Modify Self-Attention via Skeleton Decomposition for Effective Point Cloud Transformer

*Jiayi Han[1], *Longbin Zeng[1], Liang Du[1,2], Xiaoqing Ye[3], Weiyang Ding[1†], Jianfeng Feng[1†],

[1]Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, Shanghai, China, Key Laboratory of Computational Neuroscience and Brain-Inspired Intelligence (Fudan University), Ministry of Education, China, MOE Frontiers Center for Brain Science, Fudan University, Shanghai, China, Zhangjiang Fudan International Innovation Center.

[2]Interactive Entertainment Group, Tencent Inc., China.

[3]Baidu Inc., China.

dingwy@fudan.edu.cn, jffeng@fudan.edu.cn

Abstract

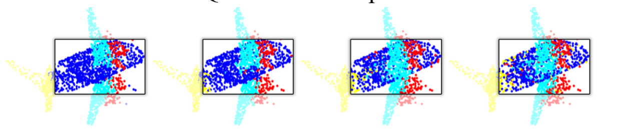
Although considerable progress has been achieved regarding the transformers in recent years, the large number of parameters, quadratic computational complexity, and memory cost conditioned on long sequences make the transformers hard to train and implement, especially in edge computing configurations. In this case, a dizzying number of works have sought to make improvements around computational and memory efficiency upon the original transformer architecture. Nevertheless, many of them restrict the context in the attention to seek a trade-off between cost and performance with prior knowledge of orderly stored data. It is imperative to dig deep into an efficient feature extractor for point clouds due to their irregularity and a large number of points. In this paper, we propose a novel skeleton decomposition-based self-attention (SD-SA) which has no sequence length limit and exhibits favorable scalability in long-sequence models. Due to the numerical low-rank nature of self-attention, we approximate it by the skeleton decomposition method while maintaining its effectiveness. At this point, we have shown that the proposed method works for the proposed approach on point cloud classification, segmentation, and detection tasks on the ModelNet40, ShapeNet, and KITTI datasets, respectively. Our approach significantly improves the efficiency of the point cloud transformer and exceeds other efficient transformers on point cloud tasks in terms of the speed at comparable performance.

Introduction

Transformer (Vaswani et al. 2017), a special network architecture based on self-attention mechanism, has attracted immense interest since it was originally proposed as a sequence-to-sequence model (Sutskever, Vinyals, and Le 2014) for machine translation. Transformer-based pretrained models (PTMs) (Qiu et al. 2020) have shown to be effective in fields such as language understanding, image processing, and point-cloud perception (Guo et al. 2021). The core of the transformer, namely, self-attention (SA), is used to re-describe embedding features with global relevance for each token. Compared with traditional deep learning approaches, SA has a global receptive field, which exceeds the scope of previous attention networks.

Transformers are particularly appropriate for point-cloud processing because SA is a set operator which does not assume any structural information over inputs. However, deploying transformerarchitecture is expensive, since the SA requires quadratic computational complexity and GPU memory with respect to sequence length (Wang et al. 2020b). For example, for the commonly used segmentation dataset ShapeNet (Yi et al. 2016), a common setting is sampling 2048 points and setting batch-size to 32. Simply storage of the self-attention and its corresponding gradient generated in a single layer take 1GB GPU memory, which makes the computation of the Transformer infeasible.

Qualitative Comparison



Quantitative Comparison

GT	Ours	Standard	Linformer
FPS	**10**	5	6
Mem (Mb)	**90**	560	100
mACC	**92.2**	92.0	90.0

Figure 1: Qualitative and quantitative (inference speed, memory, and mean accuracy) comparisons of our efficient transformer-based point cloud segmentation method and previous cutting-edge methods. With the fastest inference speed, our method predicts more accurate and crisp results than other methods, especially in those boundary regions. Our approach is much faster than the baseline and Linformer on CPU, which demonstrates our approach suits the edge computing configurations better.

In point cloud processing, in addition to the nature of a large number of points, another significant characteristic is the irregularity of point cloud physical storage. Different from sentences, points are stored in an irregular way,

*These authors contributed equally.

†indicates corresponding authors.

therefore two points can not be judged to be related according to their address adjacency in random access memory (RAM), which is a common assumption in NLP tasks. Therefore, some recent popular efficient transformers conditioned on prior knowledge of data structure such as Longformer(Beltagy, Peters, and Cohan 2020) are not applicable in point cloud tasks. To overcome this issue, YOGO (Xu et al. 2021) samples a subset of points as centroids and replace the point-point attention with point-centroid attention. However, a specialized architecture is needed in YOGO, and it is hard to be directly implemented in other models.

The above characteristic motivates us to modify the self-attention architecture, with the aim of making them simpler to implement while keeping its performance comparable to that of standard SA on point cloud tasks. In this work, we propose to boost the speed of SA by skeleton decomposition, namely SD boosting, to speed up SA. SA with SD boosting is called SD-based SA (SD-SA) in the following sections.

Because self-attention is low-rank, we adapt the Skeleton Decomposition (SD) (Goreinov, Zamarashkin, and Tyrtyshnikov 1995) (also known as CUR factorization) to approximate self-attention by using some columns and some rows of the matrix to be approximated. Importantly, this reconstruction method avoids explicitly calculating and storing of $n \times n$ attention matrix – this is achieved by simply dividing the matrix into some sub-matrices. Experiments have shown that our proposed SD-based transformer, is an effective approximation of the standard SA.

We utilize the Simple Point Cloud Transformer (SPCT)(Guo et al. 2021) as our backbone network, and validate our approach on point cloud classification and segmentation tasks on the ModelNet40 and ShapeNet datasets, respectively. Extensive experiments demonstrate that SD-SA greatly improves the efficiency of SPCT, and achieves performance on both tasks comparable to that of the baseline model. We show qualitative and quantitative comparisons of the proposed approach, baseline, and Linformer(Wang et al. 2020b) in Fig.1. Our approach performs better than Linformer, with the fastest speed and the smallest memory cost on edge computing configurations without GPUs. Our contributions are summarized as follows:

- We propose an efficient way to explicitly approximate self-attention via skeleton decomposition which dramatically reduces the computation complexity and GPU memory for point cloud processing.
- For point cloud classification, segmentation, and detection tasks, we achieve the best performance on ModelNet40, ShapeNet, and KITTI datasets, respectively, compared with other efficient transformers, with similar GPU memory and speed.
- SD-SA can be easily implemented in other transformers.

Related Works

Efficient Transformers

Structure-aware efficient transformers. Some efficient transformers introduce prior knowledge of the data structure in the modification. Longformer (Beltagy, Peters, and Cohan 2020) calculates the inner-product for each token with their neighbors and gradually enlarges the receptive field via cascaded SA layers, as proposed in VGGs networks(Simonyan and Zisserman 2014). Nystromformer (Xiong et al. 2021) utilizes 1D mean-pooling to downsample the keys and queries, and calculates SA as:

$$O = s(\frac{Q(\hat{K})^T}{\sqrt{d}})s(\frac{\hat{Q}(\hat{K})^T}{\sqrt{d}})^{\dagger}s(\frac{\hat{Q}(K)^T}{\sqrt{d}})V, \quad (1)$$

in which Q, K, V and s denote query, key, value and softmax, respectively, and $\hat{\cdot}$ denotes 1D mean-pooling. Such efficient transformers assume that the data to are ordered, for example, as sentences and images. However, point clouds are unordered, therefore the aforementioned architectures are not applicable.

Structure-free efficient transformers. Linformer samples random matrices $E \in \mathbb{R}^{k\times N}$ and $F \in \mathbb{R}^{k\times N}$ in which $k \ll N$, and form the modified SA as follows:

$$O = s(\frac{Q(EK)^T}{\sqrt{d}})FV, \quad (2)$$

where d and N denote the numbers of channels and tokens, respectively. Although Linformer highly speeds up the standard SA considerably, since E and F are randomly sampled from a Gaussian distribution, the initial entry distribution of standard SA is modified, which degrades the SA performance. this phenomenon is further demonstrated in Sec. 6. In contrast, Deformer (Cao et al. 2020) separates all tokens into N groups, and processes these groups separately. Formally, the attention of Deformer can be written as follows:

$$O = \{s(X_{\Gamma_i}W_Q^i(X_{\Gamma_i}W_Q^i)^T)W_V^iX_{\Gamma_i}\}_i^N, \quad (3)$$

where Γ_i is the index of tokens in the i_{th} group. Deformer gathers the SA-coded groups and further utilizes a supervised loss function to minimize the gap between O and the output of standard SA. As another alternative, Reformer (Kitaev, Kaiser, and Levskaya 2020) is based on the assumption that the largest values dominate the softmax output; thus, it is not needed to store the inner-product of tokens situated far from each other. Reformer, therefore, finds the neighborhood of each token and calculates their inner-product and softmax function.

In this work, we propose the use of SD to directly decompose the SA matrix. The proposed approach is structure-free and remains simple yet effective in point-based point cloud classification, segmentation, and downstream tasks.

Point Cloud Processing

A basic point cloud processing is grouping them to voxels, and introduce 3D convolutions (Zhou and Tuzel 2018; Yan, Mao, and Li 2018; Du et al. 2020). Point-based point cloud processing has recently been exploited, mainly use shared 1×1 convolution layers to extract pointwise features, and group the neighbouring points, as in PointNet and PointNet++ (Qi et al. 2017; Hao and Guibas 2017; Zhao et al. 2019; Yan et al. 2020). However, since the points are nonstructural, it is time-consuming to query the neighbors for local embedding and hierarchically downsample the points.

Hence, many works have introduced SA-based approaches to model these points (Guo et al. 2021; Zhao et al. 2021). Nevertheless, although SA methods have great advantages in modeling element-wise relationships such as tokens, it is a challenge to compute and store the SA matrix $s(QK^T/\sqrt{d})$ and its gradient: simply storing the attention map generated in a single SA layer with 1024 points takes 4Mb of GPU memory, without considering the batch size and gradient, thus it is difficult to train and implement the SA mechanism. To reduce the cost of SA, you only group once (YOGO) (Xu et al. 2021) was proposed to sample several clusters of points and replace the point-point attention with point-cluster attention.

Method

In this section, we review the basic idea of SA and introduce skeleton approximation for a general low-rank matrix with error estimation, discussing how to adapt this method to SA, and summarizing our proposed SD-SA framework.

What Is Self-Attention

Self attention (SA) serves as the backbone of the transformer and enables the model to jointly process information from different locations in different representation subspaces. Formally, an input sequence $X \in \mathbb{R}^{n \times d_x}$ comprising n tokens with d_x feature dimensions is embedded as three matrices Q, K and V by multiplying the learnable weights $W^q \in \mathbb{R}^{d_x \times d_q}$, $W^k \in \mathbb{R}^{d_x \times d_k}$, and $W^v \in \mathbb{R}^{d_x \times d_v}$ respectively, which can be interpreted as *queries* , *keys* and *values*.The regular dot-product attention mechanism modifies the embedded representation by introducing non-local information. So the output matrix is defined as follows:

$$\begin{cases} Q = XW^q, K = XW^k, V = XW^v \\ O = s(\dfrac{QK^T}{\sqrt{d}})V. \end{cases} \tag{4}$$

We modify the formulation to the following form:

$$\begin{cases} A = \exp\left(QK^T/\sqrt{d}\right) \\ D = \operatorname{diag}\left(A\mathbf{1}_L\right) \\ \mathrm{O} = D^{-1}AV \end{cases}, \tag{5}$$

where $\exp(\cdot)$ is an element-wise function, $d = d_q = d_k$ represents the dimension of the projected space, and D is a diagonal matrix in which the diagonal elements denote the sum of each row of A.

SA computations are expensive. SA needs to store n^2 similarity scores for different token pairs, and each one requires d multiplications in eq. (5), leading to time and space computing complexities of $O(n^2 d)$ and $O(n^2 + nd)$ respectively. Due to the quadratic dependency on the sequence length, in principle, dot-product attention of type (5) is inapplicable for long sequences (e.g., $n > 1000$).

SA is low rank. We first provide an intuitive representation of the SA matrix A. During the training process of standard SA, as in Sec. 6, we freeze and apply singular value decomposition (SVD) to the attention matrix across different epochs and plot the singular value distribution, in order to investigate the rank of the matrix.

Fig. 4 illustrates the low rank of the attention matrix, from which we observe that the singular values decrease exponentially and most of the mass is centred on the largest 100 values. Moreover, (Wang et al. 2020b) demonstrated that the attention matrix is low rank with a high probability, which is formally illustrated as follows:

$$\Pr\left(\left\|\hat{A}w^T - Aw^T\right\| < \epsilon \left\|Aw^T\right\|\right) > 1 - o(1), \tag{6}$$

which holds for any w, especially $w = e_i$, which means that each column of A can be approximated by each column of $\hat{A}$, where $\operatorname{rank}(\hat{A}) = \Theta(\log(n))$. Hence, we can conclude that the attention matrix A is low rank. $o(1)$ is the Higher order infinitesimal of 1.

Skeleton Decomposition for Matrix Approximation

Matrix decomposition, a popular method for approximating low-rank matrices, involves the factorization of a matrix into a product of some matrices. The major advantage of matrix decomposition is that the constituent parts can be stored and manipulated more economically than the matrix itself. Below, we describe the celebrated Skeleton decomposition which has been studied extensively, yielding many valid theories, and has been applied to many aspects of machine learning (Kuleshov, Chaganty, and Liang 2015).

SD-based estimation is similar to truncated SVD (tSVD) based estimation in that the rank-r SD of A is used to reconstruct the original data matrix. For the rest of this paper, we adopt the following notation: given $A \in \mathbb{R}^{m \times n}$, A_{Γ_r} denotes the restriction of A to rows indexed by Γ_r , and A^{Γ_c} denotes the restriction of A to columns indexed by Γ_c . The SD of A is a factorization of the form:

$$A \sim \underbrace{\begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}}_{=:A^{\Gamma_c}} \underbrace{A_{11}^{-1}}_{=:U} \underbrace{\left[A_{11} A_{12}\right]}_{=A_{\Gamma_r}}. \tag{7}$$

The search of Γ_c and Γ_r is known as a special case of the index subset selection problem (ISSP). We are interested in the accuracy estimates for the best or nearly the best rank k approximations of the matrix A by its skeleton approximation $A^{\Gamma_c} U A_{\Gamma_r}$ with some proper sampling index Γ_c and Γ_r (Goreinov, Zamarashkin, and Tyrtyshnikov 1995).

The first systematic error estimation of such a matrix approximation was conducted by Goreinov and Tyrtyshnikov (Goreinov, Zamarashkin, and Tyrtyshnikov 1995)(Goreinov, Tyrtyshnikov, and Zamarashkin 1997). They considered a sub-optimal sampling technique consisting of finding Γ_c and Γ_r to ensure that $A_{\Gamma_r}^{\Gamma_c}$ has the maximal volume, i.e., the maximal absolute determinant among all $k \times k$ submatrices of A.

Theorem 1 ((Goreinov, Zamarashkin, and Tyrtyshnikov 1995)). *Consider $A \in \mathbb{R}^{m \times n}$ and row and column indices Γ_r and Γ_c respectively, with $|\Gamma_c| = |\Gamma_r| = k$. Define $G = A_{\Gamma_r}^{\Gamma_c} \in \mathbb{R}^{k \times k}$. If G is non-singular and has maximal volume among all $k \times k$ submatrices of A, then*

$$\|A - CUR\|_{\max} \leq (1 + k)\sigma_{k+1}, \tag{8}$$

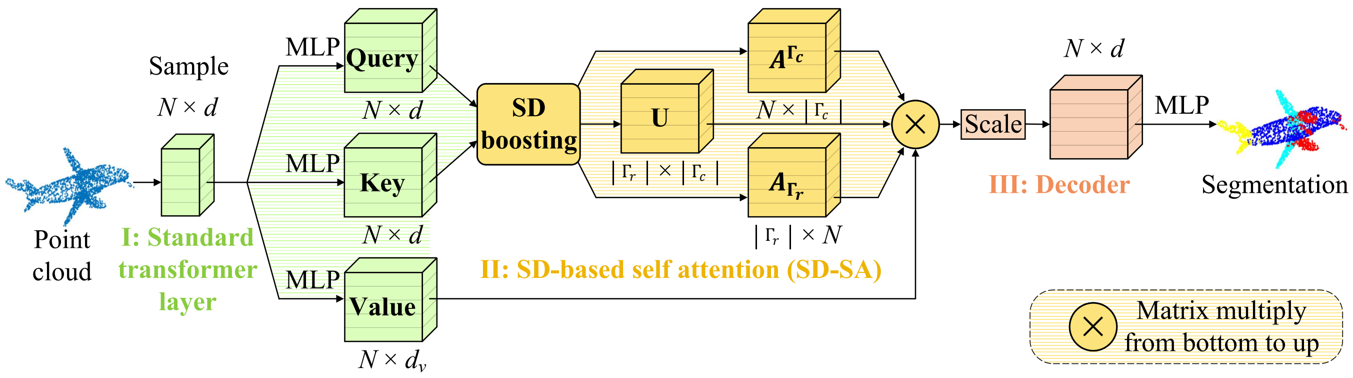


Figure 2: Schematic diagram of SD-based transformer. The framework contains three parts: (I) standard transformer layer colored in green; (II) the proposed SD-SA colored in golden; (III) the transformer decoder colored in red. To speed up SA, we decompose SA into smaller metrices A^{Γ_c}, A_{Γ_r} and U by SD boosting, whose details are in Fig.3. We replace SA by SD-SA of point cloud transformer.

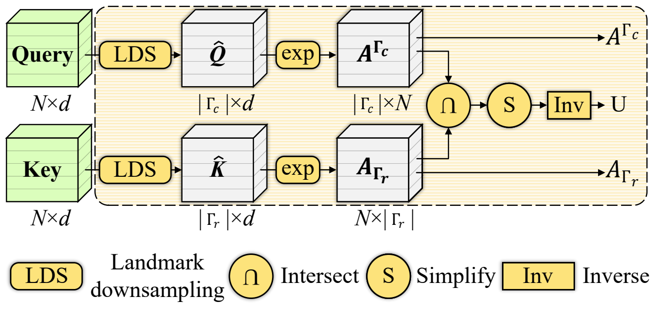


Figure 3: Details of SD boosting. Points are coded as Q, K and V. Via landmark selection (LDS), subsets of Q and K are sampled to $\hat{Q}$ and $\hat{K}$, respectively. We then achieve A^{Γ_c} and A_{Γ_r}, where Γ_c and Γ_r represent the indices of the selected landmarks (points). The pseudo-inverse of intersection of A^{Γ_c} and A_{Γ_r} is denoted as U. The sum of each row of self-attention is calculated as in Equ. 5, and denoted as D. The final encoded points is calculated as $D^{-1}A^{\Gamma_c}UA_{\Gamma_r}V$, in which $U = P^{-1}$ represents the intersection inverse.

where $C = A^{\Gamma_c}, R = A_{\Gamma_r}$, $U = G^{-1}$ *and* σ_{k+1} *is the* $(k+1)th$ *singular value of* A.

Under this ISSP situation, the sampling index defined in Theorem 1 is nearly optimal. Note that if $rank(A) = k$, then its skeleton approximation is exact, $A = CUR$.

Skeleton Decomposition-Based Self-Attention

Consider the SA case in which $A's$ entries can be specified by the function exp in closed form as mentioned above; then, the SD is fully described by $U \in \mathbb{R}^{k\times k}$, as well as the two index sets Γ_c and Γ_r. In this case, a rank-k approximation of A takes up $O(m+n)k$ space, such as the SVD approximation, but the SD requires only $O(k^2)$ space.

Based on the low-rank characteristics of SA, we can set a small $k \le rank(A)$ to search row index Γ_r and column index Γ_c and then disassemble the dot-product attention matrix with landmark matrices, while ensuring a good approximation accuracy as in Theorem 1. The complete optimal SD algorithm consists of two parts: sampling the submatrix of the maximal volume among all $k \times k$ submatrices and then computing the inverse of the intersection matrix as U.

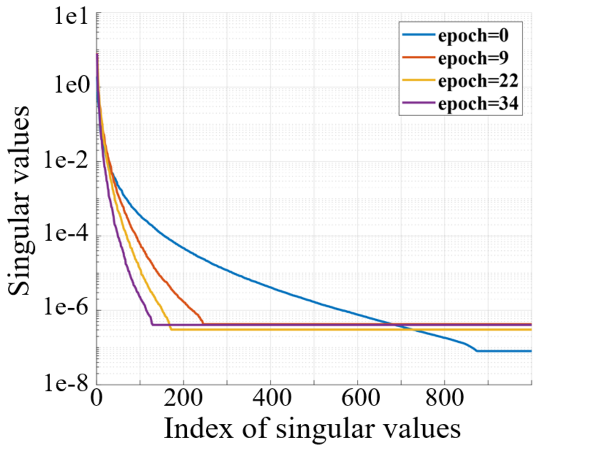


Figure 4: Singular value distribution of the SA matrix. The X-axis is the index of largest singular value and the Y-axis the singular value.

Fast index selection method. Although the deterministic index selection strategies of Theorem 1 provide a theoretical guarantee for the approximation accuracy, finding the submatrix of the maximal volume is NP-hard. Due to the computational complexity of these strategies, however, they do not apply to large matrices. This drawback inspired us to seek a simple but accurate alternative sampling method.

One natural idea of the ISSP from a matrix is to capture as much information as possible while accurately while approximating the original matrix (Thurau, Kersting, and Bauckhage 2012). That is, one computes an importance score for each column (or row) and samples the columns (or rows) according to their distribution of scores. There are many scoring criteria; the simplest but useful way is to compute the l_p-norm as the score.

Concerning the attention matrix $A \in \mathbb{R}^{n\times n}$ in Eq.(5), we can easily obtain A^{Γ_c} and A_{Γ_r} as follows:

$$\begin{cases} A^{\Gamma_c} = \exp(Q(K^T)^{\Gamma_c}) \\ A_{\Gamma_r} = \exp((Q)_{\Gamma_r} K^T). \end{cases} \tag{9}$$

Alternative inverse problem. After obtaining the landmark matrices C and R, we compute the inverse of the intersection matrix as $U = G^{-1}$. Here, we relax the assumption that G is non-singular to achieve the general form $U = G^{\dagger}$. However, the condition number of the intersection matrix is unusually large, as reflected by the very large singular value distribution range, which leads to extremely large calculation errors when calculating the generalized inverse.

There are two direct regularization methods for addressing this problem. A natural method is tSVD, which can be applied to the intersection matrix G to obtain an approximation by removing all the components with singular values less than a given tolerance; this is the general idea of regularization by de-singularing the matrix.

In the second approach, we are not concerned with the singular values but focus instead on the original values of the intersection matrix G. we observe that a gap between the entry value distribution of the involved matrix, and therefore construct a permuted diagonal matrix by removing small values in the original matrix. This means that we obtain a permutation-style matrix while preserving most of the information (i.e., we have retained the larger elements) of the original matrix. In addition, the transformer matrix easily computes the inverse as:

$$G \leftarrow P, \quad U = G^{-1} \leftarrow \{p^{\dagger}_{j,i}\}, \tag{10}$$

where $p^{\dagger}_{j,i}$ denotes the generalized inverse of matrix p at the coordinates j, i.

From the perspective of the element distribution, both the tSVD approximation and our proposed permuted diagonal matrix approximation have good approximation effects. However, we find that the computation of tSVD approximation is much more expensive than a permuted diagonal matrix. In the remainder of this article, we utilize the permuted diagonal matrix to substitute the intersection matrix.

Summarising of SD-SA. Hence, we briefly describe the procedure of the proposed SD-SA in Fig. 2 and Fig. 3. Given an input matrix $X_{n\times d}$, where n and d represent the numbers of tokens and channels, respectively. "Scale" represents multiplying D^{-1} onto AV. X is projected to Q, K, and V. Our approach of efficiently computing the SA is formally summarized in Alg. 1.

Experiment

Experimental Setup

Dataset. We adopt the two most commonly used benchmark datasets to evaluate our SD-SA: ShapeNet (Yi et al. 2016) and ModelNet40 (Wu et al. 2015), respectively. ShapeNet

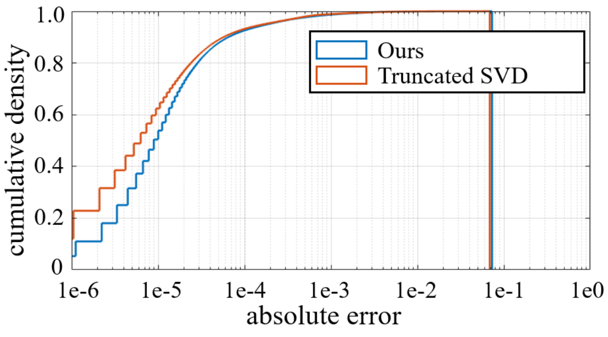


Figure 5: Cumulative density for absolute reconstruction error of SD-based SA, in which intersection is replaced with permutated diagonal matrix(blue) and tSVD approximation matrix(red).

Algorithm 1: SD-SA

```
Input : Query matrix $Q \in \mathbb{R}^{n\times d}$ , Key matrix
        $K \in \mathbb{R}^{d\times n}$, and user-defined parameters
        $|\Gamma| = l$;
Output: A skeleton representation for SA
1 independently sample $l$ columns according to the
    row score distribution of $K$ to form
    $A^{\Gamma_c} = \exp(Q(K^T)^{\Gamma_c})$;
2 independently sample $l$ columns according to the
    row score distribution of $Q$ to form
    $A_{\Gamma_r} = \exp((Q)_{\Gamma_r} K^T)$;
3 Replace the intersection matrix $G$ as permuted
    diagonal matrix and denote it as $P = p_{i,j\cdot}$ ;
4 Compute $U = P^{-1} = \{p^{\dagger}_{j,i}\}$ ;
5 Compute $D = \mathrm{diag}(A^{\Gamma_c} U A_{\Gamma_r} \mathbf{1}_L)$;
6 Output $D^{-1} A^{\Gamma_c} U A_{\Gamma_r}$
```

contains 16880 3D models, 14006 of which are used for training, and 2874 are used for testing. ModelNet40 contains 40 object categories and 12311 computer-aided design (CAD) models,the official division into 9843 training objects and 2468 testing objects is utilized. We also validate our approach on 3d object detection on KITTI dataset (Geiger et al. 2013). We utilize the official training and validation set in experiments.

Baseline. We utilize the SPCT framework proposed in (Guo et al. 2021) as the baseline approach. However, we do not implement the full version of PCT since PCT implements cascaded neighbor embedding layers, which greatly contributes to the computation time of the model.

Implementation details. In the subsequent experiments, we follow the settings in (X. 2019) for each task unless otherwise noted. We simply replace the initial SA with the proposed SD-SA in SPCT. $|\Gamma_c|$ and $|\Gamma_r|$ are set to 64; for convenience, we denote this by $|\Gamma|$ in the following sections. We use the Adam optimizer to train the model for 250 epochs on 3×NVIDIA GTX 1080 Ti (12 GB). For point cloud segmentation and object classification, the batch size is set to

Method	Publish	mIoU(%)	GPU Mem	time(ms)	Perf. gap	CPU time(s)
PointNet(Qi et al. 2017)	CVPR 2017	83.7	1.5GB	21.4	-	-
RSNet(Wang et al. 2020a)	TGRE 2020	84.9	0.8GB	73.8	-	-
PointNet++(Hao and Guibas 2017)	NIPS 2017	85.1	2.0GB	77.7	-	-
DGCNN(Wu et al. 2018)	NC 2017	85.1	2.4GB	86.7	-	-
YOGO(Xu et al. 2021)	Arxiv 2021	85.2	0.9GB	25.6	-	-
PointCNN(Li et al. 2018)	NIPS 2018	86.1	2.5GB	134.2	-	-
SPCT(baseline)†	CVM 2021	85.6	4.5GB	6.5	-	1.48
SPCT(Guo et al. 2021)	CVM 2021	**85.8**	4.5GB	-	-	-
Reformer(Kitaev et al. 2019)†	ICLR 2019	84.2	2.0GB	-	-1.6 (-1.4) %	-
Informer(Zhou et al. 2021)†	AAAI 2021	84.4	-	-	-1.4 (-1.2) %	-
Linformer(Wang et al. 2020b)†	Arxiv 2020	84.5	0.8GB	4.0	-1.1 (-0.9) %	1.30
Deformer(Cao et al. 2020)†	ACL 2020	84.8	0.8GB	4.0	-1.0 (-0.8) %	0.86
Ours ($\lvert\Gamma\rvert$=8)	-	85.3	**0.7GB**	**3.7**	-0.5 (-0.3) %	**0.79**
Ours ($\lvert\Gamma\rvert$=64)	-	85.5	0.8GB	5.0	-0.3 (-0.1) %	0.82
Ours (with PE,$\lvert\Gamma\rvert$=64)	-	**85.6**	0.8GB	5.1	**-0.2 (0.0)%**	0.83

Table 1: Performance of different models on ShapeNet. † means the result is under our implementation. Performance gap represents the gap between the efficient transformer and the baseline. "PE" demonstrates positional embedding. $|\Gamma|$ is the hyper-parameter as in Alg. 1. Performance gap is the gap of baseline and efficient transformers.

128 and 64, respectively. A gradient clip is utilized, and the maximum norm is set to 10.

Point Cloud Segmentation, Classification and Object Detection

We first validate the proposed method on ShapeNet. The experimental settings are mentioned in Sec. 6, and the results are shown in Table 1. The performance gaps inside the "()" represent the performance gap between the efficient transformers and the baseline method under our implementation, while the ones outside represent the gaps the ones compared with the baseline method reported officially. The proposed approach achieves a comparable performance compared with the baseline and requires much less memory, which shows that our approximation is effective yet simple. We also compare the proposed SD-SA with other efficient transformers by directly substituting them for SA. We set the corresponding hyper-parameter equal to $|\Gamma|$ for a fair comparison. The number of budgets in Reformer is set to 64. The result shows that the proposed SD-based transformer achieves higher performance than other efficient transformers thanks to the approximation precision. Since the intersection is sparse and also low-rank as shown Table 4, a small subset is capable of covering the information of SA.

We further validate the proposed SD-based transformer on the ModelNet40 dataset. The results are presented in Table 2, in which * means that we introduce a grouping layer as in (Qi et al. 2017). The results reveal that replacing SA with the proposed SD-SA causes a slight performance drop compared with the standard SA (91.8% vs 92.0%). Moreover, simply adding a grouping layer as in (Hao and Guibas 2017) improves the performance by 0.4%, and exceeds the baseline, which demonstrates that the proposed approach is capable of taking the advantage of the low-rank characteristic, and outperforms the other efficient transformers.

To validate SD-SA for 3d object detection, the experimental settings of LiDAR-RCNN (Li, Wang, and Wang 2021) are employed, and we simply replace the PointNet detector with an SD-based transformer. The results are presented in Table 3, revealing that the proposed SD-SA is also helpful for 3D object detection.

Method	Publish	mAcc(%)
PointNet	CVPR 2017	89.2
SpiderCNN	ECCV 2018	90.5
YOGO	Arxiv 2021	91.4
PointNet++	NIPS 2017	91.9
PointCNN	NIPS 2018	92.2
DGCNN	NC 2017	92.2
PointWeb	CVPR 2019	92.3
PointASNL	CVPR 2020	93.2
‡SPCT	CVM 2021	92.0
Reformer†	ICLR 2019	89.7
Linformer†	Arxiv 2020	90.0
Deformer†	ACL 2020	91.1
Ours	-	91.8
Ours*	-	**92.2**

Table 2: Performance of different methods on ModelNet40. † means the result is under our implementation. ‡ represents the baseline. * represents added grouping layer.

Ablation Study

Row selection approach. We validate random selection, norm-based selection (l_1-norm and l_2-norm), furthest point sampling (FPS), and use the mean of self-attention generated by randomly samples point for 3 three times ("Mean"), show the results in Table 4. FPS obtains the worst performance in the approaches, which may be because the sampled points are irrelevant to each other. "Mean" shows worse per-

Model	Easy	Moderate	Hard
baseline	86.8	79.0	70.4
SD-SA	86.7	78.7	70.2
Linformer	86.1	78.5	69.9
PointNet	86.3	78.0	69.2

Table 3: Detection performance of different stage-2 detector for LiDAR-RCNN on KITTI validation set on "Car".

formance than random sampling which may be because the attention matrix is over smoothed. Norm-based approaches have the best performance, as norm-based approaches are proved to have a limited approximation error.

Method	FPS	Mean	Random	l_2	l_1 (ours)
mIoU(%)	85.2	85.3	85.4	85.6	85.6
rank	5.38	-	4.25	4.18	4.17
cond	7e12	-	1e17	3e17	4e17

Table 4: Performance of proposed SD-based transformer on ShapeNet with different row selection methods.

Number of SD-SA	4	3	2	1	0
mIoU (%)	85.5	85.5	85.5	85.6	85.6

Table 5: Performance of proposed SD-based transformer on ShapeNet with different number of replaced standard self-attention layers.

Number of replaced self-attetion layers. To further figure out the influence of replacing standard self-attention by SD-SA, we gradually take the place of self-attention layer from 1 to four. The results are shown in Table 5. Replacement of self-attention has minor effect on the model, which demonstrates the precision of our approximation.

$\|\Gamma\|$	8	16	32	64
mIoU(%)	85.3	85.3	85.4	85.6
Acc(%)	91.3	91.4	91.6	91.8

Table 6: Performance of proposed SD-based transformer on ShapeNet (mIoU) with different number of selected indices.

Approach of calculating pseudo-inverse. As demonstrated in Table 4, the condition number of the intersection matrix is large and is therefore difficult to calculate its pseudo-inverse reasonably. We show the cumulative distribution of absolute error between SA and SD-SA with different inverse operations of the valid samples in Fig. 5. The proposed approach avoids the problem of large condition numbers and therefore achieves smaller errors.

Number of selected indexes. To validate the influence of different number of selected indexes, we set $|\Gamma| = 8, 16, 32, 64$. The results are shown in Table 6. When $|\Gamma| \geq 8$, the number of selected indexes has a minor effect on the performance of the proposed method. This result indicates that only a few main components could be effective enough to reconstruct the main information of SA.

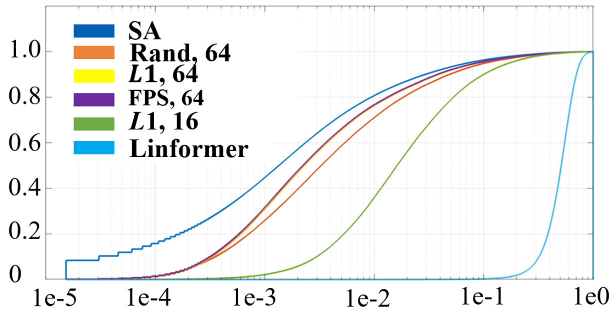


Figure 6: Normalized cumulative density function of self-attention matrix with different number of selected rows. X-axis represents the normalized weight and Y-axis represents the cumulative density. Numbers in legend represent $|\Gamma|$.

Statistics of the Proposed SD-SA

To validate the low-rank assumption, we visualize the singular value of the self-attention matrix in in Fig. 4. We show the rank and condition number of the I_{CR} in Table 4. The intersection matrix is low-rank and ill-conditioned, independent of the row selection strategy. Therefore, the solution of $I_{CR}I_{CR}^{\dagger} = E$ is non-robust in which E is identity matrix.

We draw the normalized distribution of SA, SD-SA and Linformer in Fig. 6. We also show the non-normalized absolute error of A and $\hat{A} = D^{-1}A^{\Gamma_c}UA_{\Gamma_r}$ in Appendix. The aforementioned result demonstrate that SA could be well approximated by SD-SA.

Conclusions

We propose a skeleton decomposition-based self-attention mechanism, named SD-SA. We decompose the self-attention matrix into multiple smaller matrices and validate that the multiplication of these matrices provides a good approximation of the standard self-attention while greatly reducing the computational complexity. We implement our approach on point cloud classification, segmentation, and object detection tasks and dramatically reduce the GPU memory while maintaining precision. The proposed SD-based transformer has a much smaller performance gap than the baseline compared to previous works, which demonstrates the effectiveness and efficiency of our SD-SA.

Acknowledgements

This work is supported by National Key R&D Program of China (No.2049YFA0709502, 2018YFC1312904), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), ZJ Lab, and Shanghai Center for Brain Science and Brain-Inspired Technology, the 111 Project (No.B18015).

References

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Cao, Q.; Trivedi, H.; Balasubramanian, A.; and Balasubramanian, N. 2020. DeFormer: Decomposing Pre-trained Transformers for Faster Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4487–4497.

Du, L.; Ye, X.; Tan, X.; Feng, J.; Xu, Z.; Ding, E.; and Wen, S. 2020. Associate-3Ddet: Perceptual-to-conceptual association for 3D point cloud object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13329–13338.

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32: 1231–1237.

Goreinov, S. A.; Tyrtyshnikov, E. E.; and Zamarashkin, N. L. 1997. A theory of pseudoskeleton approximations. *Linear algebra and its applications*, 261(1-3): 1–21.

Goreinov, S. A.; Zamarashkin, N. L.; and Tyrtyshnikov, E. E. 1995. Pseudo-skeleton approximations of matrices. In *Doklady Akademii Nauk*, volume 343, 151–152. Russian Academy of Sciences.

Guo, M.-H.; Cai, J.-X.; Liu, Z.-N.; Mu, T.-J.; Martin, R. R.; and Hu, S.-M. 2021. PCT: Point cloud transformer. *Computational Visual Media*, 7(2): 187–199.

Hao, C. R. Q. L. Y.; and Guibas, S. L. J. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv preprint arXiv:1706.02413*.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Kuleshov, V.; Chaganty, A.; and Liang, P. 2015. Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics*, 507–516. PMLR.

Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; and Chen, B. 2018. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31.

Li, Z.; Wang, F.; and Wang, N. 2021. Lidar r-cnn: An efficient and universal 3d object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7546–7555.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qiu, X.; Sun, T.; Xu, Y.; Shao, Y.; Dai, N.; and Huang, X. 2020. Pre-trained models for natural language processing: A survey. arXiv 2020. *arXiv preprint arXiv:2003.08271*.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Thurau, C.; Kersting, K.; and Bauckhage, C. 2012. Deterministic CUR for improved large-scale data analysis: An empirical study. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, 684–695. SIAM.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, J.; Zhong, Y.; Zheng, Z.; Ma, A.; and Zhang, L. 2020a. RSNet: The search for remote sensing deep neural networks in recognition tasks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(3): 2520–2534.

Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; and Ma, H. 2020b. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wu, B.; Liu, Y.; Lang, B.; and Huang, L. 2018. DGCNN: Disordered graph convolutional neural network based on the Gaussian mixture model. *Neurocomputing*, 321: 346–356.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

X., Y. 2019. Pointnet-Pointnet2-pytorch. Website. https://github.com/yanx27/Pointnet_Pointnet2_pytorch.

Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; and Singh, V. 2021. Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 14138–14148.

Xu, C.; Zhai, B.; Wu, B.; Li, T.; Zhan, W.; Vajda, P.; Keutzer, K.; and Tomizuka, M. 2021. You only group once: Efficient point-cloud processing with token representation and relation inference module. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4589–4596. IEEE.

Yan, X.; Zheng, C.; Li, Z.; Wang, S.; and Cui, S. 2020. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5589–5598.

Yan, Y.; Mao, Y.; and Li, B. 2018. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10): 3337.

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.-C.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; and Guibas, L. 2016. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6): 1–12.

Zhao, H.; Jiang, L.; Fu, C.-W.; and Jia, J. 2019. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5565–5573.

Zhao, H.; Jiang, L.; Jia, J.; Torr, P. H.; and Koltun, V. 2021. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 16259–16268.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11106–11115.

Zhou, Y.; and Tuzel, O. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4490–4499.