baselines on the test set.

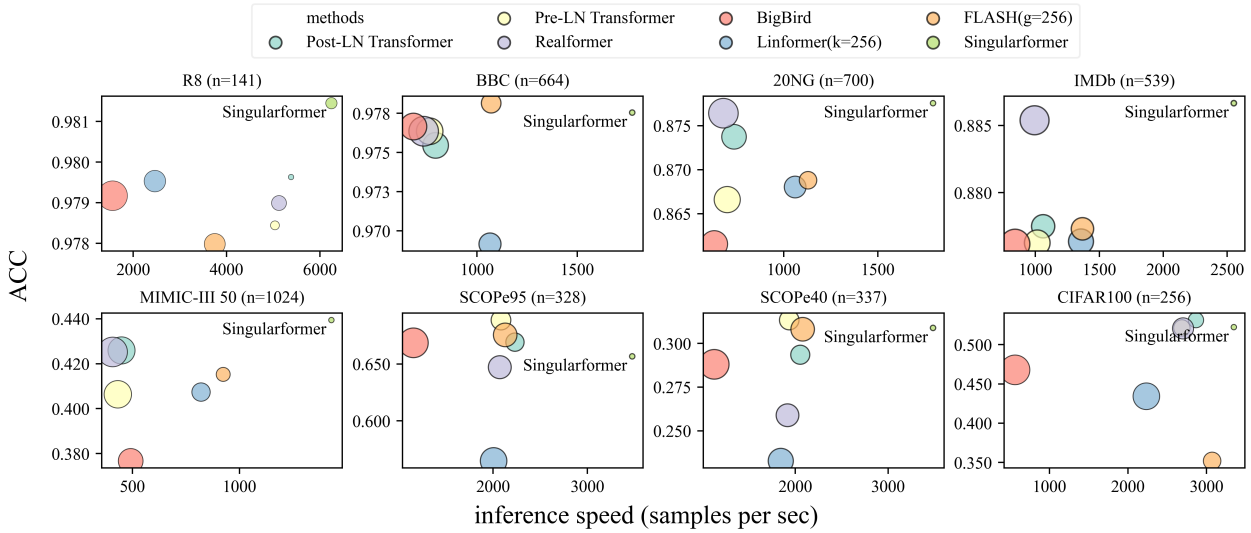

Figure 6: Accuracy (vertical axis), inference speed (horizontal axis), and GPU memory consumption (size of the circles) of different models for each dataset.

LN Transformer [Xiong *et al.*, 2020], Realformer [He *et al.*, 2021], BigBird [Zaheer *et al.*, 2020], Linformer [Wang *et al.*, 2020], FLASH [Hua *et al.*, 2022] as baselines. By referring to their original papers, the projection dimension k of Linformer is set to 256, the chunk size g of FLASH is set to 256.

In the experiments, the maximum sequence length is set to 141, 664, 700, 539, 1024, 328, 337, 256 on R8, BBC, 20NG, IMDb, MIMIC-III 50, SCOPe95, SCOPe40, CIFAR 100, respectively. To ensure the fairness of experimental comparison, we use the same model framework as shown in Figure 5 but substitute the backbone with different Transformer variants. We also ensure that the other model or training parameters are the same, which are shown in table 2. We train these models by AdamW optimizer [Loshchilov and Hutter, 2018], and the automatic mixed-precision training strategy [Micikevicius *et al.*, 2018] is used to save memory. Moreover, to ensure the reliability of the experimental results, all random seeds are fixed, the weight parameters of all models are initialized with a truncated normal distribution ($std = 0.02$), the bias parameters are initialized with zeros. We conduct 5-fold cross-validation in all datasets, that is, the training set is randomly divided into 5 parts, and one of them is selected as the validation set at each time to retain the model with the optimal number of training epochs. Then the retained models make predictions on test set and obtain the ACCs.

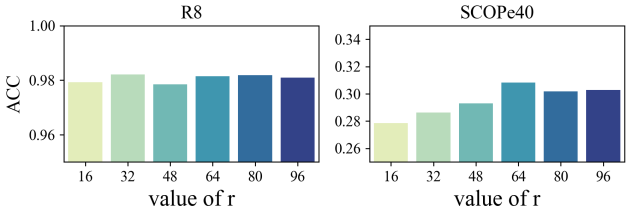

Figure 7: Performance of using different r on R8 and SCOPe40 datasets.

4.3 Results

Based on our experimental setting, the results of all models on all datasets are shown in Table 4, and the comparisons of ACC, speed, memory consumption between the methods are shown as Figure 6. As we can see, Singularformer not only has very low memory consumption, but also achieves relatively good and stable performance on all datasets. Singularformer even outperforms the quadratic Transformers (i.e. Post-LN Transformer, Pre-LN Transformer and Realformer whose complexity is quadratic dependent on n) on R8, BBC, 20NG, IMDb, MIMIC-III 50 datasets. For the linear Transformers (i.e. BigBird, Linformer, FLASH and Singularformer), Singularformer achieves the best performance in 6

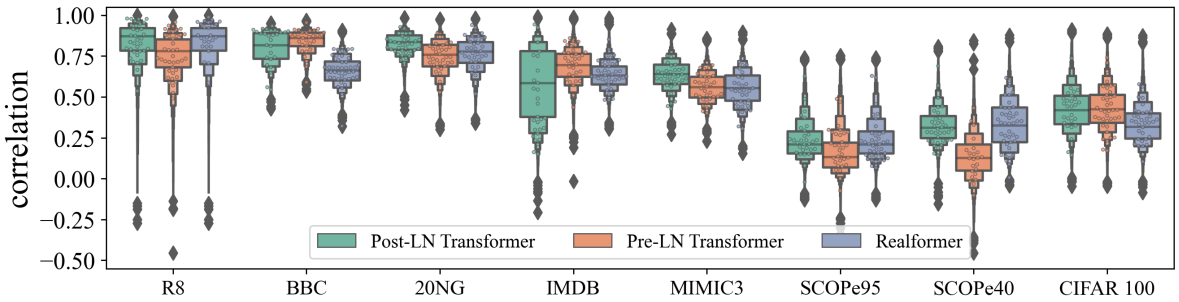

Figure 8: The distribution of attention correlation between the quadratic Transformers and Singularformer.

out of the 8 real-world datasets. Although FLASH performs better in BBC and SCOPe95 datasets, it needs more GPU memory consumption and infers slower than Singularformer. In addition, we find that the ACCs of Linformer and FLASH on some datasets (SCOPe40, SCOPe95, CIFAR 100) is much lower than other methods. We speculate that this is caused by the fixed projection of Linformer and the design of single attention head in FLASH. BigBird has the highest memory consumption and the slowest inference speed.

In addition, we compute the sequence length thresholds at which these linear Transformers outperform the standard Transformer (Post-LN Transformer) in terms of GPU memory consumption and inference speed. The results are shown in Table 3. When the sequence length exceeds 144, Singularformer has a faster inference speed than the standard Transformer. When the sequence length rises to 192, the space consumption advantage of Singularformer will become apparent. For other linear variants, the sequence lengths generally need to exceed 400 to achieve better complexity than the standard transformer. The results demonstrate the significant advantages of Singularformer for handling short sequences.

5 Discussion

5.1 The Effect of Using Different r

To verify whether the value $\frac{d}{m}$ of r is optimal, we analyze the performance of using different r for R8 and SCOPe40 datasets, as shown in Figure 7. It can be seen that the difference of using different r on R8 dataset is significantly smaller than that of SCOPe40 dataset. This is in accordance with what we expected, because R8 is a relatively simple classification dataset, which means that $\boldsymbol{Q}^i$ and $\boldsymbol{K}^i$ do not require particularly complicated expressions, that is, they are not full rank, and thus the change of r has little impacts on the performance. In contrast, SCOPe40 is a very complex classification task, and requires a far more complex expression of $\boldsymbol{Q}^i$ and $\boldsymbol{K}^i$ to guarantee classification performance. When $r \leq \frac{d}{m}$, increasing r can improve the expression ability of $\boldsymbol{Q}^i$ and $\boldsymbol{K}^i$. When r reaches $\frac{d}{m}$, the effect is optimal, further increasing r will not bring a significant improvement, and even the redundant information will interfere with the model learning. In any cases, setting r to $\frac{d}{m}$ in Singularformer is very appropriate in terms of both consumption and performance.

5.2 The Effectiveness of Learning to Decompose Self-Attention

In this study, we employ neural networks to learn the decompression of $\boldsymbol{A}^i$ to accelerate the standard Transformer. In order to verify the effectiveness of learning to decompose $\boldsymbol{A}^i$ into $\boldsymbol{\alpha}_U \boldsymbol{A}^{i'} \boldsymbol{\alpha}_V^{\mathrm{T}}$, we extract the attention matrices from Singularformer and the quadratic Transformers and compare their distributions. We use $\boldsymbol{A}^{i,l} \in \mathbb{R}^{n \times n}$ to represent the attention matrix from layer l, head i. For layer l, $\boldsymbol{A}^{1,l}, \boldsymbol{A}^{2,l}, ..., \boldsymbol{A}^{m,l}$ are aggregated into $\boldsymbol{A}^{*,l}$ by max pooling. Then, we summarize $\boldsymbol{A}^{*,1}, \boldsymbol{A}^{*,2}, ..., \boldsymbol{A}^{*,L}$ into $\boldsymbol{A}^{*,*}$ by average operation to obtain the final attention matrix:

$$\boldsymbol{A}^{*,l} = \mathrm{pooling}_{\max}(\boldsymbol{A}^{1,l}, \boldsymbol{A}^{2,l} ..., \boldsymbol{A}^{m,l}) \tag{19}$$

$$\boldsymbol{A}^{*,*} = \frac{1}{L} \sum_{l=1}^{L} \boldsymbol{A}^{*,l} \tag{20}$$

For Singularformer, we can use ${\boldsymbol{\alpha}_U}^l {\boldsymbol{A}^{i,l}}' {\boldsymbol{\alpha}_V^{\mathrm{T}}}^l$ to approximate $\boldsymbol{A}^{i,l}$ and compute $\boldsymbol{A}^{*,*'}$ in the same way. To measure the distribution difference between the attention matrices $\boldsymbol{A}^{*,*}$ and $\boldsymbol{A}^{*,*'}$, we calculate the Pearson correlation coefficient between the i-th row of the two matrices separately, and average the results of all rows to obtain the attention correlation the two matrices. Distributions of the correlation on the test set of all datasets are plotted in Figure 8. As we can see, the correlation between Singularformer and the quadratic Transformers is generally positive (correlation > 0), which also proves the effectiveness of Singularformer for decomposing the self-attention. Among them, the correlation on R8, BBC, 20NG, IMDB and MIMIC3 is generally more than 0.6, which achieves a strong positive correlation. The correlation on CIFAR100 is about 0.5, which is a moderate degree of correlation. However, the correlation between SCOPe95 and SCOPe40 is only about 0.2, which is a weak correlation. To figure out the reason for the weak correlation on SCOPe95 and SCOPe40, we further analyze the correlation between Post-LN Transformer, Pre-LN Transformer, Realformer, Singularformer and obtain the correlation heat map in Figure 9. We find that even the correlation between the quadratic Transformers only have a correlation between 0.2-0.3 on SCOPe95 and SCOPe40 datasets. We speculate that this is also due to the complexity of the datasets. When there isn't enough data for training, it is difficult for Transformers to capture consis-

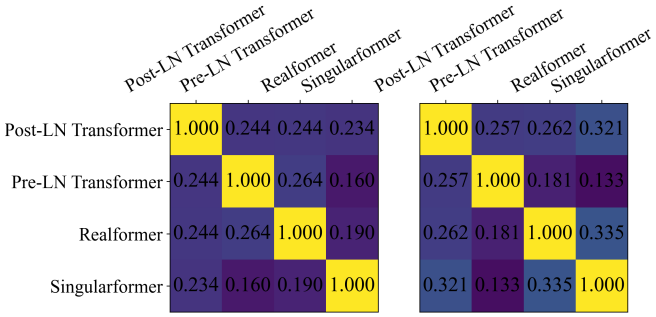

Figure 9: The attention correlation among Post-LN Transformer, Pre-LN Transformer, Realformer, Singularformer in SCOPe95 (left) and SCOPe40 (right) datasets.

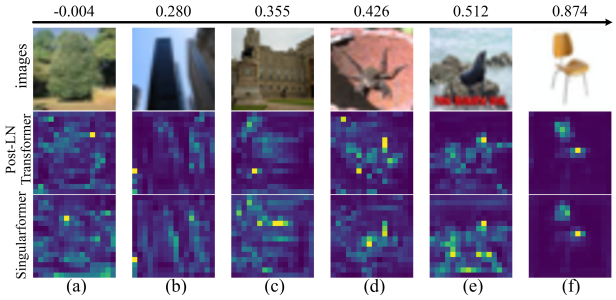

Figure 10: Attention visualizations of CIFAR 100's test set: (a)-(f) are cases with 0-quantile, 20-quantile, ..., 100-quantile correlations.

tent attention information. This explains why the weak correlation of Singularformer appears on these two datasets in Figure 8. In summary, Singularformer can learn highly similar attention information to the standard Transformers with greatly reduced space-time complexity, which also shows the effectiveness of decomposing $\boldsymbol{A}^i$ into $\boldsymbol{\alpha}_U \boldsymbol{A}^{i'} \boldsymbol{\alpha}_V^{\mathrm{T}}$. In addition, we present 6 attention visualizations (sorted by the correlation) from CIFAR 100's test set, as shown in Figure 10.

6 Conclusion

As a powerful neural network backbone, the standard Transformers have very strong feature extraction ability across multiple domains. However, the quadratic complexity greatly increase the training cost and limit the possibility of stacking design of the model. Therefore, based on the idea of SVD, we propose a novel linear-complexity Transformer called Singularformer. The experiments on 8 real-world datasets demonstrate that Singularformer has excellent performance and robustness under linear complexity. In the future, we will try to further explore the potential of Singularformer in the field of large-scale pre-training, multi-modal modeling, etc.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants (No. 61832019), and the Hunan Provincial Science and Technology Program (2019CB1007 and 2021RC4008), the Graduate Innovation Project of Central South University (No. 1053320213431), and the High Performance Computing Center of Central South University.

References

[Beltagy *et al.*, 2020] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[Child *et al.*, 2019] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[Choromanski *et al.*, 2020] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[Debole and Sebastiani, 2005] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and technology*, 56(6):584–596, 2005.

[Dosovitskiy *et al.*, 2020] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Fox *et al.*, 2014] Naomi K Fox, Steven E Brenner, and John-Marc Chandonia. Scope: Structural classification of proteins—extended, integrating scop and astral data and classification of new structures. *Nucleic acids research*, 42(D1):D304–D309, 2014.

[Gionis *et al.*, 1999] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[Greene and Cunningham, 2006] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press, 2006.

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

[Hua *et al.*, 2022] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International Conference on Machine Learning*, pages 9099–9117. PMLR, 2022.

[Johnson *et al.*, 2016] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[Jumper *et al.*, 2021] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[Katharopoulos *et al.*, 2020] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[Kenton and Toutanova, 2019] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[Kitaev *et al.*, 2020] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

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

[Liu *et al.*, 2021] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[Loshchilov and Hutter, 2018] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[Ma *et al.*, 2022] Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*, 2022.

[Maas *et al.*, 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[Micikevicius *et al.*, 2018] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.

[Shen *et al.*, 2021] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021.

[So *et al.*, 2021] David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Searching for efficient transformers for language modeling. *Advances in Neural Information Processing Systems*, 34:6010–6022, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wang *et al.*, 2020] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[Xiong *et al.*, 2020] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[Xiong *et al.*, 2021] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021.

[Zaheer *et al.*, 2020] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

[Zhang *et al.*, 2021] Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Qiang Zhang, Jiazhang Lian, and Huajun Chen. Ontoprotein: Protein pretraining with gene ontology embedding. In *International Conference on Learning Representations*, 2021.

[Zhu *et al.*, 2021] Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems*, 34:17723–17736, 2021.